DBMS UNIT_1 IMPS

1.What is impact of insert ,update, delete, anomaly on overall design of database? How normalisation used to remove these anomalies?

The impact of **insert, update, and delete anomalies** on the overall design of a database affects its **data integrity**, **consistency**, and **efficiency**:

Insert Anomaly

An insert anomaly occurs when certain attributes cannot be inserted into the database without the presence of other attributes.

Impact: This limits the ability to add data freely and can cause data redundancy and inconsistencies.

Solution through Normalization: Normalization helps by separating the data into different related tables, ensuring that you can insert a new record without requiring unrelated data. For example:

- First Normal Form (1NF) removes repeating groups
- Third Normal Form (3NF) removes transitive dependencies

Update Anomaly

An update anomaly occurs when redundant data exists and a change to one instance of that data doesn't automatically apply to all instances.

Impact: This results in data inconsistencies across the database and a high maintenance cost.

Solution through Normalization: Normalization reduces redundancy by ensuring that data is stored in only one place. By normalizing the database:

- In 2NF, partial dependencies are removed
- In 3NF, reducing redundancy.

Delete Anomaly

A delete anomaly occurs when the deletion of data results in the unintended loss of additional data.

Impact: This leads to the accidental removal of important data and can compromise data integrity.

Solution through Normalization: Normalization splits the data into logically related tables. In this case:

- In 2NF or 3NF,

Study material provided by: Vishwajeet Londhe

Join Community by clicking below links

Telegram Channel

https://t.me/SPPU_TE_BE_COMP

(for all engineering Resources)

WhatsApp Channel

(for all tech updates)

https://whatsapp.com/channel/0029ValjFriICVfpcV9HFc3b

Insta Page

(for all engg & tech updates)

https://www.instagram.com/sppu_engineering_update

2. Explain 2NF,3NF,BNCF?

Point	2NF	3NF	BCNF
1	Eliminates partial dependencies.	Eliminates transitive dependencies.	Every determinant must be a superkey.
2	Must be in 1NF.	Must be in 2NF.	Must be in 3NF.
3	May still have transitive redundancy.	May still have redundancy due to non-superkeys.	No redundancy from functional dependencies.
4	Focuses on composite key issues.	Focuses on indirect dependencies.	Focuses on superkey requirements.
5	Reduces redundancy from partial dependencies.	Reduces redundancy from transitive dependencies.	Eliminates all anomalies from non-superkeys.
6	Non-prime attributes can still depend on non-prime attributes.	Non-prime attributes cannot depend on other non-prime attributes.	All functional dependencies must involve superkeys.
7	Example violation: Non-prime attribute depending on part of composite key.	Example violation: Non-prime attribute depending on another non-prime attribute.	Example violation: Non-superkey determining attributes.
8	Involves splitting tables to reduce redundancy.	Involves further splitting to remove transitive dependencies.	Involves restructuring to ensure all determinants are superkeys.
9	Focuses on primary keys and their attributes.	Focuses on the relationship between attributes.	Focuses on enforcing superkey constraints.
10	Less strict than 3NF and BCNF.	Stricter than 2NF but less than BCNF.	Stricter than both 2NF and 3NF.

Guaranteed Access Rule

1. Definition:
   - Every piece of data in a relational database should be accessible using a combination of table name, primary key, and column name.
2. Access Method:
   - Data access should rely on the logical structure of the database, not on physical storage.
3. Primary Key Requirement:
   - Each table must have a primary key that uniquely identifies each record.
4. Example:
   - Consider a table named **Employees**:

1. Employee ID	1. Name	1. Department
1. 101	1. Alice Johnson	1. HR
1. 102	1. Bob Smith	1. IT
1. 103	1. Carol Williams	1. Finance

   - To access the name of the employee with **EmployeeID** 102, you would use:sql Copy code

   - SELECT Name FROM Employees WHERE EmployeeID = 102;

