By Maha Rana

Cs312 Final Term Preparation File

Module 12: Anomalies

Conceptual Questions

1. What are anomalies in database design? Provide examples.

Anomalies in database design refer to issues that arise when data is not properly structured or normalized. These problems can lead to inconsistencies, redundancy, and errors in the database. The most common types of anomalies include insert, delete, and update anomalies.

Examples:

- **Redundancy:** Storing the same data multiple times, e.g., storing an employee's address in multiple tables, leading to data inconsistency.
- **Inconsistency:** When data across different parts of the database contradicts, e.g., a different record for the same employee showing different addresses.

2. explains the Insert, Delete, and Update anomalies with scenarios?

Insert Anomaly: This occurs when data cannot be inserted into a database without having unnecessary or incomplete information.

Scenario: In a database that stores employee and department details, if an employee hasn't been assigned to a department yet, it might be impossible to insert a new employee record because the department field is mandatory.

Delete Anomaly: This happens when deleting data leads to the unintended loss of other related data.

Scenario: In the same employee-department database, if an employee is deleted from a department, but that department has no other employees, the deletion could also result in losing the entire department record.

Update Anomaly: This occurs when a single change in data requires multiple updates, which might not happen consistently, leading to data inconsistency.

Scenario: If an employee's salary is updated but not all records reflecting that

Maha Rana

employee's salary are updated in different tables, some records may show outdated data.

3. Why is it important to eliminate anomalies in a large database?

Eliminating anomalies is crucial for maintaining data integrity, reducing redundancy, and ensuring that the database is scalable and easy to maintain. Without addressing anomalies:

- **Data Integrity:** Inconsistent data could lead to incorrect business decisions.
- **Efficiency:** Anomalies increase the likelihood of errors and complexity in data management tasks like backups, updates, and querying.
- **Scalability:** A database with anomalies is harder to expand or modify as it grows.

In large databases, addressing these issues early through techniques like normalization helps avoid future complications and ensures smoother operations.

MSQs:

1. Which anomaly occurs when certain data cannot be entered without entering unwanted data?
a) Insert
b) Update
c) Delete
d) None of these
Answer: **a) Insert**

2. What is the primary cause of anomalies in database design?
a) Redundancy
b) Normalization
c) Lack of primary keys
d) Indexing
Answer: **a) Redundancy**

Module 13: Normalization

Conceptual Questions

Maha Rana

1. Define normalization and its objectives?

Normalization is a database design technique used to reduce redundancy and dependency by organizing data into multiple related tables. It aims to improve data integrity and eliminate anomalies.

Objectives:

Reduce Redundancy: Avoid duplicate data storage.

Eliminate Anomalies: Prevent insert, delete, and update anomalies.

Improve Data Integrity: Ensure consistency and accuracy of data.

Enhance Scalability: Simplify database maintenance as the system grows.

2. What are the rules for First Normal Form (1NF)?

Rules for First Normal Form (1NF):

A table is in 1NF if:

Atomic Values: All columns contain atomic (indivisible) values. No repeating groups or arrays.

Uniqueness: Each row in the table must be unique and identified by a primary key.

Consistent Domain: Values in a column must be of the same type (e.g., no mixing integers and strings).

Example:

StudentID	Subjects
1	Math, Science

This is not in 1NF because "Math, Science" is not atomic.

Maha Rana

To convert to 1NF:

StudentID	Subject
1	Math
1	Science

3. Differentiate between Second Normal Form (2NF) and Third Normal Form (3NF).

Second Normal Form (2NF):

A table is in 2NF if it is in 1NF and all non-key attributes are fully functionally dependent on the primary key.

No Partial Dependency: A non-key attribute must not depend on only part of a composite primary key.

Focus: Eliminates partial dependencies.

Example of partial dependency:

StudentID	CourseID	CourseName
1	101	Math

Here, CourseName depends only on CourseID, not on the composite key (StudentID, CourseID).

To normalize, split into:

| CourseID | CourseName |

| StudentID | CourseID |

Third Normal Form (3NF):

A table is in 3NF if it is in 2NF and all non-key attributes are only dependent on the primary key (not on other non-key attributes).

Maha Rana

No Transitive Dependency: Non-key attributes must not depend on other non-key attributes.

Example of transitive dependency:

EmployeeIDDepartmentID DepartmentName

Here, DepartmentName depends on DepartmentID, not on EmployeeID.

