Database Management System

Types of DATA
Structured data are facts concerning objects and events. The most important structured data are numeric, character, and dates. Structured data are stored in tabular form. **Unstructured** data are multimedia data such as documents, photographs, maps, images, sound, and video clips. Unstructured data are most commonly found on Web servers and Web-enabled databases.

Database
A database is a collection of data that is organised in such a way that it provides efficient retrieval of desired information. This collected data could be in any format like printed, audio, electronic and graphic. Consider an example of an address book, it is also a database that provides information of personal contacts of persons.

DBMS
A Database Management System (DBMS) is a system software that allows users to efficiently define, create and maintain databases. A DBMS makes it possible for end users to create, read, update and delete data in a database. A DBMS serves as an interface between the database and end users or application programs. It provides control access to the data and ensures that data is consistent and correct by defining rules on them.

Data Model
Data models define the logical structure of a database and determines in which manner data can be stored, organized and manipulated. The model describes the relationships between different parts of the data. Data Model provides a database user with a conceptual framework in which we specify the database requirements of the database user and the structure of the database to fulfill these requirements. The most popular example of database model is the relational model.

Relational Model: In this model, data is organised in two-dimesional tables called relations. The main highlights of this model are:

- Relations can be normalized.
- In normalized relations, values saved are atomic values.
- Each row in a relation contains a unique value.
- Each column in a relation contains values from a same domain.

Relational DBMS
It is a type of DBMS that organizes data in such a way that it appears to the user that data is stored in a series of interrelated tables. These tables are also known as relations. Relationships among tables are shown through shared data.

Database Schema
Database schema also represents the logical view of the entire database but in descriptive detail. It is a blueprint of the database by defining types of entities that are in it and relationship among them. It also specifies what fields will be present, what would be their types and what constraints are to be applied on the data. A database schema can be divided broadly into two categories −

Logical Database Schema: Logical data is data about database i.e. this data stores information about how data is managed. Hence this schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

Physical Database Schema: This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.

Difference between Model and Schema: A data model can, for example, be a relational model where the data will be organised in tables whereas the schema for this model would be the set of attributes and their corresponding domains.

Data Independence

A database system normally contains a lot of data in addition to users' data. But as a DBMS expands, it needs to change over time to satisfy the requirements of the users. If the entire data is dependent, it would become a tedious and highly complex job to update it. Hence layered architecture is used to obtain data independence. Data independence can be defined as an ability to modify a schema definition in one level without affecting definition in the next higher level. There are two levels of data independence:

Logical Data Independence: The ability to change the logical schema without changing the physical schema is called as Logical Data Independence. The change would be absorbed by the mapping between the physical and logival levels. For example, if some changes are made on table format, then it should not change the data residing on the disk.

Physical Data Independence: The ability to change the physical schema without changing the logical schema is called as Physical Data Independence. All the schemas are logical, and the actual data is stored on the disk. Changing the data storage system should not have any impact on logical schema.

Database Dictionary

Database dictionary or catalog contains all the information about the data present in database such as schema definition of each table, structure of file, their type, storage format of each data item and various constraints on the data. This stored information is known as metadata.

Database Keys

A key is a single or combination of multiple fields (attributes) within a table. Its purpose is to ensure that each record within a table can be uniquely identified. They help to identify the relationship between tables and also maintains data integrity (that is data should be correct and in well formed).

Super Key: A Super Key is a set of attributes (one or more) within a table that can uniquely identify each tuple of table. Super key is a superset of Candidate key. A single table can have multiple super keys.

Candidate Key: A Candidate Key is a minimal super key i.e. it contains a single attribute or the least combination of attributes that uniquely identifies each record in the table. Every table must have at least one candidate key.

Primary Key: A primary key is one of the candidate keys chosen to uniquely identify the entity set. Primary key is used to identify the table. Each table has exactly one primary key.

Foreign Key: A foreign key is an attribute or combination of attributes in a table which refer to primary key of another table. Foreign keys are used to define relationship between tables. It is also known as referential integrity. A table can have more than one foreign key. If there are two tables, table A with primary key X and table B. To link both tables, table B should also have same attribute X in it, then X would be called foreign key in B. The table which is being refernced (table A) is called referenced relation and corresponding attribute is called referenced attribute and the table (table B) which refers to referenced table is called referencing relation and corresponding attribute is called referencing attribute.

Prime Attribute: An attribute which is part of candidate key is known as prime attribute.