Integrity independence

Definition: Integrity constraints (rules ensuring data accuracy) must be stored in the database catalog, not in application programs.
Centralized Management: Allows easy modification of integrity rules without changing application code.
Automatic Enforcement: The database management system (DBMS) automatically enforces these constraints during data operations.
Application Independence: Changes to integrity constraints do not require updates to all applications accessing the data, enhancing maintainability.
Examples: Common integrity constraints include primary keys (ensuring uniqueness) and foreign keys (enforcing relationships between tables).

```
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50)
);
```

```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

compretensive data sub language rule

Single Language Requirement: A relational database must support a single comprehensive data sublanguage for all database operations.

Components Included: This language should encompass Data Definition Language (DDL), Data Manipulation Language (DML), and Data Control Language (DCL).

Expressiveness: It must be capable of defining, querying, modifying, and controlling data effectively.

Transaction Management: The language should support transaction management, allowing operations to be executed reliably and ensuring data integrity.

Integration with Programming: It should be usable in application programming languages as well as in interactive query sessions, facilitating diverse data interactions.

Systematic Treatment of Null Values

The **Systematic Treatment of Null Values** is one of Codd's rules ensuring consistent handling of `NULL` (representing missing or unknown data) in relational databases. Here's a brief summary:

1. **Distinct Meaning**: `NULL` represents missing or unknown data, not zero or an empty string.
2. **Uniform Handling**: The DBMS must treat `NULL` values consistently in all operations (queries, updates, etc.).
3. **Three-Valued Logic**: Conditions involving `NULL` can result in `TRUE`, `FALSE`, or `UNKNOWN`, affecting query results.
4. **Ignored in Aggregation**: Functions like `SUM()` and `COUNT()` ignore `NULL` values.
5. **Special Handling**: Queries use `IS NULL` or `IS NOT NULL` to explicitly check for `NULL` values.

3.features of good relations database design?

1. **Normalization**: Organize data to minimize redundancy and ensure consistency.
2. **Primary Key**: Ensure every table has a unique primary key to identify records.
3. **Foreign Keys**: Use foreign keys to enforce relationships between tables.
4. **Data Integrity**: Maintain entity, referential, and domain integrity with constraints.
5. **Efficient Indexing**: Apply appropriate indexes to speed up data retrieval.
6. **Avoiding Over-Indexing**: Use indexes wisely, as too many can slow down write operations.
7. **Optimized Queries**: Design efficient SQL queries for better performance.
8. **Scalability**: Design for future growth with partitioning or sharding when necessary.
9. **Consistency**: Ensure consistency by following best practices like ACID (Atomicity, Consistency, Isolation, Durability).
10. **Security**: Implement security measures like user roles, permissions, and encryption to protect data.

4.What is decomposition?

Decomposition in relational databases is the process of breaking a large, complex table into smaller, related tables to eliminate redundancy, prevent anomalies, and improve data structure. It helps in organizing data efficiently while preserving its integrity.

Key Points:

1. **Lossless Decomposition**: Ensures the original table can be fully reconstructed without losing information.
2. **Dependency Preservation**: Maintains the functional dependencies of the original table.
3. **Goals**: Reduce redundancy, prevent insertion/update/deletion anomalies, and improve query performance.

5. Functional dependency?

A **functional dependency** in a database occurs when one attribute (or set of attributes) uniquely determines another. It's represented as **X → Y**, meaning that if two rows have the same value for **X**, they must have the same value for **Y**.

Importance:

Functional dependencies are crucial for:

- **Normalization**: Reducing redundancy and preventing anomalies.
- **Database Design**: Ensuring proper relationships between attributes.

Unit 5_IMPS

1.Comapare SQL and NOSQL

Feature	SQL	NoSQL
Data Model	Relational (tables with fixed schema)	Non-relational (documents, key-value, wide-column, graph)
Schema	Predefined and rigid	Dynamic and flexible
Scalability	Vertically scalable (hardware upgrades)	Horizontally scalable (across servers)
Transactions	ACID compliance (strong consistency)	Eventual consistency (BASE model, flexibility with ACID)
Use Case	Structured data, complex queries	Unstructured data, flexible data storage
Joins	Supports complex joins	Typically no joins (optimized for fast queries)
Query Language	SQL (Structured Query Language)	Varies (e.g., JSON, BSON, NoSQL APIs)
Performance	Better for complex queries, analytics	Better for high-volume, simple read/write operations
Examples	MySQL, PostgreSQL, Oracle	MongoDB, Cassandra, Redis
Storage Type	Row-based storage	Document, key-value, column, graph

2.Explain NOSQL data types

Document-Oriented Databases

- **Structure**: Store data as documents (usually in formats like JSON, BSON, or XML).
