CS409- Introduction to Database Administration

Lesson 1: Defining data, information & database

What is Data and Information?

Data are the raw bits and pieces of information with no context.

Anything which is not necessarily meaningful to human being.

Raw facts/Un-processed information.

Data are the raw bits and pieces of information with no context. If I told you, "45, 32, 41, 75," you would not have learned anything.

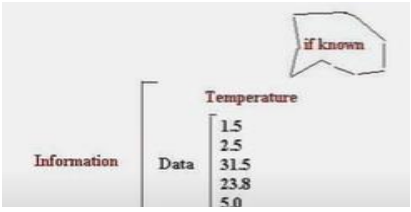
The numbers 10, 22, and 119 are data someone can think these are lengths and other may think room numbers. Similarly, 1.5, 2.5, 31.5, MAY represent lengths of some iron rods.

Data can be quantitative or qualitative. Quantitative data is numeric, the result of a measurement, count, or some other mathematical calculation. Qualitative data is descriptive. "Dark Brown" the color of hair, is an example of qualitative data. A number can be qualitative too: if I tell you my favorite number is 5, that is qualitative data because it is descriptive, not the result of a measurement or mathematical calculation.

Information

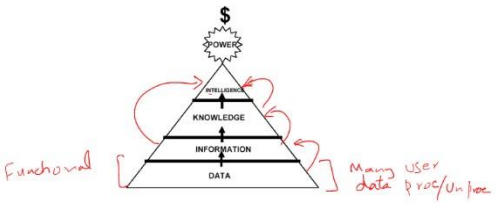
Data Processed to reveal its meaning
Information is meaningful
In today's world, accurate, relevant, and timely information is the key to good decision making good decision making is key to survival in today's competitive and global environment By itself, data is not that useful. To be useful, it needs to be given context.

Returning to the example above, if I told you that "45, 32, 41, 75" are the numbers of students that had registered for upcoming classes, that would be information. By adding the context – that the numbers represent the count of students registering for specific classes – I have converted data into information.

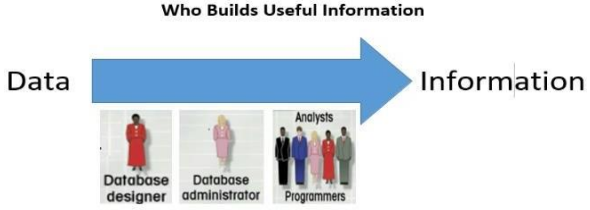


Aim of Data & Information

Once we have put our data into context, aggregated and analyzed it, we can use it to make decisions for our organization. We can say that this consumption of information produces knowledge. This knowledge can be used to make decisions, set policies, and even spark innovation.

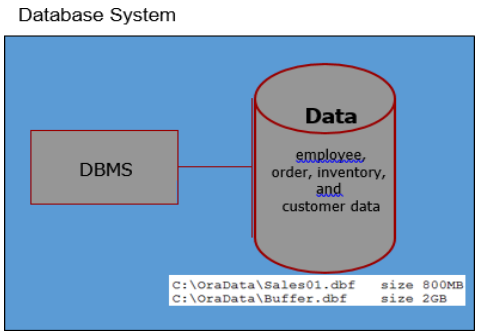


The final step up the information ladder is the step from knowledge to wisdom. We can say that someone has wisdom when they can combine their knowledge and experience to produce a deeper understanding of a topic. It often takes many years to develop wisdom on a particular topic and requires patience.



What is database?

A database is a collection of related data. By data, we mean known facts that can be recorded and that have implicit meaning. For example, consider the names, telephone numbers, and addresses of the people you know. You may have recorded this data in an indexed address book or you may have stored it on a hard drive, using a personal computer and software such as Microsoft Access or Excel. This collection of related data with an implicit meaning is a database.



Lesson 2: Introduction to DBMS & its environment for DBA

What is DBMS and its examples?

Database Management System (DBMS) is a software for storing and retrieving users' data while considering appropriate security measures. It consists of a group of programs which manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software to store and retrieve data.

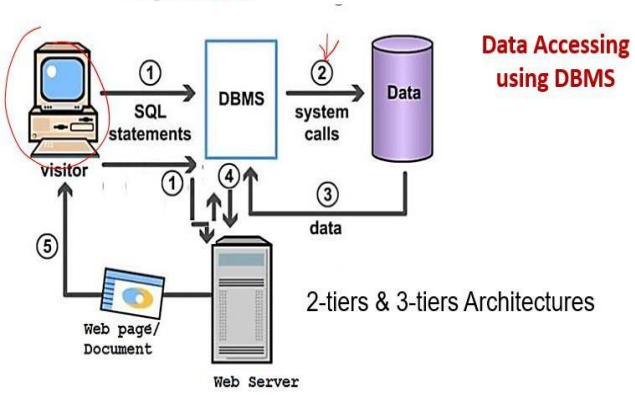
DBMS allows users to create their own databases as per their requirement. The term "DBMS" includes the user of the database and other application programs. It provides an interface between the data and the software application.

Example of DBMS

- Oracle
- IBM DB2
- Ingress
- Teradata
- MS SQL Server
- MS Access
- MySQL

2- Tier Database Architecture
In two-tier, the application logic is either buried inside the User Interface on the client or within the database on the server (or both). With two-tier client/server architectures, the user system interface is usually located in the user's desktop environment and the database management services are usually in a server that is a more powerful machine that services many clients.

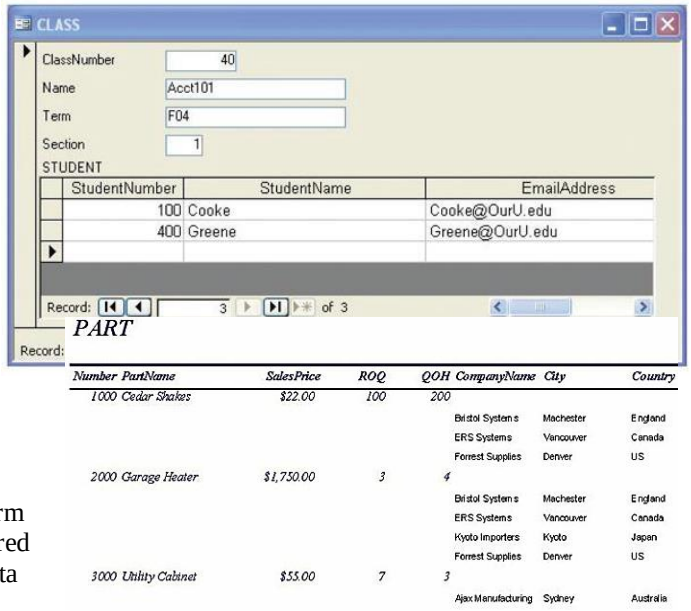
3- Tier Database Architecture
In three-tier, the application logic or process lives in the middle-tier, it is separated from the data and the user interface. Three-tier systems are more scalable,



robust and flexible. In addition, they can integrate data from multiple sources. In the three-tier architecture, a middle tier was added between the user system interface client environment and the database management server environment. There are a variety of ways of implementing this middle tier, such as transaction processing monitors, message servers, or application servers.

Forms

Forms are used for **entering**, **modifying**, and **viewing** records. You likely have had to fill out forms on many occasions, like when visiting a doctor's office, applying for a job, or registering for school. The reason forms are used so often is that they're an easy way to guide people toward entering data correctly. When you enter information into a form in Access, the data goes exactly where the database designer wants it to go in one or more related tables.



Forms make entering data easier. Working with extensive tables can be confusing, and when you have connected tables, you might need to work with more than one at a time to enter a set of data. However, with forms it's possible to enter data into multiple tables at once, all in one place. Database designers can even set restrictions on individual form components to ensure all of the needed data is entered in the correct format. All in all, forms help keep data consistent and organized, which is essential for an accurate and powerful database.

Reports

Reports offer you the ability to **present** your data **in print**. If you've ever received a computer printout of a class schedule or a printed invoice of a purchase, you've seen a database report. Reports are useful because they allow you to present components of your database in an easy-to-read format. You can even customize a report's appearance to make it visually appealing. Access offers you the ability to create a report from any **table** or **query**.

Queries

Queries are a way of **searching** for and **compiling** data from one or more tables. Running a query is like asking a detailed **question** of your database. When you build a query in Access, you are **defining specific search conditions** to find exactly the data you want.

```
SQL> SELECT empno, ename, sal
  2  FROM EMP;

     EMPNO ENAME                SAL
---------- ---------- ----------
      7369 SMITH                800
      7499 ALLEN               1600
```

Queries are far more powerful than the simple searches you might carry out within a table. While a **search** would be able to help you find the name of one customer at your business, you could run a **query** to find the name and phone number of every customer who's made a purchase within the past week. A well-designed query can give information you might not be able to find just by looking through the data in your tables.

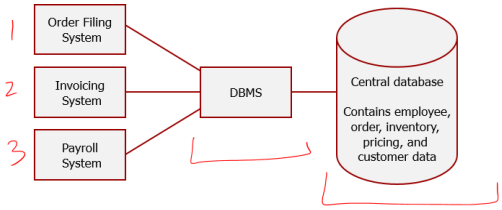
DBMS Languages

A Software Package that enables users to define, create, maintain, and control access to the database.
Data Definition Language (DDL)
Data Manipulation Language (DML)
Control access (DCL):

Security, integrity, concurrent access, recovery, support for data communication, etc.
Utility services (DCL)
File import/export, monitoring facilities, code generator, report writer, etc.
Support Ad Hoc queries

DBMS and its Sub Schemas

A software system that is used to create, maintain, and provide controlled access to user databases

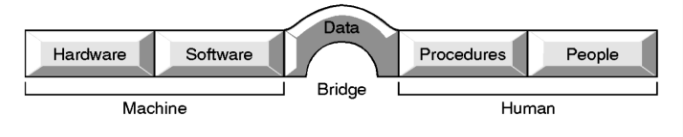


DBMS manages data resources like an operating system manages hardware resources

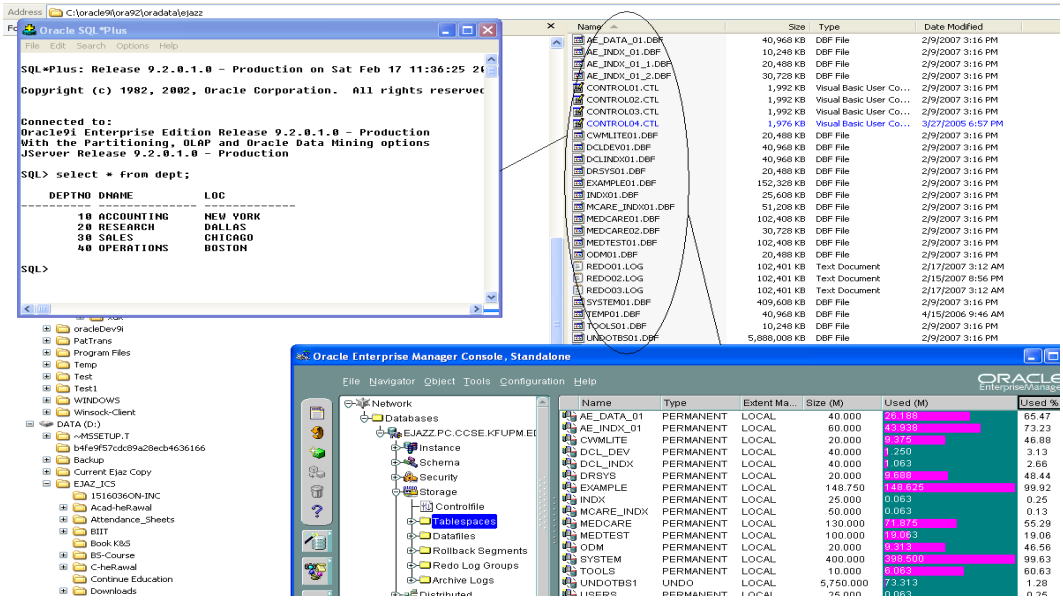
Roles/ Jobs in the Database Environment

- Data Administrator (DA)
- Database Administrator (DBA)
- Database Designers (Logical and Physical)
- Applications Programmer
- Database Developer
- Database Analyst
- End users (Naive and sophisticated)

Components of DBMS Environment

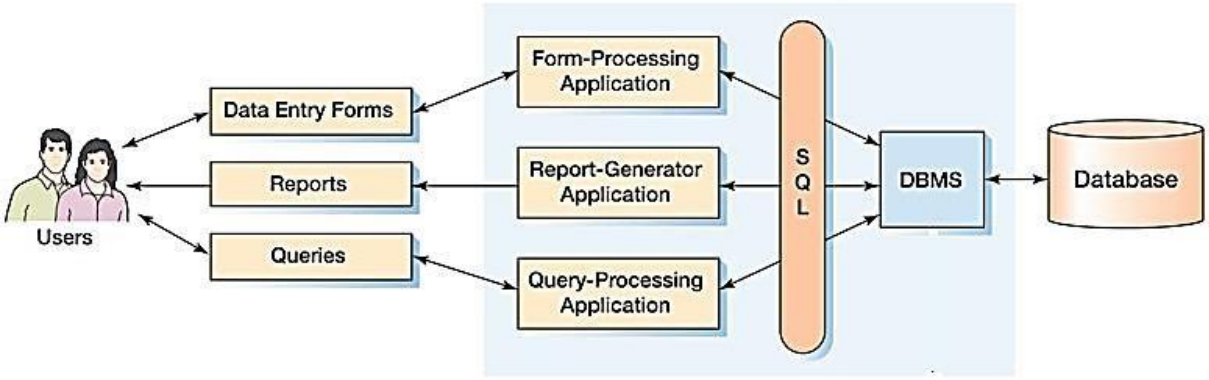


Interaction of DBMS with DB Files



Lesson 3: Introduction to DBMS Components & Architecture

Interaction of user with database



Any user can interact with the database. For example, application programmers and end user.

As its name shows, application programmers are the one who writes application programs that uses the database. These application programs are written in programming languages like COBOL or PL (Programming Language 1), Java and fourth generation language. These programs meet the user requirement and make according to user requirements. Retrieving information, creating new information and changing existing information is done by these application programs. They interact with DBMS through DML (Data manipulation language) calls. And all these functions are performed by generating a request to the DBMS. If application programmers are not there, then there will be no creativity in the whole team of Database.

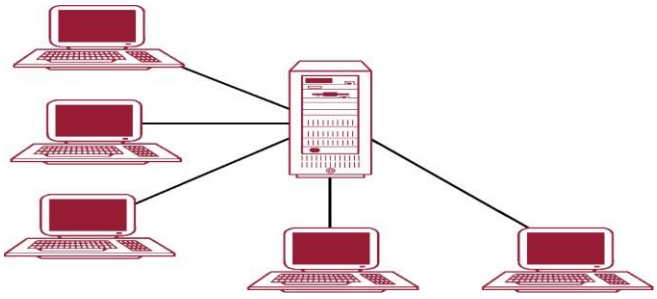
End users are those who access the database from the terminal end. They use the developed applications, and they don't have any knowledge about the design and working of database. These are the second class of users, and their main motto is just to get their task done.

Form Processing & Report Processing Applications are built by using VB, DOT Net or PHP programming. Reports can be developed by using Crystal Reports tool.
Query Processing can be managed by using vendors SQL tool or 3rd party tools such As TOAD, SQL Developer etc.

Multiuser DBMS Architecture

Teleprocessing

One computer with a single CPU and a number of terminals.



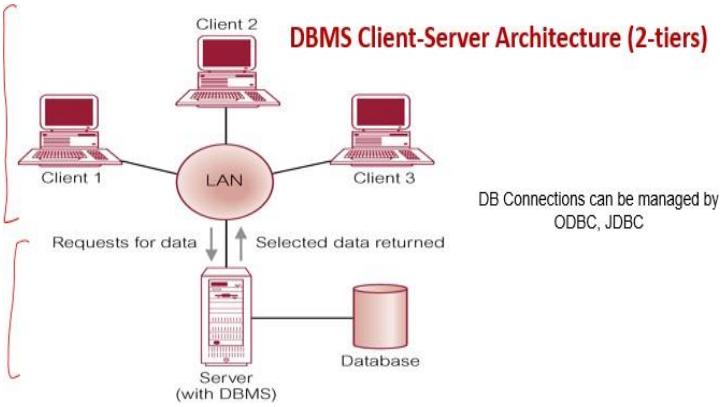
Processing performed within the same physical computer. User terminals are typically "dumb", incapable of functioning on their own, and cabled to the central computer

File-Server

In a file-server environment, the processing is distributed about the network, typically a local area network (LAN).

File-server is connected to several workstations across a network Database resides on file-server. DBMS and applications run on each workstation

Client-Server (2-tiers)



In relational database management systems (RDBMSs), many of which started as centralized systems, the system components that were first moved to the client side were the user interface and application programs. Because SQL provided a standard language for RDBMSs, this created a logical dividing point between client and server. Hence, the query and transaction functionality related to

SQL processing remained on the server side. In such an architecture, the server is often called a query server or transaction server because it provides these two functionalities. In an RDBMS, the server is also often called an SQL server.

The user interface programs, and application programs can run on the client side.

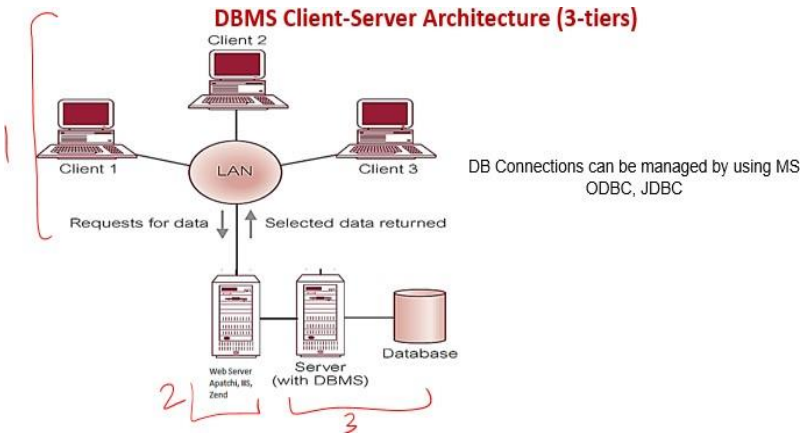
When DBMS access is required, the program establishes a connection to the DBMS (Which is on the server side); once the connection is created, the client program can communicate with the DBMS. A standard called Open Database Connectivity (ODBC) provides an application programming interface (API), which allows client-side programs to call the DBMS, as long as both client and server machines have the necessary software installed. Most DBMS vendors provide ODBC drivers for their systems. A client program can actually connect to several RDBMSs and send query and transaction requests using the ODBC API, which are then processed at the server sites. Any query results are sent back to the client program, which can process and display the results as needed. A related standard for the Java programming language, called JDBC, has also been defined.

This allows Java client programs to access one or more DBMSs through a standard interface.

The different approach to two-tier client/server architecture was taken by some object-oriented DBMSs, where the software modules of the DBMS were divided between client and server in a more integrated way. For example, the server level may include the part of the DBMS software responsible for handling data storage on disk pages, local concurrency control and recovery, buffering and caching of disk pages, and other such functions. Meanwhile, the client level may handle the user interface; data dictionary functions; DBMS interactions with programming language compilers; global query optimization, concurrency control, and recovery across multiple servers; structuring of complex objects from the data in the buffers; and other such functions. In this approach, the client/server interaction is more tightly coupled and is done internally by the DBMS modules—some of which reside on the client and some on the server—rather than by the users/programmers. The exact division of functionality can vary from system to system. In such a client/server architecture, the server has been called a data server because it provides data in disk pages to the client. This data can then be structured into objects for the client programs by the client-side DBMS software.

The architectures described here are called two-tier architectures because the software components are distributed over two systems: client and server. The advantages of this architecture are its simplicity and seamless compatibility with existing systems. The emergence of the Web changed the roles of clients and servers, leading to the three-tier architecture.

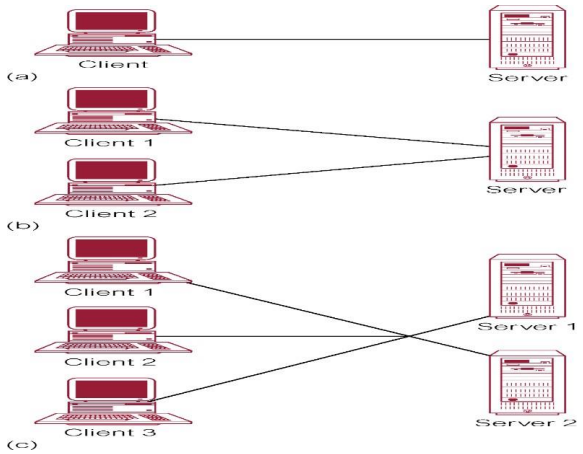
Client server (3-tier)



Many Web applications use an architecture called the three-tier architecture, which adds an intermediate layer between the client and the database server, this intermediate layer or middle tier is called the application server or the Web server, depending on the application. This server plays an intermediary role by running application programs and storing business rules (procedures or constraints) that are used to access data from the database server. It can also improve database security by checking a client's credentials before forwarding a request to the database server. Clients contain GUI interfaces and some additional application-specific business rules. The intermediate server accepts requests from the client, processes the request and sends database queries and commands to the database server, and then acts as a conduit for passing (partially) processed data from the database server to the clients, where it may be processed further and filtered to be presented to users in GUI format. Thus, the user interface, application rules, and data access act as the three tiers. Figure 2.7(b) shows another architecture used by database and other application package vendors. The presentation layer displays information to the user and allows data entry. The business logic layer handles intermediate rules and constraints before data is passed up to the user or down to the DBMS. The bottom

Layer includes all data management services. The middle layer can also act as a Web server, which retrieves query results from the database server and formats them into dynamic Web pages that are viewed by the Web browser at the client side. Other architectures have also been proposed. It is possible to divide the layers between the user and the stored data further into finer components, thereby giving rise to n-tier architectures, where n may be four or five tiers. Typically, the business logic layer is divided into multiple layers. Besides distributing programming and data throughout a network, n-tier applications afford the advantage that any one tier can run on an appropriate processor or operating system platform and can be handled independently. Vendors of ERP (enterprise resource planning) and CRM (customer relationship management) packages often use a middleware layer, which accounts for the front-end modules (clients) communicating with a number of back-end databases (servers).

Alternate Client-Server Architectures



Paradigm of Database System

- **Logical/ Conceptual Database Design**
  - How should the transactional data be modeled (OLTP)?
  - Designing the data warehouse schema (OLAP)

- **Query Processing**
  - Analysis queries are hard to answer efficiently
  - What techniques are available to the DBMS?
- **Physical Database Design (DBA)**
  - How should the data be organized on disk?
  - What data structures should be used?
- **Data Mining**
  - What use is all this data?
  - Which questions should we ask our data warehouse OLAP?
- **Data Science/Big Data**

A conceptual data model is a model that helps to identify the highest-level relationships between the different entities, while a logical data model is a model that describes the data as much detail as possible, without regard to how they will be physically implemented in the database.Query Processing is a translation of high-level queries into low-level expression. It is a step wise process that can be used at the physical level of the file system, query optimization and actual execution of the query to get the result. It requires the basic concepts of relational algebra and file structure. It refers to the range of activities that are involved in extracting data from the database. It includes translation of queries in high-level database languages into expressions that can be implemented at the physical level of the file system. In query processing, we will actually understand how these queries are processed and how they are optimized.

Physical database design is the process of transforming logical data models into physical data models. An experienced database designer will make a physical database design in parallel with conceptual data modeling if they know the type of database technology that will be used.

Lesson 4: Introduction to DBMS Three Levels Architecture

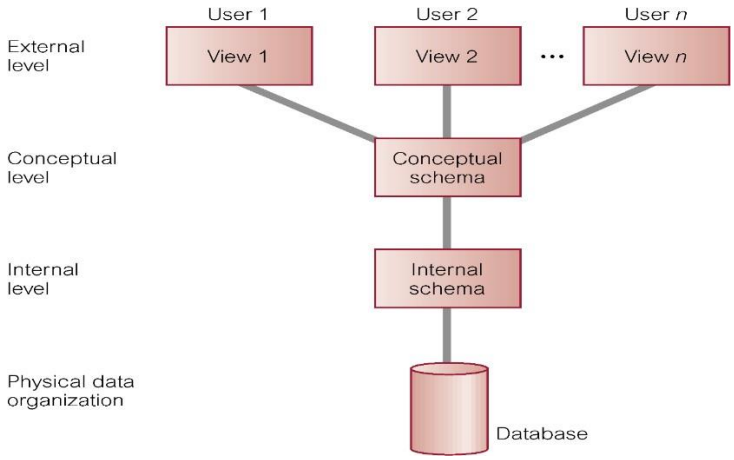
DBMS Three Levels Architecture

In this architecture, schemas can be defined at the following three levels:

1. The internal level has an internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

2. The conceptual level has a conceptual schema, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.

Usually, a representational data model is used to describe the conceptual schema when a database system is implemented. This implementation conceptual schema is often based on a conceptual schema design in a high-level data model.

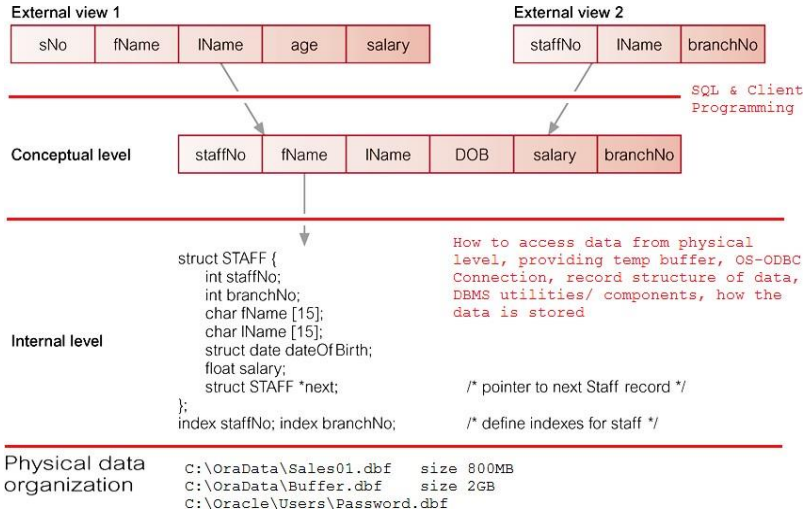


3. The external or view level includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. As in the previous level, each external schema is typically implemented using a representational data model, possibly based

on an external schema design in a high-level data model.

The three-schema architecture is a convenient tool with which the user can visualize the schema levels in a database system. Most DBMSs do not separate the three levels completely and explicitly but support the three-schema architecture to some extent.

Some older DBMSs may include physical-level details in the conceptual schema.



The three-level ANSI architecture has an important place in database technology development because it clearly separates the users' external level, the database's conceptual level, and the internal storage level for designing a database. It is very much applicable in the design of DBMSs, even today. In most DBMSs that support user views, external schemas are specified in the same data model that describes the conceptual-level information (for example, a relational DBMS like Oracle uses SQL for this). Some DBMSs allow different data models to be used at the conceptual and external levels. An example is Universal Data Base (UDB), a DBMS from IBM, which uses the relational model to describe the conceptual schema, but may use an object-oriented model to describe an external schema.

Notice that the three schemas are only descriptions of data; the stored data that actually exists is at the physical level only. In a DBMS based on the three-schema architecture, each user group refers to its own external schema. Hence, the DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database. If the request is a database retrieval, the data extracted from the stored database must be reformatted to match the user's external view. The processes of transforming requests and results between levels are called mappings.

These mappings may be time-consuming, so some DBMSs—especially those that are meant to support small databases—do not support external views. Even in such systems, however, a certain amount of mapping is necessary to transform requests between the conceptual and internal levels.

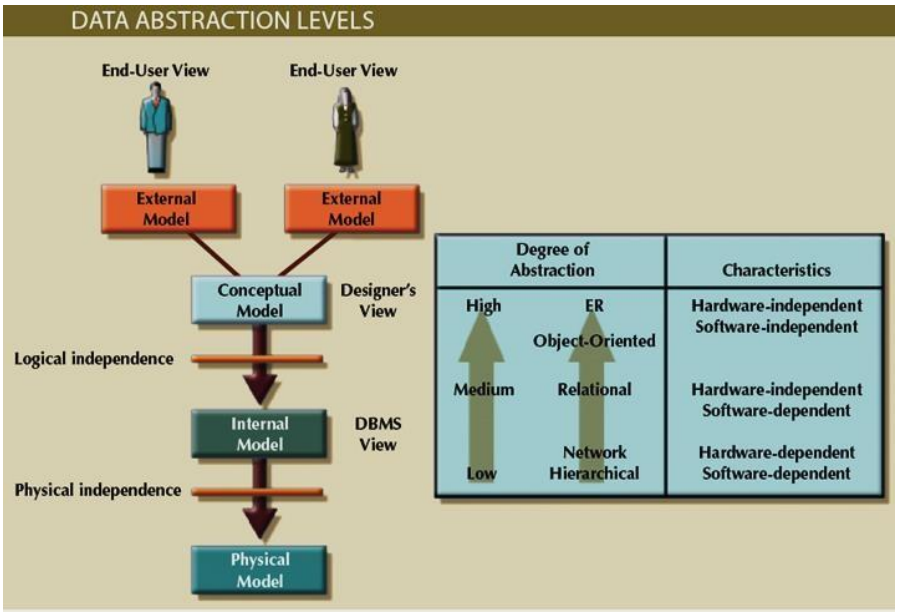
Data Independence

The three-schema architecture can be used to further explain the concept of data independence, which can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. We can define two types of data independence:

Logical data independence is the capacity to change the conceptual schema without having to change external schemas or application programs. We may change the conceptual schema to expand the database (by adding a record type or data item), to change constraints, or to reduce the database (by removing a record type or data item).

Physical data independence is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas need not be changed as well. Changes to the internal schema may be needed because some physical files were reorganized—for example, by creating additional access structures—to improve the performance of retrieval or update. If the same data as before remains in the database, we should not have to change the conceptual schema.

Generally, physical data independence exists in most databases and file environments where physical details such as the exact location of data on disk, and hardware details of storage encoding, placement, compression, splitting, merging of records, and so on are hidden from the user. Applications remain unaware of these details. On the other hand, logical data independence is harder to achieve because it allows structural and constraint changes without affecting application programs—a much stricter requirement.



Whenever we have a multiple-level DBMS, its catalog must be expanded to include information on how to map requests and data among the various levels. The DBMS uses additional software to accomplish these mappings by referring to the mapping information in the catalog. Data independence occurs because when the schema is changed at some level, the schema at the next higher level remains unchanged; only the mapping between the two levels is changed. Hence, application programs referring to the higher-level schema need not be changed.

The three-schema architecture can make it easier to achieve true data independence, both physical and logical. However, the two levels of mappings create an overhead during compilation or execution of a query or program, leading to inefficiencies in the DBMS. Because of this, few DBMSs have implemented the full three schema architecture.

Database schema A database schema is a blueprint or architecture of how our data will look. It doesn't hold data itself, but instead describes the shape of the data and how it might relate to other tables or models. An entry in our database will be an instance of the database schema. It will contain all of the properties described in the schema.

Schema types There are two main database schema types that define different parts of the schema: logical and physical. A **logical database schema** represents how the data is organized in terms of tables. It also explains how attributes from tables are linked together. Different schemas use a different syntax to define the logical architecture and constraints.

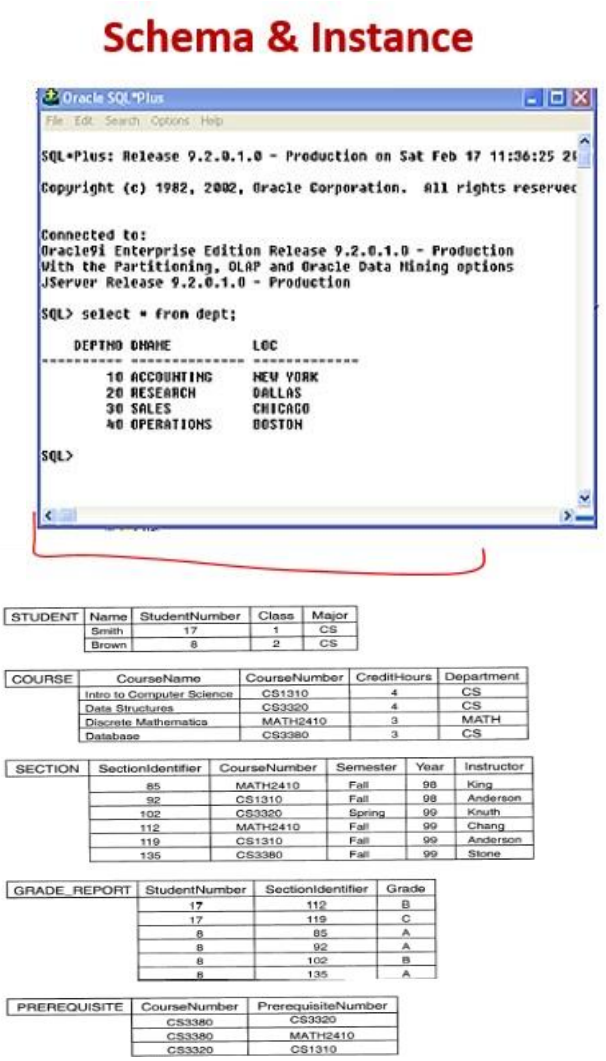
To create a logical database schema, we use tools to illustrate relationships between components of your data. This is called entity-relationship modeling (ER Modeling). It specifies what the relationships between entity types are.

The **physical database schema** represents how data is stored on disk storage. In other words, it is the actual code that will be used to create the structure of your database. In MongoDB with mongoose, for instance, this will take the form of a mongoose model. In MySQL, you will use SQL to construct a database with tables.

Schema objects

A schema is a collection of schema objects. Examples of schema objects include tables, views, sequences, synonyms, indexes, clusters, database links, procedures, and packages. This chapter explains tables, views, sequences, synonyms, indexes, and clusters.

Schema objects are logical data storage structures. Schema objects do not have a one-to-one correspondence to physical files on disk that store their information. However, Oracle stores a schema object logically within a tablespace of the database. The data of each object is physically contained in one or more of the tablespace's data files. For some objects such as tables, indexes, and clusters, you can specify how much disk space Oracle allocates for the object within the tablespace's data files.



Lesson 5: Introduction to DBA Responsibilities & Tasks

Who is DBA?

- A database administrator (DBA) is a person or group of persons who maintains a successful database environment by directing or performing all related activities to keep the data secure. The top responsibility of a DBA professional is to maintain data integrity. This means the DBA will ensure that data is secure from unauthorized access but is available to users.
- DBA job requires a high level of expertise by a person or group of persons.
- There are very rare chances that only a single person can manage all the database system activities, so companies always have a group of people who take care of database system.

DBA & Databases

- Each database requires at least one database administrator (DBA).
- A Database system can be large and can have many users.
- A database administration is sometimes not a one-person job, but a job for a group of DBAs who share responsibility.

Network Administrator

- A network administrator is a person designated in an organization whose responsibility includes maintaining computer infrastructures with emphasis on local area networks (LANs) up to wide area networks (WANs). Responsibilities may vary between organizations, but installing new hardware, on-site servers, enforcing licensing agreements, software-network interactions, as well as network integrity/resilience, are some of the key areas of focus.
- Network administrator coordinates with the DBA for database connections and other issues such as storage, OS and hardware.
- Some sites have one or more network administrators. A network administrator, for example, administers Oracle networking products, such as Oracle Net Services.

Application Developers

- Designing and developing the database application
- Designing the database structure for an application
- Estimating storage requirements for an application
- Specifying modifications of the database structure for an application
- Relaying this information to a database administrator
- Tuning the application during development
- Establishing security measures for an application during development
- Database Server Programming using Oracle PL/SQL

DBA's Tasks

- Task 1: Evaluate the Database Server Hardware
- Task 2: Install the Oracle Database Software
- Task 3: Plan the Database
- Task 4: Create and Open the Database
- Task 5: Back Up the Database
- Task 6: Enroll System Users
- Task 7: Implement the Database Design
- Task 8: Back Up the Fully Functional Database
- Task 9: Tune Database Performance
- Task 10: Download and Install Patches
- Task 11: Roll Out to Additional Hosts

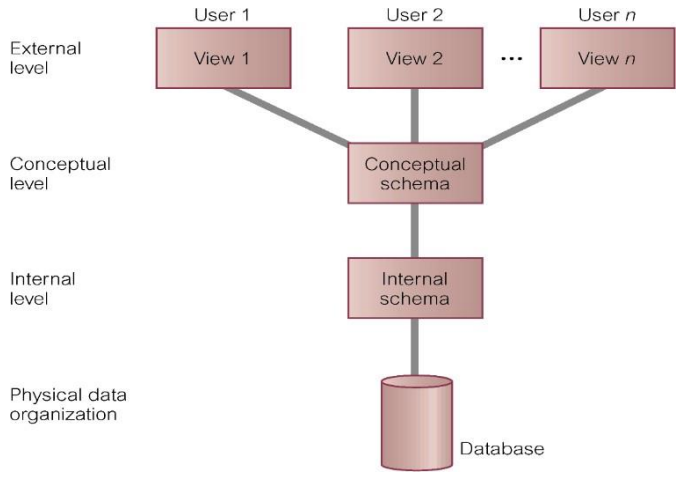
DBA's Responsibilities

- Installing and upgrading the Oracle Database server and application tools
- Allocating system storage and planning future storage requirements for the database system
- Creating primary database storage structures (tablespaces) after application developers have designed an application
- Creating primary objects (tables, views, indexes) once application developers have designed an application
- Modifying the database structure, as necessary, from information given by application developers
- Enrolling users and maintaining system security
- Ensuring compliance with Oracle license agreements
- Controlling and monitoring user access to the database

- Monitoring and optimizing the performance of the database
- Planning for backup and recovery of database information
- Maintaining archived data on tape
- Backing up and restoring the database
- Contacting Oracle for technical support

Lesson 6: Overview of Physical Database Design

Physical database design is the process of transforming logical data models into physical data models. An experienced database designer will make a physical database design in parallel with conceptual data modeling if they know the type of database technology that will be used.



Purposes

Meeting the expectations of Database Designer for the database, following are two main purposes of Physical Database Design for a DBA.

Managing Storage Structure for database or DBMS

Performance & Tuning

Factor (A): Analyzing the database queries and transactions

Before undertaking the physical database design, we must have a good idea of the intended use of the database by defining in a high-level form the queries and transactions that are expected to run on the database. For each retrieval query, the following information about the query would be needed:

1. The files that will be accessed by the query.

2. The attributes on which any selection conditions for the query are specified.

3. Whether the selection condition is an equality, inequality, or a range condition.

4. The attributes on which any join conditions or conditions to link multiple tables or objects for the query are specified.

5. The attributes whose values will be retrieved by the query.

The attributes listed in items 2 and 4 above are candidates for the definition of access structures, such as indexes, hash keys, or sorting of the file.

For each update operation or update transaction, the following information would be needed:

1. The files that will be updated.

2. The type of operation on each file (insert, update, or delete).

3. The attributes on which selection conditions for a delete or update are specified.

4. The attributes whose values will be changed by an update operation.

Again, the attributes listed in item 3 are candidates for access structures on the files, because they would be used to locate the records that will be updated or deleted. On the other hand, the attributes listed in item 4 are candidates for avoiding an access structure, since modifying them will require updating the access structures.

Factor (B): Frequency with Queries and Transactions

Besides identifying the characteristics of expected retrieval queries and update transactions, we must consider their expected rates of invocation. This frequency information, along with the attribute information collected on each query and transaction, is used to compile a cumulative list of the expected frequency of use for all queries and transactions. This is expressed as the expected frequency of using each attribute in each file as a selection attribute or a join attribute, over all the queries and transactions. Generally, for large volumes of processing, the informal 80–20 rule can be used: approximately 80 percent of the processing is accounted for by only 20 percent of the queries and transactions. Therefore, in practical situations, it is rarely necessary to collect exhaustive statistics and invocation rates on all the queries and transactions; it is sufficient to determine the 20 percent or so most important ones.

Factor (C): Time constraints of queries & transactions

Some queries and transactions may have stringent performance constraints. For example, a transaction may have the constraint that it should terminate within 5 seconds on 95 percent of the occasions when it is invoked, and that it should never take more than 20 seconds. Such timing constraints place further priorities on the attributes that are candidates for access paths. The selection attributes used by queries and transactions with time constraints become higher-priority candidates for primary access structures for the files because the primary access structures are generally the most efficient for locating records in a file.

Factor (D): Expected frequencies of update operations

A minimum number of **access paths** should be specified for a file that is frequently updated, because updating the access paths themselves slows down the update operations. For example, if a file that has frequent record insertions has 10 indexes on 10 different attributes, each of these indexes must be updated whenever a new record is inserted. The overhead for updating 10 indexes can slow down the insert operations.

Factor (E): Uniqueness constraints on attributes

Access paths should be specified on all candidate key attributes—or sets of attributes—that are either the primary key of a file or unique attributes. The existence of an index (or other access path) makes it sufficient to only search the index when checking this uniqueness constraint, since all values of the attribute will exist in the leaf nodes of the index. For example, when inserting a new record, if a key attribute value of the new record already exists in the index, the insertion of the new record should be rejected, since it would violate the uniqueness constraint on the attribute. Once the preceding information is compiled, it is possible to address the physical database design decisions, which consist mainly of deciding on the storage structures and access paths for the database files.

Design Decisions about Indexing.

The attributes whose values are required in equality or range conditions (selection operation) are those that are keys or that participate in join conditions (join operation) requiring access paths, such as indexes.

The performance of queries largely depends upon what indexes or hashing schemes exist to expedite the processing of selections and joins. On the other hand, during insert, delete, or update operations, the existence of indexes adds to the overhead. This overhead must be justified in terms of the gain in efficiency by expediting queries and transactions. The physical design decisions for indexing fall into the following categories:

1. **Whether to index an attribute.** The general rules for creating an index on an attribute are that the attribute must either be a key (unique), or there must be some query that uses that attribute either in a selection condition (equality or range of values) or in a join condition. One reason for creating multiple indexes is that some operations can be processed by just scanning the indexes, without having to access the actual data file.

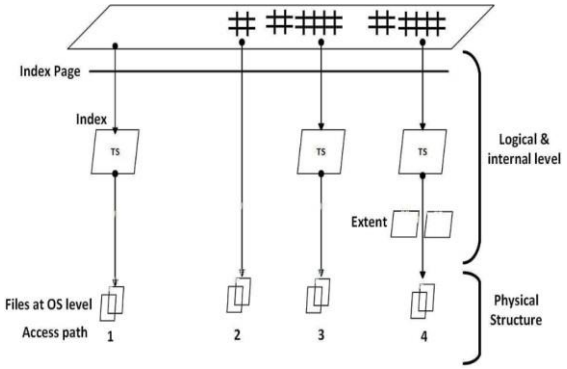
2. **What attribute or attributes to index on**. An index can be constructed on a single attribute, or on more than one attribute if it is a composite index. If multiple attributes from one relation are involved together in several queries, (for example, (Garment_style_#, Color) in a garment inventory database), a Multi attribute (composite) index is warranted. The ordering of attributes within a multiattribute index must correspond to the queries. For instance, the above index assumes that queries would be based on an ordering of colors within a Garment_style_# rather than vice versa.

3. **Whether to set up a clustered index.** At most, one index per table can be a primary or clustering index, because this implies that the file be physically ordered on that attribute. In most RDBMSs, this is specified by the keyword CLUSTER. (If the attribute is a key, a primary index is created, whereas a clustering index is created if the attribute is not a key.) If a table requires several indexes, the decision about which one should be the primary or clustering index depends upon whether keeping the table ordered on that attribute is needed. Range queries benefit a great deal from clustering. If several attributes require range queries, relative benefits must be evaluated before deciding which attribute to cluster on. If a query is to be answered by doing an index search only (without retrieving data records), the corresponding index should not be clustered, since the main benefit of clustering is achieved when retrieving the records themselves. A clustering index may be set up as a multi attribute index if range retrieval by that composite key is useful in report creation (for example, an index on Zip_code, Store_id, and Product_id may be a clustering index for sales data).

4. **Whether to use a hash index over a tree index**. In general, RDBMSs use B+- trees for indexing. However, ISAM and hash indexes are also provided in some systems (see Chapter 18). B+-trees support both equality and range queries on the attribute used as the search key. Hash indexes work well with equality conditions, particularly during joins to find a matching record(s), but they do not support range queries

5. **Whether to use dynamic hashing for the file.** For files that are very volatile—that is, those that grow and shrink continuously.

Access Path & Storage



Access path: How data is exactly stored in back end files, how and where files are stored. Data is managed logically in the form of segments and extents for tablespaces. Also, how data is indexed and stored on separate indexed tablespaces. OS disk partitions are also be a part of Access path.

Lesson 7: Introduction to Database Tuning

Database Tuning and Goal

The process of continuing to revise/adjust the physical database design by monitoring resource utilization as well as internal DBMS processing to reveal bottlenecks such as contention for the same data or devices.

The goals of tuning are as follows:

- To make applications run faster.
- To improve (lower) the response time of queries and transactions.
- To improve the overall throughput of transactions.

Tuning Indexes

The initial choice of indexes may have to be revised for the following reasons:

- Certain queries may take too long to run for lack of an index.
- Certain indexes may not get utilized at all.
- Certain indexes may undergo too much updating because the index is on an attribute that undergoes frequent changes.

Most DBMSs have a command or trace facility, which can be used by the DBA to ask the system to show how a query was executed—what operations were performed in what order and what secondary access structures (indexes) were used. By analyzing these execution plans, it is possible to diagnose the causes of the above problems.

Some indexes may be dropped and some new indexes may be created based on the tuning analysis.

The goal of tuning is to dynamically evaluate the requirements, which sometimes fluctuate seasonally or during different times of the month or week, and to reorganize the indexes and file organizations to yield the best overall performance. Dropping and building new indexes is an overhead that can be justified in terms of performance improvements. Updating of a table is generally suspended while an index is dropped or created; this loss of service must be accounted for. Besides dropping or creating indexes and changing from a nonclustered to a clustered index and vice versa, rebuilding the index may improve performance. Most

RDBMSs use B+-trees for an index. If there are many deletions on the index key, index pages may contain wasted space, which can be claimed during a rebuild operation. Similarly, too many insertions may cause overflows in a clustered index that affect performance. Rebuilding a clustered index amounts to reorganizing the entire table ordered on that key.

The available options for indexing and the way they are defined, created, and reorganized varies from system to system. As an illustration, consider the sparse and dense indexes. A sparse index such as a primary index will have one index pointer for each page (disk block) in the data file; a dense index such as a unique secondary index will have an index pointer for each record. Sybase provides clustering indexes as sparse indexes in the form of B+-trees, whereas INGRES provides sparse clustering indexes as ISAM files and dense clustering indexes as B+-trees. In some versions of Oracle and DB2, the option of setting up a clustering index is limited to a dense index (with many more index entries), and the DBA has to work with this limitation.

Overview of Database Tuning

Statistics obtained from monitoring:

Storage statistics: Data about allocation of storage into tablespaces, index spaces, and buffer pools.

I/O and device performance statistics: Total read/write activity (paging) on disk extents and disk hot spots.

Query/transaction processing statistics: Execution times of queries and transactions, and optimization times during query optimization.

Locking/logging related statistics: Rates of issuing different types of locks, transaction throughput rates, and log records activity

Index statistics: Number of levels in an index, number of noncontiguous leaf pages, and so on.

Statistics internally collected in DBMSs:

Size of individual tables

Number of distinct values in a column

The number of times a particular query or transaction is submitted/executed in an interval of time

The times required for different phases of query and transaction processing

Problems in Tuning

Tuning a database involves dealing with the following types of problems:

- How to avoid excessive lock contention, thereby increasing concurrency among transactions.
- How to minimize the overhead of logging and unnecessary dumping of data.
- How to optimize the buffer size and scheduling of processes.
- How to allocate resources such as disks, RAM, and processes for most efficient utilization.

Most of the previously mentioned problems can be solved by the DBA by setting appropriate physical DBMS parameters, changing configurations of devices, changing operating system parameters, and other similar activities. The solutions tend to be closely tied to specific systems. The DBAs are typically trained to handle these tuning problems for the specific DBMS.

Changes in Database Design

Existing tables may be joined (denormalized) because certain attributes from two or more tables are frequently needed together: This reduces the normalization level from BCNF to 3NF, 2NF, or 1NF.5

For the given set of tables, there may be alternative design choices, all of which achieve 3NF or BCNF.We illustrated alternative equivalent designs. One normalized design may be replaced by another.

A relation of the form R(K,A, B, C, D, ...)—with K as a set of key attributes— that is in BCNF can be stored in multiple tables that are also in BCNF—for example, R1(K, A, B), R2(K, C, D,), R3(K, ...)—by replicating the key K in each table. Such a process is known as vertical partitioning. Each table groups sets of attributes that are accessed together. For example, the table EMPLOYEE(Ssn, Name, Phone, Grade, Salary) may be split into two tables: EMP1(Ssn, Name, Phone) and EMP2(Ssn, Grade, Salary). If the original table has a large number of rows (say 100,000) and queries about phone numbers and salary information are totally distinct and occur with very different frequencies, then this separation of tables may work better.

Attribute(s) from one table may be repeated in another even though this creates redundancy and a potential anomaly. For example, Part_name may be replicated in tables wherever the Part# appears (as foreign key), but there may be one master table called PART_MASTER(Part#, Part_name, ...) where the Partname is guaranteed to be up-to-date.

Just as vertical partitioning splits a table vertically into multiple tables, horizontal partitioning takes horizontal slices of a table and stores them as distinct tables. For example, product sales data may be separated into ten tables based on ten product lines. Each table has the same set of columns (attributes) but contains a distinct set of products (tuples). If a query or transaction applies to all product data, it may have to run against all the tables and the results may have to be combined.

Tuning Queries

- Indications for tuning queries
    - A query issues too many disk accesses
    - The query plan shows that relevant indexes are not being used.

Some typical instances of situations prompting query tuning include the following:

1. Many query optimizers do not use indexes in the presence of arithmetic expressions (such as Salary/365 > 10.50), numerical comparisons of attributes of different sizes and precision (such as Aqty = Bqty where Aqty is of type

INTEGER and Bqty is of type SMALLINTEGER), NULL comparisons (such as

Bdate IS NULL), and substring comparisons (such as Lname LIKE '%mann').

2. Indexes are often not used for nested queries using IN; for example, the following query:

SELECT Ssn FROM EMPLOYEE

WHERE Dno IN (SELECT Dnumber FROM DEPARTMENT

WHERE Mgr_ssn = '333445555'); may not use the index on Dno in EMPLOYEE, whereas using Dno = Dnumber in the WHERE-clause with a single block query may cause the index to be used.

3. Some DISTINCTs may be redundant and can be avoided without changing the result. A DISTINCT often causes a sort operation and must be avoided as much as possible.

4. Unnecessary use of temporary result tables can be avoided by collapsing multiple queries into a single query unless the temporary relation is needed for some intermediate processing.

5. In some situations involving the use of correlated queries, temporaries are useful. Consider the following query, which retrieves the highest paid employee in each department:

SELECT Ssn

FROM EMPLOYEE E

WHERE Salary = SELECT MAX (Salary)

FROM EMPLOYEE AS M

WHERE M.Dno = E.Dno;

This has the potential danger of searching all of the inner EMPLOYEE table M for each tuple from the outer EMPLOYEE table E. To make the execution more efficient, the process can be broken into two queries, where the first query just computes the maximum salary in each department as follows:

SELECT MAX (Salary) AS High_salary, Dno INTO TEMP

FROM EMPLOYEE

GROUP BY Dno;

SELECT EMPLOYEE.Ssn

FROM EMPLOYEE, TEMP

WHERE EMPLOYEE.Salary = TEMP.High_salary

AND EMPLOYEE.Dno = TEMP.Dno;

6. If multiple options for a join condition are possible, choose one that uses a clustering index and avoid those that contain string comparisons. For example, assuming that the Name attribute is a candidate key in EMPLOYEE and

STUDENT, it is better to use EMPLOYEE.Ssn = STUDENT.Ssn as a join condition rather than EMPLOYEE.Name = STUDENT.Name if Ssn has a clustering index in one or both tables.

7. One idiosyncrasy with some query optimizers is that the order of tables in the FROM-clause may affect the join processing. If that is the case, one may have to switch this order so that the smaller of the two relations is scanned and the larger relation is used with an appropriate index.

8. Some query optimizers perform worse on nested queries compared to their equivalent unnested counterparts. There are four types of nested queries:

- Uncorrelated subqueries with aggregates in an inner query.
- Uncorrelated subqueries without aggregates.
- Correlated subqueries with aggregates in an inner query.
- Correlated subqueries without aggregates.

Of the four types above, the first one typically presents no problem, since most query optimizers evaluate the inner query once. However, for a query of the second type, such as the example in item 2, most query optimizers may not use an index on Dno in EMPLOYEE. However, the same optimizers may do so if the query is written as an unnested query. Transformation of correlated subqueries may involve setting temporary tables. Detailed examples are outside our scope here.

9. Finally, many applications are based on views that define the data of interest to those applications. Sometimes, these views become overkill, because a query may be posed directly against a base table, rather than going through a view that is defined by a JOIN.

Lesson 8: Concepts of Keys & Super Key

Concepts of Keys

- A key is a combination of one or more columns that is used to identify rows in a relation
- A key in DBMS is an attribute or a set of attributes that help to uniquely identify a tuple (or row) in a relation (or table). Keys are also used to establish relationships between the different tables and columns of a relational database. Individual values in a key are called key values.
- A composite key is a key that consists of two or more columns

Differentiating record from other records based on columns called Key-columns.

When someone is searching account balance through CNIC when account No. is not provided.

On search two account No.s will be appeared, one is single and other is joint account. A situation is raised, exactly which record of balance is needed.

Super Keys

- A super key is a combination of columns that uniquely identifies any row within a relational database management system (RDBMS) table.

Roll#	FirstName	LastName	Address	City	NIC#	Deptno
...						
12i-2565	...					CS
	Ali					CS
	Ali					EE
...						

{Roll#}, {Roll#, NIC#}, {NIC#}, {Roll#, FirstName, LastName}

{Roll#, FirstName, LastName, Address, City, NIC#, Deptno}

Super Keys Example – 1

Roll Number	First Name	Last Name
CSU0001	Shabbir	Bhimani
CSU0002	Jain	Rao
CSU0003	Nitin	Uday

Now we have the following as super keys

Roll Number	First Name	
Roll Number	**First Name**	**Last Name**

Super Keys Example – 2

Following table consists of four columns

EmployeeID, Name, Job, DeptID

- Examples of superkeys in this table would be {employeeID, Name}, {employeeID, Name, job}, and {employeeID, Name, job, departmentID}

In a real database we don't need values for all of those columns to identify a row

- We only need, per our example, the set {EmployeeID}.
  - This is a minimal superkey
  - So, employeeID is a candidate key.
  - EmployeeID can also uniquely identify the tuples.

Lesson 9: Candidate Keys Examples - I

Candidate Key Definition:

- A **candidate key** is a key that determines all of the other columns in a relation
- Candidate key columns help in searching fewer duplicated or unique records.

Examples

Product can be anything like biscuits, stationery, and books etc.

In PRODUCT relation

Prod# is a candidate key

Prod_Name is also a candidate key

In ORDER_PROD

(OrderNumber, Prod#) is a candidate key

(Candidate key can be called as Alternate Key)

Candidate Key Example Invoice (Buying Items)

- Items are sold to Customers
- Customer should buy Items with Invoice
- Same items can be sold on many invoices
- Invoice# and Item# will identifying exact record, what other columns are required
- If someone tells you inv# and Qty, can you find exact product

Therefore, different candidate keys are used in different organization e.g,

For a Bank as an Enterpise: AccountHolder (or Customer)

ACC#, Fname, Lname, DOB, CNIC#, Addr, City, TelNo, Mobile#, DriveLic#

For PTCL: Customer (single telno holders)

Consumer#, Fname, Lname, DOB, CNIC#, Addr, City, TelNo, Mobile#

For NADRA: Citizen (CNIC#, Fname, Lname, FatherName, DOB, OldCNIC#, PAddr, PCity, TAddr, TCity, TelNo, Mobile#)

Lesson 10: Candidate Keys Examples - II

Candidate key is used for the searching purposes in the logical and conceptual database system.

Candidate Key Examples

Example-1: Branch (branch-name, assets, branch-city)

Candidate Key: {branch-name}

Example-2: Customer (cname, natid, address, city, telno)

cname, address, city can be duplicated individually and cannot determine a record.

The following combinations distinguish customer records or tuples.

{cname, telno}, {natid}, {natid, cname}

As {natid} ⊂ {natid, cname}, then {natid, cname} is not candidate key and {natid} is a candidate key

Example-3: Employee(empno, name, birth_date, address, city, telno, citizenship_id)

empno, telno, citizenship_id are possible candidate keys.

Exercises

Course (cid, cname, deptno)

Semester (sid, syear, startdate)

ClassAllocation (sid, cid, sec#, building#, room#)

Identify candidate keys in each of the above relations

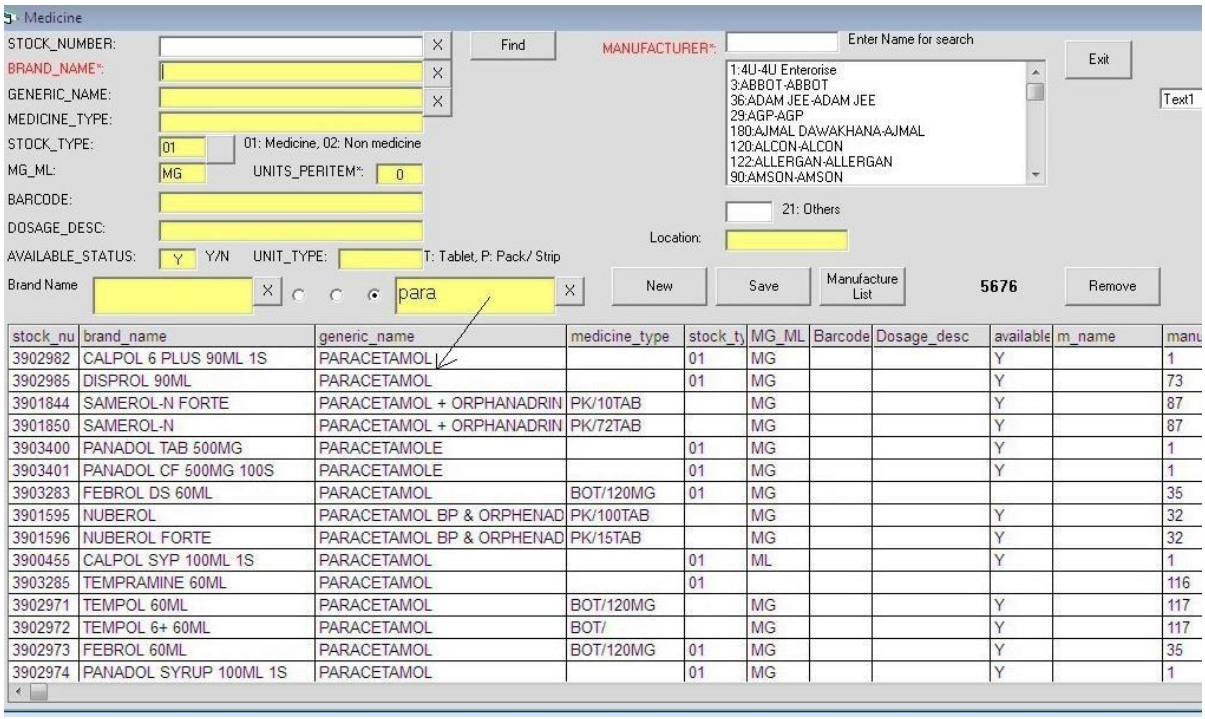
Answer:

Candidate key from Course table: {cid}, {cname}

Candidate key from Semester table: {sid}

Candidate key from ClassAllocation table: { sid, cid, sec#}

Candidate Key for Search purpose



Lesson 11: Primary Keys

Primary Key Definition:

A **primary key** is a candidate key selected as the primary means of identifying rows in a relation:

Characteristics of Primary Key:

- There is one and only one primary key per relation
- The primary key is NOT NULL, UNIQUE
- The ideal primary key is short, numeric(alpha), indexed, fixed length and never changes
- Key that has its ownership of an Enterprise
- The primary key may be a composite key

Alternate Definition of Primary key:

A primary key is a minimal identifier (means maybe 2 or 3 columns will gives you uniqueness) that is used to identify tuples uniquely.

This means that no subset of the primary key is sufficient to provide unique identification of tuples.

NULL values is not allowed in primary key attribute.

Primary key - Example & Issues:

Example:

- STU_ID

(Practically PK must have UNIQUE, NOT NULL, INDEXED on an attribute)

STUDENT

STU_ID	NAME	CID_ID	DOB	DEPTCODE

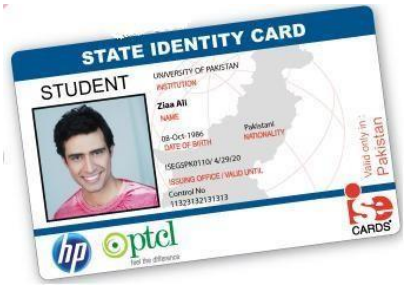
Although, candidate keys are STU_ID, CIT_ID. Organization ITSELF issues its own identifier PK. PK is a property of an organization. Driving license# is a property of Ministry of Motors/ transportation, it must not be used PK in university or other organizations. It is a candidate key for other organization for search purposes.

Note: It is not recommended to create PKs for columns like Quantity, Salary, Price and column that has datatype DATE

Lesson 12: Primary Keys Examples - I

We will now discuss how to identify the primary key in deferent examples.

Primary key - Student



Example-1:

STUDENT(StuID, FirstName, FamilyName, DOB, …)

Primary Key - Building

Example-2: Building (B#, BName, Location, Region),

B# is a primary key. Although, BName is unique, not null but it is not short.

Primary Key - Customer

Example-3: Customer (cname, citizenid, address, city, telno)

This relation indicates the information about personal details. There is a chance that cname is duplicated, some may have citizenid and telno as null. This forces us to introduce new a attribute such as cust# that would be a primary key.

Customer (cust#, cname, citizenid, address, city, telno)

Primary Key - Bank

Example-4:

BankBranch(Branch-Name, City, TotalAsset)

- What will be a PK that you will suggest?
  - BranchNo
- What is a candidate key?
  - BranchNo
  - Branch-Name
- Which attribute is unique?
  - BranchNo
  - Branch-Name
- There may be many branches in one city then finalize this relation with possible constraints

Lesson 13: Primary Keys Examples - II

In this topic we are going to discuss more example because we need to analyze further new issues from the real life that how we can decide a well formatted and well-organized kind of primary keys.

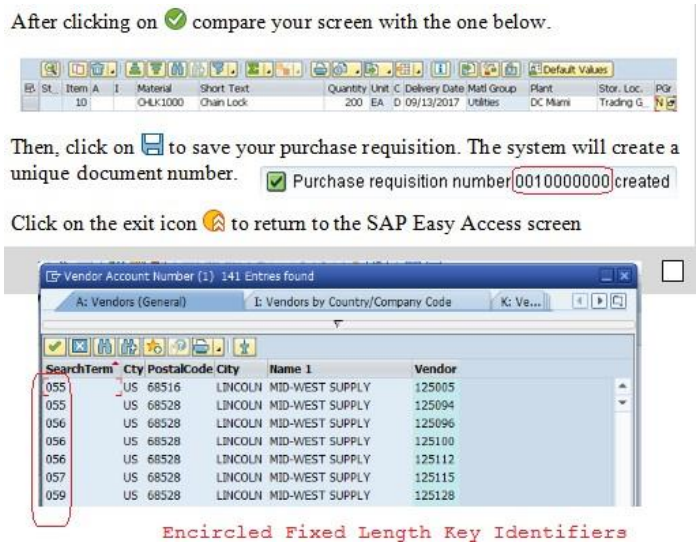
Primary key Instances:

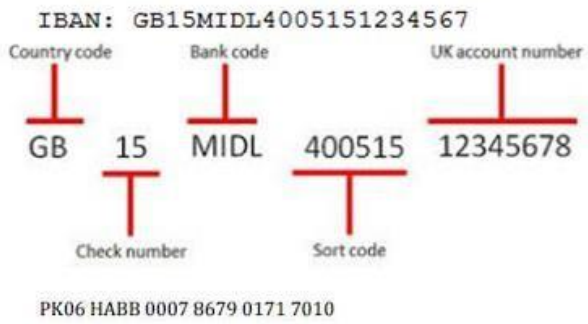
As we all know, different organizations utilize various primary keys to express their views. For example,

- 512235665 Social Security Number
- 610112654 (By Birth Asian, European, Homeland)
- 200338900 Student ID (With Registration year)
- LHR051 Product ID (with Manufacturing Loc
- 31266885 Bank Acc# (followed by Branch Code)
- 005384 Company has many branch codes 005
- 007004 invoices issued from branches

Primary key Fixed Length:

Another example of fixed length primary key:





PK06 HABB 0007 8679 0171 7010

Primary key – Indexing:

Indexing is a way to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed. It is a data structure technique which is used to quickly locate and access the data in a database. DBMS automatically set the indexing on primary key.

```
Table: PATIENT_VISIT
Index: Yes, PK
Test Query execution time:less than 1 second

Table: PATIENT_VISIT1
Index: NO
Test Query execution time:12 seconds

SQL> select count(*) Total_Records
  2  from PATIENT_VISIT;

  Total_Records
---------------
     625721

Elapsed: 00:00:00.04
```

```
SQL> select PATIENT_ID, max(PATIENT_VISITNO)
  2  from PATIENT_VISIT1
  3  where PATIENT_VISITNO = (select max(PATIENT_VISITNO)
  4     from PATIENT_VISIT1
  5     where DEPEND_SNO='0')
  6  group by PATIENT_ID
  7  having count(*)>=1;

PATIENT_ID MAX(PATIENT_VISITNO)
---------- --------------------
9999993                    1316

Elapsed: 00:00:11.54
```

```
  PRIMARY KEY ( PATIENT_ID, DEPEND_SNO, PATIENT_VISITNO )
```

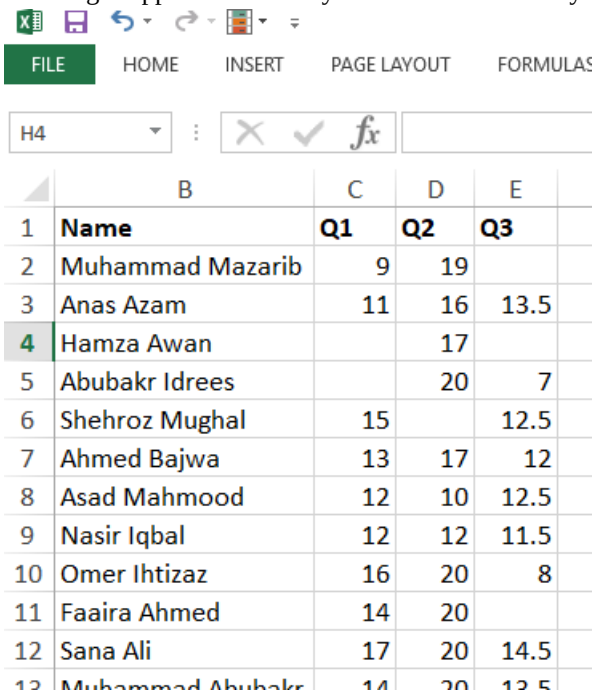
Lesson 14: Primary Keys Formats

In this topic we have discuss different formats and styles of the primary key.

Basics of Indexing:

- Paging a key's data to access fast.
- It may include sorting from A to Z or Z to A.

- Indexing is applied on non-key columns where mostly search is required

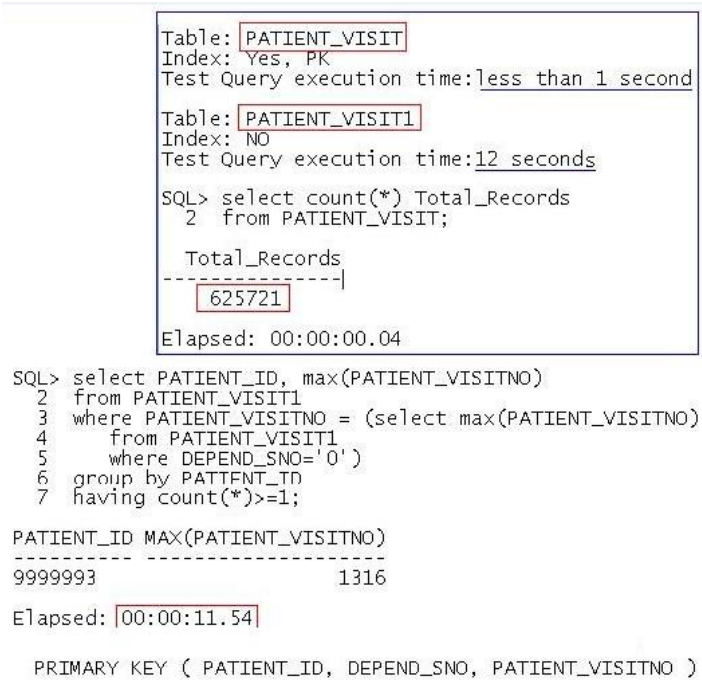


FILE HOME INSERT PAGE LAYOUT FORMULAS

H4

	B	C	D	E
1	**Name**	**Q1**	**Q2**	**Q3**
2	Muhammad Mazarib	9	19	
3	Anas Azam	11	16	13.5
4	Hamza Awan		17	
5	Abubakr Idrees		20	7
6	Shehroz Mughal	15		12.5
7	Ahmed Bajwa	13	17	12
8	Asad Mahmood	12	10	12.5
9	Nasir Iqbal	12	12	11.5
10	Omer Ihtizaz	16	20	8
11	Faaira Ahmed	14	20	
12	Sana Ali	17	20	14.5

Primary key – Indexing on DBMS:



```
Table: PATIENT_VISIT
Index: Yes, PK
Test Query execution time:less than 1 second

Table: PATIENT_VISIT1
Index: NO
Test Query execution time:12 seconds

SQL> select count(*) Total_Records
  2  from PATIENT_VISIT;

  Total_Records
---------------
     625721

Elapsed: 00:00:00.04
```

```
SQL> select PATIENT_ID, max(PATIENT_VISITNO)
  2  from PATIENT_VISIT1
  3  where PATIENT_VISITNO = (select max(PATIENT_VISITNO)
  4     from PATIENT_VISIT1
  5     where DEPEND_SNO='0')
  6  group by PATIENT_ID
  7  having count(*)>=1;

PATIENT_ID MAX(PATIENT_VISITNO)
---------- --------------------
9999993                    1316

Elapsed: 00:00:11.54
```

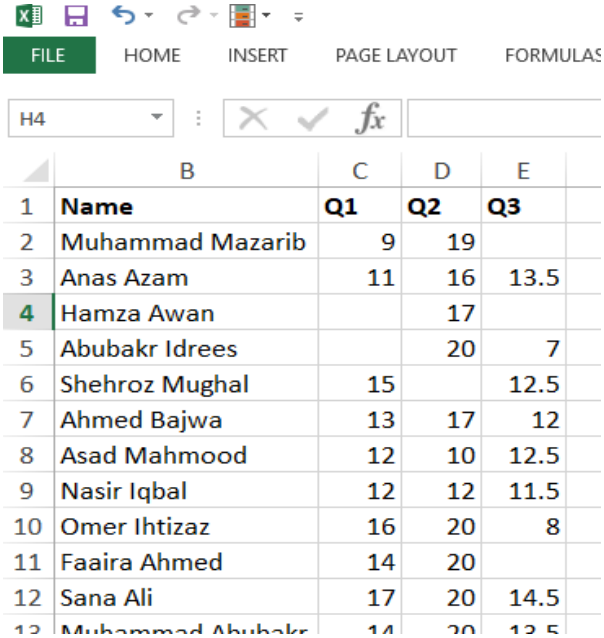
```
  PRIMARY KEY ( PATIENT_ID, DEPEND_SNO, PATIENT_VISITNO )
```

This figure indicates that when indexing is used, data is accessed more quickly than when indexing is not used.

Primary key – Who will decide?

Roll# is issued by some University or college in which these students are studying.

Format or instance (Shape of Roll#), meaningful column for PK



	B	C	D	E
1	**Name**	**Q1**	**Q2**	**Q3**
2	Muhammad Mazarib	9	19	
3	Anas Azam	11	16	13.5
4	Hamza Awan		17	
5	Abubakr Idrees		20	7
6	Shehroz Mughal	15		12.5
7	Ahmed Bajwa	13	17	12
8	Asad Mahmood	12	10	12.5
9	Nasir Iqbal	12	12	11.5
10	Omer Ihtizaz	16	20	8
11	Faaira Ahmed	14	20	
12	Sana Ali	17	20	14.5

Roll# needs to be introduced by some university or college (Enterprise) in which these students are studying

It is not wise Roll# is a serial number because there are many levels, classes and courses.

PK being Roll# should have some format of instance or meaning rather than serial number.

Lesson 15: Surrogated Keys Examples - I

Surrogated Key Definition:

A surrogate key as an artificial column added to a relation to serve as a primary key:

- DBMS supplied
- Short, numeric and never changes – an ideal primary key!
- Has artificial values that are meaningless to users
- Normally hidden in forms and reports

Example

- RENTAL_PROPERTY without surrogate key:

  RENTAL_PROPERTY
  (Street, City, State/Province, Zip/PostalCode, Country, Rental_Rate)

- RENTAL_PROPERTY with surrogate key:

  RENTAL_PROPERTY (PropertyID, Street, City, State/Province, Zip/PostalCode, Country, Rental_Rate)

Exercises

Lets solve following examples

- When to put up a complaint by using phone?

  ComplaintNo will be a surrogated key

- ComplaintRef# is normally generated
- Needs attributes?

Lesson 16: Surrogated Keys Examples – II

Surrogated Keys Examples:

Example 1: ATMTransaction

(Card#, Serial#, Amount, DrawDate)

- Need more attributes

  Machine#, Type, Status, LimitBalance

Example 2: InsurancePaid

(Policy#, PaidYear, PaidDate, Amount)

- Need more attributes

  DueDate, DueAmount, Balance, Charges

Normally insurance is paid every year in advance. Therefore, paid year is a well-defined artificial key because there is a standard criteria Insurance is managed yearly basis.

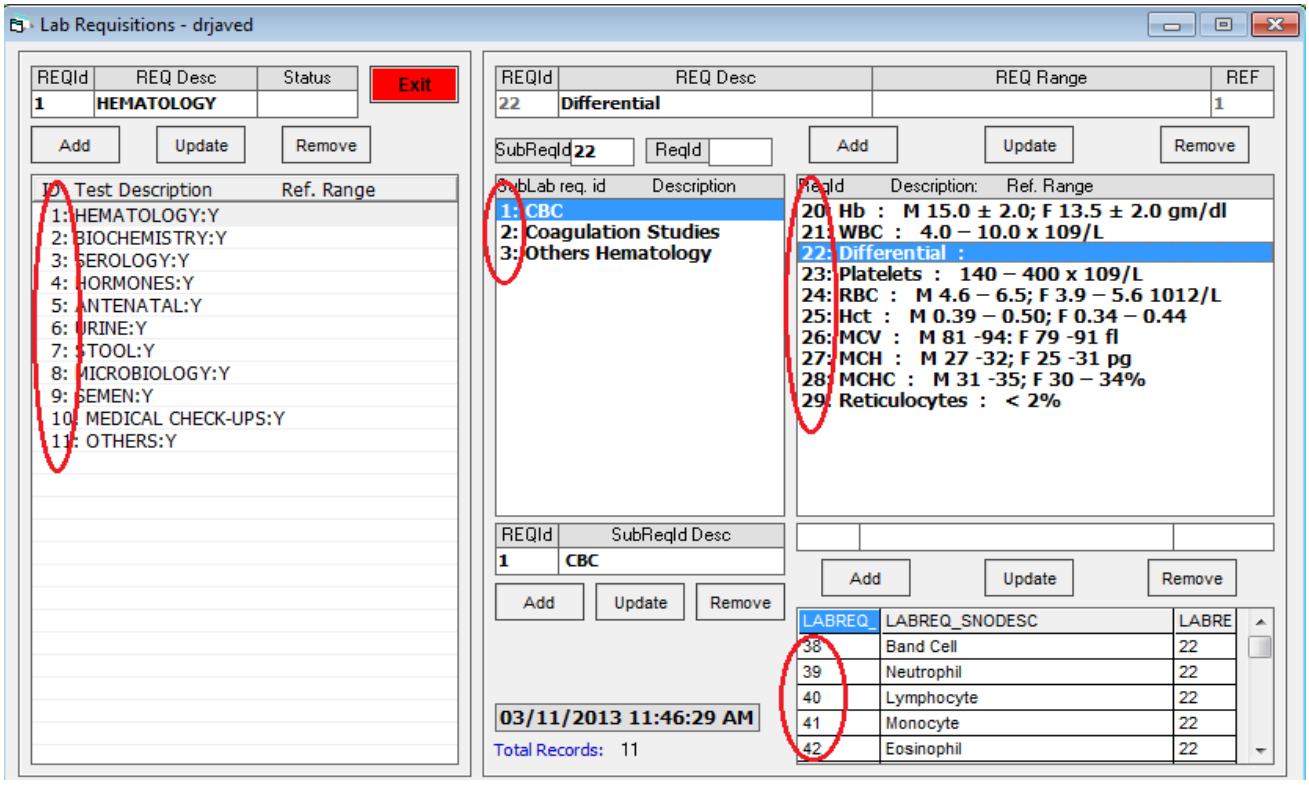
Keys with order in columns:

PatientID	Visit#	Visit Date
558741	1	...
558741	2	
558741	3	
...		
589210	1	Better Solution
589210	2	using composite keys
599211	1	
599211	2	
...		

PatientID	Visit#	Visit Date
558741	1	...
558741	2	
558741	3	
589210	4	
589210	5	
599211	6	
599211	7	
...	...	
	830000	and more

The importance of arranging the keys of columns is shown in this figure.

In continuous visit# for all patients will be hard to remember when it is greater than 100.



Lab results catalogue

Lesson 17: Surrogated Keys Examples - III

Solving Example-1

Now we must choose if invoice# should be handled as a surrogate key or not. (When we are buying something from the cash and carry store).

Let us choose

Invoice, when buying some items

- Surrogate key is called the fact less key as it is added just for our ease of identification of unique values and contains no relevant fact (or information) that is useful for the table.
- It contains unique value for all records of the table.

Solving Example-2

Let us discuss

- We visit some wedding hall to book some event.
- How staff of hall will store event's details?
    -
    - Collect all the necessary information about an event (book by, client name, and payment method etc.)
- Which columns are required?

Lesson 18: Surrogated Keys Examples - IV

In this topic we learned how to analyze the surrogated key.

Solving Example-1 (Saving blogs)

Discuss about Facebook, blogs and twitter. If we want to store data in database, how to decide which id should be given.

Columns [date, location, ID, Desc]

In this example ID is a surrogated key.

Solving Example-2

- Let's us assume a loan of Rs. 50,000
- How to decide which columns are required to fill up to keep track record of installments?
    - Columns that we need paid amount, due amount, balance, paid date, due date, penalty and status.

Who took loan→identified by some Loan#, then paid date, paid amount, due date, due amount, balance

Comparisons of keys

Let us discuss

What is difference between PK & surrogated key?

The main difference between surrogate key and primary key is that surrogate key is a type of primary key that helps to identify each record uniquely, while the primary key is a set of minimal columns that helps to identify each record uniquely.

What is difference between PK & unique key?

The primary key is the minimal set of attributes which uniquely identifies any row of a table. The primary key cannot have a NULL value. It cannot have duplicate values.

Unique Key is a key which has a unique value and is used to prevent duplicate values in a column. A unique key can have a NULL value which is not allowed in a primary key.

Give three more examples of surrogated keys from real-life.

1.
    a. System date
    b. time stamp
    c. Random alphanumeric string

Lesson 19: Foreign Keys Examples - I

Definition:

A foreign key is an attribute that refers to a primary key of same or different relation to form a link (constraint) between the relations:

- A foreign key can be a single column or a composite key
- The term refers to the fact that key values are foreign to the relation in which they appear as foreign key values
- Ideally Data type, Length/ Size of FK and referring PK must be same

Example-1

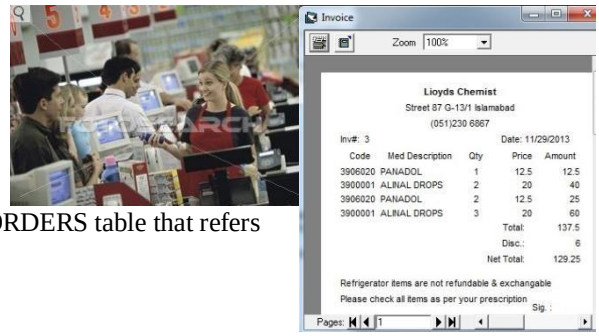
DEPARTMENT (DeptID, DepartmentName, BudgetCode, ManagerName)

EMPLOYEE (EmpNumber, EmpName, *DeptID*)

In this example *DeptID* is referred to the foreign key in Employee table that refers to a primary key in Department table.

Name of FK column may be different from the name of referencing PK

Example-2



CUSTOMER(CustId, Name, Address, Telno, Email)

ORDERS(Invoice#, InvType, InvDate, CustomerId)

In this example *CustID* is referred to the foreign key in ORDERS table that refers to a primary key in CUSTOMER table.

Does Invoice# always contain serial numbers?

Lesson 20: Foreign Keys Examples - II

In this topic, we have to discuss more examples of foreign key. In the last topic, we covered the characteristics of a foreign key.

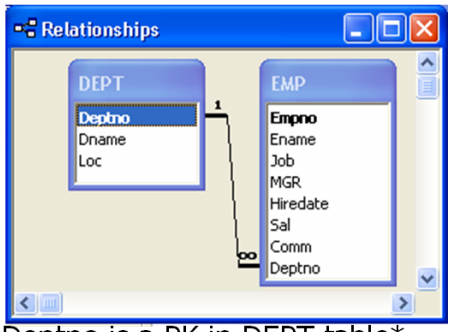
1. Foreign key can be NULL.
2. Foreign key can be repeatable.
3. Using a foreign key, create a relationship between two tables.

Relationship details:

Figure represent the basic meaning of relationship details.

FK & Recursive Relationships:

A relationship between two entities of a similar entity type is called a recursive relationship. Here the same entity type participates more than once in a relationship type with a different role for each instance. In other words, a relationship has always been between occurrences in two different entities. However, the same entity can participate in the relationship. This is termed a recursive relationship.



Deptno is a PK in DEPT table*

Empno is a PK in EMP table

Deptno is a FK in EMP table*

(it references to an attribute Deptno of DEPT table)

NOTE: An attribute within one relation that matches the key (Primary/same) in the same relation, is called recursive relation

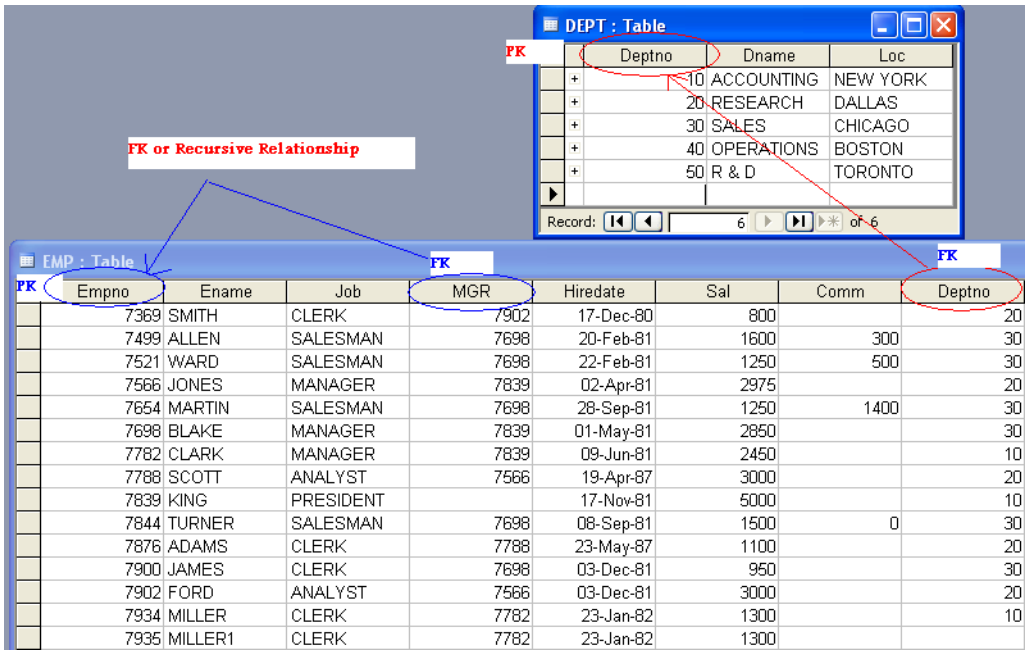
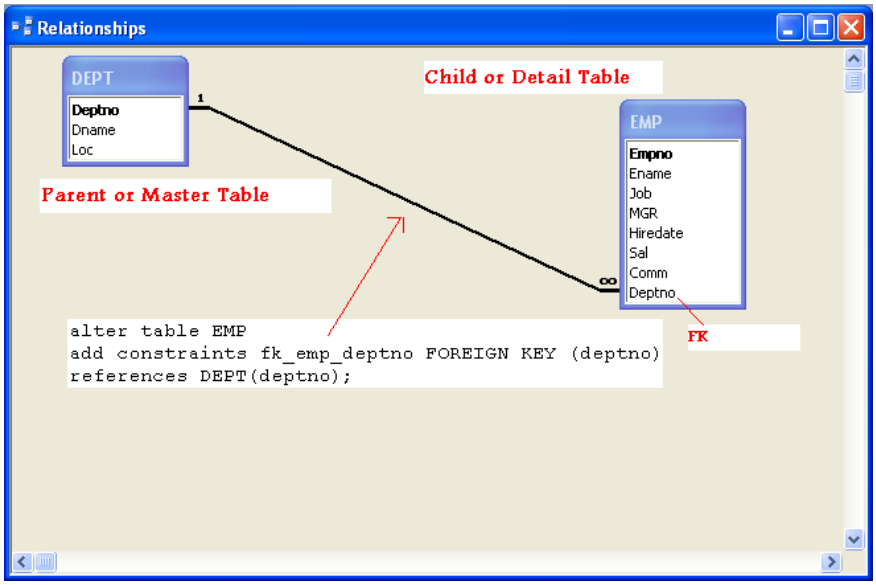


Figure represent the significance of recursive relationship.

Logical link between tables



Lesson 21: Foreign Keys Examples - III

Foreign Keys (FKs) Examples:

1. FK is a part of composite key

NOTE: PK column is underlined

ITEM (Item#, Item_Name, Department, Buyer)

ORDER_ITEM (OrderNumber, *Item#*, Quantity, Price, ExtendedPrice)

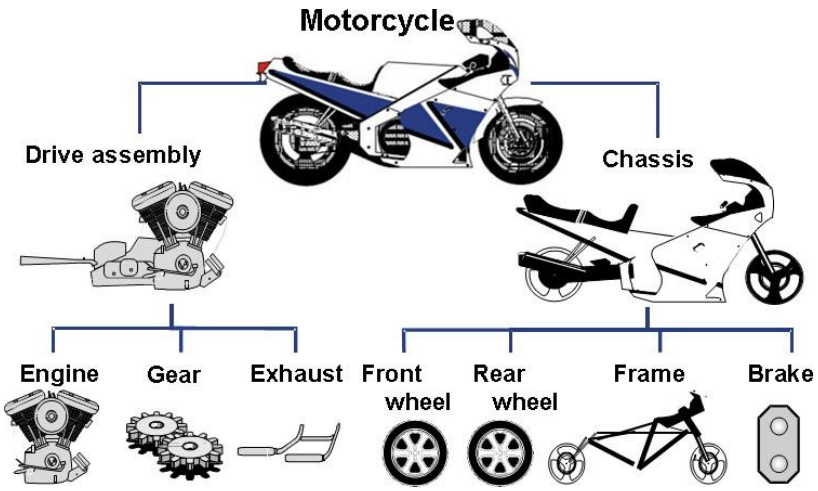
Where ORDER_ITEM.ITEM# must exist in ITEM.ITEM#

OrderNumber is an alternate name of InvoiceNumber

2. **Referential Integrity Constraint**

   A **referential integrity constraint** is a statement that limits the values of the foreign key to those already existing as primary key values in the corresponding relation

3. **Recursive Relationship**



Part (Part#, PName, Weight, MakeYear, Price, PType, *RefPart#*)

PType: Assembled product/ parts (e.g., Motorcycle) or single part

Lesson 22: Foreign Keys Examples – IV

Foreign Keys Examples

- Integrity rules
- Referential with cascade
- Integrity Example

Integrity Rules

Entity integrity

It specifies that:

- No two rows with the same primary key value
- No null values in a primary key

Detail:

The entity integrity constraint states that no primary key value can be NULL. This is because the primary key value is used to identify individual tuples in a relation. Having NULL values for the primary key implies that we cannot identify some tuples. For example, if two or more tuples had NULL for their primary keys, we may not be able to distinguish them if we try to reference them from other relations. Key constraints and entity integrity constraints are specified on individual relations. (Reference: **Database Systems (FDS),** by Ramez Elmasri and Shamkant Navathe, Addison Wesley, 6th Edition.)

Referential integrity

Foreign keys must match candidate key of source table

Foreign keys in some cases can be null

The database must not contain any unmatched foreign key values.

If B references A, A must exist.

Detail:

The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations. Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation. (Reference: **Database Systems (FDS),** by Ramez Elmasri and Shamkant Navathe, Addison Wesley, 6th Edition.)

Referential with cascade

When we update a tuple from a table, say S, that is *referenced* by another table, say SP,

There are similar choices of *referential actions for update and delete*:

ON UPDATE CASCADE

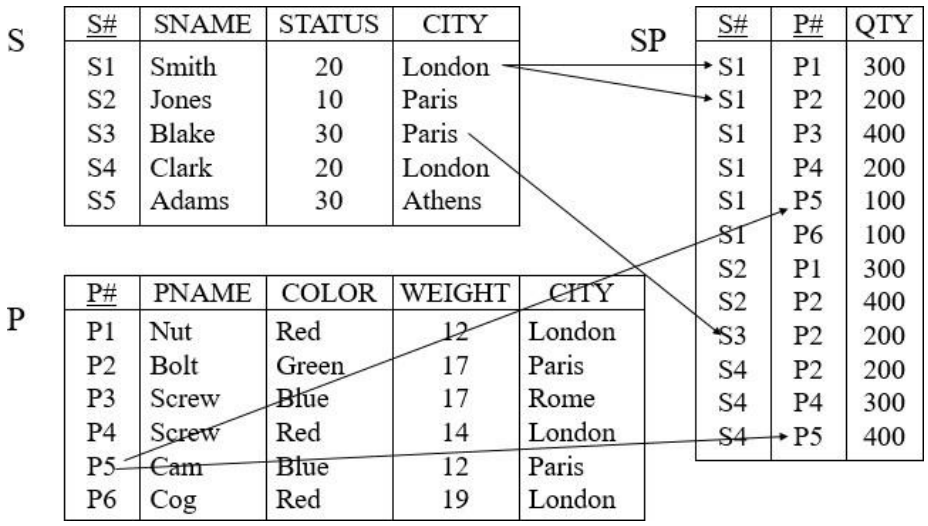
ON UPDATE RESTRICT

There could be other choices besides these three. , e.g.,

ON UPDATE SET DEFAULT.

ON UPDATE SET NULL.

- Integrity Example



Lesson 23: Composite Keys Examples - I

Composite Keys

Definition of Composite Key

It is combination of Primary Keys (PK) in a relation of selected attributes gives the concept of composite key. Composite key is an extended form of a primary key. All the characteristics of PK applies on composite Keys comprising of more than one column.

Basic Example of Composite Keys

During ATM Transaction, amounts can be drawn several times on one ATM Card.

We need follow the issues

How much amount will be withdrawn?

When amount was drawn?

Which machine has been used?

What are the types of transaction?

Card#, Amount, DrawDate, Machine#, TransType are the attributes for transaction

Other Examples of Composite Keys

Can we say Card# will be a primary key? NO

Can DrawDate will be a key with Card#? NO

Then what to do? Add Surrogated key which is TransNo

ATMTransaction(Card#, TransNo, Amount, DrawDate, Machine#, TransType)

Date includes time in seconds as well.

Many transactions are managed on same date.

Lesson 24: Composite Keys Examples - II

Composite Keys Examples

Preparing Dataset for Composite Key

Card(Card#, CardHolderName, ExpiryDate, IssueDate, Acc#)

ATMTransaction(Card#, TransNo, Amount, DrawDate, Machine#, TransType)

Answer following questions.

Do we need most closed related table?

What are PK and FK?

Hospital Related Example

PatientVisit(PatID, VisitSNO, Vdate, DocID, Diagnosis,…)

LabTest(*PatID*, *VisitSNO*, TestID, Tdate, …)

LabTest has FKs *PatID, VisitSNO*, referring to same corresponding composite keys in PatientVisit

Answer following question.

Which lab table needs to be considered?

What about Excluding Composite Key?

Card(Card#, CardHolderName, ExpiryDate, IssueDate, Acc#)

ATMTransaction(ID, Card#, TransNo, Amount, DrawDate, Machine#, TransType)

Answer following questions.

Do we need most closed related table?

What is a drawback of using ID as Surrogated Key in ATMTransaction?

What are PK and FK?

Lesson 25: Composite Keys Examples - III

Composite Keys Examples

Course Offering Example

Let's us assume, there are 1000 number of courses in a table or list.

Students want to register in courses.

Can they register by looking at 1000 courses? NO

Answer following questions.

What to do?

Offer selected courses?

Course Offering Example

Offering course table

CrsOffer(SemID, CrsID, Sec, InstrName, B#, R#)

What to do for keys, PK or Composite Key?

CrsOffer(SemID, CrsID, Sec, InstrName, B#, R#)

CrsReg(SemID, Roll#, CrsID, Sec, TotMarks, Grade)

Preparing Dataset for Composite Keys

CrsOffer(SemID, CrsID, Sec, InstrName, Building#, Room#)

CrsReg(SemID, Roll#, CrsID, Sec, TotMarks, Grade)

Answer following questions.

Do we need most closed related tables?

What are PK and FK?

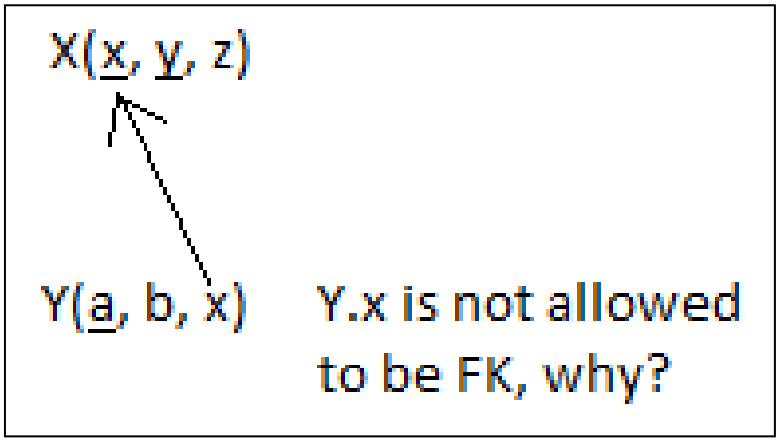
Lesson 26: Composite Keys Examples - IV

Composite Keys Examples

- Meaningful learning-1
- Meaningful learning-2
- Wrong composite key
- Indexing columns

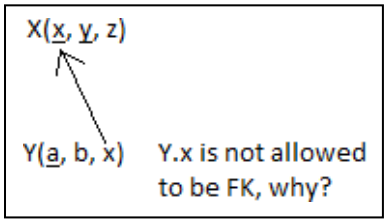
Meaningful learning-1

Draw dataset instance values



Meaningful learning-2

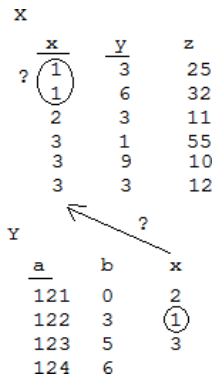
- Analysis and Semantics
- Evaluations



Semantics: Creating two tables with relevant data

Analysis: Y.x=1 cannot be mapped exactly with X.x records 1, 3 or 1, 6

Evaluations: Default FK Y.x cannot be created. Trigger can be created on Y.x to make sure Y.x will be a subset of X.x. Otherwise, data can be put in Y.x which does not belong to or mapped to X.x.



Wrong composite key

Draw dataset instance values

Indexing columns

PATIENT_VISIT Without Indexing

PATIENT_ID	DEPEND_SN	PATIENT_VIS	EMP_ID	DOCTOR_SN	VISIT_DATE	VISIT_NOTE
0222082	0	13	6772356	35	12/27/2005 1:36:08 PM	
0257421	0	2	7033244	102	12/27/2005 3:23:10 PM	
1840190	6	7	6850269	93	12/27/2005 2:22:52 PM	
1976193	0	2	6941084	1	7/10/2005 6:21:03 AM	
6901319	0	1	6891560	1	7/10/2005 7:34:16 AM	
6900626	0	1	6941084	8	7/10/2005 7:36:33 AM	next week
6911102	0	1	6941084	10	7/10/2005 7:54:07 AM	
0213919	0	1	7024037	4	7/12/2005 10:32:27 AM	
1731899	0	1	6891560	10	7/12/2005 9:31:15 AM	
6781521	8	2	6910261	9	7/12/2005 10:37:33 AM	
RS00079	2	1	6941084	11	7/12/2005 8:07:24 AM	
0207842	0	1	6772356	10	7/12/2005 10:41:52 AM	
6781521	0	1	6910261	10	7/12/2005 10:45:19 AM	
1770371	0	1	7012636	13	7/12/2005 10:48:20 AM	
0227638	0	1	6971354	11	7/12/2005 10:49:22 AM	
1926081	1	2	6941084	12	7/12/2005 8:22:36 AM	
6751368	2	3	6900551	9	7/12/2005 10:49:26 AM	

PATIENT_VISIT Keys/ Indexed

PATIENT_ID	DEPEND_SN	PATIENT_VIS	EMP_ID	DOCTOR_SN	VISIT_DATE	VISIT_N
0204531	0	15	6772356	28	10/3/2007 12:15:51 PM	
0204531	0	16	7073498	19	4/29/2008 1:40:11 PM	
0204531	0	17	6772356	37	4/29/2008 2:09:45 PM	
0204531	0	18	7024037	16	5/31/2008 9:20:18 AM	
0204531	0	19	7024037	4	8/5/2008 1:00:34 PM	
0204531	0	20	6772356	15	8/9/2008 8:36:01 AM	
0204531	0	21	7024037	4	8/11/2008 8:38:44 AM	
0204531	0	22	7073498	16	8/25/2008 1:01:39 PM	
0204531	0	23	6772356	7	10/26/2008 2:32:56 PM	
0204531	0	24	6772356	7	10/28/2008 8:56:20 AM	
0204531	0	25	7024037	39	11/10/2008 1:27:23 PM	
0204532	0	1	6850269	30	9/11/2005 7:17:21 PM	
0204532	0	3	7024037	50	10/5/2005 2:13:42 PM	
0204532	0	4	6850269	50	12/14/2005 2:46:16 PM	
0204532	0	5	6850269	59	1/24/2006 9:41:49 PM	
0204532	1	1	6850269	41	8/6/2005 2:55:29 PM	
0204532	1	2	7043508	31	9/6/2005 2:59:43 PM	
0204532	1	3	7043508	27	9/7/2005 3:25:09 PM	
0204532	1	4	6900551	12	9/8/2005 11:19:55 AM	
0204532	1	6	6971354	12	9/12/2005 11:12:52 AM	
0204532	1	7	6850269	57	1/24/2006 9:27:27 PM	
0204532	1	8	7024037	26	2/7/2006 2:56:09 PM	
0204532	2	1	7033244	3	10/22/2005 3:34:27 PM	
0204532	2	2	6971354	44	1/18/2006 2:54:51 PM	
0204573	0	1	7024037	35	1/21/2006 11:02:11 AM	

Lesson 27: Additional or Other Constraints

Other constraints

- What are additional or other constraints?
- Null
- Domain constraint

What are additional or other constraints?

Other Constraints means additional constraints in addition to the standard constraints PK, FK, Composite Key, Surrogated key

For example, NOT NULL, CHECK, DEFAULT, UNIQUE etc.

Null

Represents a value of an attribute that is currently unknown or is no applicable for this tuple, means nothing.

Cannot compute anything with the Null

Domain constraint

Specifies that the value of attribute A must be an atomic value from the domain DOM (A). E.g., Width of column or Check on column.

Domain constraints must be Atomic means no two mobile numbers saved in one cell of a domain. Size of column must be appropriate.

S.	RollNo	Name	Total:	20	20	10
30	18I-0599	Saifullah Dar		16	16	8
31	18I-0645	Muzammil Shakir		15	10	7.5

Lesson 28: Additional Constraints using tables

Other constraints

- CHECK constraint
- Unique & Default constraints
- Composite unique constraints
- Example of Trigger

CHECK constraints

The **check** clause in SQL permits domains to be restricted:

Use **check** clause to ensure that an hourly-wage domain allows only values greater than a specified value.