Integrity Constraints

Integrity constraints provide a way of ensuring that changes made to the database by authorized users do not result in a loss of data consistency.

Entity Integrity Constraint: 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. Integrity constraint is defined on individual relations.

Referential Integrity Constraint: 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.

Joins

Join is a method to retrieve data from two or more tables which is joined to appear as single table based on a related column between them. Joining operation is done on two tuples from different tables, if and only if a given join condition is satisfied.

Cross Join: Cross join returns cross product of tables in join. It will return a table which consists of records which combines each row from the first table with each row of the second table.

Inner Join: Inner join is based on the fact that only the matching entries (entries which are present in both the tables) in both tables should be listed.

Theta Join: Theta Join combines two tuples from different tables if they satisy the theta condition. Theta join allows all operators like >,<,>= etc.

Equi Join: When Join uses only equality (=) comparison operator, it is said to be equi join.

Natural Join: Natural Join does not use any comparison operator. Natural join can be performed only if there is at least one common attribute that exists between tables. In addition, the common attributes must have the same name and domain. It acts on those matching attributes where the values of attributes in both the tables are same. The resulting joined table contains only one column for each pair of equally-named attributes.

Outer Join: Outer Joins include all the tuples (matched and unmatched) from the participating tables in the resulting table.

Left Outer Join: This join operation returns all rows from the left table in conjunction with the matching rows from the right table. If there are no matching for the left table in the right table, it will return NULL values for attributes of right table in resulting table.

Right Outer Join: This join operation returns all rows from the right table in conjunction with the matching rows from the left table. If there are no matching for the right table in the left table, it will return NULL values for attributes of left table in resulting table.

Full Outer Join: This join combines Left Outer Join and Right Outer Join. It returns a table that will include all the tuples from both the participating tables. If there are no matching tuples present in other table, then their respectice unmatched attributes are made NULL.

SQL

SQL stands for Structured Query Language. SQL is an ANSI (American National Standards Institute) standard computer language for accessing and manipulating database systems. SQL statements are used to retrieve and update data in a database. Commands in SQL are further categorised according to thier functions as given below:

DDL: DDLstands for Data Definition Language. A data base schema is specifies by a set of definitions expressed by a special language called DDL. SQL commands like CREATE (to create table schema), ALTER(to change table schema), DROP (to delete table schema) and RENAME (to rename table schema) come under this.

DML: DML stands for Data Manipulation Language. This language enables user to access or manipulate data as organised by appropriate data model. SQL commands like SELECT (to retrieve data from table), INSERT (to insert data into table) and UPDATE (to update data in table) come under this.

DCL: DCL stands for Data Control Language. It is used to create roles, permissions and referential integrity as well it is used to control access to database by securing it. SQL commands like GRANT (to give database access privilege to a user) and REVOKE (to withdraw database access privilege from user) come under this.

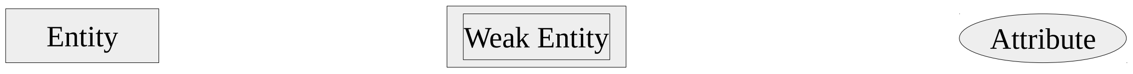
TCL: TCL stands for Transaction Control Language. It is used to manage different transactions occuring within a database. SQL commands like COMMIT (to save work done in database) and ROLLBACK (to restore database to original state since the last COMMIT command in transactions) come under this.

SDL: SDL stands for Storage Definition Language and is used to specify the internal schema. This language may specify the mapping between two schemas.

Entity-Relationship Model

ER Model is a flowchart representation of “real-world entities” such as people, objects or concepts and relationships among them. While formulating real-world scenario into the database model, the ER Model creates entity set, relationship set, general attributes and constraints.

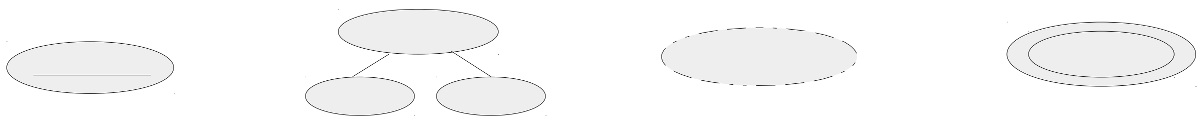
Entity: An entity can be a real-world object, either animate or inanimate, that can be easily identifiable.For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. n E-R Diagram, an entity is represented using rectangles.
Weak entity: An entity that depends on another entity. Weak entity doesn't have key attribute of their own. Double rectangle represents weak entity.