- **Storage**: Documents are grouped into collections.
- **Schema**: Flexible, allowing each document to have a different structure.
- **Use Cases**: Suitable for content management, user profiles, and catalogs where data can vary in structure.
- **Examples**: MongoDB, CouchDB.

Key-Value Stores

- **Structure**: Simple key-value pairs, where a key is associated with a specific data value.
- **Storage**: Data is retrieved by a unique key, similar to a dictionary/hashmap.
- **Schema**: No strict schema; the value can be any data type like string, JSON, or binary objects.
- **Use Cases**: Ideal for caching, session management, and user preferences.
- **Examples**: Redis, DynamoDB, Riak.

Graph Databases

- **Structure**: Data is represented as nodes, edges (relationships), and properties (attributes).
- **Storage**: Focuses on relationships between data points.
- **Schema**: Flexible, designed to model complex networks and relationships.
- **Use Cases**: Best suited for social networks, recommendation engines, fraud detection, and complex relationship mapping.
- **Examples**: Neo4j, ArangoDB, Amazon Neptune.
- **Example Data**:
    - Nodes: Users (John, Jane)
    -

3.Base properties of NOSQL

Schema Flexibility: No predefined schema, supports flexible data structures.

Horizontal Scalability: Scales by adding more servers (nodes).

Distributed Architecture: Data distributed across multiple nodes for fault tolerance.

Eventual Consistency: Ensures data consistency over time but may allow temporary inconsistencies.

BASE Model: Prioritizes availability and scalability over strong consistency (opposite of ACID).

High Availability: Designed for minimal downtime with automatic replication and failover.

Big Data Handling: Optimized for managing large volumes of both structured and unstructured data.

High Performance: Efficient read/write operations, especially in distributed environments.

Flexible Data Models: Supports documents, key-value pairs, columns, or graph structures.

Open-Source and Cost-Effective: Many NoSQL databases are open-source and lower in cost for scaling large data systems.

Q.3 Write short note on - CAP Theorem.

Ans. :

- Cap theorem is also called as brewer's theorem.
- The CAP Theorem is comprised of three components (hence its name) as they relate to distributed data stores :
  - **Consistency :** All reads receive the most recent write or an error.
  - **Availability :** All reads contain data, but it might not be the most recent.
  - **Partition tolerance :** The system continues to operate despite network failures (i.e.; dropped partitions, slow network connections or unavailable network connections between nodes.)

- The CAP theorem states that it is not possible to guarantee all three of the desirable properties - consistency, availability and partition tolerance at the same time in a distributed system with data replication.

Q.4 Compare structured semi-structured and unstructured Data.

Ans. :

Sr. No.	Structured data	Semi-structured data	Unstructured data
1.	It is having fixed and organized form of data.	It is combination of structured and unstructured data.	It is not predefined or organized form of data.
2.	It is schema dependent and less flexible.	It is more flexible than structured data but less flexible than unstructured data.	It is the most flexible data.
3.	Structured query languages are used to access the data present in the schema.	The tags and elements are used to access the data.	Only textual queries are possible.
4.	Storage requirement for data is less.	Storage requirements for the data is significant.	Storage requirements for the data is huge.
5.	Examples : Phone numbers, Customer Names,Social Security numbers.	Examples : Server logs, Tweets organized by hashtags, emails sorted by the inbox, sent or draft folders.	Examples : Emails and messages, Image files, Open ended survey answers.

Storage	Stored in databases	Stored in file systems or data lakes	Can be stored in databases or file systems
Query Language	SQL (Structured Query Language)	No standard query language	Query languages depend on the format (e.g., XPath for XML, JSONPath for JSON)
Data Retrieval	Easy and efficient (using queries)	More complex (requires parsing or searching)	Easier than unstructured but not as straightforward as structured
Schema	Rigid and predefined	None	Flexible; schema may evolve over time

UNIT_6 IMPS

1.what is object relational database system . Explain table inheritance?

- **Hybrid Approach**: Combines features of relational databases and object-oriented databases.
- **Complex Data Types**: Supports user-defined types, arrays, and complex structures.
- **Object-Oriented Features**: Incorporates concepts like classes, inheritance, and encapsulation.
- **Table Inheritance**: Allows tables to inherit attributes from other tables, reducing redundancy.
- **Polymorphism**: Enables flexible queries by treating child tables as instances of parent tables.

Table inheritance

in a database allows one table (child) to inherit the structure (columns) from another table (parent). It functions similarly to class inheritance in object-oriented programming.

Key Points:

1. **Parent Table**: Contains common columns shared across multiple tables.
2. **Child Table**: Inherits the columns from the parent table but can also have its own unique columns.
3. **Querying**: You can query both the child table (for all columns) or the parent table (to retrieve rows from all children).
