DBMS (Database Management system):

- A special software program that helps users create and maintain a database.

    - handles security, backups, Importing/Exporting data, Interacts with software and applications, concurrency.

What is database?

- It is a set/collection of related data put/collected together in a place.

ex. - amazon will interact with the DBMS in order to create, read, update and delete information.

The term "schema" refers to the organization of data as a blueprint of how the database is constructed (divided into database tables in the case of relational databases).

C.R.U.D - Create.Read.Update.Delete

- this is the four core processes done by the DBMS.

Types of databases:

1. **Relational database**:

    - Organize data into one or more tables

        - Each table has columns and rows.

        - A unique key identifies each row.

***uses SQL and store data in table with rows and columns.

2. **Non-Relational database:**

    -Organize data is anything but a traditional table

        - key-value stores

        - graphs

        - Documents (JSON, XML)

        - Flexible tables

***Stores data using other data structures.

***Relational Database (SQL):

- **Relational DBMS(RDBMS)**

    - Helps users create and maintain a relational database. (mySQL, oracle, postrgeSQL).

SQL (Structured Query Language):

    - Standardized language for interacting with the RDBMS.

    - Used to perform C.U.R.D operations.

    - used to define tables and structures.

    - SQL code used on one RDBMS is not always portable to another without modification.

***Non-relational Database (noSQL):

-**Non-relational DBMS (NRDBMS)**

    - helps user to create and maintain a non-relational database. (mongoDB, firebase).

Implementation specific:

    - Any non-relational database falls under this category, so there's no set language standard.

    - Most NRDBMS will implement their own language for performing C.U.R.D and administrative operations on the database.

***Database QUERY**:

    - Queries are the requests made to the database management system for specific information.

    - as the database structure become more and more complex, it becomes more difficult to get specific pieces of information we want.

    - a google search is a query.

    - A query is a set of instructions given to the RDBMS (written in SQL) that tells the RDBMS what information you want it to retrieve for you

---

Column - vertical

Row – horizontal

***always have a **Primary key**. - because it helps to differentiate, as it is always be unique for every value of row.

ex.: student id name major

1	ABC	M
2	DEF	N
3	GHI	O

here student id is the primary key - as it has a unique value for each row.

Types of keys:

1. **Surrogate key:**

- It is a primary key that has no mapping to the real world.

2. **Natural key:**

- It is a primary key that has a mapping to the real world. (just like the social security number)

3. **Foreign key:**

- It stores the primary key of the row to another database table.

- It basically helps to relate between databases.

4. **Composite key:**

- A primary key that needs two attributes or combining two data two make a primary key.

SQL:

SQL is actually a hybrid language, it's basically 4 types of languages in one

1. **Data query language (DQL)**

- used to query the database for information.

- get information that is already stored there.

2. **Data definition language (DDL)**

- used for defining database schemas.

3. **Data control language (DCL)**

- used for controlling access to the data in the database.

- user and permission management.

4. **Data manipulation language (DML)**

- used for inserting, updating and deleting data form the database.

Basic datatypes in SQL:

1. **INT** - whole numbers

2. **DECIMAL (M, N)** - decimal numbers --- {(10,4) - 10 total digits, and 4 digits after the decimal point.}

3. **VARCHAR(l)** - String of text of length l

4. **BLOB** - Binary large object, stores large data

5. **DATE** - "YYYY-MM-DD"

6. **TIMESTAMP** - "YYYY-MM-DD HH:MM: SS" - used for recording

---

Basic commands:

1. **CREATE TABLE x ();** - creates a table.

2. **DESCRIBE x;** - show the contents in the table.

3. **DROP TABLE x;** - drops/delete's the table.

4. **ALTER TABLE x ADD y DECIMAL (M, N)** - adds a column y to the table x.

5. **ALTER TABLE x DROP COLUMN y** - drops the column y from the table x.

6. **INSERT INTO x VALUE (put the values here)** - take the input from here and adds to the table.

7. **SELECT * FROM x** - displays the values/information in the table.

8. **UPDATE x -> SET x = 'y' -> WHERE x = 'z'** - updates the value to y where its value is z.

9. **DELETE FROM x -> WHERE x = 'y';** - deletes the row where the value of x is y.

10. **SELECT z, y FROM x ORDER BY z ASC;** - displays all the items stored in z, y and in z it will rearrange it in ascending order.

11. **DELETE FROM x WHERE x.xy = 2;** - this deletes the value of xy in x that has the value as 2.

Constraints:

1**. NOT NULL** - will never let you to input a null value

2. **UNIQUE** - Will never let you have duplicate value in the same column/row.

