Concepts and Properties:

Data: Known facts that can be recorded and that have implicit meaning.

Database: A collection of related data.

Database System: Database Systems work with relational databases (well-structured data) - use relational tables with well-defined values for each row and column.

DATABASE:

- Self-describing nature of a database system
- Insulation between programs and data, and data abstraction
- Support of multiple views of the data
- Sharing of data and multi-user transaction processing

Four main types of actions involve databases:

Selection, Updating, Deleting, Inserting.

DBMS (Database Management System): Manages the data, provides security, a means of accessing data from the database.

DBMS:

- Provides a way to organize data as records, tables, or objects
- Accepts data input
- Provides query languages - for searching, sorting, reporting...etc.
- Provides multi-user access to data - with security features that prevent some users from viewing/changing info
- Provides data integrity features - prevents more than one user from accessing and changing the same info
- Provides a data dictionary (metadata) - that describes the structure of the database, related files, and record information

Entity: Entity is a "thing" in the real world with an independent existence, a person, place, thing, event, or even a concept. It can also be an object with physical existence. It may be tangible or intangible.

Weak Entity: An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double

rectangle.

Owner Entity: An entity that is related to weak entities through combination of some of their attribute values.

Attribute: An attribute is a particular property that describes entity (i.e. person name company name). (Attribute Value - Data stored in the database.)

Key Attribute: Key attribute is an attribute whose values are distinct for each individual entity in the collection.

Composite Attribute: An attribute that can be divided into meaningful components.

Multivalued Attribute: Attribute that can have many values.

Derived Attribute: Derived attribute is an attribute whose value is computed from another attribute or combination of attributes.

Relationship: An association of entities, where the association includes exactly one entity from each participating entity type.

One-to-One Relationship: When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

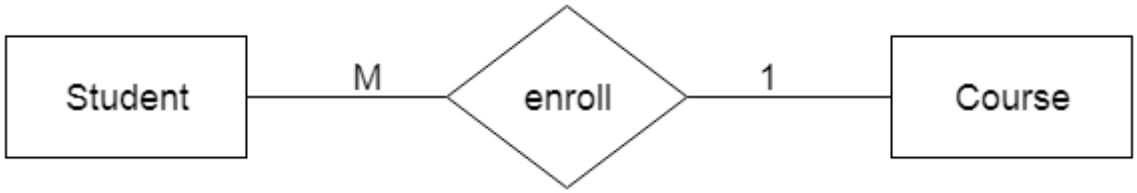
For example, A female can marry to one male, and a male can marry to one female.

One-to-Many Relationship: When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.

Many-to-one relationship: When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

For example, Student enrolls for only one course, but a course can have many students.



Many-to-many relationship: When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example, Employee can assign by many projects and project can have many employees.

Relational databases:

A relational database is a type of database that stores and provides access to data points that are related to one another. Relational databases are based on the relational model, an intuitive, straightforward way of representing data in tables. In a relational database, each row in the table is a record with a unique ID called the key. The columns of the table hold attributes of the data, and each record usually has a value for each attribute, making it easy to establish the relationships among data points.

Object-relational databases (ORD):

It is like a relational database, but with an object-oriented database model: objects, classes and inheritance are directly supported in database schemas and in the query language. In addition, just as with pure relational systems, it supports extension of the data model with custom data types and methods. Choose an object-relational database if you need more flexibility in the types of data that the database server can store and manipulate.

NoSQL databases:

NoSQL, which stands for “not only SQL,” is an approach to database design that provides flexible schemas for the storage and retrieval of data beyond the traditional table structures

found in relational databases.

Four main types of NoSQL:

Key-value Databases: Every data element in the database is stored as a key value pair consisting of an attribute name (or "key") and a value.

Document Databases: A document database stores data in JSON, BSON, or XML documents (not Word documents or Google docs, of course).

Column-Oriented Databases: While a relational database stores data in rows and reads data row by row, a column store is organized as a set of columns.

Graph Databases: A graph database focuses on the relationship between data elements. Each element is stored as a node (such as a person in a social media graph). The connections between elements are called links or relationships.

Functional dependency:

A functional dependency (FD) is a relationship between two attributes, typically between the PK and other non-key attributes within a table. For any relation R, attribute Y is functionally dependent on attribute X (usually the PK), if for every valid instance of X, that value of X uniquely determines the value of Y. This relationship is indicated by the representation as:

X ——> Y

The left side of the above FD diagram is called the determinant, and the right side is the dependent.