To normalize, split into:

| DepartmentID | DepartmentName |

| EmployeeID | DepartmentID |

Key Differences:

2NF resolves partial dependencies.

3NF resolves transitive dependencies.

MSQs:

1. Which normal form eliminates partial dependency?

a) 1NF

b) 2NF

c) 3NF

d) BCNF

Answer: b) 2NF

2. in normalization, transitive dependency is removed in which form?

a) 1NF

b) 2NF

Maha Rana

c) 3NF

d) None of these

Answer: c) 3NF

Lecture 14: Demoralization:

Conceptual Questions

What is demoralization, and why is it used in databases?

Demoralization is the process of combining normalized tables into fewer tables by introducing redundancy to improve database performance. It involves reversing some of the normalization steps to reduce the complexity of queries and improve read operations.

Why is it used?

To reduce the number of JOIN operations in queries, which can be computationally expensive?

To improve query performance in scenarios where quick access to data is critical, such as reporting or analytics.

To optimize databases for read-heavy workloads, where data retrieval speed is prioritized over write efficiency.

Explain the trade-off between normalization and demoralization?

Trade-Off: Normalization focuses on minimizing redundancy and ensuring data integrity, while demoralization prioritizes performance and faster data retrieval, especially in read-heavy environments.

Describe scenarios where demoralization improves performance?

Data Warehousing and Reporting:

When databases are used for analytics or business intelligence, demoralization speeds up queries by providing pre-joined tables.

Maha Rana

Example: Combining sales, customer, and product details into a single table for faster report generation.

High-Performance Applications:

Applications requiring low-latency responses, such as e-commerce or social media platforms, benefit from demoralized structures to serve user requests quickly.

Example: Including frequently accessed product details (e.g., price, name) directly in an order table.

Caching Data for Read-Heavy Workloads:

When a database is optimized for reading rather than writing, denormalization reduces JOINs, enabling faster data retrieval.

Example: Storing aggregated sales data in a single table to avoid calculating it dynamically.

Real-Time Systems:

Systems that prioritize speed over storage efficiency, such as recommendation engines or dashboards, often use demoralized tables to deliver results instantly.

Example: Combining user activity logs and profile data for faster recommendations.

MSQs:

Which of the following is the primary goal of demoralization?

a) Reduce data redundancy

b) Improve query performance

c) Increase database size

d) Maintain constraints

Maha Rana

Answer: b) Improve query performance

What does demoralization involve?

a) Breaking tables into smaller parts

b) Adding redundant data for faster queries

c) Removing foreign keys

d) None of these

Answer: b) Adding redundant data for faster queries

Module 18 SQL Basics:

Conceptual Question:

Define SQL and its major components?

SQL (Structured Query Language) is a standard language used to communicate with relational databases. It is used for managing, querying, and manipulating structured data.

Major Components of SQL:

Data Query Language (DQL): Used for retrieving data from a database. Example: SELECT

Data Definition Language (DDL): Used for defining or altering database structures. Example: CREATE, ALTER, DROP

Data Manipulation Language (DML): Used for modifying data in a database. Example: INSERT, UPDATE, DELETE

Data Control Language (DCL): Used for managing user permissions. Example: GRANT, REVOKE

Maha Rana

Transaction Control Language (TCL): Used for managing database transactions. Example: COMMIT, ROLLBACK

Explain the role of SELECT, INSERT, UPDATE, and DELETE commands in SQL?

SELECT: Used to retrieve data from one or more tables.

Example:

SELECT name, age FROM students WHERE age > 18;

INSERT: Used to add new records to a table.

Example:

INSERT INTO students (name, age) VALUES ('John', 20);

UPDATE: Used to modify existing records in a table.

Example:

UPDATE students SET age = 21 WHERE name = 'John';

DELETE: Used to remove records from a table.

Example:

DELETE FROM students WHERE age < 18;

How does the WHERE clause help filter data?

The WHERE clause is used to filter records based on specific conditions, allowing only those rows that meet the criteria to be included in the result set. It can be used with SELECT, UPDATE, DELETE, and other SQL statements.

Examples:

Retrieve specific data:

Maha Rana

SELECT * FROM employees WHERE department = 'HR';

This retrieves only employees in the HR department.

Update selected rows:

UPDATE employees SET salary = salary * 1.1 WHERE department = 'Sales';

This increases the salary of employees in the Sales department by 10%.