Create domain hourly-wage **numeric(5,2)**
constraint value-test **check**(value > = 4.00)

The domain hourly-wage is declared to be a decimal number with 5 digits, 2 of which are after the decimal point

create table account
(branch-name Varchar(15),
acc-number Char(10) **not null**,
balance integer,
……)

alter table account

add constraint ck1 check (bal > 0);

alter table emp

add constraint ck1_deptno

check (deptno in (10,20,30));

UNIQUE & Default constraints

create table Dependent

(empno CHAR(6),

SNO Number(2),

DepName varchar2(25) NOT NULL,

DOB Date,

BloodGroup VARCHAR2(5),

RelType CHAR(1) Default 'W');

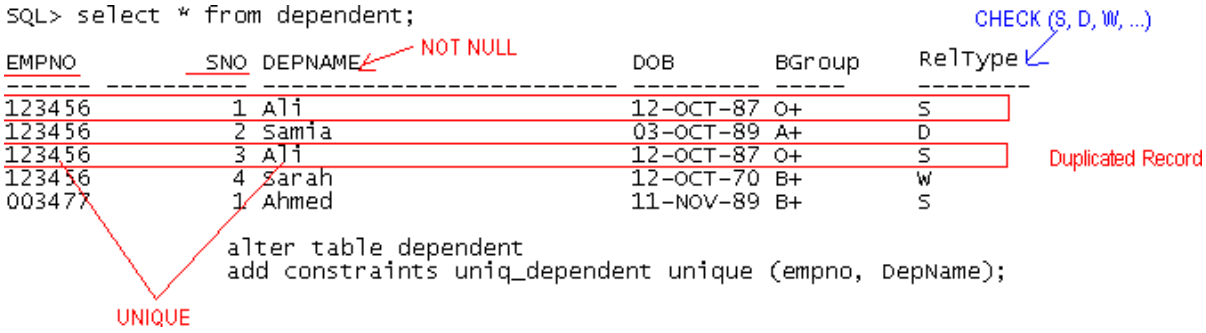
alter table dependent

add constraints pk_dep_empnoSNO primary key (empno, sno);

alter table dependent

add constraints uniq_dependent unique (empno, DepName);

Composite UNIQUE constraints



HECK (Max_Participants>=Enrolled_Participants)

CHECK (Deptno>0)

CHECK (RelType IN ('S', 'D', 'W'))

Default 'Y'

Database Triggers

Example of Trigger

```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON EMP
FOR EACH ROW
WHEN (NEW.EMPNO > 0)
DECLARE
   sal_diff number;
BEGIN
   sal_diff := :NEW.sal  - :OLD.sal;
   dbms_output.put_line('Old salary: ' || :OLD.sal);
   dbms_output.put_line('New salary: ' || :NEW.sal);
   dbms_output.put_line('Salary difference: ' ||
sal_diff);
END;
/
```

Lesson 29: Additional Constraints using DDL Alter SQL - I

Constraints with DDL Alter SQL

- PK constraint on table
- FK constraint on table
- Composite unique & PK constraints

Primary key constraint on table

create table customer
(cid char(10),
Name varchar2(100) Not null,
address varchar2(100),
city varchar2(20)

);
alter table customer
add constraints pk_cid primary key(cid);

Foreign key constraint on table

create table orders

(ordNo CHAR(5),

OrdType CHAR(1) not null,

OrdDate Date,

TotalPrice Number(10,2),

cid CHAR(10)

);

alter table orders

add constraints pk_ordNo primary key(ordNo);

alter table orders

add constraints fk_cid foreign key (cid)

references customer(cid); --already created

Composite unique & PK constraints

create table Dependent

(empno CHAR(6),

SNO Number(2),

DepName varchar2(25) NOT NULL,

DOB Date,

BloodGroup VARCHAR2(5),

RelType CHAR(1) Default 'W');

alter table dependent

add constraints pk_dep_empnoSNO primary key (empno, sno);

alter table dependent

add constraints uniq_dependent unique (empno, DepName);

Lesson 30: Additional Constraints using DDL Alter SQL - II

Constraints with DDL Alter SQL

- Alter constraints ADD & Modify
- Alter constraints using CHECK
- Create table with multiple CHECK
- Cascade and drop constraints

Alter constraints Add & Modify

Following constraints can be applied for an existing EMP table:

```
alter table emp
modify
(sal constraint nt_null_emp_sal NOT NULL);
alter table emp
add (constraint ck1_emp check (deptno>0));
```

Alter constraints using CHECK

Following constraints can be applied for an existing EMP table:

```
alter table emp
add (constraint ck2_emp check
(sal + NVL(comm,0)<=6000));
--Null is allowed, if column is not null
alter table emp
add constraint ck1_emp check
(deptno in (10,20,30));
```

Create table with multiple CHECK

```
create table AirTraveller
(TicketNo CHAR(10),
FirstName VARCHAR2(20),
FamilyName VARCHAR2(20) NOT NULL,
Nationality VARCHAR2(25),
BirthDate DATE,
FlyCode     Number,
CONSTRAINTS ch_natval CHECK (Nationality IN ('Canadian','UK','Pakistan','Bahrain')),
CONSTRAINTS ch_dob CHECK (BirthDate LIKE '[1-2][0-9][0-9][0-9]'),
```

CONSTRAINTS ch_flycode CHECK (FlyCode between 3 and 9));

Cascade & drop constraints

Cascade drops the FK of child table EMP

emp.deptno references to dept.deptno

Alter table dept

drop constraint pk_deptno cascade;

Lesson 31: Keys & Constraints (Revision)

Constraints Revisions

Standard constraints are

- Super Key
- Candidate Key
- Primary Key (PK)
- Surrogated Key
- Foreign Key (FK)
- Composite Key
- Additional constraints are

Not Null, Check, Unique, Default, DOM Constraints using triggers

Super Key and Candidate Key

Super Key describes concepts to find unique records, can be null & indexed

{Roll#, FullName} gives unique when FullName can be duplicated

{CNIC}, {Barcode} are also a super keys

Candidate key gives fewer records in search.

Candidate keys can be used for searching purposes

{CNIC}, {Roll#} and {MedName} are candidate keys

Primary Key

Gives unique record with conditions NOT NULL, UNIQUE and INDEXED

Fixed length, ownership and not frequently changed, Descriptive/ well defined key

Roll#, Empno, Item#, PatientID etc.

2019004, 16K-1005 etc.

Avoid PKs Price, Quantity, Rates

Surrogated Key

Just like a PK but it is artificial key. Decided by the designer from set of attributes and meaningless to the user.

SerialNo or Card# with TransNo

Try to avoid Column with datatype DATE

Foreign Key

Referencing key and also called referential integrity constraint

Dept(DNO, Dname, Loc)

Emp(Empno, Ename, Sal, DOB, DNO)

Emp.DNO is a FK references to Dept.DNO

FK is also Recursive Key!

Composite Key

Collection of PKs in a relation

PatientVisit(PatID, VisitSNO, Vdate, DocID, Diagnosis,…)

LabTest(*PatID*, *VisitSNO*, TestID, Tdate, …)

Columns with red text are surrogated keys

Additional Constraints

NOT NULL
DOMAIN
CHECK
UNIQUE
DEFAULT and TRIGGER

Lesson 32: Script of Referential Integrity Constraints SQL

Script of Referential Integrity

- Primary key with table
- Primary & Foreign Key with table
- Employee Prepares Order
- Item Table
- Sold Table & FKs

Primary key with Table

Follow up all DDL in a given order of slides

```
create table Customer
(custID CHAR(10),
CustName        VARCHAR2(25) NOT NULL,
Address         VARCHAR2(50),
City            VARCHAR2(50),
TelNo           VARCHAR2(15),
Email           VARCHAR2(50));
Alter table Customer
add constraint PK_Custid primary key (CustID);
```

Primary & Foreign keys

```
Create table Orders
(InvoiceNo      Varchar2(12),
InvType         CHAR(1),
InvDate Date NOT NULL,
TotalPrice      Number(6,2),
CustID          CHAR(8),
SalesEmpID      CHAR(5));
Alter table Orders
add constraint PK_InvNo primary key (InvoiceNO);
Alter table Orders
add constraint FK_Cust_CustID foreign key (custID) references Customer(CustID);
```

Employee Prepares Order

```
create table Employee
(EmpID CHAR(3),
EmpName         VARCHAR2(25) NOT NULL,
Address         VARCHAR2(50),
City            VARCHAR2(50),
TelNo           VARCHAR2(15) UNIQUE,
Email           VARCHAR2(50));
Alter table Employee
add constraint PK_EmpID primary key (EmpID);
Alter table Orders
add constraint FK_Emp_EmpID foreign key (SalesEmpID) references Employee(EmpID);
```

Item Table

```
Create table Item
(ItemID CHAR(5),
ItemName        VARCHAR2(50) NOT NULL,
PPrice          NUMBER(10,2),
TotQty          NUMBER(5),
```

```
Status          CHAR(1)
);
Alter table Item
add constraint PK_ItemID primary key (ItemID);
```

Sold Table & FKs

```
Create table Sold
(ItemID CHAR(5),
InvoiceNo       CHAR(12),
SPrice          NUMBER(10,2) Not Null,
SQty            NUMBER(10,2) Not Null
);
Alter table Sold
add constraint PK_Sold primary key
(ItemID, InvoiceNo); --Composite Key
```

Sold Table & FKs

```
Alter table Sold
add constraint FK_Sold_ItemID
foreign key (ItemID) references Item(ItemID);
Alter table Sold
add constraint FK_Sold_InvoiceNo
foreign key (InvoiceNo) references Invoice(InvoiceNo);
```

Lesson 33: Fundamentals of ER Model

Fundamentals of Entity Relationship (ER) Model

- Semantic Modeling
- Chen ER Model
- System Development & DBA
- Database Development
- Conceptual Model
- Conceptual Design

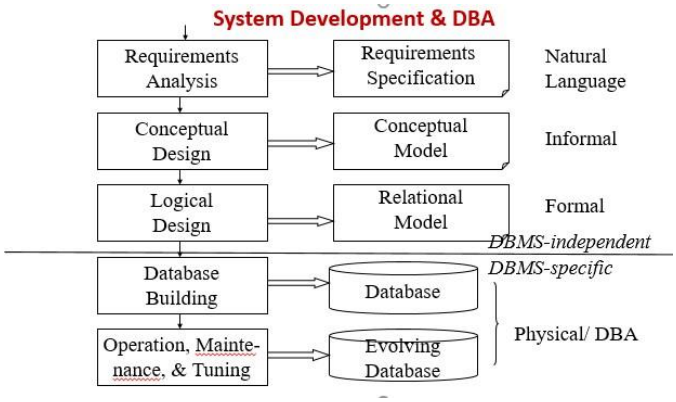
Semantic Modeling

- A class of data models
- Conveys semantic meaning
- Implements databases more intelligently
- Support more sophisticated user interfaces (SQL queries, forms & reports)
- Research started in 1970s and early 1980s
- Other names are data modeling, E/R modeling, E-modeling

Chen ER Model

- Chen in 1976 introduced the E/R model
- The E/R model allows us to sketch the design of a database informally.
- Designs are pictures called entity-relationship diagrams.
- Fairly mechanical ways to convert E/R diagrams to real implementations like relational databases exist.

System Development & DBA



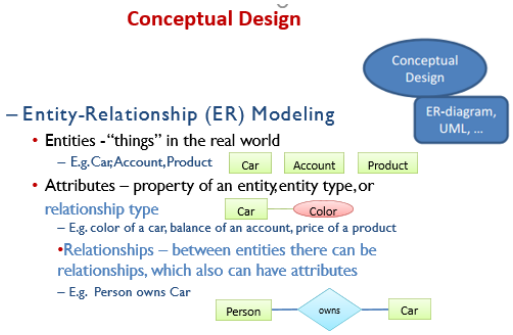
Database Development

- **Requirements Analysis**
  - Collect and Analyze the requirements of the users, e.g., Functional Specification, Prototyping.
- **Conceptual Design**
  - Design a conceptual model (schema), e.g., ER model, ER Mapping.
- **Logical Design**
  - Translate the ER model into a relational model (schema), DDL Scripts.
  - Normalization
- **Database Building**
  - Build the database and write application programs for specific DBMS & S/W Appls/ Tools.
- **Operation, Maintenance & Tuning**
  - Use/ Training, Installations, maintain and "tune" the database.

Conceptual Model

- Highest conceptual grouping of ideas
  - Data tends to naturally cluster with data from the same or similar categories relevant to the organization
- The major relationships between subjects have been defined
  - Least amount of detail

Conceptual Design



Lesson 34: ER Diagram & Attributes displays

Attributes Display

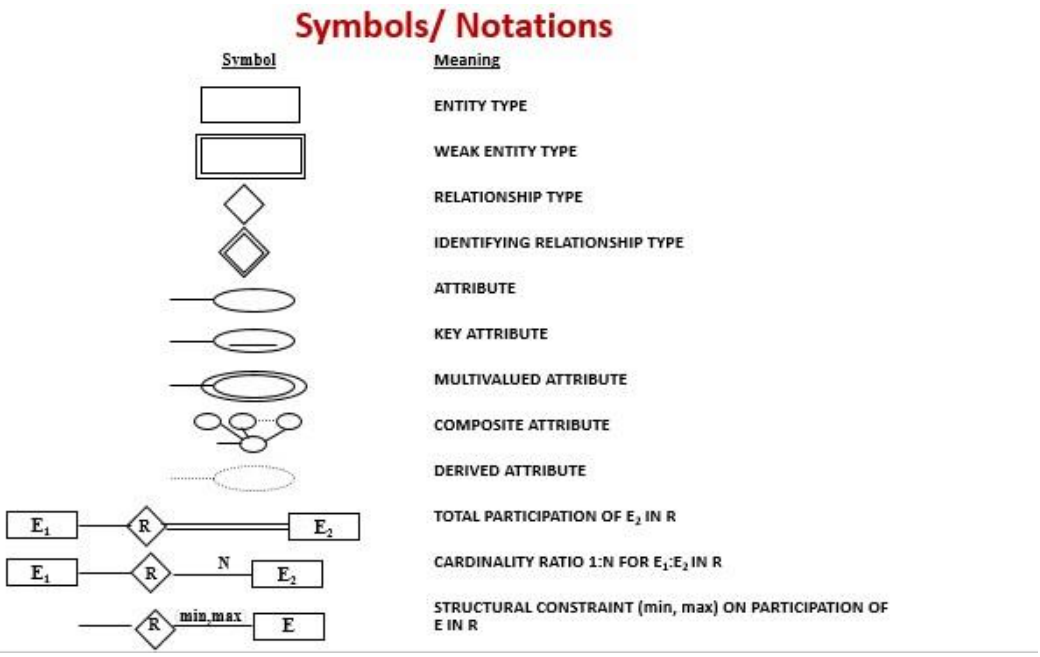
- For Logical Design
- Symbols/ Notations
- Motivational Example
- Entity or Entity Type
- Simple or Single Attributes
- Composite Attributes
- Multi valued Attributes
- Derived or Stored Attributes
- Complex Attributes

For Logical Design

- Translate an ER schema into a relational schema.
  - Entity
  - Relationship
  - Attribute
  - Subtype

Normalization

- To ensure you have a "good" design (Testing of your model).



Motivational Example

Consider building a course management system (**CMS**):

- Students
- Courses
- Professors

Entities

- Who takes what?
- Who teaches what?

Relationships

What about Sales Items System?

Entity or Entity Type

An entity is an object or a concept that is identified by the enterprise as having an independent existence.

There are two types of entities:

- Physical entities
  - Example: car, house, employee, part, supplier, building
- Conceptual entities
  - Example: job, university course, sale, work experience, CourseReg

Simple or Single Attributes

A simple attribute:

- Consists of a single component with an independent existence.
  - Example: The **Sex or Gender** attribute of an EMPLOYEE entity
  - Example: **Blood Group** of a student

Composite Attributes

A composite attribute:

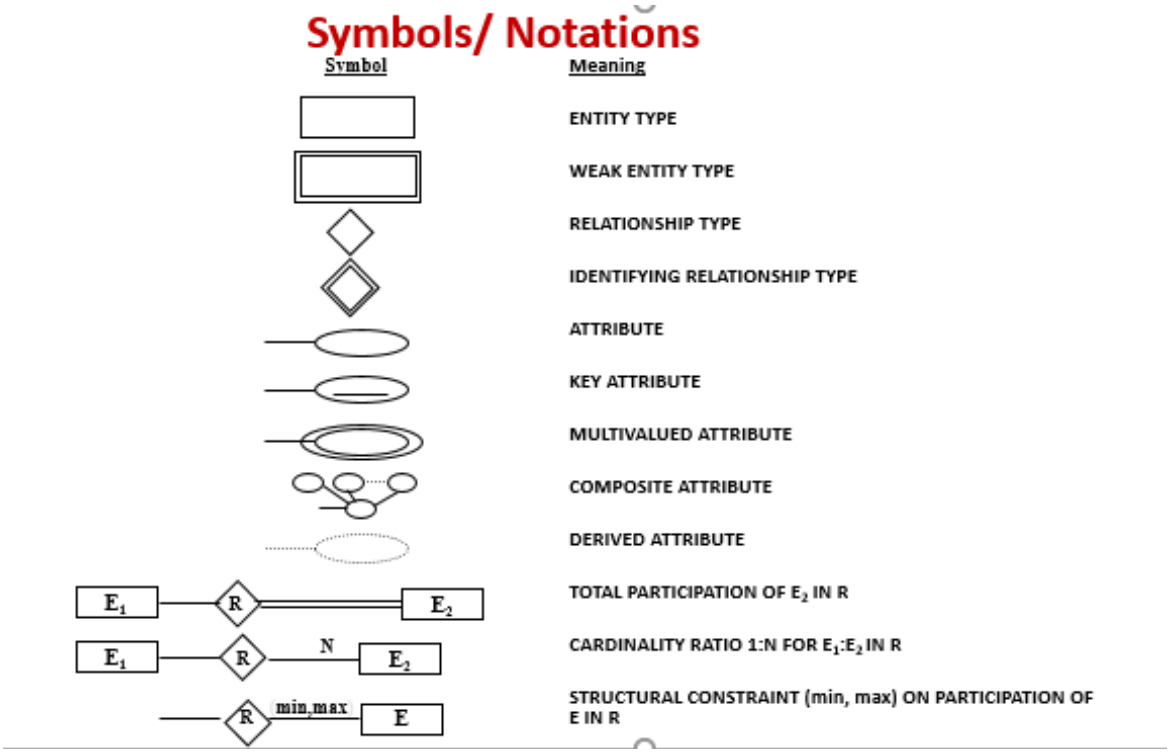
- Consists of multiple components each with an independent existence.
  - Example: The **Address** attribute of an EMPLOYEE entity can be divided into Street, City, State, and Zip.

Lesson 35: Fundamentals Notations of ER Diagram

Attributes Display

- Symbol/ Notations
- Entity Employee
- Entity Department
- Entity Project
- Entities Department & Employee

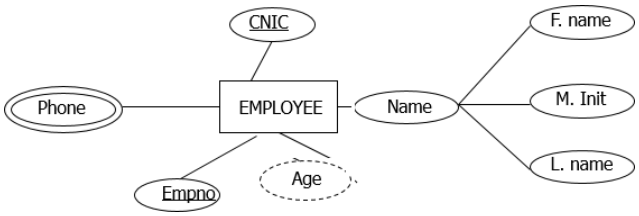
Symbols/ Notations



Entity Employee

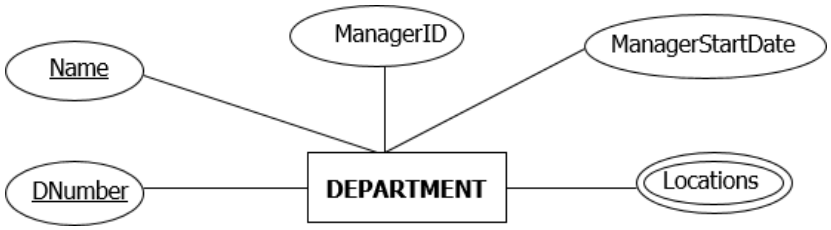
Underlined are Key Identifiers

There is no PK and FK are shown in ER Diagram



Entity Department

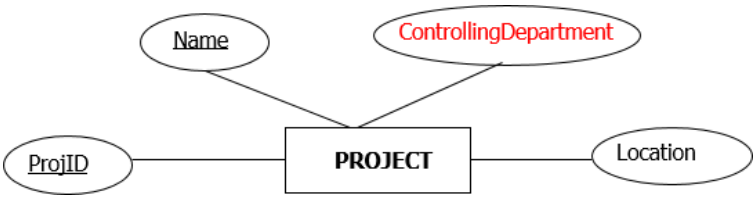
- Underlined are Key Identifiers
- There is no PK and FK are shown in ER Diagram
- Double oval represents multi-valued



Entity Project

An entity type PROJECT has attributes:

- ProjID (key identifier)
- Name (key identifier)
- DNumber (key identifier)
- Location
- ControllingDepartment
- Budget
- Start Date
- End Dates



Entities Department & Employee

Consider Employees are working in departments:

- Department
- Employee
- Manager/ Boss??

Entities

- Employees are belong to Departments
- Some employees are managers of department
- Employee may work in more than one dept
- Some are managers of other employees

Relationships

Lesson 36: ER Diagrams with Relationships One to Many

Attributes Display

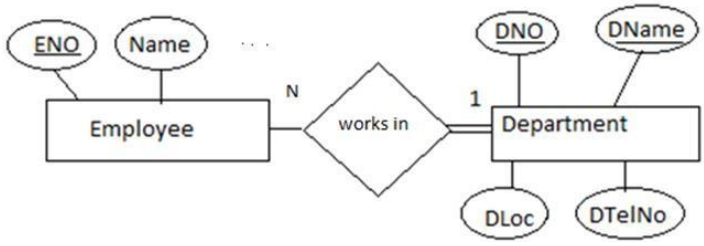
- Example-1 (1-M)
- Example-2 (1-M)
- Example-3 (1-M)
- Example-4 (1-1)
- Exercise

Example-1 (one-many)

- Any Employee has his/ her own department, where works in
- One department has many employees or
- Employee belongs to department

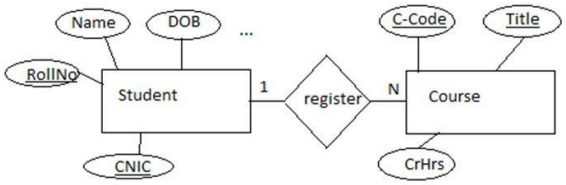
Example-2 (one-many)

- Any Employee has his/ her own department, where works in
- Any one department has at least ONE or many employees
- Employee belongs to department



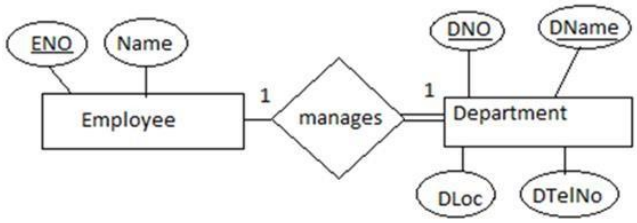
Example-3 (one-many)

- Below diagram is NOT correct.
- Any one student can takes or registers many courses
- But any one course is taken by many students



Example-4 (one-one)

- Some of employees will be a manager of departments.
- Any one employee will be a manager of any one department but at the same time cannot be a manager of multiple departments
- Any one department has one manager as an employee.



Exercise

Combining ER Diagrams of both one to one and one to many Entities are Department & Employee

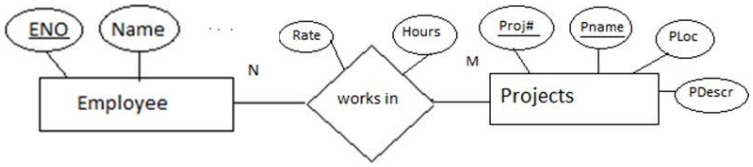
Lesson 37: ER Diagrams with Relationships Many to Many

Attributes Display

- Example-1 (M-N)
- Example-2 (M-N)
- Example-3 (M-N)
- Example-4 (M-N)
- Exercise

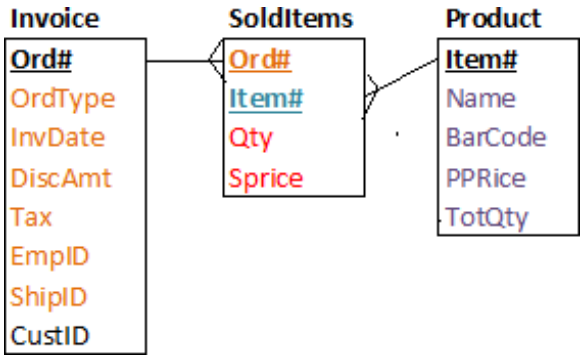
Example-1 (many-many)

- Any Employee has his/ her own department, where works in as full time employee.
- Additionally, Employee works in a Project with number of hours and rate per hour



Example-2 (many-many)

- Customer buys items with invoice.
- Any one invoice contains many products/ items and same product is sold on many invoices.

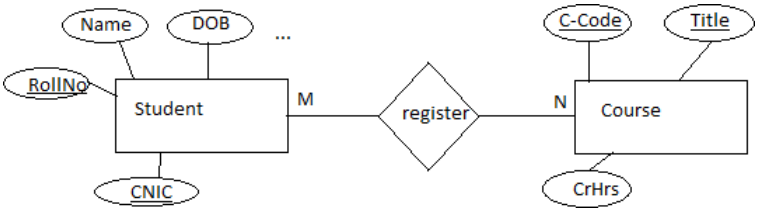


Example-3 (many-many)

- Every course is offered with book as its recommended and reference books.
- Any book can referred in many courses

Example-4 (many-many)

- Below diagram is no longer be valid if student re-enrolled in course again.
- Enrollment requires semester as well.



Exercise

Exercise-4 needs to be drawn with correctness, using semester entity after course offering. Such diagram is called ternary ER relationship.

Lesson 38: ER Diagram with Identifying & Weakness Relationship

Attributes Display

- Identifying & Weakness
- Identifying & Identified entities
- Example-1
- Example-2

Identifying and weakness

- An entity whose existence depends on some other entity type is called a **weak** entity type. Otherwise it is called a **strong** entity type.
- There is a chance of records' duplications on non-key attributes in weak entity types.
- Child entity depends on other parent entity. Parent entity is identifying the child entity.
- Weakness can be reduced only.

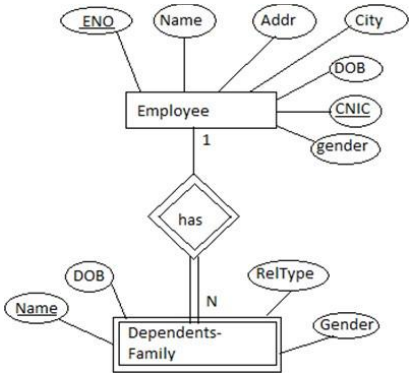
Identifying and identified entities

- All ID-Dependent (identified) entities MAY BE considered weak.
    - Examples are Employee & Family Members ATM Card & Transactions, Patient and Patient visits, Patient & Issue Medicines, Loan and its Installments

Example-1

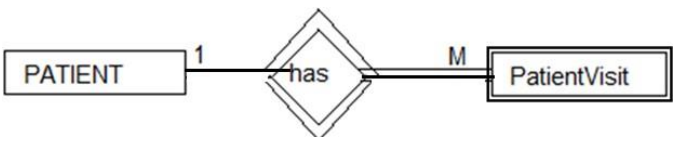
EMPLOYEE(ENO, Name, Gender, DOB, Addr, City, CNIC)

Dependent(ENO, SNO, Name, Gender, DOB, RelType)



Exercise

Scenario: Identifying & Weak relationship



Scenario: Identifying & Strong relationship

Lesson 39: ER Diagrams with the concepts of ER Mapping

ER Diagram with Mapping

- Relational Schema -Implementation Datasets
- Example-1: Mapping 1-1
- Example-2: Mapping 1-N
- Example-3: Mapping M-N
- Example-4: Mapping of Ternary Relationship
- Mapping of Schema

Relational Schema – Implementation PKs, FKs are required

Course

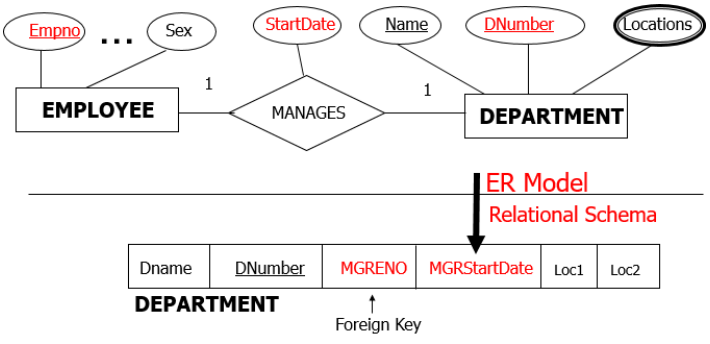
CourseID	Title	CrHrs	Pre-Req
CS203	Database System	3-3-3	CS201
CS201	Data Structure	3-0-3	CS101
CS303	Software Engineering	3-0-0	

CrsOffer

SemID	CourseID	Sec	Instr	B#	Room#	MasStu
F2018	CS201	A	Ali Hassan	2	203	40
F2018	CS201	B	Jamil	3	301	40
S2019	CS203	A	Ejaz	2	203	55
S2019	CS303	A	Jamil	1	101	50
~~S2019~~	~~CS407~~	~~A~~		~~3~~		~~40~~

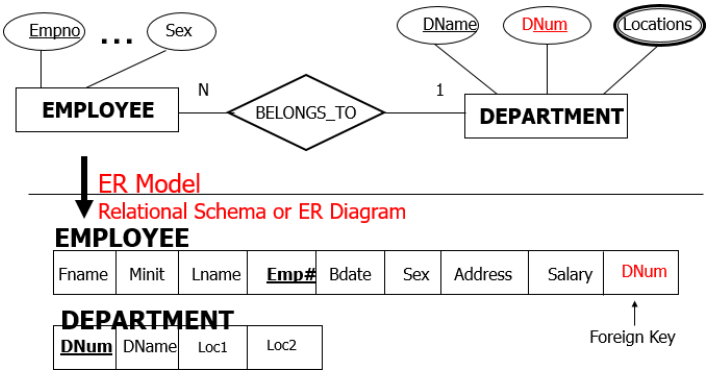
Our ER gives relational schema tables similar to above …
Applying Knowledge – Do able artifacts (Implementation)
Each mapped table will have fewer records
Records should be referenced with the possible variations in tables

Example-1: Mapping 1-1

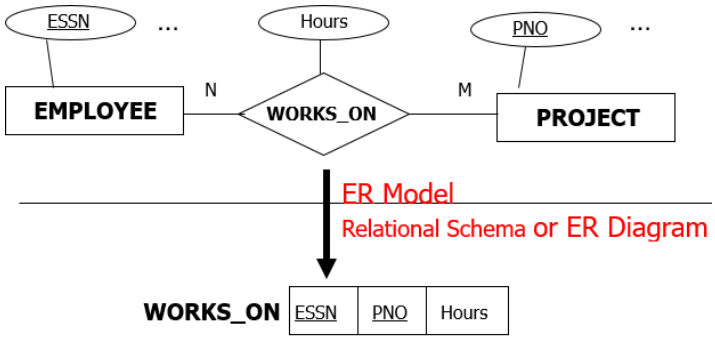


MGRSSN must be UNIQUE KEY to validate 1:1, Dname must has NOT NULL constraint

Example-2: Mapping 1-N



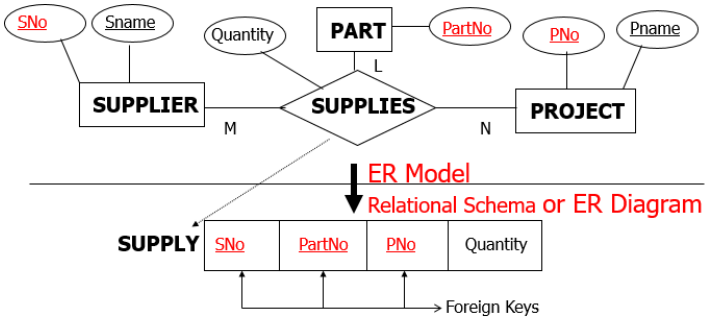
Example-3: Mapping M-N



WORK_ON table requires in addition to tables EMPLOYEE and PROJECT

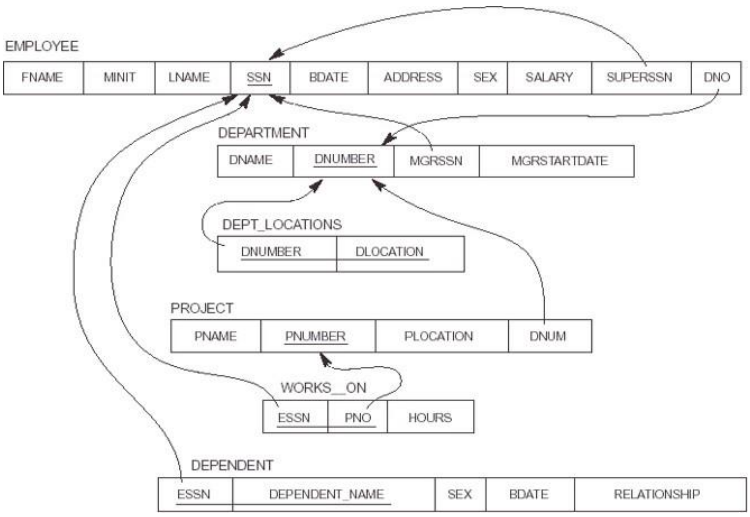
ESSN and PNO are the composite keys, Hours must has NOT NULL constraint

Example-4: Mapping of Ternary Relationship

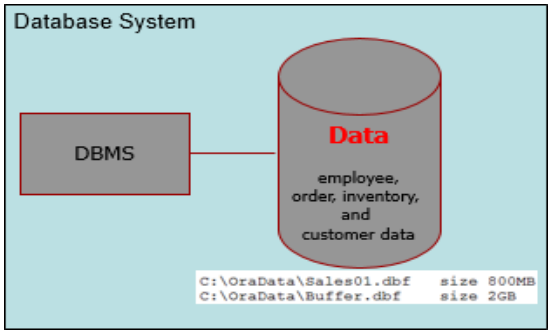


Quantity must has NOT NULL constraint. Similar example can be Lecturer teaches course to students. Other example: Vehicle, Policeman, Ticket (Ticket# is a serial unique number)

Mapping of Schema



Lesson 40: Oracle DBMS



Oracle DBMS

- Definition
- Relational Model
- Relational DB Elements
- How data is organized?
- RDBMS Operations
- Oracle DBMS Quick History

Definition

Database Management System (DBMS) is a software package that controls the storage, organization, and retrieval of data, a DBMS has the following elements:

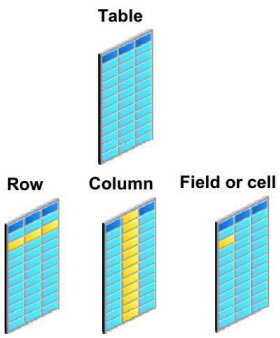
- Kernal code (manages memory and storage)
- Repository of metadata (data dictionary)
- Query Language (accessing data)

Relational Model

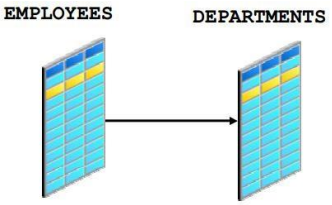
Scientist E. F. Codd defined relational model based on mathematical theory, called relational model.
The relational model has the following major aspects:

- Structures (Well-defined objects store or access the data)
- Operations (data manipulation)
- Integrity rules (constraints or restrictions)

Relational Database Elements



How data is organized?



RDBMS Operations

- **Logical operations**

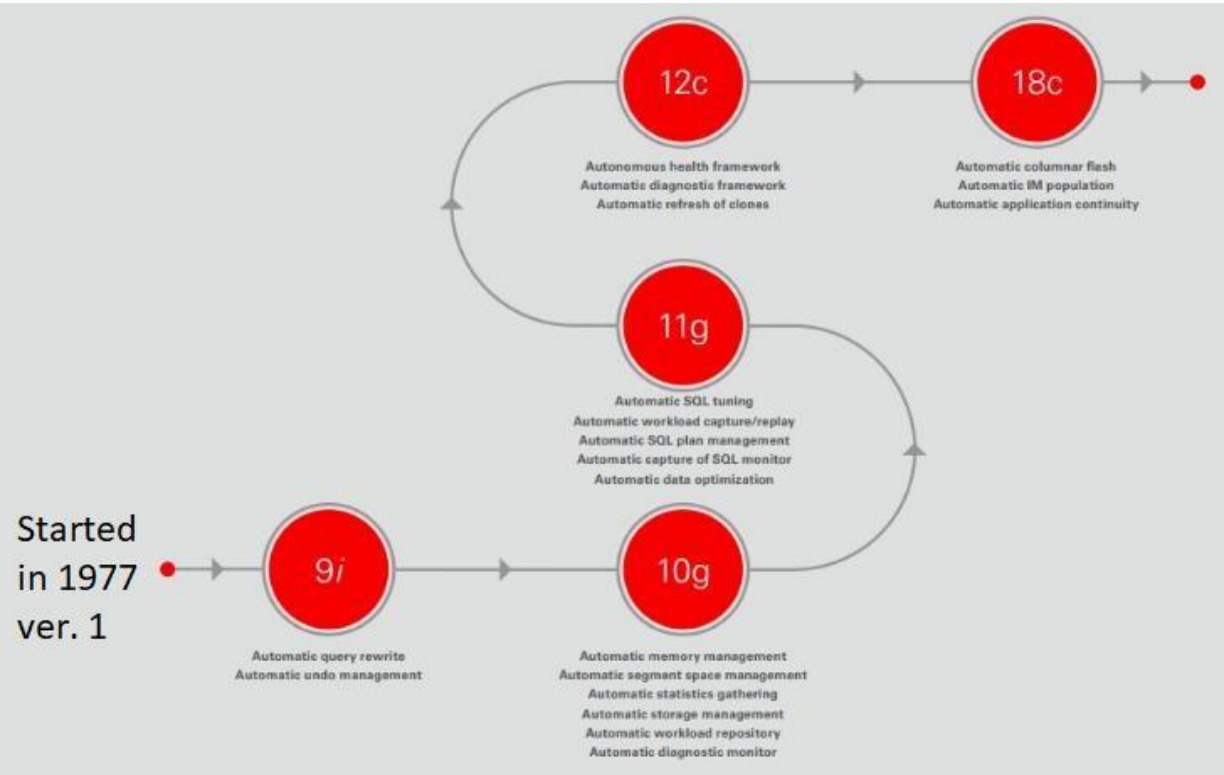
An application specifies what content is required. For example, an application requests an employee name or adds an employee record to a table. Execution of DDL in an application or schema of DBMS.

- **Physical operations**

RDBMS determines how things should be done and carries out the operation. For example, after an application queries a table, the database may use an index to find the requested rows and read the data into memory etc. before returning result.

Both operations are independent to each other

Oracle DBMS Quick History



Lesson 41: Oracle Schema Objects & Data Access

Oracle Schema Objects & Data Access

- Schema Objects
- Types of Schema Objects
- Tables
- Indexes
- Tables & Indexes in a Schema
- Views, Sequences & Synonym
- Data Access

Schema Objects

- Physical data storage is independent from logical data structures
- Schema is a collection of data structures, or schema objects
- Schema is owned by the database user name.
- Schema objects such as tables, views, indexes and sequences etc. are user-created structures that directly refer to the data (files) in the database.
- BMS created objects are data dictionary views, user's roles etc

Types of Schema Objects

- Tables
- Indexes
- Constraints
- Sequences
- Views
- Triggers
- Sequences
- Synonym
- PL/SQL subprograms and packages, Java Classes etc.

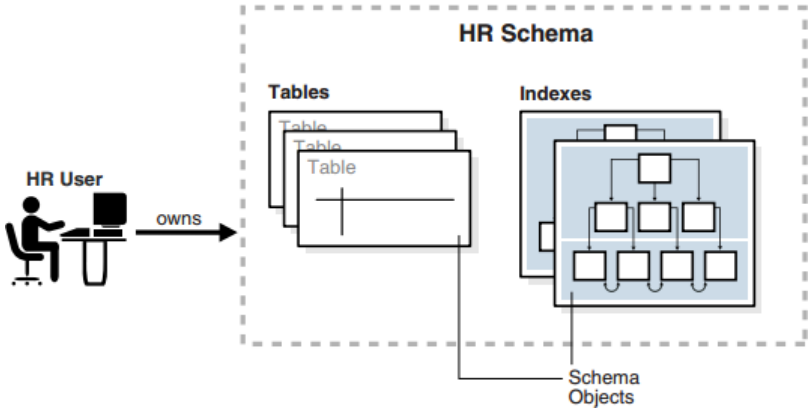
Tables

- Basic storage unit or schema object for storing data in structured form
- Table consists rows and columns with specific data types of columns
- Maximum number of columns are 255 and unlimited number of rows or tuples in a table
- Every table is uniquely identified in a schema
- Integrity constraints such as NOT NULL, CHECK and PKs etc. are applied on table's columns

Indexes

- An index is an optional data structure that can be created on one or more columns of a table.
- Index can increase performance of data retrieval
- Index can be created on searching columns
- Indexes are physically and logically independent

Tables & Indexes in a Schema



Views, Sequences & Synonym

- Stored queries are called Views. Views do not keep separate data rather they extract data from tables
- A Sequence is a user-created object that can be shared by multiple users. It can be ised for PK column.
- A Synonym is an alias for another schema object and does not require separate storage

Lesson 42: Oracle Transaction Management

Oracle Transaction Management

- Transaction management
- Multi-user Environment
- Transactions
- Example of Funds Transfer
- All or Nothing Transaction
- PL/SQL Transaction Example

Transaction Management

Oracle Database is designed as a multiuser database. The database must ensure that multiple users can work concurrently without corrupting one another's data.

Multi-user Environment

Oracle Database is designed as a multiuser database. The database must ensure that multiple users can work concurrently without corrupting one another's data.

Transactions

SQL is initiated at user level (client) and it executes on Server (central Database) at a distant, may be from one city to another.

A transaction is a logical, atomic unit of work (block) that contains one or more SQL statements.

Either transaction is done (committed) or not done (rolled back). No partial execution of transaction is allowed.

Example of Funds Transfer

A transaction is a funds transfer from a saving account to a checking account. The transfer consists of the following separate operations:

1. Decrease the saving account
2. Increase the checking account
3. Record the transaction in the transaction journal

Oracle Database guarantees that all three operations succeed or fail as a unit. In case of failure, all three operations will be rolled back.

All or Nothing Transactions

If you perform an atomic operation that updates several files, and if the system fails halfway through, then the files will not be consistent. In contrast, a transaction moves an Oracle Database from one consistent state to another

All or nothing, basic principle of atomic execution of a transaction

PL/SQL Transactions Example

```
BEGIN
     update emp
     set sal = 500
     where empno = 7988;

     delete from emp
     where empno = 7414;

     COMMIT;
EXCEPTION
    WHEN OTHERS THEN
    ROLLBACK;
END;
/
```

Lesson 43: Oracle Database Architecture

Oracle Database Architecture

- Database Architecture
- Database & Instance
- Oracle DB Server Architecture
- Instance DB Configuration
- Database Physical Storage Structure
- Database Logical Storage Structure

Database Architecture

- A database server is the key to information management
- In general, a server reliably manages a large amount of data in a multiuser environment so that users can concurrently access the same data
- A database server also prevents unauthorized access and provides efficient solutions for failure recovery.

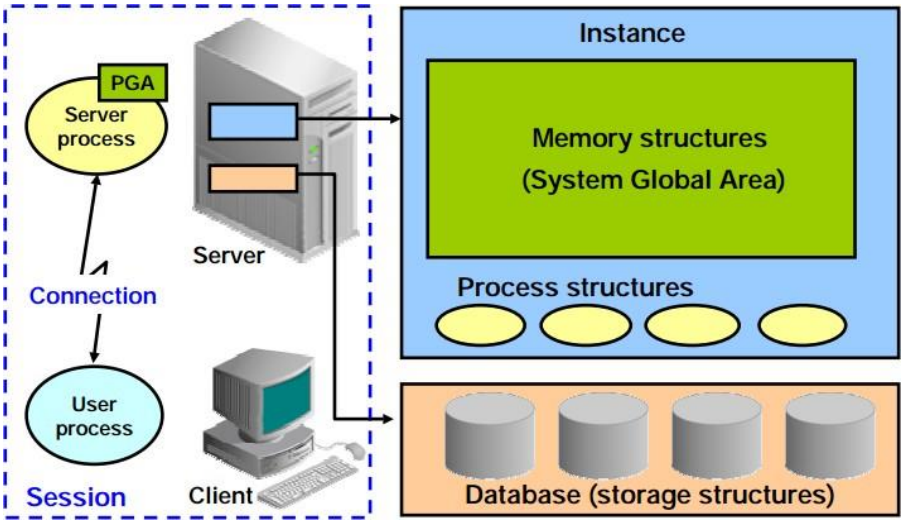
Database & Instance

- A database is a set of files, located on disk, that store data. These files can exist independently of a database instance.
- An instance is a set of memory structures that manage database files. The instance consists of a shared memory area, called the system global area (SGA), and a set of background processes. An instance can exist independently of database files.

Database & Instance

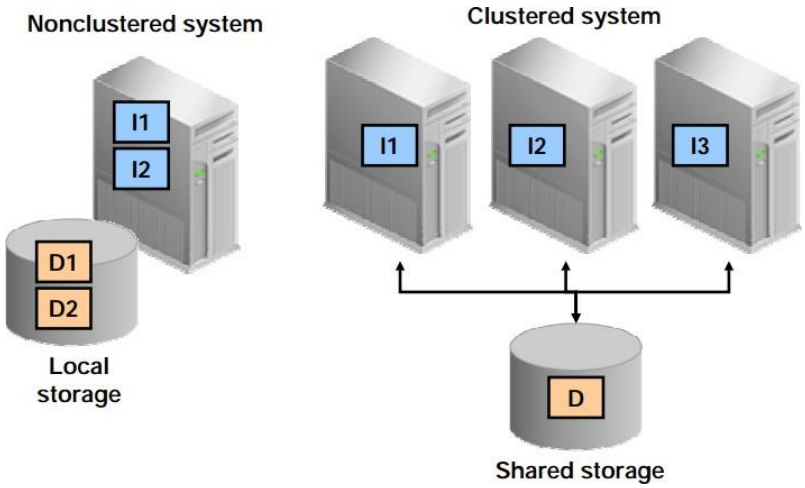
- Each client process is associated with its own server process. The server process has its own private session memory, known as the program global area (PGA).

Oracle DB Server Architecture



Each client process is associated with its own server process. The server process has its own private session memory, known as the program global area (PGA).

Instance DB Configuration



Each database instance is associated with one and only one database. If there are multiple databases on the same server, there is a separate and distinct database instance for each database. A database instance cannot be shared. A Real Application Clusters (RAC) database usually has multiple instances on separate servers for

the same shared database. In this model, the same database is associated with each RAC instance, which preserves the requirement that at most only one database be associated with an instance.

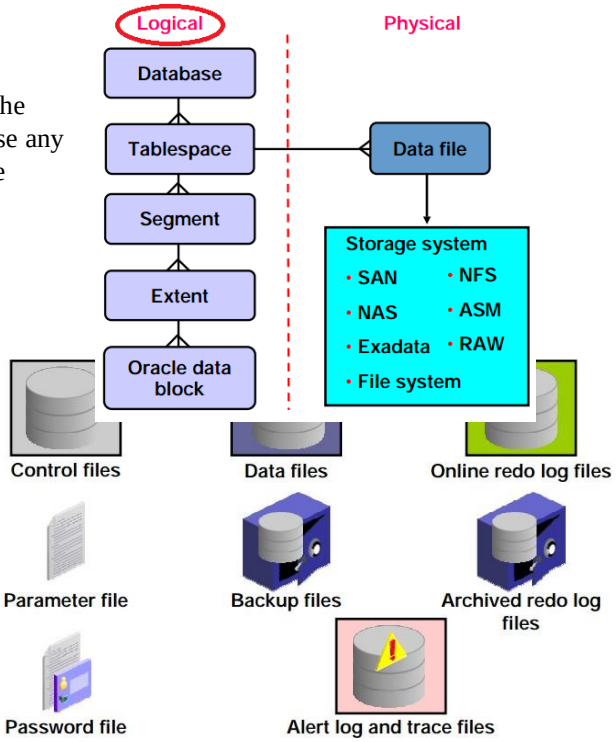
Database Logical Storage Structure

The files that constitute an Oracle database are organized into the following:

• Control files: Contain data about the database itself (that is, physical database structure information). These files are critical to the database. Without them, you cannot open data files to access the data in the database. It can also contain metadata related to backups.

• Data files: Contain the user or application data of the database, as well as metadata and the data dictionary

• Online redo log files: Allow for instance recovery of the database. If the database server crashes and does not lose any data files, the instance can recover the database with the information in these files.



Database Physical Storage Structure

The files that constitute an Oracle database are organized into the following:

• Control files: Contain data about the database itself (that is, physical database structure information). These files are critical to the database. Without them, you cannot open data files to access the data in the database. It can also contain metadata related to backups.

• Data files: Contain the user or application data of the database, as well as metadata and the data dictionary

• Online redo log files: Allow for instance recovery of the database. If the database server crashes and does not lose any data files, the instance can recover the database with the information in these files.

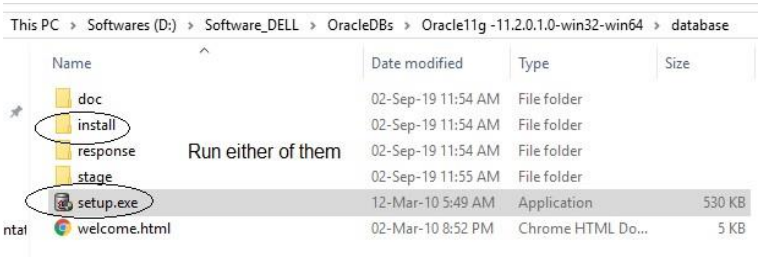
Lesson 44: Oracle 11g Installation

Installation Steps

- Installation Startup
- Installation …
- Login with Oracle Database Manager
- OS Services
- How to un-install Oracle?

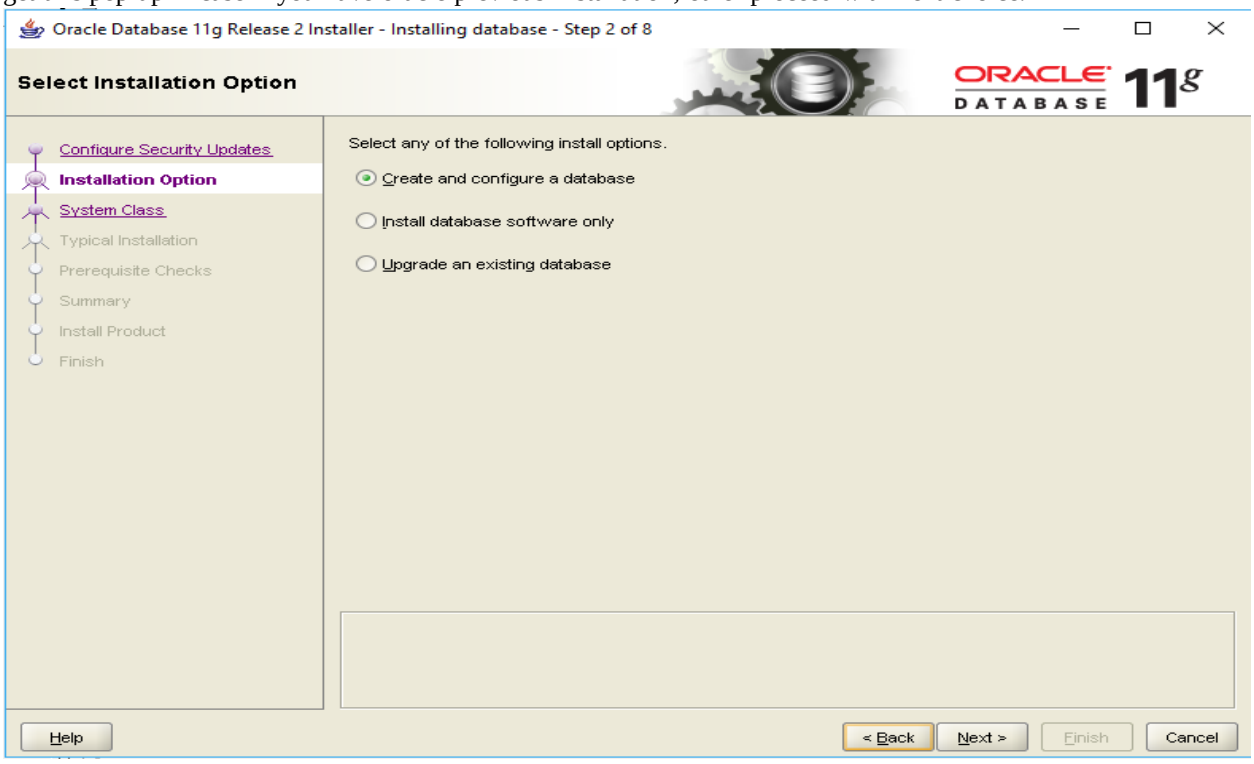
Installation Startup

Before Installing Oracle, you must installed complete OS windows with all patches and Activations (Install Java JDK 1.5 (e.g., jre-1_5_0-windows-i586.exe) Run SETUP.EXE to start Oracle Universal Installer



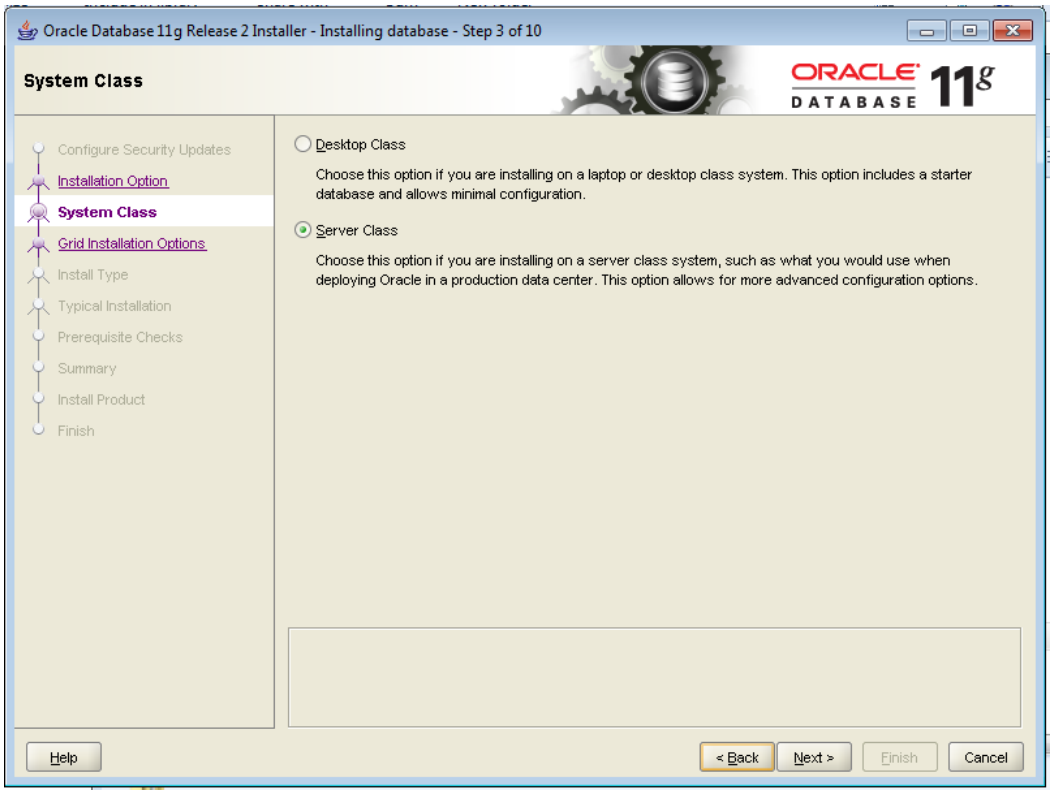
Installation Startup

Before Installing Oracle, you must install complete OS windows with all patches and activations. You may get this pop up in case if you have oracle previous Installation, other proceed with next choice.

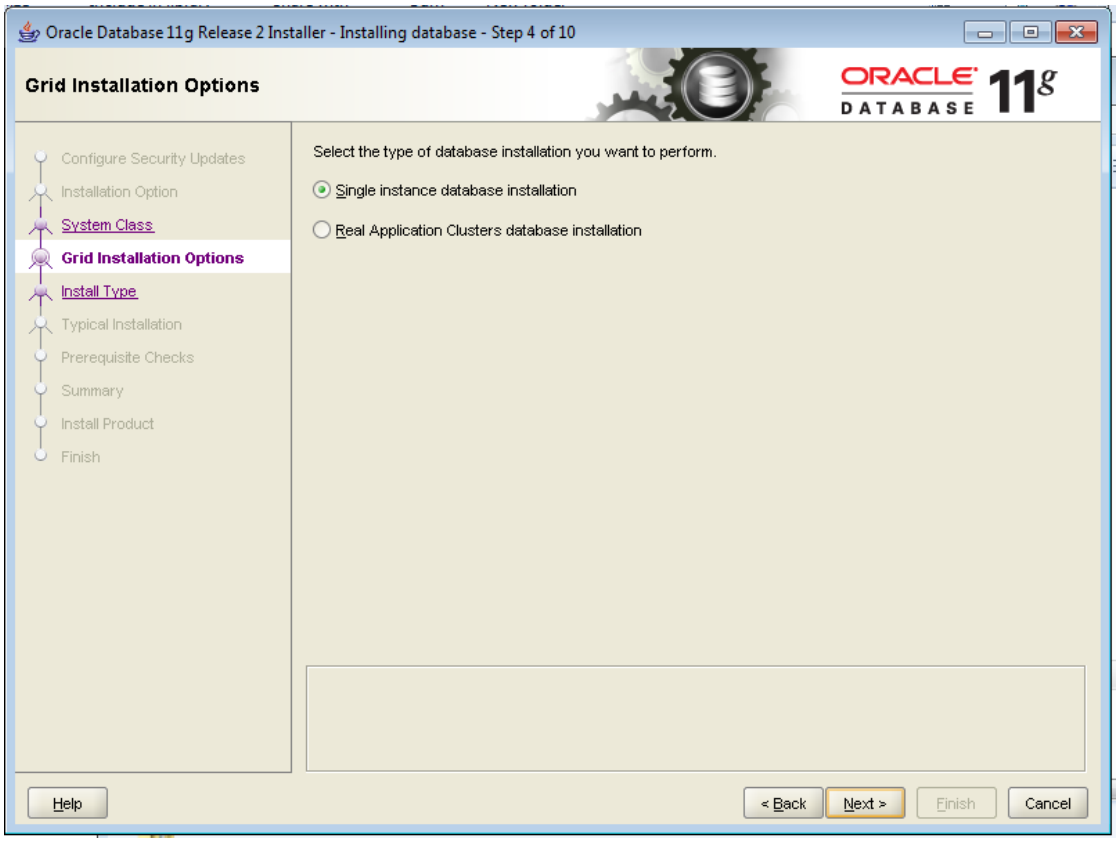


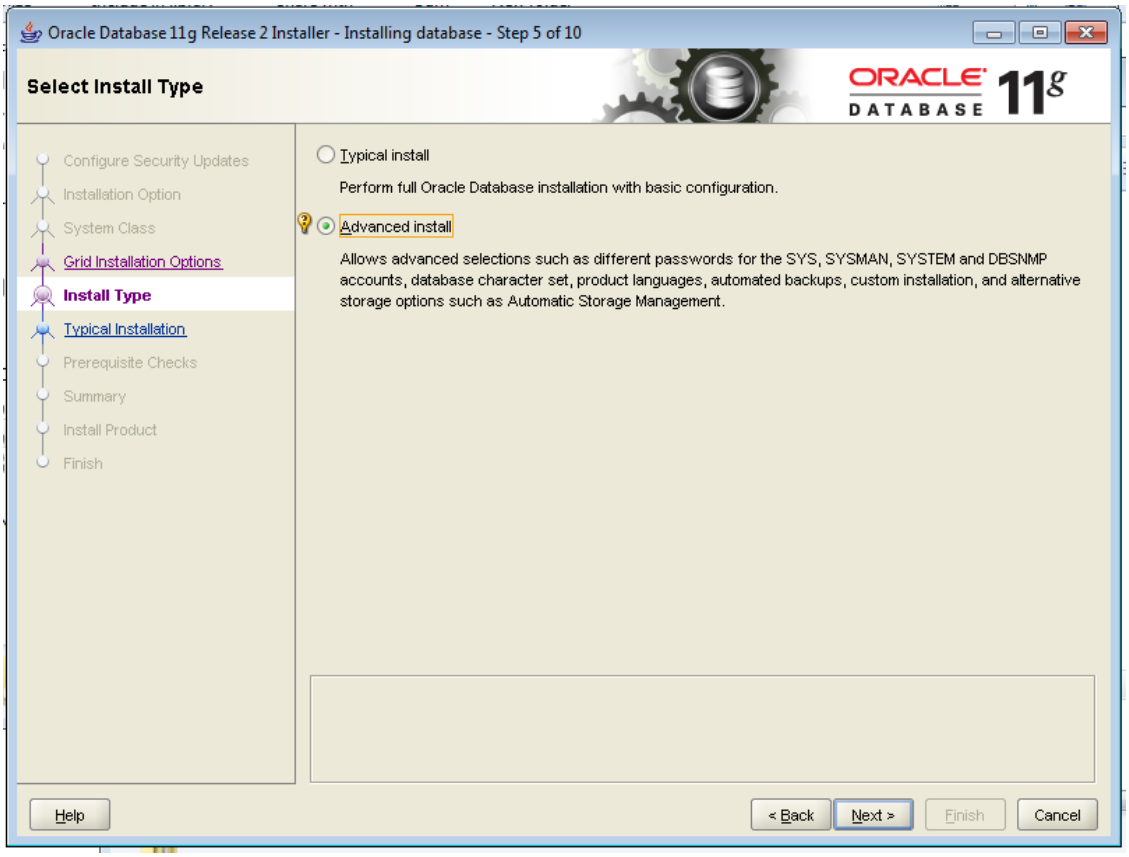
Installation Startup

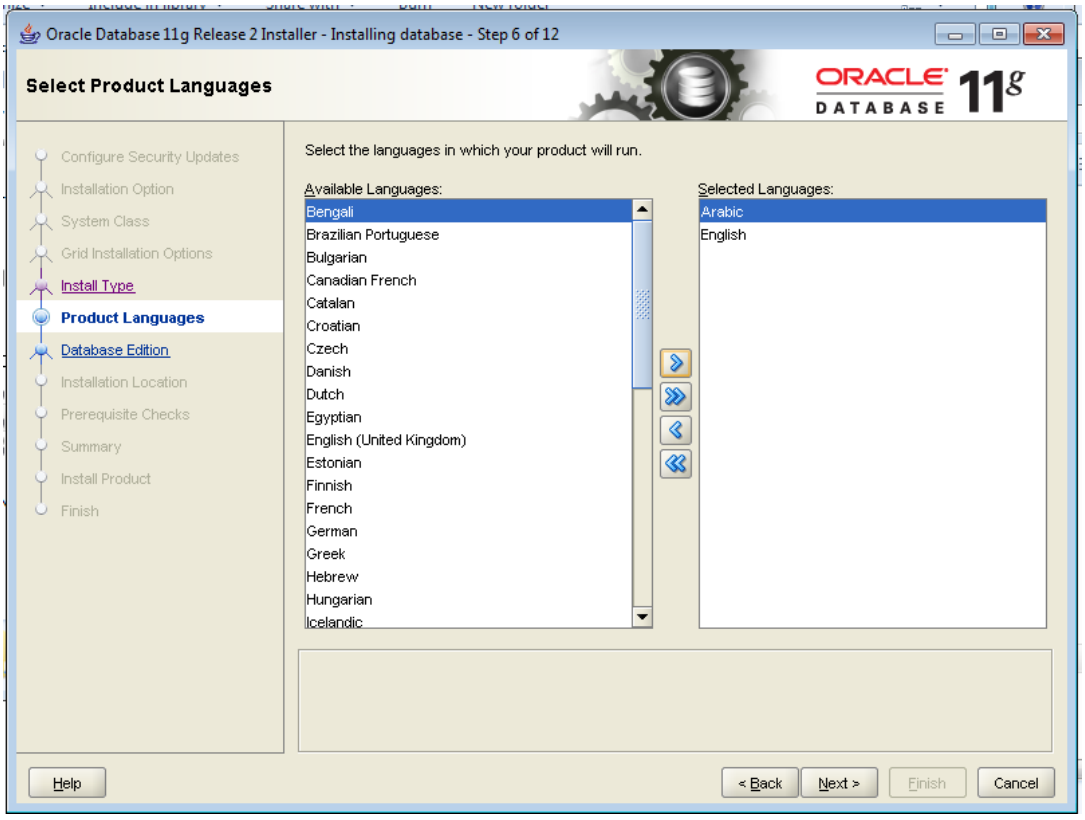
Before Installing Oracle, you must installed complete OS windows with all patches and activations Run SETUP.EXE to start Oracle Universal Installer



Installation







Lesson 45: MS SQL Server Installation

Installation Steps

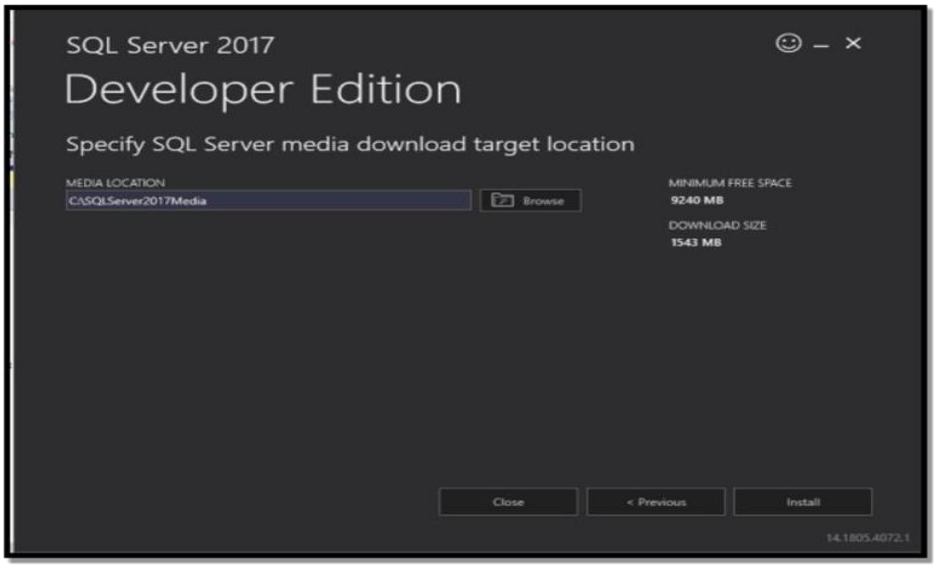
- Editions of MS SQL Server 2017
- Installation …
- Installation by Giving Password
- Installation in Progress
- Installation Complete

Editions of MS SQL Server 2017

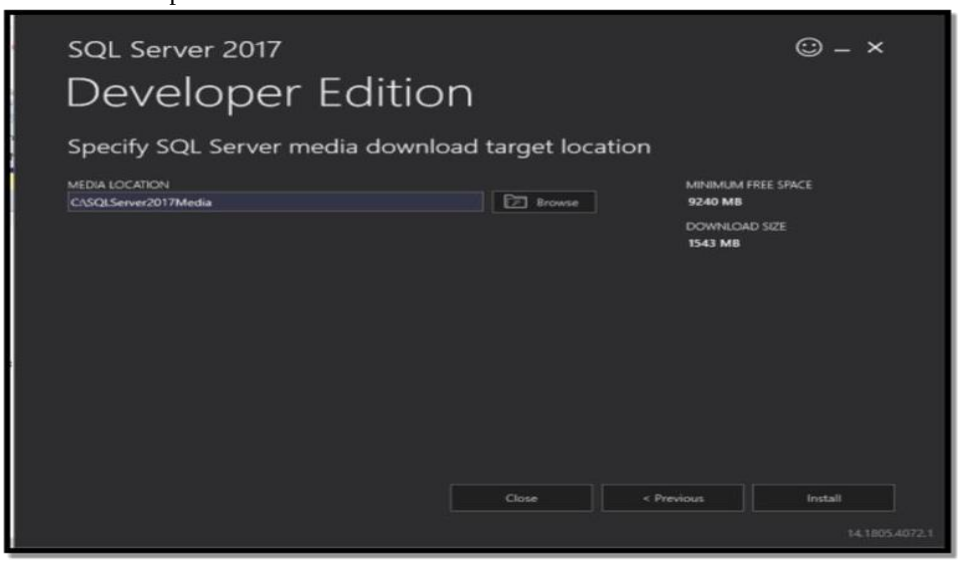
Enterprise: Comprehensive, high-end capacities for commanding database and business intelligence essentials.
Standard: Core data management and business intelligence abilities for non-critical workloads with minimal IT support.
Express: This is a free version with limited SQL Server features.
Developer: This is a free version to build, test, and demo apps in non-production environments. All Enterprise Edition features are included.
Web: SQL Server Web edition is a low-cost version for Web features

Installation …
Once SQL Server 2017 Developer edition is downloaded, double click on it to launch.

Specify the Media Location where we want to save the SQL Server media. The drive should have at minimum free space of 9240 MB in order to extract the media.

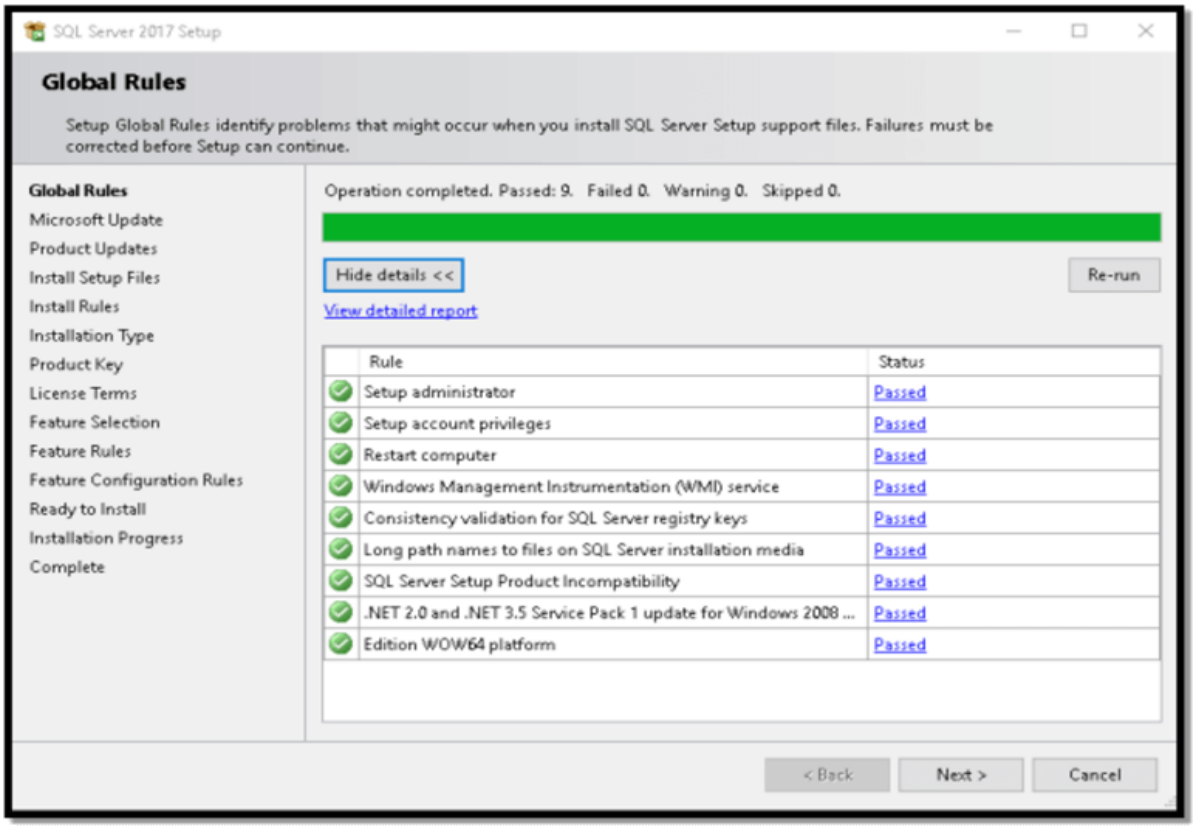


Specify the Media Location where we want to save the SQL Server media. The drive should have at minimum free space of 9240 MB in order to extract the media.



Click on New SQL Server stand-alone installation and you will get this message.

Make sure all Global rules are passed



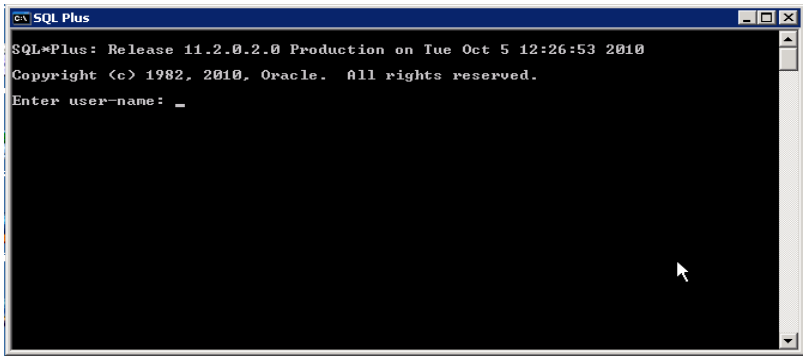
Lesson 46: Oracle Configuration for Connections, tnsnames.ora

Configuration for connection

- Connect using SQL*Plus
- Connect using SQL Developer
- Manual Connect using tnsnames.ora
- Oracle Client & Oracle Net

Connect using SQL*Plus

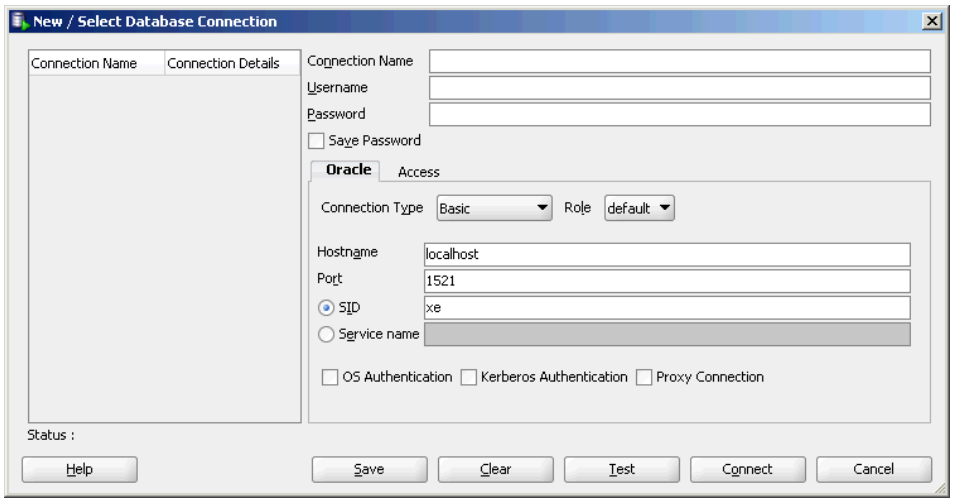
Click on SQL*Plus icon and login with your user id and password with database name (SID) that you have given during installation.



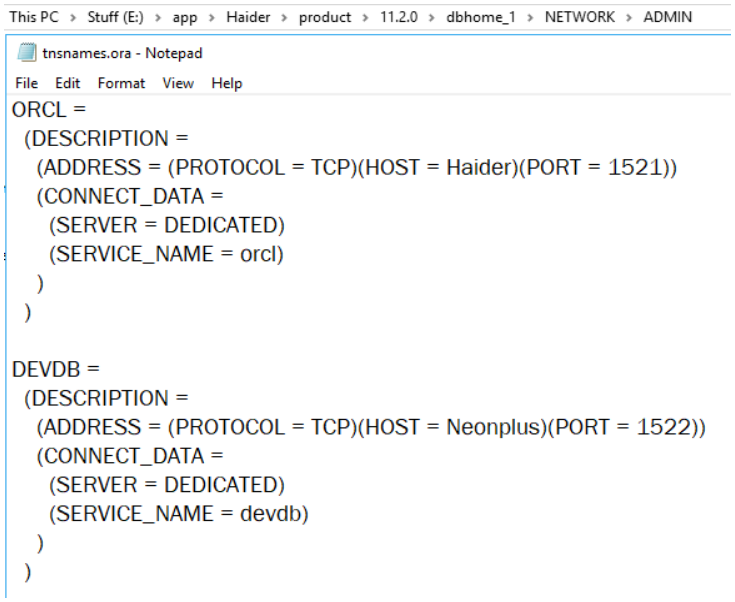
SQL>CONNECT username @ "hostname[:port][/DBname]"

Connect using SQL Developer

Right click Connection and choose New
The New/Select Database Connection dialog appears

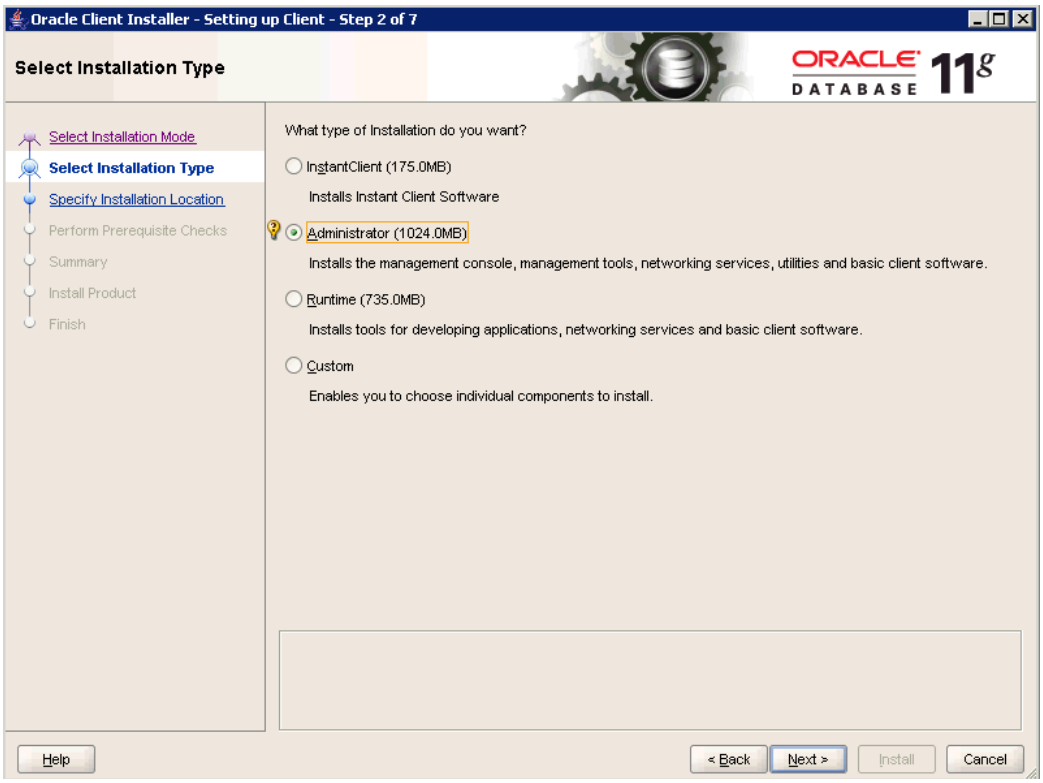


Manual Connect using tnsname.ora file

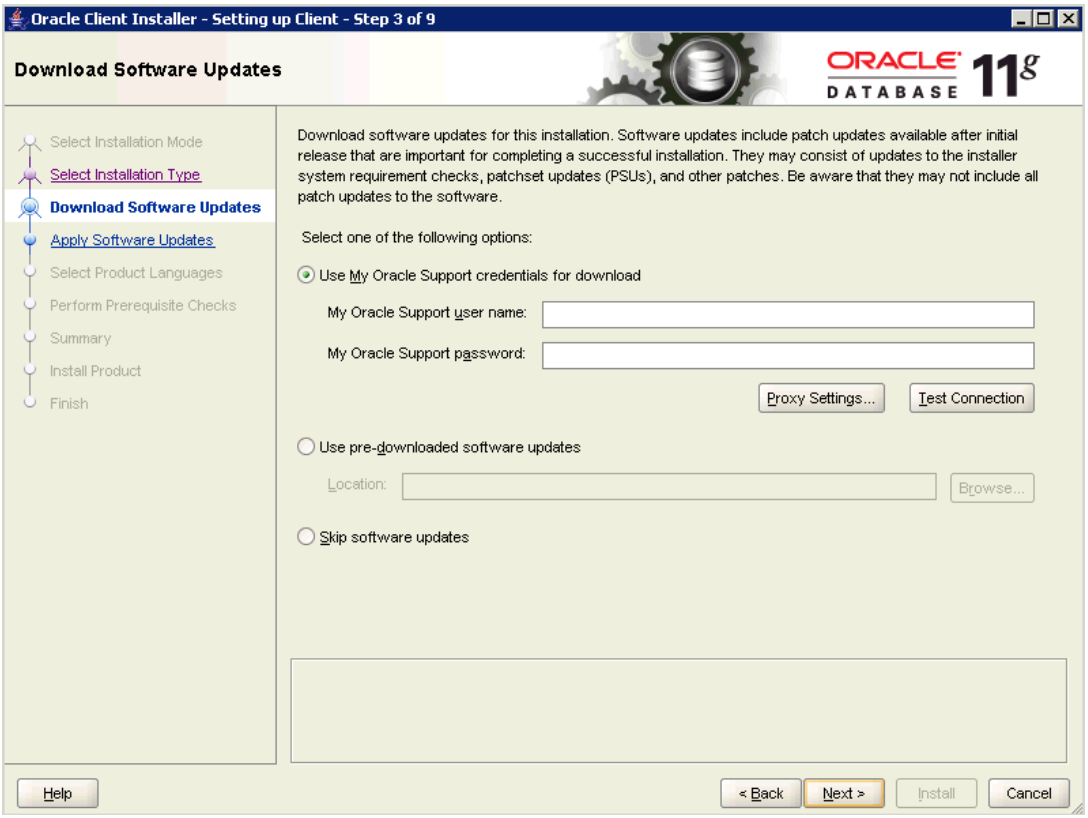


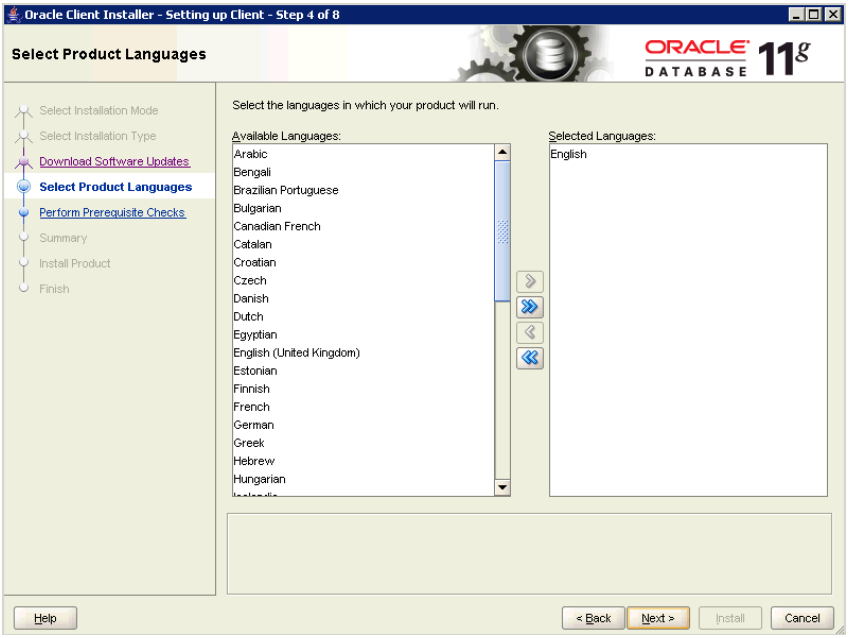
Oracle Client & Oracle Net

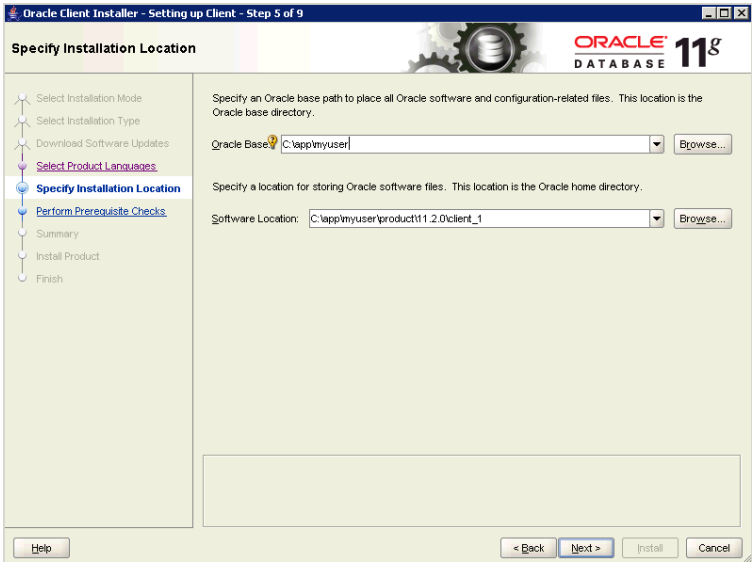
Run SETUP.EXE to start Oracle Universal Installer

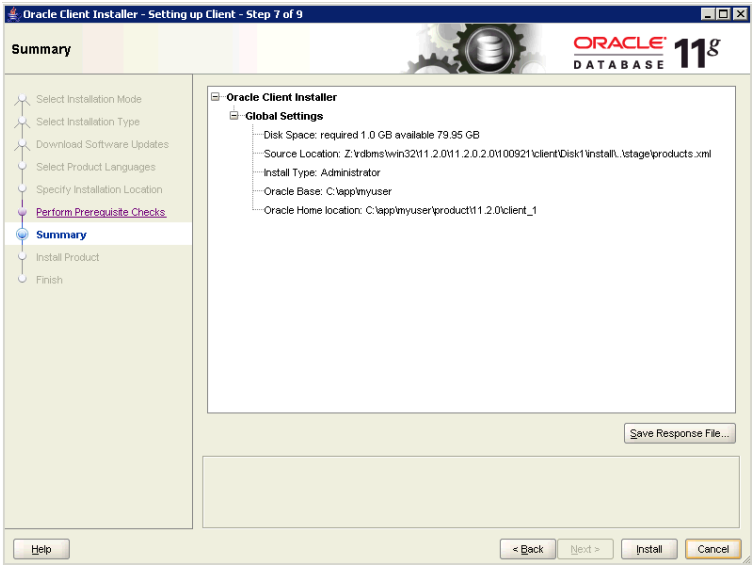


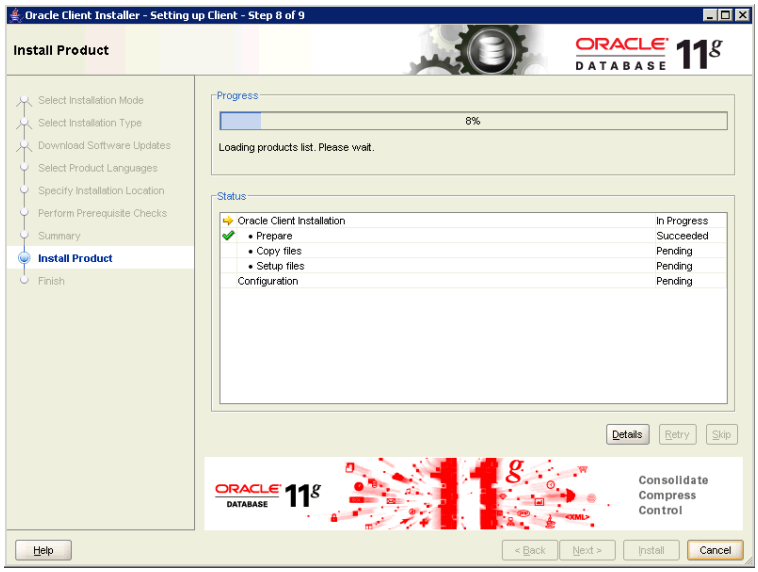
Oracle Client & Oracle Net

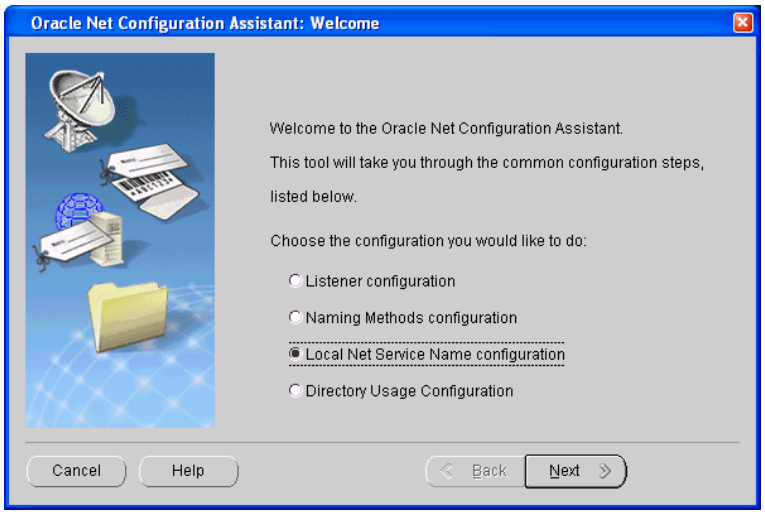


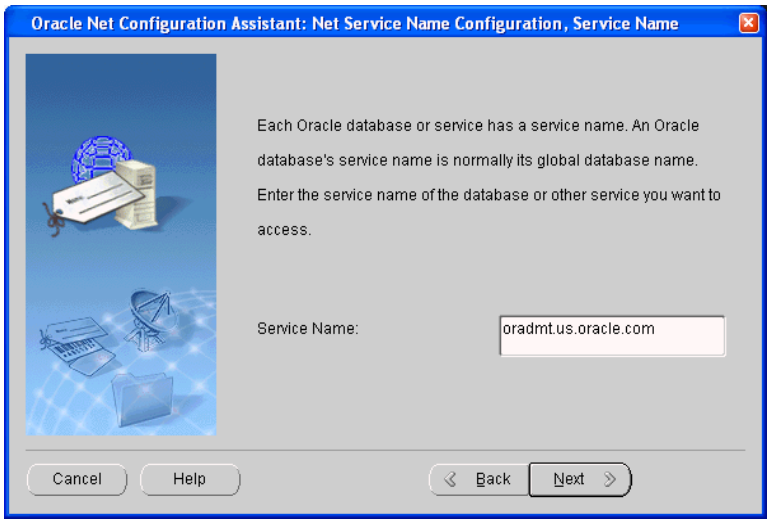


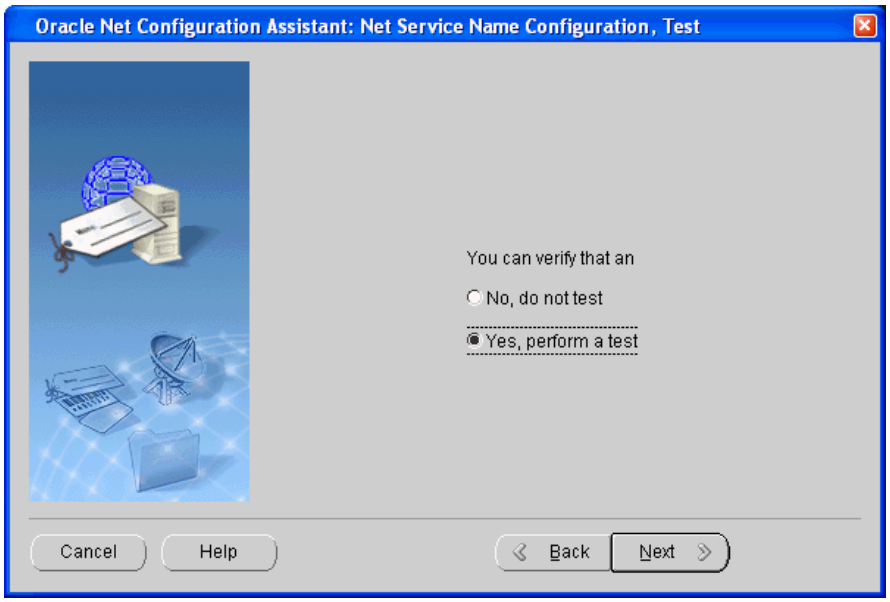


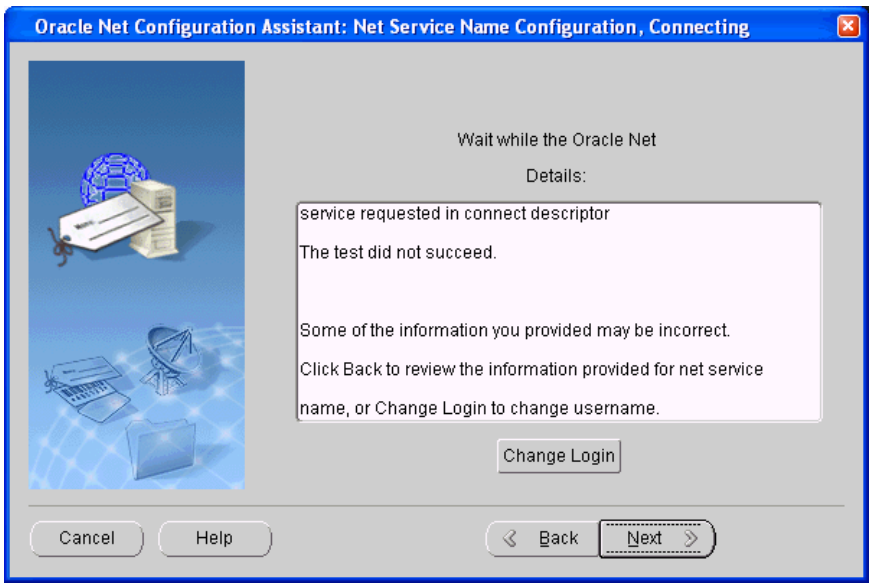


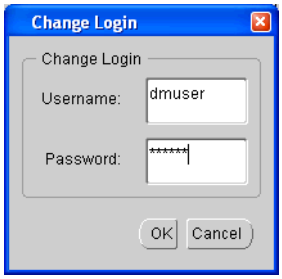












Lesson 47: Oracle Data Dictionary Views (Metadata) – I

Data Dictionary Views (Metadata)

- Metadata
- Data Dictionary
- Data Dictionary Contents
- Data Dictionary View Sets
- How to use Data Dictionary?

Metadata

Repository of information (metadata) describing the data in the database.
Data about data is called Metadata
Oracle metadata is managed by Data Dictionary (Read only access)

Data Dictionary is managed by using SQL
Catalogue, Repository are alternate names SQL>desc dict

Data Dictionary

- Schema objects in the database including default values for columns and integrity constraints
- Show up of amount of space or storage allocated by objects
- Show up of database users, privileges and roles of users.

Data Dictionary Contents

- Base Tables

These underlying tables store information about the database. Only Oracle database should write to and read these tables.

- Views

These views decode the base table data into useful information, such as user or table names, using joins and WHERE clauses to simplify the information. These views contain the names and description of all objects in the data dictionary.

Data Dictionary View Sets

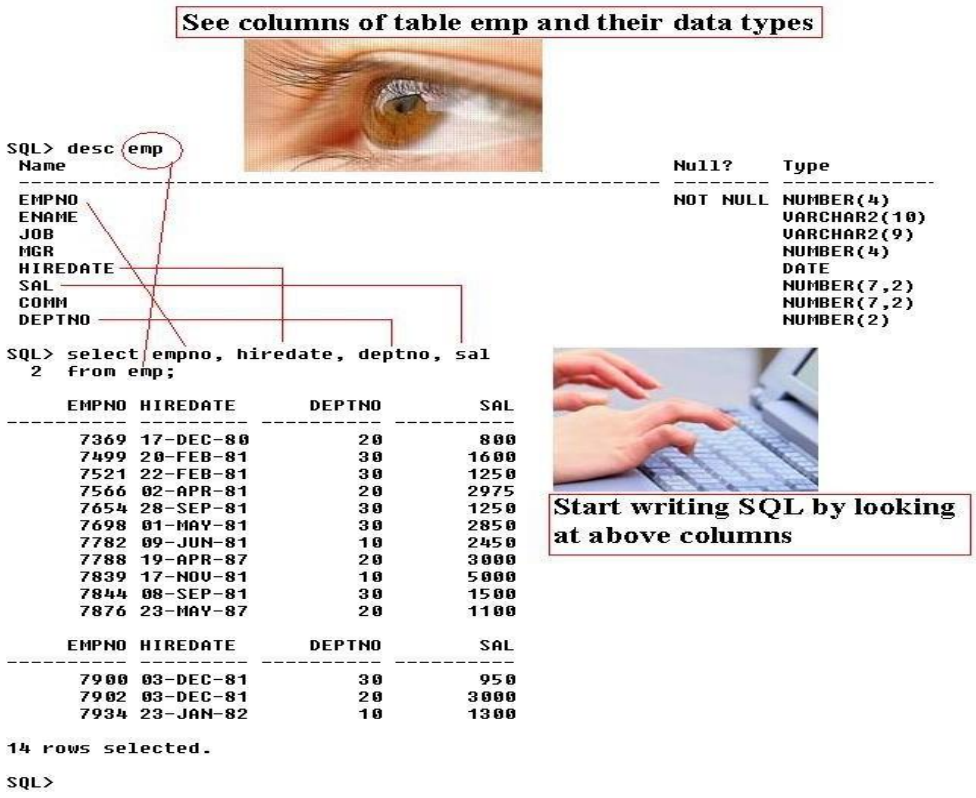
Prefix	User Access	Contents	Notes
DBA_	Database administrators	All objects	Some DBA_ views have additional columns containing information useful to the administrator.
ALL_	All users	Objects to which user has privileges	Includes objects owned by user. These views obey the current set of enabled roles.
USER_	All users	Objects owned by user	Views with the prefix USER_ usually exclude the column OWNER. This column is implied in the USER_ views to be the user issuing the query.

```
SELECT OWNER, OBJECT_NAME, OBJECT_TYPE
FROM   ALL_OBJECTS
ORDER BY OWNER, OBJECT_NAME;

SELECT OBJECT_NAME, OBJECT_TYPE
FROM   USER_OBJECTS
ORDER BY OBJECT_NAME;

SQL> SELECT ((3*4)+5)/3 FROM DUAL;

((3*4)+5)/3
-----------
 5.66666667
```

How to use Data Dictionary?

Login with Sys / as sysdba

SQL> desc Dict



Lesson 48: Oracle Data Dictionary Views (Metadata) – II

Data Dictionary Views (Metadata)

- Example-1: Data Dictionary
- Example-2: Data Dictionary
- Dynamic Performance Views

Example-1: Data Dictionary

Desc dict
select table_name
from dict
where table_name like '%CONSTRAINT%';
desc user_constraints

Example-1: Data Dictionary

To see constraints of table EMP:
select constraint_name, table_name, R_constraint_name
from user_constraints
where table_name='EMP';

Example-1: Data Dictionary

To see constraints of table EMP:
select constraint_name, table_name, R_constraint_name
from user_constraints
where table_name='EMP';

Example-2: Data Dictionary
To see the size of table EMP in KB
--login with user scott
select sum(bytes)/1024/1024 from dba_segments where segment_name ='EMP'; select segment_name, sum(bytes)/1024/1024 from USER_segments group by segment_name;

Dynamic Performance Views
These views are called dynamic performance views because they are continuously updated while a database is open and in use. The views, also sometimes called V$ views, contain

- System and session parameters
- Memory usage and allocation
- File states (including RMAN backup files)

Dynamic Performance Views

- Progress of jobs and tasks
- SQL execution
- Statistics and metrics

e.g., V$SESSION, V$DATAFILE etc.

Lesson 49: Memory Management & Memory Architecture Overview – I

Memory & Architecture

- Oracle Memory Management
- Automatic Memory Management
- Adjust Memory Target Manually
- Enable Automatic Shared Memory

Oracle Memory Management

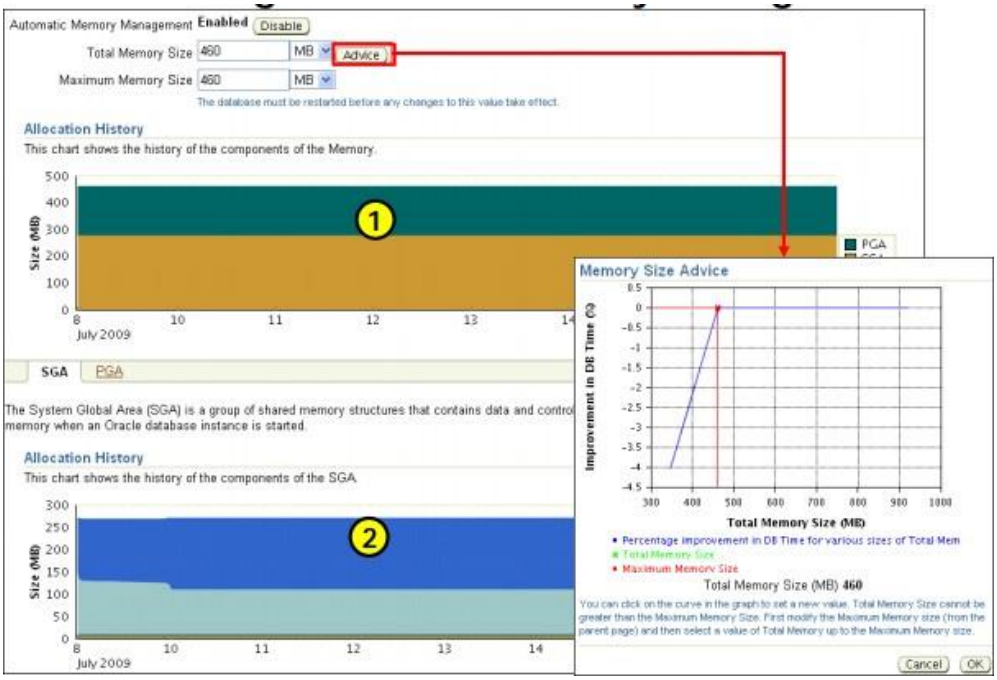
- Memory management involves maintaining optimal sizes for the Oracle Database instance memory structures as demands on the database change.
- The memory structures that must be managed are the system global area (SGA) and the instance program global area (instance PGA).

Oracle Memory Management

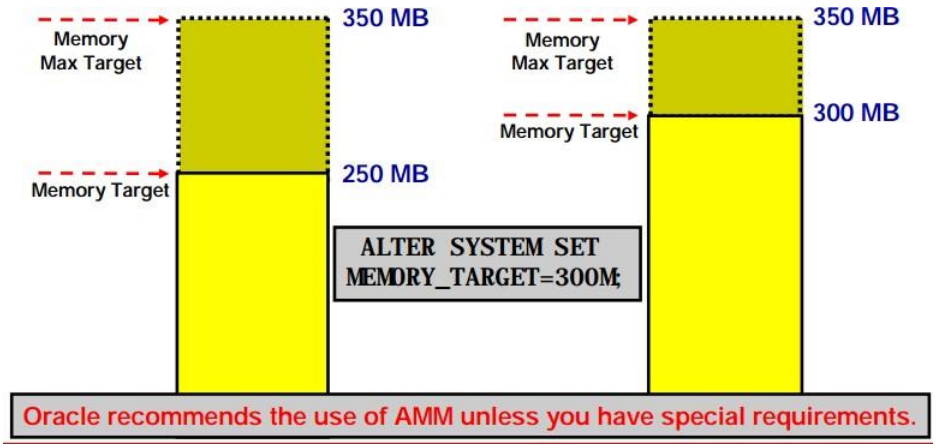
- Oracle recommends that you enable the method known as automatic memory management.

Automatic Memory Management

- You designate only the total memory size to be used by the instance, and Oracle Database dynamically exchanges memory between the SGA and the instance PGA as needed to meet processing demands.
- If you prefer to exercise more direct control over the sizes of individual memory components, you can disable automatic memory management and configure the database for manual memory management



Adjust Memory Target Manually



Enable Automatic Shared Memory

To enable ASMM from manual shared memory management:

1. Get a value for **SGA_TARGET**:

```
SELECT ((SELECT SUM(value) FROM V$SGA) - (SELECT CURRENT_SIZE
FROM V$SGA_DYNAMIC_FREE_MEMORY)) "SGA_TARGET" FROM DUAL;
```

2. Use that value to set **SGA_TARGET**.
3. Set the values of the automatically sized SGA components to 0.

To switch to ASMM from Automatic Memory Management:

1. Set the **MEMORY_TARGET** initialization parameter to 0.
2. Set the values of the automatically sized SGA components to 0.

