A database management system is a collection of programs that enables users to create and maintain a database: defining, constructing, manipulating, and sharing.

Database system provides: answer queries efficiently, execute updates efficiently.

Controlled data redundancy, sharing data, enforcing data integrity and improving data security.

A database state: refers to the mutable content of a database

A database schema: refers to the description of allowed database states.

**External level -> conceptual/logical level -> internal level**

Query language is used to access and retrieve data in the database

Data manipulation language is used to update the data in the database

Data definition language is used to define the database schemas

Ordered pair is two things in order: a first thing and a second thing.

Cartesian product

**Relation**: is a table with columns and rows

An **attribute** is a named column of a relation

The **domain** **of an attribute** is the set of admissible values for the attribute

A **tuple** is a **row** of a **relation**

A relational **database** is a **collection** of **relations**.

In a relation, each tuple is distinct. Order of attributes has no significance. The order of tuples has no significance. Each **relation** has a **distinct** **name** in a **database**. Each **attribute** has a **distinct** **name** **in** a **relation** (but attributes in different relations may have the same name.) the **values** of an attribute are all **from the same domain**.

Attributes are used to describe the properties of information. In the relational model, they usually refer to atomic data.

Domains are the sets of all possible values for attributes. (string, data, nat)

**Relational integrity constraints**: 1. Key constraints, entity integrity constraints, referential integrity constraints. An implicit constraint is the domain constraint: every value in a tuple must be from the domain of its attribute. Functional dependencies.

**Key constraints**: data does not occur independently from one another within relations.

A superkey SK of R is a subset of attributes of R, such that no two distinct tuples in r(R) can have the same values for SK. A superkey SK of Ris minimal if there is no other superkey SK’ < SK held on R. A minimal superkey is also known as a candidate key. A primary key PK of R is a minimal superkey of R. if a relation has only one candidate key then that would be the primary key.

Attributes: composite versus simple attribute, single-valued versus multivalued attributes, stored versus derived attributes, null values, complex (nesting of composite and multivalued) attributes.

Relationship types: when an attribute of one entity type refers to another entity type, we normally represent references as relationships not attributes.

Degree of relationship type: the number of participating entity types. We can have binary, ternary,…nary.

A role name signifies the role that a participating entity plays in each relationship instance.

Recursive relationships: same entity type can participate more than once in a relationship type in different roles.

**A ternary relationship can be represented as: A weak entity type**, with no partial key and with three identifying relationships.

Ternary or higher degree relationship can also be presented as a regular entity type by introducing an artificial or surrogate key.

**Specialization** is the process of defining a set of subclasses of an entity type.

**Generalization** is a reverse process of specialization.

Completeness constraint: total-every entity in the superclass must be a member of at least one subclass. Partial-an entity may not belong to any of the subclasses.

Desirable properties of a well-designed database schema: 1. Completeness, 2. Redundancy freeness, 3. Consistent understanding.

Update anomalies: insertion anomalies, deletion anomalies, modification anomalies.

The Armstrong’s inference rules: reflexive rule, augmentation rule, transitive rule.

In addition to data redundancy, the normalisation process also needs to consider: lossless join, dependency preservation.

**Lossless join** disallows the possibility of generating spurious tuples when a natural join operation a applied to the relations after decomposition.

**Relational algebra** provides an intermediate step: RA is a query language for relational databases; RA is not visible from the user interface. But at the core of SQL, RA is used by relational DBSMs internally for representing and optimising SQL queries.

**Selection**: choose certain tuples; Projection: choose certain attributes; **Renaming**: change the names of attributes or the relation name; **Union, intersection and difference**: set operations on two relations that have the same relational schema; **Cartesian product and join**: combine tuples from multiple relations together.

Query processing overview: 1. Users submits SQL queries to a DBMS. 2. The DBMS processes and executes them in a database.

Query->parser and translator->Relational algebra expression->optimiser->execution plan->evaluation engine.

**Query processing steps**: query parser and translator: check the syntax of SQL queries, verify that the relations do exist, transform into relational algebra expressions, query optimizer: transform into the best possible execution plan, specify the implementation of each operator in the execution plan, evaluation engine: evaluate the query execution plan, return result to the user.

A **transaction** is a number of database operations grouped together for execution as a logic unit in a DBMS: different from an execution of a program outside the DBMS in many ways. It is deemed complete only when a **commit** or **rollback** is issued.

Database applications often access a database by transactions rather than individual operations. Transactions enforce data integrity in the following situations: multiple users may modify and share data at same time; transaction, system, and media failures may happen from time to time.

**Concurrency control need**: 1. Lost update problem, 2. Dirty read problem, 3. The unrepeated read problem, 4. Phantom read problem.

**Threats to databases**: a potential breach of security that, if successful, will have a certain impact on database. 1. **Loss of confidentiality**: data should not be accessible to those who do not have legitimate access rights. 2. **Loss of integrity**: data should not be corrupted, through intentional or accidental acts. **Loss of availability**: data should remain accessible to those who have legitimate access rights.