Delete specific records:

DELETE FROM orders WHERE order_date < '2023-01-01';

This removes orders placed before January 1, 2023.

The WHERE clause ensures precision and efficiency by limiting the scope of SQL operations.

MSQs:

Which SQL keyword is used to retrieve unique values?

a) UNIQUE

b) DISTINCT

c) PRIMARY

d) FILTER

Answer: b) DISTINCT

Which command is used to remove all rows from a table?

a) DELETE

b) DROP

c) TRUNCATE

Maha Rana

d) REMOVE

Answer: c) TRUNCATE

Module 23: Transactions

Conceptual Questions:

1. **Explain the ACID properties of a transaction with examples?**

The **ACID** properties ensure that database transactions are processed reliably and ensure data integrity, even in the event of a system failure or errors. ACID stands for:

Atomicity: This ensures that a transaction is treated as a single, indivisible unit. Either all the operations in a transaction are completed successfully, or none are applied.

Example: If a bank transfer involves two steps—deducting money from Account A and depositing it into Account B—atomicity ensures that both actions occur together. If one part fails (e.g., money is deducted but not added to Account B), the entire transaction will be rolled back.

Consistency: A transaction takes the database from one valid state to another, maintaining data integrity. It ensures that after the transaction, the database meets all the defined rules (constraints, triggers, etc.).

Example: In a banking system, the consistency property ensures that no customer ends up with a negative balance after a transaction, maintaining the business rule that balances cannot fall below zero.

Isolation: Ensures that concurrently executed transactions do not affect each other. Each transaction should operate as if it were the only transaction in the system, even if multiple transactions are running at the same time.

Example: If two customers try to withdraw money from the same account at the same time, isolation ensures that each transaction sees a consistent version of the data. Either one transaction will execute fully before the other, or a method like locking will prevent one from running simultaneously, maintaining consistency.

Maha Rana

Durability: Once a transaction is committed, it cannot be undone, even in the event of a system crash. The changes made by a completed transaction are permanent.

Example: After transferring funds from one account to another, the new balances will persist in the database, even if the system crashes immediately after the transaction is committed.

2. Why is isolation important in concurrent transactions?

Isolation is crucial for preventing conflicts and ensuring the consistency and correctness of the database in a multi-user environment. Without isolation, the following issues can occur in concurrent transactions:

Dirty Reads: A transaction might read data that is written by another ongoing transaction, which has not yet been committed. If the second transaction is rolled back, the first transaction would have based its actions on invalid or temporary data.

Example: Transaction A updates the salary of an employee but has not yet committed. Transaction B reads the new value before Transaction A commits. If Transaction A rolls back, Transaction B has based its actions on invalid data.

Non-repeatable Reads: If a transaction reads a value, and that value is changed by another transaction before the first transaction is complete, the first transaction will see inconsistent data.

Example: Transaction A reads an employee's salary, and Transaction B changes the same salary before Transaction A is committed. Transaction A will get a different result if it reads the salary again.

Phantom Reads: When a transaction reads a set of rows, but another transaction inserts, updates, or deletes rows in that set before the first transaction completes.

Example: Transaction A reads a list of active users, while Transaction B inserts a new user. When Transaction A re-reads the list, it sees a new record that did not exist in the original read.

By ensuring that transactions are isolated, databases prevent these anomalies and maintain consistency across concurrent transactions.

Maha Rana

3. Describe the durability property of a transaction?

The **Durability** property ensures that once a transaction has been committed, the changes made to the database will persist, even in the event of a system failure, power outage, or crash. This property guarantees the permanence of the transaction's effects, and data will not be lost once a transaction has been committed.

Example: In an online shopping transaction, after a payment is processed and the transaction is committed, the purchase details (product, amount, customer information) are guaranteed to be saved to the database. Even if the system crashes immediately after the commit, the data will not be lost because it has been written to persistent storage (disk or equivalent).

In essence, durability ensures that once a transaction is completed and the system acknowledges its success (commits it), its effects are stored securely and will persist indefinitely. Even in case of failure, the transaction's results will be recoverable.