SELECT ((SELECT SUM(value) FROM V$SGA) - (SELECT CURRENT_SIZE

FROM V$SGA_DYNAMIC_FREE_MEMORY)) "SGA_TARGET" FROM DUAL;

Lesson 50: Memory Management & Memory Architecture Overview - II

Memory Management & Architecture

- Memory Architecture Overview
- Oracle Database Memory Structure
- Current Sizes of SGA_TARGET and SGA_AGGREGATE_TARGET
- V$MEMORY_TARGET_ADVICE provides tuning advice for the MEMORY_TARGET
- V$PGASTAT

Memory Architecture Overview

- System Global Area (SGA)

The SGA is shared by all server and background processes. Examples of data stored in the SGA include cached data blocks and shared SQL areas.

Memory Architecture Overview

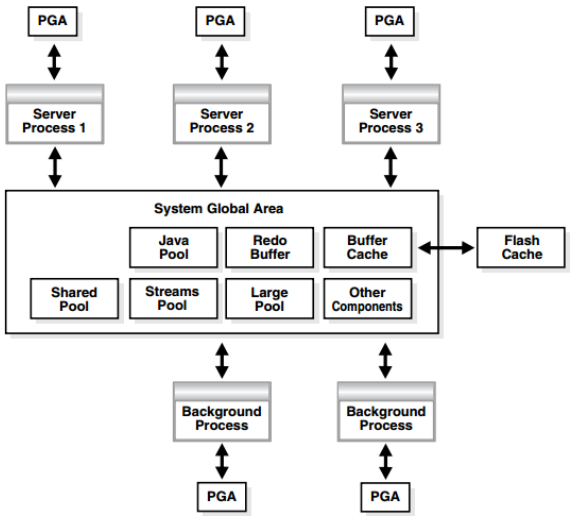
- Program Global Area (PGA)

A PGA is a memory region that contains data and control information for a server process. It is non shared memory created by Oracle Database when a server process is started. Access to the PGA is exclusive to the server process. There is one PGA for each server process.

Oracle Database Memory Structure

It can improve response time and overall throughput, especially for read-intensive online transaction processing (OLTP) workloads. The flash cache

resides on one or more flash disk devices, which are solid state storage devices that use flash memory.



Current Sizes of SGA_TARGET and SGA_AGGREGATE_TARGET

SHOW PARAMETER TARGET

```
NAME                                 TYPE        VALUE
------------------------------------ ----------- ----------------
archive_lag_target                   integer     0
db_flashback_retention_target        integer     1440
fast_start_io_target                 integer     0
fast_start_mttr_target               integer     0
memory_max_target                    big integer 0
memory_target                        big integer 0
parallel_servers_target              integer     16
pga_aggregate_target                 big integer 90M
sga_target                           big integer 272M
```

Run the following query to determine the maximum instance PGA allocated since the database was started:

```
select value from v$pgastat where name='maximum PGA allocated';
```

Compute the maximum value between the query result from step 2b and PGA_AGGREGATE_TARGET. Add SGA_TARGET to this value.

```
memory_target = sga_target + max(pga_aggregate_target, maximum PGA
allocated)
```

V$MEMORY_TARGET_ADVICE provides tuning advice for the MEMORY_TARGET

```
SQL>  select * from v$memory_target_advice order by memory_size;

MEMORY_SIZE MEMORY_SIZE_FACTOR ESTD_DB_TIME ESTD_DB_TIME_FACTOR    VERSION
----------- ------------------ ------------ -------------------- ----------
        180                .5          458                1.344          0
        270               .75          367               1.0761          0
        360                 1          341                    1          0
        450              1.25          335                .9817          0
        540               1.5          335                .9817          0
        630              1.75          335                .9817          0
```

V$PGASTAT

select value

from v$pgastat

where name='maximum PGA allocated';

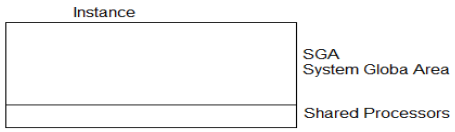
For example, if SGA_TARGET is 272M and PGA_AGGREGATE_TARGET is 90M as shown above, and if the maximum PGA allocated is determined to be 120M, then MEMORY_TARGET should be at least 392M (272M + 120M).

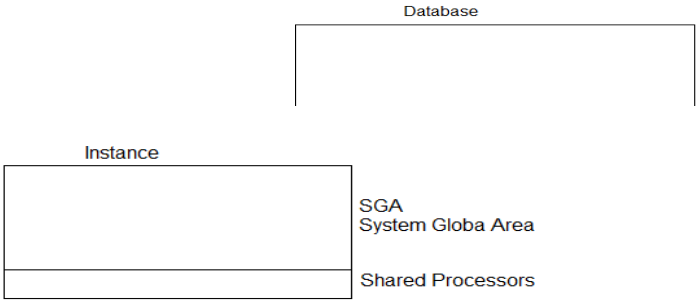
Lesson 51: Memory Architecture Overview (SGA)

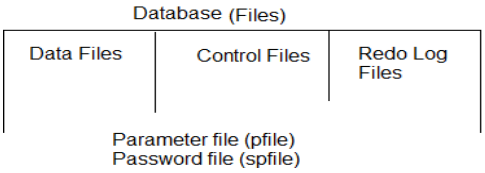
Memory Management & Architecture

- Instance & Database
- Overview of SGA
- Overview of SGA & Database

Memory Architecture Overview







Control Files: Infor. where the files are located, manages overall structure of database

Overview of SGA & Database

Shared Pool: All the executable and libraries (SQL, PLSQL) are available, Data dictionary Cache, processing queries

Buffer Cache: SQL brought data from data file into buffer cache

Entries are made into Redo Log Buffer and data changes are also managed

Many processes, but these are main 5 processes

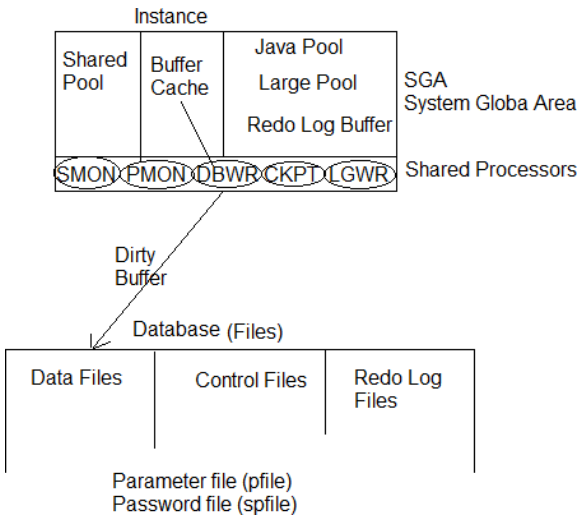
Any data block (changes in data) in buffer cache is written into data file using DBWR, called Dirty Buffer

LGWR manages log of changes done in Buffer Cache, responsible to write in Redo Log Files

CKPT synchronizes between Data Files and Redo Log Files

SMON: used for Recovery

PMON: Process monitorying of all activities



Lesson 52: Introduction to Database Administration

Configuring Memory

Configuring Memory with SGA

- Fundamentals of Configuration
- Granules
- Granule Size
- SGA Memory Size
- SGA_MAX_SIZE
- Setting SGA Target Size
- Manual Shared Memory Management

Fundamentals of Configuration

Disable automatic memory management of SGA set target and maximum sizes for the SGA With automatic shared memory management

The database then sets the total size of the SGA to your designated target, and dynamically tunes the sizes of many SGA components.

Set the sizes of several individual SGA components with manual shared memory management, thereby determining the overall SGA size. You then manually tune these individual SGA components on an ongoing basis.

Similarly, for the instance PGA, there is automatic/ manual PGA memory management, in which you set a target size for the instance PGA.

Granules

SGA memory components include

- the shared pool (used to allocate memory for SQL and PL/SQL execution),
- the java pool (used for java objects and other java execution memory), and
- the buffer cache (used for caching disk blocks).

All SGA components allocate and deal locate space in units of granules.

The granule size is based on the value of the SGA_MAX_SIZE initialization parameter.

Lesson 53:Memory Related Data Dictionary Views

Memory Data Dictionary Views

- Memory Parameters
- Memory Target & SPFILE
- Tune SGA Target
- PGA related Queries
- Table Size
- Blocks & Extents Sizes
- Database Block Size
- Extent and PCT_FREE Sizes
- Maximum Size of Data File
- Maximum Size of Data File with Blocks

Memory Parameters

```
SQL> show parameter memory

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------
hi_shared_memory_address             integer     0
memory_max_target                    big integer 1232M
memory_target                        big integer 1232M
shared_memory_address                integer     0
SQL> show memory_target
```

```
SQL> show parameter SGA

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------
lock_sga                             boolean     FALSE
pre_page_sga                         boolean     FALSE
sga_max_size                         big integer 1G
sga_target                           big integer 0
```

```
SQL> show parameter target

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------
archive_lag_target                   integer     0
db_flashback_retention_target        integer     1440
fast_start_io_target                 integer     0
fast_start_mttr_target               integer     0
memory_max_target                    big integer 1232M
memory_target                        big integer 1232M
parallel_servers_target              integer     16
pga_aggregate_target                 big integer 0
sga_target                           big integer 0
SQL>
```

```
SQL> set linesize 200
SQL> select substr(name,1,10) fname, substr(value,1,100) fvalue
  2  from v$parameter
  3  where name='spfile';

FNAME      FVALUE
---------- ----------------------------------------------------------------
spfile     E:\APP\HAIDER\PRODUCT\11.2.0\DBHOME_1\DATABASE\SPFILEORCL.ORA
```

Memory Target & SPFILE

create pfile='E:\app\Haider\admin\orcl\pfile\ init.ora.862016235218‘ from 'E:\app\Haider\product\11.2.0\dbhome_1\dbs\init.ora';

Alter system set memory_target=1021 M scope=spfile;

Alter system set SGA_TARGET=0 scope=spfile;

Alter system set PGA_AGGREGATE_TARGET=0 scope=spfile;

shutdown immediate

Show parameter memory

Tune SGA Target

Desc V$SGA_TARGET_ADVICE

select sga_size,estd_db_time

from V$SGA_TARGET_ADVICE;

SGA_SIZE	ESTD_DB_TIME
404	100326
606	97377
808	96681
1010	96333
1212	96323
1414	96323
1616	96323

7 rows selected.

PGA related Queries

select low_optimal_size/1024 low_kb,

(high_optimal_size+1)/1024 high_kb,

optimal_executions,

onepass_executions,

multipasses_executions

from v$sql_workarea_histogram

where total_executions <> 0;

LOW_KB	HIGH_KB	OPTIMAL_EXECUTIONS	ONEPASS_EXECUTIONS	MULTIPASSES_EXECUTIONS
2	4	354667	0	0
64	128	626	0	0
128	256	190	0	0
256	512	532	0	0
512	1024	3728	0	0
1024	2048	110	0	0
2048	4096	210	0	0
4096	8192	10	0	0
8192	16384	1	0	0

9 rows selected.

Table Size

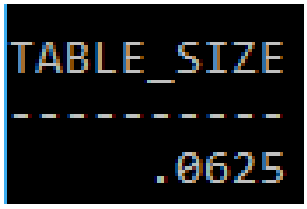
Login with scott/tiger

Desc dba_segments

select sum(bytes)/1024/1024 Table_size_MB

from dba_segments

where segment_name ='EMP';



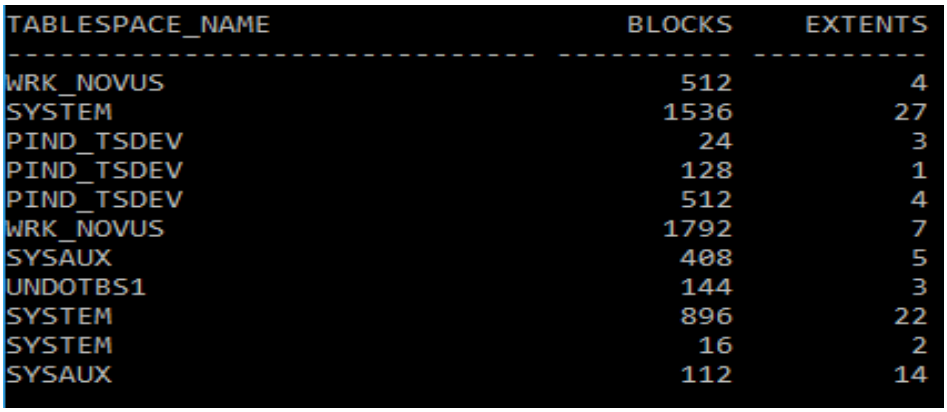
```
TABLE_SIZE
----------
     .0625
```

Blocks & Extents Sizes

select distinct tablespace_name,

blocks, extents

from dba_segments;



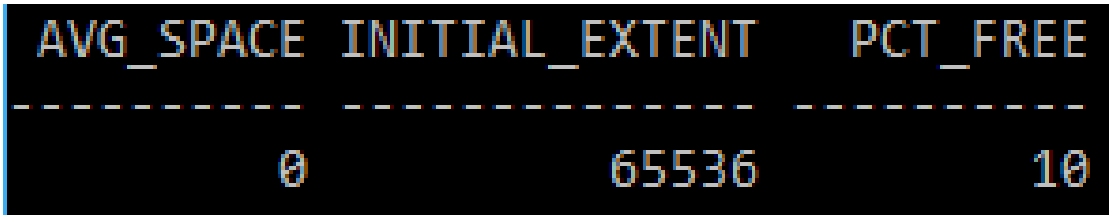
TABLESPACE_NAME	BLOCKS	EXTENTS
WRK_NOVUS	512	4
SYSTEM	1536	27
PIND_TSDEV	24	3
PIND_TSDEV	128	1
PIND_TSDEV	512	4
WRK_NOVUS	1792	7
SYSAUX	408	5
UNDOTBS1	144	3
SYSTEM	896	22
SYSTEM	16	2
SYSAUX	112	14

Extent and PCT_FREE Sizes

select avg_space, initial_extent, PCT_FREE

from ALL_TABLES

where table_name = 'EMP';



AVG_SPACE	INITIAL_EXTENT	PCT_FREE
0	65536	10

Maximum Size of Data File

How to see block size and max file size with DBA_DATA_FILES (4 Millions data blocks)

Find db block size from v$parameter

4194304 * db_block_size

Maximum Size of Data File

If your db_block_size is 8kb, then your maximum datafile size is:

4194304 * 8192 = 34359738368 / 1024 / 1024 / 1024 = 32GB.

select file_name, tablespace_name, blocks

from dba_data_files;

Maximum Size of Data File with Blocks

Each Oracle datafile can contain maximum 4194303 (4 Million) data blocks.

Block Size | Maximum Datafile Size
——————————————
2k 4194303 * 2k = 8 GB
4k 4194303 * 4k = 16 GB
8k 4194303 * 8k = 32 GB
16k 4194303 * 16k = 64 GB
32k 4194303 * 32k = 128 GB

In Oracle Database 10g, BIGFILE tablespace was introduced.
(4 billion) data blocks.

Block Size | Maximum Datafile Size
——————————————
2k 4294967295 * 2k = 8 TB
4k 4294967295 * 4k = 16 TB
8k 4294967295 * 8k = 32 TB
16k 4294967295 * 16k = 64 TB
32k 4294967295 * 32k = 128 TB

Lesson 54: Backup using

MS SQL Server

Backup using MS SQL Server

- Terminologies
- Failures of Database
- Create a Full Backup
- Create a Simple test database
- Backup using MSSMS
- Steps of Backup using MSSMS
- Backup on non-default Location

Terminologies

- Back up [verb]

The process of creating a backup [noun] by copying data records from a SQL Server database, or log records from its transaction log.

- Backup [noun]

A copy of data that can be used to restore and recover the data after a failure. Backups of a database can also be used to restore a copy the database to a new location.

- Backup device or media

A disk or tape device to which SQL Server backups are written and from which they can be restored. SQL Server backups can also be written to an Azure Blob storage service, and URL format is used to specify the destination and the name of the backup file

- Full backup

A data backup that contains all the data in a specific database or set of file groups or files, and also enough log to allow for recovering that data.

Failures of Database

- Media failure.
- User errors, for example, dropping a table by mistake.
- Hardware failures, for example, a damaged disk drive or permanent loss of a server.
- Natural disasters. By using SQL Server Backup to Azure Blob storage service, you can create an off-site backup in a different region than your on-premises location, to use in the event of a natural disaster affecting your on-premises location.

Create a Full Backup

- Security

TRUSTWORTHY is set to OFF on a database backup

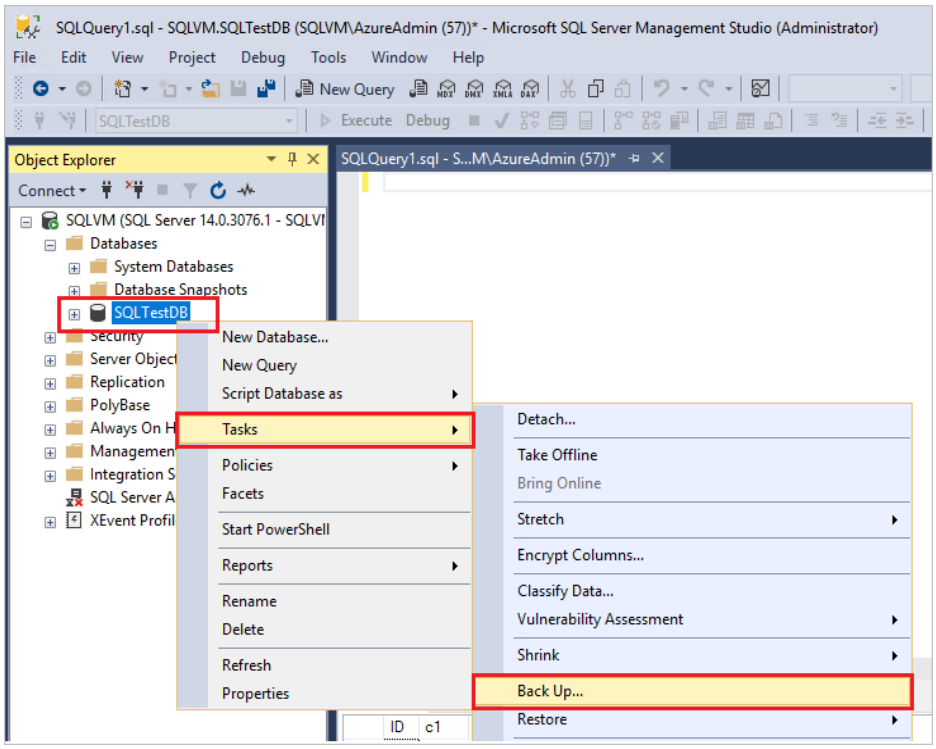
- Permission

BACKUP DATABASE and BACKUP LOG permissions default to members of the sysadmin fixed server role and the db_owner and db_backupoperator fixed database roles.

Create a Sample Test Database

```
USE [master]
GO

CREATE DATABASE [SQLTestDB]
GO

USE [SQLTestDB]
GO
CREATE TABLE SQLTest
   (
      ID INT NOT NULL PRIMARY KEY,
      c1 VARCHAR(100) NOT NULL,
      dt1 DATETIME NOT NULL DEFAULT getdate()
   );
GO

USE [SQLTestDB]
GO

INSERT INTO SQLTest (ID, c1) VALUES (1, 'test1')
INSERT INTO SQLTest (ID, c1) VALUES (2, 'test2')
INSERT INTO SQLTest (ID, c1) VALUES (3, 'test3')
INSERT INTO SQLTest (ID, c1) VALUES (4, 'test4')
INSERT INTO SQLTest (ID, c1) VALUES (5, 'test5')
GO

SELECT * FROM SQLTest
GO
```

Back up using MSSMS

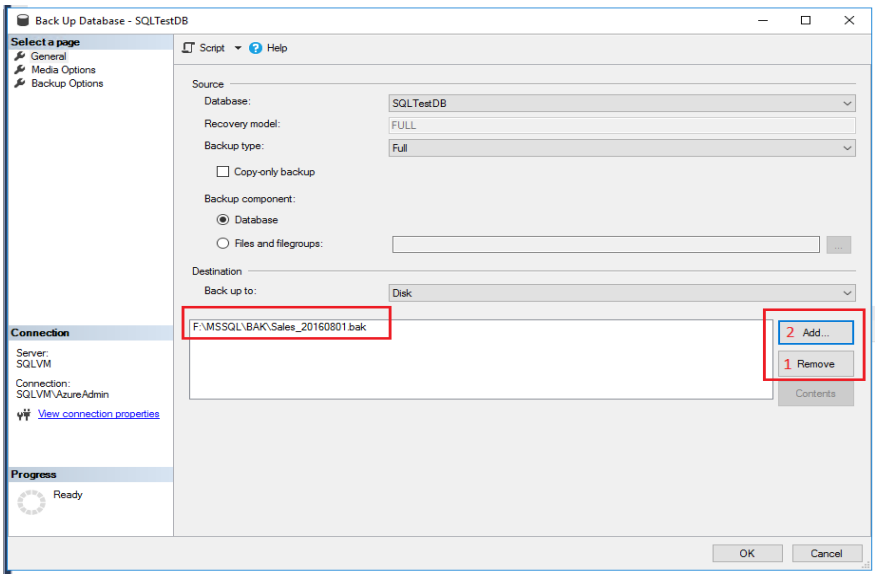


Steps of Back up using MSSMS

In this example, the `SQLTestDB` database will be backed up to disk at a location of your choice.

1. After connecting to the appropriate instance of the Microsoft SQL Server Database Engine, in **Object Explorer**, expand the server tree.
2. Expand **Databases**, right-click `SQLTestDB`, point to **Tasks**, and then click **Back Up...**.
3. On the **General** page in the **Destination** section select **Disk** from the **Back up to:** drop-down list.
4. Select **Remove** until all existing backup files have been removed.
5. Select **Add** and the **Select Backup Destination** dialog box will open.
6. Enter a valid path and file name in the **File name** text box and use **.bak** as the extension to simplify the classification of this file.
7. Click **OK** and then click **OK** again to initiate the backup.
8. When the backup completes successfully, click **OK** to close the SQL Server Management Studio dialog box.

Back up on non-default Location



Lesson 55: What are Control Files & Creating Control Files

Creating Control Files

Control Files

- What is Control Files?
- Managing Control Files
- Back up Control Files
- Creating Control Files
- Multiplex Copying of Control Files

What is Control Files?

- Control file contains
- The database name
- Names and locations of associated data files and redo log files
- The timestamp of the database creation
- The current log sequence number
- Checkpoint information

Managing Control Files?

- DB server writes into control file when DB is open.
- DB is mounted for recovery with control file.
- More control files can be created

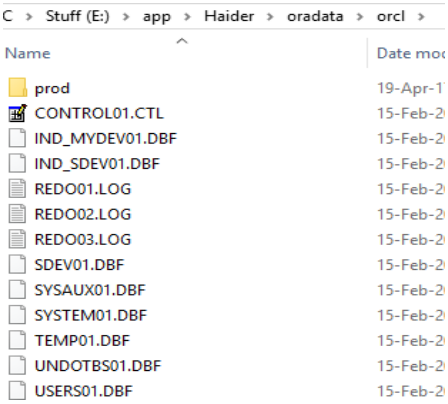
Back up Control Files?

- Adding, dropping, or renaming datafiles
- Adding or dropping a tablespace, or altering the read/write state of the tablespace
- Adding or dropping redo log files or groups

Creating Control Files?

- Creating Initial Control Files
- Creating Additional Copies, Renaming, and Relocating Control Files
- Creating New Control Files
- Initial Control Files are

CONTROL_FILES = (/u01/oracle/prod/control01.ctl,
/u02/oracle/prod/control02.ctl,
/u03/oracle/prod/control03.ctl)



Multiplex copying of Control Files

To add a multiplexed copy of the current control file or to rename a control file:

- Shut down the database.
- Copy an existing control file to a new location, using operating system commands.
- Edit the CONTROL_FILES parameter in the database initialization parameter file to add the new control file name, or to change the existing control filename.
- Restart the database.

Lesson 56: Back up Control Files

Back up and New Control Files

- When to Create Control Files?
- Create Control File Statement
- Data Dictionary Views for Control Files?
- Using Data Dictionary Views with user sys
- SQL of V$LOGFILE
- SQL of V$DATAFILE
- SQL of V$CONTROLFILE
- SQL of V$PARAMETER
- Backing up Control Files?
- Steps in creating new Control Files?

When to Create Control Files?

- All control files for the database have been permanently damaged and you do not have a control file backup.
- You want to change the database name.

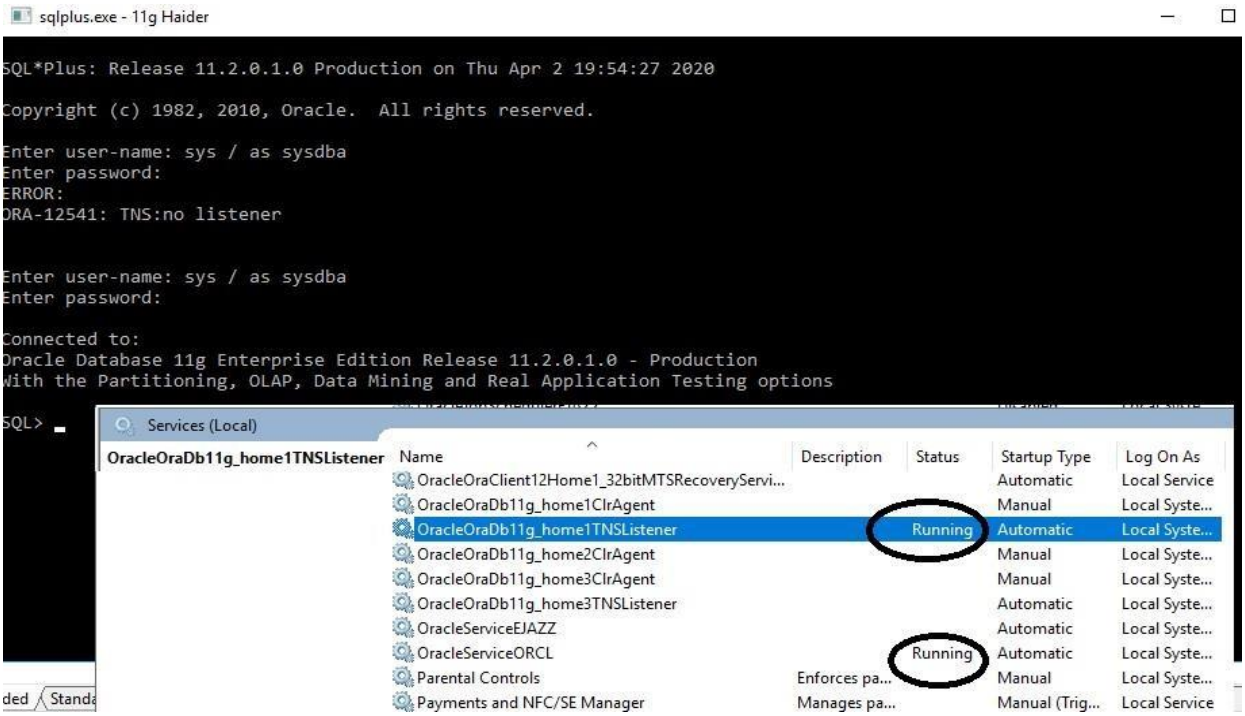
Create Control File Statement

```
CREATE CONTROLFILE
   SET DATABASE prod
   LOGFILE GROUP 1 ('/u01/oracle/prod/redo01_01.log',
                    '/u01/oracle/prod/redo01_02.log'),
           GROUP 2 ('/u01/oracle/prod/redo02_01.log',
                    '/u01/oracle/prod/redo02_02.log'),
           GROUP 3 ('/u01/oracle/prod/redo03_01.log',
                    '/u01/oracle/prod/redo03_02.log')
   RESETLOGS
   DATAFILE '/u01/oracle/prod/system01.dbf' SIZE 3M,
            '/u01/oracle/prod/rbs01.dbs' SIZE 5M,
            '/u01/oracle/prod/users01.dbs' SIZE 5M,
            '/u01/oracle/prod/temp01.dbs' SIZE 5M
   MAXLOGFILES 50
   MAXLOGMEMBERS 3
   MAXLOGHISTORY 400
   MAXDATAFILES 200
   MAXINSTANCES 6
   ARCHIVELOG;
```

Data Dictionary Views for Control Files?

- SELECT MEMBER FROM V$LOGFILE;
- SELECT NAME FROM V$DATAFILE;
- SELECT VALUE FROM V$PARAMETER WHERE NAME = 'control_files'

Using Data Dictionary Views with user sys



SQL of V$LOGFILE

```
SQL> SELECT MEMBER FROM V$LOGFILE;

MEMBER
-------------------------------------
E:\APP\HAIDER\ORADATA\ORCL\REDO03.LOG
E:\APP\HAIDER\ORADATA\ORCL\REDO02.LOG
E:\APP\HAIDER\ORADATA\ORCL\REDO01.LOG

SQL>
```

```
SQL> SELECT NAME FROM V$DATAFILE;

NAME
-----------------------------------------------------
E:\APP\HAIDER\ORADATA\ORCL\SYSTEM01.DBF
E:\APP\HAIDER\ORADATA\ORCL\SYSAUX01.DBF
E:\APP\HAIDER\ORADATA\ORCL\UNDOTBS01.DBF
E:\APP\HAIDER\ORADATA\ORCL\USERS01.DBF
E:\APP\HAIDER\ORADATA\ORCL\IND_MYDEV01.DBF
E:\APP\HAIDER\ORADATA\ORCL\SDEV01.DBF
E:\APP\HAIDER\ORADATA\ORCL\IND_SDEV01.DBF
E:\APP\HAIDER\ORADATA\ORCL\PROD\PDEV01.DBF
E:\APP\HAIDER\ORADATA\ORCL\PROD\PDEV02.DBF
E:\APP\HAIDER\ORADATA\ORCL\PROD\PDEV03.DBF
E:\APP\HAIDER\ORADATA\ORCL\PROD\PIND_DEV.DBF

NAME
-----------------------------------------------------
E:\APP\HAIDER\ORADATA\ORCL\PROD\PTSIMG01.DBF
E:\APP\HAIDER\ORADATA\ORCL\PROD\PTSIMG02.DBF
E:\APP\HAIDER\ORADATA\ORCL\PROD\PTSIMG03.DBF
E:\APP\HAIDER\ORADATA\ORCL\PROD\PTSIMG04.DBF
E:\APP\HAIDER\ORADATA\ORCL\PROD\PSTUCRSREG01.DBF
E:\APP\HAIDER\ORADATA\ORCL\PROD\PSTUCRSREG02.DBF
E:\APP\HAIDER\ORADATA\ORCL\PROD\PEMARKS01.DBF
E:\APP\HAIDER\ORADATA\ORCL\PROD\PEMARKS02.DBF
E:\APP\HAIDER\ORADATA\ORCL\PROD\TSIMG01.DBF
E:\APP\HAIDER\ORADATA\ORCL\PROD\TSIMG02.DBF
E:\APP\HAIDER\PRODUCT\11.2.0\DBHOME_1\ORADATA\NOVUS01.DBF
```

SQL of V$CONTROLFILE

With default installation, oracle contains only two control files

```
SQL> SELECT NAME FROM V$CONTROLFILE;

NAME
------------------------------------------------------------
E:\APP\HAIDER\ORADATA\ORCL\CONTROL01.CTL
E:\APP\HAIDER\FLASH_RECOVERY_AREA\ORCL\CONTROL02.CTL

SQL>
```

SQL of V$PARAMETER

```
SQL>
SQL> SELECT VALUE
  2  FROM V$PARAMETER
  3  WHERE NAME = 'control_files';

VALUE
--------------------------------------------------------------------------------
E:\APP\HAIDER\ORADATA\ORCL\CONTROL01.CTL, E:\APP\HAIDER\FLASH_RECOVERY_AREA\ORCL
\CONTROL02.CTL
```

```
SQL> show parameter control_files

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
control_files                        string      E:\APP\HAIDER\ORADATA\ORCL\CON
                                                 TROL01.CTL, E:\APP\HAIDER\FLAS
                                                 H_RECOVERY_AREA\ORCL\CONTROL02
                                                 .CTL
```

Backing up Control Files?

- Back up the control file to a binary file (duplicate of existing control file) using the following statement:

ALTER DATABASE BACKUP CONTROLFILE TO '/oracle/backup/control.bkp';

- Produce SQL statements that can later be used to re-create your control file:

ALTER DATABASE BACKUP CONTROLFILE TO TRACE;

Steps in creating new Control Files?

- Shut down the database.

If the database is open, shut down the database normally if possible. Use the

IMMEDIATE or ABORT clauses only as a last resort.

- Back up all data files and redo log files of the database.
- Start up a new instance, but do not mount or open the database: STARTUP NOMOUNT
- Create a new control file for the database using the CREATE CONTROLFILE statement.

Executing steps in creating new Control Files?

login sqlplus with user: sys / as sysdba select name from v$controlfile;

Alter database backup controlfile to trace as 'E:\APP\HAIDER\ORADATA\ORCL\control.bkp';

alter system set control_files='E:\APP\HAIDER\ORADATA\ORCL\control01.ctl',

'E:\APP\HAIDER\FLASH_RECOVERY_AREA\ORCL\control02.ctl',

'E:\APP\HAIDER\ORADATA\ORCL\control03.ctl' scope=spfile;

shutdown immediate

exit

Use copy-past file in windows file explorer or use DOS command

E:\APP\HAIDER\ORADATA\ORCL>copy control01.ctl control03.ctl

sqlplus

Enter user-name: sys / as sysdba

Enter password:

Connected to an idle instance.

SQL> startup

ORACLE instance started.

shutdown immediate

exit

SQL>select name from v$controlfile;

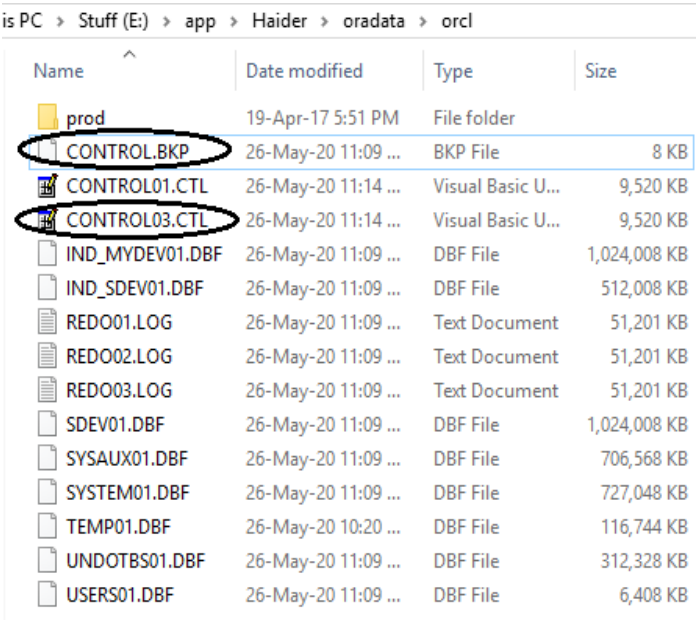
NAME

E:\APP\HAIDER\ORADATA\ORCL\CONTROL01.CTL

E:\APP\HAIDER\FLASH_RECOVERY_AREA\ORCL\CONTROL02.CTL

E:\APP\HAIDER\ORADATA\ORCL\CONTROL03.CTL

Executing steps in creating new Control Files?



Lesson 57: Trouble Shouting Control Files

- SQL of V$CONTROLFILE
- Executing steps in creating new Control Files
- Steps in creating new Control Files
- Executing Commands on SQL
- Executing steps in creating new Control Files
- Backup Control Files using Startup Mount
- Alter System Set Command & initialization File
- Executing steps in creating new
- Control Files using SQL
- Use of Database MOUNT & NO MOUNT
- Database is in normal mode (consistent database)
- Executing steps in creating new Control Files
- Warning for storing Control Files

When to Create Control Files?

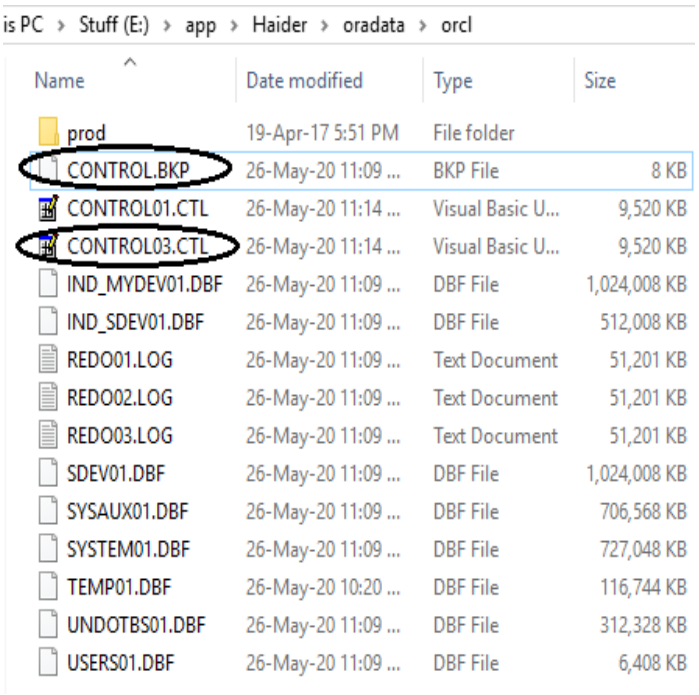
- All control files for the database have been permanently damaged and you do not have a control file backup.
- You want to change the database name.

SQL of V$CONTROLFILE

With default installation, oracle contains only two control files

```
SQL> SELECT NAME FROM V$CONTROLFILE;

NAME
--------------------------------------------------------------------------------
E:\APP\HAIDER\ORADATA\ORCL\CONTROL01.CTL
E:\APP\HAIDER\FLASH_RECOVERY_AREA\ORCL\CONTROL02.CTL

SQL>
```

Executing steps in creating new Control Files



Steps in creating new Control Files

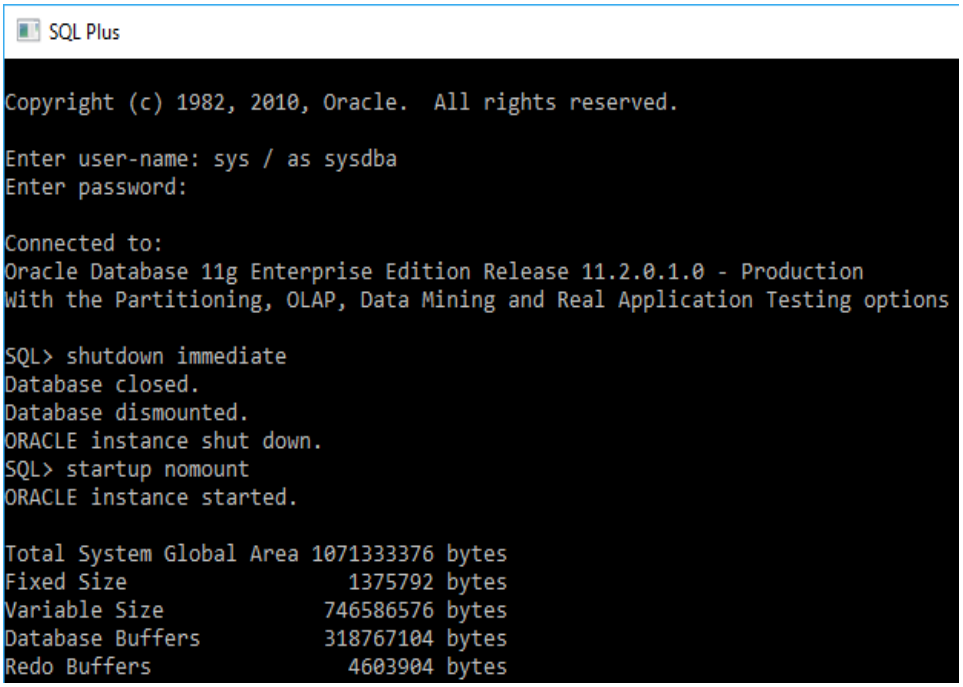
- Shut down the database.

If the database is open, shut down the database normally if possible. Use the

IMMEDIATE or ABORT clauses only as a last resort.

- Back up all data files and redo log files of the database.
- Start up a new instance, but do not mount or open the database: STARTUP NOMOUNT
- Create a new control file for the database using the CREATE CONTROLFILE statement.

Executing Commands on SQL



Executing steps in creating new Control Files

login sqlplus with user: sys / as sysdba

select name from v$controlfile;

Alter database backup controlfile to trace as 'E:\APP\HAIDER\ORADATA\ORCL\control.bkp';

Alter system set control_files='E:\APP\HAIDER\ORADATA\ORCL\control01.ctl',

'E:\APP\HAIDER\FLASH_RECOVERY_AREA\ORCL\control02.ctl',

'E:\APP\HAIDER\ORADATA\ORCL\control03.ctl' scope=spfile;

shutdown immediate

exit

Lesson 58: Recover & Dropping Control Files

- Checking for Missing or Extra Files
- Handling Errors During CREATE CONTROLFILE
- ORA Error - A Sample
- Different Locations of Control Files
- Executing Commands on SQL
- Backing up Control Files
- Viewing Alert Log using Data Dictionary
- Viewing Alert Log using Enterprise Manager
- Finding Trace Files
- Recovery from Permanent Media Failure
- Dropping Control Files

Checking for Missing or Extra Files

- Dropped control files are still shown through data dictionary.
- MISSINGnnnn is a flag in control file, gives status of file when seen by the data dictionary but actually does not exist on disk.
- If the actual data file corresponding to MISSINGnnnn is read-only or offline normal, then you can make the data file accessible by renaming MISSINGnnnn to the name of the actual data file.
- If MISSINGnnnn corresponds to a data file that was not read-only or offline normal, then you cannot use the rename operation to make the data file accessible, because the data file requires media recovery that is precluded by the results of RESETLOGS.
- In this case, you must drop the tablespace containing the data file.
- Conversely, if a data file listed in the control file is not present in the data dictionary, then the database removes references to it from the new control file. Refers to Alert log in any case.

Handling Errors During CREATE CONTROLFILE

If Oracle Database sends you an error (usually error ORA-01173, ORA-01176, ORA-01177, ORA-01215, or ORA-01216) when you attempt to mount and open the database after creating a new control file, the most likely cause is that you omitted a file from the CREATE CONTROLFILE statement or included one that should not have been listed.

ORA Error – A Sample

Oracle Error: ORA-01173

Error Description:

Data dictionary indicates missing data file from system table space

Error Cause:

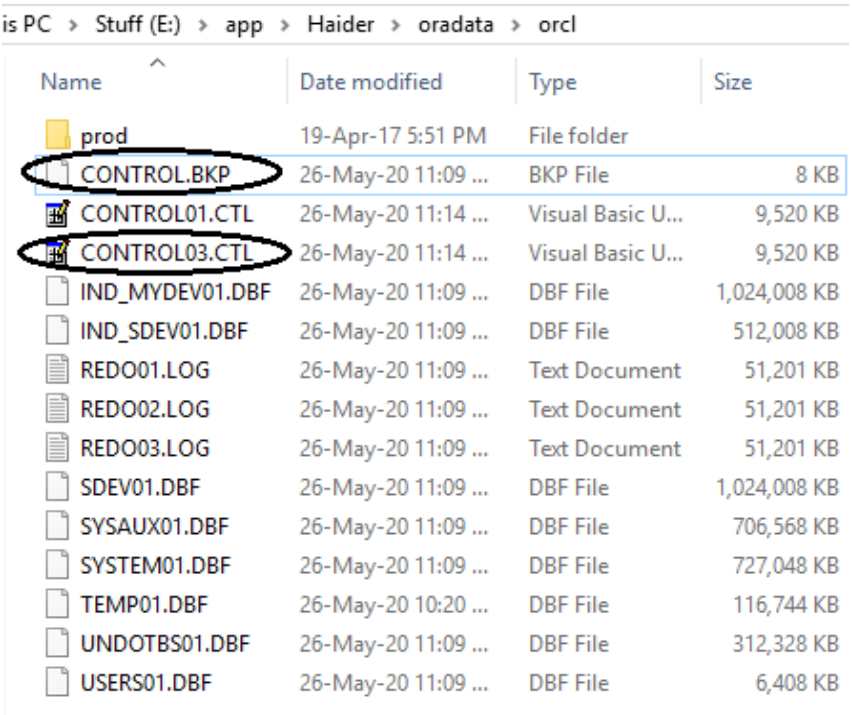
Either the database has been recovered to a point in time in the future of the control file or a datafile from the system tablespace was omitted from the create control file command previously issued.

Action:

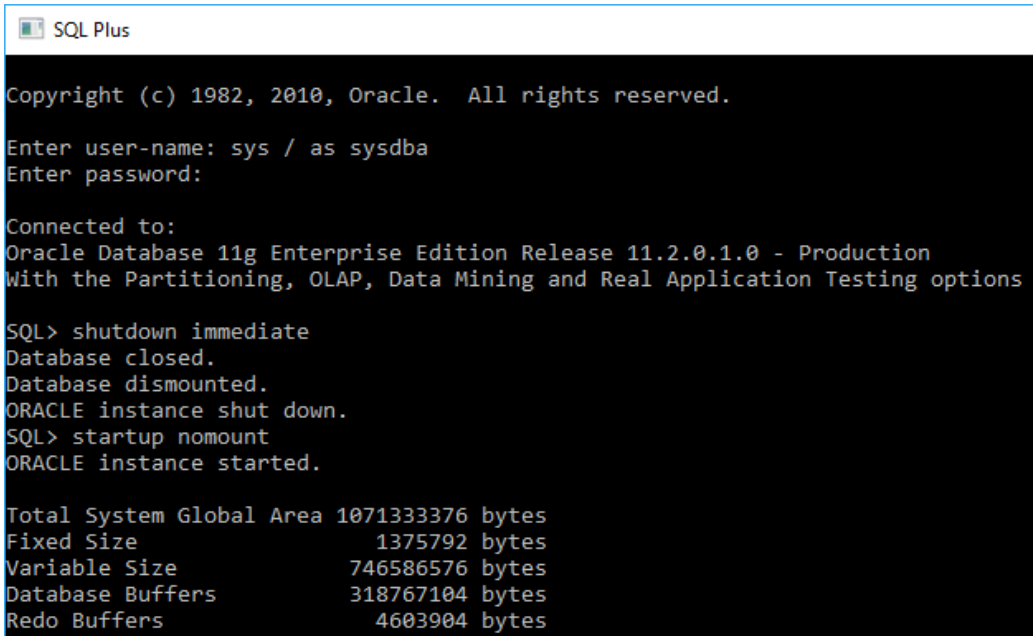
For the former problem you need to recover the database from a more recent control file. For the later problem, simply recreate the control file checking to be sure that you include all the datafiles

in the system tablespace.

Different Locations of Control Files



Executing Commands on SQL



Backing up Control Files

login sqlplus with user: sys / as sysdba

select name from v$controlfile;

Alter database backup controlfile to trace as 'E:\APP\HAIDER\ORADATA\ORCL\control.bkp';

Produce SQL statements that can later be used to re-create your control file:

ALTER DATABASE BACKUP CONTROLFILE TO TRACE;

- This command writes a SQL script to a trace file where it can be captured and edited to reproduce the control file.
- View the alert log to determine the name and location of the trace file.

Viewing Alert Log using Data Dictionary

```
SQL> SELECT substr(Inst_id,1, 3),
  2  substr(name, 1, 25) Diag_name,
  3  substr(value, 1, 100) Diag_value
  4  from V$DIAG_INFO;

SUB DIAG_NAME                 DIAG_VALUE
--- ------------------------- ----------------------------------------------------------------
----------
1   Diag Enabled              TRUE
1   ADR Base                  e:\app\haider
1   ADR Home                  e:\app\haider\diag\rdbms\orcl\orcl
1   Diag Trace                e:\app\haider\diag\rdbms\orcl\orcl\trace
1   Diag Alert                e:\app\haider\diag\rdbms\orcl\orcl\alert
1   Diag Incident             e:\app\haider\diag\rdbms\orcl\orcl\incident
1   Diag Cdump                e:\app\haider\diag\rdbms\orcl\orcl\cdump
1   Health Monitor            e:\app\haider\diag\rdbms\orcl\orcl\hm
1   Default Trace File        e:\app\haider\diag\rdbms\orcl\orcl\trace\orcl_ora_25104.trc
1   Active Problem Count      0
1   Active Incident Count     0

11 rows selected.
```

Viewing Alert Log using Enterprise Manager

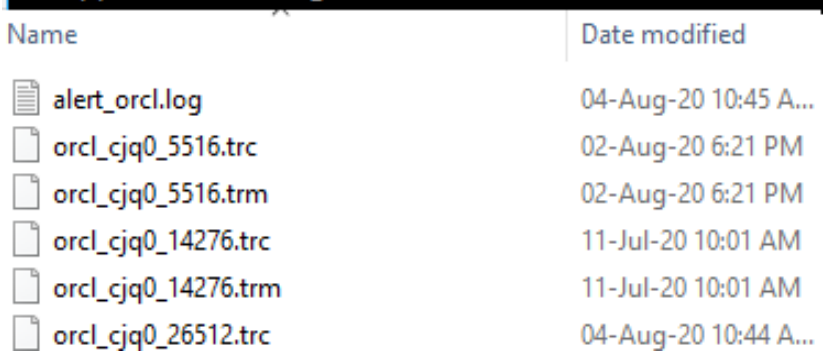
- Access the Database Home page in Enterprise Manager.
- Under Related Links, click Alert Log Contents.
- The View Alert Log Contents page appears.
- Select the number of entries to view, and then click Go

Finding Trace Files

Trace files are stored in the Automatic Diagnostic Repository (ADR), in the trace directory under each ADR home.

- To find the trace file for your current session: SELECT VALUE FROM V$DIAG_INFO WHERE NAME = 'Default Trace File';
- To find all trace files for the current instance: SELECT VALUE FROM V$DIAG_INFO WHERE NAME = 'Diag Trace'; The path to the ADR trace directory for the current instance is returned.
- To determine the trace file for each Oracle Database process: SELECT PID, PROGRAM, TRACEFILE FROM V$PROCESS;

```
SQL> SELECT VALUE FROM V$DIAG_INFO WHERE NAME = 'Diag Trace';

VALUE
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
e:\app\haider\diag\rdbms\orcl\orcl\trace
```

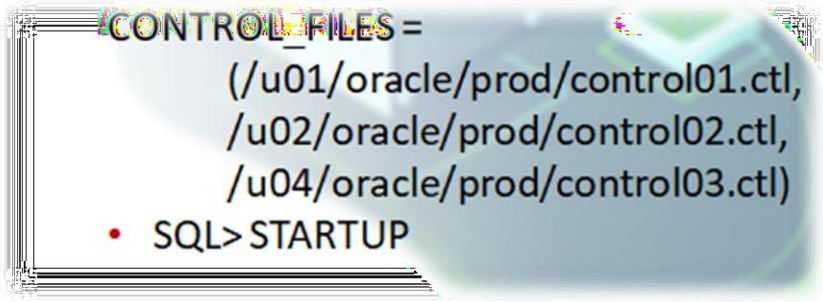


Recovery from Permanent Media Failure

- With the instance shut down, use an operating system command of copy

% cp /u01/oracle/prod/control01.ctl /u04/oracle/prod/control03.ctl

- Edit the CONTROL_FILES parameter in the initialization parameter file to replace the bad location with the new location:



Dropping Control Files

Database should have at least two control files at all times.

- Shut down the database.
- Edit the CONTROL_FILES parameter in the database initialization parameter file to delete the old control file name.
- Restart the database.

Lesson 59: What are Redo Log Files?

Redo Log Files?

- What is the Redo Log?
- Redo Threads
- Redo Log Contents
- Writing to Redo Log Files
- Active & In-active Redo Log Files
- Overwriting Redo Log Files

What is the Redo Log?

- The most crucial structure for recovery operations is the redo log
- The structure consists of two or more pre-allocated files that store all changes made to the database as they occur.
- Every instance of an Oracle Database has an associated redo log to protect the database in case of an instance failure.
- Redo Log files stores all the change information for the database and are used by Oracle during Database recovery

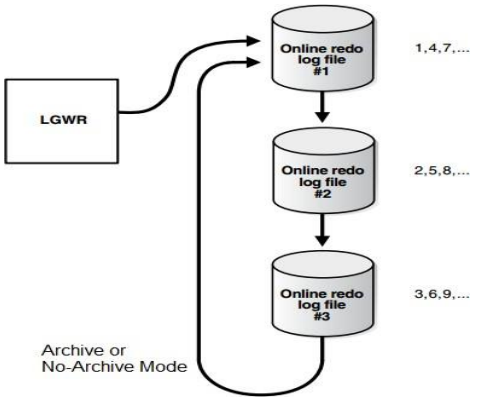
Redo Threads?

- When speaking in the context of multiple database instances, the redo log for each database instance is also referred to as a redo thread.

- In typical configurations, only one database instance accesses an Oracle Database, so only one thread is present.

Redo Log Contents?

- Redo log files are filled with redo records or redo entry, is made up of a group of change vectors, each of which is a description of a change made to a single block in the database.
- For example, if you change a salary value in an employee table, you generate a redo record containing change vectors that describe changes to the data segment block for the table, the undo segment data block, and the transaction table of the undo segments.
- Redo records are buffered in a circular fashion in the redo log buffer of the SGA.
- Redo records are written to one of the redo log files by the Log Writer (LGWR) database background process.
- Whenever a transaction is committed, LGWR writes the transaction redo records from the redo log buffer of the SGA to a redo log file, and assigns a system change number (SCN) to identify the redo records for each committed transaction.
- Transactions are committed when redo records are written to the disk safely.



Writing to Redo Log files

- LGWR writes to redo log files in a circular fashion. When the current redo log file fills,
- LGWR begins writing to the next available redo log file. When the last available redo log file is filled, LGWR returns to the first redo log file and writes to it, starting the cycle again.

Active & In-Active Redo Log files

- Oracle Database uses only one redo log file at a time to store redo records written from the redo log buffer.
- The redo log file that LGWR is actively writing to is called the current redo log file.
- Redo log files that are required for instance recovery are called active redo log files.
- Redo log files that are no longer required for instance recovery are called inactive redo log files.

Overwriting Redo Log files

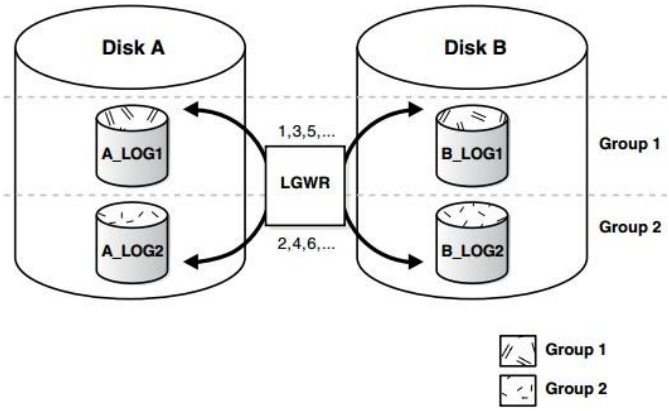
- If you have enabled archiving (the database is in ARCHIVELOG mode), then the database cannot reuse or overwrite an active online log file until one of the archiver background processes (ARCn) has archived its contents.
- A log switch is the point at which the database stops writing to one redo log file and begins writing to another. Normally, a log switch occurs when the current redo log file is completely filled and writing must continue to the next redo log file.
- Oracle Database assigns each redo log file a new log sequence number every time a log switch occurs and LGWR begins writing to it.

Lesson 60: Planning Redo Log Files

- Multiplexing Redo Log Files?
- Example of Multiplexing
- Legal & Illegal Multiplexed Redo Log Configuration
- Placing Redo Logs on Different Locations
- Adding Redo Log Files to a Group

Multiplexing Redo Log Files

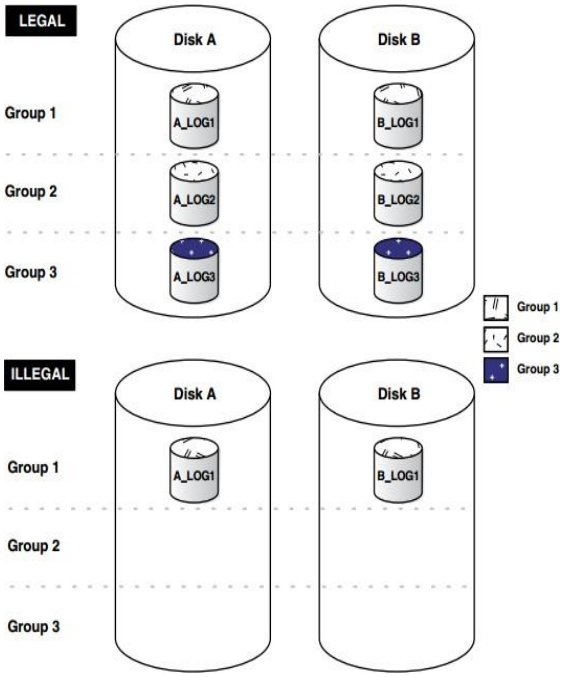
- To protect the failure of Redo Log Files
- Two or more identical copies of Redo Log Files maintained in separate locations.
- Files at same location can cause I/O errors, file corruption etc.
- LGWR concurrently writes the same information to multiple identical Redo Log files
- A group consists of a Redo Log file and its multiplexed copies.
- Each identical copy is said to be a member of the group. Groups can be group 1, group 2 and so on



Example of Multiplexing

First LGWR writes concurrently to both A_LOG1 and B_LOG1. Then it writes concurrently to both A_LOG2and B_LOG2, and so on. LGWR never writes concurrently to members of different groups (for example, to A_LOG1and B_LOG2)

Legal & Illegal Multiplexed Redo Log Configuration



Placing Redo Log on Different Locations?

- When setting up a multiplexed redo log, place members of a group on different physical disks. If a single disk fails, then only one member of a group becomes unavailable to LGWR and other members remain accessible to LGWR, so the instance can continue to function.
- Minimum size permitted to Redo Log File is 4MB.

```
SQL> SELECT name, value FROM v$sysstat WHERE name = 'redo wastage';

NAME                                                                  VALUE
---------------------------------------------------------------- ----------
redo wastage                                                         4700476
```

Planning the Block Size of Redo Log Files

- Unlike the database block size, which can be between 2K and 32K, redo log files always default to a block size that is equal to the physical sector size of the disk. Historically, this has typically been 512 bytes (512B).
- Some newer high-capacity disk drives offer 4Kbyte (4K) sector sizes for both increased ECC capability and improved format efficiency.

Adding Redo Log File to a Group

ALTER DATABASE orcl ADD LOGFILE

GROUP 4

('/u01/logs/orcl/redo04a.log',

'/u01/logs/ orcl/redo04b.log')

SIZE 100M BLOCKSIZE 512 REUSE;

```
SQL> select group#, members, BLOCKSIZE FROM V$LOG;

    GROUP#    MEMBERS  BLOCKSIZE
---------- ---------- ----------
         1          1        512
         2          1        512
         3          1        512

SQL>
```

Difference between 512B and 4B will be a wastage

Lesson 61: Creating Redo Log Files Groups

Creating Redo Log Files Groups & Members

- Creating Redo Log Files Groups
- Specify the number with a Group
- Creating Redo Log Members
- Alternate way of Creating Redo Log Members
- To create a new group of redo log files, use the SQL statement ALTER DATABASE with the ADD LOGFILE clause.
- The following statement adds a new group of redo logs to the database:

```
ALTER DATABASE

ADD LOGFILE

('/oracle/dbs/log1c.rdo', '/oracle/dbs/log2c.rdo')

SIZE 100M;
```

Specify the number with a Group

- To a group number of redo log files, use the SQL statement ALTER DATABASE with the ADD LOGFILE clause.

```
ALTER DATABASE

ADD LOGFILE GROUP 10 ('/oracle/dbs/log1c.rdo', '/oracle/dbs/log2c.rdo')

SIZE 100M BLOCKSIZE 512;
```

- Group number must be between 1 and MAXLOGFILES. Do not skip redo log file group, you will consume unnecessary space in the control files of the database.

Creating Redo Log Members

- To create new redo log members for an existing group, use the SQL statement ALTER DATABASE with the ADD LOGFILE MEMBER clause. The following statement adds a new redo log member to redo log group number 2:

```
ALTER DATABASE

ADD LOGFILE MEMBER '/oracle/dbs/log2b.rdo' TO GROUP 2;
```

- Sometimes, some members of groups dropped due to disk failure, you can add new members. Use the following SQL

```
ALTER DATABASE

ADD LOGFILE MEMBER '/oracle/dbs/log2b.rdo'

TO GROUP 2;
```

Alternate way of Creating Redo Log Members

- When using the ALTER DATABASE statement, you can alternatively identify the target group by specifying all of the other members of the group in the TO clause, as shown in the following example:

```
ALTER DATABASE ADD LOGFILE MEMBER '/oracle/dbs/log2c.rdo'

TO ('/oracle/dbs/log2a.rdo', '/oracle/dbs/log2b.rdo');
```

Lesson 62: Relocating and Renaming Redo Log Members

- Reasons for Relocating Logs
- Renaming Redo Log Members
- What to do before Relocation?
- Steps for Relocating Logs
- Metadata of Redo Logs

Reasons for Relocating Redo Log

- You can use operating system commands to relocate redo logs, then use the ALTER DATABASE statement to make their new names (locations) known to the database.
- This procedure is necessary, for example, if the disk currently used for some redo log files is going to be removed, or if data files and several redo log files are stored on the same disk and should be separated to reduce contention.

Renaming Redo Log Members

- To rename redo log members, you must have the ALTER DATABASE system privilege.
- Additionally, you might also need operating system privileges to copy files to the desired location and privileges to open and back up the database.

What to do before Relocations?

- Before relocating your redo logs, or making any other structural changes to the database, completely BACKUP the database in case you experience problems while performing the operation. As a precaution, after renaming or relocating a set of redo log files, immediately back up the database control file.

This step is very important and DBA must manage throughout his/ her tasks

Steps for Relocating Redo Logs?

- The log files are located on two disks: disk a and disk b.
- The redo log is duplexed: one group consists of the members /diska/logs/log1a.rdo and /diskb/logs/log1b.rdo, and the second group consists of the members /diska/logs/log2a.rdo and /diskb/logs/log2b.rdo.
- The redo log files located on disk a must be relocated to disk c. The new filenames will reflect the new location: /diskc/logs/log1c.rdo and /diskc/logs/log2c.rdo

Metadata of Redo Logs?

- The redo records have all relevant metadata for the change, including the following:
- SCN and time stamp of the change
- Transaction ID of the transaction that generated the change
- SCN and time stamp when the transaction committed (if it committed)
- Type of operation that made the change
- Name and type of the modified data segment: SCN: System Change Number

Lesson 63: Dropping Redo Log Groups & Members

Dropping Redo Log Groups & Members

- Cases What to Do?
- Dropping Redo Log Groups
- Dropping Redo Log Members

Cases What to Do?

- In some cases, you may want to drop an entire group of redo log members. For example, you want to reduce the number of groups in an instance redo log.
- In a different case, you may want to drop one or more specific redo log members. For example, if a disk failure occurs, you may need to drop all the redo log files on the failed disk so that the database does not try to write to the inaccessible files.

- In other situations, particular redo log files become unnecessary. For example, a file might be stored in an inappropriate location.

Dropping Redo Log Groups?

- To drop a redo log group, you must have the ALTER DATABASE system privilege. Before dropping a redo log group, consider the following restrictions and precautions:
- An instance requires at least two groups of redo log files, regardless of the number of members in the groups. (A group comprises one or more members.)
- You can drop a redo log group only if it is inactive. If you must drop the current group, then first force a log switch to occur.
- Make sure a redo log group is archived (if archiving is enabled) before dropping it.
- To see whether this has happened, use the V$LOGview.

SELECT GROUP#, ARCHIVED, STATUS FROM V$LOG;

GROUP# ARC STATUS

1 YES ACTIVE

2 NO CURRENT

3 YES INACTIVE

4 YES INACTIVE

Drop a redo log group with the SQL statement ALTER DATABASE with the DROP LOGFILE clause.

- Drop a redo log group with the SQL statement ALTER DATABASEwith the DROP LOGFILE clause.
- The following statement drops redo log group number 3:

When a redo log group is dropped from the database, and you are not using the Oracle Managed Files feature, the operating system files are not deleted from disk. Rather, the control files of the associated database are updated to drop the members of the group from the database structure. After dropping a redo log group, ensure that the drop completed successfully, and then use the appropriate operating system command to delete the dropped redo log files.

ALTER DATABASE DROP LOGFILE GROUP 3;

After dropping a redo log group, ensure that the drop completed successfully, and then use the appropriate operating system command to delete the dropped redo log files.

When a redo log group is dropped from the database, and you are not using the Oracle Managed Files feature, the operating system files are not deleted from disk. Rather, the control files of the associated database are updated to drop the members of the group from the database structure. Consider the following restrictions and precautions before dropping individual redo log members:

- It is permissible to drop redo log files so that a multiplexed redo log becomes temporarily asymmetric. For example, if you use duplexed groups of redo log files, you can drop one member of one group, even though all other groups have two members each.
- However, you should rectify this situation immediately so that all groups have at least two members, and thereby eliminate the single point of failure possible for the redo log.
- An instance always requires at least two valid groups of redo log files, regardless of the number of members in the groups. (A group comprises one or more members.) If the member you want to drop is the last valid member of the group, you cannot drop the member until the other members become valid.
- To see a redo Forcing Log Switches log file status, use the V$LOGFILE view.
- You can drop a redo log member only if it is not part of an active or current group. o drop a member of an active group, first force a log switch to occur.
- Make sure the group to which a redo log member belongs is archived (if archiving is enabled) before dropping the member. To see whether this has happened, use the V$LOG view.
- To drop specific inactive redo log members, use the following statement:

ALTER DATABASE DROP LOGFILE MEMBER '/oracle/dbs/log3c.rdo';

Lesson 64: Forcing Log Switches Between Redo Log Groups & Members

Forcing Log Switches

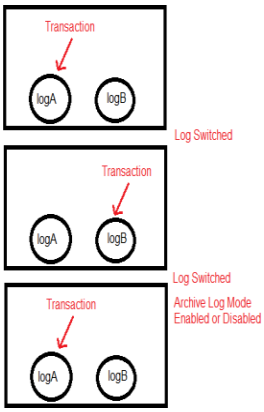
- Log Switching
- Log Switching – An Example
- Verifying Blocks in Redo Log Files

Log Switching

- A log switch occurs when LGWR stops writing to one redo log group and starts writing to another. By default, a log switch occurs automatically when the current redo log file group fills.
- You can force a log switch to make the currently active group inactive and available for redo log maintenance operations. For example, you want to drop the currently active group, but are not able to do so until the group is inactive.
- You may also want to force a log switch if the currently active group must be archived at a specific time before the members of the group are completely filled. This option is useful in configurations ith large redo log files that take a long time to fill.
- The following statement forces a log switch:

ALTER SYSTEM SWITCH LOGFILE;

Log Switching – An Example



In 3rd step, if archival mode is disabled. Log will be overwritten and no backup of redo log will be maintained. Otherwise, before switching to logA (when archival mode is enabled), backup of logA will be taken in ARCm files

Verifying Blocks in Redo Log Files

- You can configure the database to use checksums to verify blocks in the redo log files.
- If you set the initialization parameter DB_BLOCK_CHECKSUM to TYPICAL(the default), the database computes a check sum for each database block when it is written to disk, including each redo log block as it is being written to the current log.
- The checksum is stored header's block.
- Oracle Database uses the checksum to detect corruption in a redo log block.
- The database verifies the redo log block when the block is read from an archived log during recovery and when it writes the block to an archive log file. An error is raised and written to the alert log if corruption is detected.
- If corruption is detected in a redo log block while trying to archive it, the system attempts to read the block from another member in the group.
- If the block is corrupted in all members of the redo log group, then archiving cannot proceed.
- The value of the DB_BLOCK_CHECKSUM parameter can be changed dynamically using the ALTER SYSTEM statement.

Lesson 65: Clearing Redo Log Files

- How to clear Redo Log Files?
- Error in clearing Redo Log Files
- Dropping Redo Log File
- Creating Redo Log File

How to clear Redo Log Files?

- A redo log file might become corrupted while the database is open, and ultimately stop database activity because archiving cannot continue. In this situation the ALTER DATABASE CLEAR LOGFILE statement can be used to reinitialize the file without shutting down the database.
- The following statement clears the log files in redo log group number 3:
  - ALTER DATABASE CLEAR LOGFILE GROUP 3;
- If you clear a log file that is needed for recovery of a backup, then you can no longer recover from that backup. The database writes a message in the alert log describing the backups from which you cannot recover.
- To clear an unarchived redo log that is needed to bring an offline tablespace online, use the UNRECOVERABLE DATAFILE clause in the ALTER DATABASE CLEAR LOGFILE statement.

Error in clearing Redo Log Files

- SQL> alter database drop logfile group 1;
- alter database drop logfile group 1
- *
- ERROR at line 1:
- ORA-01624: log 1 needed **for** crash recovery of instance SBIP18DB (thread 1)
- ORA-00312: online log 1 thread 1: '/SIBIP16/SBIP18DB/SBIP18DB/redo01.log'

Error in clearing Redo Log Files

select l.group#, l.thread#,

substr(f.member,1,50) f_name,

l.archived,

l.status,

(bytes/1024/1024) fsize

from

v$log l, v$logfile f

where f.group# = l.group#

order by 1,2;

```
D# F_NAME                                          ARC STATUS    FSIZE
-- ---------------------------------------------- --- ---------- -----
 1 E:\APP\HAIDER\ORADATA\ORCL\REDO01.LOG           NO  INACTIVE      50
 1 E:\APP\HAIDER\ORADATA\ORCL\REDO02.LOG           NO  INACTIVE      50
 1 E:\APP\HAIDER\ORADATA\ORCL\REDO03.LOG           NO  CURRENT       50
```

Current means log is filling up

Active means already filled up

InActive means going to overwrite or going to save in archives before overwriting

```
# F_NAME                                    ARC STATUS    FSIZE
- ---------------------------------------- --- ---------- -----
1 E:\APP\HAIDER\ORADATA\ORCL\REDO01.LOG     NO  CURRENT       50
1 E:\APP\HAIDER\ORADATA\ORCL\REDO02.LOG     NO  INACTIVE      50
1 E:\APP\HAIDER\ORADATA\ORCL\REDO03.LOG     NO  ACTIVE        50
```

Error in clearing Redo Log Files

We can see the though the redolog group 1 has been archived, its status is ACTIVE. i.e this redolog is required for crash recovery or block recovery.

So to change the status from ACTIVE to INACTIVE, switch logfile multiple times, till the status is INACTIVE.

alter system **switch** logfile;

--for inactive status

alter database drop logfile group 2;

```
D# F_NAME                                  ARC STATUS    FSIZE
-- --------------------------------------- --- ---------- -----
 1 E:\APP\HAIDER\ORADATA\ORCL\REDO01.LOG NO  ACTIVE       50
 1 E:\APP\HAIDER\ORADATA\ORCL\REDO02.LOG NO  CURRENT      50
 1 E:\APP\HAIDER\ORADATA\ORCL\REDO03.LOG NO  INACTIVE     50
```

Error in clearing Redo Log Files?

We can see the though the redolog group 1 has been archived, its status is ACTIVE. i.e this redolog is required for crash recovery or block recovery.

So to change the status from ACTIVE to INACTIVE, switch logfile multiple times, till the status is INACTIVE.

alter system **switch** logfile;

--for inactive status

alter database drop logfile group 3;

```
SQL> alter database drop logfile group 3;

Database altered.

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup
ORACLE instance started.

Total System Global Area 1071333376 bytes
Fixed Size                  1375792 bytes
Variable Size             713032144 bytes
Database Buffers          352321536 bytes
Redo Buffers                4603904 bytes
Database mounted.
Database opened.
SQL>
```

Dropping Redo Log File

You shutdown and startup database after dropping on redo group 3. Drop REDO03.LOG from folder.

Creating Redo Log File

alter database add logfile group 3

'E:\app\Haider\oradata\orcl\REDO03.log'

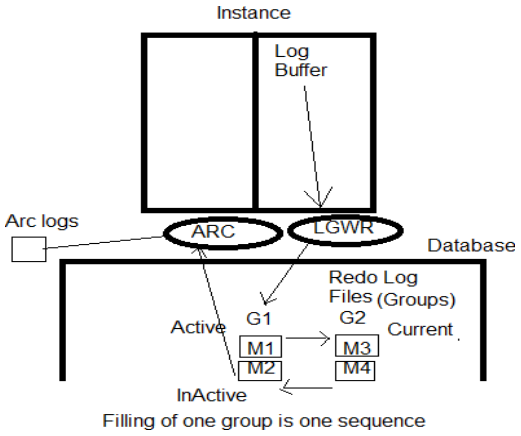
size 50m;

```
D# F_NAME                                  ARC STATUS    FSIZE
-- --------------------------------------- --- ---------- -----
 1 E:\APP\HAIDER\ORADATA\ORCL\REDO01.LOG   NO  INACTIVE     50
 1 E:\APP\HAIDER\ORADATA\ORCL\REDO02.LOG   NO  CURRENT      50
 1 E:\APP\HAIDER\ORADATA\ORCL\REDO03.LOG   YES UNUSED       50
```

Lesson 66: Metadata View of Redo Log Files

- Instance & Database Log Buffer
- Significant Views?
- Control File Information
- Log History Information
- Log of Members of Group
- Log History Information
- Use of Members of Group with Status

Instance & Database Log Buffer



Significant of Views

View	Description
V$LOG	Displays the redo log file information from the control file
V$LOGFILE	Identifies redo log groups and members and member status
V$LOG_HISTORY	Contains log history information

Control File Information

```
SELECT * FROM V$LOG;

GROUP# THREAD#   SEQ   BYTES  MEMBERS  ARC STATUS     FIRST_CHANGE# FIRST_TIM
------ ------- ----- -------  -------  --- ---------  ------------- ---------
     1       1 10605 1048576        1  YES ACTIVE          11515628 16-APR-00
     2       1 10606 1048576        1  NO  CURRENT         11517595 16-APR-00
     3       1 10603 1048576        1  YES INACTIVE        11511666 16-APR-00
     4       1 10604 1048576        1  YES INACTIVE        11513647 16-APR-00
```

Current means log is filling up

Active means already filled up

InActive means going to overwrite or going to save in archives before overwriting

Members of Group

```
SELECT * FROM V$LOG;

GROUP# THREAD#    SEQ    BYTES  MEMBERS  ARC STATUS      FIRST_CHANGE# FIRST_TIM
------ ------- ----- -------- -------- --- ---------- ------------- ---------
     1       1 10605 1048576        1  YES ACTIVE           11515628 16-APR-00
     2       1 10606 1048576        1  NO  CURRENT          11517595 16-APR-00
     3       1 10603 1048576        1  YES INACTIVE         11511666 16-APR-00
     4       1 10604 1048576        1  YES INACTIVE         11513647 16-APR-00
```

Log History Information

```
SQL> select Stamp, Thread#, First_change#, First_time, Resetlogs_change#
  2  from  v$log_history
  3  where rownum<=10;

   STAMP    THREAD# FIRST_CHANGE# FIRST_TIM RESETLOGS_CHANGE#
--------- --------- ------------ --------- ----------------
951399757         1      9079208 07-AUG-17           940976
951473702         1      9104778 07-AUG-17           940976
951559221         1      9132765 08-AUG-17           940976
951645623         1      9157391 09-AUG-17           940976
951732023         1      9182645 10-AUG-17           940976
952077284         1      9204703 11-AUG-17           940976
952085754         1      9217064 15-AUG-17           940976
952164052         1      9247525 15-AUG-17           940976
952303136         1      9278762 16-AUG-17           940976
952339592         1      9307368 18-AUG-17           940976
```

Use of member with status

```
select l.group#, l.thread#,
substr(f.member,1,50) f_name,
l.archived,
l.status,
(bytes/1024/1024) fsize
from
v$log l, v$logfile f
where f.group# = l.group#
order by 1,2;
```

```
D# F_NAME                                           ARC STATUS    FSIZE
-- -------------------------------------------------- --- ---------- -----
 1 E:\APP\HAIDER\ORADATA\ORCL\REDO01.LOG              NO  INACTIVE     50
 1 E:\APP\HAIDER\ORADATA\ORCL\REDO02.LOG              NO  INACTIVE     50
 1 E:\APP\HAIDER\ORADATA\ORCL\REDO03.LOG              NO  CURRENT      50
```

Lesson 67: Archive & NoArchive of Redo Log Files

- Archived Redo Log
- Archived Redo Log Usage
- Archived Redo Log Mode
- Archived Redo Log Disabled to Enabled
- Archived Redo Log Enabled to Disabled
- Redo Logs Use in ARCHIVELOG Mode

Archived Redo Log

- Oracle Database lets you save filled groups of redo log files to one or more offline destinations, known collectively as the archived redo log.
- The process of turning redo log files into archived redo log files is called archiving. This process is only possible if the database is running in ARCHIVELOG mode.
- You can choose automatic or manual archiving
- An archived redo log file is a copy of one of the filled members of a redo log group with log sequence number.
- For example, if you are multiplexing your redo log, and if group 1 contains identical member files a_log1and b_log1, then the archiver process (ARCn) will archive one of these member files.
- Should a_log1 become corrupted, then ARCn can still archive the identical b_log1. The archived redo log contains a copy of every group created since you enabled archiving.
- When the database is running in ARCHIVELOG mode, LGWR cannot reuse and hence overwrite a redo log group until it has been archived.
- The background process RCn automates archiving operations when automatic archiving is enabled. The database starts multiple archiver processes as needed to ensure that the archiving of filled redo logs does not fall behind.

Archived Redo Log Usage

You can use archived redo logs to:

- Recover a database
- Update a standby database
- Get information about the history of a database using the LogMiner utility

Archived Redo Log Mode

Display DB Archive Mode Disable or Enable

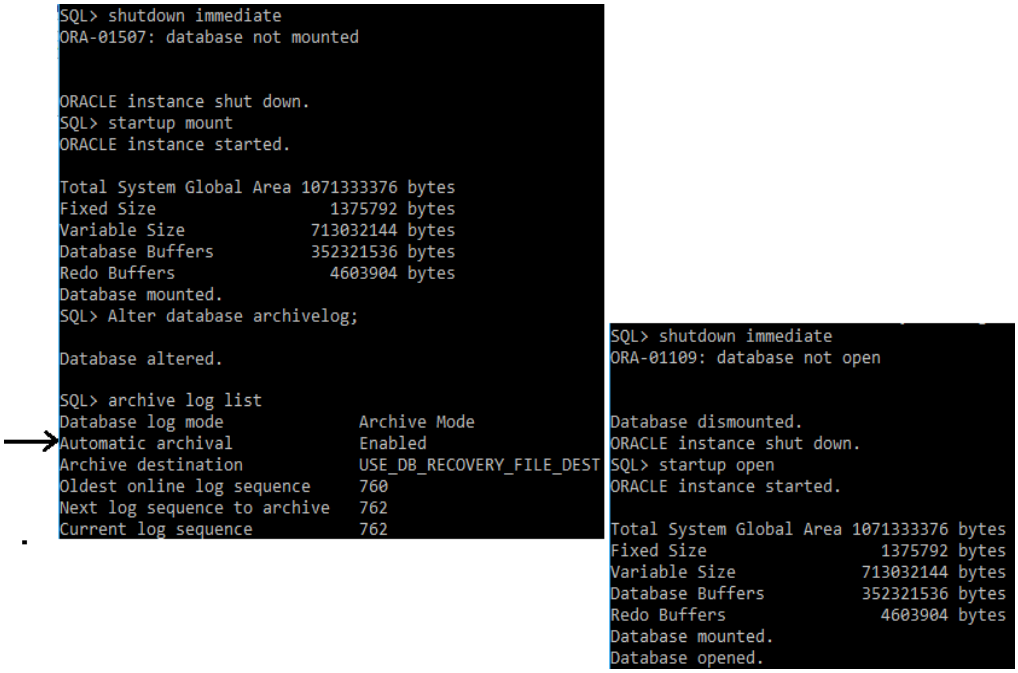
Login with sys / as sysdba.

We will discuss archival files backup later

```
SQL> archive log list
Database log mode              No Archive Mode
Automatic archival             Disabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     759
Current log sequence           761
SQL>
```

```
SQL> SHOW PARAMETER DB_RECOVERY_FILE_DEST

NAME                           TYPE        VALUE
------------------------------ ----------- ------------------------------
db_recovery_file_dest          string      E:\app\Haider\flash_recovery_a
                                           rea
db_recovery_file_dest_size     big integer 3852M
SQL>
```

Archived Redo Log Disabled to Enabled



```
SQL> shutdown immediate
ORA-01507: database not mounted


ORACLE instance shut down.
SQL> startup mount
ORACLE instance started.

Total System Global Area 1071333376 bytes
Fixed Size                  1375792 bytes
Variable Size             713032144 bytes
Database Buffers          352321536 bytes
Redo Buffers                4603904 bytes
Database mounted.
SQL> Alter database archivelog;

Database altered.

SQL> archive log list
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     760
Next log sequence to archive   762
Current log sequence           762
```

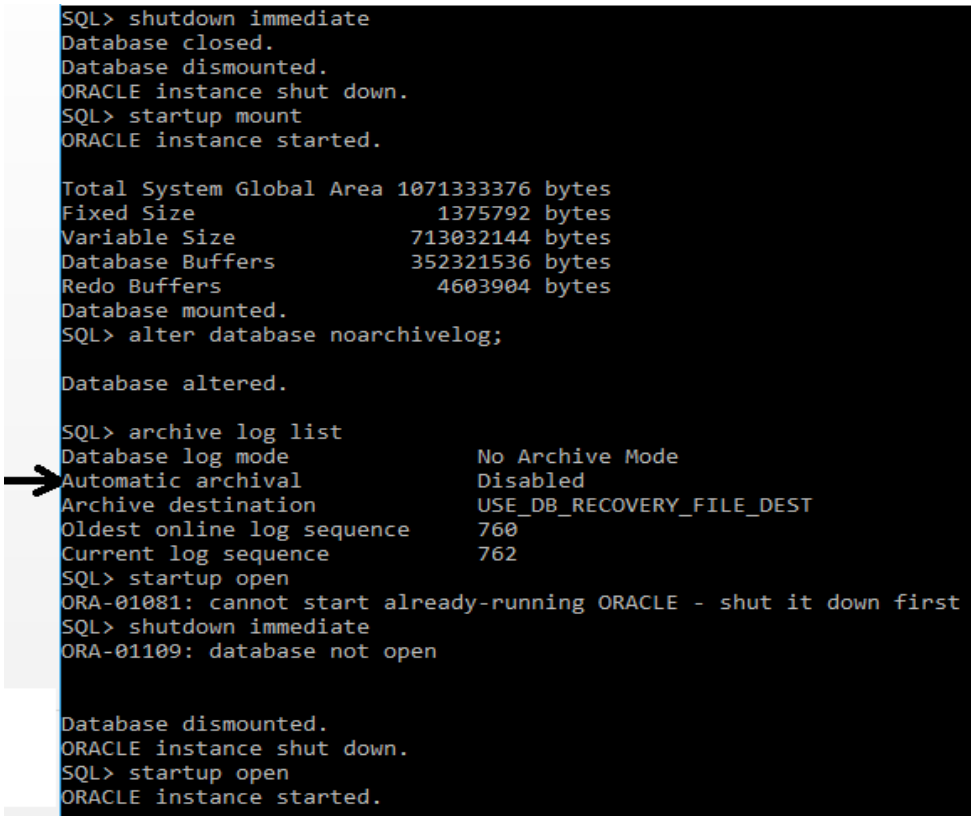
```
SQL> shutdown immediate
ORA-01109: database not open


Database dismounted.
ORACLE instance shut down.
SQL> startup open
ORACLE instance started.

Total System Global Area 1071333376 bytes
Fixed Size                  1375792 bytes
Variable Size             713032144 bytes
Database Buffers          352321536 bytes
Redo Buffers                4603904 bytes
Database mounted.
Database opened.
```

Using SQL sys to convert Archivelog Disabled to Enabled & Vice Versa

During recovery Oracle ask you to use

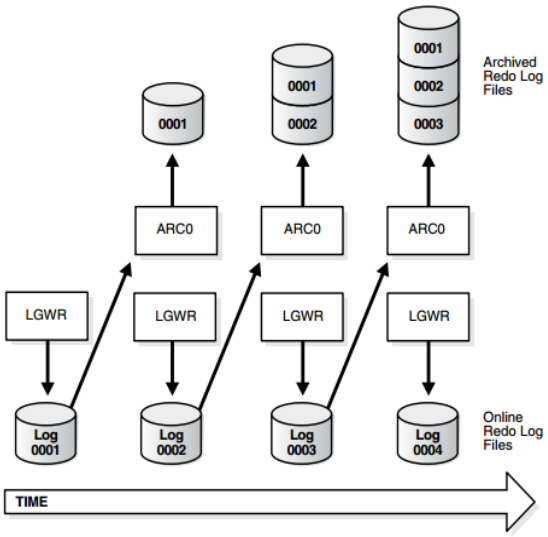
Alter database open instead of startup open

Archived Redo Log Enabled to Disabled



```
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup mount
ORACLE instance started.

Total System Global Area 1071333376 bytes
Fixed Size                  1375792 bytes
Variable Size             713032144 bytes
Database Buffers          352321536 bytes
Redo Buffers                4603904 bytes
Database mounted.
SQL> alter database noarchivelog;

Database altered.

SQL> archive log list
Database log mode              No Archive Mode
Automatic archival             Disabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     760
Current log sequence           762
SQL> startup open
ORA-01081: cannot start already-running ORACLE - shut it down first
SQL> shutdown immediate
ORA-01109: database not open


Database dismounted.
ORACLE instance shut down.
SQL> startup open
ORACLE instance started.
```

Redo Logs Use in ARCHIVELOG Mode



Default mode is Disabled Archive log

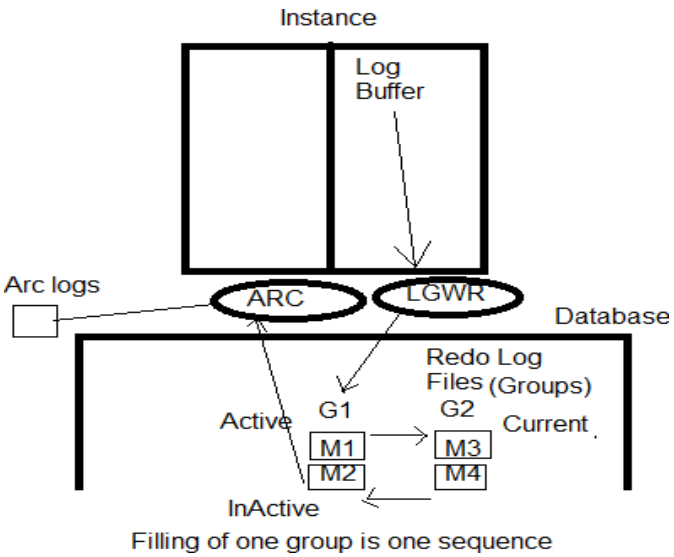
Lesson 68 : Controlling Archiving

Controlling Archive of Redo Log

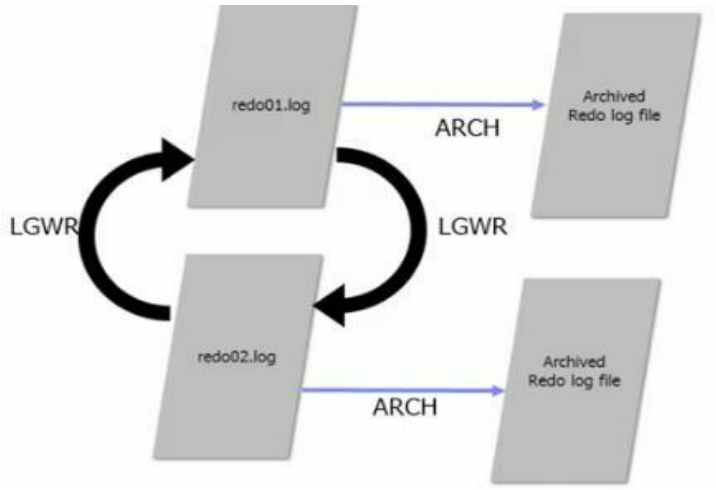
- What is Oracle Archiving?
- Setting the Initial Database Archiving Mode
- Changing the Database Archiving Mode
- Performing Manual Archiving
- Adjusting the Number of Archiver Processes

What is Oracle Archiving?

Archiving: After a switch, a copy of the Redo Log file is sent to Archive Destination



Archiving: After a switch, a copy of the Redo Log file is sent to Archive Destination



Setting the Initial Database Archiving Mode

- Usually, you can use the default of NOARCHIVELOG mode at database creation because there is no need to archive the redo information generated by that process.
- After creating the database, decide whether to change the initial archiving mode.
- If you specify ARCHIVELOG mode, you must have initialization parameters set that specify the destinations for the archived redo log files

Changing Archiving Mode

- Start a new instance and mount, but do not open, the database.

STARTUP MOUNT

- To enable or disable archiving, the database must be mounted but not open.
- Change the database archiving mode. Then open the database for normal operations.

ALTER DATABASE ARCHIVELOG;

ALTER DATABASE OPEN;

- Shut down the database.

SHUTDOWN IMMEDIATE

- Back up the database.

Changing Archiving Mode Manual

- Start a new instance and mount, but do not open, the database.

STARTUP MOUNT

- Change the database archiving mode. Then open the database for normal operations.

ALTER DATABASE ARCHIVELOG MANUAL;

ALTER DATABASE OPEN;

ALTER SYSTEM ARCHIVELOG ALL;

- Shut down the database.

SHUTDOWN IMMEDIATE

- Back up the database.

Adjusting the Number of Archiver Processes

- The LOG_ARCHIVE_MAX_PROCESSES initialization parameter specifies the number of ARCn processes that the database initially starts. The default is four processes.
- To avoid any run-time overhead of starting additional ARCn processes, you can set the LOG_ARCHIVE_MAX_PROCESSES
- Initialization parameter to specify that up to 30 ARCn ALTER SYSTEM SET LOG_ARCHIVE_MAX_PROCESSES=6;

Lesson 69: Specifying Archive Destinations

- Specifying Archive Destinations
- Archive Destinations Status
- You can choose to archive redo logs to a single destination or to multiple destinations.
- Destinations can be local—within the local file system or an Oracle Automatic Storage Management (Oracle ASM) disk group—or remote.
- When you archive to multiple destinations, a copy of each filled redo log file is written to each destination. These redundant copies help ensure that archived logs are always available in the event of a failure at one of the destinations.
- To archive to only a single destination, specify that destination using the LOG_ARCHIVE_DEST initialization parameter.
- To archive to multiple destinations, you can choose to archive to two or more locations using the
- LOG_ARCHIVE_DEST_n initialization parameters, or to archive only to a primary and secondary destination using the LOG_ARCHIVE_DEST and LOG_ARCHIVE_DUPLEX_DEST initialization parameters.

Method	Initialization Parameter	Host	Example
1	LOG_ARCHIVE_DEST_*n* where: *n* is an integer from 1 to 31. Archive destinations 1 to 10 are available for local or remote locations. Archive destinations 11 to 31 are available for remote locations only.	Local or remote	LOG_ARCHIVE_DEST_1 = 'LOCATION=/disk1/arc' LOG_ARCHIVE_DEST_2 = 'LOCATION=/disk2/arc' LOG_ARCHIVE_DEST_3 = 'SERVICE=standby1'
2	LOG_ARCHIVE_DEST and LOG_ARCHIVE_DUPLEX_DEST	Local only	LOG_ARCHIVE_DEST = '/disk1/arc' LOG_ARCHIVE_DUPLEX_DEST = '/disk2/arc'

Archive Destinations Status

- Valid/Invalid: indicates whether the disk location or service name information is specified and valid
- Enabled/Disabled: indicates the availability state of the location and whether the database can use the destination
- Active/Inactive: indicates whether there was a problem accessing the destination

Lesson 70: Managing Archive Destination Failure & Controlling Trace

- Rearchiving to a Failed Destination
- Controlling Trace Output
- Initialization Parameter
- Archived Redo Log Views

Rearchiving to a Failed Destination

- Use the REOPEN attribute of the LOG_ARCHIVE_DEST_n parameter to specify whether and when ARCn should attempt to rearchive to a failed destination following an error. REOPEN applies to all errors, not just OPEN errors.
- REOPEN=n sets the minimum number of seconds before ARCn should try to reopen a failed destination. The default value for n is 300 seconds. A value of 0 is the same as turning off the REOPEN attribute; ARCn will not attempt to archive after a failure.
- If you do not specify the REOPEN keyword, ARCn will never reopen a destination following an error.
- If you do not specify the REOPEN keyword, ARCn will never reopen a destination following an error.
- You cannot use REOPEN to specify the number of attempts ARCn should make to reconnect and transfer archived logs. The REOPEN attempt either succeeds or fails.
- You cannot use REOPEN to specify the number of attempts ARCn should make to reconnect and transfer archived logs. The REOPEN attempt either succeeds or fails.

When you specify REOPEN for an OPTIONAL destination, the database can overwrite online logs if there is an error. If you specify REOPEN for a MANDATORY destination, the database stalls the production database when it cannot successfully archive.

In this situation, consider the following options:

- Archive manually to the failed destination.
- Change the destination by deferring the destination, specifying the destination as optional, or changing the service.
- Drop the destination.

Controlling Trace Output

Background processes always write to a trace file when appropriate. In the case of the archive log process, you can control the output that is generated to the trace file. You do this by setting the LOG_ARCHIVE_TRACE initialization parameter to specify a trace level.

The following values can be specified on next slide:

Initialization Parameter

Trace Level	Meaning
0	Disable archivelog tracing. This is the default.
1	Track archival of redo log file.
2	Track archival status for each archivelog destination.
4	Track archival operational phase.
8	Track archivelog destination activity.
16	Track detailed archivelog destination activity.
32	Track archivelog destination parameter modifications.
64	Track ARC*n* process state activity.
128	Track FAL (fetch archived log) server related activities.
256	Supported in a future release.
512	Tracks asynchronous LGWR activity.
1024	RFS physical client tracking.
2048	ARC*n*/RFS heartbeat tracking.
4096	Track real-time apply
8192	Track redo apply activity (media recovery or physical standby)

You can combine tracing levels by specifying a value equal to the sum of the individual levels that you would like to trace. For example, setting LOG_ARCHIVE_TRACE=12, will generate trace level 8 and 4 output.

Archived Redo Log Views

Dynamic Performance View	Description
V$DATABASE	Shows if the database is in ARCHIVELOG or NOARCHIVELOG mode and if MANUAL (archiving mode) has been specified.
V$ARCHIVED_LOG	Displays historical archived log information from the control file. If you use a recovery catalog, the RC_ARCHIVED_LOG view contains similar information.
V$ARCHIVE_DEST	Describes the current instance, all archive destinations, and the current value, mode, and status of these destinations.
V$ARCHIVE_PROCESSES	Displays information about the state of the various archive processes for an instance.
V$BACKUP_REDOLOG	Contains information about any backups of archived logs. If you use a recovery catalog, the RC_BACKUP_REDOLOG contains similar information.
V$LOG	Displays all redo log groups for the database and indicates which need to be archived.
V$LOG_HISTORY	Contains log history information such as which logs have been archived and the SCN range for each archived log.

```
SELECT GROUP#, ARCHIVED
   FROM SYS.V$LOG;

GROUP#      ARC
--------    ---
         1  YES
         2  NO
```

Lesson 71: Database Physical Design

Physical Design

- Definition & Concepts
- Criteria for Physical Design
- Analyzing DB Queries & Transactions
- Time Constraints of Queries & Transactions
- Physical DB Design Decisions
- Physical Database Design for Tuning
- Analyzing Database Queries & Transactions

Definition & Concepts

- Physical database design is the process of choosing specific storage structures and access paths for the database files to achieve good performance for the various database applications.
- Each DBMS offers a variety of options for file organization and access path. Each DBMS includes various types of indexing, clustering of related records on disk blocks, link related records via pointers and various types of hashing/ partitions.
- For example, Oracle gives the concept of partitions as TableSpaces & MS SQL Server provides concept of sub databases

Criteria for Physical Design

- Response Time: Elapse time between submitting a database transaction for execution and receiving a response. Response time is also influenced by factors not under control of DBMS, factors are communication, scheduling and operating system etc.
- Space Utilization: An amount of storage space used by the database files and their access path structure on disk including indexes.

- Transaction Throughput: Average number of transactions processed per minute. It is measured at peak conditions on the system

Analyzing DB Queries & Transactions

- The files that will be accessed by the query.
- Whether the selection condition is an equality, inequality, or a range condition.
- The attributes whose values will be retrieved by the query.
- The attributes on which any selection conditions for the queries are specified.

Files means tables as well. Which tables have data from which files at OS level

- The attributes on which any join conditions or conditions to link multiple tables or objects for the query are specified.
- The files that will be modified.
- The attributes on which selection conditions for a delete or update is specified.

Time Constraints of Queries & Transactions

- Some queries and transactions may stringent performance constraints.
- For example, a transaction may have the constraint that it should terminate within 5 seconds on 95 percent of the occasions when it is invoked than it would never take more than 20 seconds.
- The type of operations on each file (insert, delete and update)

Physical DB Design Decisions

- Design decisions about indexing: Attributes that are joint on equality, inequality or range conditions.
- Whether to index an attribute.
- What attribute(s) to index on. (Composite keys or one of attributes in key and other is non-key)

Clustered index: When table has several indexes on its attributes. Need to give priority to an index in which attribute(s).hash index: B+ tree is mostly used in DBMS for both equality and range queries. Hash works with equality conditions, particularly during joins to find a matching record. dynamic hashing : When files are very volatile, grow and shrink continuously.

Physical DB Design Decisions

- Whether to set up a Clustered index.
- Whether to use hash index over tree index.
- Whether to use dynamic hashing for the file (files are very volatiles).
- Apply de-normalization: separate the logically related attributes into tables to minimize redundancy.

Physical Database Design for Tuning

- Size of individual tables
- Number of distinct values in a column
- Time of query submission and its execution
- Storage statistics: Partitions, index space and buffer pool
- I/O & performance statistics: Total read/write activity
- Locking/ logging related statistics: Rates of issuing different types of locks, transaction throughput rates, and log records activity
- Index statistics: Number of levels in an index, number of non-contiguous leaf pages.

Analyzing Database Queries & Transactions

- The files that will be accessed by the query
- The attributes on which any selection condition (WHERE) for the query are specified
- The attributes on which any join conditions are specified

Data files of logical tablespace area, table is located on tablespace

Access path: How data is exactly stored in back end files, how and where files are stored. Data is managed logically in the form of segments and extents for tablespaces. Also, how data is indexed and stored on separate indexed tablespaces. OS disk partitions are also be a part of Access path.

- The attributes whose values will retrieved by the queries
- The files that will be updated
- Type of operations on files (INSERT, DELETE, UPDATE)
- The attributes whose values will be changed by an update operation

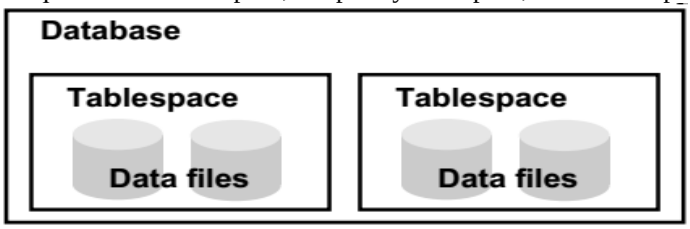
Lesson 72: Defining Tablespace, Types & Data Files

Tablespace & Data Files

- Definition & Concepts
- Benefits of using Multiple Tablespaces
- Assigning Tablespace Quotas to Users
- Types of Tablespaces

Definition & Concepts

- An Oracle database consists of one or more logical storage units called tablespaces, which collectively store all of the database's data.
- For example, Enterprise's data tablespace, temporary tablespace, index tablespace etc.



Definition & Concepts

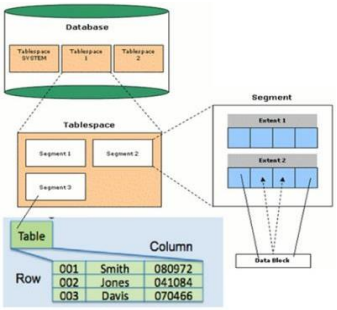
Tablespaces (Logical Level)

- Can belong to only one database at a time
- Consists of one or more data files
- Are further divided into logical units of storage

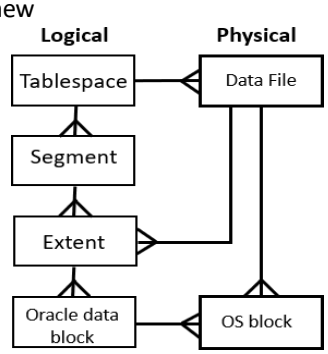
Data Files (Physical Level)

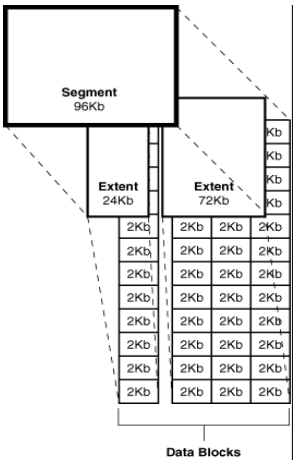
- Can belong to only one tablespace and one database
- Are a repository for schema object data

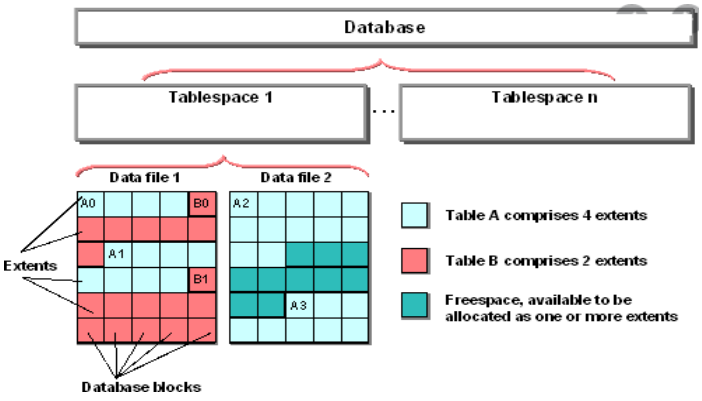
Tablespaces & Extents



An extent is a logical unit of database storage space allocation made up of a number of contiguous data blocks. One or more extents in turn make up a segment. When the existing space in a segment is completely used, Oracle allocates a new extent for the segment.







Benefits of using Multiple Tablespaces

- Separate user data from data dictionary data to reduce I/O contention.
- Separate data of one application from the data of another to prevent multiple applications for offline purpose.
- Store the data files of different tablespaces on different disk drives to reduce I/O contention.
- Take individual tablespaces offline while others remain online, providing better overall availability.
- In one database one can put off line data of Sales tablespaces when other activities of Purchasing tablespaces are going on
- Optimizing tablespace use by reserving a tablespace for a particular type of database use, such as high update activity, read-only activity, or temporary segment storage.
- Back up individual tablespaces.