3. **AUTO INCREMENT** - adds the number automatically

4. **GROUP BY** - help to group the category in the column given as the input.

Comparison operations:

1. **=** - equals

2. **<>** - not equal

3. **>** - greater than

4. **<** - lesser than

5. **>=** - greater than or equal

6. **<=** - less than or equal

note: * means to return/display all the items in the table

SQL FUNCITONS:

1. **COUNT(x)** - counts the number of items in x.

2. **AVG(x)** - return the average.

3. **SUM(x)** - returns the sum.

4. **MAX(x)** - returns the max value in x.

5. **MIN(x)** - returns the min value in x.

WILDCARDS:

1. **%** - can have any number of characters.

2. **_** - can have one character.

***Use LIKE in wildcards.

UNION:

1. Both the select statements should have the same number of columns.

2. Both the column should have the same data type.

JOINS:

1. JOIN - this is also called the inner join, which joins the mentioned row/column to the given value.

2. LEFT JOIN - returns all the values in the left column (all the possible values).

3. RIGHT JOIN - return all the values in the right column (all the possible values).

4. FULL OUTER JOIN- return all the possible values in all the left and right column. - not used in SQL.

NESTED QUERY:

- WHERE x IN (the logic);

ON DELETE SET NULL:

- it sets the value of the deleted value as NULL.

ON DELETE CASCADE:

- it totally removes the value from the reference table and returns the remaining value, the deleted item is from the different table.

**never use ON DELETE SET NULL instead use ON DELETE CASCADE on the values which are assigned to the primary keys, as they are the most important thing that is used to point to the other table.

While for a foreign key you can use ON DELETE SET NULL as this does not play as much a important role as the primary key plays.

TRIGGERS:

- use to trigger a message while any operation is performed.

ER Diagrams:

1. **ENTITY** - An object we want to model and store information about. (This is stored in a rectangle box, with the name inside it.

2. **ATTRIBUTES** - specific pieces of information about an entity. (this is stored in oval/rectangle with circular edges, with the name inside it.) and is connected to the entity with lines.

3. **PRIMARY KEY** - An attribute that uniquely identify an entity in the database table. (this is stored in rectangle with circular edge, with the name inside it. - BUT THE NAME HAS TO BE UNDERLINED).

4. **COMPOSITE ARRTIBUTE** - an attribute that can be broken up into sub-attributes. (ex. names -> first name, last name).

5. **MULTI-VALUED ATTRIBUTE** - an attribute that can have more than one value (this is stored in two rectangular boxes with circular edges, with the name inside it).

6. **DERIVED ATTRIBUTE** - An attribute that can be derived from the other attributes. (this is stored in a dashes rectangular box with circular edges, with the name inside it.)

7. **MULTIPLE ENTITIES** - You can define more than one entity in the diagram.

**relationships b/w the entities:

8. **RELATIONSHIPS** -defines a relationship between two entities. (this is stored in a diamond shape. with the name inside it.)

9. **TOTAL PARTICIPATION** - All members must participate in the relationship. (this is denoted by two BOLD LINES)

10. **PARTIAL PARTICIPATION** - All members may participate or may not in the relationship. (this is denoted by a single LINE).

11. **REALTIONSHIP CARDINALITY** - the number of instances of an entity from a relation that can be associated with the relation. (This is denoted by N:M).

12. **WEAK ENTITY** - an entity that cannot be uniquely identified by its attributes alone. (HAS relationship) (this is denoted by double diamond shape, with HAS written in it). (the entity is stored in a double rectangle, with the name written inside it). (and the Primary key is dash underlined to represent).

* this always has to have a total participation with the relationship.

13. **IDENTIFYING RELSTIONSHIP** - A relationship that serves to uniquely identify the weak enitiy.

DESIGNING ER DIAGRAM:

- After you have completed with the ER diagram, follow the below steps to convert the ER diagram in to a database schema:

1. **Mapping of regular entity types.** (For each regular entity type create a relation that includes all the simple attributes of the entity).

2. **Mapping of weak entity types.** (For each weak entity type create a relation that includes all simple attributes of the weak entity).

3. **Mapping of Binary 1:1 Relationship types**. (Include the 1 side's primary key as a foreign key on the N side relation).

4. **Mapping of 1: N relationship types.** (Include the 1 side's primary key as a foreign key on the N side relationship)

5. **Mapping of Binary M: N relationship types**. (Create a new relation whose primary key is a combination of both entities primary key's. Also include any relationship attributes).