Attribute: Every entity in ER Model have some properties called attributes of that entity. Each attribute is defined by its set of values. There exists a domain or range of values that can be assigned to attributes. For example, in a school database, a student entity has various attributes like name, age, class, etc. An attribute is represented using eclipse.

Types of Attribute

Key Attribute: Key attribute represents the main characterstic of an Entity. It is used to represent Primary key. For example, roll number of a student is key attribute which uniquely identifies each student.
Composite Attribute: An attribute can also have their own attributes. These attributes are known as Composite attribute. For example, address attribute can have its own attributes like stree, city, state, etc.
Derived Attribute: Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, age of a person can be dervied from its date of birth.
Multivalued Attribute: Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email_address, etc.



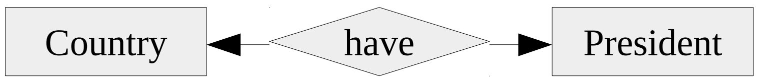
Primary Key Attribute   Composite Attribute   Derived Attribute   Multivalued Attribute

Relationship: Relationship describes the logical association among entities. For example, a student enrolls in a subject. A relationship too can have attributes.

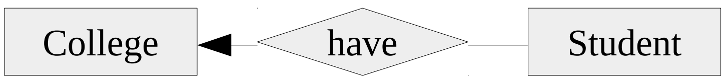
Relationship
Weak Relationship

Cardinality defines the number of entities in one entity set, which can be associated with the number of entities of other set via relationship set. Mapping cardinalities are further divided into four parts:

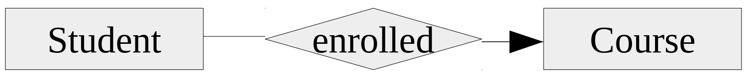
- **One to One :** One entity from entity set A can be associated with at most one entity of entity set B and vice versa. For example, One country can have only one president and vice versa.



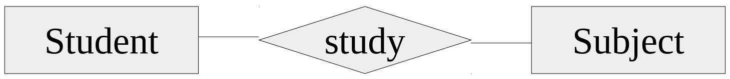
- **One to Many:** One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity. For example, a single college has many students but a single student belongs to only one college.



- M**any to One:** More than one entities from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A. For example, a student enrolls for only one course but a course can have many students.



- M**any to Many:** One entity from A can be associated with more than one entity from B and vice versa. For example, multiple students have multiple subjects.



Functional Dependencies

Functional Dependency (FD) defines a dependency/relationship among two subsets of attributes A and B of a given relation. For functional dependency $A \rightarrow B$, it specifies that the value of A determines a unique value of B in all states of a relation. In simple words, if two tuples have same values for attributes A_1 ,A_2 ,...,A_n , then those two tuples must have to have same values for attributes B_1 ,B_2 ,...,B_n .

Armstrong Axioms: If F is a set of functional dependencies then the closure of F, denoted as F , is the set of all functional dependencies logically implied by FD. Armstrong's Axioms are a set of rules, that when applied repeatedly, generates a closure of functional dependencies.

Reflexive rule: If X is a set of attributes and Y is subset of X, then X holds Y.

Augmentation rule: If $A \rightarrow B$ holds and Y is attribute set, then $AY \rightarrow BY$ also holds. That is adding attributes in dependencies, does not change the basic dependencies.

Transitivity rule: Same as transitive rule in algebra, if $A \rightarrow B$ holds and $B \rightarrow C$ holds, then $A \rightarrow C$ also holds. $A \rightarrow B$ is called as a functionally that determines B.

Trivial Functional Dependency: If a functional dependency (FD) $X \rightarrow Y$ holds, where Y is a subset of X, then it is called a trivial FD. Trivial FDs always hold.

Normalization

Normalization is the process of analyzing the given relation schemas based on their functional dependecies and primary keys by splitting relations into well structured relations that allow users to insert, delete and update tuples without introducing data inconsistencies. Without normalization many problems can occur when trying to load an integrated conceptual model into the DBMS. These problems arise from relations that are generated directly from user views are called anomalies. There are three types of anomalies: update, deletion and insertion anomalies. Main objective of normalization is to (1) minimize redundancy and (2) minimize the insertion, deletion and update anomalies.

Insertion Anomaly: An insertion anomaly is the inability to add data to the database due to absence of other data.

Deletion Anomaly: A deletion anomaly is the unintended loss of data due to deletion of other data.