Note: Some operating system can affect the number of tablespaces that can simultaneously online.

Data files size for tablespaces must be reasonable.

Review your data in light of these factors and decide how many tablespaces you need for your database design. In one database one can put off line data of Sales tablespaces when other activities of purchasing tablespaces are going on

Assigning Tablespace Quotas to Users

Grant to users who will be creating tables, clusters, materialized views, indexes, and other objects the privilege to create the object and a quota(space allowance or limit) in the tablespace intended to hold the object segment. For PL/SQL objects such as packages, procedures, and functions, users only need the privileges to create the objects. No explicit tablespace quota is required to create these PL/SQL objects

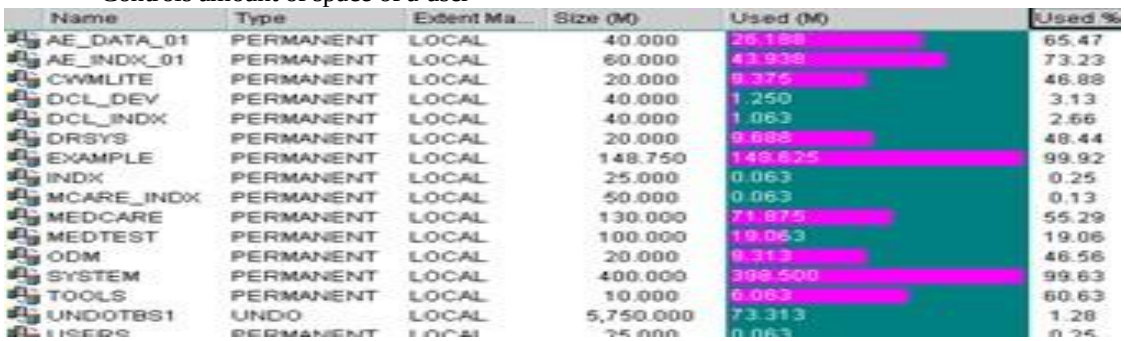
Types of Tablespaces

SYSTEM tablespace

- Created with the database
- Contains the data dictionary
- Contains the SYSTEM undo segment

Non-SYSTEM tablespace

- Separate segments
- Eases space administration
- Controls amount of space of a user



Name	Type	Extent Ma...	Size (M)	Used (M)	Used %
AE_DATA_01	PERMANENT	LOCAL	40.000	26.188	65.47
AE_INDX_01	PERMANENT	LOCAL	60.000	43.938	73.23
CWMLITE	PERMANENT	LOCAL	20.000	9.375	46.88
DCL_DEV	PERMANENT	LOCAL	40.000	1.250	3.13
DCL_INDX	PERMANENT	LOCAL	40.000	1.063	2.66
DRSYS	PERMANENT	LOCAL	20.000	9.688	48.44
EXAMPLE	PERMANENT	LOCAL	148.750	148.625	99.92
INDX	PERMANENT	LOCAL	25.000	0.063	0.25
MCARE_INDX	PERMANENT	LOCAL	50.000	0.063	0.13
MEDCARE	PERMANENT	LOCAL	130.000	71.875	55.29
MEDTEST	PERMANENT	LOCAL	100.000	19.063	19.06
ODM	PERMANENT	LOCAL	20.000	9.313	46.56
SYSTEM	PERMANENT	LOCAL	400.000	398.500	99.63
TOOLS	PERMANENT	LOCAL	10.000	6.063	60.63
UNDOTBS1	UNDO	LOCAL	5,750.000	73.313	1.28
USERS	PERMANENT	LOCAL	25.000	0.063	0.25

System is a default built in tablespace. Medcare, MedTest are enterprise's data tablespaces.

Lesson 73: Creating Tablespaces

- Creating Tablespace Syntax
- Creating Tablespace using SQL
- How a user uses Tablespace?
- Space Management in Tablespace
- Locally-Managed Tablespaces
- Dictionary-Managed Tablespaces
- Create UNDO Tablespace
- Create Temporary Tablespace
- Creating a default Temporary Tablespace
- Restrictions

Creating Tablespace Syntax

```
CREATE TABLESPACE tablespace
    [DATAFILE clause]
    [MINIMUM EXTENT integer[K|M]]
    [BLOCKSIZE integer [K]]
    [LOGGING|NOLOGGING]
    [DEFAULT storage_clause ]
    [ONLINE|OFFLINE]
    [PERMANENT|TEMPORARY]
    [extent_management_clause]
    [segment_management_clause]
```

Readonly, online, offline can also be used.

Creating Tablespace using SQL

Windows Platform:

CREATE TABLESPACE WORK

datafile 'C:\Oracle11g\ordata\wrk01.dbf'

size 500M online;

Unix/ Linux Platform:

CREATE TABLESPACE WORK

datafile '/u01/ordata/wrk01.dbf'

size 500M online;

How a user uses Tablespace?

Create user ejaz identified by pwd default tablespace WORK temporary tablespace TEMP;

--Assigning default roles to user(owner) Grant connect, resource to ejaz;

Grant create table to ejaz;

Grant create view to ejaz;

Space Management in Tablespace

Tablespaces allocate space in extents. Tablespaces can be created to use one of the following two different methods of keeping track of free and used space:

Locally managed tablespaces:

The extents are managed within the tablespace via bitmaps. Each bit in the bitmap corresponds to a block or a group of blocks.

Space Management in Tablespace

When an extent is allocated or freed for reuse, the Oracle server changes the bitmap values to show the new status of the blocks. Locally managed is the default beginning Oracle.

Dictionary-managed tablespaces:

The extents are managed by the data dictionary. The Oracle server updates the appropriate tables in the data dictionary whenever an extent is allocated or deallocated.

Locally-Managed Tablespaces

- Reduced contention on data dictionary tables
- No undo generated when space allocation or deallocation occurs
- No coalescing required CREATE TABLESPACE userdata datafile 'C:\Ora11g\ordata\usr01.dbf' size 500M EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K;

Dictionary-Managed Tablespaces

- Extents are managed in the data dictionary
- Each segment stored in the tablespace can have a different storage clause.
- No coalescing required CREATE TABLESPACE userdata datafile 'C:\Ora11g\ordata\usr01.dbf' size 500M EXTENT MANAGEMENT DICTIONARY DEFAULT STORAGE (Initial 1M NEXT 1M PCTINCREASE 0);

Create UNDO Tablespace

- Used to store undo segments
- Cannot contain any other objects
- Extents are locally managed
- Can only use the DATAFILE and EXTENT MANAGEMENT Clause CREATE UNDO TABLESPACE undo01 datafile 'C:\Oracle11g\ordata\undo01.dbf' size 40M;

Create Temporary Tablespace

- Used for sort operations
- Can be shared by multiple users
- Cannot contain any permanent objects
- Locally managed extents recommended

CREATE Temporary TABLESPACE temp1

datafile 'C:\Oracle11g\ordata\temp101.dbf'

size 20M

EXTENT MANAGEMENT LOCAL UNIFORM SIZE 4M;

When big data sort is not accommodated in a memory, use temp tablespace

Creating a default Temporary Tablespace

- After database creation:

ALTER DATABASE

DEFAULT TEMPORARY TABLESPACE default_temp1;

- To see default temporary tablespace, use

Select property_name from database_properties;

When big data sort is not accommodated in a memory, use temp tablespace

Restrictions

Default temporary tablespace cannot be:

- Dropped until after a new default is made available
- Taken offline
- Alter to a permanent tablespace

Lesson 74: Specifying Non Standard Block Sizes for Tablespaces

Block Sizes for Tablespaces

- Bigfile Tablespaces
- Creating Bigfile Tablespaces
- VIEWS for Bigfile Tablespaces
- Non Standard Block Sizes for Tablespaces
- Creating Non Standard Tablespaces

Bigfile Tablespaces : A bigfile tablespace is a tablespace with a single, but potentially very large (up to 4G blocks) data file. Traditional small file tablespaces, in contrast, can contain multiple data files, but the files cannot be as large.

The benefits of bigfile tablespaces are as follows:

- A bigfile tablespace with 8K blocks can contain a 32 terabyte data file. A bigfile tablespace with 32K blocks can contain a 128 terabyte data file. The maximum number of data files in an Oracle Database is limited (usually to 64K files). Therefore, bigfile tablespaces can significantly enhance the storage capacity of an Oracle DB
- Bigfile tablespaces can reduce the number of data files needed for a database. An additional benefit is that the DB_FILES initialization parameter and MAXDATAFILES parameter of the CREATE DATABASE and CREATE CONTROLFILE statements can be adjusted to reduce the amount of SGA space required for data file information and the size of the control file.
- Bigfile tablespaces simplify database management by providing data file transparency. SQL syntax for the ALTER TABLESPACE statement lets you perform operations on tablespaces, rather than the underlying individual data files.

Creating Bigfile Tablespaces

CREATE BIGFILE TABLESPACE bigtbs DATAFILE '/u02/oracle/data/bigtbs01.dbf' SIZE 50G...

- You can specify SIZE in kilobytes (K), megabytes (M),gigabytes (G), or terabytes (T).
- If the default tablespace type was set to BIGFILE at database creation then bigfile tablespace will be created by default.

VIEWS for Bigfile Tablespaces

- DBA_TABLESPACES
- USER_TABLESPACES
- V$TABLESPACE

Non Standard Block Sizes for Tablespaces

- You can create tablespaces with block sizes different from the standard database block size, which is specified by the DB_BLOCK_SIZE initialization parameter. This feature lets you transport tablespaces with unlike block sizes between databases.
- Use the BLOCKSIZE clause of the CREATE TABLESPACE statement to create a tablespace with a block size different from the database standard block size. In order for the BLOCKSIZE clause to succeed, you must have already set the DB_CACHE_SIZE and at least one DB_nK_CACHE_SIZE initialization parameter.

- Further, and the integer you specify in the BLOCKSIZE clause must correspond with the setting of one DB_nK_CACHE_SIZE parameter setting. Although redundant, specifying a BLOCK SIZE equal to the standard block size, as specified by the DB_BLOCK_SIZE initialization parameter, is allowed.

Creating Non Standard Tablespaces

- CREATE TABLESPACE **lmtbsb**
- DATAFILE '/u02/oracle/data/lmtbsb01.dbf'
- SIZE 50M
- EXTENT MANAGEMENT LOCAL
- UNIFORM SIZE 128K
- BLOCKSIZE 8K;

Lesson 75: Altering Tablespaces

- Altering Tablespaces Availability
- Taking Tablespaces Offline
- Tablespaces cannot Offline
- Altering Tablespace SQL
- Owner of Tablespace
- Making Tablespace Readonly
- Dictionary View for use of Altering Tablespace
- Wait for Altering Tablespace to Readonly

Altering Tablespaces Availability

Before you can make a tablespace read-only, the following conditions must be met.

- You can take an online tablespace offline so that it is temporarily unavailable for general use.
- The rest of the database remains open and available for users to access data.
- Conversely, you can bring an offline tablespace online to make the schema objects within the tablespace available to database users.
- The database must be open to alter the availability of a tablespace.

Taking Tablespaces Offline

- To make a portion of the database unavailable while allowing normal access to the remainder of the database
- To perform an offline tablespace backup (even though a tablespace can be backed up while online and in use)
- To make an application and its group of tables temporarily unavailable while updating or maintaining the application
- To rename or relocate tablespace data files

Tablespaces cannot Offline

- SYSTEM
- The undo tablespace
- Temporary tablespaces

Altering Tablespace SQL

- Here are sample examples

```
ALTER TABLESPACE users OFFLINE;

ALTER TABLESPACE users ONLINE;

ALTER TABLESPACE work

add datafile 'C:\oracle11g\oradata\wrk02.dbf' size 500M;
```

Owner of Tablespace

Following SQL statement assigns tablespace to a particular user:

CREATE USER meduser

Identified by pwd default tablespace WORK temporary tablespace TEMP;

Making Tablespace Read only

- The tablespace must be online. This is necessary to ensure that there is no undo information that must be applied to the tablespace.
- The tablespace cannot be the active undo tablespace or SYSTEM tablespace.
- The tablespace must not currently be involved in an online backup, because the end of a backup updates the header file of all data files in the tablespace.
- The tablespace cannot be a temporary tablespace.

Dictionary View for use of Altering Tablespace

```
SELECT SQL_TEXT, SADDR
    FROM V$SQLAREA,V$SESSION
    WHERE V$SQLAREA.ADDRESS = V$SESSION.SQL_ADDRESS
        AND SQL_TEXT LIKE 'alter tablespace%';

SQL_TEXT                                   SADDR
------------------------------------------ --------
alter tablespace tbs1 read only            80034AF0
```

```
SELECT SES_ADDR, START_SCNB
    FROM V$TRANSACTION
    ORDER BY START_SCNB;

SES_ADDR START_SCNB
-------- ----------
800352A0       3621   --> waiting on this txn
80035A50       3623   --> waiting on this txn
80034AF0       3628   --> this is the ALTER TABLESPACE statement
80037910       3629   --> don't care about this txn
```

Lesson 76: Readonly & Online Status of Tablespaces

- Steps to Create Readonly Tablespaces
- SQL Alter Tablespace Command
- Delaying the opening of Data Files
- Side Effect of Delaying the opening of Data Files
- Important Issues of Delaying the opening of Data Files

Steps to Create Readonly Tablespaces

- Create a writable tablespace on another device. Create the objects that belong in the tablespace and insert your data.
- Alter the tablespace to make it read-only.
- Copy the data files of the tablespace onto the WORM device. Use operating system commands to copy the files.
- Take the tablespace offline.
- Rename the data files to coincide with the names of the data files you copied onto your WORM device. Use ALTER TABLESPACE with the RENAME DATAFILE clause.
- Renaming the data files changes their names in the control file.
- Bring the tablespace back online.

SQL Alter Tablespace Command

ALTER TABLESPACE userdata READ ONLY;

- Causes a checkpoint
- Data available only for read operations
- Objects can be dropped from tablespace

RMAN>

SQL 'ALTER TABLESPACE hr READ ONLY';

CONVERT TABLESPACE hr

TO PLATFORM 'Solaris[tm] OE (64-bit)'

FORMAT '/tmp/transport_to_solaris/%U';;

We will discuss RMAN later.

Delaying the opening of Data Files

- When substantial portions of a very large database are stored in read-only tablespaces that are located on slow-access devices or hierarchical storage, you should consider setting the READ_ONLY_OPEN_DELAYED initialization parameter to TRUE.
- This speeds certain operations, primarily opening the database, by causing data files in read-only tablespaces to be accessed for the first time only when an attempt is made to read data stored within them.

Side Effect of Delaying the opening of Data Files

Setting READ_ONLY_OPEN_DELAYED=TRUE has the following side-effects

- A missing or bad read-only file is not detected at open time. It is only discovered when there is an attempt to access it.
- ALTER SYSTEM CHECK DATAFILESdoes not check read-only files.
- ALTER TABLESPACE...ONLINE and ALTER DATABASE DATAFILE...ONLINE do not check read-only files. They are checked only upon the first access.
- V$RECOVER_FILE, V$BACKUP, and V$DATAFILE_HEADER do not access read-only files.
- Read-only files are indicated in the results list with the error "DELAYED OPEN", with zeroes for the values of other columns.
- V$DATAFILE does not access read-only files. Read-only files have a size of "0" listed.
- V$RECOVERY_LOG does not access read-only files. Logs they could need for recovery are not added to the list.
- ALTER DATABASE NOARCHIVELOG does not access read-only files. It proceeds even if there is a read-only file that requires recovery.

Important Issues of Delaying the opening of Data Files

- RECOVER DATABASE and ALTER DATABASE OPEN RESETLOGS continue to access all read-only data files regardless of the parameter value. To avoid accessing read-only files for these operations, take those files offline.
- If a backup control file is used, the read-only status of some files may be inaccurate. This can cause some of these operations to return unexpected results. Care should be taken in this situation.

Lesson 77: Altering & Maintaining Tablespaces

Maintaining Tablespaces

- Altering a Locally Managed Tablespace
- Add a bigfile Tablespace
- Altering a Locally Managed Temporary Tablespace
- Shrinking a Locally Managed Temporary Tablespace

Altering a Locally Managed Tablespace

- You cannot alter a locally managed tablespace to a locally managed temporary tablespace, nor can you change its method of segment space management. Coalescing free extents is unnecessary for locally managed tablespaces. However, you can use the ALTER TABLESPACE statement on locally managed tablespaces for some operations, including the following:
- Adding a data file. For example:

ALTER TABLESPACE lmtbsb

ADD DATAFILE '/u02/oracle/data/lmtbsb02.dbf' SIZE 1M;

Altering tablespace availability (ONLINE/OFFLINE).

- Making a tablespace read-only or read/write
- Renaming a data file, or enabling or disabling the autoextension of the size of a data file in the tablespace.

Add a bigfile Tablespace

Two clauses of the ALTER TABLESPACE statement support data file transparency when you are using bigfile tablespaces:

RESIZE: The RESIZE clause lets you resize the single data file in a bigfile tablespace to an absolute size, without referring to the data file. For example:

ALTER TABLESPACE bigtbs RESIZE 80G;

AUTOEXTEND(used outside of the ADD DATAFILE clause):

With a bigfile tablespace, you can use the AUTOEXTEND clause outside of the ADD DATAFILEclause.

For example:

ALTER TABLESPACE bigtbs AUTOEXTEND ON NEXT 20G;

An error is raised if you specify an ADD DATAFILE clause for a bigfile tablespace.

Altering a Locally Managed Temporary Tablespace

You can use ALTER TABLESPACE to add a temp file, take a temp file offline, or bring a temp file online, as illustrated in the following examples:

ALTER TABLESPACE lmtemp

ADD TEMPFILE '/u02/oracle/data/lmtemp02.dbf' SIZE 18M REUSE;

ALTER TABLESPACE lmtemp TEMPFILE OFFLINE;

ALTER TABLESPACE lmtemp TEMPFILE ONLINE;

The ALTER DATABASE statement can be used to alter temp files.

The following statements take offline and bring online temp files. They behave identically to the last two ALTER TABLESPACE statements in the previous example.

ALTER DATABASE TEMPFILE '/u02/oracle/data/lmtemp02.dbf' OFFLINE;

ALTER DATABASE TEMPFILE '/u02/oracle/data/lmtemp02.dbf' ONLINE;

The following statement resizes a temp file:

ALTER DATABASE TEMPFILE '/u02/oracle/data/lmtemp02.dbf' RESIZE 18M;

The following statement drops a temp file and deletes its operating system file:

ALTER DATABASE TEMPFILE '/u02/oracle/data/lmtemp02.dbf'

DROP INCLUDING DATAFILES;

The tablespace to which this temp file belonged remains. A message is written to the alert log for the temp file that was deleted.

Shrinking a Locally Managed Temporary Tablespace

Large sort operations performed by the database may result in a temporary tablespace growing and occupying a considerable amount of disk space. After the sort operation completes, the extra space is not released; it is just marked as free and available for reuse.

Therefore, a single large sort operation might result in a large amount of allocated temporary space that remains unused after the sort operation is complete.

For this reason, the database enables you to shrink locally managed temporary tablespaces and release unused space.

ALTER TABLESPACE lmtemp1 SHRINK SPACE KEEP 20M;

The following example shrinks the temp file lmtemp02.dbf of the locally managed temporary tablespace lmtmp2.

ALTER TABLESPACE lmtemp2 SHRINK TEMPFILE '/u02/oracle/data/lmtemp02.dbf'

Lesson 78: Renaming & Dropping Tablespaces

- Renaming Tablespaces
- Affects of Renaming Tablespaces
- Dropping Tablespaces
- Dropping Tablespaces - Example
- Segments in a Tablespace

Renaming Tablespaces

Using the RENAME TO clause of the ALTER TABLESPACE, you can rename a permanent or temporary tablespace. For example, the following statement renames the users tablespace:

ALTER TABLESPACE users RENAME TO usersts;

When you rename a tablespace the database updates all references to the tablespace name in the data dictionary, control file, and (online) data file headers.

To confirm use DBA_USERS.

Affects of Renaming Tablespaces

- The COMPATIBLE parameter must be set to 10.0.0 or higher.
- If the tablespace being renamed is the SYSTEM tablespace or the SYSAUX tablespace, then it will not be renamed.
- If any data file or the tablespace is offline then the tablespace is not renamed.
- If the tablespace is read only, then data file headers are not updated. This should not be regarded as corruption; instead, it causes a message to be written to the alert log indicating that data file headers have not been renamed. The data dictionary and control file are updated.
- If the tablespace is the default temporary tablespace, then the corresponding entry in the database properties table is updated and the DATABASE_PROPERTIES view shows the new name.
- If the tablespace is an undo tablespace and if the following conditions are met, then the tablespace name is changed to the new tablespace name in the server parameter file (SPFILE).
  The server parameter file was used to start up the database.

  The tablespace name is specified as the UNDO_TABLESPACE for any instance.

Dropping Tablespaces

- You can drop a tablespace and its contents (the segments contained in the tablespace) from the database if the tablespace and its contents are no longer required. You must have the DROP TABLESPACE system privilege to drop a tablespace.
- When you drop a tablespace, the file pointers in the control file of the associated database are removed. You can optionally direct Oracle Database to delete the operating system files (data files) that constituted the dropped tablespace.
- Off line tablespace before dropping it.
- You cannot drop a tablespace that contains any active segments.
- For example, if a table in the tablespace is currently being used or the tablespace contains undo data needed to roll back uncommitted transactions, you cannot drop the tablespace.
- To drop a tablespace, use the DROP TABLESPACE statement. The following statements drops the users tablespace, including the segments in the tablespace:
  Alter tablespace users offline;

  DROP TABLESPACE users

  INCLUDING CONTENTS;

To delete the data files associated with a tablespace at the same time that the tablespace is dropped, use the INCLUDING CONTENTS AND DATAFILES clause. The following statement drops the users tablespace and its associated data files:

DROP TABLESPACE users INCLUDING CONTENTS AND DATAFILES;

Dropping Tablespaces - Example

create tablespace test

datafile 'E:\app\Haider\oradata\tst01.dbf' size 100M;

Tablespace created.

create user usr1 identified by xyz

default tablespace test

temporary tablespace temp;

grant connect, resource to usr1;

grant create table to usr1;

alter tablespace test offline;

shutdown immediate

startup open

DROP TABLESPACE test including contents;

conn usr1/xyz

SQL> create table AA

2 (id number);

create table

*

ERROR at line 1:

ORA-00959: tablespace 'TEST' does not exist

Segments in a Tablespace

```
SQL> set linesize 200;
SQL> select segment_name, segment_type
  2  from dba_segments
  3  where tablespace_name='WRK_NOVUS';

SEGMENT_NAME                       SEGMENT_TYPE
---------------------------------- --------------
PATIENT                            TABLE
PATIENT_VISIT                      TABLE
PK_EMP_ID                          INDEX
PK_PATVISIT_PAT_VSNO               INDEX
IX_PATVISIT_PID_VDATE              INDEX
IX_PATVISIT_PID_DPNSNO             INDEX
IX_PATVISIT_EMPID_VDATE            INDEX
IX_PATIENT_VISIT_CREATE_DATE       INDEX

8 rows selected.
```

Lesson 79: Managing SYSAUX Tablespaces

SYSAUX Tablespaces

- Introduction
- Monitoring Occupants
- Benefits of using SYSAUX Tablespace
- Controlling the Size of the SYSAUX Tablespace - Guidelines

Introduction

- The SYSAUX tablespace was installed as an auxiliary tablespace to the SYSTEM tablespace when you created your database.
- Some database components that formerly created and used separate tablespaces now occupy the SYSAUX tablespace.
- SYSAUX tablespace is used to reduce the load managed by SYSTEM tablespace.
- If the SYSAUX tablespace becomes unavailable, core database functionality will remain operational.
- The database features that use the SYSAUX tablespace could fail, or function with limited capability.
- SYSTEM tablespace will continue running for the users to work on if some problem occurs in SYSAUX tablespace.

- SYSAUX tablespace mainly used for performance and comparison purposes

Monitoring Occupants

You can monitor the occupants of the SYSAUX tablespace using the V$SYSAUX_OCCUPANTS view. This view lists the following information about the occupants of the SYSAUX tablespace:

- Name of the occupant
- Occupant description
- Schema name
- Move procedure
- Current space usage
- Another major occupant of the SYSAUX tablespace is the embedded Oracle Enterprise Manager repository. This repository is used by Oracle Enterprise Manager Database Control to store its metadata. The size of this repository depends on database activity and on configuration-related information stored in the repository.

Benefits of using SYSAUX Tablespace

- The SYSAUX tablespace is always created at database creation.
- The SYSAUX tablespace serves as an auxiliary tablespace to the SYSTEM tablespace. Because it is the default tablespace for many Oracle Database features and products that previously required their own tablespaces, it reduces the number of tablespaces required by the database.
- It also reduces the load on the SYSTEM tablespace.

Controlling the Size of the SYSAUX Tablespace

- The SYSAUX tablespace is occupied by several database components, and its total size is governed by the space consumed by those components.
- The space consumed by the components, in turn, depends on which features or functionality are being used and on the nature of the database workload.

Controlling the Size of the SYSAUX Tablespace – Guidelines

Parameter/Recommendation	Small	Medium	Large
Number of CPUs	2	8	32
Number of concurrently active sessions	10	20	100
Number of user objects: tables and indexes	500	5,000	50,000
Estimated SYSAUX size at steady state with default configuration	500 MB	2 GB	5 GB

Lesson 80: Repairing Tablespaces

- Repairing Locally Managed Tablespace
- Diagnosing using DBMS_SPACE_ADMIN package
- Scenario-1
- Scenario-2
- Scenario-3
- Scenario-4
- Scenario-5

DROP_EMPTY_SEGMENTS	Drops segments from empty tables or table partitions and dependent objects
MATERIALIZE_DEFERRED_SEGMENTS	Materializes segments for tables and table partitions with deferred segment creation and their dependent objects.
SEGMENT_CORRUPT	Marks the segment corrupt or valid so that appropriate error recovery can be done
SEGMENT_DROP_CORRUPT	Drops a segment currently marked corrupt (without reclaiming space)

Repairing Locally Managed Tablespace

Oracle Database includes the DBMS_SPACE_ADMIN package, which is a collection of aids for diagnosing and repairing problems in locally managed tablespaces.

Following are the some of DBMS_SPACE_ADMIN Package Procedures

Diagnosing using DBMS_SPACE_ADMIN package

The following scenarios describe typical situations in which you can use the DBMS_SPACE_ADMIN package to diagnose and resolve problems.

Note: Some of these procedures can result in lost and unrecoverable data if not used properly. You should work with Oracle Support Services if you have doubts about these procedures.

Scenario-1

Fixing Bitmap When Allocated Blocks are Marked Free out these procedures.

The TABLESPACE_VERIFY procedure discovers that a segment has allocated blocks that are marked free in the bitmap, but no overlap between segments is reported.

In this scenario, perform the following tasks:

1. Call the SEGMENT_DUMP procedure to dump the ranges that the administrator allocated to the segment.
2. For each range, call the TABLESPACE_FIX_BITMAPS procedure with the TABLESPACE_EXTENT_MAKE_USED option to mark the space as used.
3. Call TABLESPACE_REBUILD_QUOTAS to rebuild quotas.

Scenario-2

Dropping a corrupted segment:

In this scenario, perform the following tasks:

1. Call the SEGMENT_VERIFY procedure with the SEGMENT_VERIFY_EXTENTS_GLOBAL option. If no overlaps are reported, then proceed with steps 2 through 5.

2. Call the SEGMENT_DUMP procedure to dump the DBA ranges allocated to the segment.

3. For each range, call TABLESPACE_FIX_BITMAPS with the TABLESPACE_EXTENT_MAKE_FREE option to mark the space as free.

4. Call SEGMENT_DROP_CORRUPT to drop the SEG$ entry.

5. Call TABLESPACE_REBUILD_QUOTAS to rebuild quotas

Scenario-3

Fixing Bitmap Where Overlap is Reported:

The TABLESPACE_VERIFY procedure reports some overlapping. Some of the real data must be sacrificed based on previous internal errors.

After choosing the object to be sacrificed, in this case say, table t1, perform the

following tasks:

1. Make a list of all objects that t1 overlaps.

2. Drop table t1. If necessary, follow up by calling the SEGMENT_DROP_CORRUPT procedure.

3. Call the SEGMENT_VERIFY procedure on all objects that t1overlapped. If necessary, call the TABLESPACE_FIX_BITMAPS procedure to mark appropriate bitmap blocks as used.

4. Rerun the TABLESPACE_VERIFY procedure to verify that the problem is resolved

Scenario-4

Correcting Media Corruption of Bitmap Blocks

A set of bitmap blocks has media corruption.

In this scenario, perform the following tasks:

1. Call the TABLESPACE_REBUILD_BITMAPSprocedure, either on all bitmap blocks, or on a single block if only one is corrupt.

2. Call the TABLESPACE_REBUILD_QUOTAS procedure to rebuild quotas.

3. Call the TABLESPACE_VERIFY procedure to verify that the bitmaps are consistent.

Scenario-5

Assume that the database block size is 2K and the existing extent sizes in tablespace tbs_1 are 10, 50, and 10,000 blocks (used, used, and free). The MINIMUM EXTENT value is 20K (10 blocks). Allow the system to choose the bitmap allocation unit. The value of 10 blocks is chosen, because it is the highest common denominator and does not exceed MINIMUM EXTENT.

The statement to convert tbs_1 to a locally managed tablespace is as follows:

EXEC DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL ('tbs_1');

If you choose to specify an allocation unit size, it must be a factor of the unit size calculated by the system.

Lesson 81: Migrating the System Tablespace

- Introduction
- Other tablespaces
- Pre-requisites of migration
- Command for migration
- Importing and exporting data of tablespaces

Introduction

What is SYSTEM tablespace

- It is required when database is created and when database is getting start using.
- Pre bundled code and data dictionary-managed tablespace.
- It contains System Objects, your objects and Users/ Privileges.
- It contains AWR: Automatic Workload Repository

Other Tablespaces

Following tablespaces are required with the SYSTEM tablespace

- UNDO TS
    - User ability to rollback
    - old values of DML changes
    - Read Consistency of Data
    - Temporary TS
    - contents are temp stored in TS
    - order by, distinct, Index --> PGA is active
    - Kind rough work area
    - USERS TS
    - Create users, which users will use which TS information

Pre-requisites of Migration

Before performing the migration the following conditions must be met:

- The database has a default temporary tablespace that is not SYSTEM.
- There are no rollback segments in the dictionary-managed tablespace.
- There is at least one online rollback segment in a locally managed tablespace, or if using automatic undo management, an undo tablespace is online.
- All tablespaces other than the tablespace containing the undo space (that is, the tablespace containing the rollback segment or the undo tablespace) are in read-only mode.
- The SYSAUX tablespace is offline.
- The system is in restricted mode.
- There is a cold backup of the database.

Command for Migration

The following statement performs the migration:

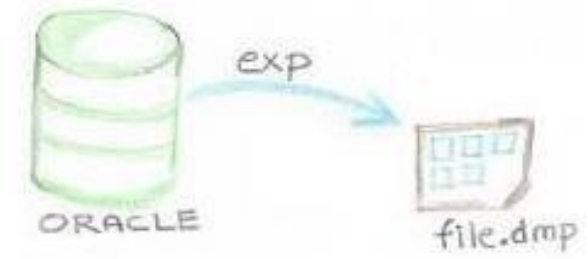
SQL> EXECUTE DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL('SYSTEM');

exp scott/tiger FILE = dat1.dmp, dat2.dmp, dat3.dmp FILESIZE=2048

Importing & Exporting Data of Tablespaces

Use following command of export on source database

C:\>exp usr/pwd file=c:\test\f1.dmp



Importing & Exporting Data of Tablespaces

Following are two tablespaces of source database. We will create sure same tablespaces in the target tablespace.

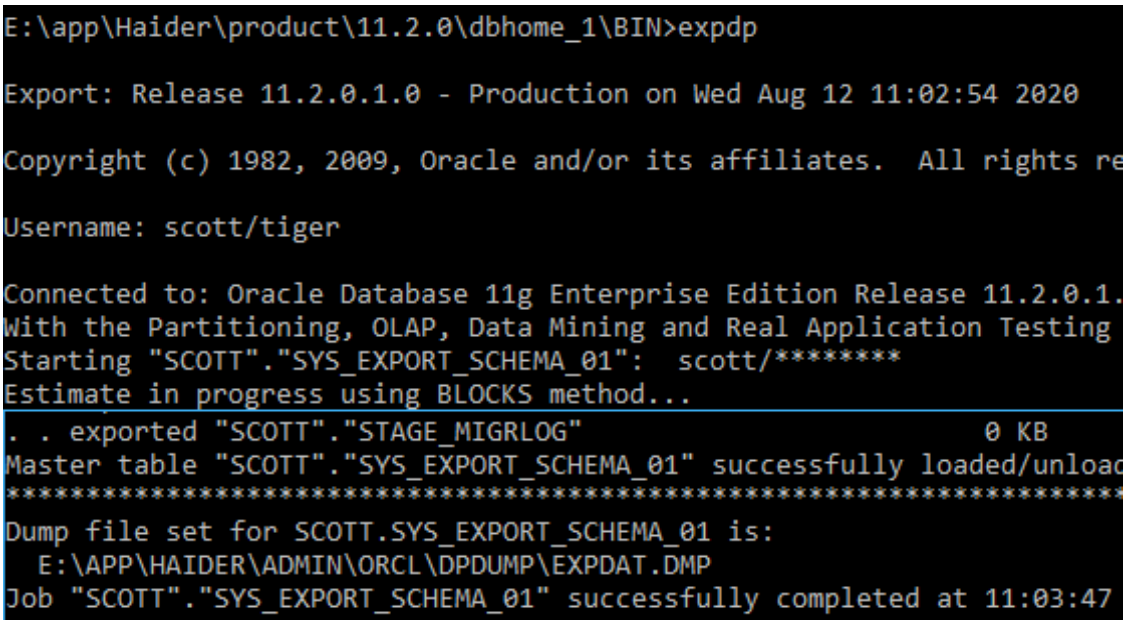
```
Create tablespace MedCare Datafile
    'D:\oracle9i\ora92\oradata\MCare01.dbf' Size 150M online
    MINIMUM EXTENT 64K
    DEFAULT STORAGE (INITIAL 64K NEXT 64K);

CREATE TABLESPACE MCare_indx
DATAFILE 'D:\oracle9i\ora92\oradata\MCare_indx01.dbf' SIZE 100M
MINIMUM EXTENT 64K
DEFAULT STORAGE (INITIAL 64K NEXT 64K);
```

Use following command of import on target database

C:\>exp usr/pwd file=c:\test\f1.dmp

Importing & Exporting Data of Tablespaces

Alternatively, use export/ import data pump command. For example,



Lesson 82: Creating Alternate TEMP Tablespace

- What is Temporary Tablespace?
- Creating TEMP Tablespace
- Verifying TEMP Tablespace
- Create Temporary Tablespace Group
- Temporary Tablespace Files

What is TEMP Tablespace?

- It is required when database is created and when database is getting start using.

Creating TEMP Tablespace

Create a new tablespace and then drop an existing temp tablespace

```
CREATE
    TEMPORARY TABLESPACE "TEMP_DEFAULT2" TEMPFILE
    'D:\ORACLE9I\ORA92\ORADATA\MEDCARE\TEMP_DEFAULT2.DBF' SIZE
    1000M EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M;

ALTER DATABASE DEFAULT TEMPORARY TABLESPACE "TEMP_DEFAULT2";
```

Shutdown and startup database, run complicated query to see whether new tablespace getting filled or not see file alert.log for more details

To check default tablespace is created or DONE or NOT: select * from database_properties;

- Tablespace must be empty before dropping off

- Drop tablespace temp
- including contents and datafiles;
- Shutdown and startup database again
- Drop related files from folder like TEMP01.DBF

Verifying TEMP Tablespace

```
SQL> select property_name from database_properties;

PROPERTY_NAME
------------------------------
DICT.BASE
DEFAULT_TEMP_TABLESPACE
DEFAULT_PERMANENT_TABLESPACE
DEFAULT_EDITION
Flashback Timestamp TimeZone
TDE_MASTER_KEY_ID
DST_UPGRADE_STATE
DST_PRIMARY_TT_VERSION
DST_SECONDARY_TT_VERSION
DEFAULT_TBS_TYPE
NLS_LANGUAGE
```

Create Temporary Tablespace Group

- Temporary Tablespace Groups allow an Oracle database to write to multiple temp tablespaces simultaneously.
- A user is assigned to a group of temp tablespaces instead of a single temporary tablespace
- It allows a single SQL operation to use multiple Temporary Tablespaces for sorting

Create Temporary Tablespace temp1

tempfile 'C:\Oracle\Oradata\temp01.dbf' size 50M

tablespace group tempgroup1;

```
SQL> select *
  2  from database_properties
  3  where property_name like '%TABLESPACE%';

PROPERTY_NAME
------------------------------
PROPERTY_VALUE
------------------------------------------------
------------------------------------------------
DESCRIPTION
------------------------------------------------
------------------------------------------------
DEFAULT_TEMP_TABLESPACE
TEMP
Name of default temporary tablespace

DEFAULT_PERMANENT_TABLESPACE
USERS
Name of default permanent tablespace
```

Temporary Tablespace Files

Using data dictionary view;

```
SQL> select tablespace_name
  2  from dba_temp_files;

TABLESPACE_NAME
------------------------------
TEMP
```

Lesson 83: Viewing Tablespace using Data Dictionary Views

Data Dictionary Views of Tablespace

- Dictionary Views of Tablespaces
- V$Tablespaces
- DBA_DATA_FILES
- DBA_TEMP_FILES
- Finding Used Size of Tablespaces
- Finding Free Space of Tablespaces

Dictionary Views of Tablespaces

Use Dictionary to display views:

```
SQL> select table_name
  2  from dict
  3  where table_name like '%TABLESPACE%';

TABLE_NAME
------------------------------
DBA_FILE_GROUP_TABLESPACES
DBA_HIST_TABLESPACE_STAT
DBA_TABLESPACES
DBA_TABLESPACE_GROUPS
DBA_TABLESPACE_THRESHOLDS
DBA_TABLESPACE_USAGE_METRICS
USER_FILE_GROUP_TABLESPACES
USER_TABLESPACES
ALL_FILE_GROUP_TABLESPACES
V$TABLESPACE
GV$ENCRYPTED_TABLESPACES

TABLE_NAME
------------------------------
V$ENCRYPTED_TABLESPACES
GV$TABLESPACE

13 rows selected.
```

V$Tablespaces

To display columns of V$Tablespaces:

```
SQL> set linesize60
SQL> desc v$tablespace
 Name                          Null?    Type
 ----------------------------- -------- -------------
 TS#                                    NUMBER
 NAME                                   VARCHAR2(30)
 INCLUDED_IN_DATABASE_BACKUP            VARCHAR2(3)
 BIGFILE                                VARCHAR2(3)
 FLASHBACK_ON                           VARCHAR2(3)
 ENCRYPT_IN_BACKUP                      VARCHAR2(3)
```

V$Tablespaces

Select column name V$Tablespaces:

```
SQL> select name
  2  from v$tablespace;

NAME
------------------------------
SYSTEM
SYSAUX
UNDOTBS1
USERS
TEMP
IND_TSMYDEV
TSNDEV
IND_TSNDEV
PTSDEV
PIND_TSDEV
PTSIMG

NAME
------------------------------
PTS_STUCRSREG
PTS_EMARKS
TSIMG
WRK_NOVUS

15 rows selected.
```

DBA_DATA_FILES

Display name and data file size in MB:

```
SQL> select tablespace_name, file_name,
  2  bytes/1024/1024 size_MB
  3  from dba_data_files
  4  where tablespace_name='&tablespace_name';
Enter value for tablespace_name: WRK_NOVUS
old   4: where tablespace_name='&tablespace_name'
new   4: where tablespace_name='WRK_NOVUS'

TABLESPACE_NAME
------------------------------
FILE_NAME
------------------------------------------------------------
   SIZE_MB
----------
WRK_NOVUS
E:\APP\HAIDER\PRODUCT\11.2.0\DBHOME_1\ORADATA\NOVUS01.DBF
       800
```

DBA_TEMP_FILES

Display name and data file size in MB:

```
SQL> select file_name,
  2  bytes/1024/1024 size_MB
  3  from dba_temp_files;

FILE_NAME
------------------------------------------------
   SIZE_MB
----------
E:\APP\HAIDER\ORADATA\ORCL\TEMP01.DBF
       114
```

TEMP01.DBF 13-Aug-20... DBF File 116,744 KB

Finding Used Data Size of Tablespaces

Using data dictionary view DBA_SEGMENTS :

```
SQL> SELECT TABLESPACE_NAME,
  2    ROUND(SUM(BYTES)/(1024*1024), 2)||' MB' AS USED_SIZE
  3    FROM DBA_SEGMENTS
  4    --WHERE TABLESPACE_NAME='EDRDBA'
  5    GROUP BY TABLESPACE_NAME;

TABLESPACE_NAME                USED_SIZE
------------------------------ ------------------------------
PIND_TSDEV                     197.31 MB
SYSAUX                         648.5 MB
TSIMG                          1120.5 MB
UNDOTBS1                       21.69 MB
TSNDEV                         643.56 MB
PTS_EMARKS                     48 MB
WRK_NOVUS                      703.75 MB
USERS                          4.19 MB
SYSTEM                         707.63 MB
PTSDEV                         364.63 MB
PTSIMG                         62 MB

TABLESPACE_NAME                USED_SIZE
------------------------------ ------------------------------
PTS_STUCRSREG                  480 MB
IND_TSNDEV                     .06 MB

13 rows selected.
```

Finding Free Space of Tablespaces

Using data dictionary view DBA_FREE_SPACE

```
SQL>   SELECT TABLESPACE_NAME,
  2    ROUND(SUM(BYTES)/(1024*1024), 2)||' MB' AS FREE_SIZE
  3    FROM DBA_FREE_SPACE
  4    GROUP BY TABLESPACE_NAME;

TABLESPACE_NAME                FREE_SIZE
------------------------------ ------------------------------
IND_TSMYDEV                    999 MB
PIND_TSDEV                     301.75 MB
SYSAUX                         50.5 MB
UNDOTBS1                       282.31 MB
TSIMG                          925.5 MB
TSNDEV                         531.44 MB
PTS_EMARKS                     1950 MB
WRK_NOVUS                      367.25 MB
USERS                          3.56 MB
SYSTEM                         1.38 MB
IND_TSNDEV                     498.94 MB

TABLESPACE_NAME                FREE_SIZE
------------------------------ ------------------------------
PTSDEV                         2704.5 MB
PTSIMG                         4030 MB
PTS_STUCRSREG                  1518 MB

14 rows selected.
```

Lesson 84: Logical Storage Structure & Hierarchy

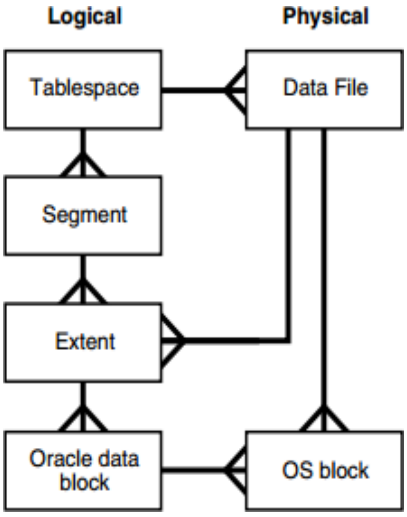
- Physical & Logical Database
- Tablespace
- Database Block
- Database Block, Extents and Segments
- Database Block, Extents and Segments within Database

Physical & Logical Database

- Physical Database
  - Control Files
  - Redo Log Files
  - Data Files
- Logical Database
  - Tablespace
    - Segments
    - Extents
    - DB Blocks

Control files: Structure of DB, DB Name, location Redo Log files: Redo Log groups, storing all types of uncommitted data in a cyclic style Data Files: Actual data of enterprise logically managed under Tablespaces

Physical & Logical Database



Logical units of database space allocation are data bocks, extents and TSs

The data in the data files is stored in operating system blocks

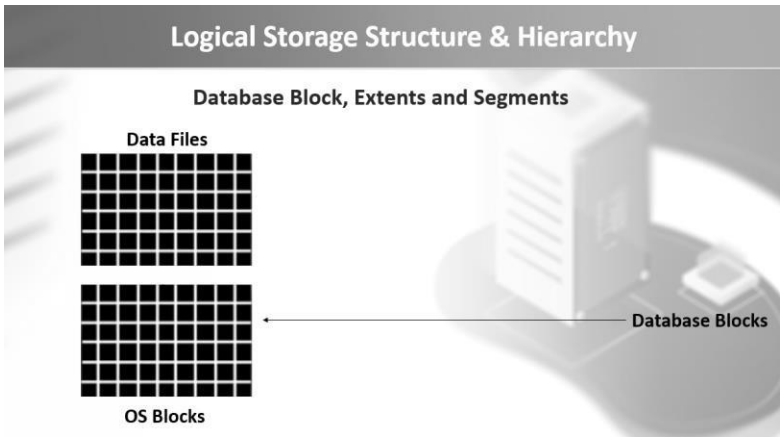
Tablespace

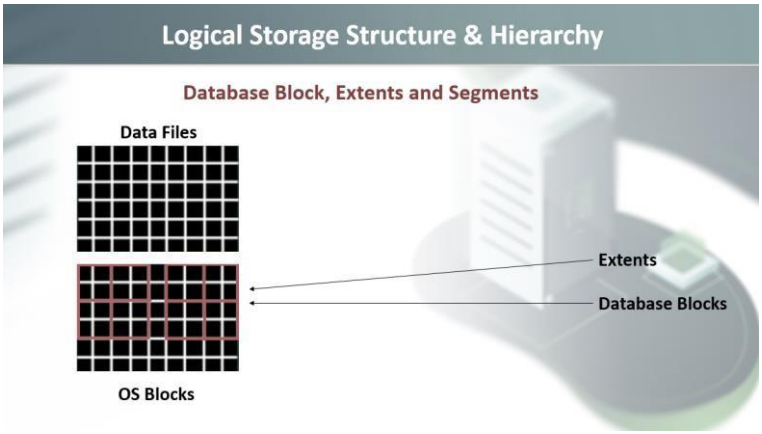
- Tablespace stores space logically to create logical objects such as Tables, Indexes and other objects.
- Tablespace's physically space stored in associated files.

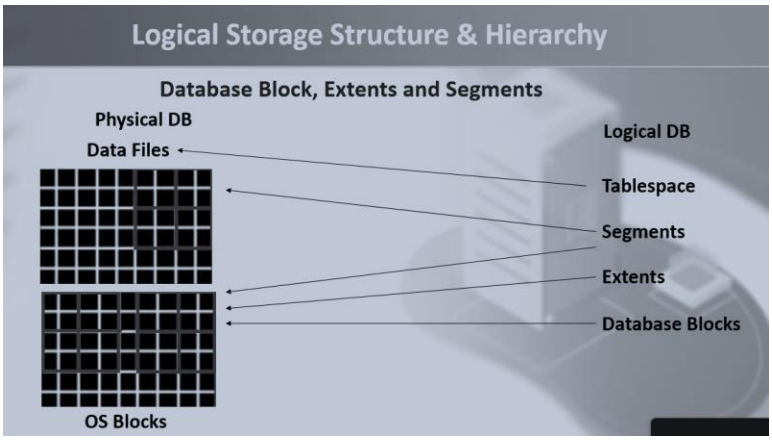
Database Block

- Logical minimum storage unit is called database block.
- Its size is decided during database creation.
- 5 default sizes: 2K, 4K, 8K, 16K, 32K (K: Kilobytes)

Database Block, Extents and Segments







One of block sizes are chosen.

At the finest level of granularity, Oracle Database stores data in data blocks. One logical data block corresponds to a specific number of bytes of physical disk space, for example, 2 KB. Data blocks are the smallest units of storage that Oracle Database can use or allocate.

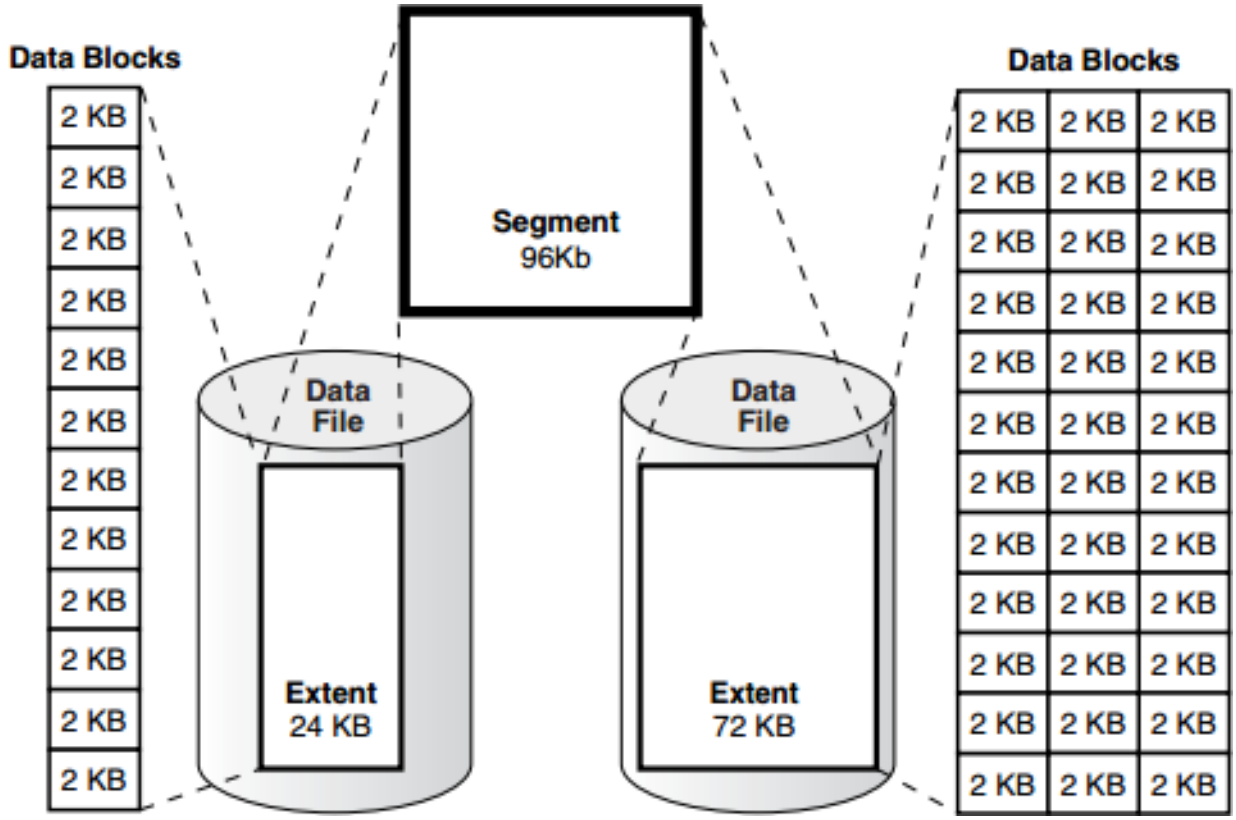
An extent is a set of logically contiguous data blocks allocated for storing a specific type of information. In Figure 12-2, the 24 KB extent has 12 data blocks, while the 72 KB extent has 36 data blocks.

A segment is a set of extents allocated for a specific database object, such as a table. For example, the data for the employees table is stored in its own data segment, whereas each index for employees is stored in its own index segment. Every database object that consumes storage consists of a single segment.

Each segment belongs to one and only one tablespace. Thus, all extents for a segment are stored in the same tablespace. Within a tablespace, a segment can include extents from multiple data files, as shown in Figure 12-2. For example, one extent for a segment may be stored in users01.dbf, while another is stored in users02.dbf. A single extent can never span data files

Database Block, Extents and Segments within Database

- At the finest level of granularity, Oracle Database stores data in data blocks.
- One logical data block corresponds to a specific number of bytes of physical disk space. Block sizes can be 2 KB, 4KB or more.
- Data block is the smallest unit of storage that Oracle Database can use or allocate.



Lesson 85: Overview of Database Blocks

- What is Data Block?
- Data Blocks & OS Blocks
- Interaction of Data Blocks & OS Blocks
- Database Block Size
- Data Block Format
- Row Format
- RowID Pseudocolumn

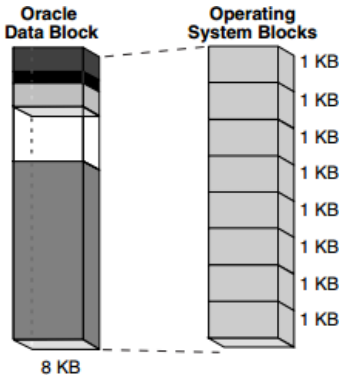
What is Data Block?

Oracle Database manages the logical storage space in the data files of a database in units called data blocks, also called Oracle blocks or pages. A data block is the minimum unit of database I/O.

Data Blocks & OS Blocks

- At the physical level, database data is stored in disk files made up of operating system blocks.
- An operating system block is the minimum unit of data that the operating system can read or write.
- In contrast, an Oracle block is a logical storage structure whose size and structure are not known to the operating system.

Figure shows that operating system blocks may differ in size from data blocks. The database requests data in multiples of data blocks, not operating system blocks.



Interaction of Data Blocks & OS Blocks

When the database requests a data block, the operating system translates this operation into a requests for data in permo9anent storage. The logical separation of data blocks from operating system blocks has the following implications:

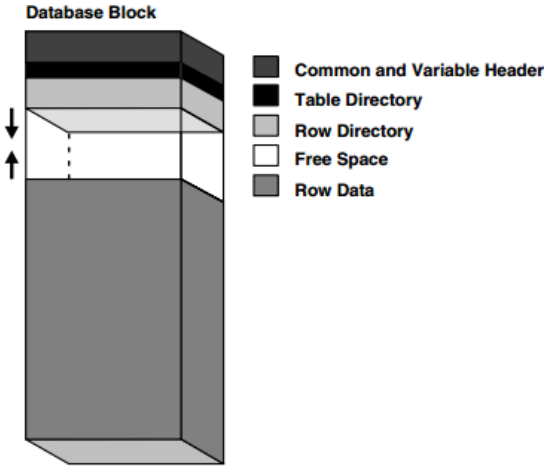
- Applications do not need to determine the physical addresses of data on disk.
- Database data can be striped or mirrored on multiple physical disks.

Database Block Size

- Every database has a database block size. The DB_BLOCK_SIZE initialization parameter sets the data block size for a database when it is created.
- The size is set for the SYSTEM and SYSAUX tablespaces and is the default for all other tablespaces.
- The database block size cannot be changed except by re-creating the database.
- If DB_BLOCK_SIZE is not set, then the default data block size is operating system-specific. The standard data block size for a database is4 KB or 8 KB.
- If the size differs for data blocks and operating system blocks, then the data block size must be a multiple of the operating system block size
- You can create individual tablespaces whose block size differs from the DB_BLOCK_SIZE setting.
- A nonstandard block size can be useful when moving a transportable tablespace to a different platform.

Data Block Format

Every data block has a format or internal structure that enables the database to track the data and free space in the block. This format is similar whether the data block contains table, index, or table cluster data



Block header

This part contains general information about the block, including disk address and segment type. For blocks that are transaction-managed, the block header contains active and historical transaction information.

A transaction entry is required for every transaction that updates the block. Oracle

Database initially reserves space in the block header for transaction entries. In data blocks allocated to segments that support transactional changes, free space can also hold transaction entries when the header space is depleted. The space required for transaction entries is operating system dependent. However, transaction entries in most operating systems require approximately 23 bytes.

■ Table direct ory

For a heap-organized table, this directory contains metadata about tables whose rows are stored in this block. Multiple tables can store rows in the same block.

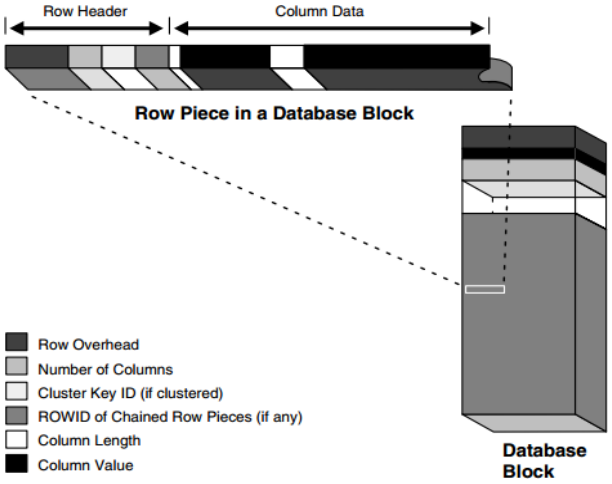
■ Row directory

For a heap-organized table, this directory describes the location of rows in the data portion of the block.

After space has been allocated in the row directory, the database does not reclaim this space after row deletion. Thus, a block that is currently empty but formerly had up to 50 rows continues to have 100 bytes allocated for the row directory. The database reuses this space only when new rows are inserted in the block.

Row Format

The row data part of the block contains the actual data, such as table rows or index key entries. Just as every data block has an internal format, every row has a row format that enables the database to track the data in the row.



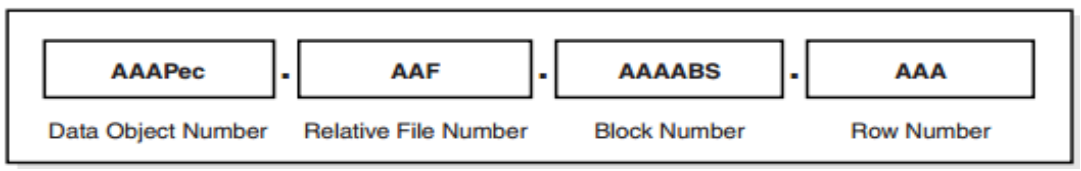
RowID Pseudocolumn

Queries the ROWID pseudocolumn to show the extended rowid of the row in the employee stable for employee 100

```
SQL> SELECT ROWID FROM employees WHERE employee_id = 100;

ROWID
------------------
AAAPecAAFAAAABSAAA
```

Figure 12–8 illustrates the format of an extended rowid.

Figure 12–8 ROWID Format



Lesson86: Percentage of Free Space in Blocks (PCTFREE)

Percentage of Free Space Blocks

- What is Free Space in Data Block?
- Data Blocks & OS Blocks
- Percentage of Free Space in Data Blocks
- PCTFREE Space in Data Block
- Optimizing of Free Space
- Reuse of Index Space
- Row Migration

What is Free Space in Data Block?

- As the database fills a data block from the bottom up, the amount of free space between the row data and the block header decreases. This free space can also shrink during updates, as when changing a trailing null to a non-null value.
- The database manages free space in the data block to optimize performance and avoid wasted space (Tuning Feature).

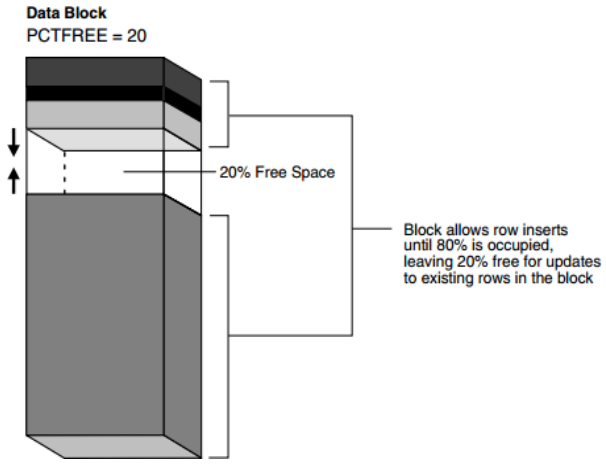
Data Blocks & OS Blocks

- At the physical level, database data is stored in disk files made up of operating system blocks.
- An operating system block is the minimum unit of data that the operating system can read or write.
- In contrast, an Oracle block is a logical storage structure whose size and structure are not known to the operating system.

Percentage of Free Space in Data Blocks

- The PCTFREE storage parameter is essential to how the database manages free space. This SQL parameter sets the minimum percentage of a data block reserved as free space for updates to existing rows.
- PCTFREE is important for preventing row migration and avoiding wasted space.

PCTFREE Space in Data Block



CREATE TABLESPACE userdata
datafile
'C:\Ora11g\ordata\usr01.dbf'
size 500M
EXTENT MANAGEMENT

PCTFREE parameter does not applied on LOB Datatypes

Shows how a PCTFREE setting of 20affects space management. The database adds rows to the block over time, causing the row data to grow upwards toward the block header, which is itself expanding **downward toward the row data.**
The PCTFREE setting ensures that at least20% of the data block is free. For example, the database prevents an INSERT statement from filling the block so that the row data and header occupy a combined 90% of the total block space, leaving only 10% free.

Optimizing of Free Space

- While the percentage of free space cannot be less than PCTFREE, the amount of free space can be greater. For example, a PCTFREE setting of 20% prevents the total amount of free space from dropping to 5% of the block, but permits 50% of the block to be free space.

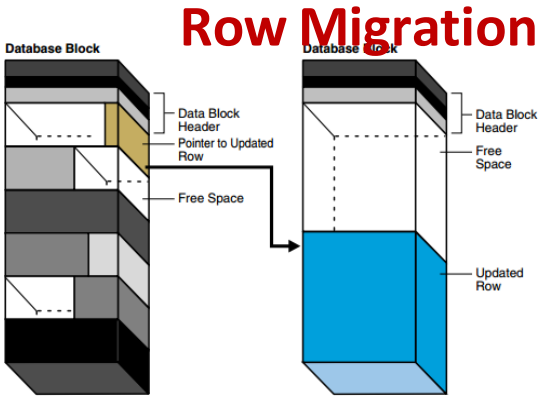
The following SQL statements can increase free space:

- DELETE statements
- UPDATE statements that either update existing values to smaller values or increase existing values and force a row to migrate
- INSERT statements on a table that uses OLTP compression. If inserts fill a block with data, then the database invokes block compression, which may result in the block having more free space.
- The space released is available for INSERT under the following conditions:
- If the INSERT statement is in the same transaction and after the statement that frees space, then the statement can use the space.
- If the INSERT statement is in a separate transaction from the statement that frees space (perhaps run by another user), then the statement can use the space made available only after the other transaction commits and only if the space is needed.

Reuse of Index Space

The database can reuse space within an index block.

For example, if you insert a value into a column and delete it, and if an index exists on this column, then the database can reuse the index slot when a row requires it



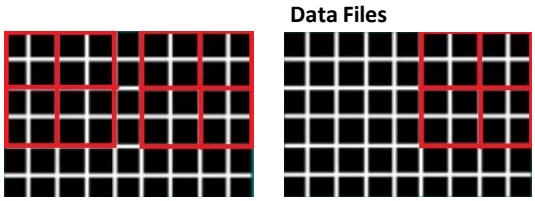
The left block contains a row that is updated so that the row is now too large for the block. The database moves the entire row to the right block and leaves a pointer to the migrated row in the left block

Lesson87: Overview of Extents

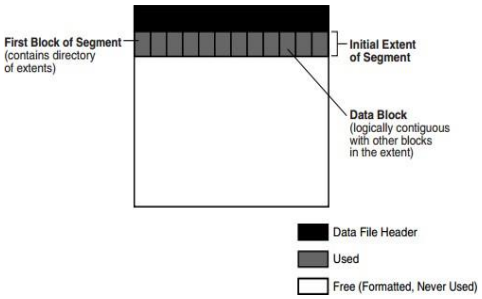
- What is Extent?
- Incremental Extents of Segment
- Deallocation of Extents
- Manually Deallocate Space
- Sample SQL DDL using Extent

What is Extent?

An extent is a logical unit of database storage space allocation made up of contiguous data blocks. Data blocks in an extent are logically contiguous but can be physically spread out on disk because of RAID striping and file system implementations.



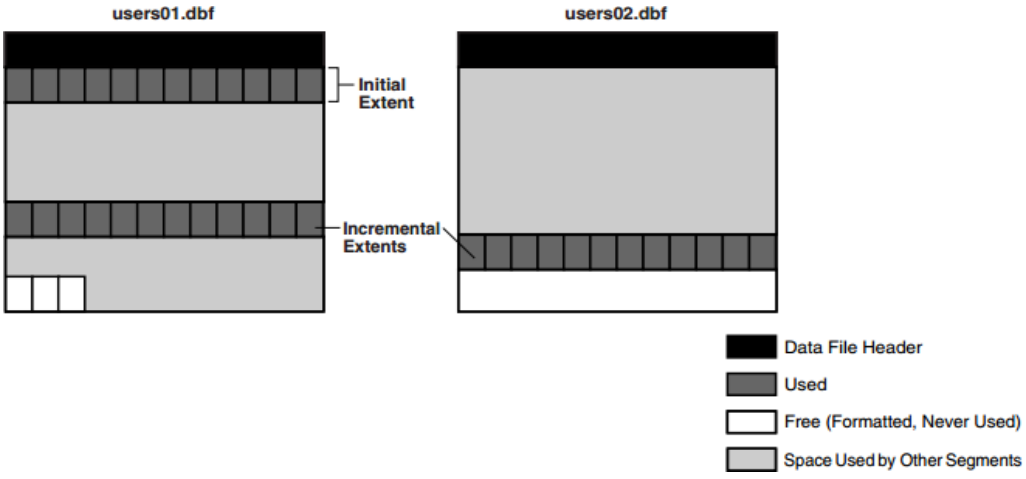
What is Extents?



By default, the database allocates an initial extent for a data segment when the segment is created. An extent is always contained in one data file. Although no data has been added to the segment, the data blocks in the initial extent are reserved for this segment exclusively. The first data block of every segment contains a directory of the extents in the segment.

Figure shows the initial extent in a segment in a data file that previously contained no data.

Incremental Extents of Segment



If the initial extent become full, and if more space is required, then the database automatically allocates an incremental extent for this segment. An incremental extent is a subsequent extent created for the segment. The allocation algorithm depends on whether the tablespace is locally managed or dictionary-managed. In the locally managed case, the database searches the bitmap of a data file for adjacent free blocks. If the data file has insufficient space, then the database looks in another data file. Extents for a segment are always in the same tablespace but may be in different data files. Figure shows that the database can allocate extents for a segment in any data file in the tablespace. For example, the segment can allocate the initial extent in users01.dbf, allocate the first incremental extent in users02.dbf, and allocate the next extent in users01.dbf.

Deallocation of Extents

In general, the extents of a user segment do not return to the tablespace unless you drop the object using a DROP command. In Oracle Database 11g Release 2, you can also drop the segment using the DBMS_SPACE_ADMIN package. For example, if you delete all rows in a table, then the database does not reclaim the data blocks for use by other objects in the tablespace.

Manually Deallocate Space

Following techniques can deallocate extents

- You can use an online segment shrink to reclaim fragmented space in a segment. Segment shrink is an online, in-place operation.
- You can move the data of a non partitioned table or table partition into a new segment, and optionally into a different tablespace for which you have quota.
- You can rebuild or coalesce the index
- You can truncate a table or table cluster, which removes all rows. By default, Oracle Database deallocates all space used by the removed rows except that specified by the MINEXTENTS storage parameter.
- You can deallocate unused space, which frees the unused space at the high water mark end of the database segment and makes the space available for other segments in the tablespace

Sample SQL DDL using Extent

```
CREATE TABLESPACE MCare_indx

DATAFILE 'D:\oracle11g\ora11g\oradata\MCare_indx01.dbf'

SIZE 100M

MINIMUM EXTENT 64K

DEFAULT STORAGE (INITIAL 64K NEXT 64K);
```

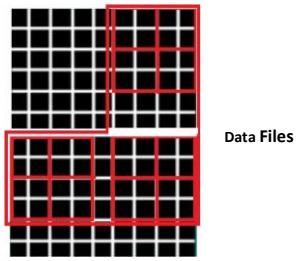
Lesson88: Overview of Segments

- What is Segment?
- Segment & Data Files
- User Segments
- User Segments Creation
- Creation of Multiple Segments
- SQL DDL for Segments under Tablespace
- Temporary Segments
- UNDO Segments
- Tablespace with Segments

What is Segment?

A segment is a set of extents that contains all the data for a logical storage structure within a tablespace. For example, Oracle Database allocates one or more extents to form the data segment for a table. The database also allocates one or more extents to form the index segment for a table implementations.

Segment & Data Files



For example, Patient table consists of three segments from two data files

Collection of extents, may be from multiple data files is called Segments. Collection of segments is called Tablespace

User Segments

A single data segment in a database stores the data for one user object. There are different types of segments. Examples of user segments include:

- Table, table partition, or table cluster
- LOB or LOB partition
- Index or index partition

Each non partitioned object and object partition is stored in its own segment. For example, if an index has five partitions, then five segments contain the index data.

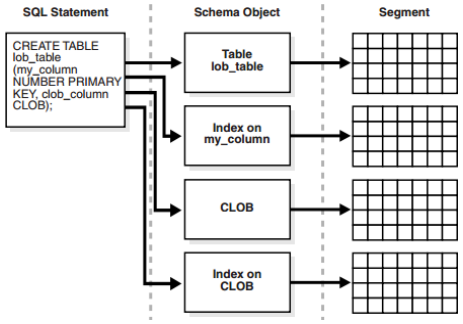
User Segments Creation

By default, the database uses deferred segment creation to update only database metadata when creating tables and indexes.
When a user inserts the first row into a table or partition, the database creates segments for the table or partition, its LOB columns, and its indexes.
Deferred segment creation enables you to avoid using database resources unnecessarily.

For example, installation of an application can create thousands of objects, on consuming significant disk space. Many of these objects may never be used.

You can use the DBMS_SPACE_ADMIN package to manage segments for empty objects.

Creation of Multiple Segments



CREATE TABLE PATIENT_HISTORY

(PATIENT_ID CHAR (007),

DEPEND_SNO VARCHAR2 (002),

ALLERGY_DESC VARCHAR2 (250) NOT NULL,

HISTORY_DESC LONG

) tablespace MCare;

ALTER TABLE PATIENT_HISTORY

ADD constraint PK_PATHIST_ID PRIMARY KEY (PATIENT_ID,DEPEND_SNO)

USING INDEX TABLESPACE MCare_indx;

Temporary Segments

When processing a query, Oracle Database often requires temporary workspace for intermediate stages of SQL statement execution.
Typical operations that may require a temporary segment include sorting, hashing, and merging bitmaps. While creating an index, Oracle Database also places index segments into temporary segments and then converts them into permanent segments when the index is complete

UNDO Segments

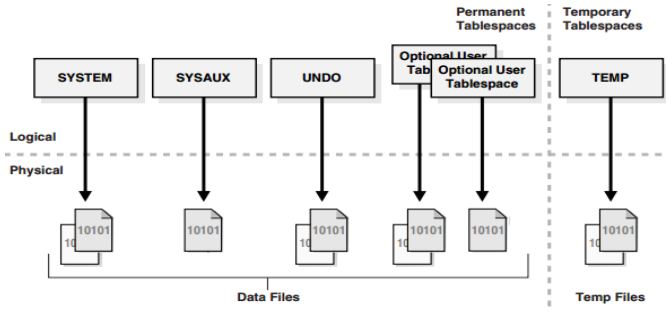
Oracle Database maintains records of the actions of transactions, collectively known as undo data.

Oracle Database uses undo to do the following:

- Roll back an active transaction
- Recover a terminated transaction
- Provide read consistency
- Perform some logical flashback operations

Tablespace with Segments

A tablespace is a logical storage container for segments. Segments are database objects, such as tables and indexes, that consume storage space. At the physical level, a tablespace stores data in one Or more data files or temp files.



Lesson89: Creating Schema Objects Tables, Views

Overview of Segments

- Introduction to Schema Objects
- User & Schema Objects
- Schema Objects Types
- Overview of Views
- Use of Views
- SQL to Create View

• View of Base Table

Introduction to Schema Objects

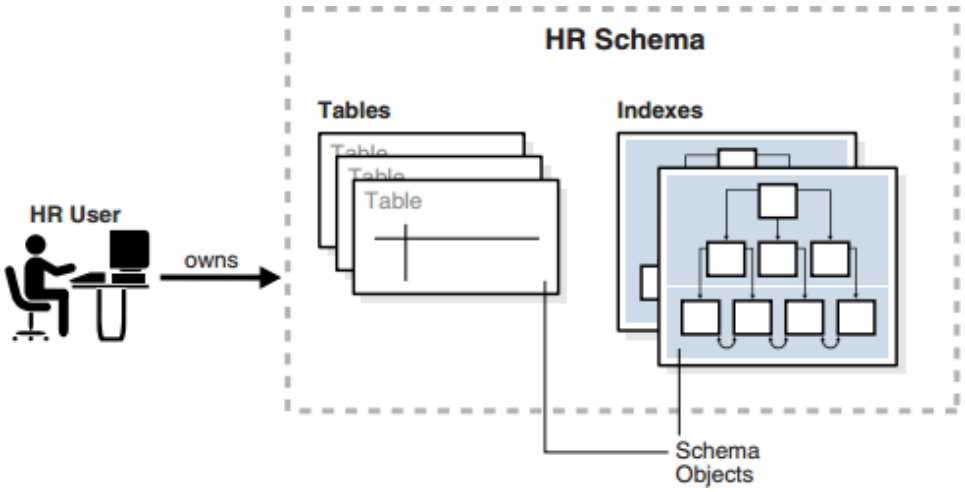
- A database schema is a logical container for data structures, called schema objects.
- Examples of schema objects are tables, views, sequences and indexes etc.
- Schema objects are created and manipulated with SQL

A database user has a password and various database privileges. Each user owns a single schema, which has the same name as the user.

The schema contains the data for the user owning the schema. For example, the hr user owns the hr schema, which contains schema objects such as the employees table.

In a production database, the schema owner usually represents a database application rather than a person.

User & Schema Objects



Schema Objects Types

- Tables

Storing data in rows of tables

- Indexes

Indexes are schema objects that contain an entry for each indexed row of the table or table cluster and provide direct, fast access to rows.

- Partitions

Partitions are pieces of large tables and indexes.

- Views

Views are customized presentations of data in one or more tables or other views.

- Sequences

A sequence is a user-created object that can be shared by multiple users to generate integers.

- Dimensions

A dimension defines a parent-child relationship between pairs of column sets, where all the columns of a column set must come from the same table.

- Synonyms

A synonym is an alias for another schema object.

- PL/SQL Programs

Procedural server programming

Overview of Views

- A view is a logical representation of one or more tables.
- In essence, a view is a stored query.
- A view derives its data from the tables on which it is based, called base tables.
- Base tables can be tables or other views.
- All operations performed on a view actually affect the base tables.
- You can use views in most places where tables are used.

Use of Views

- Provide an additional level of table security by restricting access to a predetermined set of rows or columns of a table
- Hide data complexity
- Present the data in a different perspective from that of the base table

For example, the columns of a view can be renamed without affecting the tables on which the view is based.

- Isolate applications from changes in definitions of base tables used.

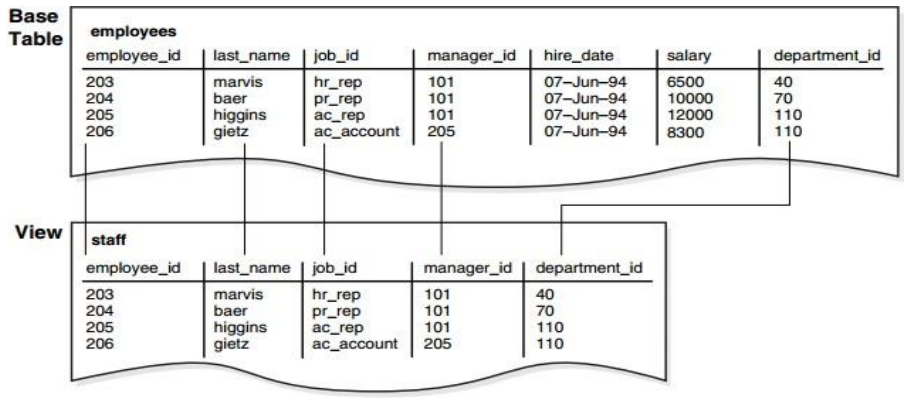
SQL to Create View

CREATE VIEW staff AS

SELECT employee_id, last_name, job_id, manager_id, department_id

FROM employees;

View of Base Table



Lesson 90: Analyzing Tables, Indexes & Clusters

- Overview of Table Clusters
- Cluster Key
- Overview of Indexed Clusters
- DDL for Indexed Clusters
- Overview of Indexes
- Composite Indexes

Overview of Table Clusters

- A table cluster is a group of tables that share common columns and store related data in the same blocks. When tables are clustered, a single data block can contain rows from multiple tables.
- For example, a block can store rows from both the employees and Departments tables rather than from only a single table.

Cluster Key

- The cluster key is the column or columns that the clustered tables have in common.
- For example, the employees and departments tables share the department_id column.
- You specify the cluster key when creating the table cluster and when creating every table added to the table cluster.

Cluster Key Value

- The cluster key value is the value of the cluster key columns for a particular set of rows. All data that contains the same cluster key value, such as department_id=20, is physically stored together. Each cluster key value is stored only once in the cluster and the cluster index, no matter how many rows of different tables contain the value

Overview of Indexed Clusters

An indexed cluster is a table cluster that uses an index to locate data. The cluster index is a B-tree index on the cluster key. A cluster index must be created before any rows can be inserted into clustered tables

DDL for Indexed Clusters

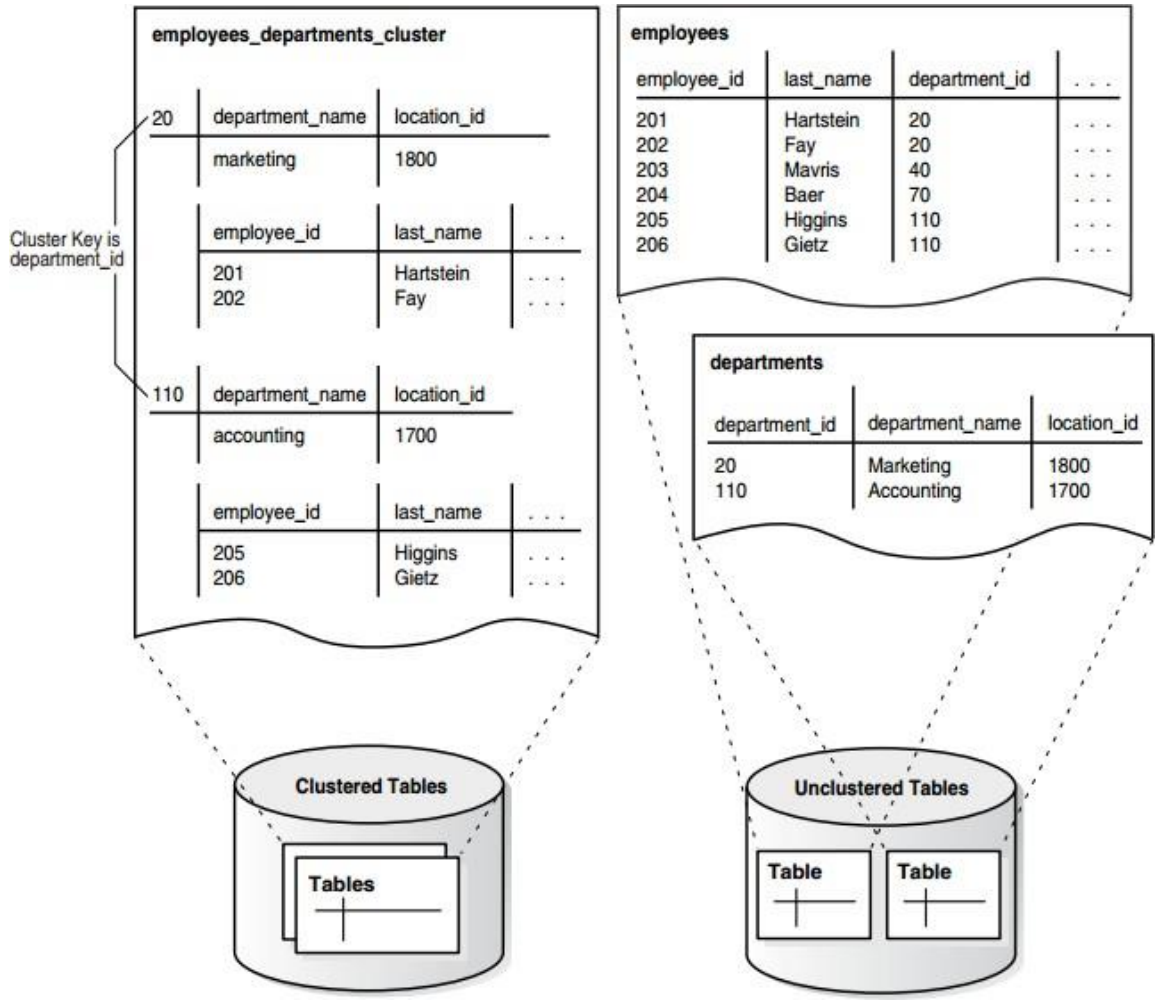
CREATE CLUSTER emp_dept_cluster

(department_id NUMBER(4))

SIZE 512;

CREATE INDEX idx_emp_dept_cluster ON CLUSTER emp_dept_cluster;

DDL for Indexed Clusters



Overview of Indexes

An index is an optional structure, associated with a table or table cluster, that can sometimes speed data access. By creating an index on one or more columns of a table, you gain the ability in some cases to retrieve a small set of randomly distributed rows from the table.

Indexes are one of many means of reducing disk I/O.

A composite index, also called a concatenated index, is an index on multiple columns in a table. Columns in a composite index should appear in the order that makes the most sense for the queries that will retrieve data and need not be adjacent in the table.

CREATE INDEX employees_ix

ON employees (last_name, job_id, salary);

Lesson91: Managing Integrity Constraints-1

Integrity Constraints

- Data Integrity
- Techniques for Data Integrity
- Advantages of Integrity Constraints
- Types of Integrity Constraints

Data Integrity

- Business rules specify conditions and relationships that must always be true or must always be false. For example, each company defines its own policies about salaries, employee numbers, inventory tracking, and so on.
- It is important that data maintain data integrity, which is adherence to these rules, as determined by the database administrator or application developer.

Techniques for Data Integrity

Following are possible options:

- Enforcing business rules with triggered stored database procedures
- Using stored procedures to completely control access to data
- Enforcing business rules in the code of a database application
- Using Oracle Database integrity constraints, which are rules defined at the column or object level that restrict values in the database

Advantages of Integrity Constraints

An integrity constraint is a schema object that is created and dropped using SQL.

Advantages of integrity constraints over alternatives for enforcing data integrity include:

- Declarative ease
- Centralized rules
- Flexibility when loading data

Declarative ease

Because you define integrity constraints using SQL statements, no additional programming is required when you define or alter a table. The SQL statements are easy to write and eliminate programming errors.

■ Centralized rules

Integrity constraints are defined for tables and are stored in the data dictionary (see "Overview of the Data Dictionary“ on page 6-1). Thus, data entered by all applications must adhere to the same integrity constraints. If the rules change at the table level, then applications need not change. Also,

applications can use metadata in the data dictionary to immediately inform users of violations, even before the database checks the SQL statement.
■ Flexibility when loading data
You can disable integrity constraints temporarily to avoid performance overhead when loading large amounts of data. When the data load is complete, you can re-enable the integrity constraints.

Advantages of Integrity Constraints

- Declarative ease Because you define integrity constraints using SQL DDL statements.
- Centralized rules Integrity constraints are defined for tables and are stored in the data dictionary
- Flexibility when loading data You can disable integrity constraints temporarily to avoid performance overhead when loading large amounts of data.

Types of Integrity Constraints

Oracle Database enables you to apply constraints both at the table and column level. A constraint specified as part of the definition of a column or attribute is called an inline specification.

NOT NULL

UNIQUE

CHECK

DEFAULT

PK/ FK: Standard Integrity Constraints

Lesson92: Managing Integrity Constraints-2

Integrity Constraints

- NOT NULL Constraint
- UNIQUE Constraint
- CHECK Constraint
- DEFAULT Constraint
- FK CASCADE Constraints
- FK CASCADE Constraints – Example
- Trigger's Constraint
- ENABLE/ DISABLE or Drop Constraints

NOT NULL Constraint

- A NOT NULL constraint requires that a column of a table contain no null values. A null is the absence of a value. By default, all columns in a table allow nulls.
- By default, FK is NULL.

Create table Student(

(RollNo CHAR(10),

FullName VARCHAR2(50) NOT NULL,

…..);

UNIQUE Constraint

- A unique key (Null able) constraint requires that every value in a column or set of columns be unique. No rows of a table may have duplicate values in a column (the unique key) or set of columns (the composite unique key) with a unique key constraint.

Create table Medicine(

(ItemNo CHAR(10),

Name VARCHAR2(50) NOT NULL UNIQUE,

…..);

CHECK Constraint

- A check constraint on a column or set of columns requires that a specified condition be true or unknown for every row. If DML results in the condition of the constraint evaluating to false, then the SQL statement is rolled back.

Alter table Medicine

Add constraint chk_medType

CHECK (MedType IN ('S','P'));

DEFAULT Constraint

- A default constraint on a column requires that a specified value be true when null value is entered.

Create table Medicine(

(ItemNo CHAR(10),

Name VARCHAR2(50) NOT NULL,

Status CHAR(1) default 'Y‘,

…..);

FK CASCADE Constraints

- CASCADE is applied with INSERT, DELETE or UPDATE.
- For example, a foreign key with cascade delete means that if a record in the parent table is deleted, then the corresponding records in the child table will automatically be deleted. This is called a cascade delete in Oracle.

FK CASCADE Constraints - Example

CREATE TABLE supplier

(supplier_id numeric(10) not null,

supplier_name varchar2(50) not null,

contact_name varchar2(50),

CONSTRAINT supplier_pk PRIMARY KEY (supplier_id));

CREATE TABLE products

(product_id numeric(10) not null,

supplier_id numeric(10) not null,

CONSTRAINT fk_supplier FOREIGN KEY (supplier_id)

REFERENCES supplier(supplier_id) ON DELETE CASCADE);

Trigger’s Constraints

CREATE TRIGGER cascade_del_stu_record

BEFORE DELETE ON STUDENT

FOR EACH ROW

BEGIN

DELETE FROM RECORD

```
  WHERE RECORD.SID = :OLD.SID;
END;
CREATE TRIGGER cascade_del_stu_record
BEFORE DELETE ON STUDENT
FOR EACH ROW
BEGIN
  DELETE FROM RECORD
  WHERE RECORD.SID = :OLD.SID;
END;
```

ENABLE/ DISABLE or Drop Constraints

Existing constraints on table can be

- Enabled
- Disabled
- Dropped

For example,

```
Alter table Invoice
DROP constraint fk_custid;
```

Lesson93: Schema Objects, Queries & Views

Schema Objects

- How a Schema is Created?
- How a Schema is Implemented?
- Simple & Complex Views
- Creating Tables
- Creating Sequence or AutoNumber
- Data Dictionary for Tables & Views

How a Schema is Created

- We have studied that DBA has to create default Tablespace and assign temporary Tablespace with storage in terms of data files.
- Create first user (owner of schema) with grants such as rights and privileges.
- We will cover schema objects Tables and Views.
- Ref. DBA-73

How a Schema is Implemented?

- A view is a logical representation of a table or combination of tables.
- In essence, a view is a stored query.
- A view derives its data from the tables on which it is based.
- These tables are called base tables. Base tables might in turn be actual tables or might be views themselves.
- All operations performed on a view actually affect the base table of the view.

- You can use views in almost the same way as tables. You can query, update, insert into, and delete from views, just as you can standard tables.
- Views can provide a different representation (such as subsets or supersets) of the data that resides within other tables and views.
- Views can be Simple or Complex
- Views can be used for security purposes
- View does not contain its own data

Simple & Complex Views

CREATE VIEW sales_staff AS

SELECT empno, ename, deptno

FROM emp

WHERE deptno = 10;

CREATE VIEW sales_staff AS

SELECT job, count(*)

FROM emp

WHERE deptno = 10

Group by job;

Simple & Complex Views

CREATE OR REPLACE VIEW division1_staff AS

SELECT ename, empno, job, dname

FROM emp, dept

WHERE emp.deptno IN (10, 30)

AND emp.deptno = dept.deptno;

Insert, delete and update operations for base tables are applied on simple views.

Creating Tables

CREATE TABLE dept (
deptno NUMBER(4) PRIMARY KEY,
dname VARCHAR2(14),
loc VARCHAR2(13));
CREATE TABLE emp (
empno NUMBER(4) PRIMARY KEY,
ename VARCHAR2(10),
…
deptno NUMBER(2);
…

Creating Sequence or AutoNumber

CREATE SEQUENCE emp_sequence

INCREMENT BY 1

START WITH 1

NOMAXVALUE

NOCYCLE

CACHE 10;

Data Dictionary for Tables & Views

SELECT COLUMN_NAME, UPDATABLE

FROM USER_UPDATABLE_COLUMNS

WHERE TABLE_NAME = 'EMP_DEPT';

```
SELECT COLUMN_NAME, UPDATABLE
       FROM USER_UPDATABLE_COLUMNS
       WHERE TABLE_NAME = 'EMP_DEPT';

COLUMN_NAME                    UPD
------------------------------ ---
EMPNO                          YES
ENAME                          YES
DEPTNO                         YES
SAL                            YES
DNAME                          NO
LOC                            NO
```

Lesson94: Creating Tables & Alter Tables

Creating & Alter Tables

- What is a Table?
- Design Table Guidelines
- Heap-Organized Table
- Clustered Table
- Indexed-Organized Table
- Partitioned Table
- Storage Location of a Table
- Alter Table
- Tables with Segment Creation

What is a Table?

- Tables are the basic unit of data storage in an Oracle Database. Data is stored in rows and columns. You define a table with a table name, such as employees, and a set of columns.
- You give each column a column name, such as emp_id, last_name, and job_id; a data type, such as VARCHAR2, DATE, or NUMBER; and a width. The width can be predetermined by the data type, as in DATE.
- If columns are of the NUMBER data type, define precision and scale instead of width. A row is a collection of column information corresponding to a single record.

- You can specify rules for each column of a table. These rules are called integrity constraints. One example is a NOT NULL integrity constraint. This constraint forces the column to contain a value in every row.
- Some column types, such as LOBs, varrays, and nested tables, are stored in their own segments. LOBs and varrays are stored in LOB segments, while nested tables are stored in storage tables.

Design Table Guidelines

- Use descriptive names for tables, columns, indexes, and clusters.
- Be consistent in abbreviations and in the use of singular and plural forms of table names and columns.
- Document the meaning of each table and its columns with the COMMENT command.
- Normalize each table.
- Select the appropriate datatype for each column. Consider whether your applications would benefit from adding one or more virtual columns to some tables.
- Define columns that allow nulls last, to conserve storage space.
- Cluster tables whenever appropriate, to conserve storage space and optimize performance of SQL statements.

Heap-Organized Table

- This is the basic, general purpose type of table which is the primary subject of this chapter. Its data (heap).

Clustered Table

- A clustered table is a table that is part of a cluster. A cluster is a group of tables that share the same data blocks because they share common columns and are often used together

Indexed-Organized Table

- Unlike an ordinary (heap-organized) table, data for an index-organized table is stored in a B-tree index structure in a primary key sorted manner.
- Besides storing the primary key column values of an index-organized table row, each index entry in the B-tree stores the non key column values as well

Partitioned Table

- Partitioned tables enable your data to be broken down into smaller, more manageable pieces called partitions, or even subpartitions. Each partition can have separate physical attributes, such as compression enabled or disabled, type of compression, physical storage settings, and tablespace, thus providing a structure that can be better tuned for availability and performance.
- In addition, each partition can be managed individually, which can simplify and reduce the time required for backup and administration.

Storage Location of a Table

- It is advisable to specify the TABLESPACE clause in a CREATE TABLE statement to identify the tablespace that is to store the new table.
- For partitioned tables, you can optionally identify the tablespace that is to store each partition. Ensure that you have the appropriate privileges and quota on any tablespaces that you use.
- If you do not specify a tablespace in a CREATE TABLE statement, the table is created in your default tablespace.

Alter Table

- Alter table command applies when table is already exist in a schema.
- We can add constraints, columns and modifications operations on table using Alter table DDL statements.

Tables with Segment Creation

```
CREATE TABLE part_time_employees (
    empno NUMBER(8),
    name VARCHAR2(30),
    hourly_rate NUMBER (7,2)
    )
    SEGMENT CREATION DEFERRED;

CREATE TABLE hourly_employees (
    empno NUMBER(8),
    name VARCHAR2(30),
    hourly_rate NUMBER (7,2)
    )
   SEGMENT CREATION IMMEDIATE
   PARTITION BY RANGE(empno)
    (PARTITION empno_to_100 VALUES LESS THAN (100),
    PARTITION empno_to_200 VALUES LESS THAN (200));
```

```
SELECT segment_name, partition_name FROM user_segments;

SEGMENT_NAME         PARTITION_NAME
-------------------- ------------------------------
HOURLY_EMPLOYEES     EMPNO_TO_100
HOURLY_EMPLOYEES     EMPNO_TO_200
```

Lesson95:DDL Script Writing-1

Script Writing

- What is a Script?
- A Sample DDL in a Script?
- Creating Temporary Tables
- Table with TEMP Tablespace
- Bulk Data Load in a Table
- Alter Table
- Index with Tablespace
- DDL Partition Table

What is a Script?

- Normally writing or generating DDL commands to create/ alter tables, views, constraints and storages etc., in an order in plain text using Notepad or text files
- Mostly scripts contain structure of tables. Sometimes, DML commands are also put in scripts as per management requirement.
- Scripts can be with Create, Alter and Drop Tables

A Sample DDL in a Script

```
CREATE TABLE hr.admin_emp (
         empno      NUMBER(5) PRIMARY KEY,
         ename      VARCHAR2(15) NOT NULL,
         ssn        NUMBER(9) ENCRYPT USING 'AES256',
         job        VARCHAR2(10),
         mgr        NUMBER(5),
         hiredate   DATE DEFAULT (sysdate),
         photo      BLOB,
         sal        NUMBER(7,2),
         hrly_rate  NUMBER(7,2) GENERATED ALWAYS AS (sal/2080),
         comm       NUMBER(7,2),
         deptno     NUMBER(3) NOT NULL
                     CONSTRAINT admin_dept_fkey REFERENCES hr.departments
                     (department_id))
   TABLESPACE admin_tbs
   STORAGE ( INITIAL 50K);

COMMENT ON TABLE hr.admin_emp IS 'Enhanced employee table';
```

Creating Temporary Tables

Temporary tables are useful in applications where a result set is to be buffered (temporarily persisted), perhaps because it is constructed by running multiple DML operations.

For example, consider the following:

```
CREATE GLOBAL TEMPORARY TABLE admin_work_area
        (startdate DATE,
         enddate DATE,
         class CHAR(20))
      ON COMMIT DELETE ROWS;
```

Table with TEMP Tablespace

```
CREATE TEMPORARY TABLESPACE tbs_t1
    TEMPFILE 'tbs_t1.f' SIZE 50m REUSE AUTOEXTEND ON
    MAXSIZE UNLIMITED
    EXTENT MANAGEMENT LOCAL UNIFORM SIZE 64K;

CREATE GLOBAL TEMPORARY TABLE admin_work_area
        (startdate DATE,
         enddate DATE,
         class CHAR(20))
       ON COMMIT DELETE ROWS
       TABLESPACE tbs_t1;
```

Bulk Data Load in a Table

```
FORALL i IN 1..numrecords
  INSERT /*+ APPEND_VALUES */ INTO orderdata
  VALUES(ordernum(i), custid(i), orderdate(i),shipmode(i), paymentid(i));
COMMIT;
```

Alter Table

```
ALTER TABLE hr.admin_emp MOVE
      STORAGE ( INITIAL 20K
                NEXT 40K
                MINEXTENTS 2
                MAXEXTENTS 20
                PCTINCREASE 0 )
  TABLESPACE hr_tbs;
```

```
ALTER TABLE hr.admin_emp
      ADD (bonus NUMBER (7,2));
```

Index with Tablespace

CREATE INDEX int_sales_index ON int_salestable

(s_saledate, s_productid, s_custid)

TABLESPACE tbs_low_freq;

DDL Partition Table

```
CREATE TABLE salestable
(s_productid NUMBER,
s_saledate DATE,
s_custid NUMBER,
s_totalprice NUMBER)
TABLESPACE users
PARTITION BY RANGE(s_saledate)
(PARTITION sal03q1 VALUES LESS THAN (TO_DATE('01-APR-2003', 'DD-MON-YYYY')),
PARTITION sal03q2 VALUES LESS THAN (TO_DATE('01-JUL-2003', 'DD-MON-YYYY')),
PARTITION sal03q3 VALUES LESS THAN (TO_DATE('01-OCT-2003', 'DD-MON-YYYY')),
PARTITION sal03q4 VALUES LESS THAN (TO_DATE('01-JAN-2004', 'DD-MON-YYYY')));
```

Lesson96: DDL Script Writing-2

Script Writing

- Writing Script using Notepad
- Tables in an Order
- Drop Tables in an Order
- Other Schema Objects

Writing Script using Notepad

- Script is a plain text structure of schema

Cust-Ord-Prod Schema script.txt - Notepad

File Edit Format View Help

```
create table Customer
(custID CHAR(10),
CustName VARCHAR2(25) NOT NULL,
Address  VARCHAR2(50),
City  VARCHAR2(50),
TelNo VARCHAR2(15),
Email VARCHAR2(50)
);

Alter table Customer
add constraint PK_Custid primary key (CustID);

Create table Orders
(InvoiceNo CHAR(12),
InvType CHAR(1),
InvDate Date NOT NULL,
TotalPrice Number(6,2),
CustID CHAR(8),
SalesEmpID CHAR(5)
);

Alter table Orders
add constraint PK_InvNo primary key (InvoiceNO);

Alter table Orders
add constraint FK_Cust_CustID foreign key (custID)
references Customer(CustID);

create table Employee
(EmpID CHAR(3)
```

Tables in an Order

```
create table Customer
(custID          CHAR(10),
CustName         VARCHAR2(25) NOT NULL,
Address          VARCHAR2(50),
City             VARCHAR2(50),
TelNo            VARCHAR2(15),
Email            VARCHAR2(50) UNIQUE)
tablespace TS_Sales;

Alter table Customer
add constraint PK_Custid primary key (CustID);
```

```
Create table Orders
(InvoiceNo       CHAR(12),
InvType          CHAR(1),
InvDate          Date NOT NULL,
TotalPrice       Number(6,2),
SalesEmpID       CHAR(5),
CustID           CHAR(8))
tablespace TS_Sales;

Alter table Orders
add constraint PK_InvNo primary key (InvoiceNO)
USING INDEX TABLESPACE Sale_indx;

Alter table Orders
add constraint FK_Cust_CustID foreign key (custID)
references Customer(CustID);
```

```
create table Employee
(EmpID           CHAR(3),
EmpName          VARCHAR2(25) NOT NULL,
Address          VARCHAR2(50),
City             VARCHAR2(50),
TelNo            VARCHAR2(15),
Email            VARCHAR2(50) UNIQUE)
tablespace TS_Sales;

Alter table Employee
add constraint PK_EmpID primary key (EmpID)
USING INDEX TABLESPACE Sale_indx;

Alter table Orders
add constraint FK_Emp_EmpID foreign key (SalesEmpID)
references Employee(EmpID)
USING INDEX TABLESPACE Sale_indx;
```

```
Create table Item
(ItemID          CHAR(5),
ItemName         VARCHAr2(50) NOT NULL,
PPrice           NUMBER(10,2),
TotQty           NUMBER(5),
Status           CHAR(1))
tablespace TS_Stock;

Alter table Item
add constraint PK_ItemID primary key (ItemID)
USING INDEX TABLESPACE Sale_indx;
```

```
Create table Sold
(
ItemID          CHAR(5),
InvoiceNo       CHAR(12),
SPrice          NUMBER(10,2) Not Null,
SQty            NUMBER(10,2) Not Null)
tablespace TS_ItemSold;

Alter table Sold
add constraint PK_Sold primary key (ItemID, InvoiceNo)
USING INDEX TABLESPACE Sold_indx;

Alter table Sold
add constraint FK_Sold_ItemID foreign key (ItemID)
references Item(ItemID);

Alter table Sold
add constraint FK_Sold_InvoiceNo foreign key (InvoiceNo)
references Invoice(InvoiceNo);
```

Drop Tables in an Order

```
Drop table Sold;
Drop table employee;
Drop table orders;
Drop table customer;
Drop table Item;
```

Other Schema Objects

Script may contains other objects such as

Views

Sequences

Synonym

PL/SQL Triggers & Code

DML for sample data

Lesson97:Drop Tables & Recycle Bin

Overview of Segments

- Drop Tables
- Drop Tables Precautions
- Drop Tables Commands
- What is the Recycle Bin?
- Object Naming in Recycle Bin
- Enable & Disable the Recycle Bin
- Purging is the Recycle Bin
- Data Dictionary Views for Recycle Bin

Drop Tables

- To drop a table that you no longer need, use the DROP TABLE statement. The table must be contained in your schema or you must have the DROP ANY TABLE system privilege.

Drop Tables Precautions

- Dropping a table removes the table definition from the data dictionary. All rows of the table are no longer accessible.
- All indexes and triggers associated with a table are dropped.
- All views and PL/SQL program units dependent on a dropped table remain, yet become invalid (not usable).
- All synonyms for a dropped table remain, but return an error when used.
- All extents allocated for a table that is dropped are returned to the free space of the tablespace and can be used by any other object requiring new extents or new objects. All rows corresponding to a clustered table are deleted from the blocks of the cluster

Drop Tables Commands

DROP TABLE hr.int_admin_emp;

- If the table to be dropped contains any primary or unique keys referenced by foreign keys of other tables and you intend to drop the FOREIGN KEY constraints of the child tables, then include the CASCADE clause in the DROP TABLE statement

DROP TABLE hr.admin_emp CASCADE CONSTRAINTS;

If you should want to immediately release the space associated with the table at the time you issue the DROP TABLE statement, include the PURGE clause as shown in the following statement:

DROP TABLE hr.admin_emp PURGE;

What is the Recycle Bin?

- The recycle bin is actually a data dictionary table containing information about dropped objects.
- Dropped tables and any associated objects such as indexes, constraints, nested tables, and the likes are not removed and still occupy space.
- They continue to count against user space quotas, until specifically purged from the recycle bin.
- SELECT * FROM RECYCLEBIN;

```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
BIN$hllPRYaURGeQBtlLS4/DHg==$0 TABLE
BIN$pgI7NTBnRKSh+170M90xHg==$0 TABLE
BLOBS                          TABLE
BONUS                          TABLE
```

Object Naming in Recycle Bin

- The renaming convention is as follows: BIN$unique_id$version
- unique_id is a 26-character globally unique identifier for this object, which makes the recycle bin name unique across all databases
- Version is a version number assigned by the database

Enable & Disable the Recycle Bin

To disable the recycle bin:

1. Issue one of the following statements:

```
ALTER SESSION SET recyclebin = OFF;

ALTER SYSTEM SET recyclebin = OFF SCOPE = SPFILE;
```

2. If you used ALTER SYSTEM, restart the database.

To enable the recycle bin:

1. Issue one of the following statements:

```
ALTER SESSION SET recyclebin = ON;

ALTER SYSTEM SET recyclebin = ON SCOPE = SPFILE;
```

2. If you used ALTER SYSTEM, restart the database.

Purging is the Recycle Bin

- If you decide that you are never going to restore an item from the recycle bin, you can use the PURGE statement to remove the items and their associated objects from the recycle bin and release their storage space.