MSQs:

1. **Which property ensures that a transaction is treated as a single unit of work?**
   a) Atomicity
   b) Consistency
   c) Isolation
   d) Durability
   Answer: a) Atomicity

2. **What happens if a transaction violates the consistency property?**
   a) Data loss
   b) Dirty reads
   c) Database in an inconsistent state
   d) None of these
   Answer: c) Database in an inconsistent state

Maha Rana

Module 24: Locks & Granularity

Conceptual Questions:

1. **What are locks in a database, and why are they important?**
   Locks in a database are mechanisms used to manage concurrent access to data. They prevent conflicts, maintain data integrity, and ensure consistency when multiple users or processes attempt to access or modify the same data simultaneously.

   Importance of Locks:

   Data Integrity: Prevents conflicting operations on the same data.

   Consistency: Ensures that transactions follow ACID (Atomicity, Consistency, Isolation, and Durability) principles.

   Concurrency Control: Manages simultaneous access to avoid issues like deadlocks or race conditions.

2. **Explain the concept of granularity in locks with examples?**
   Granularity in locks refers to the level at which locks are applied, ranging from fine-grained (smaller units) to coarse-grained (larger units).

   Levels of Granularity:

   - **Row-Level Lock:** Locks a single row.
     Example: A bank application locks a specific account row during a transfer to prevent simultaneous updates.

   - **Page-Level Lock:** Locks a group of rows (a page).
     Example: A database locks a 4KB page containing 10 rows during batch processing.

Maha Rana

- **Table-Level Lock:** Locks the entire table.
  Example: During a schema change, the entire table is locked to prevent access.

- **Database-Level Lock:** Locks the entire database.
  Example: Backup operations lock the database to ensure consistency during the process.

3. **Discuss the difference between shared and exclusive locks**

Shared Lock:

Example:

SELECT * FROM employees WHERE department = 'HR';

Multiple users can read data from the HR department simultaneously.

Exclusive Lock:

Example:

UPDATE employees SET salary = salary + 500 WHERE department = 'Sales';

Only one user can update salaries in the Sales department at a time to prevent inconsistencies.

By managing shared and exclusive locks appropriately, databases achieve a balance between concurrency and data integrity.

MSQs:

1. **Which type of lock allows multiple users to read data but prevents writing?**
   a) Exclusive
   b) Shared
   c) Row
   d) Table
   Answer: b) Shared

Maha Rana

2. **What is the main purpose of database locks?**
   a) Improve performance
   b) Prevent data corruption
   c) Reduce data size
   d) None of these
   Answer: b) Prevent data corruption

Module 25: Indexing

Conceptual Questions

1. **What is an index in a database, and why is it important?**
   An index is a database object that improves the speed of data retrieval operations by creating a data structure that allows quick searches. It works like a catalog or a table of contents in a book, helping the database system locate rows without scanning the entire table.

   Importance:

   - Speeds up query execution.
   - Optimizes database performance.
   - Facilitates quick access to rows based on column values.

2. **Explain the types of indexes used in relational databases.**

Primary Index: Automatically created for primary key columns. Ensures each row has a unique identifier.

Unique Index: Ensures that all values in a column or combination of columns are unique.

Composite Index: Created on two or more columns. Useful for queries involving multiple columns.

Maha Rana

Clustered Index: Physically reorders the data in the table to match the index. Each table can have only one clustered index.

Non-Clustered Index: Maintains a separate structure for indexing without altering the physical order of table data.

3. **How does indexing improve query performance?**
   Indexing improves query performance by reducing the amount of data the database needs to scan:

Quick Lookup: Indexes allow the database to find rows directly instead of scanning the whole table.

Efficient Sorting: Data in an index is sorted, enabling faster range queries and ORDER BY operations.

Minimized I/O: Fewer data blocks are accessed, reducing input/output operations.

Example: Searching for a specific employee ID in an indexed column takes O(log n) time compared to O(n) in an unhindered column.

MSQs:

1. **Which of the following is a type of database index?**
   a) Composite index
   b) Redundant index
   c) Primary index
   d) Both a and c
   Answer: d) Both a and c

2. **What happens if an index is not maintained properly?**
   a) Faster queries
   b) Data corruption
   c) Slower query performance
   d) None of these
   Answer: c) Slower query performance

Maha Rana

May Allah grant you success, ease your efforts, and bless you with wisdom and perseverance. Keep your faith strong, for with prayer, every challenge becomes an opportunity. I hope this file helps you a lot, and please remember me in your prayers.

Best wishes for Finals!
Maha ☺

Maha Rana