Conceptional database design:

There are three phases of database design namely Conceptual, Logical, Physical

Methodology for the database design:

Conceptual database design - to build the conceptual representation of the database, which has the identification of the important entities, relationships, and attributes.

Logical database design - to convert the conceptual representation to the logical structure of the database, which includes designing the relations.

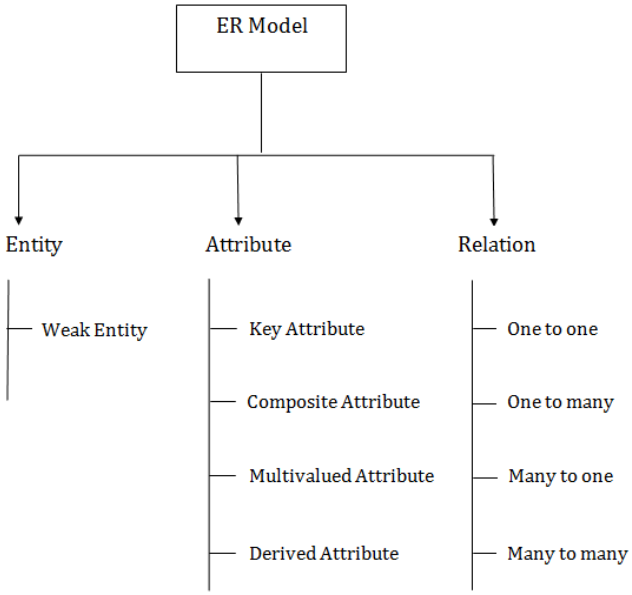
Physical database design - to decide how the logical structure is to be physically implemented (as base relations) in the target Database Management System (DBMS).

Conceptual design is the first stage in the database design process. The goal at this stage is to design a database that is independent of database software and physical details. The output of this process is a conceptual data model that describes the main data entities, attributes, relationships, and constraints of a given problem domain. The rule is that "make sure that all data needed are in the model and that all data in the model are needed." The conceptual design has four steps:

1. Data analysis and requirement.
2. Entity-relationship modeling and normalization.
3. Data model verification.
4. Distributed database design.

ER model:

- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.



ER-to-Relational mapping:

There are several processes and algorithms available to convert ER Diagrams into Relational

Schema.

ER diagrams mainly comprise of:

Entity and its attributes

Relationship, which is association among entities.

1. **Mapping Entity:**

   Create table for each entity.

   Entity's attributes should become fields of tables with their respective data types.

   Declare primary key.

2. **Mapping Relationship:**

   Create table for a relationship.

   Add the primary keys of all participating Entities as fields of table with their respective data types.

   If relationship has any attribute, add each attribute as field of table.

   Declare a primary key composing all the primary keys of participating entities.

   Declare all foreign key constraints.

3. **Mapping Weak Entity Sets**

   Create table for weak entity set.

   Add all its attributes to table as field.

   Add the primary key of identifying entity set.

   Declare all foreign key constraints.

4. **Mapping Hierarchical Entities**

   Create tables for all higher-level entities.

   Create tables for lower-level entities.

   Add primary keys of higher-level entities in the table of lower-level entities.

   In lower-level tables, add all other attributes of lower-level entities.

   Declare primary key of higher-level table and the primary key for lower-level table.

   Declare foreign key constraints.

(From https://www.tutorialspoint.com/dbms/er_model_to_relational_model.htm)

SQL:

DDL: Data Definition Language. used by DBA and dev to create DB With specific Schema.

DML: Data Manipulation Language. Used to perform data manipulations: data addition, retrieval, insertion, deletion, and modification.

DCL: Data Control Language. it's a syntax similar to a computer programming language used to control access to data stored in a database (Authorization).

Simple queries: Simple queries will display data from a few tables. An SQL query consists of three pieces, or blocks: the SELECT block, the FROM block and the WHERE block. The SELECT clause tells the database which columns of data you want it to return. The FROM clause specifies which table (or tables) you want to search. The WHERE clause allows you to search for records with certain characteristics.

Joining tables: A join table contains common fields from two or more other tables. A JOIN clause combines columns from one or more tables into a new table.

Different Types of SQL JOINs:

(INNER) JOIN: Returns records that have matching values in both tables

LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table

RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table

FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table