PURGE TABLE "BIN$jsleilx392mk2=293$0";

PURGE TABLESPACE example;

PURGE RECYCLEBIN;

Data Dictionary Views for Recycle Bin

```
SELECT object_name, original_name FROM dba_recyclebin
   WHERE owner = 'HR';

OBJECT_NAME                    ORIGINAL_NAME
------------------------------ --------------------------------
BIN$yrMKlZaLMhfgNAgAIMenRA==$0 EMPLOYEES
```

You can also view the contents of the recycle bin using the SQL*Plus command SHOW RECYCLEBIN.

```
SQL> show recyclebin

ORIGINAL NAME    RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
---------------- ------------------------------ ------------ -------------------
EMPLOYEES        BIN$yrMKlZaVMhfgNAgAIMenRA==$0 TABLE        2003-10-27:14:00:19
```

Lesson98: Indexed Organized Tables

Overview of Segments

- Drop Tables

Drop Tables

- To drop a table that you no longer need, use the DROP TABLE statement. The table must be contained in your schema or you must have the DROP ANY TABLE system privilege.

Lesson99: Data Dictionary Views for Tables

Integrity Constraints

- Three Types of Views
- Displaying Views for Tables
- Size of a Table
- Displaying Table's Constraints
- Displaying Table's Columns Details

Three Types of Views

- There are three types of Views to manage Tables related information:
  DBA_
  ALL_
  USER_

Displaying Views for Tables

- SQL statement to display Tables related Views:

Select Table_name

From Dict

Where Table_name like '%TAB%';

You can use following choices for more views:

Table_name like '%COL%'

Table_name like '%EXTERNAL%'

Table_name like '%CONSTRAINT%'

Size of a Table

- To find size of table in MB

```
select sum(bytes)/1024/1024 SizeMB

from dba_segments

where segment_name ='EMP';
```

Displaying Table's Constraints

```
select u.constraint_name, u.CONSTRAINT_TYPE, substr(c.column_name,1,30) col_name
from user_constraints u, USER_CONS_COLUMNS c
where u.table_name='EMP'
and u.CONSTRAINT_TYPE = 'P'
and (u.table_name=c.table_name
and c.constraint_name=u.constraint_name);


CONSTRAINT_NAME                C COL_NAME
------------------------------ - ------------------------------
PK_EMP                         P EMPNO
```

Displaying Table's Columns Details

Column information, such as name, data type, length, precision, scale, and default data values can be listed using one of the views ending with the _COLUMNS suffix. For example, the following query lists all of the default column values for the emp and dept tables:

SELECT TABLE_NAME, COLUMN_NAME, DATA_TYPE, DATA_LENGTH, LAST_ANALYZED

FROM DBA_TAB_COLUMNS

WHERE OWNER = 'HR'

ORDER BY TABLE_NAME;

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DATA_LENGTH	LAST_ANALYZED
COUNTRIES	COUNTRY_ID	CHAR	2	05-FEB-03
COUNTRIES	COUNTRY_NAME	VARCHAR2	40	05-FEB-03
COUNTRIES	REGION_ID	NUMBER	22	05-FEB-03
DEPARTMENTS	DEPARTMENT_ID	NUMBER	22	05-FEB-03

Lesson 100: Creating & Altering an Index

Creating & Altering Index

- Altering Indexes
- Creating Indexes
- Creating & Alter Indexes
- Rebuild Index
- Making Index Visible/ Invisible
- Making Index Useable

Altering Indexes

To alter an index, your schema must contain the index or you must have the ALTER ANY INDEX system privilege. With the ALTER INDEX statement, you can:

- Rebuild or coalesce an existing index
- Deallocate unused space or allocate a new extent Specify parallel execution (or not) and alter the degree of parallelism
- Alter storage parameters or physical attributes Specify LOGGING or NOLOGGING

Creating Indexes

- Enable or disable key compression
- Mark the index unusable
- Make the index invisible
- Rename the index
- Start or stop the monitoring of index usage

CREATE INDEX emp_ename ON emp(ename)

TABLESPACE users

STORAGE (INITIAL 20K NEXT 20k);

CREATE UNIQUE INDEX dept_unique_index ON dept (dname) TABLESPACE indx;

Creating & Alter Indexes

```
        CREATE TABLE emp (
empno NUMBER(5) PRIMARY KEY,
age     INTEGER)
ENABLE PRIMARY KEY
USING INDEX TABLESPACE Work_Indx;
ALTER INDEX emp_ename
STORAGE (NEXT 40);
```

Rebuild Index

You have the option of rebuilding the index online. Rebuilding online enables you to update base tables at the same time that you are rebuilding.
The following statement rebuilds the emp_name index online:
ALTER INDEX emp_name REBUILD ONLINE;

Making Index Visible/ Invisible

An invisible index is ignored by the optimizer unless you explicitly set the OPTIMIZER_USE_INVISIBLE_INDEXES initialization parameter to TRUE at the session or system level. Making an index invisible is an alternative to making it unusable or dropping it.

ALTER INDEX *index* INVISIBLE;

ALTER INDEX *index* VISIBLE;

Making Index Useable

Query the data dictionary to determine whether an existing index or index partition is usable or unusable.

```
SELECT INDEX_NAME AS "INDEX OR PART NAME", STATUS, SEGMENT_CREATED

FROM USER_INDEXES

UNION ALL

SELECT PARTITION_NAME AS "INDEX OR PART NAME", STATUS, SEGMENT_CREATED

FROM USER_IND_PARTITIONS;
```

```
INDEX OR PART NAME              STATUS   SEG
------------------------------ -------- ---
I_EMP_ENAME                    N/A      N/A
JHIST_EMP_ID_ST_DATE_PK        VALID    YES
JHIST_JOB_IX                   VALID    YES
JHIST_EMPLOYEE_IX              VALID    YES
JHIST_DEPARTMENT_IX            VALID    YES
EMP_EMAIL_UK                   VALID    NO
.
```

Lesson 101: Recycle Bin

Schema Recycle Bin

- What is the Recycle Bin?
- What is the Recycle Bin?
- Object Naming the Recycle Bin
- Recycle Bin Tables
- Enable or Disable Recycle Bin
- Purging Objects in the Recycle Bin
- Restoring Objects From Recycle Bin
- Enable or Disable Recycle Bin

What is the Recycle Bin?

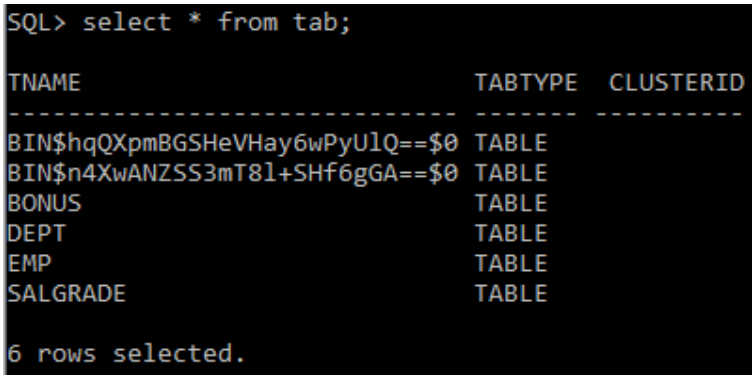
- The recycle bin is actually a data dictionary table containing information about dropped objects.
- Dropped tables and any associated objects such as indexes, constraints, nested tables, and the likes are not removed and still occupy space.
- They continue to count against user space quotas, until specifically purged from the recycle bin or the unlikely situation where they must be purged by the database because of tablespace space constraints.
- Each user can be thought of as having his own recycle bin, because, unless a user has the SYSDBA privilege.

SELECT * FROM RECYCLEBIN;

Object Naming the Recycle Bin
When a dropped table is moved to the recycle bin, the table and its associated objects are given system-generated names. This is necessary to avoid name conflicts that may arise if multiple tables have the same name. circumstances are:

- A user drops a table, re-creates it with the same name, then drops it again.
- Two users have tables with the same name, and both users drop their tables.
  BIN$*unique_id*$*version*

Recycle Bin Tables



```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
BIN$hqQXpmBGSHeVHay6wPyUlQ==$0 TABLE
BIN$n4XwANZSS3mT8l+SHf6gGA==$0 TABLE
BONUS                          TABLE
DEPT                           TABLE
EMP                            TABLE
SALGRADE                       TABLE

6 rows selected.
```

Enable or Disable Recycle Bin

Ref. DBA E25494 Oracle11g.pdf pp. 20-58

To disable the recycle bin:

1. Issue one of the following statements:

```
ALTER SESSION SET recyclebin = OFF;

ALTER SYSTEM SET recyclebin = OFF SCOPE = SPFILE;
```

2. If you used ALTER SYSTEM, restart the database.

To enable the recycle bin:

1. Issue one of the following statements:

```
ALTER SESSION SET recyclebin = ON;

ALTER SYSTEM SET recyclebin = ON SCOPE = SPFILE;
```

2. If you used ALTER SYSTEM, restart the database.

Enable or Disable Recycle Bin

Ref. DBA E25494 Oracle11g.pdf pp. 20-58

Purging Objects in the Recycle Bin

If you decide that you are never going to restore an item from the recycle bin, you can use the PURGE statement to remove the items and their associated objects from the recycle bin and release their storage space. You need the same privileges as if you were dropping the item.
PURGE TABLE "BIN$jsleilx392mk2=293$0";
PURGE TABLESPACE example;
PURGE TABLESPACE example USER oe;

Restoring Objects From Recycle Bin

The following example restores int_admin_emp table and assigns to it a new name:

FLASHBACK TABLE int_admin_emp TO BEFORE DROP
RENAME TO

int2_admin_emp;

Enable or Disable Recycle Bin
Ref. DBA E25494 Oracle11g.pdf pp. 20-59

```
SELECT object_name, original_name FROM dba_recyclebin
   WHERE owner = 'HR';

OBJECT_NAME                    ORIGINAL_NAME
------------------------------ --------------------------------
BIN$yrMKlZaLMhfgNAgAIMenRA==$0 EMPLOYEES
```

You can also view the contents of the recycle bin using the SQL*Plus command SHOW RECYCLEBIN.

```
SQL> show recyclebin

ORIGINAL NAME    RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
---------------- ------------------------------ ------------ -------------------
EMPLOYEES        BIN$yrMKlZaVMhfgNAgAIMenRA==$0 TABLE        2003-10-27:14:00:19
```

Enable or Disable Recycle Bin

Ref. DBA E25494 Oracle11g.pdf pp. 20-59

Lesson 102: Introduction to Database Administration

Dropping Index

- Dropping Index
- Reasons for Dropping Index
- Extents for Index
- SQL for Dropping Index
- When not to Drop Index
- To drop an index, the index must be contained in your schema, or you must have the DROP ANY INDEX system privilege.

Reasons for Dropping Index

Some reasons for dropping an index include:

- The index is no longer required.
- The index is not providing anticipated performance improvements for queries issued against the associated table. For example, the table might be very small, or there might be many rows in the table but very few index entries.

Reasons for Dropping Index

- Applications do not use the index to query the data.
- The index has become invalid and must be dropped before being rebuilt.
  The index has become too fragmented and must be dropped before being rebuilt.

Extents for Index

- When you drop an index, all extents of the index segment are returned to the containing tablespace and become available for other objects in the tablespace.

SQL for Dropping Index

- The following statement drops the emp_ename index:
  DROP INDEX emp_ename;

When not to Drop Index

You cannot drop only the index associated with an enabled UNIQUE key or PRIMARY KEY constraint. To drop a constraints associated index, you must disable or drop the constraint itself.

Lesson 103: Data Dictionary Views for Index

Data Dictionary Views

- Three Types of Views
- Displaying Views for Indexes
- Display Index Name
- Displaying Index on Table's Column
- Displaying Index's Columns Details

Three Types of Views

- There are three types of Views to manage Index related information:

DBA_

ALL_

USER_

Displaying Views for Indexes

- SQL statement to display Indexes related Views:

Select Table_name

From Dict

Where Table_name like '%IND%';

You can use following views as well:

INDEX_STATS

INDEX_HISTOGRAM

V$OBJECT_USAGE

Display Index Name

- To find table which contains Index

```
SQL> set linesize100
SQL> select index_name, index_type, Table_name
  2  from user_indexes
  3  where Table_name = 'EMP';

INDEX_NAME                     INDEX_TYPE                  TABLE_NAME
------------------------------ --------------------------- -----------
PK_EMP                         NORMAL                      EMP
```

Displaying Index on Table's Column

```
SQL> select i.index_name, i.index_type, substr(c.column_name, 1, 20) col_name
  2  from user_indexes i, user_ind_columns c
  3  where i.Table_name = 'EMP'
  4  and i.table_name=c.table_name;

INDEX_NAME                     INDEX_TYPE                  COL_NAME
------------------------------ --------------------------- --------------------
PK_EMP                         NORMAL                      EMPNO
```

Displaying Index's Columns Details

create index ind_name on EMP(ename);

select i.index_name, i.index_type, substr(c.column_name, 1, 20) col_name

from user_indexes i, user_ind_columns c

where i.Table_name = 'EMP'

and i.table_name=c.table_name

and i.index_name=c.index_name;

```
INDEX_NAME                     INDEX_TYPE                     COL_NAME
------------------------------ ------------------------------ ---------
IND_NAME                       NORMAL                         ENAME
PK_EMP                         NORMAL                         EMPNO
```

Lesson 104: Defining Clusters

Clusters

- What is a Cluster?
- Cluster Key
- Choosing Appropriate Tables for Cluster
- Appropriate Column for Cluster

What is a Cluster?

- A cluster provides an optional method of storing table data. A cluster is made up of a group of tables that share the same data blocks. The tables are grouped together because they share common columns and are often used together.
- For example, the
- emp and dept table share the deptno column. When you cluster the emp and dept tables, Oracle stores all rows for each department from both the emp and dept tables in the same data blocks.
- A cluster provides an optional method of storing table data. A cluster is made up of a group of tables that share the same data blocks. The tables are grouped together because they share common columns and are often used together.
- For example, the
- emp and dept table share the deptno column. When you cluster the emp and dept tables, Oracle stores all rows for each department from both the emp and dept tables in the same data blocks.

Cluster Key

- The **cluster key** is the column, or group of columns, that the clustered tables have in common. You specify the columns of the cluster key when creating the cluster.
- You subsequently specify the same columns when creating every table added to the cluster. Each cluster key value is stored only once each in the cluster and the cluster index, no matter how many rows of different tables contain the value.
- After creating a cluster, you can create tables in the cluster. However, before any rows can be inserted into the clustered tables, a cluster index must be created.
- Using clusters does not affect the creation of additional indexes on the clustered tables; they can be created and dropped as usual.

Choosing Appropriate Tables for Cluster

- Use clusters for tables for which the following conditions are true:
- The tables are primarily queried--that is, tables that are *not* predominantly inserted into or updated.
- Records from the tables are frequently queried together or joined.

Appropriate Column for Cluster

- Choose cluster key columns carefully. If multiple columns are used in queries that join the tables, make the cluster key a composite key.
- In general, the characteristics that indicate a good cluster index are the same as those for any index.

Lesson 105: Create & Alter Clusters

- Creating a Cluster
- SQL DDL Command for Cluster
- Creating a Table in a Cluster
- Creating Cluster using SQL
- Altering Clusters
- Dropping Clusters

Creating a Cluster

To create a cluster in your schema, you must have the CREATE CLUSTER system privilege and a quota for the tablespace intended to contain the cluster or the UNLIMITED TABLESPACE system privilege.
To create a cluster in another user's schema you must have the CREATE ANY CLUSTER system privilege, and the owner must have a quota for the tablespace intended to contain the cluster or the UNLIMITED TABLESPACE system privilege.

SQL DDL Command for Cluster

```
CREATE CLUSTER emp_dept

(deptno NUMBER(3))
SIZE 600
TABLESPACE users
STORAGE (INITIAL 200K NEXT 300K
MINEXTENTS 2 PCTINCREASE 33);
```

Creating a Table in a Cluster

To create a table in a cluster, you must have either the CREATE TABLE or CREATE ANY TABLE system privilege. You do not need a tablespace quota or the UNLIMITED TABLESPACE system privilege to create a table in a cluster.

Creating Cluster using SQL

```
CREATE TABLE emp (
empno NUMBER(5) PRIMARY KEY,
ename VARCHAR2(15) NOT NULL,
. . .
deptno NUMBER(3) REFERENCES dept)
CLUSTER emp_dept (deptno);
CREATE TABLE dept (
deptno NUMBER(3) PRIMARY KEY, . . . )
CLUSTER emp_dept (deptno);
```

Altering Clusters

- To alter a cluster, your schema must contain the cluster or you must have the ALTER ANY CLUSTER system privilege. You can alter an existing cluster to change the following settings:
- Physical attributes (INITRANS and storage characteristics)
- The average amount of space required to store all the rows for a cluster key value (SIZE)
- The default degree of parallelism
- To alter a cluster, your schema must contain the cluster or you must have the ALTER ANY CLUSTER system privilege. You can alter an existing cluster to change the following settings:
- Physical attributes (INITRANS and storage characteristics)

- The average amount of space required to store all the rows for a cluster key value (SIZE)
- The default degree of parallelism

Dropping Clusters

- To alter a cluster, your schema must contain the cluster or you must have the ALTER ANY CLUSTER system privilege. You can alter an existing cluster to change the following settings:
- Physical attributes (INITRANS and storage characteristics)
- The average amount of space required to store all the rows for a cluster key value (SIZE)
- The default degree of parallelism
- DROP CLUSTER emp_dept INCLUDING TABLES;
- DROP CLUSTER emp_dept INCLUDING TABLES CASCADE CONSTRAINTS;

Lesson 106: Data Dictionary Views for Clusters

Data Dictionary Views

- Three Types of Views
- Displaying Views for Clusters
- Display Cluster's Details
- Displaying Cluster on Table's Column

Three Types of Views

- There are three types of Views to manage Cluster related information:

  DBA_

  ALL_

  USER_

Displaying Views for Clusters

- SQL statement to display Indexes related Views:

  Select Table_name

  From Dict

  Where Table_name like '%CLU%';

  You can use following views as well:

  DBA_CLU_COLUMNS

  USER_CLU_COLUMNS

Display Cluster's Details

- To find cluster's information

```
SQL> conn sys / as sysdba
Enter password:
Connected.
SQL> select cluster_name, Tablespace_name, cache
  2  from user_clusters;

CLUSTER_NAME                   TABLESPACE_NAME                CACHE
------------------------------ ------------------------------ -----
C_USER#                        SYSTEM                             N
C_TS#                          SYSTEM                             N
C_TOID_VERSION#                SYSTEM                             N
C_RG#                          SYSTEM                             N
C_OBJ#_INTCOL#                 SYSTEM                             N
C_OBJ#                         SYSTEM                             N
C_MLOG#                        SYSTEM                             N
C_FILE#_BLOCK#                 SYSTEM                             N
C_COBJ#                        SYSTEM                             N
SMON_SCN_TO_TIME_AUX           SYSAUX                             N

10 rows selected.
```

Displaying Cluster on Table's Column

```
SQL> select c.cluster_name, c.Tablespace_name, c.cluster_name, cc.clu_column_name
  2  from user_clusters c, user_clu_columns cc
  3  where c.cluster_name=cc.cluster_name;

CLUSTER_NAME                   TABLESPACE_NAME                CLUSTER_NAME                   CLU_COLUMN_NAME
------------------------------ ------------------------------ ------------------------------ ---------------
---
C_COBJ#                        SYSTEM                         C_COBJ#                        OBJ#
C_COBJ#                        SYSTEM                         C_COBJ#                        OBJ#
C_FILE#_BLOCK#                 SYSTEM                         C_FILE#_BLOCK#                 SEGFILE#
C_FILE#_BLOCK#                 SYSTEM                         C_FILE#_BLOCK#                 SEGBLOCK#
C_FILE#_BLOCK#                 SYSTEM                         C_FILE#_BLOCK#                 TS#
C_FILE#_BLOCK#                 SYSTEM                         C_FILE#_BLOCK#                 SEGFILE#
C_FILE#_BLOCK#                 SYSTEM                         C_FILE#_BLOCK#                 SEGBLOCK#
C_FILE#_BLOCK#                 SYSTEM                         C_FILE#_BLOCK#                 TS#
```

Lesson 107: Defining Hash Clusters

Hash Clusters

- What is a Hash Cluster?
- Finding Rows in Index or Cluster?
- Situation where useful Hashing
- Creating Hash Clusters
- Querying Hash Clusters

What is a Hash Cluster?

- Storing a table in a hash cluster is an optional way to improve the performance of data retrieval.
- A hash cluster provides an alternative to a non-clustered table with an index or an index cluster.
- With an indexed table or index cluster, Oracle Database locates the rows in a table using key values that the database stores in a separate index.
- To use hashing, you create a hash cluster and load tables into it. The database physically stores the rows of a table in a hash cluster and retrieves them according to the results of a **hash function**.
- Oracle Database uses a hash function to generate a distribution of numeric values, called **hash values**, that are based on specific cluster key values.
- The key of a hash cluster, like the key of an index cluster, can be a single column or composite key (multiple column key).
- To find or store a row in a hash cluster, the database applies the hash function to the cluster key value of the row.

- The resulting hash value corresponds
  to a data block in the cluster, which the database then reads or writes on behalf of the issued statement.

Finding Rows in Index or Cluster?

- One or more I/Os to find or store the key value in the index
- Another I/O to read or write the row in the table or cluster

Situation where useful Hashing

- Most queries are equality queries on the cluster key:
  SELECT ... WHERE cluster_key = ...;
  In such cases, the cluster key in the equality condition is hashed, and the corresponding hash key is usually found with a single read. In comparison, for an indexed table the key value must first be found in the index, and then the row is read from the table (another read).

- The tables in the hash cluster are primarily static in size so that you can determine the number of rows and amount of space required for the tables in the cluster. If tables in a hash cluster require more space than the initial allocation for the cluster, performance degradation can be substantial because overflow blocks are required.

Creating Hash Clusters

- CREATE CLUSTER trial_cluster (trialno NUMBER(5,0))
- TABLESPACE users
- STORAGE (INITIAL 250K
- NEXT 50K
- MINEXTENTS 1
- MAXEXTENTS 3
- PCTINCREASE 0)
- HASH IS trialno
- HASHKEYS 150;

The following statement creates the trial table in the trial_cluster hash cluster:

```
CREATE TABLE trial (
trialno NUMBER(5,0) PRIMARY KEY,
...
)
CLUSTER trial_cluster (trialno);
```

Querying Hash Clusters

```
TELEPHONE_NUMBER CALL_TIMESTAMP CALL_DURATION OTHER_INFO
---------------- -------------- ------------- -------------------------
      6505551212              0             9 misc info
      6505551212              1            17 misc info
      6505551212              2             5 misc info
      6505551212              3            90 misc info

Execution Plan
----------------------------------------------------------
Plan hash value: 2118876266

-------------------------------------------------------------------------
| Id  | Operation         | Name        | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |             |     1 |    56 |     0   (0)|
|*  1 |  TABLE ACCESS HASH| CALL_DETAIL |     1 |    56 |            |
-------------------------------------------------------------------------
```

Lesson108: Creating & Altering Hash Clusters

- Creating Hash Clusters
- Querying Hash Clusters
- Estimating Hash Clusters Size
- Altering Hash Clusters
- Dropping Hash Clusters

Creating Hash Clusters

- CREATE CLUSTER trial_cluster (trialno NUMBER(5,0))
- TABLESPACE users
- STORAGE (INITIAL 250K
- NEXT 50K
- MINEXTENTS 1
- MAXEXTENTS 3
- PCTINCREASE 0)
- HASH IS trialno
- HASHKEYS 150;

The following statement creates the trial table in the trial_cluster hash cluster:

CREATE TABLE trial (

trialno NUMBER(5,0) PRIMARY KEY,

...

)

CLUSTER trial_cluster (trialno);

Querying Hash Clusters

```
TELEPHONE_NUMBER CALL_TIMESTAMP CALL_DURATION OTHER_INFO
---------------- -------------- ------------- --------------------------
      6505551212              0             9 misc info
      6505551212              1            17 misc info
      6505551212              2             5 misc info
      6505551212              3            90 misc info

Execution Plan
----------------------------------------------------------
Plan hash value: 2118876266

-------------------------------------------------------------------------
| Id  | Operation         | Name        | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |             |     1 |    56 |     0   (0)|
|*  1 |  TABLE ACCESS HASH| CALL_DETAIL |     1 |    56 |            |
-------------------------------------------------------------------------
```

Estimating Hash Clusters Size

Example:

```
STORAGE (INITIAL 100K
    NEXT 150K
    MINEXTENTS 1
    PCTINCREASE 0)
SIZE 1500
HASHKEYS 100
```

In this example, only one hash key can be assigned for each data block. Therefore, the initial space required for the hash cluster is at least 100*2K or 200K.

Altering Hash Clusters

You can alter a hash cluster with the ALTER CLUSTER statement:
ALTER CLUSTER emp_dept . . . ;
The implications for altering a hash cluster are identical to those for altering an index cluster, described in. However, the SIZE, HASHKEYS, "Altering Clusters" on page 22-6 and HASH IS parameters cannot be specified in an ALTER CLUSTER statement. To change these parameters, you must re-create the cluster, then copy the data from the original cluster.

Dropping Hash Clusters

You can drop a hash cluster using the DROP CLUSTER statement:
DROP CLUSTER emp_dept;
A table in a hash cluster is dropped using the DROP TABLE statement. The implications of dropping hash clusters and tables in hash clusters are the same as those for dropping index clusters.

Lesson109: Data Dictionary Views for Hash Clusters

Data Dictionary Views

- Three Types of Views
- Displaying Views for Hash Clusters
- Display Hash or just Cluster’s Details
- Displaying Cluster on Table’s Column

Three Types of Views

- There are three types of Views to manage Hash Cluster related information:

DBA_

ALL_

USER_

Displaying Views for Hash Clusters

- SQL statement to display Indexes related Views:

```
Select Table_name

From Dict

Where Table_name like '%SYN%';
```

You can use following views as well:

DBA_UPDATABLE_COLUMNS

ALL_UPDATABLE_COLUMNS

USER_UPDATABLE_COLUMNS

Display Hash or just Cluster's Details

- To find cluster's information

```
SQL> conn sys / as sysdba
Enter password:
Connected.
SQL> select cluster_name, Tablespace_name, cache
  2  from user_clusters;

CLUSTER_NAME                   TABLESPACE_NAME                CACHE
------------------------------ ------------------------------ -----
C_USER#                        SYSTEM                             N
C_TS#                          SYSTEM                             N
C_TOID_VERSION#                SYSTEM                             N
C_RG#                          SYSTEM                             N
C_OBJ#_INTCOL#                 SYSTEM                             N
C_OBJ#                         SYSTEM                             N
C_MLOG#                        SYSTEM                             N
C_FILE#_BLOCK#                 SYSTEM                             N
C_COBJ#                        SYSTEM                             N
SMON_SCN_TO_TIME_AUX           SYSAUX                             N

10 rows selected.
```

Displaying Cluster on Table's Column

```
SQL> select c.cluster_name, c.Tablespace_name, c.cluster_name, cc.clu_column_name
  2  from user_clusters c, user_clu_columns cc
  3  where c.cluster_name=cc.cluster_name;

CLUSTER_NAME                   TABLESPACE_NAME                CLUSTER_NAME                   CLU_COLUMN_NAME
------------------------------ ------------------------------ ------------------------------ ---------------
C_COBJ#                        SYSTEM                         C_COBJ#                        OBJ#
C_COBJ#                        SYSTEM                         C_COBJ#                        OBJ#
C_FILE#_BLOCK#                 SYSTEM                         C_FILE#_BLOCK#                 SEGFILE#
C_FILE#_BLOCK#                 SYSTEM                         C_FILE#_BLOCK#                 SEGBLOCK#
C_FILE#_BLOCK#                 SYSTEM                         C_FILE#_BLOCK#                 TS#
C_FILE#_BLOCK#                 SYSTEM                         C_FILE#_BLOCK#                 SEGFILE#
C_FILE#_BLOCK#                 SYSTEM                         C_FILE#_BLOCK#                 SEGBLOCK#
C_FILE#_BLOCK#                 SYSTEM                         C_FILE#_BLOCK#                 TS#
```

Lesson110: Defining & Managing Views

- How a Schema is Created?
- How a Schema is Implemented?
- Simple & Complex Views
- SQL Practices on Views

How a Schema is Created?

- We have studied that DBA has to create default Tablespace and assign temporary Tablespace with storage in terms of data files.
- Create first user (owner of schema) with grants such as rights and privileges.
- We will cover schema objects Tables and Views.
- Ref. DBA-73

How a Schema is Implemented?

- A view is a logical representation of a table or combination of tables.
- In essence, a view is a stored query.
- A view derives its data from the tables on which it is based.
- These tables are called base tables. Base tables might in turn be actual tables or might be views themselves.
- All operations performed on a view actually affect the base table of the view.
- You can use views in almost the same way as tables. You can query, update, insert into, and delete from views, just as you can standard tables.
- Views can provide a different representation (such as subsets or supersets) of the data that resides within other tables and views.
- Views can be Simple or Complex
- Views can be used for security purposes
- View does not contain its own data

Simple & Complex Views

- CREATE VIEW sales_staff AS
- SELECT empno, ename, deptno
- FROM emp
- WHERE deptno = 10;
- CREATE VIEW sales_staff AS
- SELECT job, count(*)
- FROM emp
- WHERE deptno = 10
- Group by job;

SQL Practices on Views

- We can apply Insert, delete and update operations on simple views. for base tables

Lesson111: Updating a Join View

- Introduction
- Rules for Updating Join Views
- Example-1 of a Join View
- Example-2 of a Join View
- Data Dictionary View for Join Views

Introduction

- An updatable join view (also referred to as a **modifiable join view**) is a view that contains multiple tables in the top-level FROM clause of the SELECT statement, and is not restricted by the WITH READ ONLY clause.

Rules for Updating Join Views

- The rules for updatable join views are shown in the following table. Views that meet these criteria are said to be inherently updatable.

Rule	Description
General Rule	Any `INSERT`, `UPDATE`, or `DELETE` operation on a join view can modify only one underlying base table at a time.
`UPDATE` Rule	All updatable columns of a join view must map to columns of a **key-preserved table**. See "Key-Preserved Tables" on page 24-7 for a discussion of key-preserved tables. If the view is defined with the `WITH CHECK OPTION` clause, then all join columns and all columns of repeated tables are not updatable.
`DELETE` Rule	Rows from a join view can be deleted as long as there is exactly one key-preserved table in the join. The key preserved table can be repeated in the `FROM` clause. If the view is defined with the `WITH CHECK OPTION` clause and the key preserved table is repeated, then the rows cannot be deleted from the view.
`INSERT` Rule	An `INSERT` statement must not explicitly or implicitly refer to the columns of a **non-key-preserved table**. If the join view is defined with the `WITH CHECK OPTION` clause, `INSERT` statements are not permitted.

Example-1 of a Join View

```
CREATE VIEW emp_dept AS
SELECT emp.empno, emp.ename, emp.deptno, emp.sal, dept.dname, dept.loc
FROM emp, dept
WHERE emp.deptno = dept.deptno
AND

dept.loc IN ('DALLAS', 'NEW YORK', 'BOSTON');
```

```
EMPNO      ENAME      DEPTNO  DNAME          LOC
---------- ---------- ------- -------------- -----------
      7782 CLARK           10 ACCOUNTING     NEW YORK
      7839 KING            10 ACCOUNTING     NEW YORK
      7934 MILLER          10 ACCOUNTING     NEW YORK
      7369 SMITH           20 RESEARCH       DALLAS
      7876 ADAMS           20 RESEARCH       DALLAS
```

```
CREATE VIEW emp_dept AS
SELECT emp.empno, emp.ename, emp.deptno, emp.sal, dept.dname, dept.loc
```

FROM emp, dept
WHERE emp.deptno = dept.deptno
AND

dept.loc IN ('DALLAS', 'NEW YORK', 'BOSTON');

```
UPDATE emp_dept
     SET loc = 'BOSTON'
     WHERE ename = 'SMITH';
ORA-01779 cannot modify a column which mapsto a non key-preserved table
```

Example-2 of a Join View

CREATE VIEW emp_mgr AS
SELECT e1.ename, e2.ename mname
FROM emp e1, emp e2
WHERE e1.mgr = e2.empno
WITH CHECK OPTION;

Following statement works successfully:

INSERT INTO emp_dept (ename, empno, deptno)
VALUES ('KURODA', 9010, 40);

The following INSERT statements would fail with an error (ORA-01776 cannot modify more than one base table through a join view):

INSERT INTO emp_dept (ename, empno, deptno)

VALUES ('KURODA', 9010, 77); --FK value not found in department table

INSERT INTO emp_dept (empno, ename, loc)

VALUES (9010, 'KURODA', 'BOSTON'); --cannot insert data in two tables

Data Dictionary View for Join Views

If you are in doubt whether a view is modifiable, then you can select from the USER_UPDATABLE_COLUMNS view to see if it is. For example:

Lesson112: Create Sequences

- An Overview
- Sequence Characteristics
- Sequence Example-1
- Concurrent Access to Sequence
- Sequence Syntax
- Sequence Example-2
- Data Dictionary Views for Sequence

An Overview

A **sequence (Autonumber)** is a schema object from which multiple users can generate unique integers. A sequence generator provides a highly scalable and well-performing method to generate surrogate keys for a number data type.

Sequence Characteristics

- A sequence definition indicates general information, such as the following:
- The name of the sequence
- Whether the sequence ascends or descends The interval between numbers

- Whether the database should cache sets of generated sequence numbers in memory
- Whether the sequence should cycle when a limit is reached

Sequence Example-1

```
CREATE SEQUENCE customers_seq
START WITH 1000
INCREMENT BY 1
NOCACHE
NOCYCLE;
```

Concurrent Access to Sequence

The same sequence generator can generate numbers for multiple tables. In this way, the database can generate primary keys automatically and coordinate keys across multiple rows or tables.

For example, a sequence can generate primary keys for an orders table and a customers table.

Sequence Syntax

CREATE SEQUENCE user.sequence_name [INCREMENT BY interval]

[START WITH first_number]

[MAXVALUE max_value | NOMAXVALUE]

[MINVALUE min_value | NOMINVALUE]

[CYCLE | NOCYCLE]

[CACHE cache_size | NOCACHE]

[ORDER | NOORDER];

Sequence Example-2

CREATE SEQUENCE id_seq

INCREMENT BY 10

START WITH 10

MINVALUE 10

MAXVALUE 100

CYCLE

CACHE 2;

To get the next value of the sequence, you use the NEXTVAL pseudo-column:

SELECT id_seq.NEXTVAL

FROM dual;

INSERT INTO tasks(id, title) VALUES(seq.NEXTVAL, 'Create Sequence in Oracle');

SELECT id_seq.NEXTVAL FROM dual;

Data Dictionary Views for Sequence

USER_SEQUENCES

ALL_SEQUENCES

SEQ

Lesson113: Alter & Drop Sequences

- An Overview
- NEXTVAL & CURRVAL
- Caching Sequence Number
- Dropping Sequences

An Overview

- To alter a sequence, your schema must contain the sequence, you must have the ALTER object privilege on the sequence, or you must have the ALTER ANY SEQUENCE system privilege.
- You can alter a sequence to change any of the parameters that define how it generates sequence numbers except the sequence starting number. To change the starting point of a sequence, drop the sequence and then re-create it.
- Alter a sequence using the ALTER SEQUENCE statement. For example, the following statement alters the emp_seq:
  ALTER SEQUENCE emp_seq
  INCREMENT BY 10
  MAXVALUE 10000
  CYCLE
  CACHE 20;

NEXTVAL & CURRVAL

CURRVAL and NEXTVAL cannot be used in the following places:

- A subquery
- A view query or materialized view query
- A SELECT statement with the DISTINCT operator
- A SELECT statement with a GROUP BY or ORDER BY clause
- A SELECT statement that is combined with another SELECT statement with the UNION, INTERSECT, or MINUS set operator
- The WHERE clause of a SELECT statement
- The condition of a CHECK constraint

Caching Sequence Number

Follow these guidelines for fast access to all sequence numbers:

- Be sure the sequence cache can hold all the sequences used concurrently by your applications.
- Increase the number of values for each sequence held in the sequence cache.

Dropping Sequences

You can drop any sequence in your schema. To drop a sequence in another schema, you must have the DROP ANY SEQUENCE system privilege. If a sequence is no longer required, you can drop the sequence using the DROP SEQUENCE statement.
For example, the following statement drops the order_seq sequence:
DROP SEQUENCE order_seq;
When a sequence is dropped, its definition is removed from the data dictionary. Any synonyms for the sequence remain, but return an error when referenced

Lesson114: Defining Synonyms

- An Overview
- Public & Private Synonyms
- Securable Synonyms
- DDL Create Synonyms
- Select script for all Tables

An Overview

- A synonym is an alias for a schema object.
- Synonyms can provide a level of security by masking the name and owner of an object and by providing location transparency for remote objects of a distributed database.
- Also, they are convenient to use and reduce the complexity of SQL statements for database users.
- Synonyms allow underlying objects to be renamed or moved, where only the synonym must be redefined and applications based on the synonym continue to function without modification.
- User u1 can access table of user scott as

  Select * from scott.emp;

  But we want u1 uses statement as

  Select * from emp;

  User scott should create emp synonym with the same name

Public & Private Synonyms

You can create both public and private synonyms.

- A **public** synonym is owned by the special user group named PUBLIC and is accessible to every user in a database.
- A **private** synonym is contained in the schema of a specific user and available only to the user and to grantees for the underlying object.

Securable Synonyms

- Synonyms themselves are not securable. When you grant object privileges on a synonym, you are really granting privileges on the underlying object, and the
  synonym is acting only as an alias for the object in the GRANT statement.
- To create a private synonym in your own schema, you must have the CREATE SYNONYM privilege.
- To create a private synonym in another user's schema, you must have the CREATE ANY SYNONYM privilege.
- To create a public synonym, you must have the CREATE PUBLIC SYNONYM system privilege.

DDL Create Synonyms

- Following sample DDL command is required to create public synonym:
- CREATE PUBLIC SYNONYM public_emp FOR ejaz.emp
- Creating synonym inside same user’s schema

- create synonym syn_emp for scott.emp;

Select Script for all Tables

- select 'create synonym '||tname||' for u1.'||tname||';'
- from tab;

Lesson115:Create & Drop Synonyms

- Specifying Schema
- FOR Clause
- Create Synonym for Objects
- Create Synonym Examples
- Accessing Synonym
- Drop Synonym
- Synonym Syntax

Specifying Schema

- Specify the schema to contain the synonym. If you omit schema, then Oracle Database creates the synonym in your own schema. You cannot specify a schema for the synonym if you have specified PUBLIC.

FOR Clause

Specify the object for which the synonym is created. The schema object for which you are creating the synonym can be of the following types:

- Table or object table
- View or object view
- Sequence
- Stored procedure, function, or package
- Materialized view
- Java class schema object
- User-defined object type & Synonym

Create Synonym for Objects

Specify the object for which the synonym is created. The schema object for which you are creating the synonym can be of the following types:

- Table or object table
- View or object view
- Sequence
- Stored procedure, function, or package
- Materialized view

- Java class schema object
- User-defined object type & Synonym

Create Synonym Examples

To define the synonym offices for the table locations in the schema hr, issue the following statement:

CREATE SYNONYM offices FOR hr.locations;

To create a PUBLIC synonym for the emp table in the schema hr on the remote database, you could issue the following statement:

CREATE PUBLIC SYNONYM emp_table

FOR hr.emp@remote.us.oracle.com;

Accessing Synonym

If the user hr's schema does not contain an object named customers, and if hr has select permission on oe.customers, then hr can access the customers table in oe's schema by using the public synonym customers:

SELECT COUNT(*) FROM customers;

Drop Synonym

Following commands can be used to drop synonyms

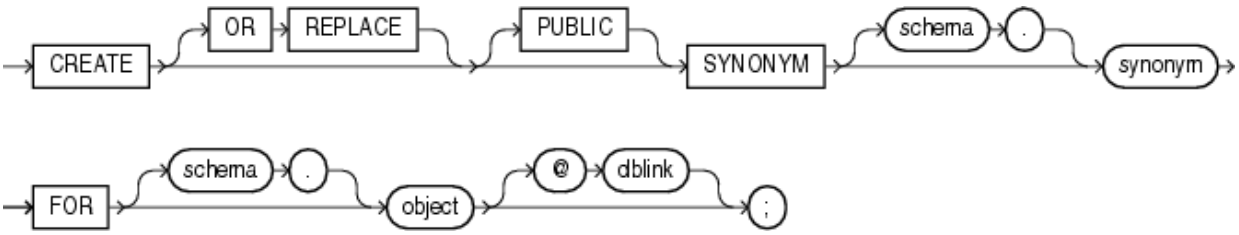
drop synonym dept;

drop synonym emp;

drop Public synonym customer;

Synonym Syntax

Following is a complete syntax of creating Synonym



Lesson116: Data Dictionary Views for Synonyms, Views & Sequences

Create & Drop Sequences

- Three Types of Views
- Displaying Views
- Data Dictionary Views for Synonym
- Data Dictionary Views for Sequence
- Data Dictionary Views for VIEWS

Three Types of Views

- There are three types of Views to manage Hash Cluster related information:

DBA_

ALL_

USER_

Displaying Views

- SQL statement to display Indexes related Views:

Select Table_name

From Dict

Where Table_name like '%SYN%';

Or use ('%VIEW%' or '%SEQ%‘)

You can use following views as well:

DBA_UPDATABLE_COLUMNS

ALL_UPDATABLE_COLUMNS

USER_UPDATABLE_COLUMNS

Data Dictionary Views for Synonym

Following views can be used to see synonyms

Select object_name, object_type

from user_objects

where object_type='SYNONYM';

Select * from TAB;

Data Dictionary Views for Synonym

Following view can be used to see synonyms

select synonym_name, table_owner, table_name

from user_synonyms;

```
SQL> select synonym_name, table_owner, table_name
  2  from user_synonyms;

SYNONYM_NAME                   TABLE_OWNER                    TABLE_NAME
------------------------------ ------------------------------ -----------
SYN_EMP                        SCOTT                          EMP
```

Data Dictionary Views for Sequence

Following sequence is created:

CREATE SEQUENCE customers_seq
START WITH 1000
INCREMENT BY 1
NOCACHE
NOCYCLE;

Data Dictionary Views for Sequence

Following view can be used to see synonyms

select object_name, object_type

from user_objects

where object_type='SEQUENCE';

Data Dictionary Views for Sequence

Following view can be used to see synonyms

select sequence_name, cycle_flag, cache_size, last_number

from user_sequences;

```
SEQUENCE_NAME                  C CACHE_SIZE LAST_NUMBER
------------------------------ - ---------- -----------
CUSTOMERS_SEQ                  N          0        1000
```

Data Dictionary Views for VIEWS

grant create view to scott;

create view v_emp

AS

select job, count(*) Count_EMP

from emp

group by job;

Data Dictionary Views for VIEWS

Following view can be used to see synonyms

```
SQL> select view_name, text
  2  from user_views;

VIEW_NAME
------------------------------
TEXT
--------------------------------------------------------------------------------
V_EMP
select job, count(*) Count_EMP
from emp
group by job


SQL>
```

Lesson117: Introduction to Database Server Programming using PL/SQL

Introduction to PL/SQL

- What is PL/SQL
- PL/SQL & Other Languages

What is PL/SQL

- Embedded structured programming
- Performed on Server Machine
- Centrally accessible from single database
- Executed on server without any network overhead

PL/SQL & Other Languages

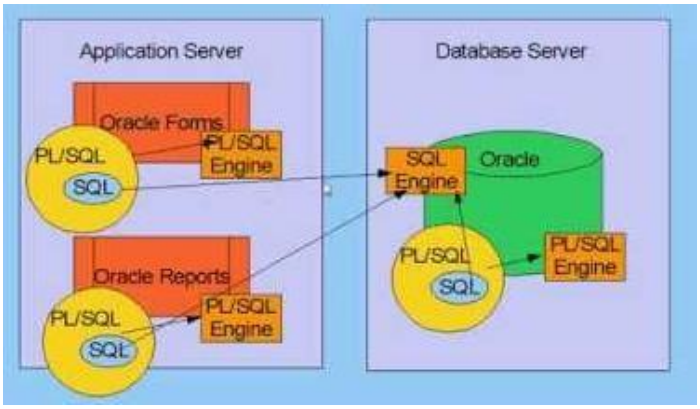
- Embedded SQL (PL/SQL, JAVA/ VB & DB)
- Database Server Level Programming(PL/SQL, Transact-SQL, IBM DB2-Cobol, ProC, ProCobol)
- (Front end programming can be changed without affecting Server programming)
- Database Client Programming (Developer 9i, JDeveloper 9i, Java (J2EE), VB, .Net)

Lesson118: PL/SQL Constructs

- SQL & PL/SQL
- PL/SQL Data Types
- PL/SQL Contents

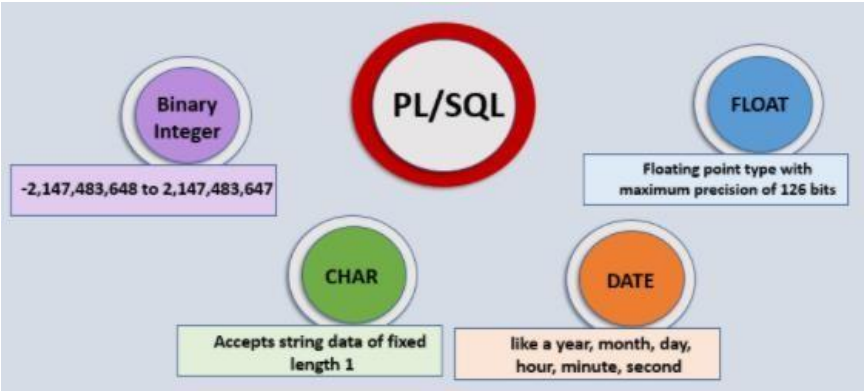
SQL & PL/SQL

SQL & PL/SQL on Application & Server Sides



PL/SQL Data Types

Following main data types are used in PL/SQL



PL/SQL Contents

- Benefits
- Basic Constructs
- Anonymous blocks
- Procedures
- Functions
- Packages
- Triggers
- Cursors
- Dynamic SQL

Lesson119: PL/SQL Benefits & Basic Constructs

- PL/SQL Benefits
- PL/SQL Basic Constructs

PL/SQL Benefits

More powerful than pure SQL because it combines the power of SQL and

- Iteration (loops)
- Selection (Ifs)
- Cursors
- Block Structures
- Stored Procedures
- etc.

PL/SQL Basic Constructs

Basic Structure

Running a program

Variables

SELECT INTO

Comments

IFs

LOOPs

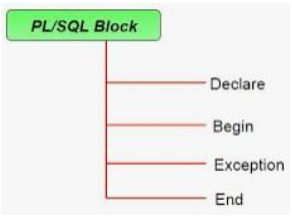
Output

Lesson120:Basic Program Block Structure and Display output

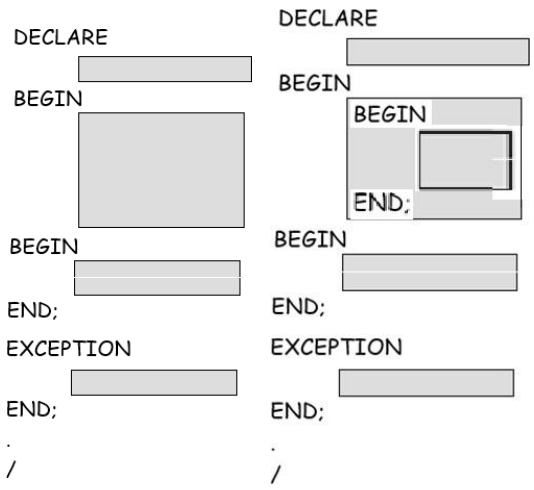
Basic Program Structure

- Nested Basic Structures
- First PL/SQL Program
- First PL/SQL Program with Declaration

Basic Structure



Nested Basic Structures



First PL/SQL Program

```
SET SERVEROUTPUT ON;
BEGIN
    DBMS_OUTPUT.PUT_LINE('This is my first program');
END;
/
```

Before executing code that contains DBMS_OUTPUT.PUT_LINE,

must run (in a current session) at SQL prompt: set serveroutput on.

System Package DBMS_OUTPUT exists in Oracle Dictionary. It accepts and return SINGLE argument.

First PL/SQL Program with Declaration

```
SET SERVEROUTPUT ON; --if not runing already
DECLARE
    msg varchar2(40) := 'I love Oracle Programing' ;
BEGIN
    dbms_output.put_line(msg);
END;
/
```

```
SQL> DECLARE
  2       msg varchar2(40) := 'I love Oracle Programing' ;
  3  BEGIN
  4       dbms_output.put_line(msg);
  5  END;
  6  /
I love Oracle Programing

PL/SQL procedure successfully completed.

SQL>
```

Lesson121: Basic Programs Block Structure with more examples

Basic Program Structure

- Program Execution with an error
- Program Fixing of an error
- Program Compilation

Program Execution with an Error

Following program declarations, assigned values and display result:

```
SQL> Declare
  2  -- declare variable x, y and z of datatype number
  3  x number(5);
  4  y number(5);
  5  z number(7);
  6  Begin
  7  -- Here we Assigning 10 into x
  8  x:=10;
  9  -- Assigning 20 into x
 10  y:=20;
 11  -- Assigning sum of x and y into z
 12  z:=x+y;
 13  --Print the Result
 14  dbms_output.put_line('Sum is '||z);
 15  End;
 16  /
Sum is 30
```

Program Fixing of an Error

Following program terminates:

```
SQL> Declare
  2  -- declare variable x, y and z of datatype number
  3  x varchar2(5);
  4  y number(5);
  5  z number(5);
  6  Begin
  7  -- Here we Assigning 10 into x
  8  x:=10;
  9  -- Assigning 20 into x
 10  y:=2V0;
 11  -- Assigning sum of x and y into z
 12  z:=x+y;
 13  --Print the Result
 14  dbms_output.put_line('Sum is '||z);
 15  End;
 16  /
y:=2V0;
    *
ERROR at line 10:
ORA-06550: line 10, column 5:
PLS-00103: Encountered the symbol "V0" when expecting one of the following:
* & = - + ; < / > at in is mod remainder not rem
<an exponent (**)> <> or != or ~= >= <= <> and or like like2
like4 likec between || multiset member submultiset
The symbol "*" was substituted for "V0" to continue.
```

Program Compilation

Following program (Anonymous Block) executes with error due to compile & execute together

Lesson122: Basic Programs Block Structure with more examples

Basic Program Structure

- Program execution with an error
- Program fixing of an error
- Program with SELECT INTO

Program execution with an error

Following program executes with an error during compilation/ execution:

```
SQL> DECLARE
  2        -- taking input for variable a
  3        a number := &a;
  4        -- taking input for variable b
  5        b varchar2(30) := &b;
  6  BEGIN
  7        c := a + b ;
  8       dbms_output.put_line('Sum of '||a||' and '||b||' is = '||c);
  9  END;
 10  /
Enter value for a: 4
old   3:       a number := &a;
new   3:       a number := 4;
Enter value for b: 9
old   5:       b varchar2(30) := &b;
new   5:       b varchar2(30) := 9;
      c := a + b ;
      *
ERROR at line 7:
ORA-06550: line 7, column 7:
PLS-00201: identifier 'C' must be declared
ORA-06550: line 7, column 7:
PL/SQL: Statement ignored
```

Program fixing of an error

Following program executes successfully:

Program with SELECT INTO

Following program executes successfully with the retrieval of single record

```
SQL> DECLARE
  2       v_id     INTEGER;
  3       v_empno  NUMBER;
  4  BEGIN
  5      v_id := 7369;
  6      SELECT EMPNO
  7         INTO V_EMPNO
  8         FROM  EMP
  9         WHERE empno = v_id;
 10      DBMS_OUTPUT.PUT_LINE('Value is '||v_empno);
 11  EXCEPTION
 12      WHEN NO_DATA_FOUND THEN
 13       DBMS_OUTPUT.PUT_LINE('No record exists');
 14  END;
 15  /
Value is 7369

PL/SQL procedure successfully completed.
```

Program with SELECT INTO

Following program is terminated with the retrieval of single record

```
SQL> DECLARE
  2       v_id     INTEGER;
  3       v_empno  NUMBER;
  4  BEGIN
  5      v_id := 7777; --employee does not exists
  6      SELECT EMPNO
  7         INTO V_EMPNO
  8         FROM  EMP
  9         WHERE empno = v_id;
 10      DBMS_OUTPUT.PUT_LINE('Value is '||v_empno);
 11  END;
 12  /
DECLARE
*
ERROR at line 1:
ORA-01403: no data found
ORA-06512: at line 6
```

Program with SELECT INTO

Following program is successfully executed with the retrieval of single record and exception raised.

```
SQL> DECLARE
  2       v_id     INTEGER;
  3       v_empno  NUMBER;
  4  BEGIN
  5      v_id := 7777; --employee does not exists
  6      SELECT EMPNO
  7         INTO V_EMPNO
  8         FROM  EMP
  9         WHERE empno = v_id;
 10      DBMS_OUTPUT.PUT_LINE('Value is '||v_empno);
 11  EXCEPTION
 12      WHEN NO_DATA_FOUND THEN
 13       DBMS_OUTPUT.PUT_LINE('No record exists');
 14  END;
 15  /
No record exists

PL/SQL procedure successfully completed.
```

Lesson123: Compile & Run Program

- Program Compilation & Execution
- Program execution with an error
- Run Program using SQL Developer
- Compile Program using SQL Developer

Program Compilation & Execution

Following program compile and executes on SQL prompt:

DECLARE

BEGIN

EXCEPTION

END;

Following program run without compiling and execution on SQL prompt:

DECLARE

BEGIN

EXCEPTION

END;

.

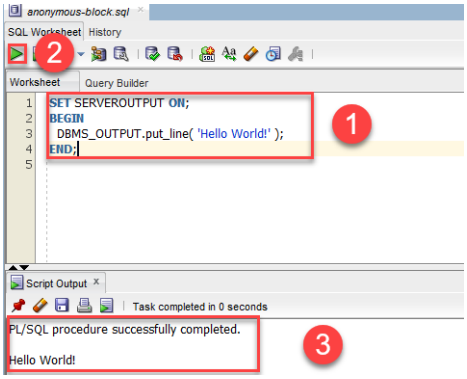
Program execution with an error

Following program executes with an error during compilation/ execution:

```
SQL> DECLARE
  2        -- taking input for variable a
  3        a number := &a;
  4        -- taking input for variable b
  5        b varchar2(30) := &b;
  6  BEGIN
  7        c := a + b ;
  8       dbms_output.put_line('Sum of '||a||' and '||b||' is = '||c);
  9  END;
 10  /
Enter value for a: 4
old   3:       a number := &a;
new   3:       a number := 4;
Enter value for b: 9
old   5:       b varchar2(30) := &b;
new   5:       b varchar2(30) := 9;
      c := a + b ;
      *
ERROR at line 7:
ORA-06550: line 7, column 7:
PLS-00201: identifier 'C' must be declared
ORA-06550: line 7, column 7:
PL/SQL: Statement ignored
```

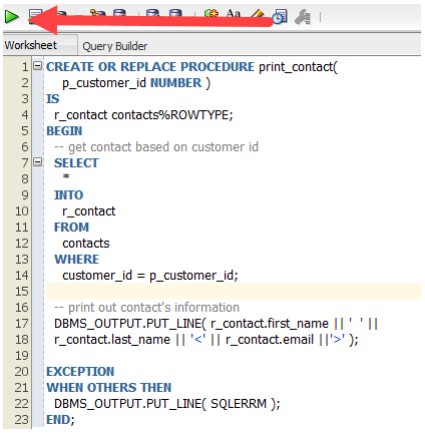
Run Program using SQL Developer

To compile the procedure, you click the Run Statement button:



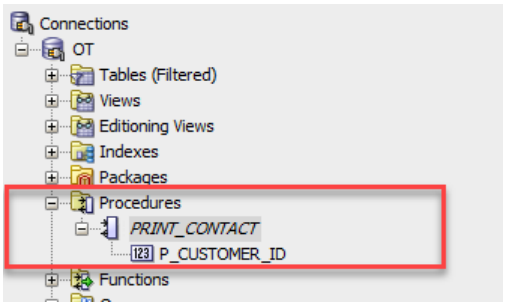
Run Program using SQL Developer

To compile the procedure, you click the Run Statement button:



Run Program using SQL Developer

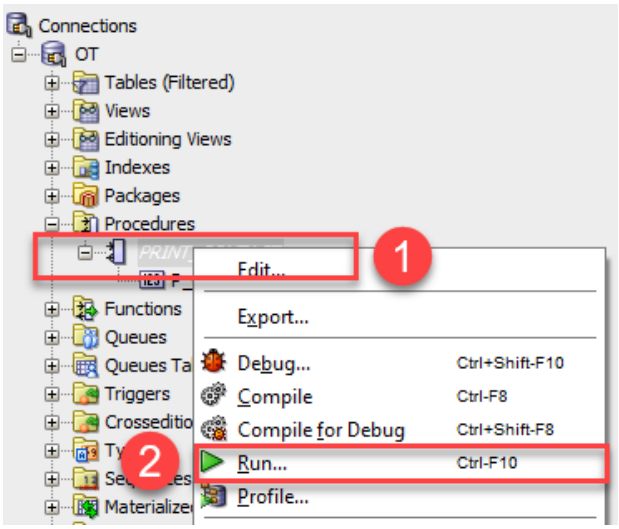
If the procedure is compiled successfully, you will find the new procedure under the Procedures node as shown below:



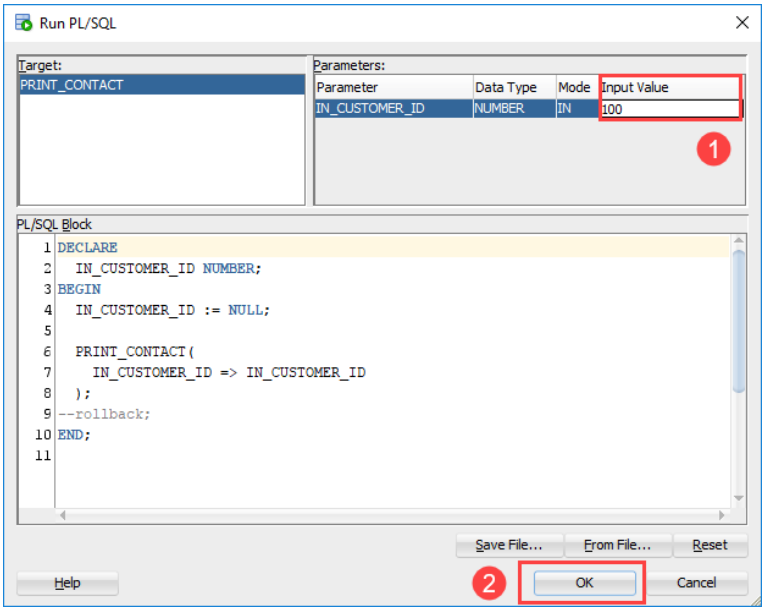
EXEC procedure_name(arguments);

This link or path is not available for blocks codes without name

Right-click the procedure name and choose Run ... menu item:

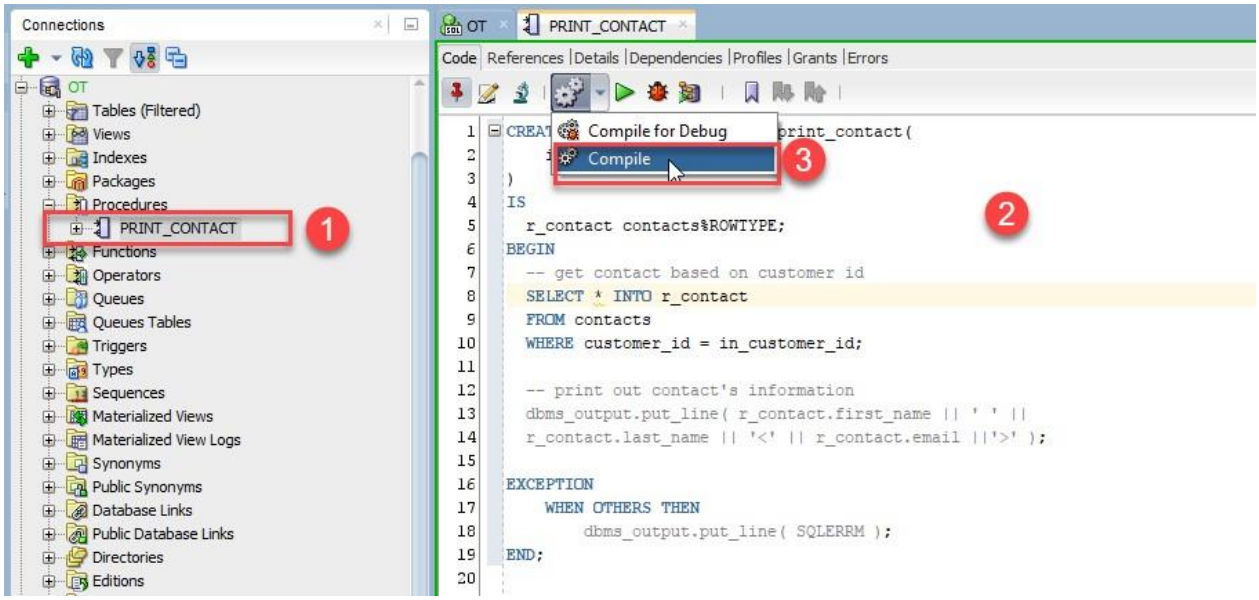


Enter a value for the in_customer_id parameter and click OK button:



Compile Program using SQL Developer

Compiling program with selection steps as shown:



Lesson124: Data Types & Select INTO Clause

- Introduction
- Scalar Data Types Families
- Numeric Data Types
- Boolean Data Types
- Character or String Data Types
- DateTime Data Types
- Composite Data Types
- Data Types Synonyms

Introduction

- Each value in PL/SQL such as a constant, variable and parameter has a data type that determines the storage format, valid values, and allowed operations.
- PL/SQL has two kinds of data types: scalar and composite.

Scalar Data Types Families

- Number
- Boolean
- Character
- Datetime

A scalar data type may have subtypes. A subtype is a data type that is a subset of another data type, which is its base type. A subtype further defines a base type by restricting the value or size of the base data type.

Numeric Data Types

- The numeric data types represent real numbers, integers, and floating-point numbers. They are stored as NUMBER, IEEE floating-point storage types (BINARY_FLOAT and BINARY_DOUBLE), and PLS_INTEGER.
- The data types NUMBER, BINARY_FLOAT, and BINARY_DOUBLE are SQL data types.
- The PLS_INTEGER datatype is specific to PL/SQL. It represents signed 32 bits int.

Boolean Data Types

- The BOOLEAN datatype has three data values: TRUE, FALSE, and NULL. Boolean values are typically used in control flow structure such as IF-THEN, CASE, and loop statements like LOOP, FOR LOOP, and WHILE LOOP.
- SQL does not have the BOOLEAN data type, you can assign data type NUMBER(1) or CHAR(1)

Character or String Data Types

- The character data types represent alphanumeric text. PL/SQL uses the SQL character data types such as
- CHAR, VARCHAR2, LONG, RAW, LONG RAW, ROWID, and UROWID
- CHAR is used for fixed length columns such primary keys or boolean
- The character data types represent alphanumeric text. PL/SQL uses the SQL character data types such as
- CHAR, VARCHAR2, LONG, RAW, LONG RAW, ROWID, and UROWID
- CHAR is used for fixed length columns such primary keys or boolean

DateTime Data Types

- The datetime data types represent dates, timestamp with or without time zone and intervals. PL/SQL datetime data types are DATE, TIMESTAMP, TIMESTAMP WITH TIME ZONE, TIMESTAMP WITH LOCAL TIME ZONE, INTERVAL YEAR TO MONTH, and INTERVAL DAY TO SECOND

Composite Data Types

- TableName.Column%TYPE is a composite data type. For example, EMP.sal%TYPE
- It will follow same data type and size of column of a table.
- In case if size or type of column is changed then there is no need to re-compile the program

Data Types Synonyms

Data Type	Synonyms
NUMBER	DEC, DECIMAL, DOUBLE PRECISION, FLOAT, INTEGER, INT, NUMERIC, REAL, SMALLINT
CHAR	CHARACTER, STRING
VARCHAR2	VARCHAR

SELECT INTO Clause

We cannot use simple SQL Select clause statement, rather we use Select INTO by storing values in variables with condition of choosing single record

```
DECLARE
        v_salary    EMP.sal%TYPE;
        v_deptno  NUMBER(3);
BEGIN
        SELECT sal, deptno
        INTO v_salary, v_deptno
        FROM EMP
        WHERE empno = 7369;
...
END;
/
```

Lesson125: Program Examples

Program Examples with Select INTO Clause

- Program-1 without Exception
- Program-2 using Exception
- Program-3 using Constant identifier with Exception

Program-1 without Exception

```
declare
  v_ename           emp.ename%TYPE;
  v_sal             emp.sal%TYPE;
begin
   Select ename, sal
   INTO v_ename, v_sal
   from emp
   where empno = '7844'; --This id exists

     DBMS_OUTPUT.PUT_LINE('solution is:'||v_ename||' has '||v_sal);

end;
/

Before executing code that contains DBMS_OUTPUT.PUT_LINE,
must run at SQL prompt: set serveroutput on
```

Program-2 using Exception

```
declare
  v_ename           emp.ename%TYPE;
  v_sal             emp.sal%TYPE;
begin
   Select ename, sal
   INTO v_ename, v_sal
   from emp
   where empno = '7844'; --This id exists

     DBMS_OUTPUT.PUT_LINE('solution is:'||v_ename||' has '||v_sal);

exception
  When others then
   DBMS_OUTPUT.PUT_LINE('no record found');
end;
/
```

Program-3 using Constant identifier with Exception

```
declare
  v_ename           emp.ename%TYPE;
  v_sal             emp.sal%TYPE;
  v_empno           emp.empno%TYPE := '7844'; --Constant identifier
begin
   Select ename, sal
   INTO v_ename, v_sal
   from emp
   where empno = v_empno; --This id exists

     DBMS_OUTPUT.PUT_LINE('solution is:'||v_ename||' has salary '||v_sal);

exception
  When others then
   DBMS_OUTPUT.PUT_LINE('no record found');
end;
/
```

Lesson 126: Control IF Conditions

PL/SQL Control IF-THEN-ELSE Conditions

- IF THEN Condition
- Structure of IF THEN Condition
- Example of IF THEN Condition
- Structure of IF THEN ELSE Condition
- Example of IF THEN ELSE Condition
- Structure of IF THEN ELSIF Condition
- Example of IF THEN ELSIF Condition

IF THEN Condition

There are three ways to write IF condition:

- IF – END IF
- IF – ELSE – END IF
- IF – ELSIF – ELSE – END IF

Structure of IF THEN Condition

There are three ways to write IF condition:

IF condition **THEN**

statements;

END IF;

For example:

IF v_sal > 2000 Then

total_sal := v_sal + 5000;

END IF;

Example of IF THEN Condition

```
DECLARE
   v_sal EMP.SAL%TYPE;
BEGIN
   Select sal INTO v_sal
   From emp
   Where ename = 'SMITH';

   IF v_sal < 850 THEN
    DBMS_OUTPUT.PUT_LINE('Low paid salary person');
   END IF;
END;
/
```

Structure of IF THEN ELSE Condition

There are three ways to write IF condition:

IF condition **THEN**

statements;

ELSE

statements;

END IF;

Example of IF THEN ELSE Condition

```
DECLARE
   v_sal EMP.SAL%TYPE;
BEGIN
   Select sal INTO v_sal
   From emp
   Where ename = 'SMITH';

   IF v_sal < 850 THEN
    DBMS_OUTPUT.PUT_LINE('Low paid salary person');
   ELSE
    DBMS_OUTPUT.PUT_LINE('Higly paid salary person');
   END IF;
END;
/
```

Structure of IF THEN ELSIF Condition

There are three ways to write IF condition:

IF condition **THEN**
  statements;
ELSIF
  statements;
ELSE

  statements ;

END IF;

Example of IF THEN ELSIF Condition

```
DECLARE
   v_sal EMP.SAL%TYPE;
BEGIN
   Select sal INTO v_sal
   From emp
   Where ename = 'SMITH';

   IF v_sal < 850 THEN
     DBMS_OUTPUT.PUT_LINE('Low paid salary person');
   ELSIF (v_sal >= 850 AND v_sal <=1000) THEN
     DBMS_OUTPUT.PUT_LINE('Average paid salary person');
   ELSE
     DBMS_OUTPUT.PUT_LINE('Higly paid salary person');
   END IF;
END;
/
```

Lesson 127: FOR & WHILE Loops

LOOPS, FOR & WHILE Loops

- LOOPS
- Loop with EXIT
- FOR Loop
- FOR Loop Syntax
- FOR Loop - Example
- FOR Loop - Bulk Data Insertion
- WHILE Loop
- WHILE Loop - Example
- WHILE Loop Syntax

LOOPS

- LOOP – EXIT WHEN – END LOOP
- FOR – LOOP – END LOOP
- WHILE – LOOP – END LOOP

Loop with EXIT

Following is its example (Repeat until LOOP):

```
DECLARE
     ...
     v_deptno   dept.deptno%TYPE  := 50;
       v_counter  integer := 1;
     ...
BEGIN
      ...
      LOOP
         INSERT INTO dept(deptno)
                   VALUES(v_deptno);
         v_counter := v_counter + 1;
         v_deptno := v_deptno + 10;
         EXIT WHEN v_counter > 5;
     END LOOP;
     ...
END;
/
```

FOR Loop

Following is its Syntax:

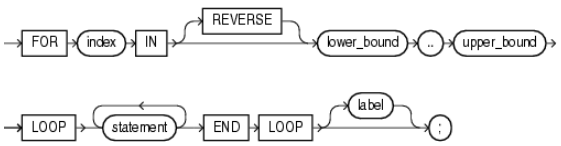
FOR index IN lower_bound .. upper_bound **LOOP**

statements;

END LOOP;

FOR Loop Syntax

Following is its generic syntax:



FOR Loop - Example

Following example displays serial numbers:

```
BEGIN
  FOR l_counter IN 1..5
  LOOP
    DBMS_OUTPUT.PUT_LINE( l_counter );
  END LOOP;
END;
/
```

FOR Loop - Example

Following example displays serial numbers in reverse:

```
BEGIN
  FOR l_counter IN REVERSE 1..3
  LOOP
    DBMS_OUTPUT.PUT_LINE( l_counter );
  END LOOP;
END;
/
```

FOR Loop - Bulk Data Insertion

Following example displays serial numbers in reverse:

```
BEGIN
  FOR i IN 55..70
  LOOP
    insert into dept(deptno, dname, loc)
    values (i, 'Dummy Dept-'||i, 'No Loc');
    DBMS_OUTPUT.PUT_LINE( i );
  END LOOP;
  Commit;
END;
/
```

```
DEPTNO DNAME          LOC
------- -------------- -------------
     55 Dummy Dept-55  No Loc
     56 Dummy Dept-56  No Loc
     57 Dummy Dept-57  No Loc
     58 Dummy Dept-58  No Loc
     59 Dummy Dept-59  No Loc
     60 Dummy Dept-60  No Loc
     10 ACCOUNTING     NEW YORK
```

WHILE Loop

Following is its Syntax:

WHILE condition

LOOP

statements;

END LOOP;

WHILE Loop - Example

Following is its example:

```
DECLARE
  n_counter NUMBER := 1;
BEGIN
  WHILE n_counter <= 5
  LOOP
    DBMS_OUTPUT.PUT_LINE( 'Counter : ' || n_counter );
    n_counter := n_counter + 1
  END LOOP;
END;
/
```

WHILE Loop Syntax

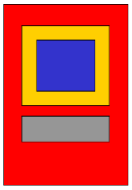
Following is its generic syntax:

Lesson 128: Anonymous Blocks with Examples

- Nesting Anonymous Blocks
- Nesting Anonymous Blocks – Example
- Repeat Until LOOP - Example

Nesting Anonymous Blocks

Blocks within blocks give the concept of Nested blocks



Repeat Until LOOP - Example

This loop executes once first time. It is same as DO-WHILE Loop.

```
LOOP
   monthly_value := daily_value * 31;
   EXIT WHEN monthly_value > 4000;
END LOOP;
```

Nesting Anonymous Blocks - Example

```
DECLARE
  n_counter NUMBER := 1;
BEGIN
  DBMS_OUTPUT.PUT_LINE('*************----Outer Block----*************');
  FOR i IN 55..70
  LOOP
    DBMS_OUTPUT.PUT_LINE( 'Outer Block Counter : ' || i );
  END LOOP;
  BEGIN
        WHILE n_counter <= 5
        LOOP
        DBMS_OUTPUT.PUT_LINE( 'Inner Block Counter : ' || n_counter );
        n_counter := n_counter + 1;
  END LOOP;
  END; --inner block end
END; --outer block end
/
```

Lesson 129: Exception Handlings-1

PL/SQL Exception Handlings

- PL/SQL Exception
- PL/SQL Exception Usage
- PL/SQL Exception - Example-1
- PL/SQL Exception - Example-2

PL/SQL Exception

- PL/SQL treats all errors that occur in an anonymous block, procedure, or function as exceptions.
- The exceptions can have different causes such as coding mistakes, bugs, even hardware failures.
- It is not possible to anticipate all potential exceptions, however, you can write code to handle exceptions to enable the program to continue running as normal.

PL/SQL Exception

- The code that you write to handle exceptions is called an exception handler.
- A PL/SQL block can have an exception-handling section, which can have one or more exception handlers.

PL/SQL Exception Usage

```
BEGIN
   -- executable section ...
   -- exception-handling section
   EXCEPTION
      WHEN e1 THEN
         -- exception_handler1
      WHEN e2 THEN
         -- exception_handler1
      WHEN OTHERS THEN
         -- other_exception_handler
END;
/
```

PL/SQL Exception – Example-1

Following program will terminate without successfully execution when employee no does not exist in table EMP.

```
DECLARE
    v_ename EMP.ename%TYPE;
    v_empno EMP.empno%TYPE := &empno;
BEGIN
    -- get the employee name by id
    SELECT ename INTO v_ename
    FROM EMP
    WHERE empno = v_empno;

    -- show the customer ename
    dbms_output.put_line('Employee name is ' || v_ename);
END;
/
```

PL/SQL Exception – Example-1
Following program will execute successfully when employee no does not exist in table EMP.

```
DECLARE
    v_ename EMP.ename%TYPE;
    v_empno EMP.empno%TYPE := &empno;
BEGIN
    -- get the employee name by id
    SELECT ename INTO v_ename
    FROM EMP
    WHERE empno = v_empno;

    -- show the customer ename
    dbms_output.put_line('Employee name is ' || v_ename);
EXCEPTION
        WHEN NO_DATA_FOUND THEN
          dbms_output.put_line('Employee ID' || v_empno ||  ' does not exist');
END;
/
```

PL/SQL Exception – Example-2

Following program will execute successfully using multiple Exceptions.

```
DECLARE
    v_ename EMP.ename%TYPE;
    v_job EMP.job%TYPE := &job;
BEGIN
    -- get the employee name by id
    SELECT ename INTO v_ename
    FROM EMP
    WHERE job = v_job;

    -- show the customer ename
    dbms_output.put_line('Employee name is ' || v_ename);
EXCEPTION
        WHEN NO_DATA_FOUND THEN
          dbms_output.put_line('Employee ID' || v_ename ||  ' does not exist');
        WHEN TOO_MANY_ROWS THEN
          dbms_output.put_line('Database returns many employees');
END;
/
```

```
Enter value for job: 'CLERK'
old   3:     v_job EMP.job%TYPE := &job;
new   3:     v_job EMP.job%TYPE := 'CLERK';
Database returns many employees

PL/SQL procedure successfully completed.
```

Lesson 130: Exception Handlings-2

PL/SQL Exception Handlings

- Raise Exception
- User-Defined Exception
- Affecting PL/SQL Exception
- Other Exceptions

Raise Exception

Exceptions are raised by the database server automatically whenever there is any internal database error, but exceptions can be raised explicitly by the programmer by using the command **RAISE**. Following is the simple syntax for raising an exception:

```
DECLARE
   exception_name EXCEPTION;
BEGIN
   IF condition THEN
      RAISE exception_name;
   END IF;
EXCEPTION
   WHEN exception_name THEN
   statement;
END;
/
```

User-Defined Exception

```
DECLARE
   v_empno EMP.empno%type := &cv_empno;
   v_name  EMP.ename%type;
   v_job EMP.job%type;
   -- user defined exception
   ex_invalid_empno  EXCEPTION;
BEGIN
   IF v_empno <= 0 THEN
      RAISE ex_invalid_empno;
   ELSE
      SELECT  ename, job INTO  v_name, v_job
      FROM EMP
      WHERE empno = v_empno;
      DBMS_OUTPUT.PUT_LINE ('Name: '||  v_name);
      DBMS_OUTPUT.PUT_LINE ('job: ' || v_job);
   END IF;

EXCEPTION |
   WHEN ex_invalid_empno THEN
      dbms_output.put_line('empno must be greater than zero!');
   WHEN no_data_found THEN
      dbms_output.put_line('No such employee!');
   WHEN others THEN
      dbms_output.put_line('Error!');
END;
/
```

Affecting PL/SQL Exception

One exception does not affect the rest of program

```
DECLARE
v_tname varchar2(100);
BEGIN
 BEGIN
  select tname into v_tname
  from tab
  where tname='CMG_EFFORTT';
  DBMS_OUTPUT.PUT_LINE('table name:'||v_tname);
  exception
    when no_data_found then
    DBMS_OUTPUT.PUT_LINE('No data');
 END;
 select tname into v_tname
  from tab
  where tname='CMG_EFFORT';
  DBMS_OUTPUT.PUT_LINE('table name:'||v_tname);
 exception
 when no_data_found then
    DBMS_OUTPUT.PUT_LINE('No data second exception');
 --go to 1;
END;
/
```

Other Exceptions

Examples are
NO_DATA_FOUND, ZERO_DIVIDE, OTHERS
Raise some exception
WHEN NO_DATA_FOUND THEN
raise_application_error(-20011,'Invalid FK value');
Raise My_Excep;
To display details of oracle standard error message
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Error detail is: '||SQLERRM)

Lesson 131: PL/SQL Procedure

PL/SQL Procedure

- What is a Procedure in PL/SQL?
- Procedure - Example
- Compiling & Executing Procedure
- Invoking a Procedure in a Block
- Executing a Procedure & Show Errors
- Data Dictionary for Procedures
- Managing Procedure using SQL Developer

What is a Procedure in PL/SQL?

- Is a block with a name
- The DECLARE key word is not used
- Parameters can be

IN

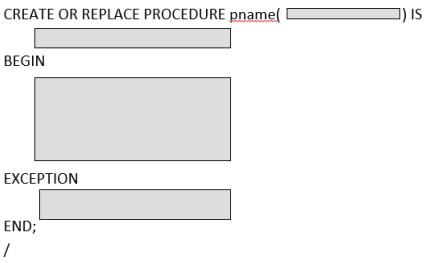
OUT

IN OUT

- Is stored (USER_SOURCE)

What is Procedure in PL/SQL?

Writing PL/SQL procedure



Procedure – Example

```
SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE
    proc_test(p_empno IN VARCHAR2)  IS
       v_job     EMP.job%TYPE;
       v_sal     EMP.sal%TYPE;
BEGIN
         SELECT job, sal
         INTO  v_job,v_sal
         FROM  emp
         WHERE  empno = p_empno;
         DBMS_OUTPUT.PUT_LINE('job is '||v_job);
EXCEPTION
     WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('ERROR...');
END;
/
```

Compiling & Executing Procedure

```
SQL> Show errors


SQL> execute proc_test(5893);
```

Invoking a Procedure in a Block

```
DECLARE
   …
BEGIN
   …
   proc_test('23');
   …
END;
/

Or
SQL> exec proc_test('1123')
```

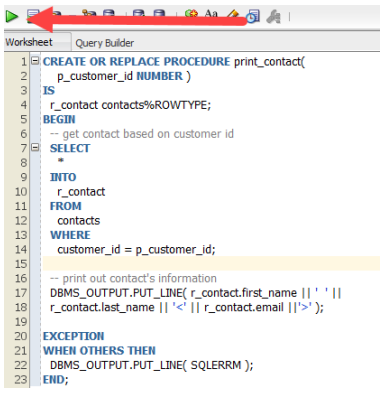
Executing a Procedure & Show Errors

```
  1  create or replace procedure test_proc is
  2       v_id  INTEGER;
  3      v_empno emp.empno%TYPE;
  4  BEGIN
  5      v_id := 1234567;
  6      select empno into v_empno
  7      FROM  EMP
  8      WHERE empno = v_id;
  9  EXCEPTION
 10      WHEN NO_DATA_FOUND THEN
 11       DBMS_OUTPUT.PUT_LINE('No record exists');
 12*   END;
SQL> /

Procedure created.

SQL> exec test_proc
No record exists

PL/SQL procedure successfully completed.

SQL> show errors               (to see errors for procedures, functions)
```

Data Dictionary for Procedures

```
SQL> select Text from user_source
  2  where name='PROC_TEST';

TEXT
--------------------------------------------------
PROCEDURE
    proc_test(p_empno IN VARCHAR2)  IS
       v_job  EMP.job%TYPE;
       v_sal  EMP.sal%TYPE;
BEGIN
 SELECT job, sal
 INTO  v_job,v_sal
 FROM  emp
         WHERE  empno = p_empno;
 DBMS_OUTPUT.PUT_LINE('job is '||v_job);
EXCEPTION

TEXT
--------------------------------------------------
     WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('ERROR...');
END;
```

Managing Procedure using SQL Developer



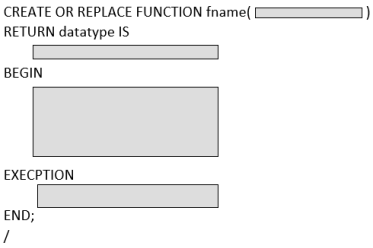
Lesson 132: PL/SQL Function

PL/SQL Function

- What is Function in PL/SQL?
- Function - Example
- Compiling & Executing Function
- Executing a Function & Show Errors
- Data Dictionary for Function
- Is a block with a name
- Returns one value only.
- The DECLARE key word is not used
- Parameters can only be IN
- Is stored (USER_SOURCE)

What is Function in PL/SQL?

Writing PL/SQL Function



Function - Example

```
CREATE OR REPLACE FUNCTION
    func_test(p_empno IN VARCHAR2)  RETURN NUMBER IS
       v_job     EMP.job%TYPE;
       v_sal     EMP.sal%TYPE;
BEGIN
         SELECT job, sal
         INTO  v_job,v_sal
         FROM  emp
         WHERE  empno = p_empno;
         DBMS_OUTPUT.PUT_LINE('job is '||v_job);
         return v_sal;
EXCEPTION
     WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('ERROR...');
END;
/

SQL> Show errors
```

Compiling & Executing Function

SET SERVEROUTPUT ON;

DECLARE

…

v_sal emp.sal%TYPE;

…

```
BEGIN
    …
    v_sal := func_test(3477);
     DBMS_OUTPUT.PUT_LINE(TO_CHAR(v_sal));
    …
END;
/
```

Executing a Function & Show Errors

```
SQL> CREATE OR REPLACE FUNCTION
  2      func_test(p_empno IN VARCHAR2)  RETURN NUMBER IS
  3         v_job  EMP.job%TYPE;
  4         v_sal  EMP.sal%TYPE;
  5  BEGIN
  6   SELECT job, sal
  7   INTO  v_job,v_sal
  8   FROM  emp
  9           WHERE  empno = p_empno;
 10   DBMS_OUTPUT.PUT_LINE('job is '||v_job);
 11           return v_sal;
 12  EXCEPTION
 13       WHEN OTHERS THENv_sal;
 14           DBMS_OUTPUT.PUT_LINE('ERROR...');
 15  END;
 16  /

Warning: Function created with compilation errors.

SQL> show error
Errors for FUNCTION FUNC_TEST:

LINE/COL ERROR
-------- -----------------------------------------------------------------
13/18    PLS-00103: Encountered the symbol "THENV_SAL" when expecting one
         of the following:
         then or
         The symbol "then" was substituted for "THENV_SAL" to continue.
```

Data Dictionary for Function

```
SQL> select text from user_source
  2  where name='FUNC_TEST';

TEXT
--------------------------------------------------------------------------------
FUNCTION
    func_test(p_empno IN VARCHAR2)  RETURN NUMBER IS
       v_job  EMP.job%TYPE;
       v_sal  EMP.sal%TYPE;
BEGIN
 SELECT job, sal
 INTO  v_job,v_sal
 FROM  emp
         WHERE  empno = p_empno;
 DBMS_OUTPUT.PUT_LINE('job is '||v_job);
         return v_sal;

TEXT
--------------------------------------------------------------------------------
EXCEPTION
     WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('ERROR...');
END;
```

Lesson 133: Calling Procedures & Functions

- Calling Procedure
- Calling Function
- Calling Procedure with Parameters

Calling Procedure

Procedure can be called after compilation in some other function, procedure, block or package. For example,

```
DECLARE
   …
BEGIN
   …
   proc_test('23');
   …
END;
/

Or
SQL> exec proc_test('1123')
```

Calling Function

Function can be called after compilation in some other function, procedure, block or package by assigning it to variable.

```
DECLARE
   …
   v_sal        emp.sal%TYPE;
   …
BEGIN
    …
    v_sal := func_test(3477);
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(v_sal));
    …
END;
/
```

Calling Procedure with Parameters

```
create or replace procedure proc_inout
                            (p_empno in number,
                             p_sal out number) is
BEGIN
   select sal
   into p_sal
   from emp
     where empno=p_empno;
END;
/

create or replace procedure proc_out is
  v_sal EMP.SAL%TYPE;
BEGIN
    proc_inout(7369,v_sal);
    DBMS_OUTPUT.PUT_LINE('salary is :'||v_sal);
END;
/

SQL> exec proc_out
salary is :2000

PL/SQL procedure successfully completed.
```

Lesson 134: Exercises of Functions

Exercises of Functions

- Compiling & Executing Function
- Function - Exercise-1
- Function - Exercise-2

Compiling & Executing Function

```
SET SERVEROUTPUT ON;

DECLARE

   …

   v_sal       emp.sal%TYPE;

   …

BEGIN

    …

    v_sal := func_test(3477);

     DBMS_OUTPUT.PUT_LINE(TO_CHAR(v_sal));

    …

END;

/
```

Function – Exercise-1

```
CREATE OR REPLACE FUNCTION
    func_MaxSal(p_deptno IN NUMBER)  RETURN NUMBER IS
        v_MaxSal         EMP.sal%TYPE;
BEGIN
         SELECT max(sal)
         INTO  v_MaxSal
         FROM  emp
         where deptno = p_deptno;
         DBMS_OUTPUT.PUT_LINE('Highest Salary is '||v_MaxSal);
         return v_MaxSal;
EXCEPTION
     WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('ERROR...'||SQLERRM);
END;
/
```

```
SQL> DECLARE
  2    vv_MaxSal EMP.SAL%TYPE;
  3  BEGIN
  4    vv_MaxSal := func_MaxSal(10);
  5    DBMS_OUTPUT.PUT_LINE('Calling from block:'||vv_MaxSal);
  6  END;
  7  /
Highest Salary is 5000
Calling from block:5000
```

Function – Exercise-2

```
CREATE OR REPLACE FUNCTION fact(x number)
RETURN number IS
   f number;
BEGIN
   IF x=0 THEN
      f := 1;
   ELSE
      f := x * fact(x-1);
   END IF;
RETURN f;
END;
/

DECLARE
   num number;
   factorial number;
BEGIN
   num:= 6;
   factorial := fact(num);
   dbms_output.put_line(' Factorial '|| num || ' is ' || factorial);
END;
/
```

```
Factorial 6 is 720

PL/SQL procedure successfully completed.
```

Lesson 135: Exercises of Procedures

Exercises of Procedures

- Compiling & Executing Function
- Procedure Exercise-I

Compiling & Executing Procedure

DECLARE

…

BEGIN

…

proc_test('23');

…

END;

/

Or

SQL> exec proc_test(‘1123’)

Procedure – Exercise-1

```
CREATE OR REPLACE PROCEDURE
    proc_MaxSal(p_deptno IN NUMBER, p_MaxSal OUT NUMBER) IS
       v_MaxSal         EMP.sal%TYPE;
BEGIN
         SELECT max(sal)
         INTO  v_MaxSal
         FROM  emp
         where deptno = p_deptno;
         DBMS_OUTPUT.PUT_LINE('Highest Salary is '||v_MaxSal);
         p_MaxSal := v_MaxSal;
EXCEPTION
     WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('ERROR...'||SQLERRM);
END;
/

CREATE OR REPLACE PROCEDURE proc_CALLING(p_dno IN NUMBER) IS
  vv_MaxSal EMP.SAL%TYPE;
BEGIN
  proc_MaxSal(p_dno, vv_MaxSal);
  DBMS_OUTPUT.PUT_LINE('Calling from block:'||vv_MaxSal);
END;
/


SQL> execute proc_CALLING(20);
```

```
SQL> execute proc_CALLING(20);
Highest Salary is 3400
Calling from block:3400

PL/SQL procedure successfully completed.
```

Lesson 136: PL/SQL Package-1

PL/SQL Package-1

- Purpose of PL/SQL Package
- PL/SQL Package contains …
- Schematic Diagram of Package
- PL/SQL Package Specification
- PL/SQL Package Body
- Package Specification & its Body

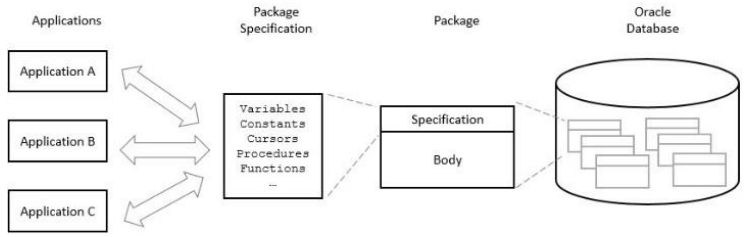
Purpose of PL/SQL Package

- In PL/SQL, a package is a schema object that contains definitions for a group of related functionalities.
- A package includes variables, constants, cursors, exceptions, procedures, functions, and subprograms.
- It is compiled and stored in the Oracle Database.

PL/SQL Package contains …

- Typically, a package has a specification and a body. A package specification is mandatory while the package body can be required or optional, depending on the package specification.

Schematic Diagram of Package



https://www.oracletutorial.com/plsql-tutorial/plsql-package/

PL/SQL Package Specification

- The package specification declares the public objects that are accessible from outside the package.
- If a package specification whose public objects include cursors and subprograms, then it must have a body which defines queries for the cursors and code for the subprograms.

PL/SQL Package Body

- A package body contains the implementation of the cursors or subprograms declared in the package specification. In the package body, you can declare or define private variables, cursors, etc., used only by package body itself.
- A package body can have an initialization part whose statements initialize variables or perform other one-time setups for the whole package.
- A package body can also have an exception-handling part used to handle exceptions.

Package Specification & its Body

```
CREATE OR REPLACE PACKAGE emp_info IS

END emp_info;
/

CREATE OR REPLACE PACKAGE BODY emp_info is

END emp_info; --end of package body
/
```

Lesson 137: PL/SQL Package-2

- What is a Package in PL/SQL?
- Package Specification
- Package Body
- Add in a Package Body by hiding from User Specification
- Invoking Package Subprogram

What is a Package in PL/SQL?

- Selected and purposeful collection of procedures, functions and variables is managed in package.
- Package
- Package Specification
- Package Body
- Invoking Package subprogram
- Abstract Programming
- To avoid spreading of procedures & functions
- Is stored (USER_SOURCE)

Package Specification

```
CREATE OR REPLACE PACKAGE emp_info IS

   v_count   INTEGER;  --global variable for all sub programs inside package

   PROCEDURE insert_record( p_empno IN NUMBER
                          , p_ename IN VARCHAR2
                          , p_job IN VARCHAR2
                          , p_sal IN NUMBER
                          , p_comm IN NUMBER
                          , p_deptno IN VARCHAR2);

   PROCEDURE delete_record(p_empno IN NUMBER);

   FUNCTION sum_dept_sal( p_deptno IN dept.deptno%TYPE) RETURN is dept.sal%TYPE;

END emp_info;
/
```

We have declarative parts, parameters options

Package Body

```
PROCEDURE insert_record(p_empno IN NUMBER,p_ename IN VARCHAR2,
p_job IN VARCHAR2,p_sal IN NUMBER,
p_comm IN NUMBER,p_deptno IN VARCHAR2) IS
BEGIN
      INSERT INTO EMP(empno,ename,job,sal,comm,deptno)
      VALUES(p_empno,p_ename,p_job,p_sal,p_comm,p_deptno);
END insert_record;
```

```
PROCEDURE delete_record(p_empno IN NUMBER) IS
BEGIN
    DELETE FROM EMP
    WHERE empno=p_empno;
END delete_record;
```

```
FUNCTION  sum_dept_sal(p_deptno IN NUMBER) RETURN emp.sal%TYPE IS
v_sum_sal   emp.sal%TYPE;
BEGIN
   SELECT SUM(sal)
   INTO v_sum_sal
    FROM emp
    WHERE deptno = p_deptno;
   RETURN v_sum_sal;
END;
```

Purpose of PL/SQL Package

Following are its main purposes:

- Make code more modular
- Hide implementation details
- Improve application performance
- Minimize unnecessary recompiling code
- Manage authorization easily

Add in a Package Body by hiding from User Specification

```
Procedure call_from_out is
   v_empno EMP.empno%TYPE;
   v_ename EMP.ename%TYPE;
   v_job   EMP.job%TYPE;
   v_sal   EMP.sal%TYPE;
   v_comm  EMP.comm%TYPE;
   v_deptno EMP.dept%TYPE;
begin

   v_empno := 8899;
   v_ename := 'ALI';
   v_job   := 'MANAGER';
   v_sal   := 8000;
   v_comm  := 300;
   v_deptno := 10;

   insert_record(  v_empno, v_ename, v_job, v_sal, v_comm,v_deptno);
    commit;
EXCEPTION
   WHEN OTHERS THEN
     DBMS_OUTPUT.PUT_LINE('Except call_from_out:'||SQLERRM);
end;
/
```

Invoking Package Subprogram

```
DECLARE
   v_sum_sal   emp.sal%TYPE;
BEGIN
   ...
   v_sum_sal := emp_info.sum_dept_sal(10);
   ...
   emp_info.delete_record(1234567);
   ...
END;
/
```

Or

```
SQL> exec emp_info.insert_record(...,...)
```

Lesson 138: Execution of Packages

- Compiling a Package using SQL*Plus
- Compiling a Package using Oracle Developer
- Compiling a Package using Files on SQL*Plus
- Calling Function and Procedure from Package

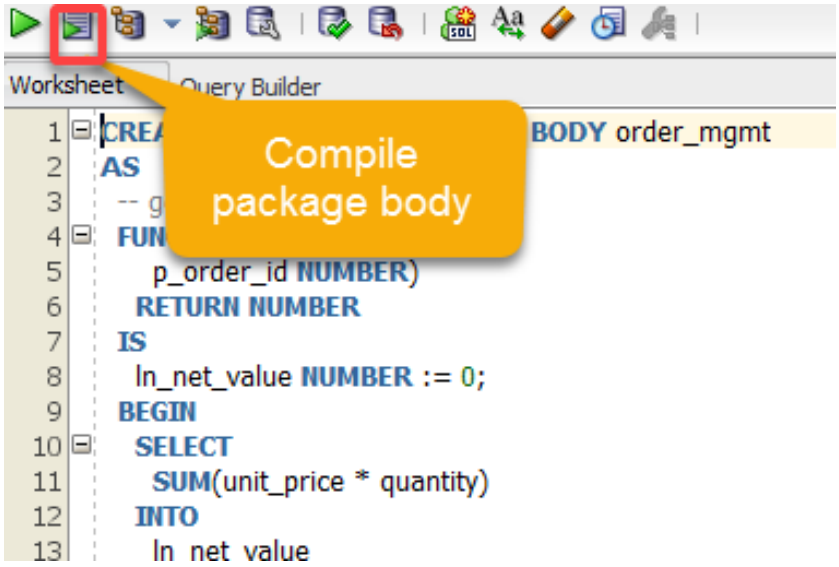
Compiling a Package using SQL*Plus

If you use SQL*Plus for compiling and creating a package body, you type forward slash (/) as follows:

```
44   return ln_net_value;
45   EXCEPTION
46   WHEN no_data_found THEN
47   DBMS_OUTPUT.PUT_LINE( SQLERRM );
48     END get_net_value_by_customer;
49
50   END order_mgmt;
51   /

Package body created.

SQL> _
```

Compiling a Package using Oracle Developer

From PL/SQL Developer, you click the **Run Script** button to compile the package body.



Compiling a Package using Files on SQL*Plus

If you manage package specification & body using separate files, you can compile separately using the following commands:

SQL> @path_to_packge_body_file

SQL > @path_to_package_body_file

2 /

Calling Function and Procedure from Package

```
DECLARE
   v_sum_sal    emp.sal%TYPE;
BEGIN
   ...
   v_sum_sal := emp_info.sum_dept_sal(10);
   ...
   emp_info.delete_record(1234567);
   ...
END;
/
```

Or

```
SQL> exec emp_info.insert_record(...,...)
```

Lesson 139: PL/SQL Triggers

PL/SQL Triggers

- What is an Oracle Triggers?
- Oracle Trigger Usage
- Oracle Trigger Syntax
- Oracle Trigger Header
- Oracle Trigger Body
- Create or Replace a Trigger

What is an Oracle Triggers?

- A trigger is a named PL/SQL block stored in the Oracle Database and executed automatically when a triggering event takes place. The event can be any of the following:
- A data manipulation language (DML) statement executed against a table e.g., INSERT, UPDATE, or DELETE.
- For example, if you define a trigger that fires before an INSERT statement on the customers table, the trigger will fire once before a new row is inserted into the customers table.
- A data definition language (DDL) statement executes e.g., CREATE or ALTER statement. These triggers are often used for auditing purposes to record changes of the schema.
- A system event such as startup or shutdown of the Oracle Database.
- A user event such as login or logout.
- The act of executing a trigger is also known as firing a trigger. We say that the trigger is fired.

Oracle Trigger Usage

Oracle triggers are useful in many cases such as the following:

- Enforcing complex business rules that cannot be established using integrity constraint such as UNIQUE, NOT NULL, and CHECK.
- Preventing invalid transactions.
- Gathering statistical information on table accesses.
- Generating value automatically for derived columns.

- Auditing sensitive data.

Oracle Trigger Syntax

- Following is its syntax

```
CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE | AFTER } triggering_event ON table_name
[FOR EACH ROW]
[FOLLOWS | PRECEDES another_trigger]
[ENABLE / DISABLE ]
[WHEN condition]
DECLARE
    declaration statements
BEGIN
    executable statements
EXCEPTION
    exception_handling statements
END;
/
```

Oracle Trigger Header

```
CREATE [OR REPLACE] TRIGGER trigger_name
  {BEFORE | AFTER } triggering_event ON table_name
  [FOR EACH ROW]
  [FOLLOWS | PRECEDES another_trigger]
  [ENABLE / DISABLE ]
  [WHEN condition]
```

Oracle Trigger Body

```
DECLARE
   declaration statements
BEGIN
   executable statements
EXCEPTION
   exception_handling statements
END;
```

Create or Replace a Trigger

- The CREATE keyword specifies that you are creating a new trigger. The OR REPLACE keywords are optional. They are used to modify an existing trigger.
- Even though the OR REPLACE keywords are optional, they appear with the CREATE keyword in most cases.