4. **Reduces Redundancy**: Avoids duplicating shared columns in multiple tables.
5. **Hierarchical Structure**: Models real-world relationships,

Q.5 What is XML ? Give its advantages.
Ans. :

- XML stands for eXtensible Markup Language.
- This scripting language is similar to HTML. That means, this scripting language contains various tags. But these tags are not predefined tags, in-fact user can define his own tags.
- Thus HTML is designed for **representation of data** on the web page whereas the XML is designed for **transport or to store data.**

Advantages of XML

1. XML document is human readable and we can edit any XML document in simple text editors.
2. The XML document is language neutral. That means a Java program can generate an XML document and this document can be parsed by Perl.
3. Every XML document has a tree structure. Hence complex data can be arranged systematically and can be understood in simple manner.
4. XML files are independent of an operating system.

An **example** of an XML database is **BaseX**, which is a native XML database.

- **Use case**: A company stores product catalogs as XML files. Each product has varying attributes (e.g., size, color), which are nested in a hierarchical structure.
- **Benefit**: BaseX allows efficient querying of these product catalogs using **XQuery** and **XPath**, enabling fast retrieval and flexible management of the data without needing a fixed schema.

3.what are spital data . Explain geographic and geometric data?

Spatial data refers to any data that is related to or represents objects or phenomena with a specific location on the Earth's surface. It describes the geometry, location, and sometimes the properties of objects. Spatial data is often used in Geographic Information Systems (GIS), mapping, urban planning, and location-based services.

Geographic data

Ans. :

- Geographic data are spatial in nature. For example - maps and satellite images are geometric data.
- Maps not only provides the location but along with location it also provides detailed information about the location.

A Guide for Engineering Students

Applications of geographic data

1) Web based road map services which allows us to use map data for vehicle navigation.
2) Vehicle navigation systems store information about roads and services for use of drivers.

Q.10 What is geometric data ?

Ans. :

- Various geometric data constructs can be represented in a database in normalized fashion.
- Following is a list of various geometric constructions and the description on how to store the information of these geometric constructs in the database systems -
  - **Line :** The line segment is represented by co-ordinates of its endpoints.
  - **Curve :** Approximate a curve by partitioning it into a storage. Create a list of vertices in order. Represent each segment as a separate tuple that also carries with it the identifier of the curve.

1. **Active database :**

- Active databases are the databases which consists of triggers. The situation and action rules are embedded in the active databases. The active databases are able to react automatically to the situations in the database. The trigger is a technique for specifying certain types of active rules. The commercial databases such as Oracle, DB2, Microsoft SQLServer allows the use of triggers.

2. **Deductive database :**

- Deductive database is a database system that can make deductions based on rules and facts stored in the database.
- Deductive databases use the concept of logic programming for specifying the rules and the facts. Prolog is a popular programming language which is based on the concept of logic programming.

Q.3 Enlist the features of Semi-structured Data models.

Ans. :

1. The semi-structured data can not be stored in the form of rows and columns in databases.
2. The semi structured data does not obey the tabular structure of data models. But it has some structure.
3. Semi-structured data contains tags and elements which is used to group data and describe how data is stored.
4. The entities can be grouped together based on their properties.
5. The entities in the same group may or may not have the same attributes(properties).
6. As the semi-structured data does not have well defined structure, it can not be programmed easily by the traditional programming languages.

What is semi-structured data model

Flexible Structure: No strict, predefined schema; data can vary between records.
Self-Describing: Data contains tags or labels (e.g., XML, JSON) to describe its structure.
Hierarchical: Often organized in a nested, tree-like format.
Partial Schema: Combines both structured elements (fields) and unstructured data.
Used in Big Data: Common in web data, logs, emails, and social media where data formats evolve.

Features of deductive data base

Rule-Based Inference: Uses logical rules (like Prolog) to derive new information from existing data.
Declarative Queries: Supports queries in a declarative style, often with logic-based languages.
Recursive Queries: Can handle recursive relationships, enabling complex queries like hierarchical data navigation.
Knowledge Representation: Combines facts and rules to model knowledge, making it suitable for reasoning tasks.
Data and Logic Integration: Merges traditional database capabilities with logical reasoning for advanced data analysis.