Update Anomaly: An update anomaly is a data inconsistency that results from data redundancy and a partial update

Types of Normalization

First Normal Form: According to 1 st Normal Form, no two rows must contain repeating group of information i.e. each set of column must have a unique value, such that multiple columns cannot be used to fetch the same row. It simplies each attribute in a relation. The table cells must be of single value i.e. any row having a column in which more than one value is saved must be separated in multiple rows. Separate table for each set of related data is created and each table must be identified with a primary key.

Second Normal Form: In 2NF, the table is required in 1NF. According to Second Normal Form, there must not be any partial dependency of any column on primary key. It means that each column in the table that is not part of primary key should depend fully on primary key (primary key must be a composite key). To convert tables into second normal form, create separate table with the functionally dependent data and the part of the key on which it depends.

Third Normal Form: In 3NF, the table is required in 2NF. A table is said to be in 3NF if there is not any transitive functional dependency i.e. every non-prime attribute of table must be non transitively dependent on primary key. There should not be the case that a non-prime attribute is determined by another non-prime attribute. To convert tables into third normal form, create a separate table containing the attribute and the fields that are functionally dependent on it. 3NF is used to reduce duplication of data by minimizing transitive redundancy.

Boyce-Code Normal Form: BCNF is a higher version of the 3NF. When a relation has more than one candidate key, anomalies may result even though the relation is in 3NF. 3NF does not deal satisfactorily with the case of a relation with overlapping candidate keys i.e. composite candidate keys with at least one attribute in common. BCNF is based on the concept of a determinant. A determinant is any attribute (simple or composite) on which some other attribute is fully functionally dependent. A relation is in BCNF is, and only if, every determinant is a candidate key i.e. a 3NF table which does not have multiple overlapping candidate keys. The difference between 3NF and BCNF is that for a functional dependency $A \rightarrow B$, 3NF allows this dependency in a relation if B is a primary-key attribute and A is not a candidate key, Whereas BCNF insists that for this dependency to remain in a relation, A must be a candidate key.

Fourth Normal Form: In 4NF, the table is required in 3NF. 4NF doesnot contain more than one multivalued dependency i.e. no entity can have more than a single one-to-many relationship within an entity if the one-to-many attributes are independent of each other. It does not have non-trivial multivalued dependencies other than a candidate key. It applies to situations involving many-to-many relationships.

Fifth Normal Form: A table is in fifth normal form (5NF) or Project-Join Normal Form (PJNF) if it is in 4NF and it cannot have a lossless decomposition into any number of smaller tables.

Transactions

A transaction is a very small unit of a program and it may contain several low level tasks. A transaction in a database system must maintain Atomicity, Consistency, Isolation, and Durability − commonly known as ACID properties − in order to ensure accuracy, completeness, and data integrity.

Atomicity: This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none. There must be no state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.

Consistency: The database must remain in a consistent state after any transaction. No transaction should have any adverse effect on the data residing in the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.

Durability: The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data. If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.

Isolation: In a database system where more than one transaction are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system. No transaction will affect the existence of any other transaction.

Trigger: A trigger is a code that associated with insert, update or delete operations. The code is executed automatically whenever the associated query is executed on a table. Triggers can be useful to maintain integrity in database.

Stored Procedure: A stored procedure is like a function that contains a set of operations compiled together. It contains a set of operations that are commonly used in an application to do some common database tasks.

Difference between Trigger and Stored Procedure: Unlike Stored Procedures, Triggers cannot be called directly. They can only be associated with queries.

Checkpoint: A Checkpoint is like a snapshot of the DBMS state. By taking checkpoints, the DBMS can reduce the amount of work to be done during restart in the event of subsequent crashes.

Indexes: A database index is a data structure that improves the speed of data retrieval operations on a database table at the cost of additional writes and the use of more storage space to maintain the extra copy of data. Data can be stored only in one order on disk. To support faster access according to different values, faster search like binary search for different values is desired, For this purpose, indexes are created on tables. These indexes need extra space on disk, but they allow faster search according to different frequently searched values.

Clustered and Non-Clustered Indexes: Clustered indexes is the index according to which data is physically stored on disk. Therefore, only one clustered index can be created on a given database table. Non-clustered indexes don't define physical ordering of data, but logical ordering. Typically, a tree is created whose leaf point to disk records. B Tree or B+Tree are used for this purpose.

Query Optimization: The phase that identifies an efficient execution plan for evaluating a query that has the least estimated cost is referred to as query optimization.

Concurrency Control: Concurrency control is the process managing simultaneous operations against a database so that database integrity is no compromised. There are two approaches to concurrency control.