Lesson 140: Trigger Examples

- Trigger Example-1
- Execution of Trigger Example-1
- Trigger Example-2
- Trigger & Function Example-3

Trigger Example-1

Display difference of salary when salary of employee is changed

```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON EMP
FOR EACH ROW
WHEN (NEW.EMPNO > 0)
DECLARE
   sal_diff number;
BEGIN
   sal_diff := :NEW.sal  - :OLD.sal;
   dbms_output.put_line('Old salary: ' || :OLD.sal);
   dbms_output.put_line('New salary: ' || :NEW.sal);
   dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/
```

Execution of Trigger Example-1

Display difference of salary when salary of employee is changed

```
SQL> CREATE OR REPLACE TRIGGER display_salary_changes
  2  BEFORE DELETE OR INSERT OR UPDATE ON EMP
  3  FOR EACH ROW
  4  WHEN (NEW.EMPNO > 0)
  5  DECLARE
  6     sal_diff number;
  7  BEGIN
  8     sal_diff := :NEW.sal  - :OLD.sal;
  9     dbms_output.put_line('Old salary: ' || :OLD.sal);
 10     dbms_output.put_line('New salary: ' || :NEW.sal);
 11     dbms_output.put_line('Salary difference: ' || sal_diff);
 12  END;
 13  /

Trigger created.

SQL> set serveroutput on
SQL> update emp
  2  set sal=777
  3  where ename='SMITH';
Old salary: 800
New salary: 777
Salary difference: -23

1 row updated.
```

Trigger Example-2

Keeping log of old data into a log table

```
CREATE OR REPLACE TRIGGER del_emp
BEFORE DELETE ON emp
FOR EACH ROW
DECLARE
   x number(2);
BEGIN
     INSERT INTO emp_audit
            VALUES(:NEW.empno, USER, sysdate);
END;
/

--Other forms can be BEFORE INSERT OR UPDATE
--OF salary ON EMP
--Need to create table emp_audit
--select * from emp_audit;
```

Trigger & Function Example-3

```
create table emp1
as select * from emp;
```

```
alter table emp1
add bonus number(6,2);
CREATE OR REPLACE  FUNCTION "BONUS_CALC"
                    (p_sal IN NUMBER) RETURN NUMBER IS
BEGIN
    return(p_sal/600);
   dbms_output.put_line(' ***** THE RECORD HAS SUCCESSFULLY BEEN INSERTED *****');
END;
/


CREATE OR REPLACE TRIGGER "UPDATE_BONUS" before INSERT OR UPDATE  OF SAL ON "EMP1" FOR EACH ROW
BEGIN
    --AFTER option gives error of mutating
    --select Bonus_Calc(:New.sal) into :New.bonus from dual;
    --Or use the following asignment statement
    :New.bonus := Bonus_Calc(:New.sal);

   dbms_output.put_line(' ***** THE RECORD HAS SUCCESSFULLY BEEN INSERTED *****');
end;
/

Update emp1
set sal=1600
where ename in ('ALLEN');
```

Lesson 141: Trigger Exercises

- Exercise-1 (Off day DML Restriction)
- Exercise-2 (for Data Entry)
- Trigger's Disable/ Enable
- Trigger's Disable/ Enable using SQL Developer
- Exercise-3 (Mutating Error)
- Fired Triggers
- Updating & Deleting on Table with specific Columns

Exercise-1 (Off day DML Restriction)

This is a statement trigger on the EMP table, and it fires before an INSERT, UPDATE, or DELETE statement is issued. First, the trigger determines the day of the week. If the day is Saturday or Sunday, an error message is generated.

```
SQL> CREATE OR REPLACE TRIGGER EMP_ModifyIssue
  2  BEFORE INSERT OR UPDATE OR DELETE ON EMP
  3  DECLARE
  4     v_day VARCHAR2(10);
  5  BEGIN
  6     v_day := RTRIM(TO_CHAR(SYSDATE, 'DAY'));
  7     IF v_day LIKE ('S%') THEN
  8   RAISE_APPLICATION_ERROR (-20000, 'A table cannot be modified during off hours');
  9  END IF;
 10  END;
 11  /

Trigger created.

SQL>
SQL> update emp
  2  set sal=700
  3  where ename='SMITH';
update emp
       *
ERROR at line 1:
ORA-20000: A table cannot be modified during off hours
ORA-06512: at "SCOTT.EMP_MODIFYISSUE", line 6
ORA-04088: error during execution of trigger 'SCOTT.EMP_MODIFYISSUE'
```

Exercise-2 (for Data Entry)

```
CREATE OR REPLACE TRIGGER Derived
BEFORE INSERT OR UPDATE OF Ename ON Emp

/* Before updating the ENAME field, derive the values for
   the UPPERNAME and SOUNDEXNAME fields. Restrict users
   from updating these fields directly: */
FOR EACH ROW
BEGIN
  :NEW.Uppername := UPPER(:NEW.Ename);
  :NEW.Soundexname := SOUNDEX(:NEW.Ename);
END;
/
```

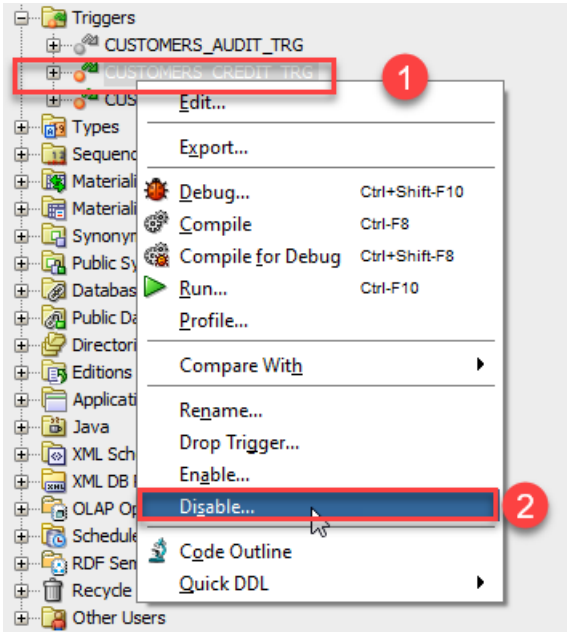
Trigger's Disable/ Enable

Sometimes, you may want to disable a trigger for testing and troubleshooting purposes. To disable a trigger, you use the ALTER TRIGGER DISABLE statement:

ALTER TRIGGER trigger_name DISABLE;

ALTER TABLE table_name DISABLE ALL TRIGGERS;

Trigger's Disable/ Enable using SQL Developer



Exercise-3 (Mutating Error)

```
SQL> CREATE OR REPLACE TRIGGER Update_EmpDeptno
  2  BEFORE INSERT OR UPDATE ON EMP
  3  FOR EACH ROW
  4  DECLARE
  5  v_cnt Number(3);
  6  BEGIN
  7  IF :NEW.deptno IS NOT NULL THEN
  8  BEGIN
  9  SELECT count(*)
 10  INTO v_cnt
 11  FROM EMP
 12  WHERE deptno = :NEW.deptno;
 13  EXCEPTION
 14  WHEN NO_DATA_FOUND THEN
 15  RAISE_APPLICATION_ERROR
 16  (-20001, 'This is not a valid deptno');
 17  END;
 18  END IF;
 19  END;
 20  /

Trigger created.

SQL> Update emp
  2  set deptno = 11
  3  where deptno=10;
Update emp
       *
ERROR at line 1:
ORA-04091: table SCOTT.EMP is mutating, trigger/function may not see it
ORA-06512: at "SCOTT.UPDATE_EMPDEPTNO", line 6
ORA-04088: error during execution of trigger 'SCOTT.UPDATE_EMPDEPTNO'
```

Exercise-3 (Resolving Mutating Error)

```
CREATE OR REPLACE TRIGGER Update_EmpDeptno
BEFORE INSERT OR UPDATE ON EMP
FOR EACH ROW
DECLARE
v_cnt Number(3);
BEGIN
   IF :NEW.deptno IS NOT NULL THEN
   BEGIN
        SELECT count(*)
        INTO v_cnt
        FROM dept
        WHERE deptno = :NEW.deptno;
        DBMS_OUTPUT.PUT_LINE(:NEW.deptno);
   EXCEPTION
   WHEN NO_DATA_FOUND THEN
   RAISE_APPLICATION_ERROR (-20001, 'This is not a valid deptno');
   END;
   END IF;
END;
/
```

Fired Triggers

Need to see which trigger names are fired on commands such as

```
SQL> update emp
  2  set deptno=11
  3  where deptno=10;
11
Old salary:
New salary:
Salary difference:
11
Old salary: 4000
New salary: 4000
Salary difference: 0
11
Old salary: 2450
New salary: 2450
Salary difference: 0
```

```
select distinct name
from user_source
where type='TRIGGER';

drop trigger DEL_EMP;
drop trigger UPDATE_EMPDEPTNO;
```

Updating & Deleting on Table with specific Columns

```
CREATE OR REPLACE TRIGGER Updating_EMP
  BEFORE
    INSERT OR
    UPDATE OF sal, deptno OR
    DELETE
  ON EMP
BEGIN
  CASE
    WHEN INSERTING THEN
      DBMS_OUTPUT.PUT_LINE('Inserting');
    WHEN UPDATING('sal') THEN
      DBMS_OUTPUT.PUT_LINE('Updating sal');
    WHEN UPDATING('deptno') THEN
      DBMS_OUTPUT.PUT_LINE('Updating department ID');
    WHEN DELETING THEN
      DBMS_OUTPUT.PUT_LINE('Deleting');
  END CASE;
END;
/

update emp
set sal=2000
where ename='SMITH';
```

Lesson 142: Cursor Programming-1

Cursor Programming-1

- What is a Cursor?
- Implicit Cursor
- Execution Cycle of Explicit Cursor
- Cursor's usage
- Explicit Cursor Attributes

What is a Cursor?

- A cursor is a pointer that points to a result of a query.
- It is mainly used with tables which return more than one row.
- It is handled implicitly and explicitly.

Implicit Cursor

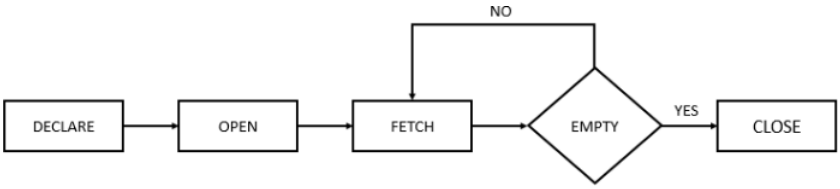
- Whenever Oracle executes an SQL statement such as SELECT INTO, INSERT, UPDATE, and DELETE, it automatically creates an implicit cursor.
- Oracle internally manages the whole execution cycle of implicit cursors and reveals only the cursor's information and statuses such as SQL%ROWCOUNT, SQL%ISOPEN, SQL%FOUND, and SQL%NOTFOUND.

- The implicit cursor is not elegant when the query returns zero or multiple rows which cause NO_DATA_FOUND or TOO_MANY_ROWS exception respectively.

Explicit Cursor

- An explicit cursor is an SELECT statement declared explicitly in the declaration section of the current block or a package specification.
- For an explicit cursor, you have control over its execution cycle from OPEN, FETCH, and CLOSE.
- Oracle defines an execution cycle that executes an SQL statement and associates a cursor with it.

Execution Cycle of Explicit Cursor



Cursor's usage

- **Declare a cursor**

  CURSOR cursor_name IS query;

- **Open a cursor**

  OPEN cursor_name;

- **Fetch from a cursor**

The FETCH statement places the contents of the current row into var.

FETCH cursor_name INTO variable_list; To retrieve all rows in a result set, you need to fetch each row till the last one.

Closing a cursor

CLOSE cursor_name;

Explicit Cursor Attributes

- %ISOPEN
- %FOUND
- %NOTFOUND
- %ROWCOUNT

Lesson 143: Cursor Programming-2

Cursor Programming-2

- Explicit Cursor Attributes
- Implicit Cursor - Record Type
- ROWTYPE Example - 1
- ROWCOUNT Example
- How to Declare Cursor?
- %ISOPEN, %FOUND, FETCH Example

Explicit Cursor Attributes

- %ISOPEN
- %FOUND
- %NOTFOUND
- %ROWCOUNT

Implicit Cursor – Record Type

- A record is a composite data structure, which means that it is composed of one or more elements.
- Records are very much like a row of a database table, but each element of the record does not stand on its own. PL/SQL supports three kinds of records: table-based, cursor-based, and programmer-defined.

  v_EmpRec EMP%TYPE;

CURSOR c_emp IS

SELECT empno, ename, job

FROM emp

WHERE deptno = 20;

vr_emp c_emp%ROWTYPE;

ROWTYPE Example - 1

```
DECLARE
   v_emp  EMP%ROWTYPE;
BEGIN
   SELECT *
   INTO v_emp
   FROM EMP
   WHERE ename='SMITH';
   DBMS_OUTPUT.PUT_LINE('Records columns are '
   ||v_emp.empno||','||v_emp.ename);
EXCEPTION
WHEN NO_DATA_FOUND THEN
     DBMS_OUTPUT.PUT_LINE('The Employee '
     ||'is not in the database');
END;
```

ROWTYPE Example - 2

```
/
```

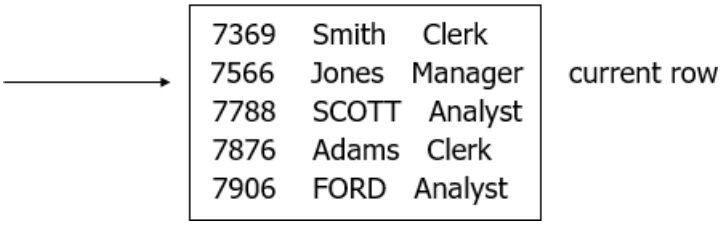
```
DECLARE
CURSOR c_emp IS
   SELECT *
   FROM EMP;
   vr_emp c_emp%ROWTYPE;
BEGIN
OPEN c_emp;
LOOP
FETCH c_emp INTO vr_emp;
EXIT WHEN c_emp%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(vr_emp.empno||
' '||vr_emp.ename||' '||vr_emp.sal);
END LOOP;
END;
/
```

ROWCOUNT Example

```
DECLARE
    v_job EMP.job%type;
BEGIN
    SELECT job
    INTO v_job
    FROM EMP
    WHERE empno = 7698;
  IF SQL%ROWCOUNT = 1 THEN
    DBMS_OUTPUT.PUT_LINE(v_job ||' has a '||
    'Empno of 7698');
  ELSIF SQL%ROWCOUNT = 0 THEN
    DBMS_OUTPUT.PUT_LINE('The Empno 7698 is '||
    ' not in the database');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Stop... ');
  END IF;
END;
/
```

How to Declare Cursor?

```
CURSOR c_emp  IS
SELECT  empno, ename, job
FROM    emp
WHERE  deptno = 20;
```



%ISOPEN, %FOUND, FETCH Example

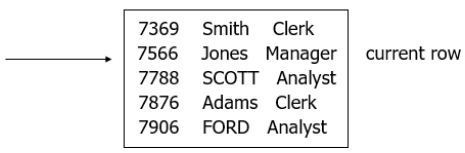
```
DECLARE
        v_empno EMP.Empno%TYPE;
        CURSOR c_EMP IS
        SELECT Empno
          FROM EMP
          WHERE Empno > 1000;
BEGIN
        OPEN c_EMP;
        LOOP
        FETCH c_EMP INTO v_empno;
        IF c_EMP%FOUND THEN
           DBMS_OUTPUT.PUT_LINE('Just FETCHED row '
           ||TO_CHAR(c_EMP%ROWCOUNT)||' EMP ID: '||v_empno);
        ELSE
           EXIT;
        END IF;
        END LOOP;
        CLOSE c_EMP;
        EXCEPTION
        WHEN OTHERS THEN
           IF c_EMP%ISOPEN
        THEN
           CLOSE c_EMP;
        END IF;
END;
/
```

Lesson 144: Cursor Programming-3

- How to Declare Cursor?
- Sample Program using Cursor
- Cursor for DML Insert
- Cursor with Parameters
- Cursor Program-1
- Cursor Program-1 (Solution)

How to Declare Cursor?

```
CURSOR c_emp  IS
SELECT  empno, ename, job
FROM    emp
WHERE  deptno = 20;
```



Sample Program using Cursor

```
DECLARE

    CURSOR c_emp IS
    SELECT  empno, ename, job
    FROM    emp
    WHERE  deptno = 20;

BEGIN

     FOR v_c IN c_emp  LOOP
      ...
       DBMS_OUTPUT.PUT_LINE(v_c.ename);
      ...
     END LOOP;
   ...
end;
/
```

Cursor for DML Insert

```
DECLARE
  CURSOR c_emp IS
  SELECT empno, ename, job
  FROM EMP
  WHERE empno > 1100;
BEGIN
  FOR r_emp IN c_emp
  LOOP
    INSERT INTO table_log
    VALUES(r_emp.ename);
  END LOOP;
END;
/
```

Cursor with Parameters

```
DECLARE
  CURSOR c_emp (p_job IN EMP.job%TYPE) IS
  SELECT *
  FROM EMP
  WHERE job = p_job;
BEGIN
  FOR r_emp IN c_emp('CLERK')
  LOOP
        DBMS_OUTPUT.PUT_LINE(r_emp.ename||'--'||r_emp.job);
  END LOOP;
END;
/
```

Cursor Program-1

Given a table with first three columns are composite keys. Write a PL/SQL program using cursor to update cat_template1 with LONG column cat template. Assuming this tables contains more than 40,000 records.

```
SQL> desc patient_visit
 Name                                      Null?    Type
 ----------------------------------------- -------- --------------
 PATIENT_ID                                NOT NULL CHAR(7)
 DEPEND_SNO                                NOT NULL VARCHAR2(2)
 PATIENT_VISITNO                           NOT NULL NUMBER(4)
 EMP_ID                                             CHAR(7)
 DOCTOR_SNO                                         NUMBER(3)
 VISIT_DATE                                         DATE
 VISIT_NOTE                                         VARCHAR2(50)
 CAT_TEMPLATE                                       LONG
 MEDICATION                                         VARCHAR2(500)
 CHECKUP_STATUS                                     CHAR(1)
 SUB_CAT_ID                                         NUMBER(6)
 CAT_TEMPLATE1                                       VARCHAR2(3000)
```

Cursor Program-1 (Solution)

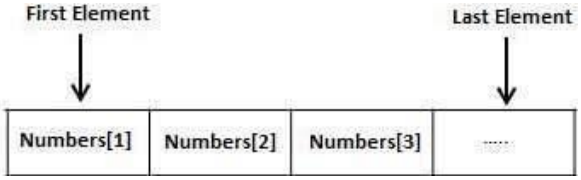
```
declare
  v_count number(7);
cursor c1 is
   select  PATIENT_ID,DEPEND_SNO,PATIENT_VISITNO,CAT_TEMPLATE
    from patient_visit;
begin
  v_count := 0;
  for v_c in c1 LOOP
    update  patient_visit
    set  CAT_TEMPLATE1   = v_c.CAT_TEMPLATE
    where PATIENT_ID = v_c.PATIENT_ID
        and DEPEND_SNO = v_c.DEPEND_SNO
        and PATIENT_VISITNO = v_c.PATIENT_VISITNO;
    If v_count = 300 Then
      v_count := 0;
      --DBMS_OUTPUT.PUT_LINE('commit test');
    else
      v_count := v_count + 1;
    end if;
  end LOOP;
  commit;
end;
/
```

Lesson 145: Programming using an Array

- What is an Array?
- Declaration of an Array
- Basic Example of an Array
- Associative Array
- Associative Array - An Example
- Use of Array with an Exception
- Array Null and Empty Collection

What is an Array?

All VARRAYS consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.



https://www.tutorialspoint.com/plsql/plsql_collections.htm

Declaration of an Array

A varray type is created with the **CREATE TYPE** statement. You must specify the maximum size and the type of elements stored in the varray.

CREATE Or REPLACE TYPE namearray AS VARRAY(3) OF VARCHAR2(10);

/

Basic Example of an Array

```
DECLARE
   CURSOR c_emp is
   SELECT  ename FROM EMP;
   TYPE c_list is varray (20) of EMP.ename%type;
   ename_list c_list := c_list();
   counter integer :=0;
BEGIN
   FOR n IN c_emp LOOP
      counter := counter + 1;
      ename_list.extend;
      ename_list(counter)   := n.ename;
      dbms_output.put_line('Employee('||counter ||'):'
      ||ename_list(counter));
   END LOOP;
END;
/
```

Associative Array

An index-by table is a set of key-value pairs. Each key is unique and is used to locate the corresponding value. The key can be either an integer or a string.
TYPE type_name IS TABLE OF element_type [NOT NULL] INDEX BY subscript_type;

table_name type_name;

Associative Array – An Example

```
DECLARE
   TYPE salary IS TABLE OF NUMBER INDEX BY VARCHAR2(20);
   salary_list salary;
   name   VARCHAR2(20);
BEGIN
   -- adding elements to the table
   salary_list('Sarah') := 62000;
   salary_list('Jamil') := 75000;
   salary_list('Wisal') := 100000;
   salary_list('Umar') := 78000;

   -- printing the table
   name := salary_list.FIRST;
   WHILE name IS NOT null LOOP
      dbms_output.put_line
      ('Salary of ' || name || ' is ' || TO_CHAR(salary_list(name)));
      name := salary_list.NEXT(name);
   END LOOP;
END;
/
```

Use of Array with an Exception

```
SQL> DECLARE
  2    CURSOR name_cur IS
  3    SELECT ename
  4    FROM EMP
  5    WHERE rownum <= 10;
  6    TYPE ename_type IS TABLE OF EMP.ename%TYPE
  7    INDEX BY BINARY_INTEGER;
  8    ename_tab ename_type;
  9    v_counter INTEGER := 0;
 10  BEGIN
 11    FOR name_rec IN name_cur LOOP
 12  v_counter := v_counter + 1;
 13  ename_tab(v_counter) := name_rec.ename;
 14  DBMS_OUTPUT.PUT_LINE ('ename('||v_counter||'): '||
 15  ename_tab(v_counter));
 16    END LOOP;
 17    DBMS_OUTPUT.PUT_LINE ('ename(11): '|| ename_tab(11));
 18  EXCEPTION
 19    WHEN NO_DATA_FOUND THEN
 20    DBMS_OUTPUT.PUT_LINE(SQLERRM);
 21  END;
 22  /
ename(1):
ename(2): ABC
ename(3): XYZ
ename(4): SMITH
ename(5): ALLEN
ename(6): WARD
ename(7): JONES
ename(8): MARTIN
ename(9): BLAKE
ename(10): CLARK
ORA-01403: no data found
```

Array Null and Empty Collection

```
DECLARE
   TYPE integer_type IS TABLE OF INTEGER;
   integer_tab integer_type;

   v_counter integer := 1;
BEGIN
   DBMS_OUTPUT.PUT_LINE (integer_tab(v_counter));
END;

DECLARE
*
ERROR at line 1:
ORA-06531: Reference to uninitialized collection
ORA-06512: at line 7
```

```
DECLARE
   TYPE integer_type IS TABLE OF INTEGER;
   integer_tab integer_type := integer_type();

   v_counter integer := 1;
BEGIN
   DBMS_OUTPUT.PUT_LINE (integer_tab(v_counter));
END;

DECLARE
*
ERROR at line 1:
ORA-06533: Subscript beyond count
ORA-06512: at line 7
```

Lesson 146: Examples with Advance Topics

- Use of Records Arrays
- Use of PRAGMA Exception
- PL/SQL Dynamic SQL
- PL/SQL Dynamic SQL - Example-1
- PL/SQL Dynamic SQL - Example-2

Use of Records Arrays

```
Declare
Cursor c1 is
  select sal
  from emp;
TYPE sal_type IS VARRAY(20) OF Integer;
v_col sal_TYPE := sal_type();
k number := 0;
Begin
  for i in C1 LOOP
    k := k+1;
    v_col.extend; --collection method
    v_col(k) := i.sal;
  end loop;
  for j in 1..4 LOOP
    DBMS_OUTPUT.PUT_LINE('sal is col('||j||') '
    ||v_col(j));
  end loop;
End;
```

Use of PRAGMA Exception

```
CREATE OR REPLACE TRIGGER Employee_permit_changes
  BEFORE INSERT OR DELETE OR UPDATE ON EMP
DECLARE
  Dummy              INTEGER;
  Not_on_weekends    EXCEPTION;
  Nonworking_hours   EXCEPTION;
  PRAGMA EXCEPTION_INIT (Not_on_weekends, -4097);
  PRAGMA EXCEPTION_INIT (Nonworking_hours, -4099);
BEGIN
   -- Check for weekends:
   IF (TO_CHAR(Sysdate, 'DAY') = 'SAT' OR
     TO_CHAR(Sysdate, 'DAY') = 'SUN') THEN
       RAISE Not_on_weekends;
   END IF;
  -- Check for work hours (8am to 6pm):
  IF (TO_CHAR(Sysdate, 'HH24') < 8 OR
    TO_CHAR(Sysdate, 'HH24') > 18) THEN
      RAISE Nonworking_hours;
  END IF;
EXCEPTION
  WHEN Not_on_weekends THEN
    Raise_application_error(-20324,'Might not change '
      ||'employee table during the weekend');
  WHEN Nonworking_hours THEN
    Raise_application_error(-20326,'Might not change '
     ||'emp table during Nonworking hours');
END;
/
```

PL/SQL Dynamic SQL

- Dynamic SQL is a programming methodology for generating and running SQL statements at run time.
- It is useful when writing general-purpose and flexible programs like ad hoc query systems, when writing programs that must run database definition language (DDL) statements, or when you do not know at compilation time the full text of a SQL statement or the number or data types of its input and output variables.

PL/SQL Dynamic SQL – Example-1

Creating a table during run-time.

```
BEGIN
   execute immediate 'create table MyTable('
                                          ||'ID CHAR(3),'
                                          ||'Name VARCHAR2(50))';
END;
/
```

```
Note: grant create table to <user> is required
```

PL/SQL Dynamic SQL - Example-2

Set to Null a column at run time

```
CREATE TABLE EMP_temp AS SELECT * FROM EMP;
/

DECLARE
  -- Set to NULL automatically at run time
  a_null  CHAR(1);
BEGIN
  EXECUTE IMMEDIATE 'UPDATE EMP_TEMP SET sal = :x'
    USING a_null;
END;
/
```

Lesson 147: Concepts of Oracle Instance-1

Concepts of Oracle Instance-1

- Introduction
- Configuration Option
- Instance Structure
- Database Instance Framework

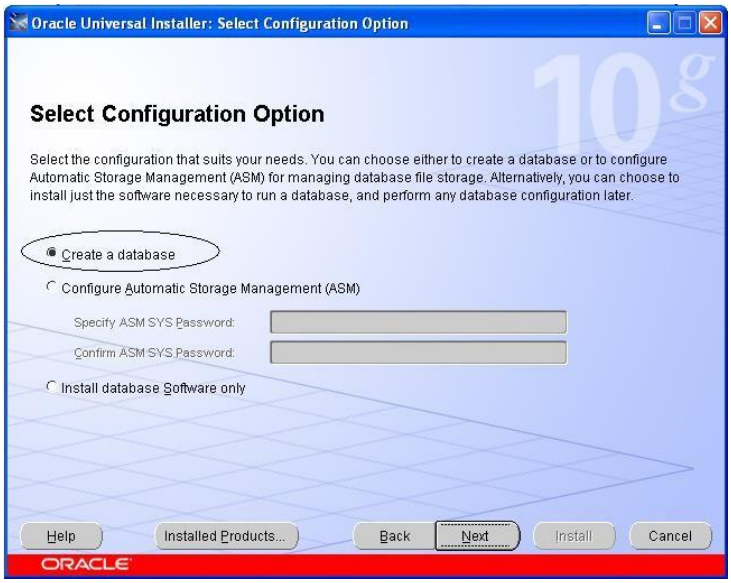
Introduction

- A database instance is a set of memory structures that manage database files.
- A Database is a set of physical files on disk created by the CREATE DATABASE statement.
- The instance manages its associated data and serves the users of the database.

- Every running Oracle database is associated with at least one Oracle database instance. Because an instance exists in memory and a database exists on disk, an instance can exist without a database and a database can exist without an instance.

Configuration Option

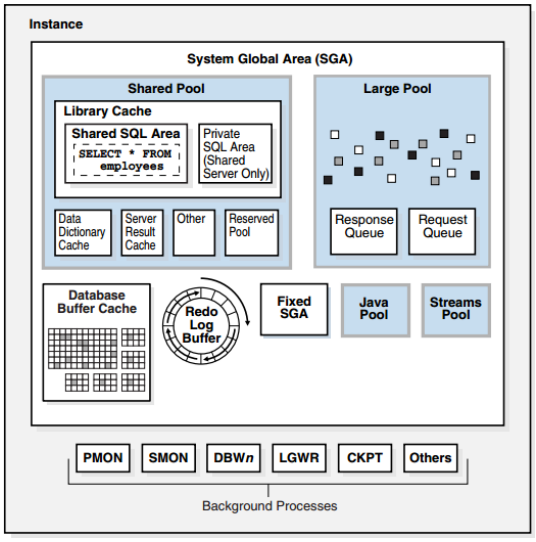
During installation, oracle has an option to install instance, database or both



Instance Structure

- When an instance is started, Oracle Database allocates a memory area called the system global area (SGA) and starts one or more background processes. The SGA serves various purposes, including the following:
- Maintaining internal data structures that are accessed by many processes and threads concurrently
- Caching data blocks read from disk
- Buffering redo data before writing it to the online redo log files
- Storing SQL execution plans The SGA is shared by the Oracle processes, which include server processes and background processes, running on a single computer. The way in which Oracle processes are associated with the SGA varies according to operating system
- A database instance includes background processes. Server processes, and the process memory allocated in these processes, also exist in the instance. The instance continues to function when server processes terminate.

Database Instance Framework



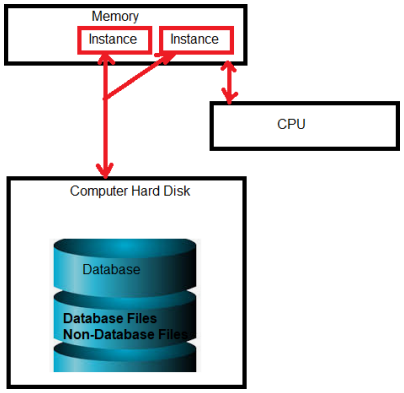
Lesson 148: Concepts of Oracle Instance-2

- Database Instance Configuration
- Database Instance Configuration
- Database Single & RAC Instance Configuration

Database Instance Configuration

You can run Oracle Database in either of the following mutually exclusive configurations:

- Single-instance configuration A one-to-one relationship exists between the database and an instance.
- Oracle Real Application Clusters (Oracle RAC) configuration A one-to-many relationship exists between the database and instances.

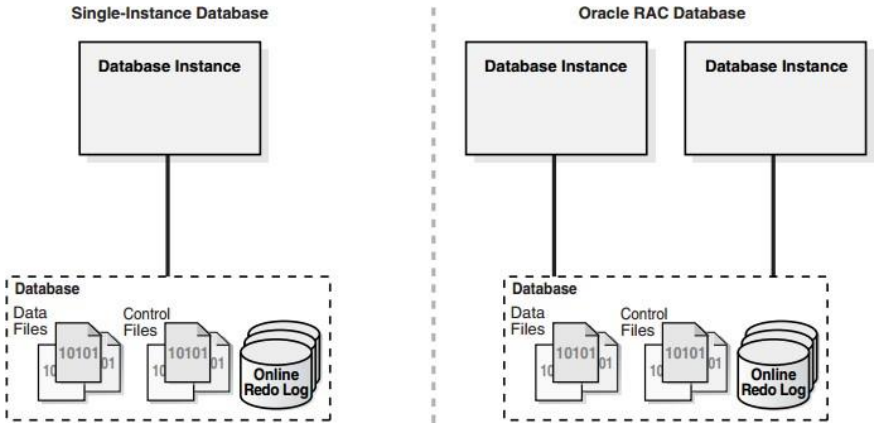


Database files can be control files, data files, Temp files etc

Non Database files are Log files

Files include Physical and Logical concepts

Database Single & RAC Instance Configuration



Real Application Cluster RAC

Whether in a single-instance or Oracle RAC configuration, a database instance is associated with only one database at a time. You can start a database instance and mount (associate the instance with) one database, but not mount two databases simultaneously with the same instance

Multiple instances can run concurrently on the same computer, each accessing its own database. For example, a computer can host two distinct databases: prod1and prod2.

One database instance manages prod1, while a separate instance manages prod2

Lesson 149: Concepts of Oracle Instance-3

- Database Instance Configuration
- Duration of an Instance
- Instance Startup
- Instance Shutdown & Startup
- Instance & SID
- Oracle Home & SID at Windows OS
- Display Instance Name using Data Dictionary View

Database Instance Configuration

Duration of an Instance

- An instance begins when it is created with the STARTUP command and ends when it is terminated. During this period, an instance can associate itself with one and only one database.
- Furthermore, the instance can mount a database only once, close it only once, and open it only once. After a database has been closed or shut down, you must start a different instance to mount and open this database.

Instance Startup

- Login with user sys / as sysdba
- After shutdown database, run the following:

```
SQL> STARTUP
ORACLE instance started.

Total System Global Area  468729856 bytes
Fixed Size                  1333556 bytes
Variable Size             440403660 bytes
Database Buffers           16777216 bytes
Redo Buffers               10215424 bytes
Database mounted.
Database opened.
```

This query shows the time that the current instance was started:

```
SQL> SELECT
  2  TO_CHAR(STARTUP_TIME,'MON-DD-RR HH24:MI:SS')
  3  AS "Inst Start Time" FROM V$INSTANCE;

Inst Start Time
------------------
SEP-30-20 20:56:54
```

Instance Shutdown & Startup

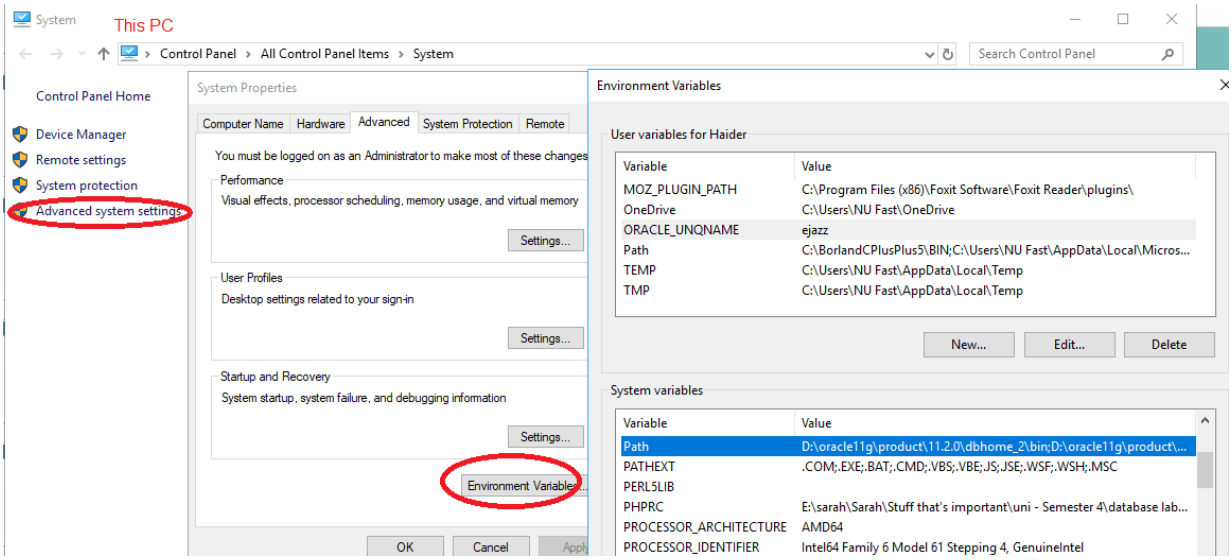
This query shows the time that the current instance was started:

```
SQL>
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup
ORACLE instance started.

Total System Global Area 1071333376 bytes
Fixed Size                  1375792 bytes
Variable Size             696254928 bytes
Database Buffers          369098752 bytes
Redo Buffers                4603904 bytes
Database mounted.
Database opened.
SQL> SELECT
  2  TO_CHAR(STARTUP_TIME,'MON-DD-RR HH24:MI:SS')
  3  AS "Inst Start Time" FROM V$INSTANCE;

Inst Start Time
------------------
OCT-01-20 08:48:52
```

Instance & SID

- The system identifier (SID)is a unique name for an Oracle database instance on a specific host. On UNIX and Linux, Oracle Database uses the SID and Oracle home values to create a key to shared memory.
- Also, the SID is used by default to locate the parameter file, which is used to locate relevant files such as the database control files.
- On most platforms, the ORACLE_SID environment variable sets the SID, whereas the ORACLE_HOME variable sets the Oracle home.
- When connecting to an instance, clients can specify the SID in an Oracle Net connection or use a net service name.
- Oracle Database converts a service name into an ORACLE_HOME and ORACLE_SID.

Oracle Home & SID at Windows OS



Display Instance Name using Data Dictionary View

```
SQL> SELECT TO_CHAR(STARTUP_TIME,'MON-DD-RR HH24:MI:SS') AS "Inst Start Time",
  2  Instance_name , substr(Host_name,1,20)
  3  FROM V$INSTANCE;

Inst Start Time    INSTANCE_NAME    SUBSTR(HOST_NAME,1,2
------------------ ---------------- --------------------
OCT-01-20 08:48:52 orcl             HAIDER
```

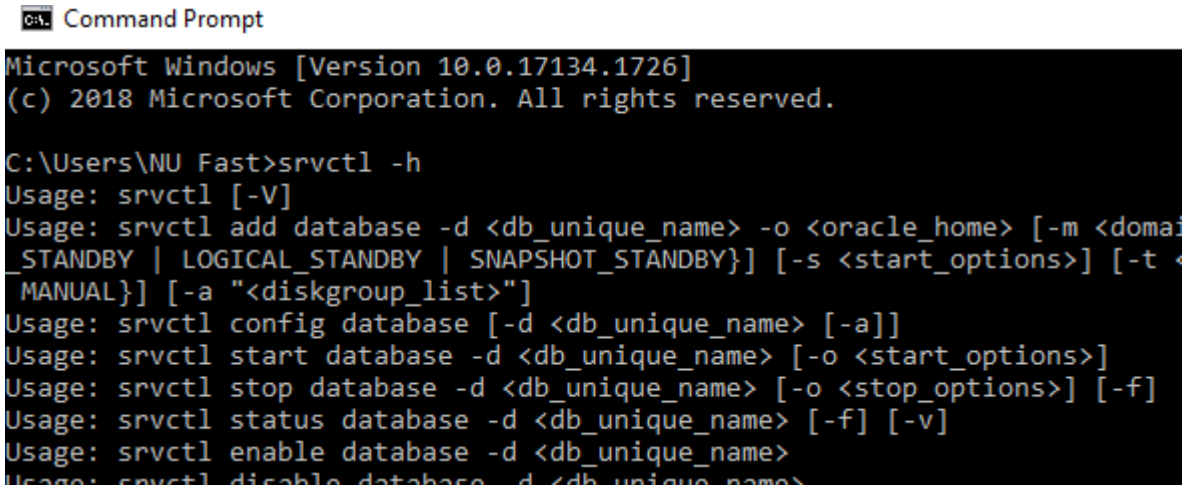
Lesson 150: Instance Startup & Shutdown-1

- Overview of Instance and Database Startup
- SVRCTL Utility Help
- Example - SVRCTL command
- Progress from Shutdown & Startup OPEN
- Instance Startup
- Instance Startup MOUNT
- Instance Startup NOMOUNT

Overview of Instance and Database Startup

- A database instance provides user access to a database.
- In a typical use case, you manually start an instance and then mount and open the database, making it available for users. You can use the SQL*Plus STARTUP command, Oracle Enterprise Manager (Enterprise Manager), or the SRVCTL utility to perform these steps.

SVRCTL Utility Help

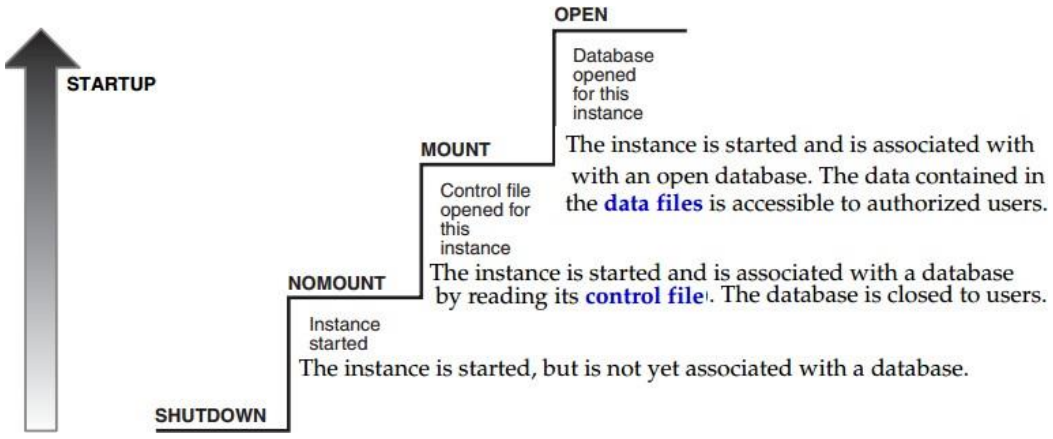


Example - SVRCTL command

Syntax to add a named service to a database with preferred instances in list one and available instances in list two, using basic failover for the available instances:

```
srvctl add  service -d crm -s sales -r crm01,crm02 -a crm03
```

Progress from Shutdown & Startup OPEN



At this stage, no database is associated with the instance. Scenarios that require a

NOMOUNT state include database creation and certain backup and recovery operations.

Instance Startup

- Login with user sys / as sysdba
- After shutdown database, run the following:

```
SQL> STARTUP
ORACLE instance started.

Total System Global Area  468729856 bytes
Fixed Size                  1333556 bytes
Variable Size             440403660 bytes
Database Buffers           16777216 bytes
Redo Buffers               10215424 bytes
Database mounted.
Database opened.
```

Instance Startup

This query shows the time that the current instance was started:

```
SQL> SELECT
  2  TO_CHAR(STARTUP_TIME,'MON-DD-RR HH24:MI:SS')
  3  AS "Inst Start Time" FROM V$INSTANCE;

Inst Start Time
-----------------
SEP-30-20 20:56:54
```

Instance Startup MOUNT

First perform shutdown immediate then startup mount:

```
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
SQL> startup mount
ORACLE instance started.

Total System Global Area 1071333376 bytes
Fixed Size                  1375792 bytes
Variable Size             696254928 bytes
Database Buffers          369098752 bytes
Redo Buffers                4603904 bytes
Database mounted.
```

Instance Startup NOMOUNT

First perform shutdown immediate then startup nomount:

```
SQL> shutdown immediate
ORA-01109: database not open


Database dismounted.
ORACLE instance shut down.
SQL>
SQL> startup nomount
ORACLE instance started.

Total System Global Area 1071333376 bytes
Fixed Size                  1375792 bytes
Variable Size             696254928 bytes
Database Buffers          369098752 bytes
Redo Buffers                4603904 bytes
SQL>
```

Database creation and certain backup & recovery operations are performed.

Lesson 151:Instance Startup & Shutdown-2

- Connection with Administrator Privileges
- Instance Started to perform steps
- How Database is Mounted?
- How Database is opened?
- Read-only Opened Database
- Progress from Shutdown & Startup OPEN

Connection with Administrator Privileges

- Database startup and shutdown are powerful administrative options that are restricted to users who connect to Oracle Database with administrator privileges.
- Normal users do not have control over the current status of an Oracle database.
- Depending on the operating system, one of the following conditions establishes administrator privileges for a user:
- The operating system privileges of the user enable him or her to connect using administrator privileges.
- The user is granted the SYSDBA or SYSOPER system privileges and the database uses password files to authenticate database administrators over the network.
- When you connect with the SYSDBA system privilege, you are in the schema owned by SYS. When you connect as SYSOPER, you are in the public schema.

Instance Started to perform steps

- Searches for a server parameter file in a platform-specific default location and, if not found, for a text initialization parameter file(specifying STARTUP with the SPFILE or PFILE parameters overrides the default behavior)
- Reads the parameter file to determine the values of initialization parameters
- Allocates the SGA based on the initialization parameter settings
- Starts the Oracle background processes
- Opens the alert log and trace files and writes all explicit parameter settings to the alert login valid parameter syntax

How Database is Mounted?

- The instance mounts a database to associate the database with this instance. To mount the database, the instance obtains the names of the database control files specified in the CONTROL_FILES initialization parameter and opens the files.
- Oracle Database reads the control files to find the names of the data files and the online redo log files that it will attempt to access when opening the database.
- In a mounted database, the database is closed and accessible only to database administrators.
- Administrators can keep the database closed while completing specific maintenance operations. However, the database is not available for normal operations

How Database is opened?

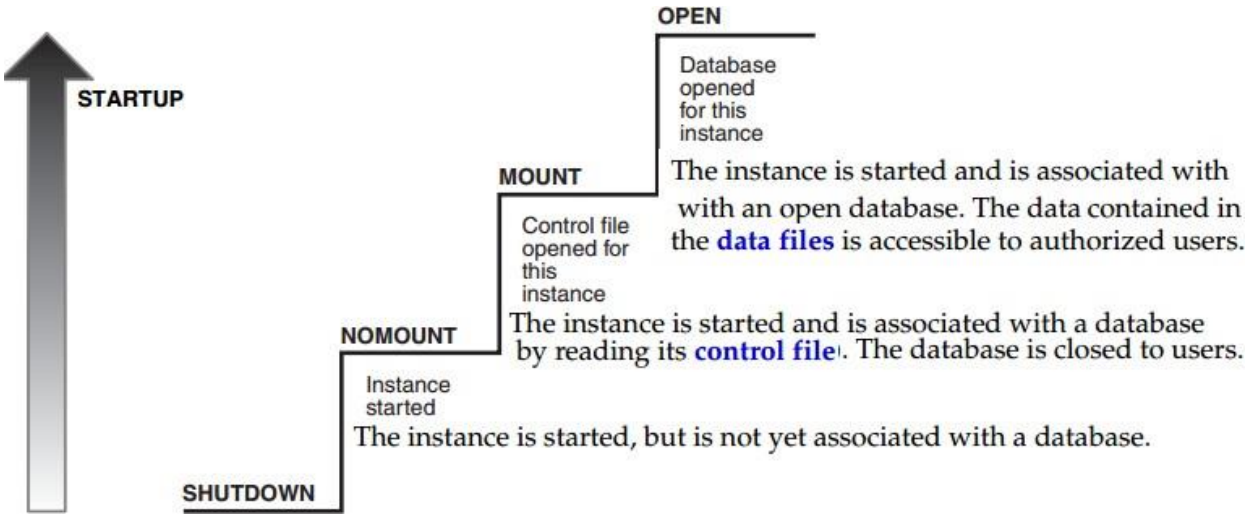
- Opening a mounted database makes it available for normal database operations

- Opens the online data files in tablespaces other than undo tablespaces
- Acquires an undo tablespace
- Opens the online redo log files
- By default, the database opens in read/write mode.

Read-only Opened Database

- Data files can be taken offline and online. However, you cannot take permanent tablespaces offline.
- Offline data files and tablespaces can be recovered.
- The control file remains available
- Temporary tablespaces created with the CREATE TEMPORARY TABLESPACE statement are read/write.
- Writes to operating system audit trails, trace files, and alert logs can continue.

Progress from Shutdown & Startup OPEN



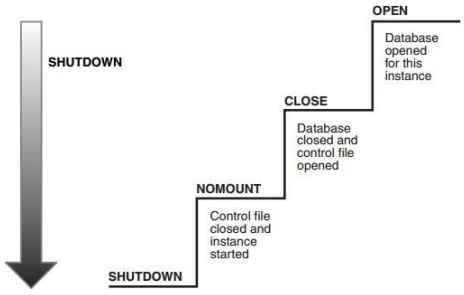
Lesson 152: Instance Startup & Shutdown-3

- Database with Shutdown Instance
- Progress from OPEN to Shutdown
- Steps to perform Database Shutdown
- Shutdown ABORT
- Shutdown IMMEDIATE
- Shutdown TRANSACTIONAL
- Shutdown NORMAL
- Shutdown Modes Summary

Database with Shutdown Instance

- In a typical use case, you manually shut down the database, making it unavailable for users while you perform maintenance or other administrative tasks.
- You can use the SQL*Plus SHUTDOWN command or Enterprise Manager to perform these steps.

Progress from OPEN to Shutdown



Steps to perform Database Shutdown

- Database closed

  The database is mounted, but online data files and redo log files are closed.

- Database unmounted

  The instance is started, but is no longer associated with the control file of the database.

- Database instance shut down

  The database instance is no longer started.

Steps to perform Database Shutdown

Oracle Database does not go through all of the preceding steps in an instance failure or SHUTDOWN ABORT, which immediately terminates the instance.

Shutdown ABORT

Used in emergency situation. Shuting down instance may take substantially longer. SHUTDOWN ABORT does not checkpoint the open data files

Shutdown IMMEDIATE

Fastest way to manage. Active transactions are terminated and uncommitted changes are rolled back.

Shutdown TRANSACTIONAL

This mode prevents users from starting new transactions, but waits for all current transactions to complete before shutting down.

Shutdown NORMAL

This is the default mode of shutdown. The database waits for all connected users

to disconnect before shutting down.

Shutdown Modes Summary

A database administrator with SYSDBA or SYSOPER privileges can shut down the database using the SQL*Plus SHUTDOWN command or Enterprise Manager.

Database Behavior	ABORT	IMMEDIATE	TRANSACTIONAL	NORMAL
Permits new user connections	No	No	No	No
Waits until current sessions end	No	No	No	Yes
Waits until current transactions end	No	No	Yes	Yes
Performs a checkpoint and closes open files	No	Yes	Yes	Yes

Lesson 153: Overview of Checkpoints

- What is a Checkpoint?
- Purpose of Checkpoint?
- Recovery without Checkpoints
- Recovery using Checkpoints

What is a Checkpoint?

- A checkpoint is a crucial mechanism in consistent database shutdowns, instance recovery, and Oracle Database operation generally.
- A data structure that indicates the checkpoint position, which is the SCN in the redo stream where instance recovery must begin The checkpoint position is determined by the oldest dirty buffer in the database buffer cache.
- The checkpoint position acts as a pointer to the redo stream and is stored in the control file and in each data file header.
- The writing of modified database buffers in the database buffer cache to disk.
- Whenever checkpoint happens all the dirty buffers of DB buffer cache (volatile storage) moves into data files (Non volatile storage)

Purpose of Checkpoint?

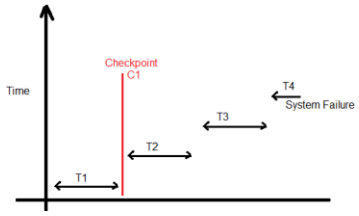
- Oracle Database uses checkpoints to achieve the following goals:
- Reduce the time required for recovery incase of an instance or media failure
- Ensure that dirty buffers in the buffer cache are written to disk regularly
- Ensure that all committed data is written to disk during a consistent shutdown
- Establish data consistency

Recovery without Checkpoints

- Extra time is wasted in processing the entire log file
- Redoing of unnecessary transactions

Recovery using Checkpoints

- Checkpoint is a mechanism in which all the previous logs are removed from the system and are stored permanently in the storage disk.



Lesson 154: Initiate Checkpoints

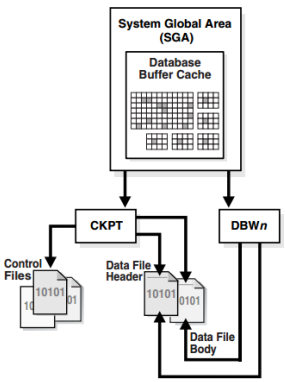
- When to initiate Checkpoint
- Checkpoint Process
- Thread Checkpoints Type
- Tablespace and Data File Checkpoints Type
- Incremental Checkpoints Type
- Checkpoints Processor and Database files

When to initiate Checkpoint

- The checkpoint process (CKPT) is responsible for writing checkpoints to the data file headers and control file. Checkpoints occur in a variety of situations

Checkpoint Process

- The checkpoint process (CKPT)updates the control file and data file headers with checkpoint information and signals DBWn to write blocks to disk. Checkpoint information includes the checkpoint position, SCN, location in online redo log to begin recovery, and so on.



Thread Checkpoints Type

The database writes to disk all buffers modified by redo in a specific thread before a certain target. The set of thread checkpoints on all instances in a database is a database checkpoint. Thread checkpoints occur in the following situations:

- Consistent database shutdown
- ALTER SYSTEM CHECKPOINT statement
- Online redo log switch
- ALTER DATABASE BEGIN BACKUP statement

Tablespace and Data File Checkpoints Type

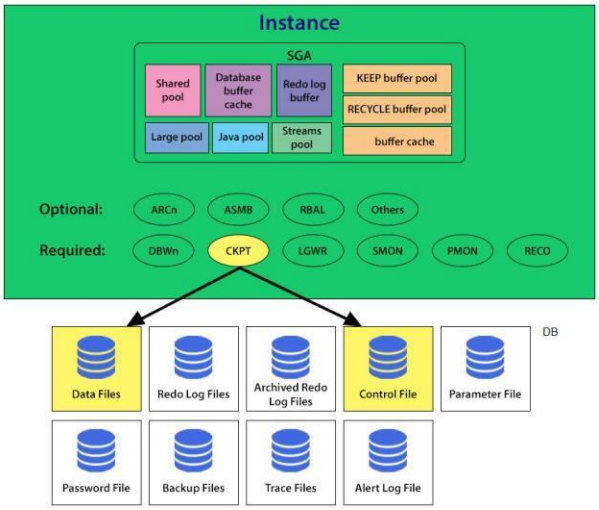
- The database writes to disk all buffers modified by redo before a specific target. A tablespace checkpoint is a set of data file checkpoints, one for each data file in the tablespace. These checkpoints occur in a variety of situations, including making a tablespace read-only or taking it offline normal, shrinking a data file, or executing

  ALTER TABLESPACE BEGIN BACKUP;

Incremental Checkpoints Type

- An incremental checkpoint is a type of thread checkpoint partly intended to avoid writing large numbers of blocks at online redo log switches. DBWn checks at least every three seconds to determine whether it has work to do.
- When DBWn writes dirty buffers, it advances the checkpoint position, causing CKPT to write the checkpoint position to the control file, but not to the data file headers.

Checkpoints Processor and Database files



Lesson 155: Checkpoints Commands Samples

- When will a Checkpoint occur
- Checkpoint Occurring Command
- Data Dictionary View for Checkpoint
- SCN is Changed After another Checkpoint
- SCN is not Changed After Switch Logfile

When will a Checkpoint occur

- At each switch of the redo log files
- Once the number of seconds defined in the LOG_CHECKPOINT_TIMEOUT is reached Once the current redo log file reaches the size

LOG Checkpoint Interval x size of IO-OS blocks, Example 10000*512=5MB

- Directly by the Alter System switch Logfile command
- Directly by the Alter System Checkpoint command

Checkpoint Occurring Command

- Run the following command with sys user

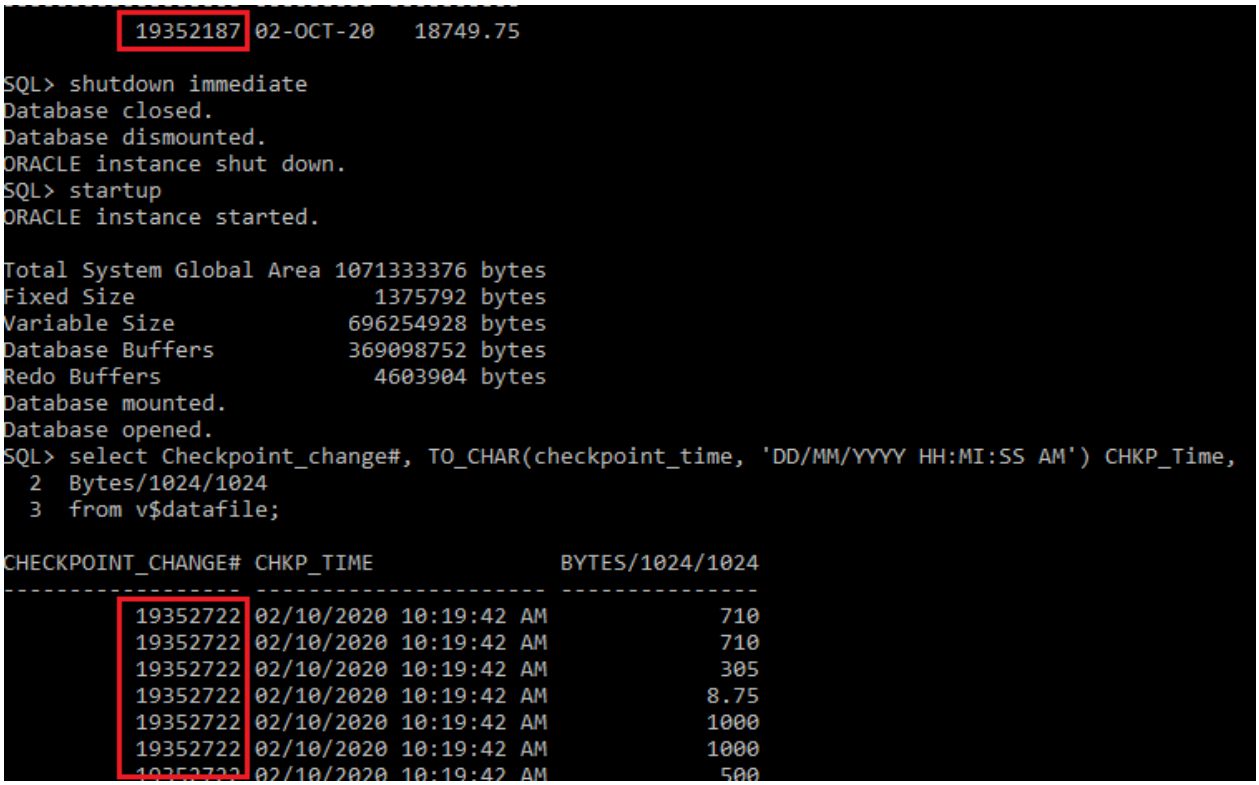
Alter system checkpoint;

Data Dictionary View for Checkpoint

```
SQL> select Checkpoint_change#, TO_CHAR(checkpoint_time, 'DD/MM/YYYY HH:MI:SS AM') CHKP_Time,
  2  Bytes/1024/1024
  3  from v$datafile;

CHECKPOINT_CHANGE# CHKP_TIME              BYTES/1024/1024
------------------ ---------------------- ---------------
          19352187 02/10/2020 10:06:26 AM             710
          19352187 02/10/2020 10:06:26 AM             710
          19352187 02/10/2020 10:06:26 AM             305
          19352187 02/10/2020 10:06:26 AM            8.75
          19352187 02/10/2020 10:06:26 AM            1000
```

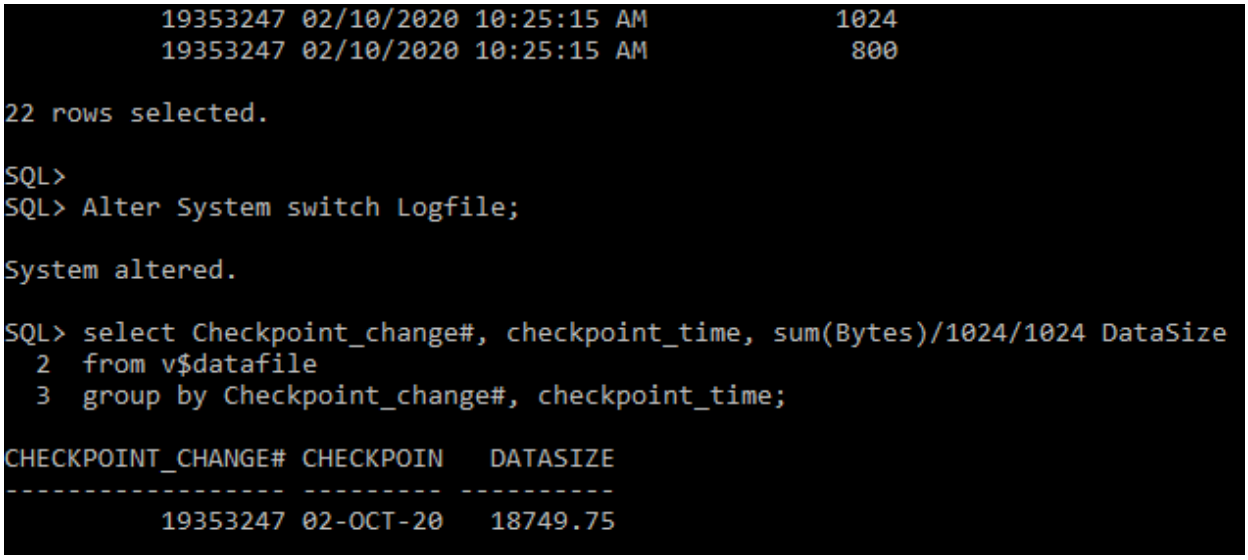
```
SQL> select Checkpoint_change#, checkpoint_time, sum(Bytes)/1024/1024 DataSize
  2  from v$datafile
  3  group by Checkpoint_change#, checkpoint_time;

CHECKPOINT_CHANGE# CHECKPOIN   DATASIZE
------------------ --------- ----------
          19352187 02-OCT-20   18749.75
```

SCN is Changed After another Checkpoint



```
        19352187 02-OCT-20   18749.75

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup
ORACLE instance started.

Total System Global Area 1071333376 bytes
Fixed Size                  1375792 bytes
Variable Size             696254928 bytes
Database Buffers          369098752 bytes
Redo Buffers                4603904 bytes
Database mounted.
Database opened.
SQL> select Checkpoint_change#, TO_CHAR(checkpoint_time, 'DD/MM/YYYY HH:MI:SS AM') CHKP_Time,
  2  Bytes/1024/1024
  3  from v$datafile;

CHECKPOINT_CHANGE# CHKP_TIME              BYTES/1024/1024
------------------ ---------------------- --------------
        19352722 02/10/2020 10:19:42 AM             710
        19352722 02/10/2020 10:19:42 AM             710
        19352722 02/10/2020 10:19:42 AM             305
        19352722 02/10/2020 10:19:42 AM            8.75
        19352722 02/10/2020 10:19:42 AM            1000
        19352722 02/10/2020 10:19:42 AM            1000
        19352722 02/10/2020 10:19:42 AM             500
```

SCN is not Changed After Switch Logfile



```
         19353247 02/10/2020 10:25:15 AM            1024
         19353247 02/10/2020 10:25:15 AM             800

22 rows selected.

SQL>
SQL> Alter System switch Logfile;

System altered.

SQL> select Checkpoint_change#, checkpoint_time, sum(Bytes)/1024/1024 DataSize
  2  from v$datafile
  3  group by Checkpoint_change#, checkpoint_time;

CHECKPOINT_CHANGE# CHECKPOIN   DATASIZE
------------------ --------- ----------
         19353247 02-OCT-20   18749.75
```

Lesson 156:Instance Recovery

- Overview of Instance Recovery
- Purpose of Instance Recovery
- How Instance Recovery is Performed?

Overview of Instance Recovery

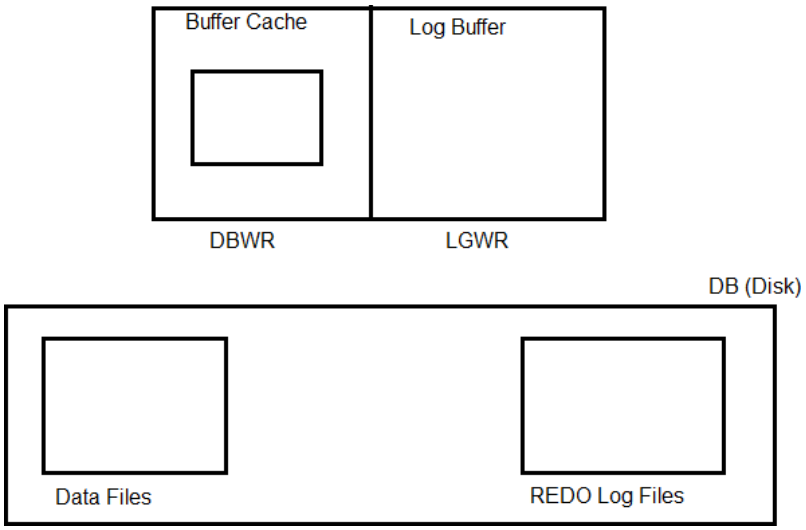
- Instance recovery is the process of applying records in the online redo log to data files to reconstruct changes made after the most recent checkpoint.
- Instance recovery occurs automatically when an administrator attempts to open a database that was previously shut down inconsistently.
- Instance recovery occurs automatically when an administrator attempts to open a database that was previously shut down inconsistently.

```
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup
ORACLE instance started.

Total System Global Area 1071333376 bytes
Fixed Size                  1375792 bytes
Variable Size             696254928 bytes
Database Buffers          369098752 bytes
Redo Buffers                4603904 bytes
Database mounted.
Database opened.
```

Purpose of Instance Recovery

- Instance recovery ensures that the database is in a consistent state after an instance failure.
- The files of a database can be left in an inconsistent state because of how Oracle Database manages database changes.

How Instance Recovery is Performed?



Lesson 157: Instance Recovery Protocol

Instance Recovery

- Situations for Instance Recovery
- Performing Instance Recovery
- Importance of CKPT for Instance Recovery

Situations for Instance Recovery

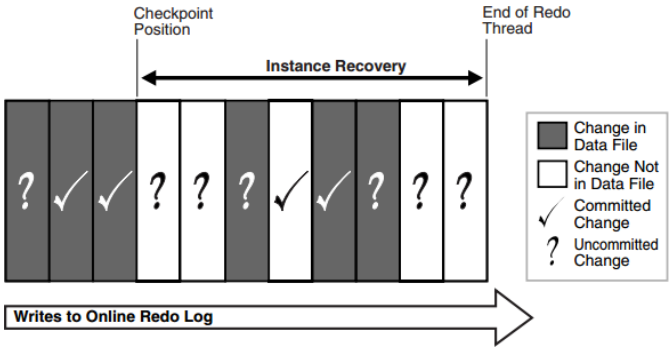
- If an instance of an open database fails, either because of a SHUTDOWN ABORT statement or abnormal termination, then the following situations can result:
- Data blocks committed by a transaction are not written to the data files and appear only in the online redo log. These changes must be reapplied to the database.
- The data files contains changes that had not been committed when the instance failed. These changes must be rolled back to ensure transactional consistency.
- Instance recovery uses only online redo log files and current online data files to
- synchronize the data files and ensure that they are consistent

Performing Instance Recovery

- Whether instance recovery is required depends on the state of the redo threads. A redo thread is marked open in the control file when a database instance opens in read/write mode, and is marked closed when the instance is shut down consistently.
- If redo threads are marked open in the control file, but no live instances hold the thread enqueues corresponding to these threads, then the database requires instance recovery.
- Oracle Database performs instance recovery automatically in the following situations:
- The database opens for the first time after the failure of a single-instance database or all instances of an Oracle RAC database. This form of instance recovery is also called crash recovery. Oracle Database recovers the online redo threads of the terminated instances together.
- Some but not all instances of an Oracle RAC database fail. Instance recovery is performed automatically by a surviving instance in the configuration.
- The SMON background process performs instance recovery, applying online redo automatically. No user intervention is required

Importance of CKPT for Instance Recovery

- Instance recovery uses checkpoints to determine which changes must be applied to the data files. The checkpoint position guarantees that every committed change with an SCN lower than the checkpoint SCN is saved to the data files.
- During instance recovery, the database must apply the changes that occur between the checkpoint position and the end of the redo thread.



Lesson 158 :Managing Recovery using MS SQL Server

MS SQL Server's Recovery

- Restore & Recovery Overview
- Overview of Restore Scenarios
- Steps to Restore a Database
- Accelerated Database Recovery
- Steps to Restore a Database
- Decision to adopt ADR
- To View & Change the Recovery Model - Transact SQL
- Without ADR, DB Recovery Follows ARIES
- DB Recovery Follows ARIES using ADR

Restore & Recovery Overview

- To recover a SQL Server database from a failure, a database administrator has to restore a set of SQL Server backups in a logically correct and meaningful restore sequence .:
- The database (a complete database restore)
- The data file (a file restore)
- The data page (Bulk load, restore individual DB)

Overview of Restore Scenarios

A *restore scenario* in SQL Server is the process of restoring data from one or more backups and then recovering the database. The supported restore scenarios depend on the recovery model of the database and the edition of SQL Server.

- Complete DB Restore
- File Restore (individual files)

Steps to Restore a Database

To perform a database restore, the Database Engine executes three steps:

- Creates the database and transaction log files if they do not already exist.
- Copies all the data, log, and index pages from the backup media of a database to the database files.
- Applies the transaction log in what is known as the recovery process.

Accelerated Database Recovery

- Accelerated Database Recovery (ADR) improves database availability, especially in the presence of long running transactions, by redesigning the SQL database engine recovery process. ADR is new in SQL Server 2019.
- ADR is available for Azure SQL Database, Azure SQL Managed Instance and Azure Synapse SQL.

Steps to Restore a Database

To perform a file restore, the Database Engine executes two steps:

- Creates any missing database file(s).
- Copies the data from the backup devices to the database file(s).

If you restore a file, you cannot recover it and bring it online until it has been rolled far enough forward to be consistent with the database.

Decision to adopt ADR

- The following types of customers should consider enabling ADR:
- Customers that have workloads with long running transactions.

- Customers that have seen cases where active transactions are causing the transaction log to grow significantly.
- Customers that have experienced long periods of database unavailability due to SQL Server long running recovery (such as unexpected SQL Server restart or manual transaction rollback).

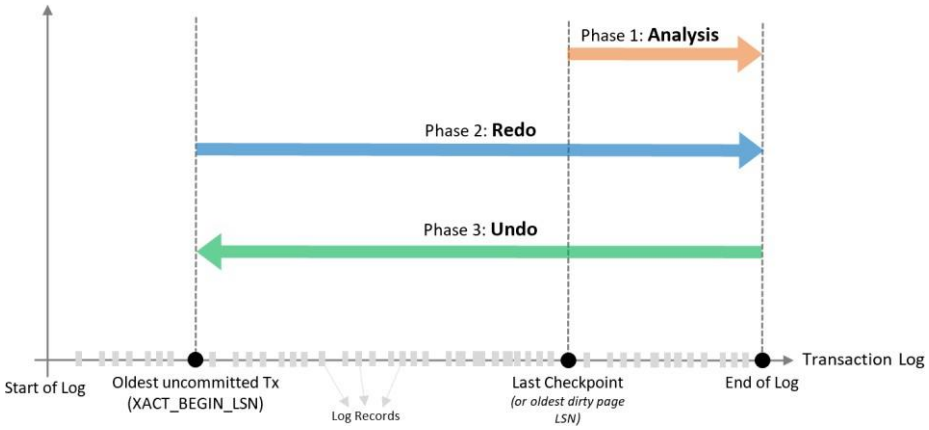
To View & Change the Recovery Model - Transact SQL

- Connect to the Database Engine. From the Standard bar, click **New Query**.
- Copy and paste the following example into the query window and click **Execute**. This example shows how to query the sys.databases catalog view to learn the recovery model of the **model** database.

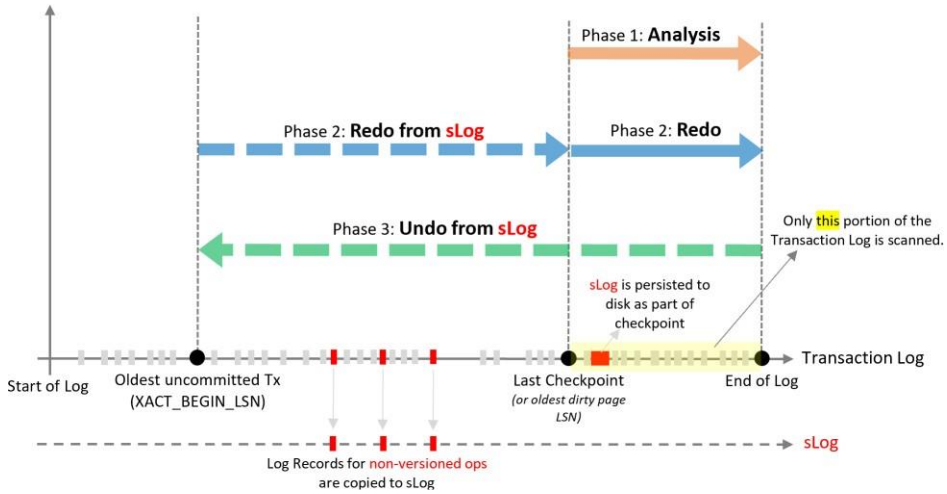
```
SQL

SELECT name, recovery_model_desc
   FROM sys.databases
      WHERE name = 'model' ;
GO
```

```
SQL

USE [master] ;
ALTER DATABASE [model] SET RECOVERY FULL ;
```

Without ADR, DB Recovery Follows ARIES



DB Recovery Follows ARIES using ADR



Lesson159: Phases of Instance Recovery using Oracle

Phases of Instance Recovery

- Introduction
- Rolling Forward
- After Roll Forward
- Rolling Forward & Rolling Backward

Introduction

- The first phase of instance recovery is called cache recovery or rolling forward, and involves reapplying all of the changes recorded in the online redo log to the data files.
- Because rollback data is recorded in the online redo log, rolling forward also regenerates the corresponding undo segments.

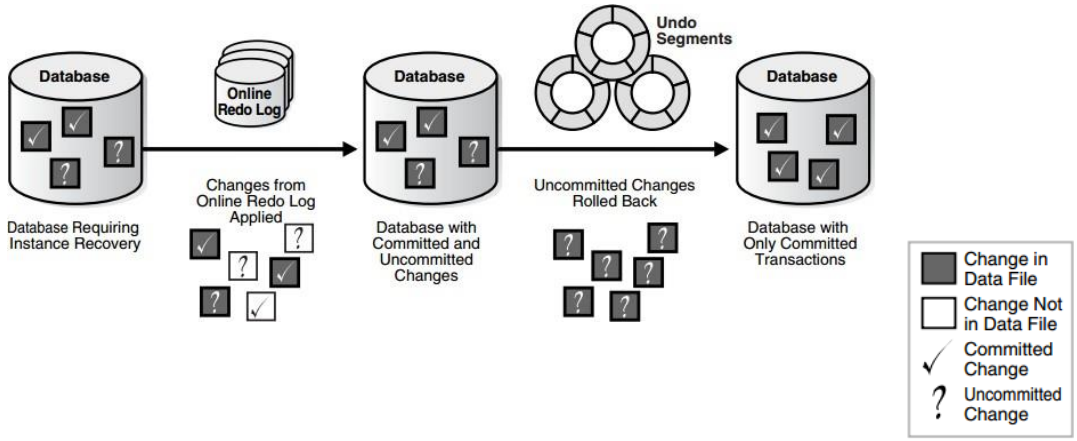
Rolling Forward

- Rolling forward proceeds through as many online redo log files as necessary to bring the database forward in time.
- After rolling forward, the data blocks contain all committed changes recorded in the online redo log files. These files could also contain uncommitted changes that were either saved to the data files before the failure, or were recorded in the online redo log and introduced during cache recovery.

After Roll Forward

- After the roll forward, any changes that were not committed must be undone. Oracle Database uses the checkpoint position, which guarantees that every committed change with an SCN lower than the checkpoint SCN is saved on disk. Oracle Database applies undo blocks to roll back uncommitted changes in data blocks that were written before the failure or introduced during cache recovery. This phase is called rolling back or transaction recovery.

Rolling Forward & Rolling Backward



Lesson 160:Overview of Parameter Files with commands

- Introduction
- Initialization Parameters
- Functional Group of Initialization Parameters
- Purpose of Variable Parameters
- Basic Initialization Parameters
- Advance Initialization Parameters
- Parameters during Installation
- Parameters Files spfile and pfile

Introduction

- To start a database instance, Oracle Database must read either a server parameter file, which is recommended, or a text initialization parameter file, which is a legacy implementation. These files contain a list of configuration parameters.

Initialization Parameters

- Initialization parameters are configuration parameters that affect the basic operation of an instance. The instance reads initialization parameters from a file at startup.
- Oracle Database provides many initialization parameters to optimize its operation in diverse environments. Only a few of these parameters must be explicitly set because the default values are adequate in most cases.

Functional Group of Initialization Parameters

Most initialization parameters belong to one of the following functional groups:

- Parameters that name entities such as files or directories
- Parameters that set limits for a process, database resource, or the database itself
- Parameters that affect capacity, such as the size of the SGA (these parameters are called variable parameters)

Purpose of Variable Parameters

- Variable parameters are of particular interest to database administrators because they can use these parameters to improve database performance.

Basic Initialization Parameters

- In most cases, you must set and tune only the approximately 30 basic parameters to obtain reasonable performance. The basic parameters set characteristics such as the database name,
- locations of the control files,
- database block size, and
- undo tablespace

Advance Initialization Parameters

- In rare situations, modification to the advanced parameters may be required for optimal performance.
- The advanced parameters enable expert DBAs to adapt the behavior of the Oracle Database to meet unique requirements

Parameters during Installation

- Oracle Database provides values in the starter initialization parameter file provided with your database software, or as created for you by the Database Configuration Assistant.

```
db_name=sample
control_files=/disk1/oradata/sample_cf.dbf
db_block_size=8192
open_cursors=52
undo_management=auto
shared_pool_size=280M
pga_aggregate_target=29M
.
.
```

Parameters during Installation

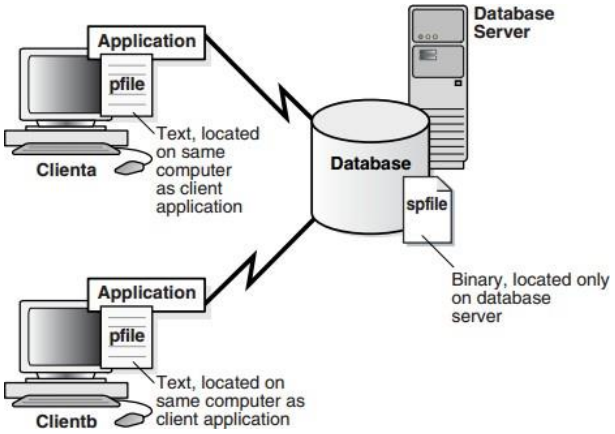
create spfile from pfile;

create pfile from spfile;

show parameter processes;

show parameter spfile;

Parameters Files spfile and pfile



Lesson 161 :Initializing Parameter Files

- Server's Parameters Characteristics
- Text Parameters File
- Text Parameters Characteristics
- Use of alter system command
- Static and Dynamic Parameters
- **Initializing Parameter Files**

Server's Parameters Characteristics

- Only one server parameter file (spfile) exists for a database. This file must reside on the database host.

- The server parameter file is written to and read by only by Oracle Database, not by client applications.
- The server parameter file is binary and cannot be modified by a text editor.
- Initialization parameters stored in the server parameter file are persistent. Any changes made to the parameters while a database instance is running can persist across instance shutdown and startup.

Text Parameters File

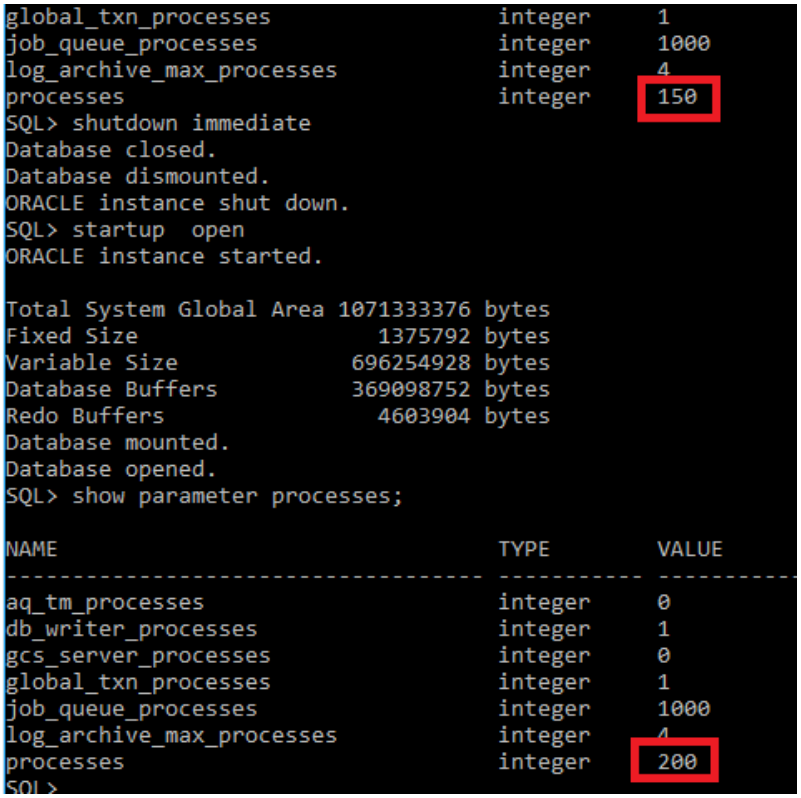
- A text initialization parameter file (pfile) is a text file that contains a list of initialization parameters. It is readable and writeable to DBA and its changes are reflected into spfile on Server.
- The text initialization parameter file contains a series of key=value pairs, one per line

Text Parameters Characteristics

- When starting up or shutting down a database, the text initialization parameter file must reside on the same host as the client application that connects to the database.
- A text initialization parameter file is text-based, not binary.
- Oracle Database can read but not write to the text initialization parameter file. To change the parameter values you must manually alter the file with a text editor.
- Changes to initialization parameter values by ALTER SYSTEM are only in effect for the current instance. You must manually update the text initialization parameter file and restart the instance for the changes to be known.

Use of Alter System command

- Alter system set processes=200 scope=spfile;
- create pfile='E:\app\Haider\product\11.2.0\dbhome_1\database\pfileORCL.ora' from SPFILE;
- This change can be seen by opening spfile



Static and Dynamic Parameters

Characteristic	Static	Dynamic
Requires modification of the parameter file (text or server)	Yes	No
Requires database instance restart before setting takes affect	Yes	No
Described as "Modifiable" in *Oracle Database Reference* initialization parameter entry	No	Yes
Modifiable only for the database or instance	Yes	No

Static parameters include DB_BLOCK_SIZE, DB_NAME, and COMPATIBLE. Dynamic parameters are grouped into session-level parameters, which affect only the current user session, and system-level parameters, which affect the database and all sessions.

MEMORY_TARGET=system-level & NLS_DATE_FORMAT=session level parameters

Lesson 162: Diagnostic Instance Files

- Overview of Diagnostic Files
- Goals of fault Diagnosability Infrastructure
- Automatic Diagnostic Repository
- Problems & Incidence
- DDL & DEBUG Log files - Example
- ADR Directory Structure

Overview of Diagnostic Files

- Oracle Database includes a fault diagnosability infrastructure for preventing, detecting, diagnosing, and resolving database problems. Problems include critical errors such as code bugs, metadata corruption, and customer data corruption.

Goals of fault Diagnosability Infrastructure

- Detecting problems proactively
- Limiting damage and interruptions after a problem is detected
- Reducing problem diagnostic and resolution time
- Simplifying customer interaction with Oracle Support

Automatic Diagnostic Repository

- Automatic Diagnostic Repository (ADR) is a file-based repository that stores database diagnostic data such as trace files, the alert log, and Health Monitor reports. Key characteristics of ADR include:
- Unified directory structure
- Consistent diagnostic data formats
- Unified tool set

Problems & Incidence

- ADR proactively tracks problems, which are critical errors in the database. Critical errors manifest as internal errors, such as ORA-600, or other severe errors. Each problem has a problem key, which is a text string that describes the problem.

- When a problem occurs multiple times, ADR creates a time-stamped incident for each occurrence. An incident is uniquely identified by a numeric incident ID. When an incident occurs, ADR sends an incident alert to Enterprise Manager. Diagnosis and resolution of a critical error usually starts with an incident alert.
- ADR informs you that a critical error is ongoing without overloading the system with diagnostic data

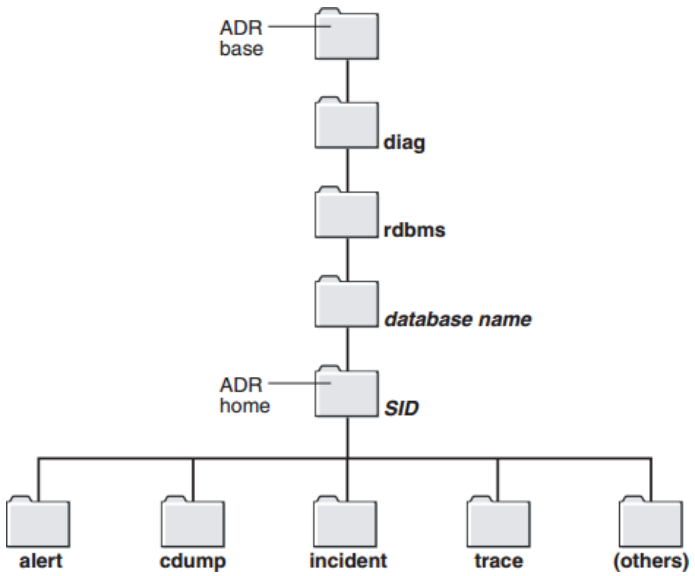
DDL & DEBUG Log Files - Example

- DDL Log: In the alert log file, DDL statements can be recorded when using parameter ENABLE_DDL_LOGGING to TRUE with NOMOUNT startup of database (in Oracle 11g version).
- Following commands can be recorded with TRUE option:
- Create Table
- Create Index
- Create User
- Drop Table

ADR Directory Structure

- The ADR base is the ADR root directory. The ADR base can contain multiple ADR
- homes, where each ADR home is the root directory for all diagnostic data—traces,
- dumps, the alert log, and so on—for an instance of an Oracle product or component.

```
SQL> SELECT substr(NAME,1,30) Diag_name, substr(VALUE, 1, 50) diag_val
  2  FROM V$DIAG_INFO;

DIAG_NAME                      DIAG_VAL
------------------------------ --------------------------------------------------
Diag Enabled                   TRUE
ADR Base                       e:\app\haider
ADR Home                       e:\app\haider\diag\rdbms\orcl\orcl
Diag Trace                     e:\app\haider\diag\rdbms\orcl\orcl\trace
Diag Alert                     e:\app\haider\diag\rdbms\orcl\orcl\alert
Diag Incident                  e:\app\haider\diag\rdbms\orcl\orcl\incident
Diag Cdump                     e:\app\haider\diag\rdbms\orcl\orcl\cdump
Health Monitor                 e:\app\haider\diag\rdbms\orcl\orcl\hm
Default Trace File             e:\app\haider\diag\rdbms\orcl\orcl\trace\orcl_ora_
Active Problem Count           0
Active Incident Count          0

11 rows selected.
```



Lesson 163: Commands & Log Alerts

- Overview of Alert File
- Alert Log File
- Alert Log File – Contents
- Alert Log File - Maintenance
- Alert Log File - Location

Overview of Alert File

Each database has an alert log, which is an XML and text file containing a chronological log of database messages and errors. The alert log contents include the following:

- Deadlock occurs
- Any error written by Oracle Server process
- Any unexpected messages and corruptions happened
- Any DBA DDL operations are managed

Alert Log File

- The Alert log file (ALERT.LOG) is a
- chronological log of messages and errors written out by an Oracle Database
- This file should constantly be monitored to detect unexpected messages and corruptions
- The Alert log file (ALERT.LOG) is a
- chronological log of messages and errors written out by an Oracle Database
- This file should constantly be monitored to detect unexpected messages and corruptions

Alert Log File - Contents

The alert log file contains the following information that can be used to aid in tuning the database

- Time to perform archiving
- Instance recovery start and complete times
- Deadlock and timeout errors
- Incomplete checkpoints
- Checkpoint start and end times

Alert Log File - Maintenance

As Alert file grows bigger,

- DBA can periodically backup the alert log file and delete the current file

When database attempts to write to the alert file again, it will create a new file

Alert Log File - Location

Each database has an Alert_SID.log file

Resides on Server with the database

Default location

Oracle-Base/diag/rdbms/DB_Name/SID/trace

Lesson 164:Trace Files Commands

- Overview of Trace Files
- Types of Trace Files
- Contents of Trace Files
- Locations of Trace Files

Overview of Trace Files

- A trace file is an administrative file that contain diagnostic data used to investigate problems. Also, trace files can provide guidance for tuning applications or an instance.

Types of Trace Files

- Each server and background process can periodically write to an associated trace file. The files information on the process environment, status, activities, and errors.
- The SQL trace facility also creates trace files, which provide performance information on individual SQL statements.
- To enable tracing for a client identifier, service, module, action, session, instance, or database, you must execute the appropriate procedures in the DBMS_MONITOR package or use Oracle Enterprise Manager.
- A dump is a special type of trace file. Whereas a trace tends to be continuous output of diagnostic data, a dump is typically a one-time output of diagnostic data in response to an event (such as an incident).
- When an incident occurs, the database writes one or more dumps to the incident directory created for the incident. Incident dumps also contain the incident number in the file name

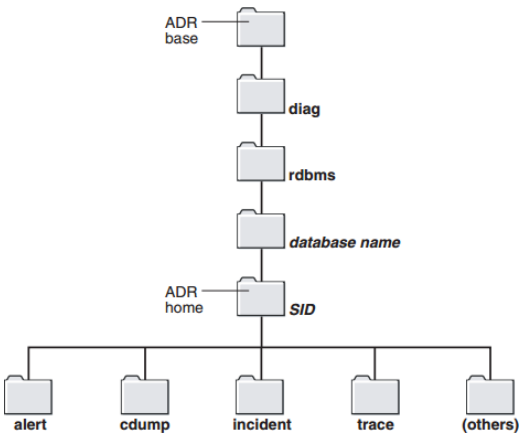
Contents of Trace Files

- Typically, database background process trace file names contain the
- Oracle SID,
- background process name, and
- operating system process number

An example of a trace file for the RECO process is mytest_reco_10355.trc

Locations of Trace Files

ADR stores trace files in the trace sub directory, as shown in Figure Trace file names are platform dependent and use the extension .trc.



Lesson 165:Managing users & Security

- Types of Security
- Threats to Databases
- Security Domain
- Schema Objects & User
- User & Database
- Creating Oracle User
- Pre-defined Accounts

Types of Security

Following are the database security types and issues

- Legal & Ethical Issues
- Policy Issues in Enterprises
- Systems' related Issues
- Needs of Enterprises

Threats to Databases

Following are the database related threats and issues

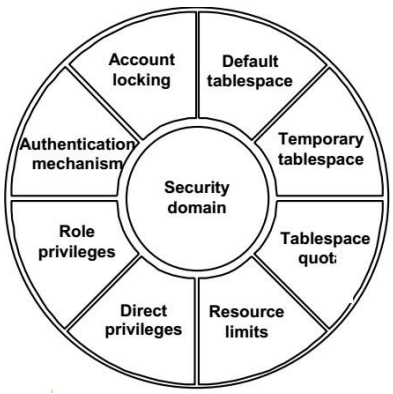
- Loss of integrity
- Loss of availability
- Loss of Confidentiality

Security Domain

The database administrator defines the names of the users who are allowed to access a database. A security domain defines the settings that apply to the user.

Authentication mechanisms are

- Data Dictionary
- Operating System
- Network



Schema Objects & User

- A Schema is a collection of objects
- A user is associated (owner) of schema
- Creating schema means creating a user as an owner of schema

User & Database

- Accessing Database (connection with the database)
- Specifying database unique user account of ID
- Specifying authentication (roles & privileges) to user account

Allocating space, quota, buffer etc. are additional features in security.

Creating Oracle User

- A unique user name or account
- Authentications (roles/ rights)
- A default tablespace
- A temporary tablespace
- A user profile
- An account status (lock/unlock)

Pre-defined Accounts

- SYS account (sys as sysdba)
    - is granted the DBA role
    - has all privileges with ADMIN OPTION
    - is used for startup, shutdown etc.
    - owns the data dictionary
- SYSTEM account is granted the DBA role
    - Normally not used for routine operations

Lesson166: Creating Users with Commands

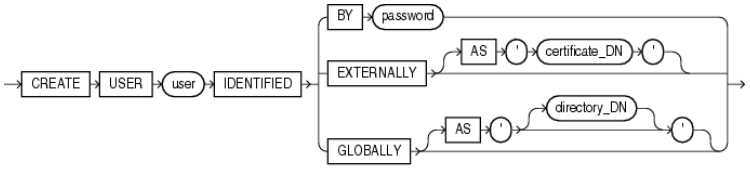
- Creating Oracle User
- Syntax for Creating Oracle User
- Data Dictionary View DBA_USERS

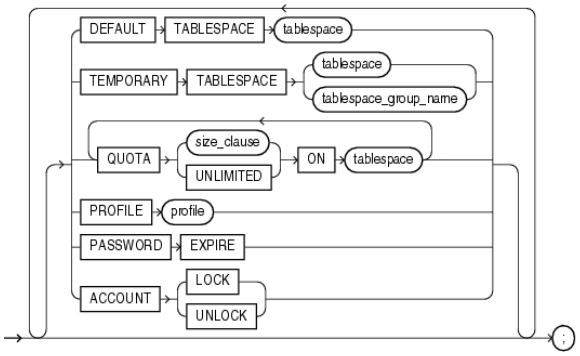
Creating Oracle User

Checklist's clauses

- Tablespaces
- Quota
- Roles/ privileges

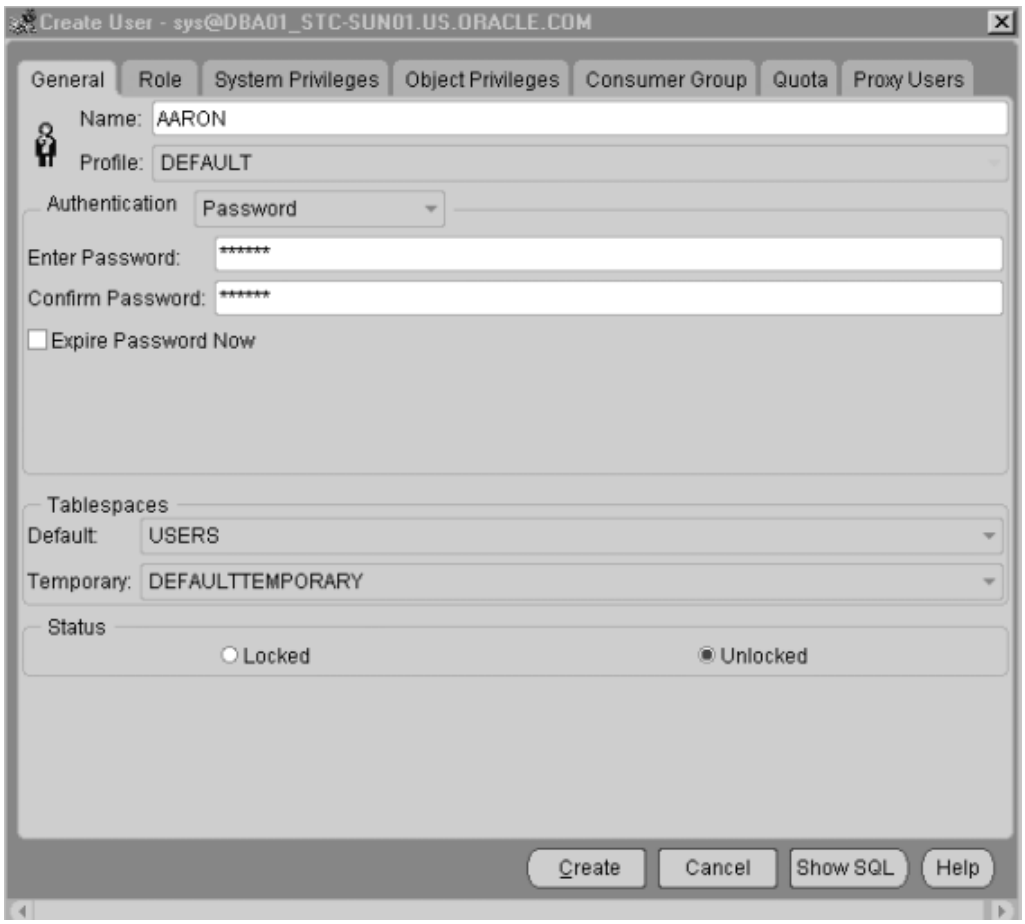
Syntax for Creating Oracle User






```
CREATE USER sidney
    IDENTIFIED BY out standing1
    DEFAULT TABLESPACE example
    QUOTA 10M ON example
    TEMPORARY TABLESPACE temp
    QUOTA 5M ON system
    PROFILE app_user
    PASSWORD EXPIRE;
```

Creating Oracle User using Oracle-EM



Data Dictionary View DBA_USERS

```
SQL> select username, account_status, created
  2  from dba_users;

USERNAME                       ACCOUNT_STATUS                   CREATED
------------------------------ -------------------------------- ---------
NOVUS                          OPEN                             18-DEC-17
NOVUSDB                        OPEN                             16-OCT-17
PROD                           OPEN                             19-JUL-17
NOVUSDB2                       OPEN                             13-DEC-17
NEONDEV                        EXPIRED                          23-JAN-17
SCOTT                          OPEN                             02-APR-10
NEONDB                         OPEN                             31-MAY-17
TEST                           OPEN                             14-FEB-17
MDDATA                         EXPIRED                          02-APR-10
ORACLE_OCM                     EXPIRED & LOCKED                 02-APR-10
```

Lesson167: Users information using Data Dictionary Views

- Searching Security Dictionary Views
- Single SQL to Display All Views
- Listing All Tablespace Quotas
- Listing All Profiles & Assigned Limit
- Searching Security Dictionary Views
- Searching Data Dictionary Views closest to security users' information. Search is managed by following SQL LIKE operator:
- '%QUOTA%', '%PASSWORD%'
- '%PROFILE%', '%COST%'
- '%SESS%', '%PROXY%' and *_USERS

Single SQL to Display All Views

```
SQL> SELECT *
  2  FROM DICTIONARY
  3  where table_name like '%QUOTA%'
  4  or table_name like '%PASSWORD%'
  5  or table_name like '%PROFILE%'
  6  or table_name like '%COST%'
  7  or table_name like '%SESS%'
  8  or table_name like '%PROXY%';

TABLE_NAME           COMMENTS
-------------------- ----------------------------------------------

DBA_HIST_SESS_TIME_S CPU and I/O time for interesting (STREAMS)
TATS                 sessions

DBA_LOGMNR_SESSION
DBA_PROFILES         Display all profiles and their limits
DBA_SQL_PROFILES     set of sql profiles
DBA_TS_QUOTAS        Tablespace quotas for all users
USER_AUDIT_SESSION   All audit trail records concerning CONNECT and
                     DISCONNECT
```

Listing All Tablespace Quotas

```
SELECT * FROM DBA_TS_QUOTAS;

TABLESPACE    USERNAME    BYTES     MAX_BYTES    BLOCKS    MAX_BLOCKS
----------    ---------  --------   ----------   -------   ----------
USERS         JFEE              0       512000         0          250
USERS         DCRANNEY          0           -1         0           -1
```

Listing All Profiles & Assigned Limit

```
SELECT * FROM DBA_PROFILES
   ORDER BY PROFILE;

PROFILE              RESOURCE_NAME              RESOURCE   LIMIT
-----------------    ---------------            ---------- ----------
CLERK                COMPOSITE_LIMIT            KERNEL     DEFAULT
CLERK                FAILED_LOGIN_ATTEMPTS      PASSWORD   DEFAULT
CLERK                PASSWORD_LIFE_TIME         PASSWORD   DEFAULT
CLERK                PASSWORD_REUSE_TIME        PASSWORD   DEFAULT
CLERK                PASSWORD_REUSE_MAX         PASSWORD   DEFAULT
CLERK                PASSWORD_VERIFY_FUNCTION   PASSWORD   DEFAULT
CLERK                PASSWORD_LOCK_TIME         PASSWORD   DEFAULT
CLERK                PASSWORD_GRACE_TIME        PASSWORD   DEFAULT
CLERK                PRIVATE_SGA                KERNEL     DEFAULT
CLERK                CONNECT_TIME               KERNEL     600
CLERK                IDLE_TIME                  KERNEL     30
```

Lesson 168: Managing Privileges

- Types of Oracle User Privileges
- System Privileges
- System Privileges with ADMIN OPTION
- Object Privileges Syntax
- Object Privileges - Examples
- Objects Privileges Options
- Object Privileges using Oracle EM

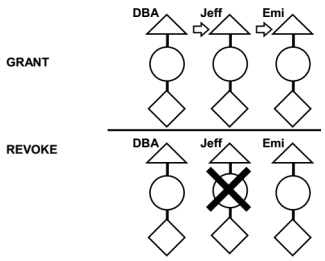
Types of Oracle User Privileges

- System: Enables users to perform particular actions in the database
- Object: Enables users to access and manipulate a specific object

System Privileges

- More than 100 distinct system privileges
- ANY keyword in privileges signifies that users have the privilege in any schema
- GRANT command adds a privilege to a user or a group of users
- REVOKE command deletes the privileges

System Privileges with ADMIN OPTION



Scenario

1. The DBA grants the `CREATE TABLE` system privilege to Jeff with the `ADMIN OPTION`.
2. Jeff creates a table.
3. Jeff grants the `CREATE TABLE` system privilege to Emi.
4. Emi creates a table.
5. The DBA revokes the `CREATE TABLE` system privilege from Jeff.

The result

Jeff's table still exists, but no new tables can be created.

Emi's table still exists and she still has the `CREATE TABLE` system privilege.

Object Privileges Syntax

```
GRANT { object_privilege [(column_list)]
     [, object_privilege [(column_list)] ]...
     |ALL [PRIVILEGES]}
ON  [schema.]object
TO  {user|role|PUBLIC}[, {user|role|PUBLIC} ]
    [WITH GRANT OPTION]
```

Object Privileges - Examples

grant select on EMP to utest;

grant select, update on DEPT to utest;

grant select on v_emp to utest;

--login with utest

select *

from scott.dept;

Objects Privileges Options

--scott is the owner of EMP table and granting

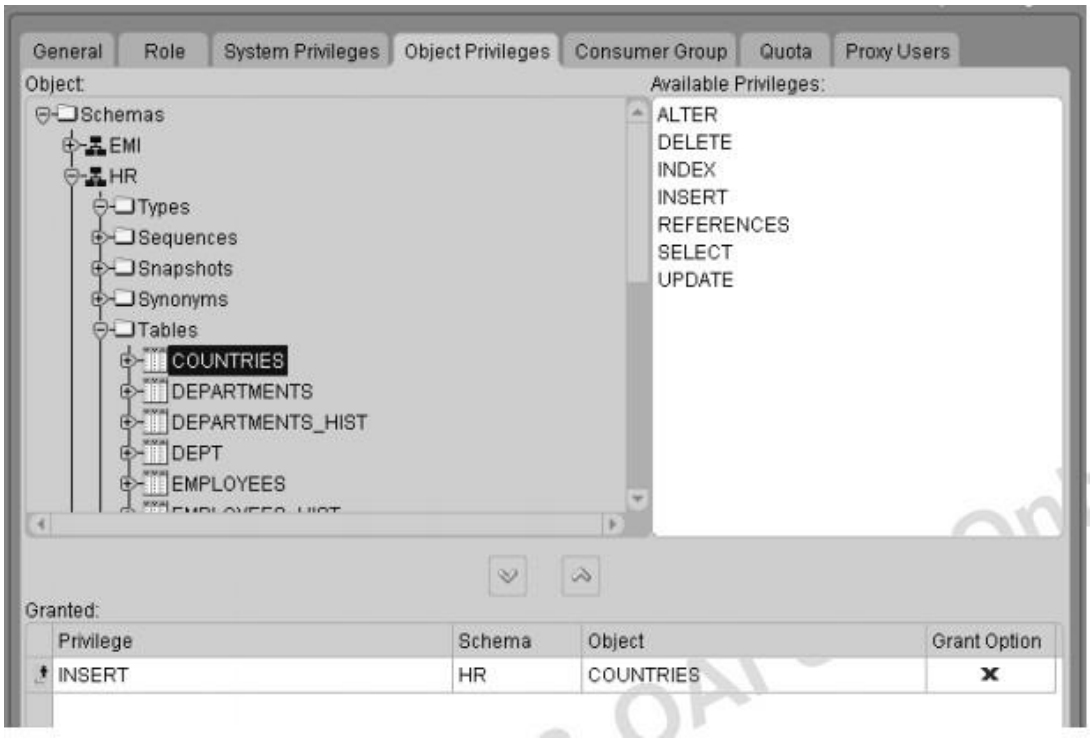
GRANT select on EMP to jef;

Object priv.	Table	View	Sequence	Procedure
ALTER	√	√	√	
DELETE	√	√		
EXECUTE				√
INDEX	√	√		
INSERT	√	√		
REFERENCES	√			
SELECT	√	√	√	
UPDATE	√	√		

--scott is the owner of EMP table and granting

GRANT select on EMP to jef;

Object Privileges using Oracle EM



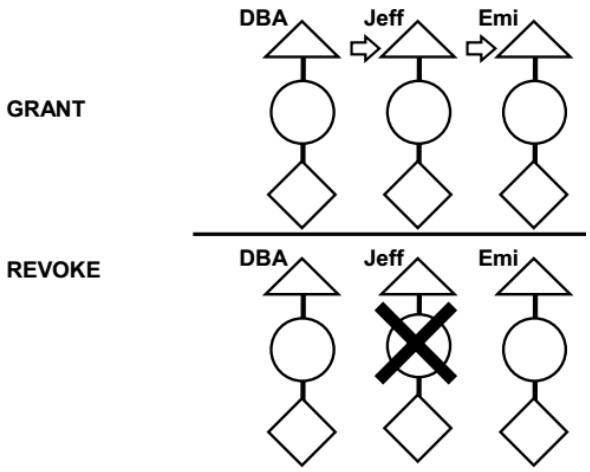
Lesson 169: System Privileges Commands

- System Privileges
- System Privileges with ADMIN OPTION
- System Privileges - Examples
- System Privileges SYSOPER & SYSDBA
- System Privileges using Oracle EM

System Privileges

- More than 100 distinct system privileges
- ANY keyword in privileges signifies that users have the privilege in any schema
- GRANT command adds a privilege to a user or a group of users
- REVOKE command deletes the privileges

System Privileges with ADMIN OPTION



Scenario

1. The DBA grants the `CREATE TABLE` system privilege to Jeff with the `ADMIN OPTION`.
2. Jeff creates a table.
3. Jeff grants the `CREATE TABLE` system privilege to Emi.
4. Emi creates a table.
5. The DBA revokes the `CREATE TABLE` system privilege from Jeff.

The result

Jeff's table still exists, but no new tables can be created.

Emi's table still exists and she still has the `CREATE TABLE` system privilege.

System Privileges - Examples

Grant create table to utest;

Grant create view, create procedure to utest;

Grant create view to scott;

Revoke create table from utest;

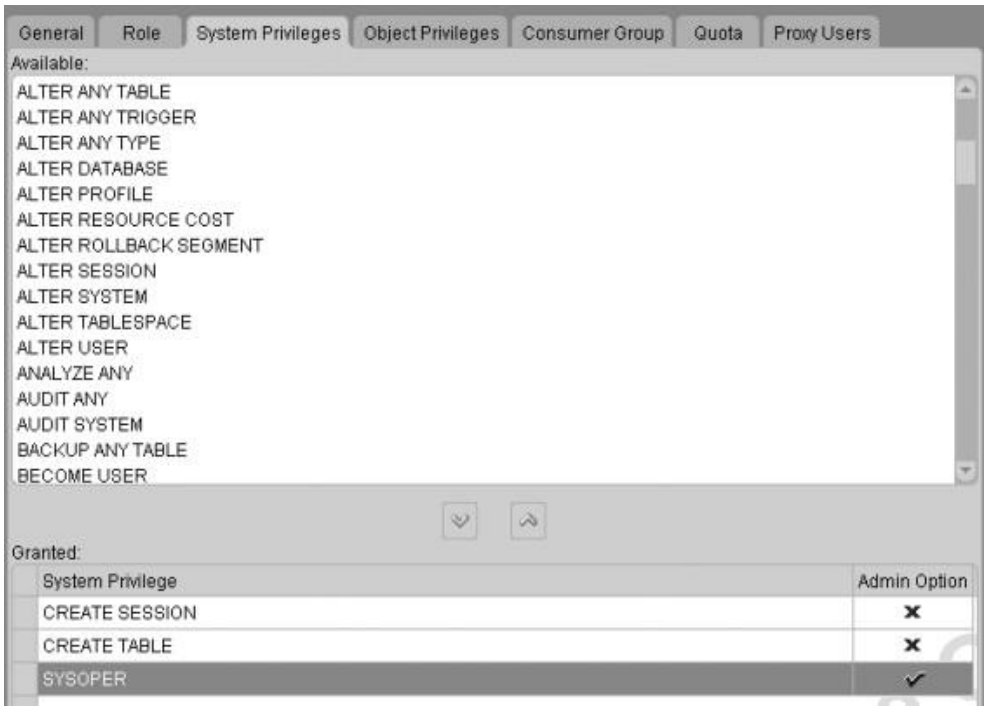
System Privileges SYSOPER & SYSDBA

Category	Examples
SYSOPER	STARTUP SHUTDOWN ALTER DATABASE OPEN | MOUNT ALTER DATABASE BACKUP CONTROLFILE TO RECOVER DATABASE ALTER DATABASE ARCHIVELOG RESTRICTED SESSION
SYSDBA	SYSOPER PRIVILEGES WITH ADMIN OPTION CREATE DATABASE ALTER TABLESPACE BEGIN/END BACKUP RESTRICTED SESSION RECOVER DATABASE UNTIL

--sys is giving grants to utest

GRANT create table, create table to utest;

System Privileges using Oracle EM



Lesson 170:System Multiple Privileges Commands

- What is User Privileges?
- Giving Privileges to Users
- Script Create Synonym
- Synonym Privileges to Users
- System and Object Privileges

What is User Privileges?

A user privilege is the right to run a particular type of SQL statement, or the right to access an object that belongs to another user, run a PL/SQL package, and so on.

- System Privileges
- Object Privileges
- Roles

Giving Privileges to Users

- -login sys / as sysdba
- create user u1
- identified by u1
- temporary tablespace temp;
- conn u1/u1
- --error ...
- grant create session, create table to u1;

Script Create Synonym

- Run following SQL in owner's (scott) schema and run each command of script in granting user (say u1). Synonyms will be created in u1 schama.

```
SQL> select 'create synonym '||tname||' for scott.'||tname||';'
  2  from tab;

'CREATESYNONYM'||TNAME||'FORSCOTT.'||TNAME||';'
--------------------------------------------------------------------------------
create synonym BLOBS for scott.BLOBS;
create synonym BONUS for scott.BONUS;
create synonym CAMPUS for scott.CAMPUS;
create synonym CHAINED_ROWS for scott.CHAINED_ROWS;
create synonym CLIENT for scott.CLIENT;
create synonym CUSTOMER for scott.CUSTOMER;
```

Synonym Privileges to Users

User u1 is using object table as scott.emp, for example

Select * from scott.EMP; --when scott is authorizing select on EMP to u1

User u1 can run the following command

create synonym emp for scott.emp;

select * from tab; --can display EMP as synonym

select * from emp;

(Subject to all assigned privileges)

System and Object Privileges

SYSTEM PRIVILEGES	OBJECT PRIVILEGES
In System privilege are more than 100 privileges	All DML are come into object privileges.
CREATE SESSION	SELECT
CREATE TABLE	INSERT
CREATE USER	UPDATE
CREATE VIEW	DELETE
CREATE SEQUENCE	REFERENCES
CREATE PROCEDURE	ALTER
CREATE ROLE etc.	INDEX
	EXECUTE

Lesson 171:Revoke Privileg

- Data Dictionary for Privileges
- Revoke & Example
- To See Existing Privileges
- Revoke Privileges CASCADE

Data Dictionary for Privileges

- We can use string search PRIV to display possible data dictionary views.
- For Example,
- select table_name
- from dict
- where table_name like '%PRIV%';
- We choose following suitable PRIV View:
- desc USER_TAB_PRIVS is showing columns
- GRANTEE
- OWNER
- TABLE_NAME
- GRANTOR
- PRIVILEGE
- GRANTABLE
- HIERARCHY

Revoke & Example

The REVOKE statement is used to revoke object privileges.

- You previously granted the object privilege to the user or role.
- You possess the GRANT ANY OBJECT PRIVILEGE system privilege that enables you to grant and revoke privileges on behalf of the object owner.

REVOKE REFERENCES ON dept FROM jward CASCADE CONSTRAINTS;

Assuming you are the original grantor, the following statement revokes the SELECT and INSERT privileges on the emp table from the users jfee and tsmith:

REVOKE SELECT, insert ON emp FROM jfee, tsmith;

To See Existing Privileges

GRANT select on EMP to u1;

```
SQL> COLUMN OWNER FORMAT A20
SQL> COLUMN GRANTOR FORMAT A20
SQL> COLUMN GRANTEE FORMAT A20
SQL> COLUMN privilege FORMAT A20
SQL> select OWNER, GRANTOR, GRANTEE, privilege
  2  from USER_TAB_PRIVS
  3  where table_name='EMP';

OWNER                GRANTOR              GRANTEE              PRIVILEGE
-------------------- -------------------- -------------------- ----------
SCOTT                SCOTT                U1                   SELECT
```

Revoke Privileges CASCADE

--Login with Bob

Grant select on EMP to Jeff with grant option;

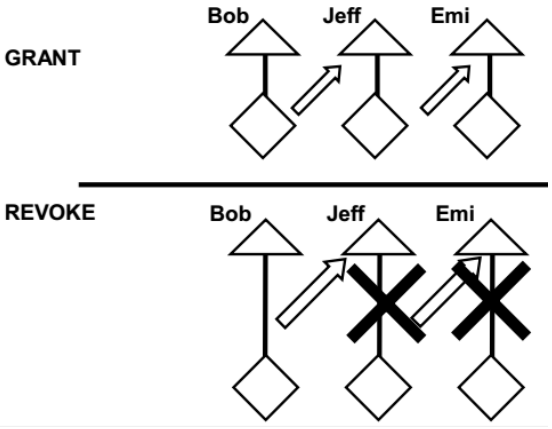
--Login Jeff

Grant select on EMP to Emi;

--Login with Bob

Revoke select on EMP from Jeff;

--This revoke is cascaded to Emi



Lesson 172:Managing Roles

- Introduction
- Data Dictionary for Privileges
- Privileges & Roles
- Roles & Users

Introduction

- Roles are named groups of related privileges or rights.
- Each user can be granted roles and privileges for each role, giving them a tailored set of abilities.
- Roles define what actions the user can take, and privileges specify the targets to which their roles apply.
- You can view the permissions granted by each role, add and remove users, and assign roles and notification profiles to users.

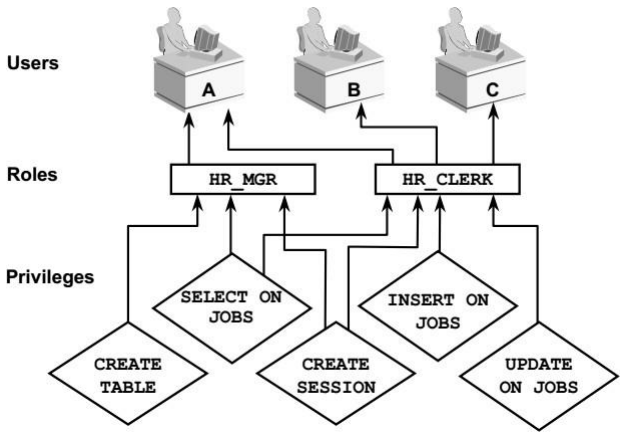
Data Dictionary for Privileges

- We can use string search PRIV to display possible data dictionary views.
- For Example,
- select table_name
- from dict
- where table_name like '%PRIV%';

Privileges & Roles

Type	Description
System privilege	A system-defined privilege usually granted only by administrators. These privileges allow users to perform specific database operations.
Object privilege	A system-defined privilege that controls access to a specific object.
Role	A collection of privileges and other roles. Some system-defined roles exist, but most are created by administrators. Roles group together privileges and other roles, which facilitates the granting of multiple privileges and roles to users.

Roles & Users



Lesson 173:Benefits of Roles-1

- Benefits of Roles
- Standard and Default Roles
- Roles of User SYS
- Analyzing Roles & Privileges to Users

Benefits of Roles

- Easier privilege management
- Dynamic privilege management
- Selective availability of privileges
- Can be granted through the operating system

Easier privilege management

- Use roles to simplify privilege management. Rather than granting the same set of privileges to several users, you can grant the privileges to a role, and then grant that role to each user.

Dynamic privilege management

- If the privileges associated with a role are modified, all the users who are granted the role acquire the modified privileges automatically and immediately.

Selective availability of privileges

- Roles can be enabled or disabled to turn privileges on or off temporarily. Enabling a role can also be used to verify that a user has been granted that role.

Granting through the operating system

- Operating system commands or utilities can be used to assign roles to users in the database.

Standard and Default Roles

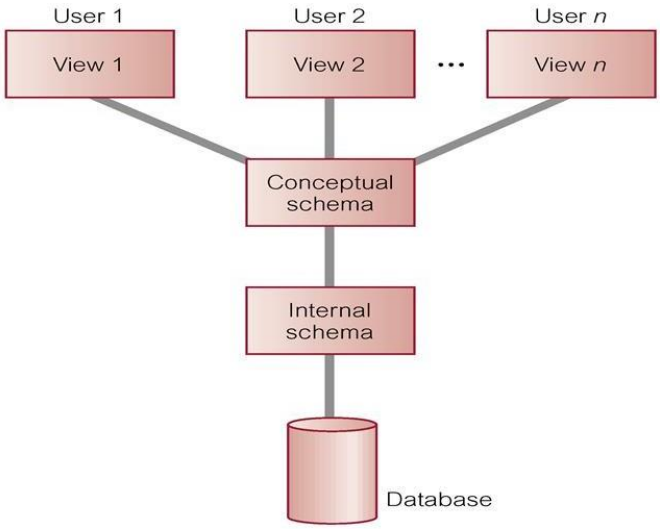
- CONNECT is a role that Enterprise Manager automatically grants to a user when you create a user as shown in "Creating Users". This role has the CREATE SESSION privilege.
- RESOURCE extends the privileges of a user beyond those granted by the CONNECT role. It includes CREATE PROCEDURE, CREATE TRIGGER, and other system privileges.
- DBA is the standard role that can be granted by an administrator to another administrator. It includes all system privileges and should only be granted to the most trusted and qualified of users. Assigning this role to a user enables the user to administer the database.

Roles of User SYS

- When you create an Oracle database, the user SYS is automatically created and granted the DBA role.
- All base tables and views for the database data dictionary are stored in the schema SYS. These base tables and views are critical for the operation of Oracle Database.
- To maintain the integrity of the data dictionary, tables in the SYS schema are manipulated only by the database. They should never be modified by any user or database administrator.
- Also, you should not create any tables in the schema of user SYS, although you can change the storage parameters of the data dictionary settings if necessary.

Analyzing Roles & Privileges to Users

- Assuming there are three departments in some Enterprise
    - Sales
    - Accounting
    - Human Development & Resource (HR)
- Employees are working as an external users in these departments
- Analyzing how Roles and Privileges are need to be decided
- Users Ali(Sales), Jamil(HR) and Nadeem (Accounting)



Lesson 174:Benefits of Roles-2

- Characteristics of Roles
- Roles with Synonyms
- Comparing Roles with Privileges

Characteristics of Roles

- Roles can be granted to and revoked from users with the same commands that are used to grant and revoke system privileges.
- Roles can be granted to any user or role. However, a role cannot be granted to itself and cannot be granted circularly.
- A role can consist of both system and object privileges.
- A role can be enabled or disabled for each user who is granted the role.
- A role can be required a password to be enabled.
- Each role name must be unique among existing usernames and role names.

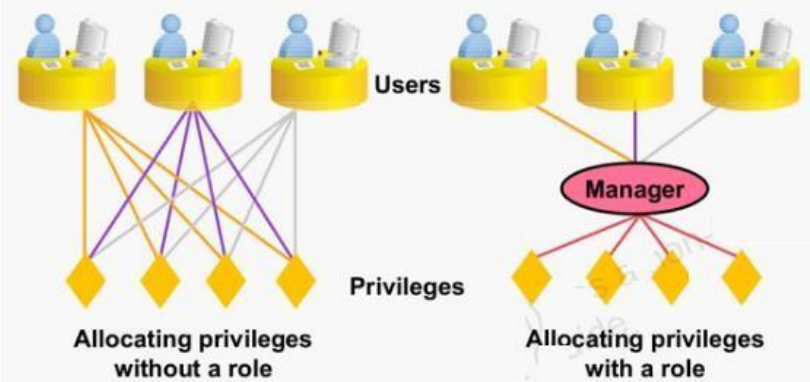
- Roles are not owned by anyone; and they are not in any schema.
- Roles have their description stored in the data dictionary.

Roles with Synonyms

- Roles and privileges are assigned to base table or view. Any Synonym will be granted as per base table or view.
- For example,

  --Table EMP owner gives grant: grant select on emp to u1;--run in u1's schema create synonym emp for scott.emp; select * from emp;

  Comparing Roles with Privileges



Lesson 175:Creating Roles

- SQL-Create Role
- Syntax of SQL-Create Role
- Roles with ADMIN option
- Syntax of SQL-Create Role using EM

SQL-Create Role

- Use the CREATE ROLE statement to create roles. You must have the CREATE ROLE system privilege to create roles. When you create a role that is not identified externally or by password, the role is granted with the ADMIN option.
- You need to decide set of privileges to put in one role and a user who will use those privileges by assigning just role.

Syntax of SQL-Create Role

Use the following command to create a role:

```
CREATE ROLE role [NOT IDENTIFIED | IDENTIFIED
  {BY password | EXTERNALLY | GLOBALLY | USING package}]
```

where:

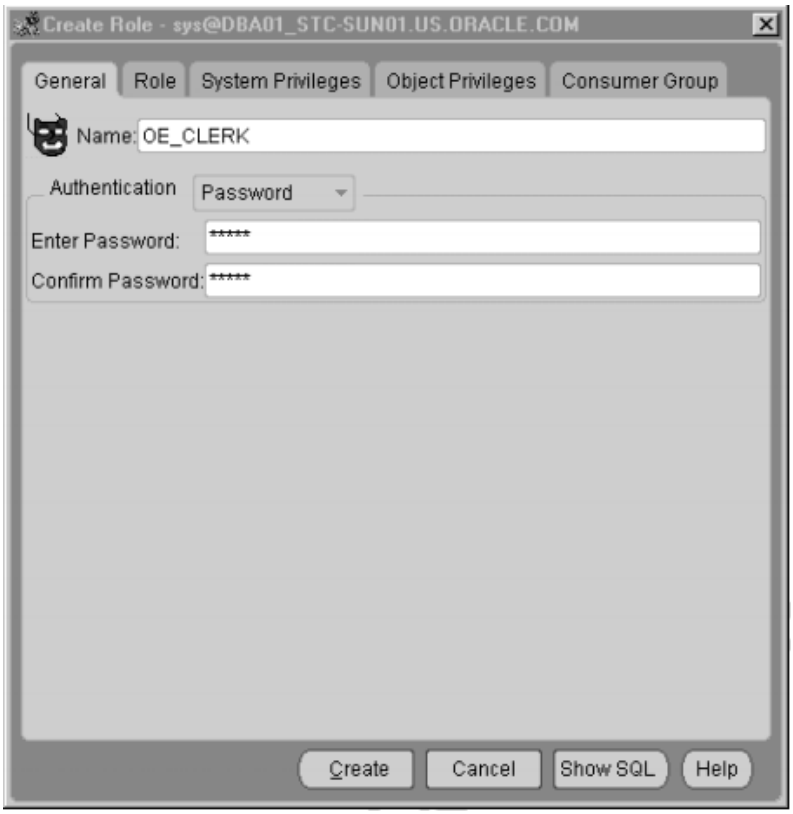
- `role`: Is the name of the role
- `NOT IDENTIFIED`: Indicates that no verification is required when enabling the role
- `IDENTIFIED`: Indicates that verification is required when enabling the role
- `BY password`: Provides the password that the user must specify when enabling the role
- `USING package`: Creates an application role, which is a role that can be enabled only by applications using an authorized package
- `EXTERNALLY`: Indicates that a user must be authorized by an external service (such as the operating system or a third-party service) before enabling the role
- `GLOBALLY`: Indicates that a user must be authorized to use the role by the enterprise directory service before the role is enabled with the `SET ROLE` statement, or at login.

Roles with ADMIN option

- Not identified: Create role oe_clerk;
- By password: Create role hr_clerk IDENTIFIED BY bonus;
- Identified externally:

Create role hr_manager identified Externally;

Syntax of SQL-Create Role using EM



Lesson 176:LPre-defined Roles & Assigning Roles

- Meaning of Pre-defined Roles & Assigning Roles
- Other Special Roles
- List of Pre-defined Roles & Assigning Roles
- Assigning Roles - An Enterprise Example

Meaning of Pre-defined Roles & Assigning Roles

- The roles listed are defined automatically for Oracle databases when you run database creation scripts.
- CONTENT, RESOURCE, and DBA roles are provided for backward compatibility to earlier versions of the Oracle server.
- The EXP_FULL_DATABASE and IMP_FULL_DATABASE roles are provided for convenience in using the Import and Export utilities.
- The roles DELETE_CATALOG_ROLE, EXECUTE_CATALOG_ROLE, and SELECT_CATALOG_ROLE are provided for accessing data dictionary views and packages.
- These roles can be granted to users who do not have the DBA role but who requires an access to the views and tables in the data dictionary.

Other Special Roles

- The Oracle server also creates other roles that authorize you to administer the database. On many operating systems, these roles are called OSOPER and OSDBA. Their names may be different on your operating system.
- Other roles are defined by SQL scripts provided with the database. For example, AQ_ADMINISTRATOR_ROLE provides privileges to administer advanced queuing. AQ_USER_ROLE is obsolete but is kept mainly for release 8.0 compatibility.
- SELECT * FROM DBA_SYS_PRIVS;

List of Pre-defined Roles & Assigning Roles

Role Name	Description
CONNECT, RESOURCE, DBA	These roles are provided for backward compatibility
EXP_FULL_DATABASE	Privileges to export the database
IMP_FULL_DATABASE	Privileges to import the database
DELETE_CATALOG_ROLE	DELETE privileges on data dictionary tables
EXECUTE_CATALOG_ROLE	EXECUTE privilege on data dictionary packages
SELECT_CATALOG_ROLE	SELECT privilege on data dictionary tables

Assigning Roles – An Enterprise Example

Empno	Name	DOB	CNIC	Address	City	TelNo	Deptno	EmpCard#

Empno	Name	DOB	CNIC	Address	City	TelNo	Salary	Bonus	Deptno

Empno	Name	DOB	CNIC	Address	City	TelNo	BloodGroup	Salary	Bonus	Deptno	EmpCard#

Lesson 177:Modifying & Align Roles

- Alter Role
- Alter Role - Sample SQLs
- Alter Role - Syntax
- Alter Role - EM

Alter Role

- ALTER ROLE modifies the authentication method.
- Modifying roles requires the ADMIN option or ALTER ANY ROLE privilege

Alter Role – Sample SQLs

- ALTER ROLE oe_clerk IDENTIFIED BY order;
- ALTER ROLE hr_clerk IDENTIFIED EXTERNALLY;
- ALTER ROLE hr_manager NOT IDENTIFIED;

Alter Role – Syntax

```
ALTER ROLE role {NOT IDENTIFIED | IDENTIFIED
 {BY password |USING package| EXTERNALLY | GLOBALLY }};
```

where:

- `role`: Is the name of the role
- `NOT IDENTIFIED`: Indicates that no verification is required when enabling the role
- `IDENTIFIED`: Indicates that verification is required when enabling the role
- `BY password`: Provides the password used when enabling the role
- `EXTERNALLY`: Indicates that a user must be authorized by an external service (such as the operating system or a third-party service) before enabling the role
- `GLOBALLY`: Indicates that a user must be authorized to use the role by the enterprise directory service before the role is enabled with the `SET ROLE` statement, or at login

Alter Role – EM

Lesson 178: Default Roles

- Default Roles
- Assigning & Disable Roles - Sample SQLs
- Establishing Default Roles

Assigning & Disable Roles – Sample SQLs

- GRANT oe_clerk TO scott;
- GRANT hr_clerk TO hr_manager;
- GRANT hr_manager TO scott WITH ADMIN OPTION;
- Disable role during session of user u1

SQL> SET ROLE NONE;

Establishing Default Roles

- **A user can be assigned many roles.**
- **A user can be assigned a default role.**
- **Limit the number of default roles for a user.**

```
ALTER USER scott
     DEFAULT ROLE hr_clerk, oe_clerk;
```

```
ALTER USER scott DEFAULT ROLE ALL;
```

```
ALTER USER scott DEFAULT ROLE ALL EXCEPT
     hr_clerk;
```

```
ALTER USER scott DEFAULT ROLE NONE;
```

Establishing Default Roles

A user can be assigned many roles. A default role is a subset of these roles that is automatically enabled when the user logs on. By default, all the roles are assigned to a user are enabled at logon without the need of a password. Limit the default roles for a user with the *ALTER USER* command.
The DEFAULT ROLE clause applies only to roles that have been granted directly to the user with a GRANT statement. The DEFAULT ROLE clause cannot be used to enable the following:
Roles not granted to the user.
Roles granted through other roles.
Roles managed by an external service (such as the operating system).

Use the following syntax to assign default roles to a user:

ALTER USER user *DEAFULT ROLE*

{role [,role]…| ALL [EXCEPT role [,role]…] | NONE}

Where:

User: is the name of the user who is granted the roles

Role: is the role to be made the default role for the user

Lesson 179: Application Roles

Application Roles - SQL

Application roles can be enabled only by authorized PL/SQL packages

- The USING package clause creates an application role

  CREATE ROLE admin_role

  IDENTIFIED USING hr.employee;

Application Roles

- The USING package clause in the CREATE ROLE statement creates an application role. An application role can be enabled only by applications by using an authorized PL/SQL package.
- Application developers do not need to secure a role by embedding passwords inside applications. Instead, they can create an application role and specify which PL/SQL package is authorized to enable the role.

Application Roles - SQL

- CREATE ROLE admin_role
- IDENTIFIED USING hr.employee;
- In this example, admin_role is an application role and the role can be enabled only by modules that are defined inside the hr.employee PL/SQL package.

Lesson 180:Enable & Disable Role

- Restrictions on Enable & Disable Role
- Specifying Roles to be Enabled
- Restrictions on Enable & Disable Role - Sample
- Restrictions on Enable & Disable Role – Syntax
- Disable a role to temporarily revoke the role from a user.
- Enable a role to grant it temporarily.
- The SET ROLE command enables and disables roles
- Default roles are enabled for a user at login.
- A password may be required to enable a role.
- Enable and disable roles to temporarily activate and deactivate the privileges associated with the roles.
- To enable a role, a role must first be granted to the user.
- When a role is enabled, the user can use the privileges granted to that role. If a role is disabled, the user cannot use the privileges associated with that role unless those privileges are granted directly to a user or to another role enabled for that user. Roles are enabled for a session. At the next session, the user's active roles revert to default roles.

Specifying Roles to be Enabled

- The SET command and the DBMS_SESSION.SET_ROLE procedure enable all of the roles included in the command and disable all other roles. Roles can be enabled from any tool or program that allows PL/SQL commands; however, a role cannot be enabled in a stored procedure.
- You can use the ALTER USER-DEFAULT ROLE command to indicate which roles will be enabled for a user at login. All other roles are disabled.
- A password may be required to enable a role. The password must be included in the SET ROLE command to enable the role.
- Default roles assigned to a user do not require a password; they are enabled at login, the same as a role without a password.

Restrictions on Enable & Disable Role

- A role cannot be enabled from a stored procedure, because this action may change the security domain (set of privileges) that allowed the procedure to be called in the first place.
- So, in PL/SQL, Role can be enabled and disabled in anonymous blocks and application procedures (for example, client Oracle Forms procedures), but not in stored procedures.
- IF a stored procedure contains the command SET ROLE, the following error is generated at run time: ORA-06565: cannot execute SET ROLE from within stored procedure

Restrictions on Enable & Disable Role - Sample

```
SET ROLE hr_clerk;
```

```
SET ROLE oe_clerk IDENTIFIED BY order;
```

```
SET ROLE ALL EXCEPT oe_clerk;
```

Restrictions on Enable & Disable Role - Syntax

```
SET ROLE {role [ IDENTIFIED BY password ]
   [, role [ IDENTIFIED BY password ]]
   | ALL [ EXCEPT role [, role ]     ]
   | NONE }
```

Lesson 181:Removing Roles

- Revoke Roles
- Revoke Roles from Users
- Revoke Roles using OEM
- Removing Roles
- Disable a role to temporarily revoke the role from a user.
- Enable a role to grant it temporarily.
- The SET ROLE command enables and disables roles
- Default roles are enabled for a user at login.
- A password may be required to enable a role.

Revoke Roles from Users

- To revoke a role from a user, use the SQL statement REVOKE. Any user with the admin option for a role can revoke the role from any other database user or role. Also, users with the GRANT ANY ROLE privilege can revoke any role.

```
REVOKE role [, role ]
  FROM {user|role|PUBLIC}
  [, {user|role|PUBLIC} ]
```

where

- role: Is the role to be revoked or the role from which roles are revoked.
- user: is the user from which the system privileges or roles are revoked.
- PUBLIC: Revokes the privilege or role from all users

Revoke Roles using OEM

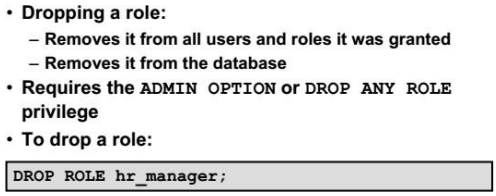
Using Oracle Enterprise Manager to Revoke a Role from a User

From the OEM Console:

- Navigate to Security>Users.
- Highlight the user for whom a role is to be revoked.
- Navigate to Roles Granted.
- Select the role to be revoked.
- Select Revoke from the right-mouse menu.
- Select Yes to confirm revocation.

Removing Roles

- To remove a role from the database, use the following syntax:
- SQL> DROP ROLE role;
- When you drop a role, the Oracle server revokes it from all users and roles to whom it has been granted and removes it from the database.
- In order to drop the role, you must have been granted the role with ADMIN OPTION or have the DROP ANY ROLE system privilege.



Lesson 182:Guidelines for Creating Roles

- Guidelines
- Guidelines for an Example
- Guidelines for Assigning Roles
- Guidelines for using Password and Default Roles
- Roles with Password

Guidelines

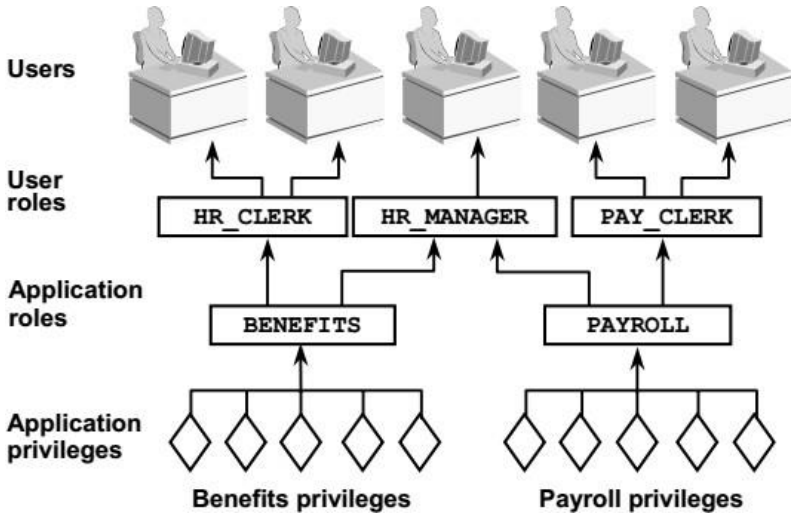
- Because a role includes the privileges that are necessary to perform a task, the role name is usually an application task or a job title.
- Our example uses both application tasks and job titles for role names.

Guidelines for an Example

Use the following steps to create, assign, and grant users roles:

- Create a role for each application task. The name of the application role corresponds to a task in the application, such as PAYROLL.
- Assign the privileges necessary to perform the task to the application role.
- Create a role for each type of user. The name of the user role corresponds to a job title, such as PAY_CLERK.
- Grant application roles to user's roles.
- Grant user's roles to users.
- If a modification to the application requires that the new privileges are needed to perform the payroll task, then the DBA only needs to assign the new privileges to the PAYROLL application role. All of the users that are currently performing this task will receive the new privileges.

Guidelines for Assigning Roles



Guidelines for using Password and Default Roles

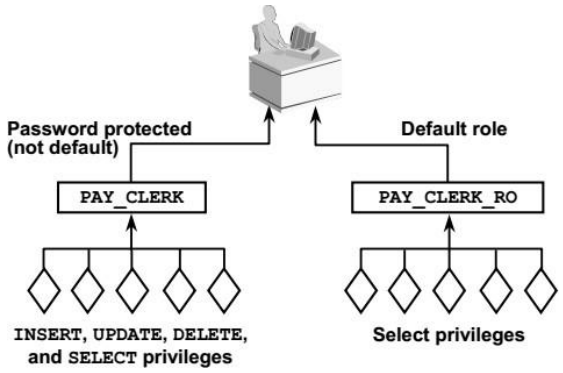
Passwords provide an additional level of security when enabling a role.

For example, the application might require a user to enter a password when enabling the PAY_CLERK role, because this role can be used to issue checks.

Passwords allow a role to be enabled only through an application. This technique is shown in the example in the slide.

- The DBA has granted the user two roles, PAY_CLERK and PAY_CLERK_RO.
- The PAY_CLERK has been granted all of the privileges that are necessary to perform the payroll clerk function.

Roles with Password



Lesson 183:Using Roles Data Dictionary Views

- Data Dictionary Views of Roles
- Roles' Views
- Privileges and Roles - An Example

Data Dictionary Views of Roles

- Information about roles can be obtained by querying the following SQL
- select table_name, comments
- from DICT
- where table_name like '%ROLE

Roles' Views

DBA_ROLES: All roles that exist in the database
DBA_ROLE_PRIVS: Roles granted to users and roles
ROLE_ROL_PRIVS: Roles that are granted to roles
DBA_SYS_PRIVS: System privileges granted to users and roles
ROLE_SYS_PRIVS: System privileges granted to roles
ROLE_TAB_PRIVS: Object privileges granted to roles
SESSION_ROLES: Roles that the user currently has enabled

Privileges and Roles – An Example

User is u1

Privileges are select, update for table EMP

Assigning Role EMP_SU

Assigning EMP_SU to user u1

Privileges and Roles – An Example

To user has which roles

```
SQL> select Grantee, granted_role
  2  from DBA_ROLE_PRIVS
  3  where rownum<=10;

GRANTEE                        GRANTED_ROLE
------------------------------ --------------------
U1                             EMP_SU
DBA                            OLAP_DBA
DBA                            XDBADMIN
DBA                            JAVA_ADMIN
DBA                            JAVA_DEPLOY
DBA                            OLAP_XS_ADMIN
DBA                            WM_ADMIN_ROLE
DBA                            SCHEDULER_ADMIN
DBA                            XDB_SET_INVOKER
DBA                            EXP_FULL_DATABASE
```

Lesson 184:Introduction to Data Migration & Data Loading

- Data Migration
- Data Migration - Source to Target Schema
- Data Migration Choices
- Data Migration & Staging Schema

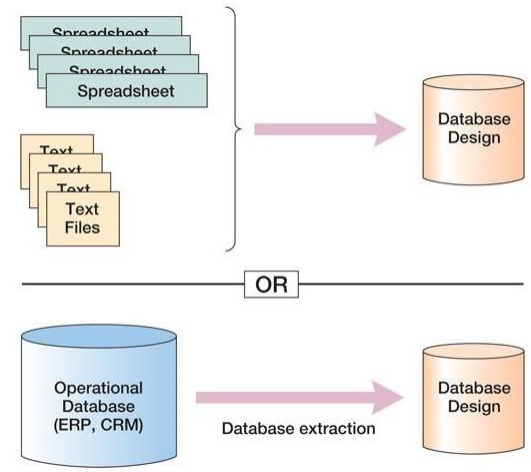
Data Migration

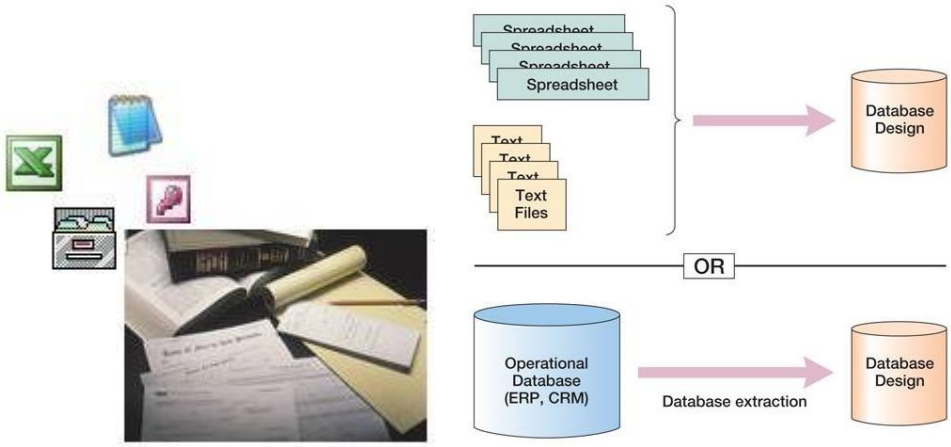
- Data migration is the process of transferring data from one storage system or computing environment to another.
- Non integrated/ distributed data can be integrated through data migration
- There are many reasons your enterprise might need to undertake a data migration project.
- For example, you might be replacing servers or storage devices or consolidating or decommissioning data center.
- Data migration is also an essential step in the overall process of migrating on-premises IT infrastructure to a cloud computing environment.

- Whether you are moving to a public cloud, private cloud, hybrid cloud, or multi-cloud environment, you will need to find a secure, cost-effective, and efficient method of migrating your data to its new storage location.

Data Migration – Source to Target Schema

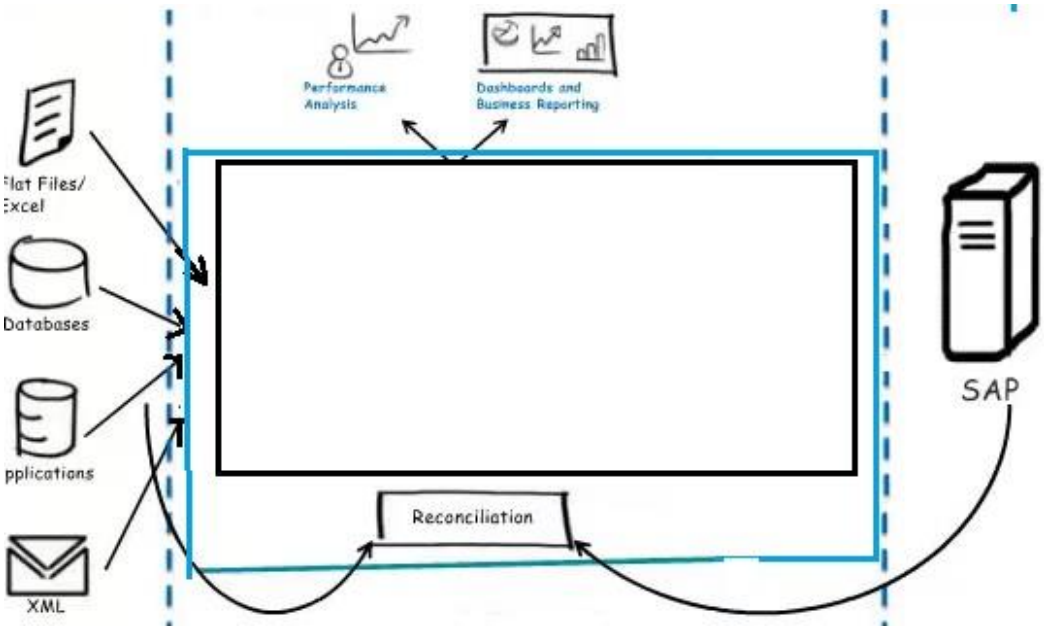
- Typically, the first step in the database migration process involves converting the source database's schema (if necessary) so that it's compatible with the target database. A database's schema is like a blueprint for how it is organized, controlling its logical architecture and structure. If the target database management system uses a data definition language (DDL) that is not compatible with the source's, the schema will need to be converted.





Data Migration Ways

Data Migration & Staging Schema



Lesson 185: Data Loading Methods

- Introduction
- SQL*Loader Method
- Direct Load Method
- Oracle Export and Oracle Import

Introduction

- Several methods are available for loading data into tables in an Oracle database. Of the methods available, Direct Load insert and SQL*Loader are discussed here.

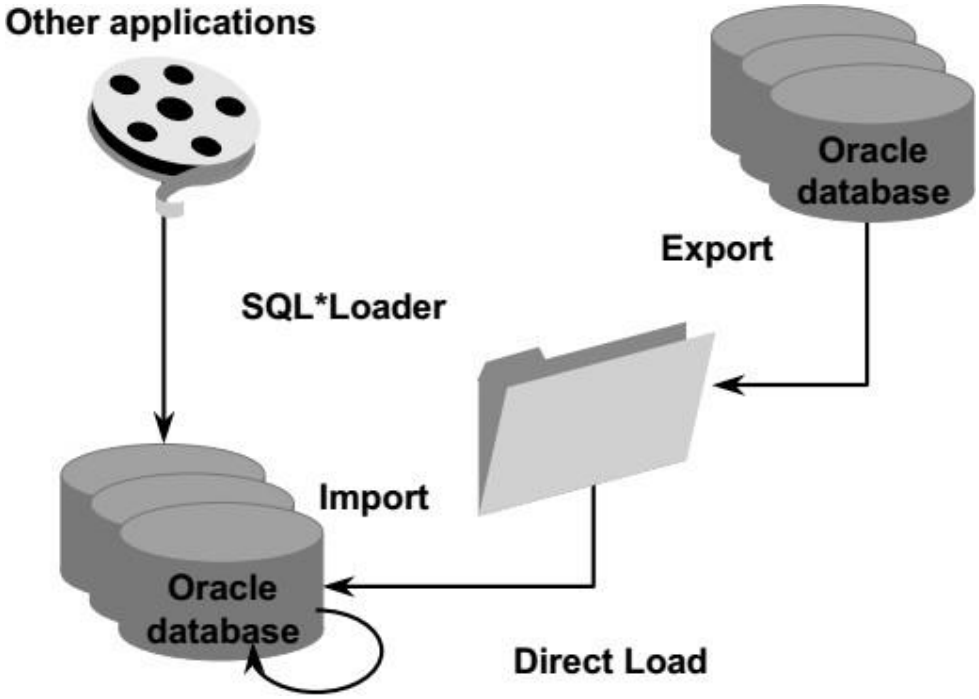
SQL*Loader Method

- SQL*Loader loads data from external files into tables of an Oracle database. It has a powerful data parsing engine that places little limitation on the format of the data in the data file.

Direct Load Method

- Direct Load insert can be used to copy data from one table to another table within the same database. It speeds up the insert operation, bypassing the database buffer cache and writing data directly into the data files.

Oracle Export and Oracle Import



Lesson 186:Direct & Parallel Load

- Direct Load
- Serial Direct Load - Example
- Parallel Direct Load
- Parallel Direct Load - Example

Direct Load

Direct Load insert can be performed in the following ways:

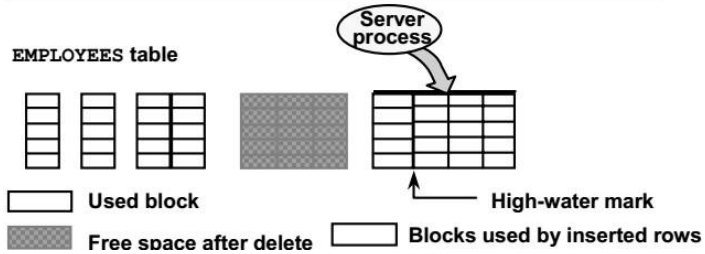
- Normal (Serially), or in parallel
- Into partition tables, non-partitioned tables, or single partitions of a table
- With or without logging of redo data
- Direct Load insert (serial or parallel) can only support the INSERT …
- SELECT syntax of an INSERT statement, not the INSERT Values syntax. The parallelism for INSERT
- SELECT is determined from either parallel hints or parallel table definition.
- Oracle provides syntax extensions that extend the scope of the INSERT SELECT statement, so that you can insert rows into multiple tables as part of a single DML statement.
- A Direct Load insert can be invoked by using the APPEND hint, as shown in following command:

INSERT /*+APPEND * / INTO [schema.] table [[NO] LOGGING]

- sub query;
- Where
- *schema*: Is the owner of the table
- *table*: Is the name of the table
- *sub-query*: Is the sub-query used to select the columns and rows for insert

Serial Direct Load - Example

```
INSERT /*+ APPEND */ INTO emp
NOLOGGING
SELECT * FROM t_employees;
COMMIT;
```



Lesson 187:SQL*Loader

- Introduction
- Files used by SQL*Loader
- Using SQL*Loader

Introduction

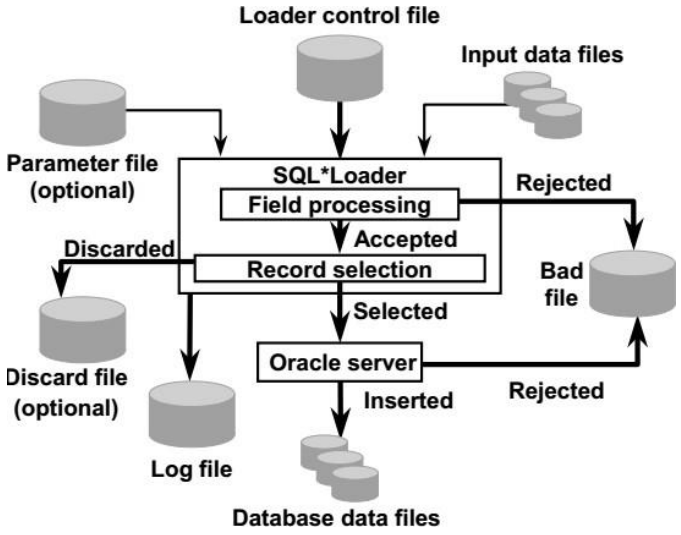
SQL*Loader loads data from external files into tables in an Oracle database. SQL*Loader has the following features:

- SQL*Loader can use one or more input files.
- Several input records can be combined into one logical record for loading.
- Input fields can be of fixed or variable lengths.
- Input data can be in any format: character, binary, packed decimal, date, and zoned decimal.
- Data can be loaded from different types of media such as disk, tape, or named pipes.
- Data can be loaded into several tables in one run.
- Options are available to replace or to append to existing data in the tables.
- SQL functions can be applied on the input data before the row is stored in the database.
- Column values can be auto generated based on rules. For example, a sequential key value can be generated and stored in a column.
- Data can be loaded directly into the table, bypassing the database buffer cache.

Files used by SQL*Loader

SQL*Loader uses the following files:

- Loader control file: Specifies the input format, output tables, and optional conditions that can be used to load only part of the records found in the input data files
- Input data files: Contain the data in the format defined in the control file
- Parameter file: Is an optional file that can be used to define the command line parameters for the load
- Log file: Is created by SQL*Loader and contains a record of the load
- Bad file: Is used by the utility to write the records that are rejected during the load.
- Discard file: Is a file that can be created, if necessary, to store all records that did not satisfy the selection criteria



Using SQL*Loader

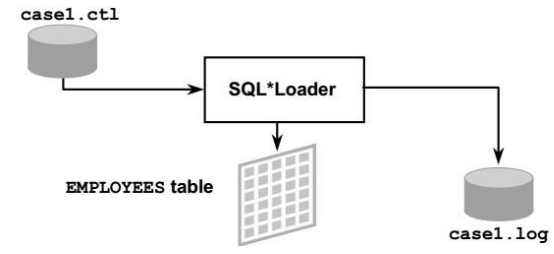
When you invoke SQL*Loader, you can specify parameters that establish session characteristics. Parameters can be entered in any order, optionally separated by commas.

You can specify values for parameters, or in some cases, you can accept the default without entering a value.

If you invoke SQL*Loader without specifying any parameters, SQL*Loader displays a Help screen that lists the available parameters and their default values.

Using SQL*Loader

```
$sqlldr hr/hr \
> control=case1.ctl \
> log=case1.log direct=Y
```



Lesson 188:SQL*Loader Control Files

- Introduction
- Three Sections of Control Files
- Sample Control File
- Sample Control File Explanation

Introduction

The loader control file tells SQL*Loader:

- Where to find the load data
- The data format
- Configuration details:
    - Memory management
    - Record rejection
    - Interrupted load handling details
- How to manipulate the data

The SQL*Loader control file is a text file that contains DDL instructions. DDL is used to control the following aspects of a SQL*Loader session:

- Where SQL*Loader finds the data to load
- How SQL*Loader expects that data to be formatted
- How SQL*Loader configures (memory management, rejecting records, interrupted load handling, and so on) as it loads the data
- How SQL*Loader manipulates the data being loaded

Three Sections of Control Files

- The first section contains session-wide information, for example:

    - Global options such as bind size, rows, records to skip, and so on

    - INFILE clauses to specify where the input data is located

    - How data is to be loaded

- The second section consists of one or more INTO TABLE blocks. Each of these blocks contain information about the table into which the data is to be loaded, such as the table name and the columns of the table.
- The third section is optional and, if present, contains input data.
- The second section consists of one or more INTO TABLE blocks. Each of these blocks contain information about the table into which the data is to be loaded, such as the table name and the columns of the table.
- The third section is optional and, if present, contains input data.

Sample Control File

```
1   -- This is a sample control file
2  LOAD DATA
3  INFILE 'SAMPLE.DAT'
4  BADFILE 'sample.bad'
5  DISCARDFILE 'sample.dsc'
6  APPEND
7  INTO TABLE emp
8  WHEN (57) = '.'
9  TRAILING NULLCOLS
10 (hiredate SYSDATE,
       deptno POSITION(1:2) INTEGER EXTERNAL(3)
        NULLIF deptno=BLANKS,
       job POSITION(7:14) CHAR TERMINATED BY WHITESPACE
       NULLIF job=BLANKS "UPPER(:job)",
       mgr POSITION(28:31) INTEGER EXTERNAL
       TERMINATED BY WHITESPACE, NULLIF mgr=BLANKS,
       ename POSITION(34:41) CHAR
```

Sample Control File Explanation

- This is how comments are entered in a control file. Comments can appear anywhere in the command section of the file, but they should not appear within the data.
- The LOAD DATA statement tells SQL*Loader that this is the beginning of a new data load. If you were continuing a load that had been interrupted in progress, you would use the CONTINUE LOAD DATA statement.
- The INFILE keyword specifies the name of a data file containing data that you want to load.

Control File Syntax Consideration

- The syntax is free-format
- Syntax is not case sensitive
- Comments extend from the two hyphen (--) that mark the beginning of the comment to the end of the line
- The CONSTANT keyword is reserved
- The syntax is free format. (Statements can extend over multiple lines.)
- It is case insensitive; however, strings enclosed in single or double quotation marks are taken literally, including case.
- In control file syntax, comments extend from the two hyphens (—) that mark the beginning of the comment to the end of the line. The optional third section of the control file is interpreted as data rather than as control file syntax; consequently, comments in this section are not supported.
- The CONSTANT keyword has special meaning to SQL*Loader and is therefore reserved. To avoid potential conflicts, do not use the word CONSTANT as a name for any tables or columns.

Lesson 189:Input & Data Files

- Introduction
- Fixed-Record Format
- Variable-Record Format
- Stream-Record Format

Introduction

SQL*Loader reads data from one or more files specified in the control file.

From SQL*Loader's perspective, the data in the data file is organized as records.

A data file can be in one of three formats:

- Fixed-record format
- Variable-record format
- Stream-record format

Fixed-Record Format

- Records in a file are of same byte
- Format is specified as
- INFILE <data file> "fix n"
- Following is a control file contents for two columns
- load data
- infile 'example.dat' "fix 10"
- into table example
- fields terminated by ' ' (col1, col2)
- example.dat:
- 0001, abcd
- 0002, fghi
- 0003, klmn

Variable-Record Format

- A file is in variable format when the length of each record in a character field is included at the beginning of each record in the data file.

  For example, specifying data file as

INFILE "datafile_name" "var n"n represents the number of bytes in the record length field

If n is not specified, SQL*Loader assumes a length of 6.

In example.dat, record length fields/ columns are 3 bytes long

```
load data
infile 'example.dat' "var 3"
into table example
fields terminated by ',' optionally enclosed by '"'
(col1 char(5),col2 char(7))
example.dat:
009hello,cd,
010world,im,
012my,name is,
```

Stream-Record Format

A file is in stream-record format when records are not specified by size; instead SQL*Loader forms records by scanning for the record terminator. It is flexible but it has performance issues.

For example, specifying data file as

INFILE <datafile_name> ["str terminator_string"]

Stream-Record Format

Non printable characters will be

\n new line

\t horizontal tab

\f form feed

\v vertical tab

\r carriage return

In example.dat, data loading with stream-record format

```
load data
infile 'example.dat' "str '|\n'"
into table example
fields terminated by ',' optionally enclosed by '"'
(col1 char(5),
col2 char(7))
example.dat:
hello,world,|
james,bond,|
```

Lesson 190:Logical Records & Loading Methods

- Logical Records
- Using CONCATENATE to Assemble Logical Records
- Using CONTINUEIF to Assemble Logical Records
- Conventional Path Load
- Direct Path Load

Loading Methods

- SQL*Loader can be instructed to follow one of the following two logical record-forming strategies:
- Combine a fixed number of physical records to form each logical record
- Combine physical records into logical records while a certain condition is true

  SQL*Loader organizes the input data into physical records, according to the specified record format. By default, a physical record is a logical record. However, for added flexibility, SQL*Loader can be instructed to combine a number of physical records into a logical record. SQL*Loader can do

- Combine fixed number of physical records to form each logical record
- Combine physical records into logical records while a certain condition is true

Using CONCATENATE to Assemble Logical Records

- CONCATENATE is used when SQL*Loader should always add the same number of physical records to form one logical record. For example,
- CONCATENATE integer
- Means number of physical records to combine
- CONTINUEIF must be used if the number of physical records to be continued varies. The CONTINUEIF keyword is followed by a condition that is evaluated for each physical record as it is read.
- For example, two records might be combined if there was a pound sign (#) in character position 80 of the first record.

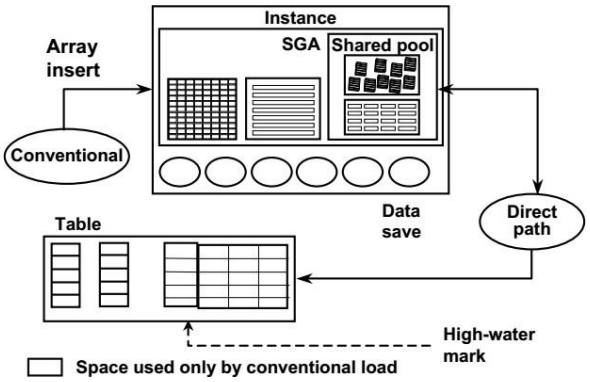
Conventional Path Load

- Conventional path Load builds an array of rows to be inserted and uses the SQL INSERT statement to load the data. During conventional path loads, input records are parsed based on field specifications, and an array of records is built and inserted into the table specified in the control file.
- Records that do not conform to the field specifications are rejected and those records that do not satisfy the selection criteria are discarded.
- Conventional path loads can be used to load data into both the clustered and unclustered tables.
- Redo generation is controlled by the logging attribute for the table that is being loaded.
- Conventional path loads can be used to load data into both the clustered and unclustered tables.
- Redo generation is controlled by the logging attribute for the table that is being loaded.

Direct Path Load

- A Direct path load builds blocks of data in memory and saves these blocks directly into the extents allocated for the table being loaded. Online redo log file entries are not generated unless the database is in ARCHIVELOG mode.
- Direct load path is generally faster than conventional path load.

Loading Methods



Lesson 191:Direct & Conventional Path Loads

- Method of Saving Data
- Comparison of Conventional and Direct Path Loads
- Logging Changes
- Enforcing Constraints
- Parallel Direct Paths
- Enforcing Constraints

Method of Saving Data

- Conventional path loads use SQL processing and database COMMITS for saving data. The insertion of an array of records is followed by a commit operation. Each data load might involve several transactions.
- Direct path loads use data saves to write blocks of data to Oracle data files. The following features differentiate a data save from a COMMIT:
- During a data save, only full database blocks are written to the database
- The blocks are written after the high-water mark of the table
- After a data save, the high-water mark is moved
- Internal resources are not released after a data save

- A data save does not end the transaction
- Indexes are not updated at each data save

Comparison of Conventional and Direct Path Loads

Conventional Load	Direct Path Load
Uses COMMITs to make changes permanent	Uses data saves
Redo entries always generated	Generates redo only under specific conditions
Enforces all constraints	Enforces only primary key, unique, and NOT NULL
INSERT triggers fire	INSERT triggers do not fire
Can load into clustered tables	Cannot load into clustered tables
Other users can make changes to tables	Other users cannot make changes to tables

Logging Changes

Conventional path loading generates redo entries just as any DML statement. When using a Direct path load, redo entries are not generated if:

The database is in NOARCHIVELOG mode The database is in ARCHIVELOG mode, but logging is disabled. Logging can be disabled by setting the NOLOGGING attribute for the table or by using the UNRECOVERABLE clause in the control file

Enforcing Constraints

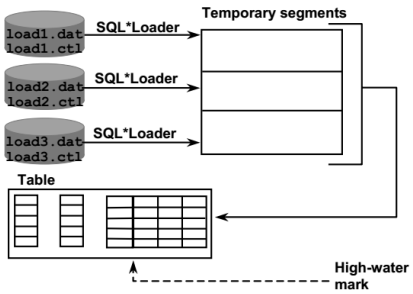
During Direct path loads, the constraints are handled as follows:

- NOT NULL constraints are checked when arrays are built.
- FK and CHECK constraints are disabled
- PK and unique constraints are checked during and at the end of the run and may be disabled

Parallel Direct Paths

- Multiple SQL*Loader sessions improve the performance of a Direct path load. Three models of concurrency can be used to minimize the time required for data loading:
- Parallel conventional path loads
- Intersegment concurrency with Direct path load method
- Intra segment concurrency with Direct path load method
- Concurrent conventional path
- When trigger or integrity constraints pose a problem
- Intersegment concurrency with Direct path load method
- Loading of different objects such as tables and partitions of same table.
- Intra segment concurrency with Direct path load method
- Multiple Direct path load session concurrently into same table or partition

Enforcing Constraints



Lesson 192:Data Conversion

- Introduction
- SQL*Loader Rejects
- Discarded or Rejected Records

Introduction

During a conventional path load, data fields in the data file are converted into columns in the database in two steps:

- The field specifications in the control file are used to interpret the format of the data file and convert it to a SQL INSERT statement using that data.
- The Oracle database server accepts the data and executes the INSERT statement to sore the data in the database

SQL*Loader Rejects

When input format is invalid

- When delaminated field exceeds its maximum length
- All bad records are saved in bad file
- Oracle determines row is valid or not.
- If row or record is rejected, SQL*Loader puts it in the bad file
- Why row can be rejected?

(unique or not null constraints)

Discarded or Rejected Records

Bad file:

- **SQL*Loader rejects records when the input format is invalid.**
- **If the Oracle database finds that the row is invalid, then the record is rejected and SQL*Loader puts it in the bad file.**

Discard file:

- **This can be used only if it has been enabled.**
- **This file contains records that were filtered out because they did not match any record-selection criteria specified in the control file.**

Lesson 193:Log Files & Steps to adopt

Log File Contents

- Header Information
- Global Information
- Table Information
- Data File Information
- Table Load Information
- Summary Statistics
- Additional statistics for Direct path loads and multithreading Information

Header Information

- Date of the run
- Software version number

Global Information

- Name of all input/ output file
- Echo of command-line arguments
- Continuation character specification

Table Information

- Table Name
- Load Conditions
- INSERT, APPEND or REPLACE specifications
- Columns, length, data type and deliminator

Data File Information

- SQL*Loader and Oracle data record errors

Log Files & Steps to adopt

Table Load Information

- Number of rows loaded, rejected, discarded, Null fields discarded

Summary Statistics

- Amount of space used for bind array
- Cumulative load statistics for data files, skipped, read and rejected records
- Additional statistics for Direct path loads and multithreading Information
- Direct path load of a partitioned table reports per-partition statistics
- Conventional-path load cannot report per-partition statistics

Lesson 194:Sample Example to Load Data-1

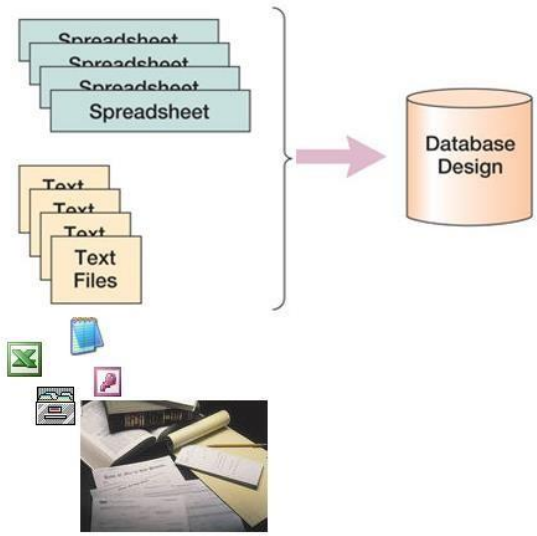
Example's Contents

- Data Loading
- Data Migration – From File to Database
- Example of Data Loading

Data Loading

- Loading data from files
- Files can be comma delaminated
- Files can be in XML format
- Binary files (Oracle to Oracle)

Data Migration – Files to Database



Example of Data Loading

```
Login scott
create table TEMP_EMP
(Empno Number(4),
Ename  VARCHAR2(50),
Salary Number(8)
);

c:\test\control.ctl
load data
infile 'c:\test\DBA-EXP-194.csv'
truncate
into table scott.temp_emp
fields terminated by ','
(empno, ename, salary)

cd E:\app\Haider\product\11.2.0\dbhome_1\BIN
c:\test>sqlldr control=c:\test\control.ctl
data=c:\test\DBA-EXP-194.csv log=c:\test\ulcase2.log
```

Example of Data Loading

```
E:\app\Haider\product\11.2.0\dbhome 1\BIN>sqlldr control=c:\test\control.ctl
                        data=c:\test\DBA-EXP-194.csv log=c:\test\ulcase2.log
Username:scott
Password:
SQL*Loader: Release 11.2.0.1.0 - Production on Wed Nov 4 19:13:30 2020

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Commit point reached - logical record count 6

E:\app\Haider\product\11.2.0\dbhome 1\BIN>
```

Example of Data Loading

```
SQL> create table TEMP_EMP
  2  (Empno Number(4),
  3  Ename  VARCHAR2(50),
  4  Salary Number(8)
  5  );

Table created.

SQL> select * from temp_emp;

     EMPNO ENAME                                              SALARY
---------- -------------------------------------------------- ----------
      1001 Ejaz                                                 5000
      1002 Ali                                                  3500
      1003 Qadeer                                               3400
      1004 Yaseen                                               4000
      1005 Asim                                                 3400
```

Camtesia

Lesson 195:Sample Example to Load Data-2

Example's Contents

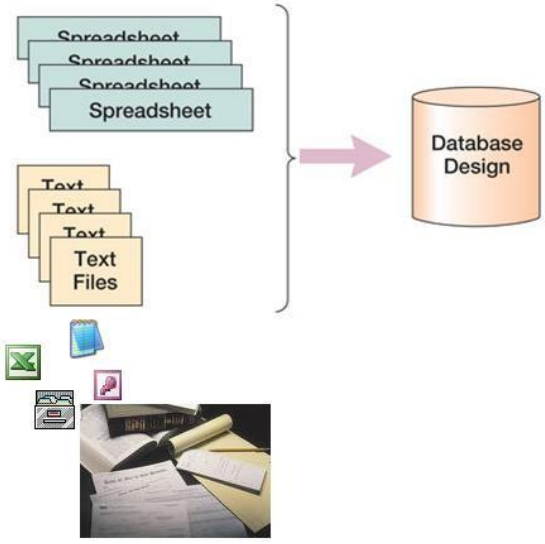
- Material's Data Loading
- Data Migration - Files to Database
- Reasons to Load Data into TEMP_Material
- Required Files Data Loading of Material
- Data Loading Files with Contents

Material's Data Loading

Data is comma delaminated regarding material or products

Need to load into table Material via TMP_Material

Data Migration – Files to Database



Reasons to Load Data into TEMP_Material

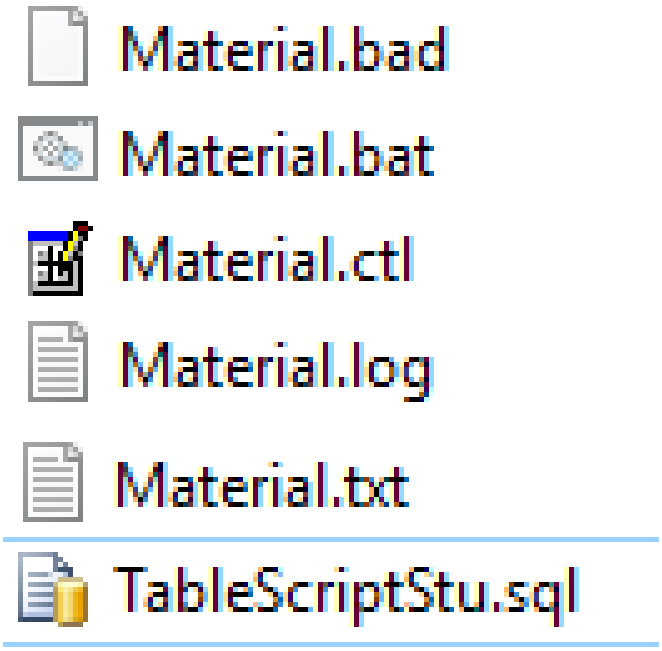
Try to avoid to load data into bad files

Format of data

For example $ symbol is given with numeric currency column

Size of column may less in the target table

Required Files Data Loading of Material

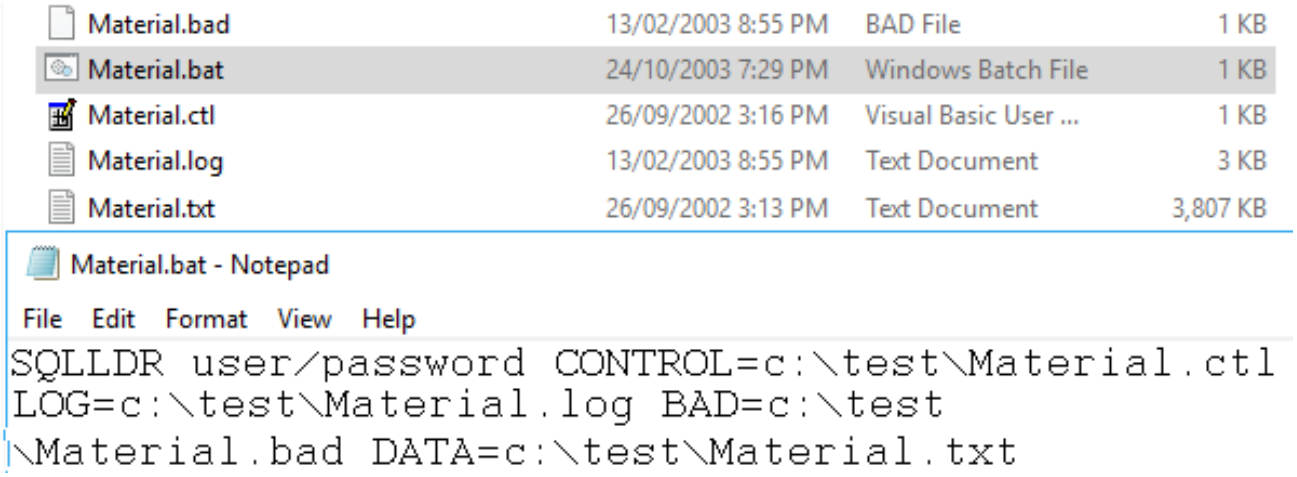


Material.bat - Notepad

File Edit Format View Help

```
SQLLDR user/password CONTROL=c:\test\Material.ctl LOG=c:\test\Material.log BAD=c:\test
\Material.bad DATA=c:\test\Material.txt
```

Data Loading Files with Contents



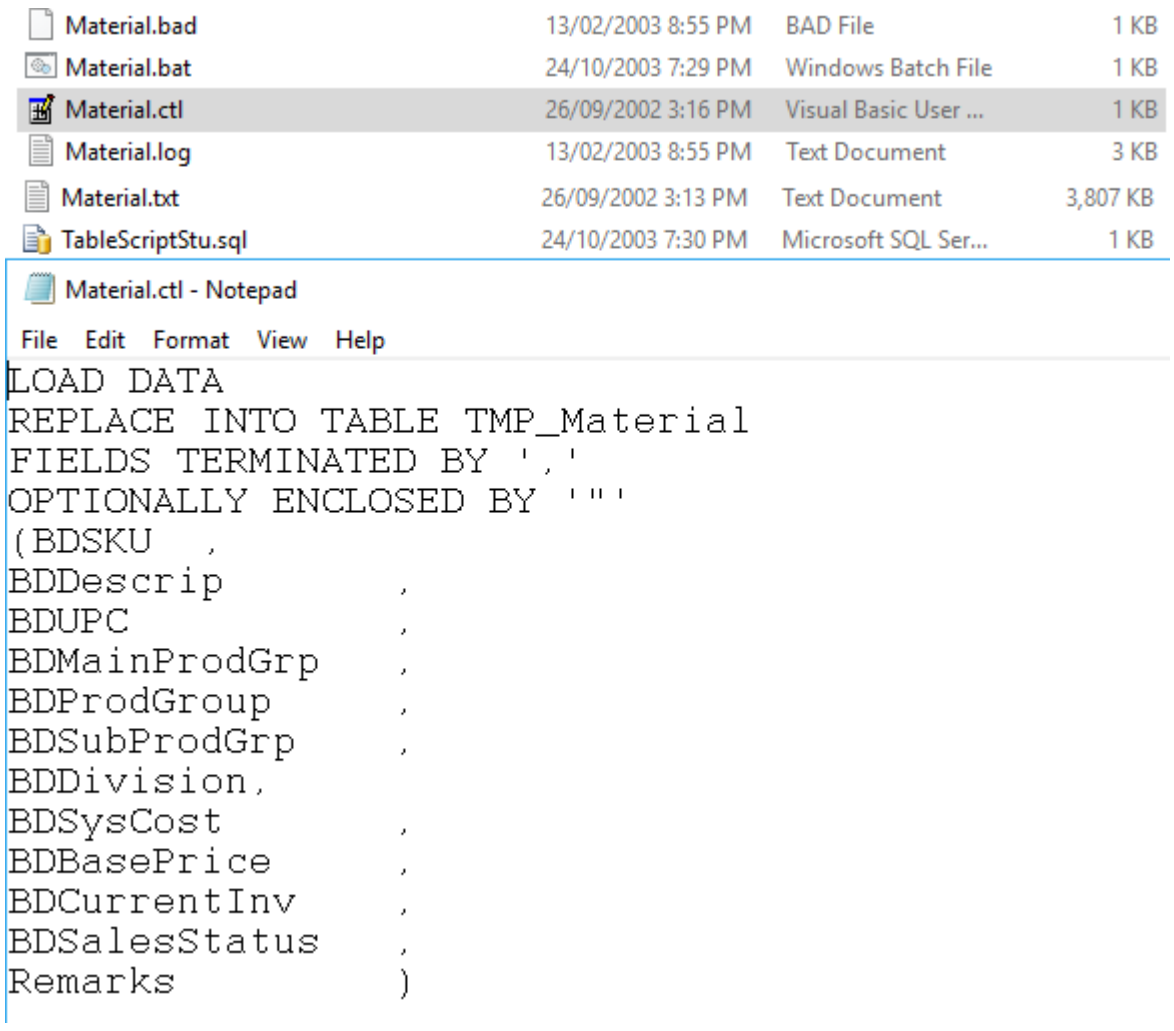
Material.bad	13/02/2003 8:55 PM	BAD File	1 KB
Material.bat	24/10/2003 7:29 PM	Windows Batch File	1 KB
Material.ctl	26/09/2002 3:16 PM	Visual Basic User ...	1 KB
Material.log	13/02/2003 8:55 PM	Text Document	3 KB
Material.txt	26/09/2002 3:13 PM	Text Document	3,807 KB

Material.bat - Notepad

File Edit Format View Help

```
SQLLDR user/password CONTROL=c:\test\Material.ctl
LOG=c:\test\Material.log BAD=c:\test
\Material.bad DATA=c:\test\Material.txt
```

Data Loading Files with Contents



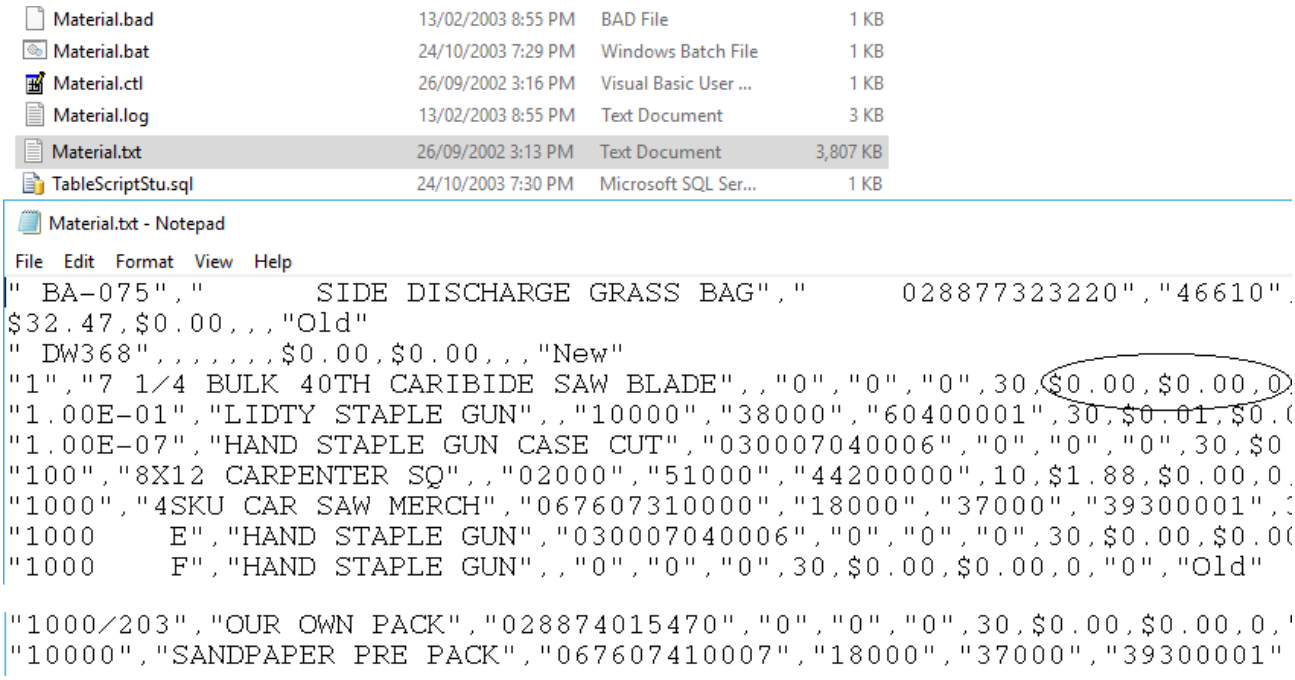
Material.bad	13/02/2003 8:55 PM	BAD File	1 KB
Material.bat	24/10/2003 7:29 PM	Windows Batch File	1 KB
Material.ctl	26/09/2002 3:16 PM	Visual Basic User ...	1 KB
Material.log	13/02/2003 8:55 PM	Text Document	3 KB
Material.txt	26/09/2002 3:13 PM	Text Document	3,807 KB
TableScriptStu.sql	24/10/2003 7:30 PM	Microsoft SQL Ser...	1 KB

Material.ctl - Notepad

File Edit Format View Help

```
LOAD DATA
REPLACE INTO TABLE TMP_Material
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
(BDSKU   ,
BDDescrip       ,
BDUPC           ,
BDMainProdGrp   ,
BDProdGroup     ,
BDSubProdGrp    ,
BDDivision,
BDSysCost       ,
BDBasePrice     ,
BDCurrentInv    ,
BDSalesStatus   ,
Remarks         )
```

Data Loading Files with Contents



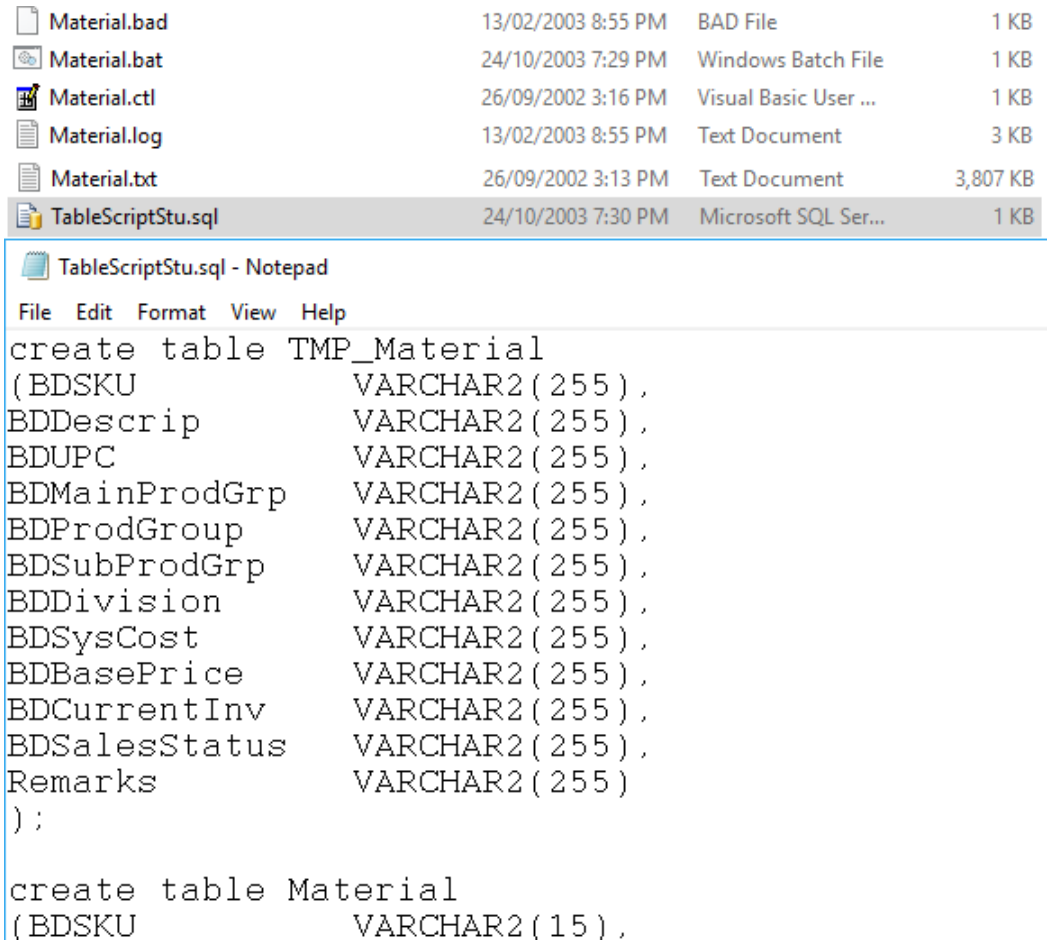
Material.bad	13/02/2003 8:55 PM	BAD File	1 KB
Material.bat	24/10/2003 7:29 PM	Windows Batch File	1 KB
Material.ctl	26/09/2002 3:16 PM	Visual Basic User ...	1 KB
Material.log	13/02/2003 8:55 PM	Text Document	3 KB
Material.txt	26/09/2002 3:13 PM	Text Document	3,807 KB
TableScriptStu.sql	24/10/2003 7:30 PM	Microsoft SQL Ser...	1 KB

Material.txt - Notepad

File Edit Format View Help

```
" BA-075","      SIDE DISCHARGE GRASS BAG","     028877323220","46610",
$32.47,$0.00,,,"Old"
" DW368",,,,,,,$0.00,$0.00,,,"New"
"1","7 1/4 BULK 40TH CARIBIDE SAW BLADE",,"0","0","0",30,$0.00,$0.00,0,
"1.00E-01","LIDTY STAPLE GUN",,"10000","38000","60400001",30,$0.01,$0.(
"1.00E-07","HAND STAPLE GUN CASE CUT","030007040006","0","0","0",30,$0
"100","8X12 CARPENTER SQ",,"02000","51000","44200000",10,$1.88,$0.00,0,
"1000","4SKU CAR SAW MERCH","067607310000","18000","37000","39300001",3
"1000    E","HAND STAPLE GUN","030007040006","0","0","0",30,$0.00,$0.0(
"1000    F","HAND STAPLE GUN",,"0","0","0",30,$0.00,$0.00,0,"0","Old"

"1000/203","OUR OWN PACK","028874015470","0","0","0",30,$0.00,$0.00,0,'
"10000","SANDPAPER PRE PACK","067607410007","18000","37000","39300001"
```

Data Loading Files with Contents



Material.bad	13/02/2003 8:55 PM	BAD File	1 KB
Material.bat	24/10/2003 7:29 PM	Windows Batch File	1 KB
Material.ctl	26/09/2002 3:16 PM	Visual Basic User ...	1 KB
Material.log	13/02/2003 8:55 PM	Text Document	3 KB
Material.txt	26/09/2002 3:13 PM	Text Document	3,807 KB
TableScriptStu.sql	24/10/2003 7:30 PM	Microsoft SQL Ser...	1 KB

TableScriptStu.sql - Notepad

File Edit Format View Help

```
create table TMP_Material
(BDSKU            VARCHAR2(255),
BDDescrip         VARCHAR2(255),
BDUPC             VARCHAR2(255),
BDMainProdGrp     VARCHAR2(255),
BDProdGroup       VARCHAR2(255),
BDSubProdGrp      VARCHAR2(255),
BDDivision        VARCHAR2(255),
BDSysCost         VARCHAR2(255),
BDBasePrice       VARCHAR2(255),
BDCurrentInv      VARCHAR2(255),
BDSalesStatus     VARCHAR2(255),
Remarks           VARCHAR2(255)
);

create table Material
(BDSKU            VARCHAR2(15),
```

Camtesia

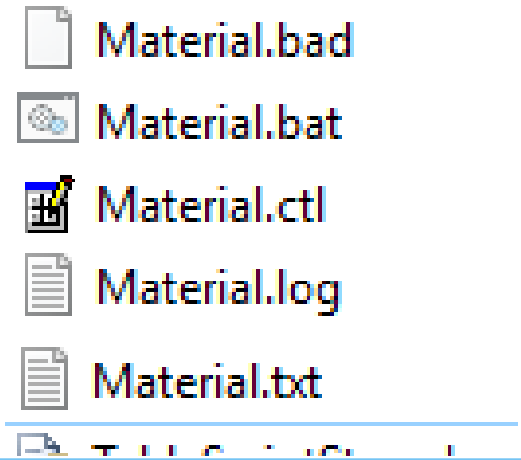
Lesson 196:Sample Example to Load Data-3

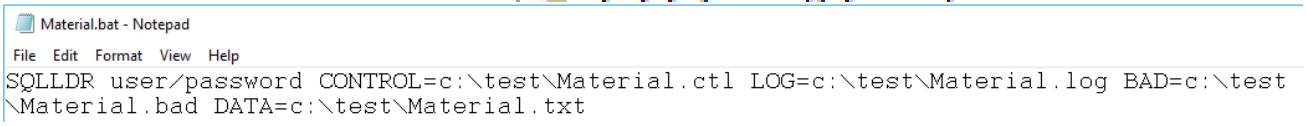
Example's Contents

- Executing Data Loading for Material (for Slide#195)
- Data Migration - Files to Database

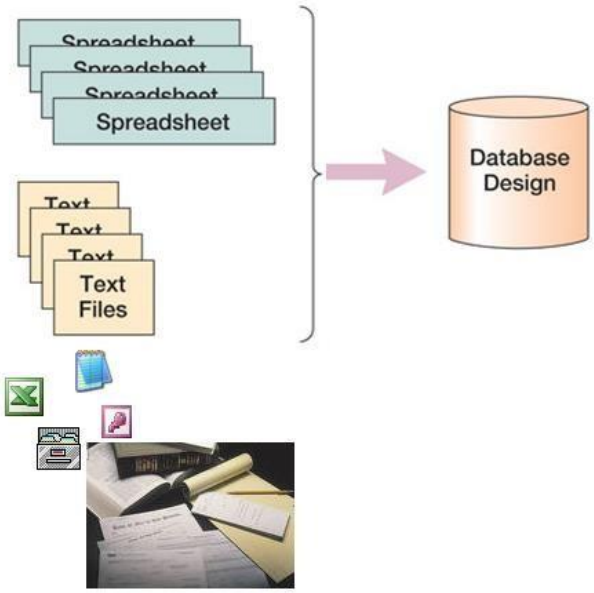
Executing Data Loading for Material

We will execute SQLLDR command to load data of material into scott's schema





Data Migration – Files to Database



Camtesia

Lesson 197:Sample Example to Load Data-4

Data Loading Tasks in Schema

- Loading data from TEMP to actual Material Table
- Tasks to load data from TEMP to actual Material Table

Loading data from TEMP to actual Material Table

- You have data in TMP_Material and you want to move or copy data into Material Table

Tasks to load data from TEMP to actual Material Table

- Use INSERT INTO … SELECT command
- Understand an error
- See where data discrepancies are found
- Run SQL UPDATE statements to fix discrepancies

Camtesia

Lesson 198:Sample Example to Load Data-5

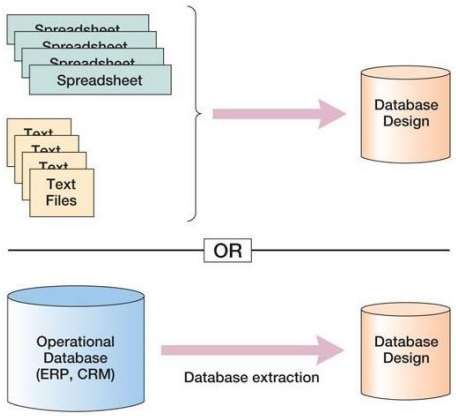
Data Loading Tasks using MS Access

- Tasks to Connect MS Access & Oracle
- Choosing MS Access by Connecting with Oracle Schema
- Link Tables of MS Access & Oracle

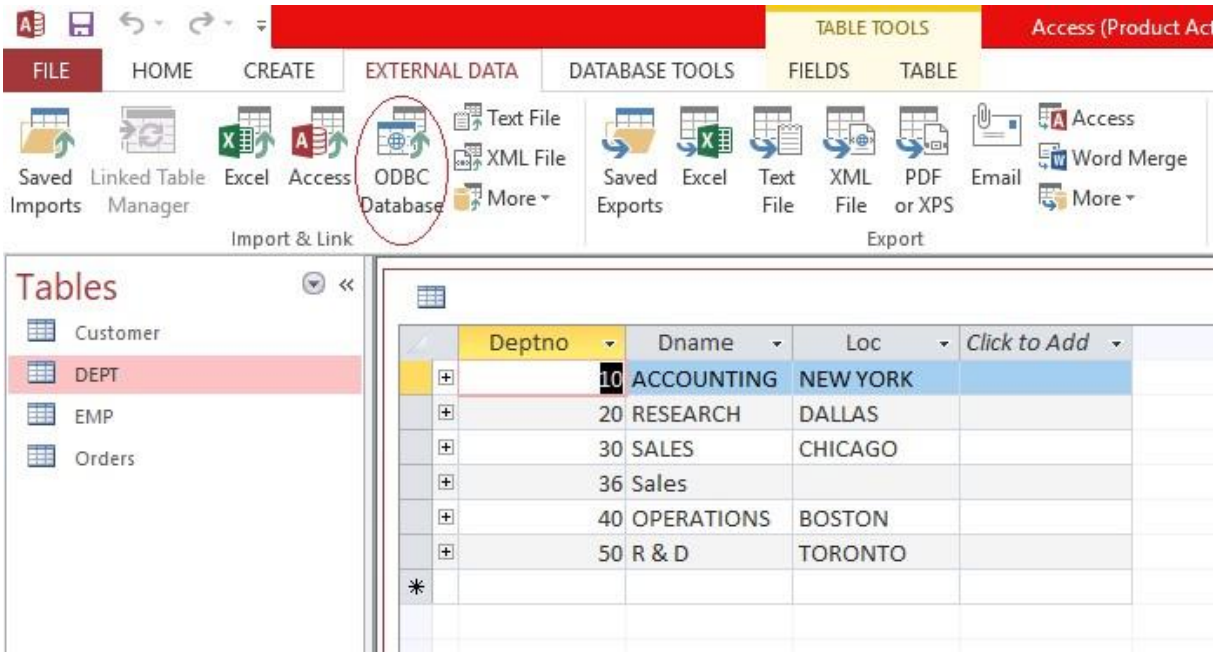
Tasks to Connect MS Access & Oracle

- Create ODBC data source of Oracle with use scott
- Open sample MS Access File and connect it with ODBC source user scott
- Link or data import is possible in MS Access
- Data can be inserted from MS Access into Oracle schema scott

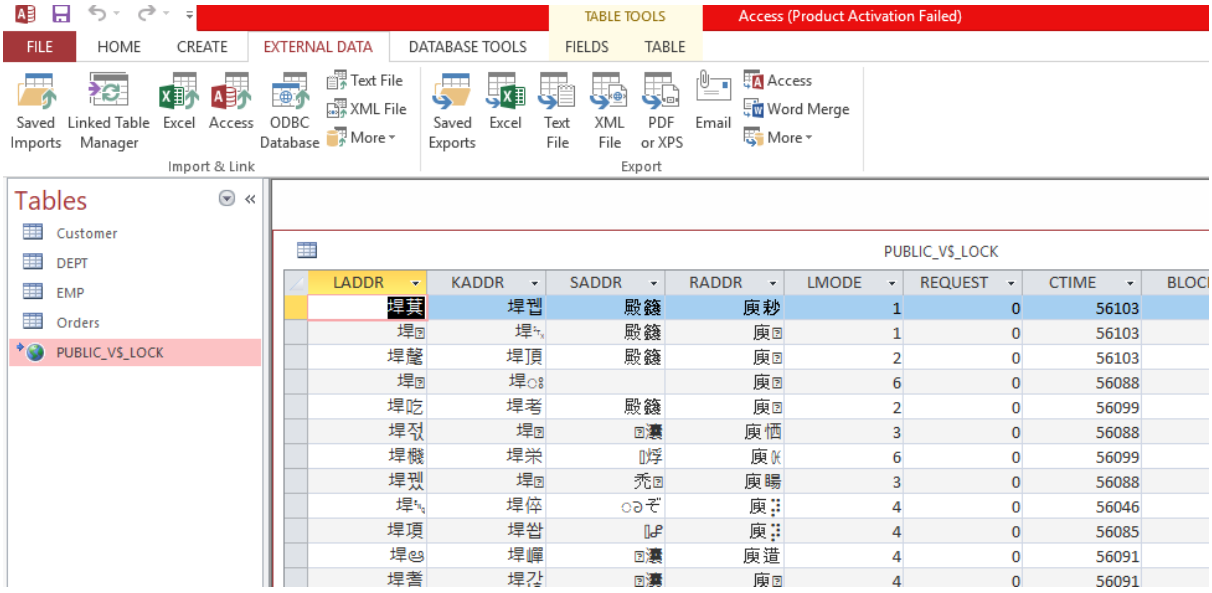
Choosing MS Access by Connecting with Oracle Schema



Link Tables of MS Access & Oracle Schema



Link Tables of MS Access & Oracle Schema



Camtesia

Lesson 199:Sample Example to Load Data-6

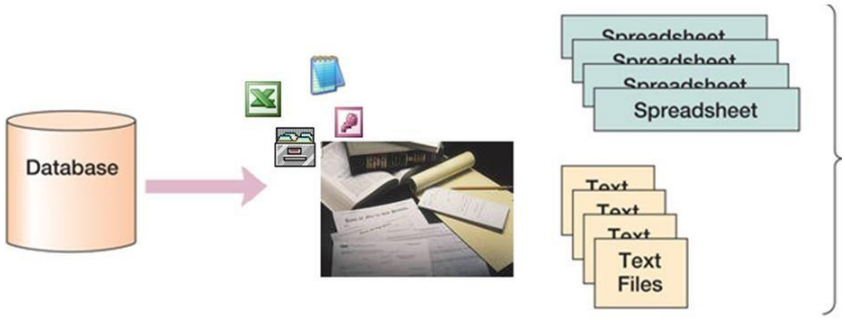
Data Loading Tasks for Schema to Schema/ Files using Text Files

- Preparing Spool File of Schema's Table using SQL
- Data Load from Oracle Schema into Files

Preparing Spool File of Schema's Table using SQL

- In Scott schema
    - Spool c:\test\emp.txt
    - Select … from EMP;
    - Spool off
- Spool file contains comma delaminated rows
- File can be used to load data into any DBMS

Data Load from Oracle Schema into Files



Camtesia

Lesson 200:Sample Example to Load Data-7

Data Preparation for Tasks using XML data

- Preparing Spool File of Schema's Table using XML SQL
- Sample SQL using XML Elements
- Data Load from Oracle Schema into Files
- Output Results of Sample XML SQLs

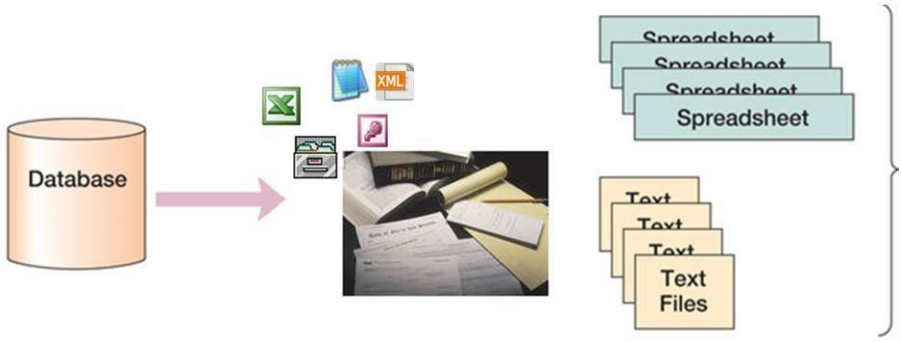
Preparing Spool File of Schema's Table using XML SQL

- In Scott schema
    - Spool c:\test\emp.txt
    - XML: Select … from EMP;
    - Spool off
- Spool file contains XML-taged rows
- File can be used to load data into any DBMS

Sample SQL using XML Elements

```
select xmlelement(name "emp",
      xmlelement(name "eno",empno),
      xmlelement(name "names",ename)) XML_output
from emp
/
create table XML1 as
select xmlelement("EMP",
xmlelement("empno",e.empno),
xmlelement("ename",e.ename),
xmlelement("job", e.job),
    xmlelement("mgr", e.mgr),
xmlelement("mgr1", e.mgr)) xml_desc
from emp e
 where rownum<=1;
```

Data Load from Oracle Schema into Files



Output Results of Sample XML SQLs

```
SQL> select xmlelement(name "emp",
  2         xmlelement(name "eno",empno),
  3         xmlelement(name "names",ename)) XML_output
  4  from emp
  5  /

XML_OUTPUT
--------------------------------------------------------------
<emp><eno>1234</eno><names></names></emp>
<emp><eno>8822</eno><names>ABC</names></emp>
<emp><eno>3513</eno><names>XYZ</names></emp>
<emp><eno>7369</eno><names>SMITH</names></emp>
<emp><eno>7499</eno><names>ALLEN</names></emp>
<emp><eno>7521</eno><names>WARD</names></emp>
<emp><eno>7566</eno><names>JONES</names></emp>
<emp><eno>7654</eno><names>MARTIN</names></emp>
<emp><eno>7698</eno><names>BLAKE</names></emp>
<emp><eno>7782</eno><names>CLARK</names></emp>
<emp><eno>7788</eno><names>SCOTT</names></emp>
```

```
SQL> create table XML1 as
  2  select xmlelement("EMP",
  3         xmlelement("empno",e.empno),
  4         xmlelement("ename",e.ename),
  5         xmlelement("job", e.job),
  6         xmlelement("mgr", e.mgr),
  7         xmlelement("mgr1", e.mgr)) xml_desc
  8  from emp e
  9  where rownum<=1;

Table created.

SQL> select * from XML1;

XML_DESC
--------------------------------------------------------------------------------
<EMP><empno>1234</empno><ename></ename><job></job><mgr></mgr><mgr1></mgr1></EMP>
```

Camtesia

Lesson 201: Backup & Recovery Strategies

- Backup & Recovery Concepts
- Physical & Logical Backup
- Backup and recovery is the set of concepts, procedures, and strategies involved in protecting the database against data loss caused by media failure or users errors.
- In general, the purpose of a backup and recovery strategy is to protect the database against data loss and reconstruct lost data.
- A backup is a copy of data. A backup can include crucial parts of the database such as data files, the server parameter file, and control file. A sample backup and recovery scenario is a failed disk drive that causes the loss of a data file.
- If a backup of the lost file exists, then you can restore and recover it. Media recovery refers to the operations involved in restoring data to its state before the loss occurred.

Physical & Logical Backup

- Backups can be divided into physical backups and logical backups.
- Physical backups are backups of the physical files used in storing and recovering your database, such as datafiles, control files, and archived redo logs. Ultimately, every physical backup is a copy of files storing database information to some other location, whether on disk or some offline storage such as tape.
- Logical backups contain logical data (for example, tables or stored procedures) exported from a database with an Oracle export utility and stored in a binary file, for later re-importing into a database using the corresponding Oracle import utility.
- Physical backups are the foundation of any sound backup and recovery strategy.
- Logical backups are a useful supplement to physical backups in many circumstances but are not sufficient protection against data loss without physical backups.

Camtesia

Lesson 202: Understanding Physical & Logical Data Loss

- Errors & Failures
- Understanding User Error
- Understanding Media Failure

Errors & Failures

- While there are several types of problem that can halt the normal operation of an Oracle database or affect database I/O operations, only two typically require DBA intervention and media recovery: media failure, and user errors.
- Other failures may require DBA intervention to restart the database (after an instance failure) or allocate more disk space (after statement failure due to, for instance, a full datafile) but these situations will not generally cause data loss or require recovery from backup.

Understanding User Error

- User errors occur when, either due to an error in application logic or a manual mis-step, data in your database is changed or deleted incorrectly.
- Data loss due to user error includes such missteps as dropping important tables or deleting or changing the contents of a table.
- While user training and careful management of privileges can prevent most user errors, your backup strategy determines how gracefully you recover the lost data when user error does cause data loss.

Understanding Media Failure

- A **media failure** is the failure of a read or write of a disk file required to run the database, due to a physical problem with the disk such as a head crash. Any database file can be vulnerable to a media failure.
- The appropriate recovery technique following a media failure depends on the files affected and the types of backup available.

Camtesia

Lesson 203: Full Logical Backup

- Logical Backup
- Logical Backup Entire Applications
- Outcome of Logical Backup
- Periodic Logical Backup

Logical Backup

- A logical backup copies data, but not physical files, from one location to another. A logical backup is used to move or archive a database, tables, or schemas and to verify database structures.

Logical Backup Entire Applications

A full logical backup enables you to copy these items across environments that use different components, such as operating systems:

Entire applications

Data repositories such as the Oracle Hyperion Shared Services Registry and Oracle Essbase cubes

Individual artifacts such as scripts, data forms, and rule files

Outcome of Logical Backup

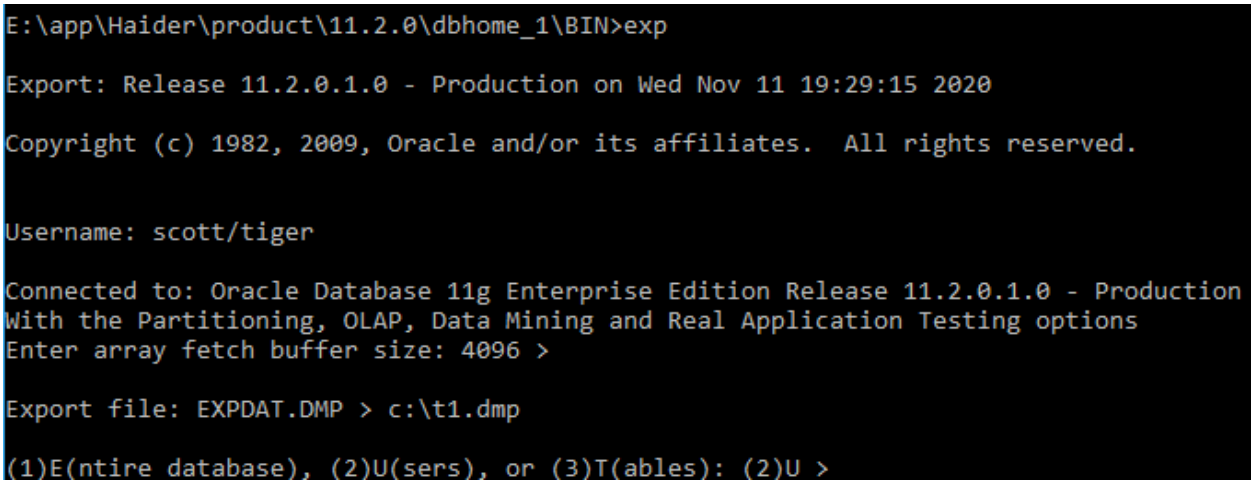
A logical export backup generates necessary Structured Query Language (SQL) statements to obtain all table data that is written to a binary file.

A logical export backup does not contain database instance-related information, such as the physical disk location, so you can restore the same data on another database machine.

Periodic Logical Backup

Periodic logical export backups (at least weekly) are recommended in case physical backups fail or the database machine becomes unavailable.

Logical Backup using EXPORT



Camtesia

Lesson 204: Logical Backup of a Schema/ User

- Logical Backup of Schema
- Export utility for Logical Backup
- Benefits of Export Utility for Logical Backup

Logical Backup of Schema

- A logical backup copies the data in the database and does not records the location of data.
- It can copy the data of schema, database and tables

Export utility for Logical Backup

- The export utility (export and export pump) offered by the Oracle can be used to take logical backup.
- The export utility copies the data and database definitions, and saves in binary OS file in Oracle internal format

Benefits of Export Utility for Logical Backup

- Data block corruption can be detected while exporting data and export procedure will fail.
- If use drops some table and it will recover from import command.
- What data and definition/ structure can be exported?
- Portable backup, can be imported in same or higher version of Oracle database.

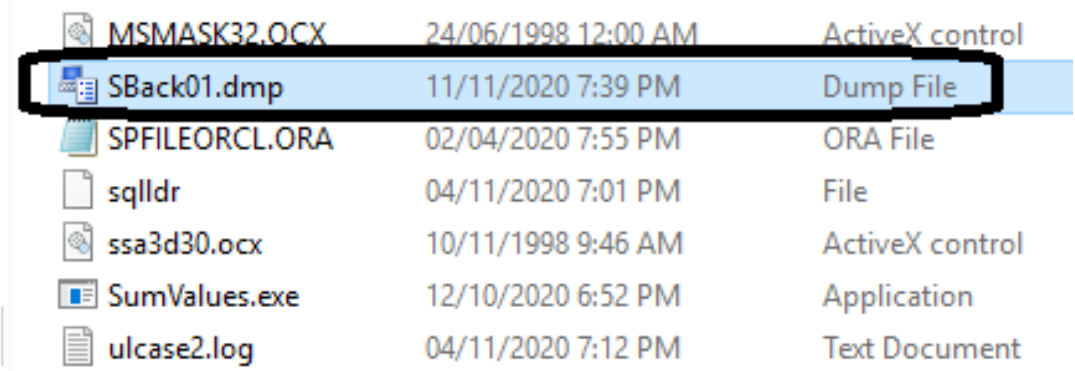
Camtesia

Lesson 205: Logical Backup of Schema Contents Tables

- Logical Backup of Schema Contents Tables
- Logical Backup Full/ Tables
- Export using Full Option
- Export with Individual Table

Logical Backup Full/ Tables

- Run export command with Full=y option.
- User will be scott/tiger
- Export data of some table



Export using Full Option

```
E:\app\Haider\product\11.2.0\dbhome_1\BIN>exp scott/tiger full=y file=c:\test\FBack01.dmp constraints=Y

Export: Release 11.2.0.1.0 - Production on Wed Nov 11 19:49:30 2020

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.


Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Export done in WE8MSWIN1252 character set and AL16UTF16 NCHAR character set

About to export the entire database ...
. exporting tablespace definitions
. exporting profiles
. exporting user definitions
. exporting roles
. exporting resource costs
. exporting rollback segment definitions
. exporting database links
. exporting sequence numbers
. exporting directory aliases
. exporting context namespaces
. exporting foreign function library names
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions
. exporting system procedural objects and actions
```

Export with Individual Table

```
Username: scott/tiger

Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Enter array fetch buffer size: 4096 >

Export file: EXPDAT.DMP > c:\test\emp.dmp

(1)E(ntire database), (2)U(sers), or (3)T(ables): (2)U > t

Export table data (yes/no): yes >

Compress extents (yes/no): yes >

Export done in WE8MSWIN1252 character set and AL16UTF16 NCHAR character set

About to export specified tables via Conventional Path ...
Table(T) or Partition(T:P) to be exported: (RETURN to quit) > EMP

. . exporting table                            EMP         17 rows exported
Table(T) or Partition(T:P) to be exported: (RETURN to quit) > DEPT

. . exporting table                           DEPT          4 rows exported
Table(T) or Partition(T:P) to be exported: (RETURN to quit) >

Export terminated successfully without warnings.

E:\app\Haider\product\11.2.0\dbhome_1\BIN>
```

Camtesia

Lesson 206: Cold Physical Backup

- Cold Physical Backup
- Cold Physical Backup Files
- Performing Cold Backup

Cold Physical Backup

- Offline or cold backups are performed when the database is completely shutdown.
- The disadvantage of an offline backup is that it cannot be done if the database needs to be run 24/7.
- Additionally, you can only recover the database up to the point when the last backup was made unless the database is running in ARCHIVELOG mode.

Cold Physical Backup Files

An offline backup consists of physically copying the following files:

- Data files
- Control files
- Init.ora and config.ora files

Warning: If you make a cold backup in ARCHIVELOG mode do not backup redo log files.

Performing Cold Backup

- Before performing a cold backup, you need to know the location of the files that need to be backed up.
- Because the database structure changes day to day as more files get added or moved between directories, it is always better to query the database to get the physical structure of database before making a cold backup.

Camtesia

Lesson 207: Cold Backup using RMAN

Cold Backup using RMAN

- Cold Backup using RMAN
- Start Backup using RMAN

Cold Backup using RMAN

- Keep Database in Archive Mode
- Go to location where RMAN can run

E:\app\Haider\product\11.2.0\dbhome_1\BIN>rman target /

RMAN>show all;

Start Backup using RMAN

RMAN> Backup as Backupset database;

Camtesia

Lesson 208:Hot Backup Concepts

- Hot Backup Concepts
- Startup Archive Mode

Hot Backup Concepts

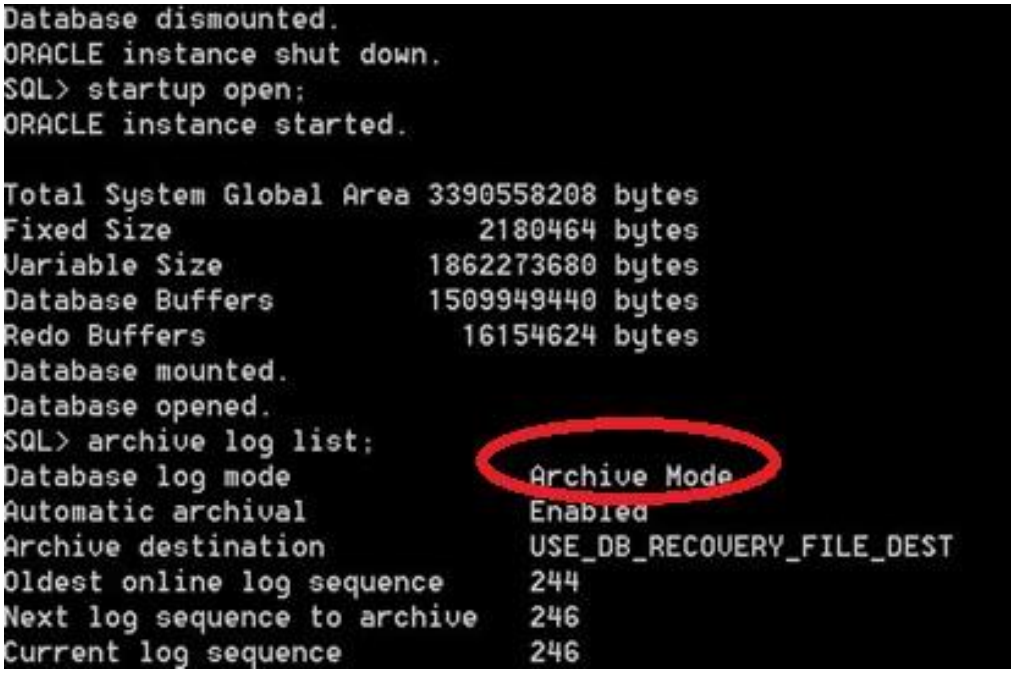
- A hot backup is taken when the database needs to run all the time.
- It is an online backup.
- All files of the database are copied and there may be changes to the database during the copy.
- An online backup or hot backup is also referred to as ARCHIVE LOG backup. An online backup can only be done when the database is running in ARCHIVELOG mode and the database is open.
- When the database is running in ARCHIVELOG mode, the archiver (ARCH) background process will make a copy of the online redo log file to archive backup location.

An online backup consists of backing up the following files. But, because the database is open while performing a backup, you have to follow the procedure to backup the following files:

- Data files of each tablespace
- Archived redo log files
- Control file
- Init.ora and config.ora files

Startup Archive Mode

Shutdown database and startup NoArchive to Archive database process



Camtesia

Lesson 209: Hot Backup Demo

Hot Backup Demo

- Hot Backup Demo
- Advise for Hot Backup

Hot Backup Demo

Following steps are required for Hot Backup:

- Put the tablespace in the Backup mode and copy the data files.
- Back up the control and Init.ora files.
- Stop archiving
- Back up the archive files
- Restart the archive process.

Advise for Hot Backup

An online backup of a database will keep the database open and functional for 24/7 operations. It is advised to schedule online backups when there is the least user activity on the database, because backing up the database is very I/O intensive and users can see slow response during the backup period. Additionally, if the user activity is very high, the archive destination might fill up very fast.

Camtesia

E:\app\Haider\oradata\orcl

SELECT TABLESPACE_NAME, FILE_NAME

FROM SYS.DBA_DATA_FILES

WHERE TABLESPACE_NAME = 'USERS';

Lesson 210: Restore from Backup using RMAN

Restore from Backup using RMAN

- Shutdown database
- Remove files from E:\app\Haider\oradata\orcl
- Startup nomount
- Login with RMAN target /
- Restore controlfile
- Alter database mount;
- Restore database
- Recover database;
- Resetlogs & Recovered Files

Shutdown Database

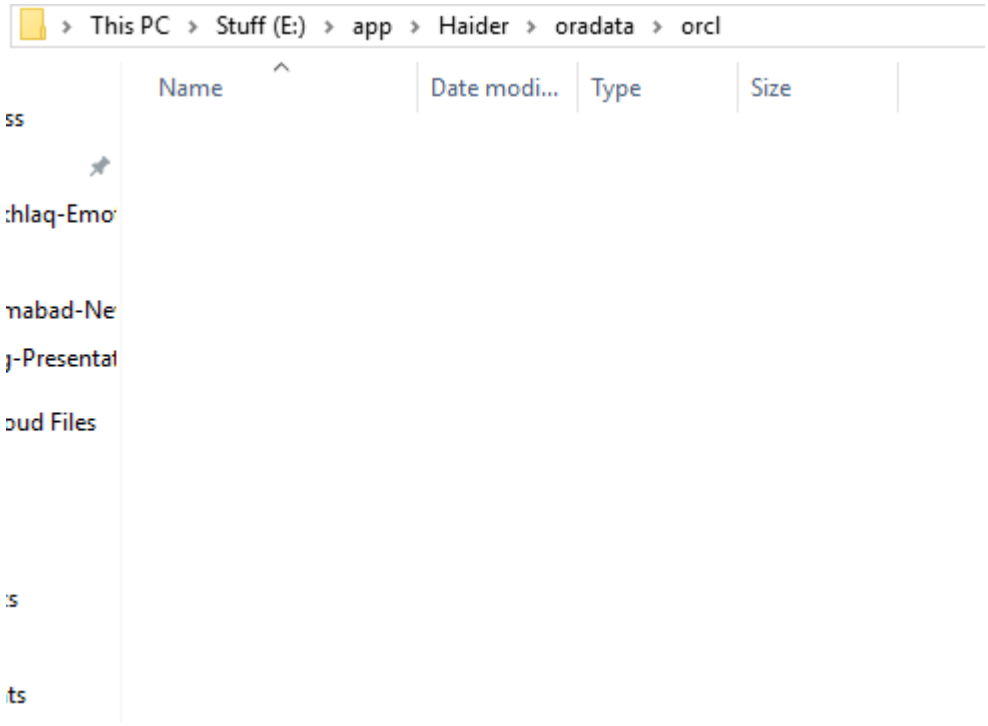
Now for recovery first off all shutdown your database using

“shutdown abort”

```
RMAN> shutdown abort

using target database control file instead of recovery catalog
Oracle instance shut down
```

Remove Files from E:\app\Haider\oradata\orcl

Now remove your database or do any change in database, here we physically removed all database files from database



Startup nomount

```
RMAN> startup nomount;

Oracle instance started

Total System Global Area     293601280 bytes

Fixed Size                     1290208 bytes
Variable Size                146800672 bytes
Database Buffers             138412032 bytes
Redo Buffers                   7098368 bytes

RMAN>
```

Login with RMAN target /

```
C:\oracle\product\10.2.0\db_1\bin>rman target /

Recovery Manager: Release 10.2.0.3.0 - Production on Thu Nov 27 06:58:55 2014

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

connected to target database (not started)

RMAN>
```

Restore Control Files

```
RMAN> restore controlfile from 'C:\rmanbackup\rman\FULL_0IPOK2TH_18_1';

Starting restore at 27-NOV-14
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: sid=156 devtype=DISK

channel ORA_DISK_1: restoring control file
channel ORA_DISK_1: restore complete, elapsed time: 00:00:10
output filename=C:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\CONTROL01.CTL
output filename=C:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\CONTROL02.CTL
output filename=C:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\CONTROL03.CTL
Finished restore at 27-NOV-14
```

Alter Database Mount

```
RMAN> alter database mount;


database mounted
released channel: ORA_DISK_1
```

Restore Database

```
RMAN> restore database;

Starting restore at 27-NOV-14
Starting implicit crosscheck backup at 27-NOV-14
allocated channel: ORA_DISK_1
channel ORA_DISK_1: sid=156 devtype=DISK
```

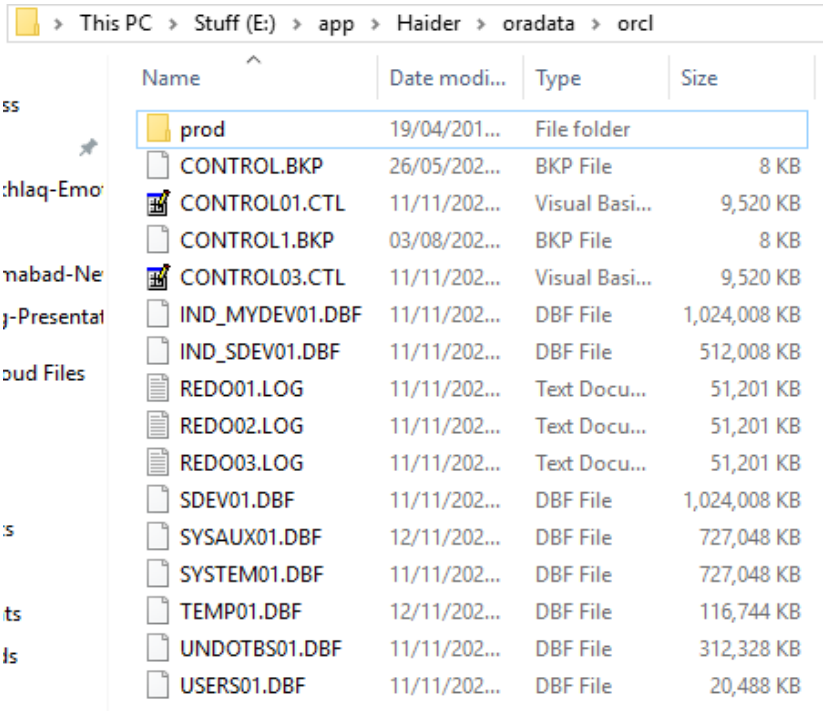
Recover Database

```
restoring datafile 00001 to C:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\SYSTEM01.DBF
restoring datafile 00002 to C:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\UNDOTBS01.DBF
restoring datafile 00003 to C:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\SYSAUX01.DBF
restoring datafile 00004 to C:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\USERS01.DBF
restoring datafile 00005 to C:\ORACLE\PRODUCT\10.2.0\DB_1\ORADATA\TBS_PERM_04.DA
T
restoring datafile 00006 to C:\ORACLE\PRODUCT\10.2.0\ORADATA\WORK01.DBF
restoring datafile 00007 to C:\ORACLE\PRODUCT\10.2.0\ORADATA\WORK02.DBF
restoring datafile 00008 to C:\ORACLE\PRODUCT\10.2.0\ORADATA\MCARE_PVISIT01.DBF
restoring datafile 00009 to C:\ORACLE\PRODUCT\10.2.0\ORADATA\MCARE_MEDLAB01.DBF
restoring datafile 00010 to C:\ORACLE\PRODUCT\10.2.0\ORADATA\MCARE_INDX01.DBF
restoring datafile 00011 to C:\ORACLE\PRODUCT\10.2.0\ORADATA\TESTING01.DBF
restoring datafile 00012 to C:\ORACLE\PRODUCT\10.2.0\ORADATA\MED_PHARMA01.DBF
restoring datafile 00013 to C:\ORACLE\PRODUCT\10.2.0\ORADATA\MED_INDX01.DBF
channel ORA_DISK_1: reading from backup piece C:\RMANBACKUP\RMAN\FULL_0HPOK2RP_1
7_1
channel ORA_DISK_1: restored backup piece 1
piece handle=C:\RMANBACKUP\RMAN\FULL_0HPOK2RP_17_1 tag=TAG20141126T221521
channel ORA_DISK_1: restore complete, elapsed time: 00:04:25
Finished restore at 27-NOV-14

RMAN>
```

```
RMAN> recover database;

Starting recover at 27-NOV-14
using channel ORA_DISK_1

starting media recovery
```

Resetlogs & Recovered Files

```
RMAN> alter database open resetlogs;

database opened

RMAN> _
```



Camtesia

Lesson211: Partial Restore Logical Backup

Partial Restore Logical Backup

- Partial Restore Logical Backup
- Steps Partial Restore Logical Backup
- Partial Restore Tablespace
- Restore Archive Logs

Partial Restore Logical Backup

- A partial database backup includes a subset of the database: individual tablespaces or data files.
- A tablespace backup is a backup of all the data files in a tablespace or in multiple tablespaces.
- Tablespace backups, whether consistent or inconsistent, are valid only if the database is operating in ARCHIVELOG mode because redo is required to make the restored tablespace consistent with the rest of the database.

Steps of Partial Restore Logical Backup

- Before beginning a backup of a tablespace, identify all of the datafiles in the tablespace with the DBA_DATA_FILES data dictionary view:

```
SELECT TABLESPACE_NAME, FILE_NAME

FROM SYS.DBA_DATA_FILES

WHERE TABLESPACE_NAME = 'USERS';
```

- Mark the beginning of the online tablespace backup:
- Back up the online datafiles of the online tablespace with operating system commands
- After backing up the datafiles of the online tablespace, run the SQL statement ALTER TABLESPACE with the END BACKUP option
- Archive the unarchived redo logs so that the redo required to recover the tablespace backup is archived

Partial Restore Tablespace

```
RMAN> restore tablespace users;


RMAN> recover tablespace users;
```

Restore Archive Logs

```
mkdir d:\archive
rman target /
RMAN> run {
2> set archivelog destination to 'd:\archive';
3> restore archivelog all;
4> }
```

Camtesia

Lesson212: Complete & Incomplete Recovery

- Complete Recovery
- Incomplete Recovery
- Database point-in-time Recovery
- Requirements for DPITR
- Parents, Ancestor & Sibling Incarnations
- Database Incarnation History with Multiple Resetlogs
- DPITR Incarnations
- DPITR Commands

Complete Recovery

- Complete recovery applies all redo changes contained in the archived and online logs to a backup. Typically, you perform complete media recovery after a media failure damages data files or the control file. You can perform complete recovery on a database, tablespace, or data file.

Incomplete Recovery

- Incomplete recovery, also called database point-in-time recovery, results in a noncurrent version of the database. In this case, you do not apply all of the redo generated after the restored backup. Typically, you perform point-in-time database recovery to undo a user error when Flashback Database is not possible.
- To perform incomplete recovery, you must restore all data files from backups created before the time to which you want to recover and then open the database with the RESETLOGS option when recovery completes. Resetting the logs creates a new stream of log sequence numbers starting with log sequence 1.

Database point-in-time Recovery

- **Database point-in-time recovery (DBPITR)** restores the database from backups prior to the target time for recovery, then uses incremental backups and redo to roll the database forward to the target time.
- DBPITR is sometimes called **incomplete recovery** because it does not use all of the available redo or completely recover all changes to your database.
- The target SCN can be specified using a date and time, in which case the operation is sometimes called **time-based recovery**.

Requirements for DPITR

The requirements for database point-in-time recovery are as follows:

- Your database must be running in ARCHIVELOG mode.
- You must have backups of all data files from before the target SCN for DBPITR and archived redo logs for the period between the SCN of the backups and the target SCN.

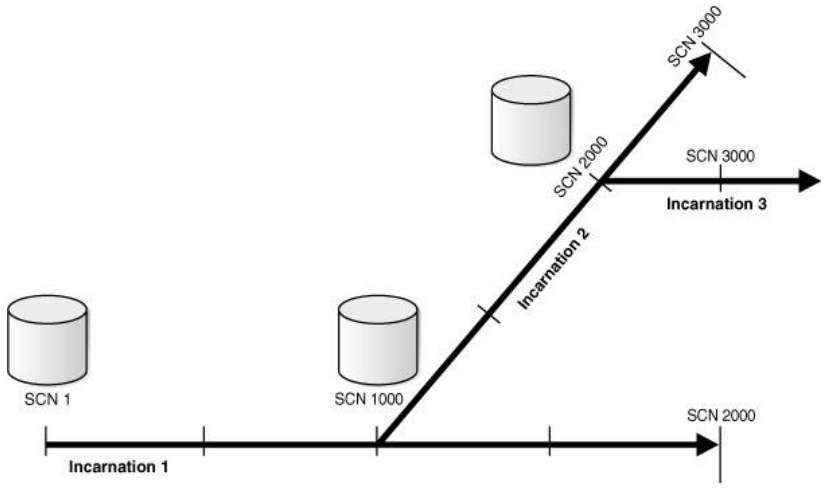
Parents, Ancestor & Sibling Incarnations

The incarnation from which the current incarnation branched following an OPEN RESETLOGS operation is called the parent incarnation of the current incarnation.

The parent incarnation and all of its parent incarnations are the ancestor incarnations of the current incarnation.

Two incarnations that share a common ancestor are sibling incarnations if neither one is an ancestor of the other.

Database Incarnation History with Multiple Resetlogs



Incarnation 1 of the database starts at SCN 1, and continues through SCN 1000 to SCN 2000. At SCN 2000 in incarnation 1, you perform a point-in-time recovery back to SCN 1000, and open the database with a RESETLOGS operation. This creates incarnation 2, which begins at SCN 1000 and continues to SCN 3000. At SCN 3000 in incarnation 2, you perform another point-in-time recovery and RESETLOGS operation. This creates incarnation 3, starting at SCN 2000.

DPITR Incarnations

The requirements for database point-in-time recovery are as follows:

- Your database must be running in ARCHIVELOG mode.
- You must have backups of all data files from before the target SCN for DBPITR and archived redo logs for the period between the SCN of the backups and the target SCN.

DPITR Commands

RMAN> RECOVER DATABASE TO SCN 1500;

RMAN> RESET DATABASE INCARNATION TO 1;

RMAN> RECOVER DATABASE TO SCN 1500;

Camtesia

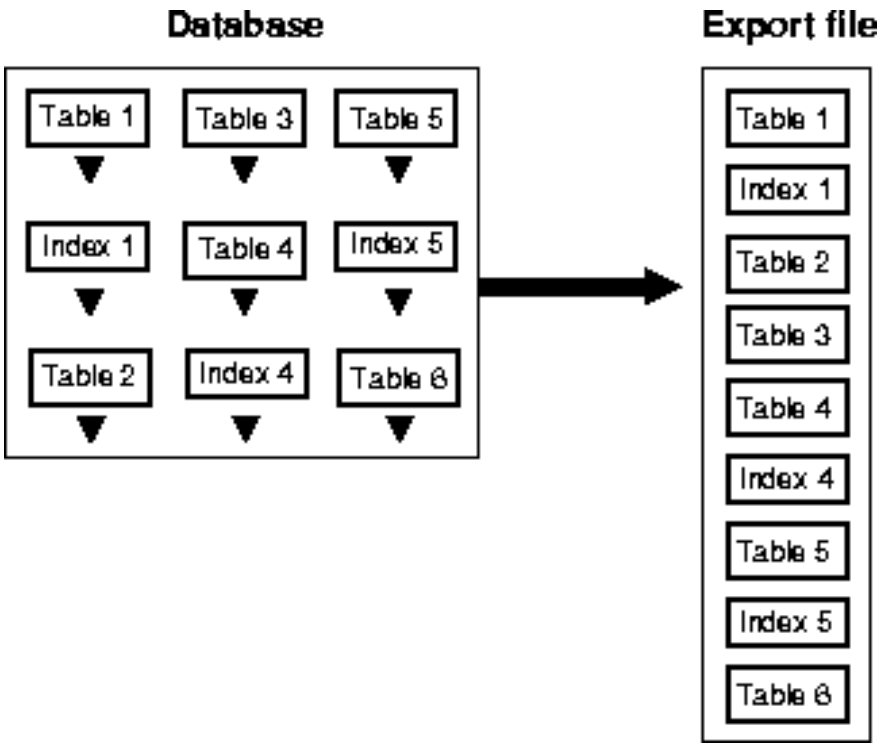
Lesson213: Understanding of Import/ Export

- Export Utility
- Export Dump File
- Export Commands
- Import Utility
- Import Ordering
- Export Commands

Export Utility

- The Export utility provides a simple way for you to transfer data objects between Oracle databases, even if they reside on platforms with different hardware and software configurations.
- When you run Export against an Oracle database, objects (such as tables) are extracted, followed by their related objects (such as indexes, comments, and grants), if any. The extracted data is written to an Export file.

Export Utility



Export Dump File

- An Export file is an Oracle binary-format dump file that is typically located on disk or tape. The dump files can be transferred using FTP or physically transported (in the case of tape) to a different site. The files can then be used with the Import utility to transfer data between databases that are on systems not connected through a network. The files can also be used as backups in addition to normal backup procedures.
- Export dump files can only be read by the Oracle Import utility. The version of the Import utility cannot be earlier than the version of the Export utility used to create the dump file.
- You can also display the contents of an export file without actually performing an import. To do this, use the Import SHOW parameter.

Export Commands

exp PARFILE=filename

exp username/password PARFILE=filename

FULL=y FILE=dba.imp GRANTS=y INDEXES=y CONSISTENT=y

exp username/password PARFILE=params.dat INDEXES=n

exp \'username/password AS SYSDBA\'

Import Utility

- The Import utility reads the object definitions and table data from an Export dump file. It inserts the data objects into an Oracle database.

Import Ordering

The export file contains objects in the following order

1. Type definitions
2. Table definitions

3. Table data
4. Table indexes
5. Integrity constraints, views, procedures, and triggers
6. Bitmap, functional, and domain indexes

Export Commands

imp username/password PARAMETER=value

or

imp username/password PARAMETER=(value1,value2,...,valuen)

imp PARFILE=filename

imp username/password PARFILE=filename

FULL=y FILE=dbay INDEXES=y CONSISTENT=y

Lesson:214: Controlling & Configuring Import/ Export

- Error Handling
- Failed Integrity Constraints
- Invalid Data
- Errors Importing Objects
- Object Already Exists
- Sequence
- Partition Level Import

Error Handling

- If a row is rejected due to an integrity constraint violation or invalid data, Import displays a warning message but continues processing the rest of the table. Some errors, such as "tablespace full," apply to all subsequent rows in the table. These errors cause Import to stop processing the current table and skip to the next table.
- A "tablespace full" error can suspend the import if the RESUMABLE=y parameter is specified.

Failed Integrity Constraints

A row error is generated if a row violates one of the integrity constraints in force on your system, including:

- NOT NULL constraints
- Uniqueness constraints
- Primary key (not null and unique) constraints
- Referential integrity constraints
- Check constraints

Invalid Data

- Row errors can also occur when the column definition for a table in a database is different from the column definition in the export file.
- The error is caused by data that is too long to fit into a new table's columns, by invalid datatypes, or by any other INSERT error.

Errors Importing Objects

- Errors can occur for many reasons when you import database objects.
- When these errors occur, import of the current database object is discontinued. Import then attempts to continue with the next database object in the export file.

Object Already Exists

- If a database object to be imported already exists in the database, an object creation error occurs. What happens next depends on the setting of the IGNORE parameter.

- If IGNORE=n (the default), the error is reported, and Import continues with the next database object. The current database object is not replaced.

Object Already Exists

- If IGNORE=y, object creation errors are not reported. The database object is not replaced. If the object is a table, rows are imported into it. Note that only object creation errors are ignored; all other errors (such as operating system, database, and SQL errors) are reported and processing may stop.

Sequence

- If sequence numbers need to be reset to the value in an export file as part of an import, you should drop sequences.
- If a sequence is not dropped before the import, it is not set to the value captured in the export file, because Import does not drop and re-create a sequence that already exists. If the sequence already exists, the export file's CREATE SEQUENCE statement fails and the sequence is not imported.

Partition Level Import

- You can import tables, partitions, and sub-partitions in the following ways:
- Table-level Import: imports all data from the specified tables in an Export file.
- Partition-level Import: imports only data from the specified source partitions or sub-partitions.
- You must set the parameter IGNORE=y when loading data into an existing table.

Lesson: 215:Walkthrough Import when Table Data Exist

- Import When Table Already Exists

Import when Table Already Exist

- When tables are manually created before data is imported, the CREATE TABLE statement in the export dump file will fail because the table already exists. To avoid this failure and continue loading data into the table, set the import parameter IGNORE=y. Otherwise, no data will be loaded into the table because of the table creation error or use table_exists_action=append using expdp and impdp

Camtesia

Lesson:216:;Walkthrough Import with Moving Objects

- Portal Import
- Moving Data to the Target System

Portal Import

- Create transport sets and extract the content of the transport sets to transport tables. Transport sets contain the portal objects that you want to export to your target portal environment. This information is displayed in a manifest. The manifest is a list of objects in a transport set, used to provide a granular level of control over the export. For more information on the export process and using the Transport Sets

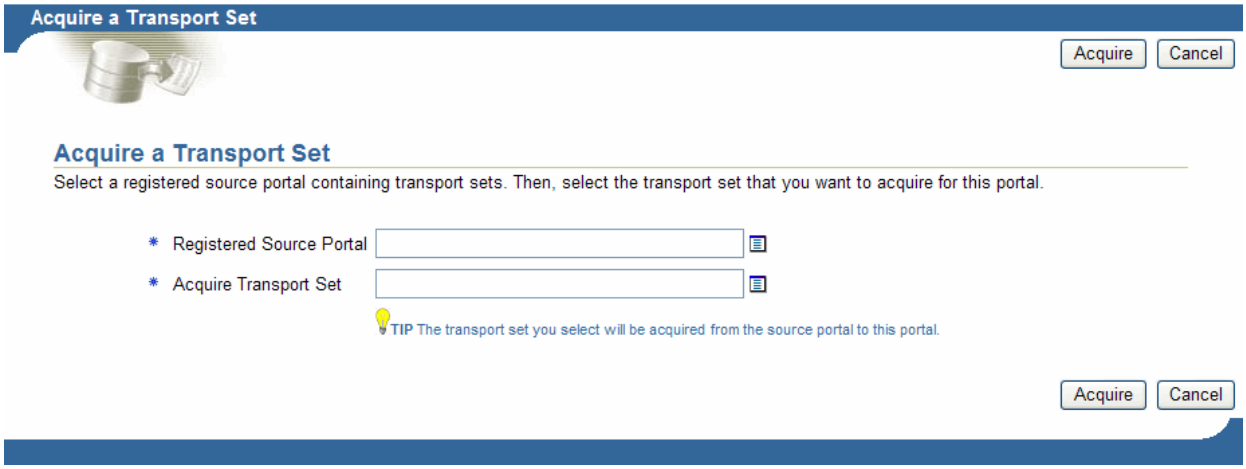
https://docs.oracle.com/cd/E16340_01/portal.1111/e10239/cg_imex.htm

Portal Import

- Move the transport sets from one system (source) to another (target) using the Transport Sets - Acquire Service portlet, or using the Oracle Portal Export and Import command-line scripts if there is an intervening firewall. Moving the transport set from the source to the target system using the Transport Sets
- Import the objects from the transport tables to the target portal repository using the Import Transport Set portlet.

Moving Data to the Target System

- To move your content using the Acquire Transport Sets portlet:
- In the Transport Set - Acquire Services portlet, click the Browse Source Portals on the Administer page and check that the source portal you're moving data from is registered on the target instance.
- If the source portal has not previously been registered, from the Acquire Transport Sets portlet, click Register A Source Portal and register. From the Transport Set - Acquire Services portlet, click Acquire Transport Sets to display the Acquire a Transport Set page.
- Specify the source portal and the transport set and click **Acquire** to start transferring the data.
- When the transfer is complete, click **Display Manifest** to display the transport set manifest. Click on the explicitly selected object to display the detailed manifest. Click **Tree View** to see the nested dependencies.
- Click **Close** to return to the manifest, and then click **Pre-check Now** to perform a pre-check of the transferred data. The system checks that all object dependencies have been resolved.
- Click **Display Manifest** when the pre-check is complete and check the status column for errors. Resolve any errors before continuing with importing the transport set.



Lesson:217: Logical Backup with Oracle Utilities

- Logical Backup
- Oracle Data Pump Architecture
- Modes of Data Pump
- Examples of Data Pump

Logical Backup

- Logical backups are backups in which the export utility (for example Exp) uses SQL to read database data and then export it into a binary file at the operating system level. You can then import the data back into a database using the import utility (Imp).

https://oracle-help.com/backup-and-recovery/logical-backup/#:~:text=Logical%20backups%20are%20backups%20in,the%20import%20utility%20(Imp).

- The Oracle Export Utility can create a file that contains all the objects in a particular schema. The file that is created can then be imported to the same schema to recover from a failure or to a different instance or schema for archiving purposes.

Oracle Data Pump Architecture

Data Pump consists of the following components:

expdp (Data Pump export utility)

impdp(Data Pump import utility)

DBMS_DATAPUMP PL/SQL package(Data Pump application interface [API])

DBMS_METADATA PL/SQL package (Data Pump Metadata API)

Modes of Data Pump

- Entire database
- Schema level
- Table level
- Tablespace level
- Transportable tablespace level

Examples of Data Pump

An example of Export:-

$ expdp mv_maint/foo directory=dp_dir dumpfile=full.dmp logfile=full.log full=y

An example of Import:-

$ impdp mv_maint/foo directory=dp_dir dumpfile=full.dmp logfile=fullimp.log full=y

Lesson218: Walkthrough Export with Moving Objects

- Pre-requisite Portal Export & Import
- Command-line Access
- Oracle Export & Import with Character Sets
- Role of Character Sets in Export & Import

Pre-requisite Portal Export & Import

Oracle Portal instance information:

Before beginning the export and import process, ensure you have the following information:

- System Requirements
- Additional Considerations

https://docs.oracle.com/cd/E16340_01/portal.1111/e10239/cg_imex.htm

Export and import functions only within the same release of Oracle Portal and the same patch release, for example, release 10.1.4 to release 10.1.4 or release 11.1.1 to release 11.1.1. You cannot export and import between two different releases, such as release 10.1.2 to release 10.1.4 or release 10.1.4 to release 11.1.1.

For successful migration of objects, the version of the portal repository should be the same in the target and the source. Any difference in the versions of the middle tiers does not impact migration.

When exporting or importing large data sets, check that there is sufficient space in the TEMP tablespace. This ensures that the export or import process does not fail due to insufficient memory.

For exporting large page groups from the command line, use the opeasst.csh script. See Section 11.4.1.1.3, "Exporting Large Page Groups from the Command Line" for more information.

For importing large page groups from the command line, use the import script with the -automatic_merge option. See Section 11.7.3, "Importing the Transport Set Tables to the Target System" for more information.

Pre-requisite Portal Export & Import

- Privileges for Exporting and Importing Content
  - Portal schema name.
  - Portal schema password.
  - Portal connect string information.
  - Portal user name.
  - Portal user password.
  - Company name (used only for hosted portal installations). In most cases, this should be left blank.

Oracle Export & Import with Character Sets

- The Oracle Database exp utility always exports user data, including Unicode data, in the character sets of the export server. The character sets are specified when the database is created.
- The Oracle Database imp utility automatically converts the data to the character sets of the import server.

Command-line Access

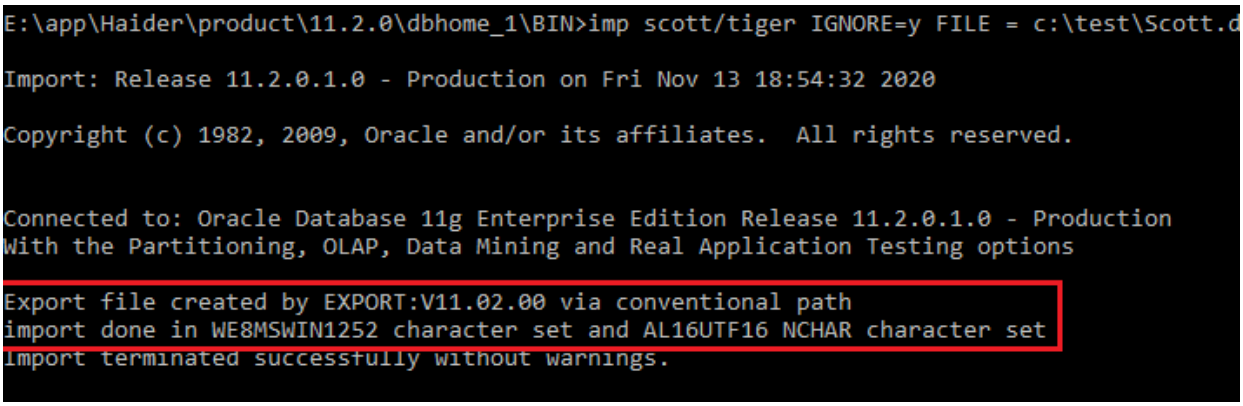
- If you are using the Oracle Portal Export and Import command-line scripts to move the transport sets from one system to another, you must have command-line access to run the shell or command utilities generated by the export import process. The command-line utilities, in turn, access the Oracle Database exp and imp utilities, and the Oracle Portal instance.

Oracle Export & Import with Character Sets

- Some 8-bit characters can be lost (that is, converted to 7-bit equivalents) when you import an 8-bit character set export file. This occurs if the client system has a native 7-bit character set or if the NLS_LANG operating system environment variable is set to a 7-bit character set. Most often, you notice that accented characters lose their accent marks.
- Both the Oracle Database exp and imp utilities alert you of any required character set conversion before exporting or importing the data.

https://docs.oracle.com/cd/E16340_01/portal.1111/e10239/cg_imex.htm

Role of Character Sets in Export & Import



Lesson: 219:General Overview of Database Manager

General Overview of Oracle Enterprise Manager (OEM)

- Starting OEM
- Oracle Database Control Login
- OEM Home Page
- OEM - Grant Management Privileges
- OEM - Users Credentials

Starting OEM

On cmd (DOS prompt), you may run

set oracle_sid=orcl

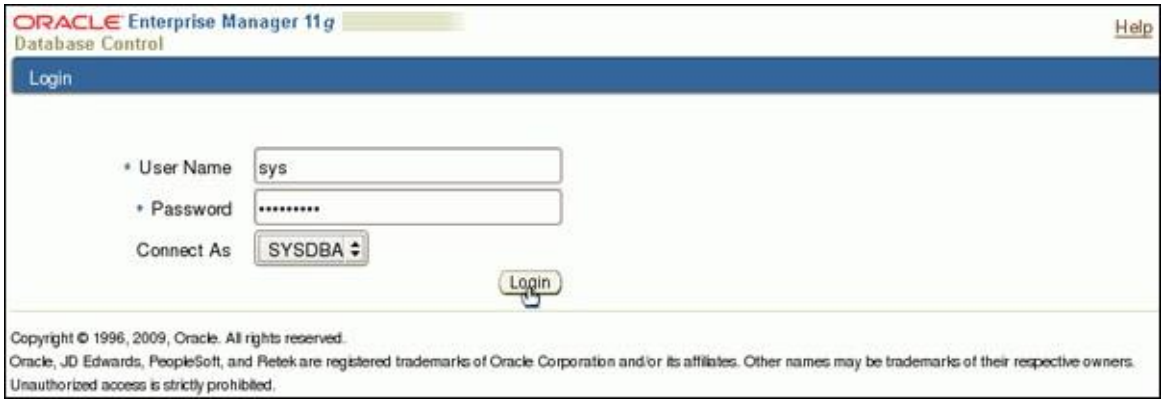
emctl stop dbconsole

emctl start dbconsole

In the browser, you can run URL

https://localhost:1158/em

Oracle Database Control Login



OEM Home Page

OEM – Grant Management Privileges

OEM – Users Credentials

OEM – Users Credentials

https://www.oracle.com/webfolder/technetwork/tutorials/obe/db/11g/r2/2day_dba/gettingstarted/gettingstarted.htm

Lesson:220: Transaction Processing

- What is a Transaction?
- Processing of Transactions
- Processing of Transactions in Oracle

What is a Transaction?

SQL is initiated at user level (client) with Start-End and it executes on Server (central Database) at a certain distance.

A transaction is a logical, atomic unit of work (block) that contains one or more SQL statements.

Either transaction is done (committed) or not done (rolled back). No partial execution of transaction is allowed.

Processing of Transactions

Generally reflects the activities in the real world system

Depositing some amount in an account, will be a transaction

Transferring amount from one account to another; if ok then DB is in consistent state otherwise inconsistent.

One transaction can be calculating GPA in our exam system

Processing of Transactions in Oracle

In Oracle Database, each user must see a consistent view of the data, including visible changes made by a user's own transactions and committed transactions of other users.

- Single-user VS multi-user systems
  A DBMS is **single-user** if at most one user can use the system at a time

  A DBMS is **multi-user** if many users can use the system concurrently

- Problem

How to make the simultaneous interactions of multiple users with the database safe, consistent, correct, and efficient?

Lesson 221:Defining Transaction

- What is a Transaction?
- Transaction with SQL
- Converting SQL into Transaction

What is a Transaction?

A transaction T is a logical unit of database processing that includes one or more database access operations

Embedded within an application program

Specified interactively (e.g., via SQL)

Transaction boundaries:

Begin/end transaction

Types of transactions

Read transaction

write transaction

Transaction with SQL

For example, following SQL update will change the salary of some Employee

Update EMP

Set sal = 850

Where empno = 7369;

Is it statement or a transaction?

Converting SQL into Transaction

```
begin

    Update EMP
    Set sal = 850
    Where empno = 7369;

end;
/
```

Camtesia

Lesson 222:Role of Transaction

Role or Importance of Transactions

- Transaction & SQL
- Importance of Transaction Processing System
- Transaction with Client-Server Environment

Transaction & SQL

Transaction is travelled on network, it may have chances of lost and it requires transaction logs for tracing

Can a simple SQL manages logs?

Can transaction in block manageable?

Issues of Multi-user environment.

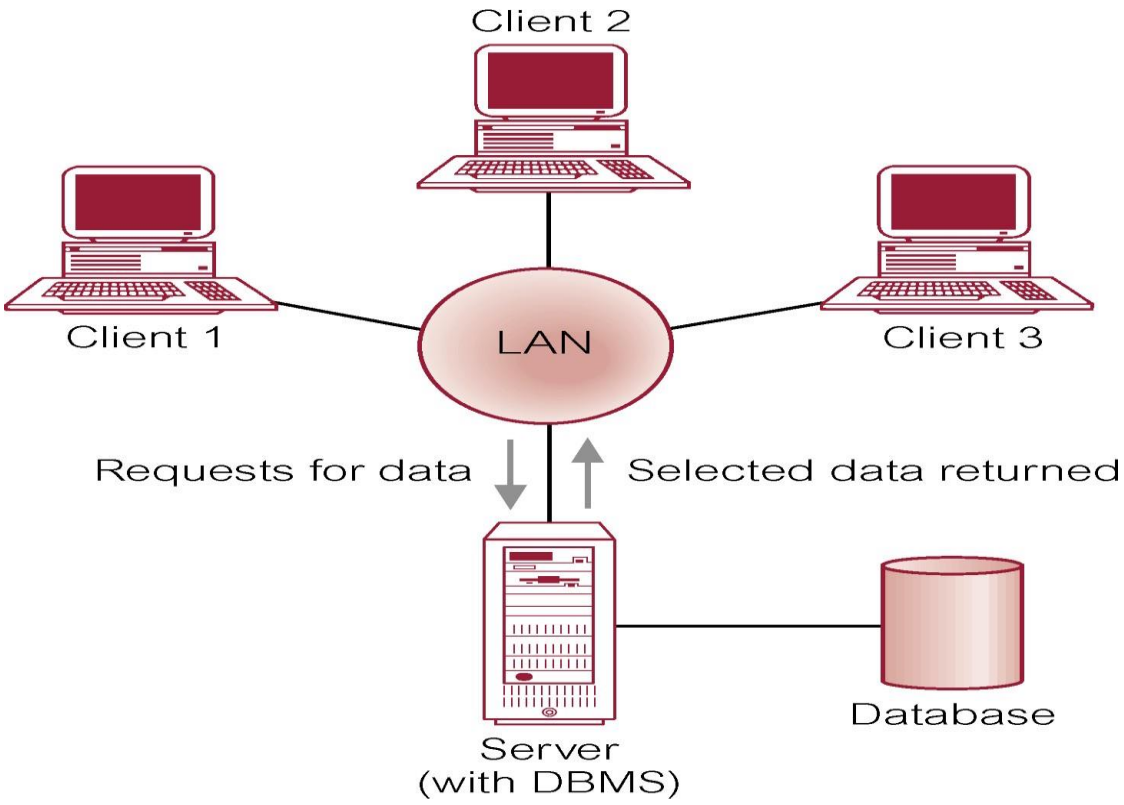
Issues of Consistent, Correct and Efficiency are important.

Importance of Transaction Processing System

Transaction processing (TP) systems underlie the online applications. a transaction rate of 15 simple Debit Credit transactions per second was “common”, 50 TPS was a “good” system, 100 TPS was “fast”, and a 400 TPS system was characterized as being a very “lean and mean” system.

Teradata capabilities, usage of parallel databases, configuration and tuning etc.

Transaction with Client-Server Environment



Transaction is travelled on network, it may have chances of lost and it requires transaction logs for tracing

Lesson 223: Example of Transactions

- Usage of ATM - An Example
- Funds Transfer Concept
- Writing a Transaction

Usage of ATM – An Example

Usage of ATM to draw/ transfer money, to collect currency with dispenser

Funds Transfer Concept

Transfer 400,000 from checking account to savings account

For a user it is one activity

To database:

Read balance of checking account: read(X)

Read balance of savings account: read (Y)

Subtract 400,000 from X

Add 400,000 to Y

Write new value of X back to disk

Write new value of Y back to disk

Writing a Transaction

T1

t_0
read_item(X);
read_item(Y);
X:=X - 400000;
Y:=Y + 400000;
write _item(X);
t_k write_item(Y);

Lesson 224: Example of Transactions

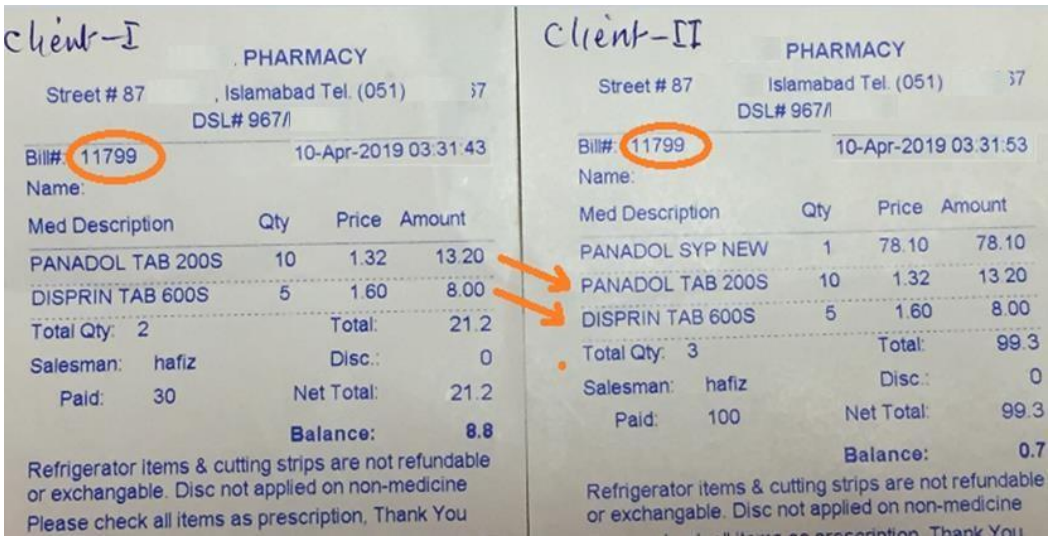
- Retail System - An Example
- Transactions & Their Issues

Retail Store – An Example

Invoice is issued a transaction when buying items

What happened multiple invoices are issued in Multi-user environment

Transactions and their Issues



Client-I

PHARMACY

Street # 87 , Islamabad Tel. (051) 37

DSL# 967/

Bill#: 11799 10-Apr-2019 03:31:43

Name:

Med Description	Qty	Price	Amount
PANADOL TAB 200S	10	1.32	13.20
DISPRIN TAB 600S	5	1.60	8.00

Total Qty: 2 Total: 21.2

Salesman: hafiz Disc.: 0

Paid: 30 Net Total: 21.2

Balance: 8.8

Refrigerator items & cutting strips are not refundable or exchangable. Disc not applied on non-medicine

Please check all items as prescription, Thank You

Client-II

PHARMACY

Street # 87 Islamabad Tel. (051) 37

DSL# 967/

Bill#: 11799 10-Apr-2019 03:31:53

Name:

Med Description	Qty	Price	Amount
PANADOL SYP NEW	1	78.10	78.10
PANADOL TAB 200S	10	1.32	13.20
DISPRIN TAB 600S	5	1.60	8.00

Total Qty: 3 Total: 99.3

Salesman: hafiz Disc.: 0

Paid: 100 Net Total: 99.3

Balance: 0.7

Refrigerator items & cutting strips are not refundable or exchangable. Disc not applied on non-medicine

Lesson 225: Transactions Read & Write

Example of Transactions

- Transaction Read Item
- Transaction Write Item
- Two Transactions

Transaction Read Item

A database is represented as a collection of named data items

Read-item (X)

- Find the address of the disk block that contains item X
- Copy the disk block into a buffer in main memory
- Copy the item X from the buffer to the program variable named X

Transactions Write Item

Write-item (X)

- Find the address of the disk block that contains item X.
- Copy that disk block into a buffer in main memory
- Copy item X from the program variable named X into its correct location in the buffer.
- Store the updated block from the buffer back to disk (either immediately or at some later point in time).

Two Transactions

	To	T1
t_0	Read (A) A:=A-50 Write (A) Read (B) B:=B+50 Write (B)	
		Read (A) Temp:=A*0.1 A:=A-temp Write (A) Read(B) B:=B+temp Write(B)
t_k		

Lesson 226: Concurrent Transactions Examples

Example of Transactions

- Concurrent Transaction
- Interleaved & Parallel Processing
- Concurrent Processing - An Example

Concurrent Transaction

In the transaction process, a system usually allows executing more than one transaction simultaneously. This process is called a concurrent execution.

The reasons for wanting to run transactions concurrently are slow input/output devices (usually disks) and the presence of multiple processors on which different transactions can run at the same time.

Informally, if there are two transactions in progress concurrently (that is, the Begin of one happens between the Begin and the Commit of the other), we want all the observable effects to be as though all the actions of one transaction happen either before or after all the actions of the other.

Interleaved & Parallel Processing

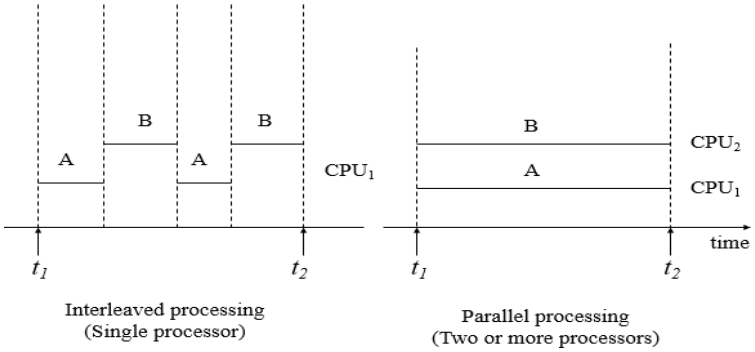
Concurrent transactions refer to two or more transactions that appear to users as they are being processed against a database at the same time

In reality, CPU can execute only one instruction at a time

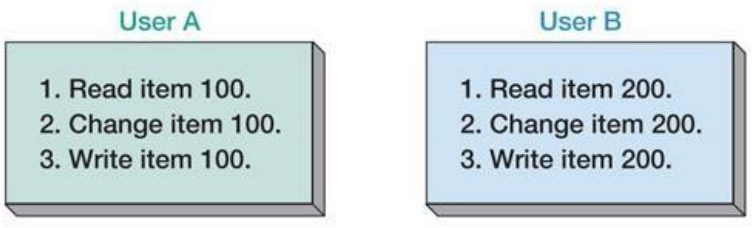
Transactions are interleaved meaning that the operating system quickly switches CPU services among tasks so that some portion of each of them is carried out in a given interval

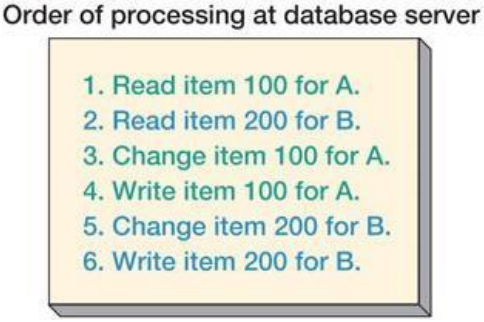
Concurrency problems: lost update and inconsistent reads

Interleaved & Parallel Processing



Concurrent Processing – An Example





Lesson227: Lost Update Example-1

Example of Transactions

- Concurrency Problems
- Lost Update Problems
- Lost Update Example-1

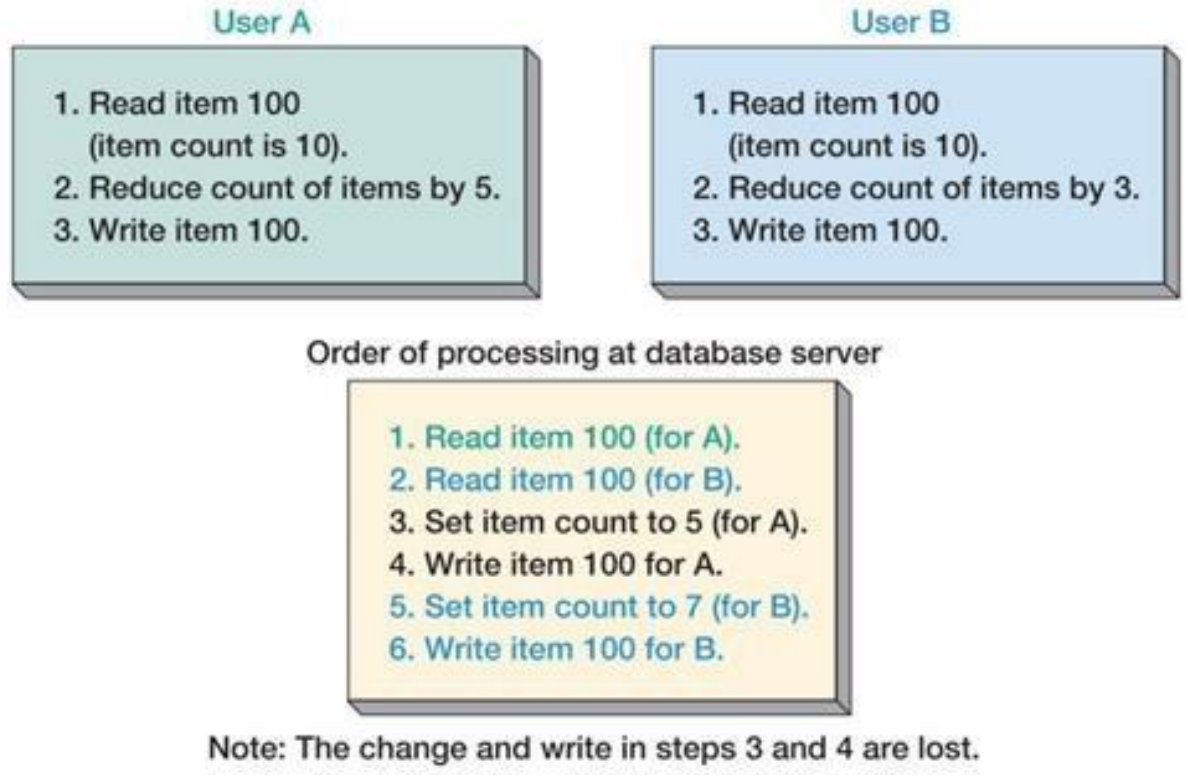
Concurrency Problems

Several problems can occur when concurrent transactions are run in an uncontrolled manner, such type of problems is known as concurrency problems.

Lost Update Problems

This type of problem occurs when two transactions in database access the same data item and have their operations in an interleaved manner that makes the value of some database item incorrect.

Lost Update Example-1



Lesson228: Lost Update Example-2

Example of Transactions

- Concurrency Problems
- Lost Update Problems
- Lost Update Example-2

Concurrency Problems

Several problems can occur when concurrent transactions are run in an uncontrolled manner, such type of problems is known as concurrency problems.

Lost Update Problems

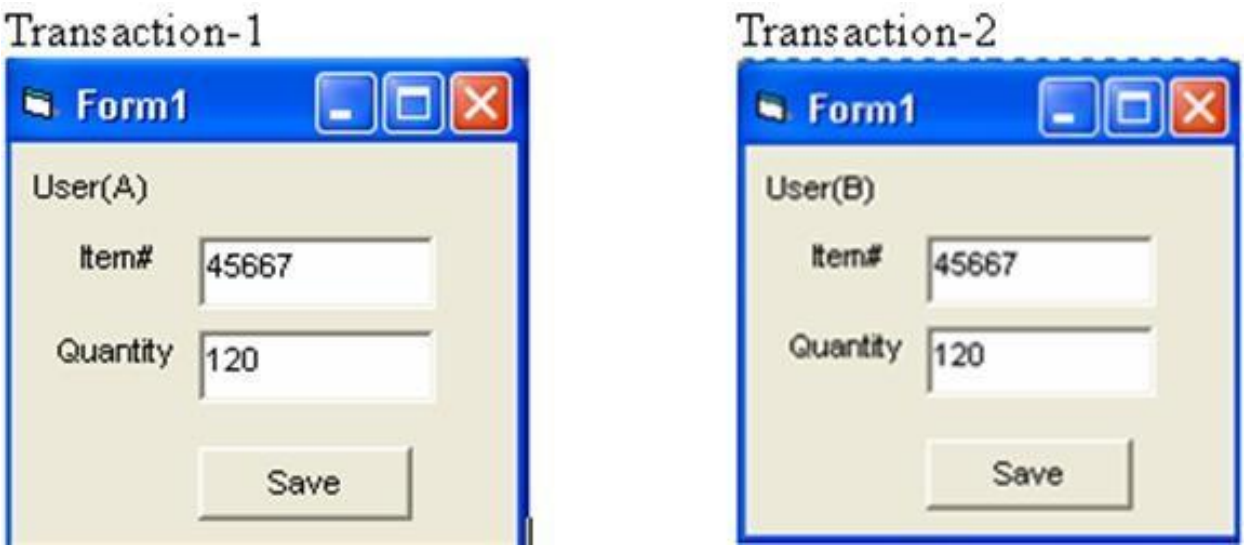
This type of problem occurs when two transactions in database access the same data item and have their operations in an interleaved manner that makes the value of some database item incorrect.

Lost Update Example-2

Time	T1	T2	balx
t1		BEGIN	100
t2	BEGIN	read(balx)	100
t3	read(balx)	balx=balx+100	100
t4	balx=balx-10	write(balx)	200
t5	write(balx)	commit	90
t6	commit	END	90
t7	END		

Lost Update Example-2

Two users are reading a same Item#45667, reading current quantity 120 simultaneously



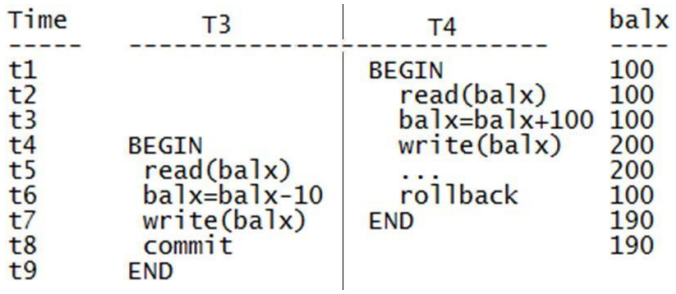
Lesson229: Dirty Read Example

Dirty Read Transactions

- Dirty Read Problem
- Dirty Read Example

This type of problem occurs when one transaction T1 updates a data item of the database, and then that transaction fails due to some reason, but its updates are accessed by some other transaction.

Dirty Read Example



Time	T3	T4	balx
t1		BEGIN	100
t2		read(balx)	100
t3		balx=balx+100	100
t4	BEGIN	write(balx)	200
t5	read(balx)	...	200
t6	balx=balx-10	rollback	100
t7	write(balx)	END	190
t8	commit		190
t9	END		

Dirty Read Example

Time	Transaction T1	Transaction T2
t1	Read(A)	
t2	A=A+20	
t3	Write(A)	
t4		Read(A)
t5		A=A+30
t6		Write(A)
t7	Write(B)	

At t1 time, T1 transaction reads the value of A i.e., 100.

At t2 time, T1 transaction adds the value of A by 20.

At t3 time, T1transaction writes the value of A (120) in the database.

At t4 time, T2 transactions read the value of A data item i.e., 120.

At t5 time, T2 transaction adds the value of A data item by 30.

At t6 time, T2transaction writes the value of A (150) in the database.

At t7 time, a T1 transaction fails due to power failure then it is rollback according to atomicity property of transaction (either all or none).

So, transaction T2 at t4 time contains a value which has not been committed in the database. The value read by the transaction T2 is known as a dirty read.

Lesson230: Incorrect Summary Problem

Incorrect Summary Problem

- Incorrect Summary Problem
- Incorrect Summary Problem - An Example

It is also known as an inconsistent retrieval problem. If a transaction T_1 reads a value of data item twice and the data item is changed by another transaction T_2 in between the two read operation. Hence T_1 access two different values for its two read operation of the same data item.

Incorrect Summary Problem – An Example

Time	Transaction T1	Transaction T2
t1	Read(A)	
t2		Read(A)
t3		A=A+30
t4		Write(A)
t5	Read(A)	

At t1 time, T1 transaction reads the value of A i.e., 100.

At t2 time, T2transaction reads the value of A i.e., 100.

At t3 time, T2 transaction adds the value of A data item by 30.

At t4 time, T2 transaction writes the value of A (130) in the database.

Transaction T2 updates the value of A. Thus, when another read statement is performed by transaction T1, it accesses the new value of A, which was updated by T2. Such type of conflict is known as R-W

Incorrect Summary Problem – An Example

```
Time    T3              T4              balx    baly    balz    sum
-----   ------------    ------------    ----    ----    ----    ----
t1                      BEGIN           100     50      25
t2      BEGIN             sum=0
t3        read(balx)      read(balx)
t4        balx=balx-10    sum=sum+balx
t5        write(balx)     read(baly)
t6        read(balz)      sum=sum+baly
t7        balz=balz+10
t8        write(balz)
t9        commit          read(balz)
t10     END               sum=sum+balz
t11                       commit
t12                     END
```

Sum under T4 is different from sum of balx, baly, balz

Lesson231: Data Inconsistency in Two Invoices

Data Inconsistency with Multiple Users

- Single User in a Retail Store
- Retail Store - An Example
- Transactions and their Issues

Single User in a Retail Store

Single user issue an Invoice one at a time when only one POS terminal is active

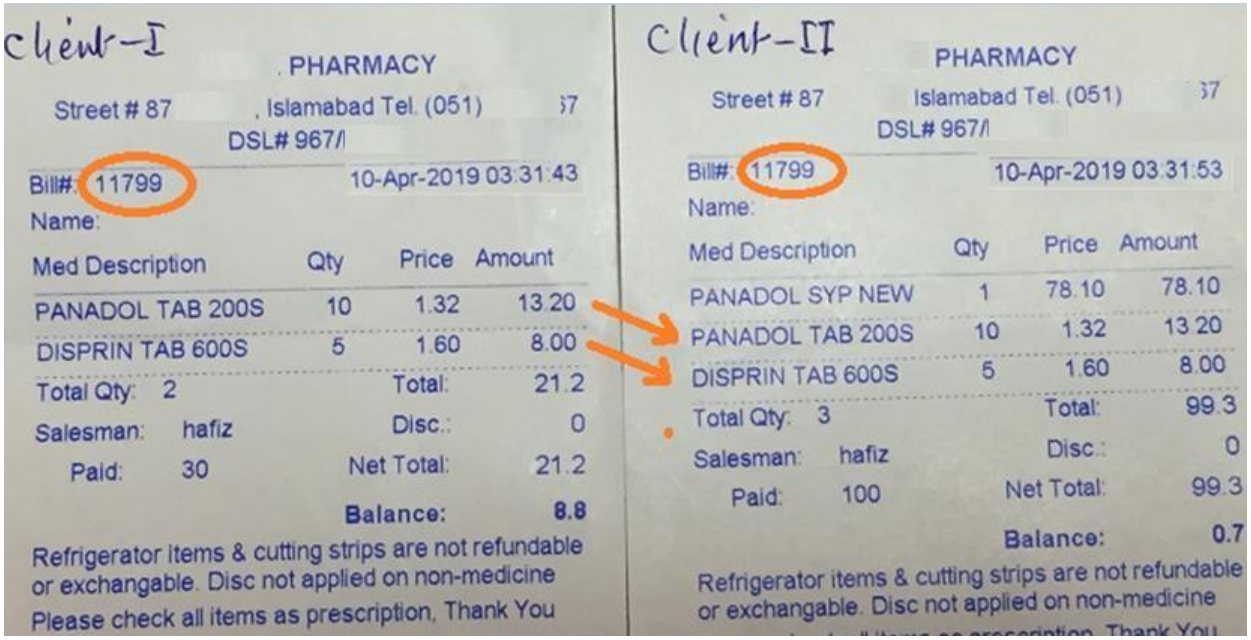
When two or more users are issuing invoices, there is a chance of data inconsistency

Retail Store – An Example

For example, two invoices are issued at the same time from two POS terminals

What happened multiple invoices are issued in Multi-user environment

Transactions and their Issues



client-I

PHARMACY

Street # 87 , Islamabad Tel. (051) 57

DSL# 967/

Bill#: 11799 10-Apr-2019 03:31:43

Name:

Med Description	Qty	Price	Amount
PANADOL TAB 200S	10	1.32	13.20
DISPRIN TAB 600S	5	1.60	8.00

Total Qty: 2 Total: 21.2

Salesman: hafiz Disc.: 0

Paid: 30 Net Total: 21.2

Balance: 8.8

Refrigerator items & cutting strips are not refundable or exchangable. Disc not applied on non-medicine

Please check all items as prescription, Thank You

client-II

PHARMACY

Street # 87 Islamabad Tel. (051) 57

DSL# 967/

Bill#: 11799 10-Apr-2019 03:31:53

Name:

Med Description	Qty	Price	Amount
PANADOL SYP NEW	1	78.10	78.10
PANADOL TAB 200S	10	1.32	13.20
DISPRIN TAB 600S	5	1.60	8.00

Total Qty: 3 Total: 99.3

Salesman: hafiz Disc.: 0

Paid: 100 Net Total: 99.3

Balance: 0.7

Refrigerator items & cutting strips are not refundable or exchangable. Disc not applied on non-medicine

Lesson 232: System Failure Problems

- What Can Go Wrong?
- System Failure Problems

What Can Go Wrong?

- System may crash before data is written back to disk
    - Problem of *atomicity*
- Some transaction is modifying shared data while another transaction is ongoing (or vice versa)
    - Problem of *serialization* and *isolation*
- System may not be able to obtain one or more of the data items
- System may not be able to write one or more of the data items (Problems of *atomicity)*
- DBMS has a *Concurrency Control* subsytem to assure database remains in consistent state despite concurrent execution of transactions

System Failure Problems

System failures may occur

Types of failures:

System crash

Transaction or system error

Local errors

Concurrency control enforcement

Disk failure

Physical failures

DBMS has a *Recovery* Subsystem to protect database against system failures

Lesson:233 : System Log

- Introduction
- System Log
- How the Log File is used?
- Transaction Log Example

Introduction

- **Log or Journal** : The log keeps track of all transaction operations that affect the values of database items. This information may be needed to permit recovery from transaction failures. The log is kept on disk, so it is not affected by any type of failure except for disk or catastrophic failure. In addition, the log is periodically backed up to archival storage (tape) to guard against such catastrophic failures.

Introduction

- T refers to a unique **transaction-id** that is generated automatically by the system and is used to identify each transaction

System Log

- Transaction –id
- System log
    - Multiple record-type file
    - Log is kept on disk
    - Periodically backed up
- Log records
    - [start_transaction, T]
    - [write_item, T,X,old_value,new_value]:
    - [read_item, T,X]
    - [commit,T]
    - [abort,T]
    - [checkpoint]
- Commit point of a transaction

How the Log File is used?

- All permanent changes to data are recorded
    - Possible to undo changes to data
- After crash, search log backwards until find last checkpoint
    - Know that beyond this point, effects of transaction are permanently recorded
- Need to either *redo* or *undo* everything that happened since last checkpoint
    - Undo: When transaction only partially completed (before crash)
    - Redo: Transaction completed but we are unsure whether data was written to disk

Transaction Log Example

TransID	Table	RowID	column	Before	After
100	**start**				
100	EMPLOYEE	1005472	Salary	8000	8250
100	PROJECT	1423PRJ	PrjRate	253	256
100	**end committed**				

Lesson234: Schedules & Recoverability

- Transaction and Schedule
- Schedule
- Example of Schedule

Transaction and Schedule

We have discussed that

Multiple transactions can be executed concurrently by interleaving their operations

Schedule

Ordering of execution of operations from various transactions T_1, T_2, … , T_n is called a schedule *S*

Schedule

Schedule S of n transactions T_1, T_2, … , T_n is an ordering of the operations of the transactions subject to the constraint that, for each transaction T_i that participates in S, the operations of T_i in S must appear in the same order in which they occur in T_i.

Example of Schedule

Transaction T_1: $r_1(X)$; $w_1(X)$; $r_1(Y)$; $w_1(Y)$; c_1

Transaction T_2: $r_2(X)$; $w_2(X)$; c_2

A schedule, S:

$r_1(X)$; $r_2(X)$; $w_1(X)$; $r_1(Y)$; $w_2(X)$; $w_1(Y)$; c_1; c_2

Lesson235: Conflicts

Schedule with Conflicts

- Conflicts
- Conditions for Conflicts
- Example of Conflicts

Conflicts

Conflict operations can creates inconsistency

Two operations *conflict* if they satisfy ALL three conditions:

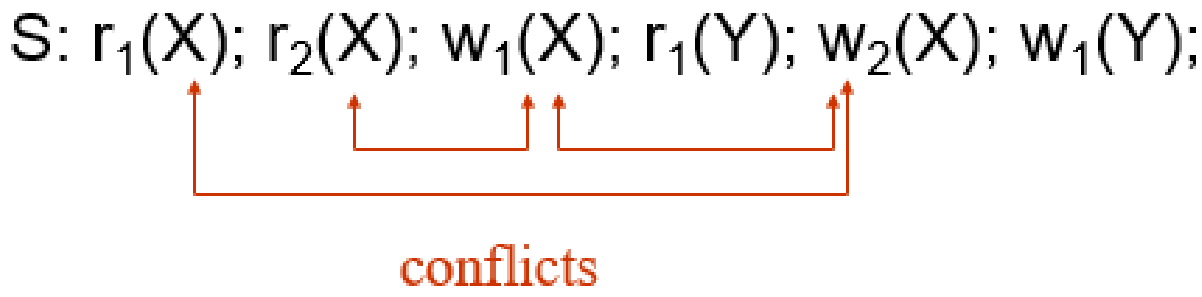
1. they belong to different transactions **AND**
2. they access the same item **AND**
3. at least one is a **write_item()**operation

Conditions for Conflicts

Two operations *conflict* if they satisfy ALL three conditions:

1. they belong to different transactions **AND**
2. they access the same item **AND**
3. at least one is a **write_item()**operation

Example of Conflicts



Lesson236: Serializability

Schedule with Serializability

- Serializability
- Serial Schedule
- Characteristics of Serial Schedule

Serializability

- Serial Schedule
- Non-serial schedule
- Serializable schedule
    - Conflict-serializable schedule
    - View-serializable schedule

Serial and Non-serial schedule
A schedule *S* is **serial** if, for every transaction *T* participating in the schedule, all the operations of *T* are executed consecutively in the schedule; otherwise, the schedule is called **non-serial**

Serializable schedule
A schedule *S* of *n* transactions is **serializable** if it is *equivalent to some serial schedule* of the same *n* transactions

Serial Schedule

- Serial schedule is guaranteed correct, should we adopt it?
- No, that will be inefficient, under the utilization of resources
- Dislike by the users

Characteristics of Serial Schedule

Result equivalent: Two schedules are called result equivalent if they produce the same final state of the database.

Conflict equivalent: Two schedules are said to be conflict equivalent if the order of any two conflicting operations is the same in both schedules.

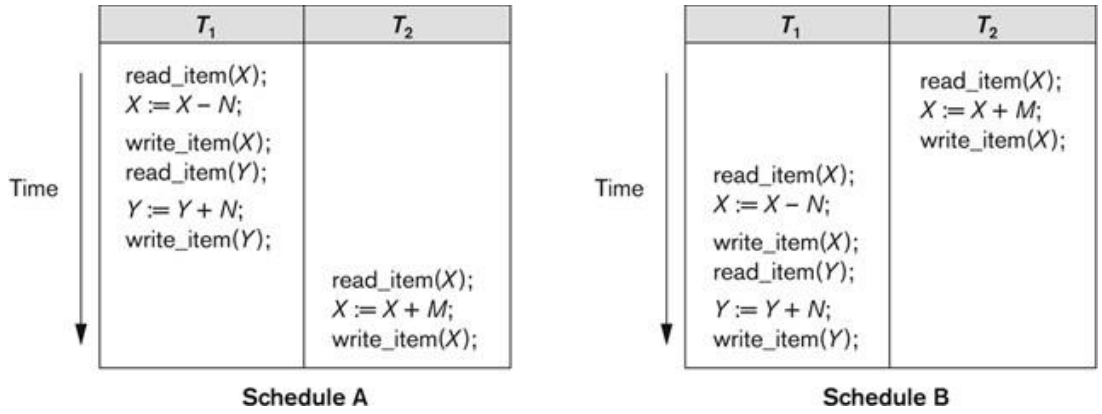
Conflict serializable: A schedule S is said to be conflict serializable if it is conflict equivalent to some serial schedule S'.

Lesson237: Serial Schedules Example-1

Schedule with Serial Schedules Example-1

- Example-1

Example-1



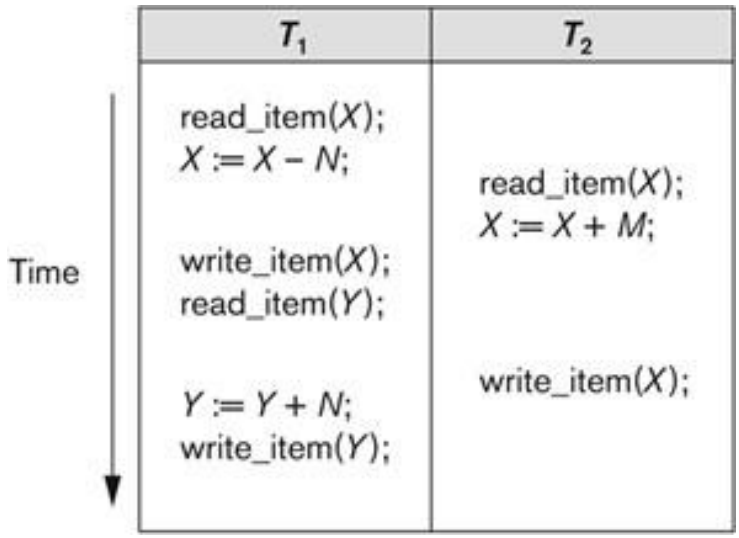
Assume X=90, Y=90, N=3, M=2

Lesson238: Serial Schedules Example-2

Schedule with Serial Schedules Example-2

- Example-2

Example-2 (Schedule S')



Assume X=90, Y=90, N=3, M=2

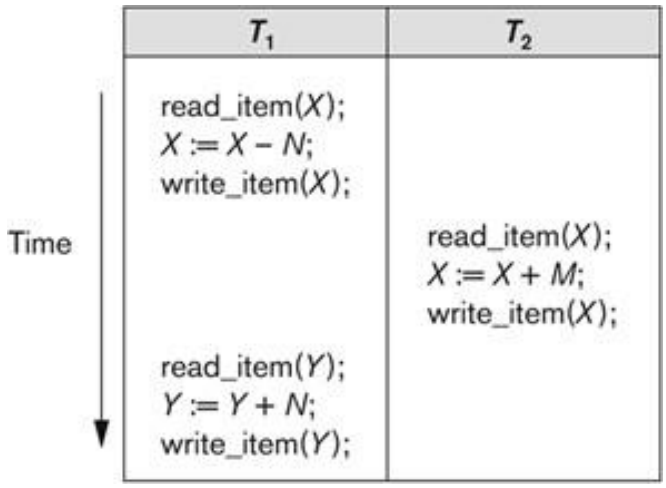
Not Correct Values

Lesson239: Serial Schedules Example-3

Schedule with Serial Schedules Example-3

- Example-3

Example-3 (Schedule S'')



Assume X=90, Y=90, N=3, M=2

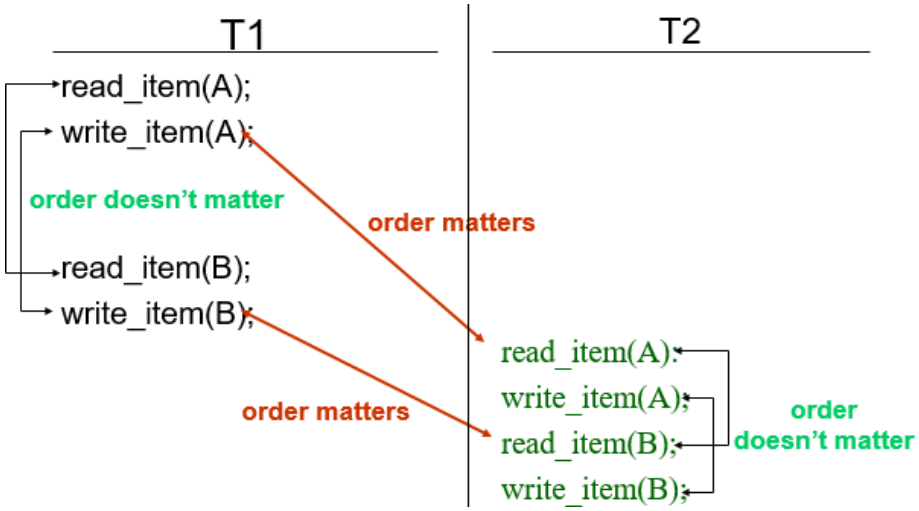
Produces same result as serial schedule

Lesson240: Conflict Serializability Example-1

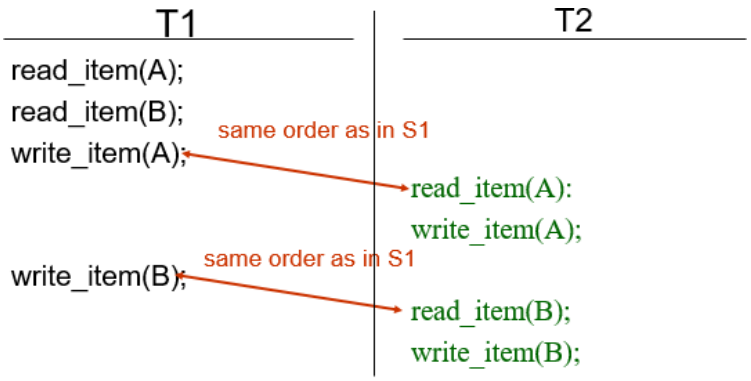
Schedule with Conflict Serializability Example-1

- Example-1 (S1)
- Example-1 (S1')
- Example-1 (Schedules A, B)

Example-1 (Schedule S)



Example-1 (Schedule S1')



S1 and S1' are conflict equivalent
(S1' produces the same result as S1)

Example-1 (Schedules A, B)

Schedule A

T1	T2
read_item(X);	
X:=X-N;	
write_item(X);	
read_item(Y);	
Y:=Y+N;	
write_item(Y);	
	read_item(X);
	X:=X+M;
	write_item(X);

Schedule B

T1	T2
read_item(X);	
X:=X-N;	
write_item(X);	
	read_item(X);
	X:=X+M;
	write_item(X);
read_item(Y);	
Y:=Y+N;	
write_item(Y);	

B is conflict equivalent to A ⇒ B is serializable

Lesson241: Conflict Serializability Example-2 (Graph Test)

Conflict Serializability Example-2 (Graph Test)

- Test for Serializability
- Example-2 (Graph Test)

Test for Serializability

Construct a directed graph, *precedence graph*, G = (V, E)

V: set of all transactions participating in schedule

E: set of edges $T_i \rightarrow T_j$ for which one of the following holds:

T_i executes a write_item(X) before T_j executes read_item(X)

T_i executes a read_item(X) before T_j executes write_item(X)

T_i executes a write_item(X) before T_j executes write_item(X)

Test for Serializability

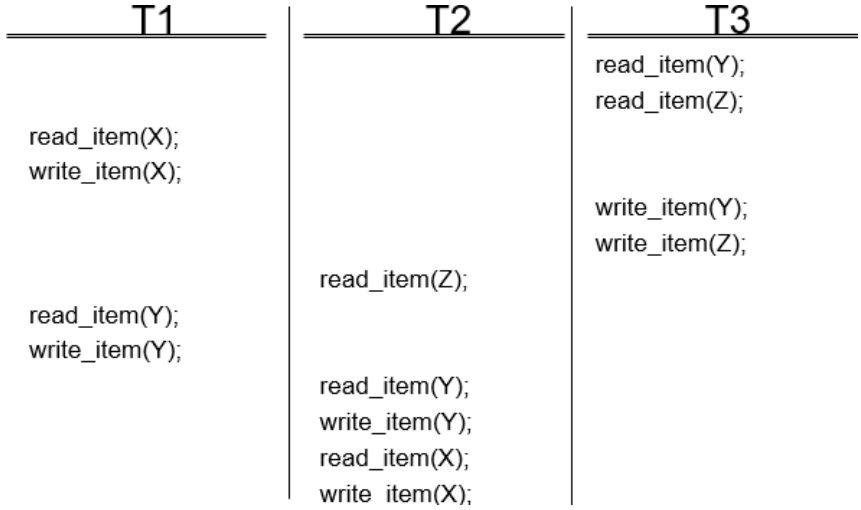
An edge $T_i \rightarrow T_j$ means that in any serial schedule equivalent to S, T_i must come before T_j

If G has a cycle, then S is not conflict serializable

If not, use topological sort to obtain serializable schedule (linear order consistent with precedence order of graph)

To check consistency if graph is acyclic otherwise schedule has deadlock means inconsistent

Example-2 (Graph Test)



Lesson242: Apply Locking

- Database Locks
- Locks in Oracle SQL
- Locks Update

Database Locks

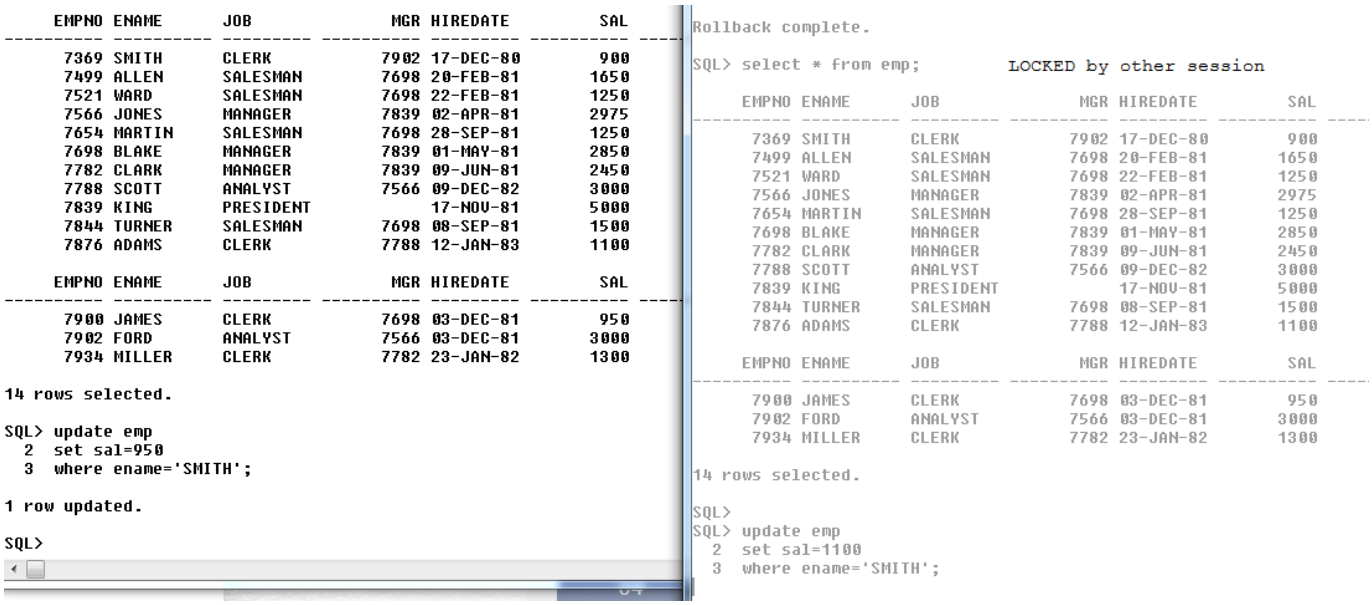
- When two sessions or users of database try to update or delete the same data in a table, then there will be a concurrent update problem. In order to avoid this problem, database locks the data for the first user and allows him to update/delete the data.
- Once he is done with his update/delete, he COMMITs or ROLLBACK the transaction, which will release the lock on the data. When lock on the data is released, another user can lock it for his changes.

Database Locks

- An Object is locked before it performs any operation
- Two types of operations, two types of locks

  Shared & Exclusive Locks

Locks in Oracle SQL

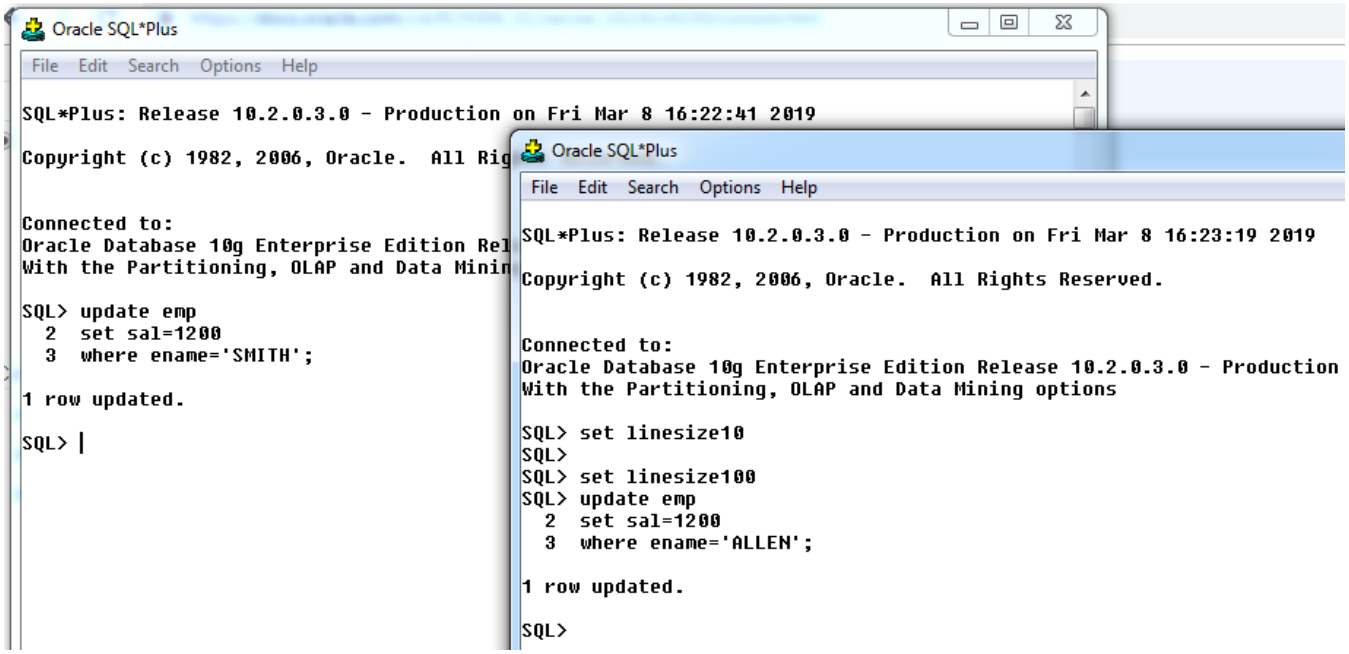


```
     EMPNO ENAME      JOB              MGR HIREDATE         SAL
---------- ---------- --------- ---------- --------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        900
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1650
      7521 WARD       SALESMAN        7698 22-FEB-81       1250
      7566 JONES      MANAGER         7839 02-APR-81       2975
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850
      7782 CLARK      MANAGER         7839 09-JUN-81       2450
      7788 SCOTT      ANALYST         7566 09-DEC-82       3000
      7839 KING       PRESIDENT            17-NOV-81       5000
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500
      7876 ADAMS      CLERK           7788 12-JAN-83       1100

     EMPNO ENAME      JOB              MGR HIREDATE         SAL
---------- ---------- --------- ---------- --------- ----------
      7900 JAMES      CLERK           7698 03-DEC-81        950
      7902 FORD       ANALYST         7566 03-DEC-81       3000
      7934 MILLER     CLERK           7782 23-JAN-82       1300

14 rows selected.

SQL> update emp
  2  set sal=950
  3  where ename='SMITH';

1 row updated.

SQL>
```

```
Rollback complete.

SQL> select * from emp;        LOCKED by other session

     EMPNO ENAME      JOB              MGR HIREDATE         SAL
---------- ---------- --------- ---------- --------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        900
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1650
      7521 WARD       SALESMAN        7698 22-FEB-81       1250
      7566 JONES      MANAGER         7839 02-APR-81       2975
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850
      7782 CLARK      MANAGER         7839 09-JUN-81       2450
      7788 SCOTT      ANALYST         7566 09-DEC-82       3000
      7839 KING       PRESIDENT            17-NOV-81       5000
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500
      7876 ADAMS      CLERK           7788 12-JAN-83       1100

     EMPNO ENAME      JOB              MGR HIREDATE         SAL
---------- ---------- --------- ---------- --------- ----------
      7900 JAMES      CLERK           7698 03-DEC-81        950
      7902 FORD       ANALYST         7566 03-DEC-81       3000
      7934 MILLER     CLERK           7782 23-JAN-82       1300

14 rows selected.

SQL>
SQL> update emp
  2  set sal=1100
  3  where ename='SMITH';
```

Record level lock/ Row Lock – same record

Locks Update



```
Oracle SQL*Plus
File  Edit  Search  Options  Help

SQL*Plus: Release 10.2.0.3.0 - Production on Fri Mar 8 16:22:41 2019

Copyright (c) 1982, 2006, Oracle.  All Rig


Connected to:
Oracle Database 10g Enterprise Edition Rel
With the Partitioning, OLAP and Data Minin

SQL> update emp
  2  set sal=1200
  3  where ename='SMITH';

1 row updated.

SQL>
```

```
Oracle SQL*Plus
File  Edit  Search  Options  Help

SQL*Plus: Release 10.2.0.3.0 - Production on Fri Mar 8 16:23:19 2019

Copyright (c) 1982, 2006, Oracle.  All Rights Reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL> set linesize10
SQL>
SQL> set linesize100
SQL> update emp
  2  set sal=1200
  3  where ename='ALLEN';

1 row updated.

SQL>
```

Lesson243: Deadlocks

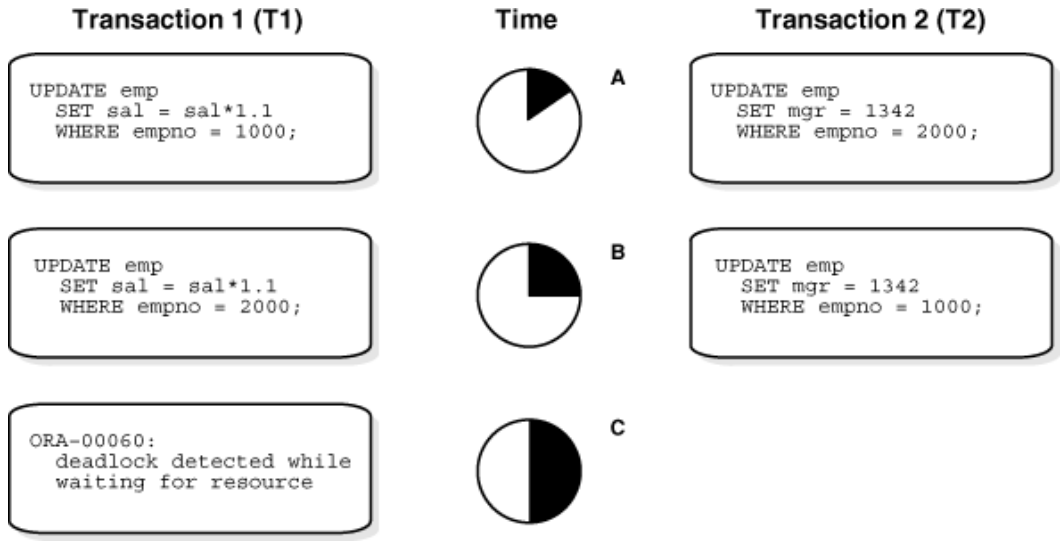
Deadlocks

- Database Deadlocks
- Deadlocks - Two Transactions
- Applying Locks at Table Level
- Deadlock - Example

Database Deadlocks

In a **database**, a **deadlock** is a situation in which two or more transactions are waiting for one another to give up locks. For example, Transaction A might hold a lock on some rows in the Accounts table and needs to update some rows in the Orders table to finish.

Deadlocks – Two Transactions



Applying Locks at Table Level

```
--user scott
Lock table dept IN EXCLUSIVE MODE;

--user can use rollback to unlock this table

--user sys
 select object_id, session_id, oracle_username, locked_mode
 from V$Locked_Object;

 OBJECT_ID SESSION_ID ORACLE_USERNAME                LOCKED_MODE
---------- ---------- ------------------------------ -----------
     53768        140 SCOTT                                    6  (Locked applied)
     53768        148 SCOTT                                    0  (Locked in other session)

--in session 148
SQL> update dept
  2  set loc='BOSTON1'
  3  where deptno=40;

----LOCKED-------

How to put in share mode?
Lock table dept in share mode;
```

Deadlock - Example

Time	T1	T2
t1	XLock(x)	…
t2	…	XLock(y)
t3	Request y	…
t4	wait	Request x
t5	wait	wait
t6	wait	wait

Lesson244: DDL Schema Script for Project

DDL Schema Script for Project

- Plan the Script for Schema
- Sample Script

Plan the Script for Schema

ANSI standard script of SQL consists of

Create and Alter Statements

Standard & Additional Constraints

Tables with Tablespace Storage

Database Security

Shared Logical Objects such as Views, Synonyms

Logical & Physical Storage

DML – SQL

Sample Script

```
Drop table employee;
Drop table orders;
Drop table customer;
Drop table Item;

create table Customer
(custID          CHAR(10),
CustName         VARCHAR2(25) NOT NULL,
Address          VARCHAR2(50),
City             VARCHAR2(50),
TelNo            VARCHAR2(15),
Email            VARCHAR2(50) UNIQUE)
tablespace TS_Sales;

Alter table Customer
add constraint PK_Custid primary key (CustID);

Create table Orders
(InvoiceNo       CHAR(12),
InvType          CHAR(1),
InvDate          Date NOT NULL,
TotalPrice       Number(6,2),
SalesEmpID       CHAR(5),
CustID           CHAR(8))
tablespace TS_Sales;

Alter table Orders
add constraint PK_InvNo primary key (InvoiceNO)
USING INDEX TABLESPACE Sale_indx;

Alter table Orders
add constraint FK_Cust_CustID foreign key (custID)
references Customer(CustID);
```

Alter Table with Move on Storage

```
ALTER TABLE hr.admin_emp MOVE
      STORAGE ( INITIAL 20K
                NEXT 40K
                MINEXTENTS 2
                MAXEXTENTS 20
                PCTINCREASE 0 )
  TABLESPACE hr_tbs;

ALTER TABLE hr.admin_emp
       ADD (bonus NUMBER (7,2));
```

Lesson245:DDL Schema Script including Tables, Columns and Data Types

DDL Schema Script including Tables, Columns and Data Types

- DDL Script for Schema
- Sample Create Table - SQL
- Sample Create/ Alter Table with Constraints
- Sample Create Table with Additional Constraints

DDL Script for Schema

Scripts are used to create objects (structures) such as tables, sequences, views and PL/SQL programs etc in a schema. Many script contains DDL SQL.

Script can contain DML for populating data for development and testing.

Similarly, structure of schema objects can be populated using exported DMP file.

Sample Create Table - SQL

```
CREATE TABLE hr.admin_emp (
          empno       NUMBER(5) PRIMARY KEY,
          ename       VARCHAR2(15) NOT NULL,
          ssn         NUMBER(9) ENCRYPT USING 'AES256',
          job         VARCHAR2(10),
          mgr         NUMBER(5),
          hiredate    DATE DEFAULT (sysdate),
          photo       BLOB,
          sal         NUMBER(7,2),
          hrly_rate   NUMBER(7,2) GENERATED ALWAYS AS (sal/2080),
          comm        NUMBER(7,2),
          deptno      NUMBER(3) NOT NULL
                       CONSTRAINT admin_dept_fkey REFERENCES hr.departments
                       (department_id))
   TABLESPACE admin_tbs
   STORAGE ( INITIAL 50K);

COMMENT ON TABLE hr.admin_emp IS 'Enhanced employee table';
```

Sample Create/ Alter Table with Constraints

```
Drop table employee;
Drop table orders;
Drop table customer;
Drop table Item;

create table Customer
(custID          CHAR(10),
CustName         VARCHAR2(25) NOT NULL,
Address          VARCHAR2(50),
City             VARCHAR2(50),
TelNo            VARCHAR2(15),
Email            VARCHAR2(50) UNIQUE)
tablespace TS_Sales;

Alter table Customer
add constraint PK_Custid primary key (CustID);

Create table Orders
(InvoiceNo       CHAR(12),
InvType          CHAR(1),
InvDate          Date NOT NULL,
TotalPrice       Number(6,2),
SalesEmpID       CHAR(5),
CustID           CHAR(8))
tablespace TS_Sales;

Alter table Orders
add constraint PK_InvNo primary key (InvoiceNO)
USING INDEX TABLESPACE Sale_indx;

Alter table Orders
add constraint FK_Cust_CustID foreign key (custID)
references Customer(CustID);
```

Sample Create Table with Additional Constraints

```
create table Employee
(EmpID           CHAR(3),
EmpName          VARCHAR2(25) NOT NULL,
Address          VARCHAR2(50),
City             VARCHAR2(50),
TelNo            VARCHAR2(15),
Email            VARCHAR2(50) UNIQUE,
Status           CHAR(1))
tablespace TS_Sales;

Alter table Employee
add constraint ck_status
check (Status in ('Y', 'N'));

Alter table Employee
add constraint PK_EmpID primary key (EmpID)
USING INDEX TABLESPACE Sale_indx;
```

Lesson246: DDL Schema Script including Allocation of Tablespaces with Sizes

DDL Schema Script for Tablespaces

- Logical & Physical Storage
- Data Tablespaces for Tables
- Tablespace for Indexes
- Temporary Tablespace for Project

Logical & Physical Storage

- Creating tablespaces for data, indexes
- Tablespaces for table or tables
- Tablespace with OS level storage

Data Tablespaces for Tables

```
Create tablespace MedCare Datafile
    'D:\oracle9i\ora92\oradata\MCare01.dbf' Size 150M online
    MINIMUM EXTENT 64K
    DEFAULT STORAGE (INITIAL 64K NEXT 64K);


Create tablespace MCare_IVF Datafile
     'D:\oracle9i\ora92\oradata\MCIVF01.dbf' Size 50M online
     MINIMUM EXTENT 64K
     DEFAULT STORAGE (INITIAL 64K NEXT 64K);

CREATE TABLESPACE MCare_IVFCHART DATAFILE
    'D:\oracle9i\ora92\oradata\MCare_IVFCHART01.dbf' SIZE 50M
    MINIMUM EXTENT 1024K
    DEFAULT STORAGE (INITIAL 1024K NEXT 1024K);
```

Tablespace for Indexes

```
CREATE TABLESPACE MCare_indx
DATAFILE 'D:\oracle9i\ora92\oradata\MCare_indx01.dbf' SIZE 100M
MINIMUM EXTENT 64K
DEFAULT STORAGE (INITIAL 64K NEXT 64K);
```

Temporary Tablespace for Project

```
CREATE TEMPORARY TABLESPACE "TEMP_DEFAULT3" TEMPFILE
    'D:\ORACLE9I\ORA92\ORADATA\eahmed\TEMP_DEFAUL3.DBF' SIZE
    1000M EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M;

ALTER DATABASE DEFAULT TEMPORARY TABLESPACE "TEMP_DEFAULT3";
```

Lesson247: Creation of DML Insert Statements for Data

DML Schema Script

- DDL & DML Script
- Sample DML Insert Statements
- Sample DML Insert & Update Statements

DDL & DML Script

- One script file can be created to manage DDL & DML commands
- Separate files for DDL & DML
- DML Insert, Delete and Update commands
- All commands are in an order
- Well-defined datasets are selected

Sample DML Insert Statements

```
insert into customer(
CUSTID,
CUSTNAME,
ADDRESS,
CITY,
TELNO,
EMAIL)
values(
'88700110',
'3M Co',
'41 Main st.',
'Lahore',
'0514568877',
'ejaz@nu.edu.pk'
);

insert into customer(
CUSTID,
CUSTNAME,
ADDRESS,
```

Sample DML Insert & Update Statements

```
Insert into Customer

...

Update Sold
set qty = (select qty
           from Item
           where ItemID='254011');
```

Lesson248: DDL Schema Script including List of Roles and Privileges

Script for Security

- Roles & Privileges
- Creating User with Grants

Roles & Privileges

- Create roles
- Assign privileges to roles
- Assign privileges to users

Creating User with Grants

```
create tablespace work
datafile 'E:\oracle\product\10.2.0\db_2\oradata\work01.dbf'size 500M online;

create user uwork
identified by xyz
default tablespace work
temporary tablespace temp;

grant connect, resource to uwork;
grant create table to uwork;


create tablespace work_indx
datafile 'E:\oracle\product\10.2.0\db_2\oradata\WIndx01.dbf'size 200M online;

create tablespace work_depend
datafile 'E:\oracle\product\10.2.0\db_2\oradata\WDepend01.dbf'size 300M online;
```

Creating User with Grants

```
grant select on MED_STOCK  to public;
grant select on TRANSACTION   to public;
grant select on TRANS_DETAIL  to public;


grant select, insert, update on EMP to public;
grant select, insert, update on DEPT to ejaz;
```

Lesson249: DDL Schema Script including Synonyms of Tables & Drop of all Objects

Synonyms & Drop Objects

- Generate SQL Script using SELECT Clause
- Creating Synonyms
- Drop Synonyms
- Drop Tables

Generate SQL Script using Select Clause

```
--Table EMP owner (scott) gives grant:
grant select on emp to u1;
--generate the following SQLs from scott schema and run in u1 schema
select 'create synonym '||tname||' for scott.'||tname||';'
from tab;
```

Creating Synonyms

```
grant create synonym to u1;
grant create public synonym to u1;
grant create synonym to scott;
grant create public synonym to scott;
```

Drop Synonyms

```
--inside same schema of scott:
create public synonym syn_emp for scott.emp;

drop synonym dept;
drop synonym emp;
```

Drop Tables

```
drop table EMP;
drop table ProjectAssign;
drop table DEPT;
drop project;
```

Lesson250: Revision of Course Contents

Revision of DBA Course

- Introduction to Database & Architecture
- Database Design (Development)
- DBA
- PL/SQL
- Project's Preparations & Scripts
- Logical & Physical Storage

Introduction to Database & Architecture

- What is Database?
- What is DBMS?
- ANSI SPARK 3 Level Architecture
- 2 & 3 Tiers Client Server Architecture
- Keys & Constraints

Database Design (Development)

- What is ER Model?
- ER Model Artifacts
- ER Mapping & Implementation

DBA

- DBA tasks & responsibilities
- Data Dictionary
- Oracle Installation & Configuration
- Logical & Physical Storage
- Backup & Recovery
- Database Tuning & Performance
- Support & Services
- Transaction Processing

PLSQL

- Procedure programming using Blocks
- Programming exceptions
- Procedures, functions & packages
- Cursor and loops
- Advance topics using Arrays, Dynamic SQL.

Project's Preparation and Scripts

- Database Mapping or Implementation
- Scripts for DDL
- Scripts for logical tablespaces
- Scripts for Security
- Data Migration

Logical & Physical Storage