

Introduction to dataBase

Data is a raw **fact** of **information** that needs to be processed.

Information is a **processed data** that is organized, structured and presented to make it useful in **decision making**.

DataBase a collection of **related data**.

- **Database definition (MetaData)** is a set of information about the data stored by BDMS (data about data).
 - **Stored data** is the data itself.
-

DBMS (database management system) creating and maintenance of computerized databases.

Falte file is a collection of records that can be **ordered** or **unordered**.

DBMS Advantages	DBMS Disadvantages
<ul style="list-style-type: none">→ Controlling Redundancy (The data is found in one place and many users can access this data).→ Restricting Unauthorized Access (By creating the users and giving them the permissions and privileges on this database).→ Sharing data .→ Enforcing Integrity Constraints (the data apply specific business rules and data logical rules).→ Inconsistency can be avoided (when data is updated we can all share the same updates at the same time).→ Providing Backup and Recovery (the DBMS can recover the data and avoid losing it).	<ul style="list-style-type: none">→ Needs expertise to use.→ DBMS is expensive (the cost of infrastructure and networks that I may need to add to store the database itself and can support the desired number of users).→ May be incompatible with any other available DBMS (by using metadata).

Database Users

<p>The system Analyst <u>gathers the requirements from the stakeholders</u> and understands why they want the DBMS and who will use it and have the access to it, the number of users, and the budget the stakeholders could afford.</p>	<p>The database designer takes the requirements from the system analyst and then <u>designs how the data will be and how the tables look like (conceptual schema)</u>.</p>	<p>The DBA (database Administrator) install DBMS, <u>Create DB schema and populate data, Create users and authorize access to DB, Maintain DB performance</u>.</p>	<p>The application programmer <u>develops, tests and debugs the application</u>.</p>	<p>End user</p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------	------------------------

DBMS Architecture (Three Schema Architecture)

<p>External Schema They are concerned with <u>what data the user will see and how the data will be presented to the user (What the user sees)</u>.</p>	<p>Conceptual Schema They are concerned with <u>what is represented</u> (define database structures such as tables and constraints).</p>	<p>Physical Schema How the data are <u>represented in the database?</u> how the data structures are implemented (<u>where the data is allocated on the Hard disk, the spaces</u>).</p>
---------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

→ The **Schemas** are separated from each others because of the **data independencies**.

→ The Schemas are separated from each other because of the **data independencies**.

Data Models: Collection of **concepts** that describe the DB.

<p>The Conceptual Data Model <u>Provide the concepts</u> that are close to the way many users perceive data, entities, attributes and relationships.</p>	<p>The Physical Data Model <u>Describes how data is stored</u> in the computer and the access path needed to access and search for data (Physical Schema).</p>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

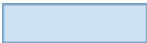
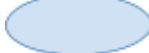





Mapping The **processes of transforming requests and results** between levels.

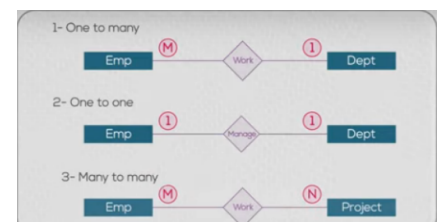
DBMS Functions:

- Text/Images/Numbers/Audio/Video
 - Spatial Data (GIS and maps data)
 - Time Series (stokes)
 - Data Mining (clustering, classification, association rules) (supermarket)
-

Centralized Database Environment		
Mainframe Environment: There is a mainframe(contains database server and application server) and a <u>group of connected monitors</u> .	Client/Server Environment (Two tier environment): <u>Each machine has the application locally installed on it.</u> (thick client)	Internet Computing Environment (Three tier architecture): have <u>multiple tiers</u> such as the database server and the application server and the (Thin client).
Problems: <ul style="list-style-type: none">→ The processing depends on <u>one server</u>→ The <u>performance is very slow</u>→ Database and application layer has <u>Single Point Of Failure</u> (If the mainframe dropped then all the other computers would drop)	Problems: <ul style="list-style-type: none">→ Database is a <u>single point of failure</u>→ <u>High cost</u> for support Advantages: <ul style="list-style-type: none"><u>Application layer isn't a single point of failure</u>	Problems: <ul style="list-style-type: none">→ Application server is a <u>single point of failure</u>→ Database is a <u>single point of failure</u> Advantages: <ul style="list-style-type: none"><u>Lower cost</u> for support and maintenance

Distributed Database Environment	
Replication (copy paste for the database). <ul style="list-style-type: none">→ Partial Replication makes a copy of a specific part of this data to another server.→ Full Replication makes a copy of all the part of this data to another server.	Fragmentation (cut paste): Distribute the database into <u>fragments</u> (horizontal, vertical, <u>hybird</u>) and put each one of them in a database server and make setup and restart and run (<u>It's important that all this fragment is connected through a specific network</u>).

Entity Relationship Modeling ERD		
Entity is a thing in the real world with an independent existence (Physical OR conceptual existence) which <u>I can describe as a set of characteristics or attributes</u> .	Attribute A particular property that describes the entity.	Relationship is a <u>connection between entity classes</u> .
Entity described by a rectangular shape. 	Attribute described by an oval shape 	Relationship described by a Rhombus (diamond) shape. 
<p>Strong Entity is an entity that <u>has a unique value (independent)</u>.</p> <p>Weak Entity an entity that <u>doesn't have a key attribute</u>, must be fully dependent on another entity. </p>	<p>Multivariable attribute: multiple values. </p> <p>Single attribute: single value.</p> <p>Composite attribute: divided into smaller subparts.</p> <p>Simple attribute: single value.</p> <p>Stored attribute.</p> <p>Derived attribute: calculated from another attribute or entity. </p> <p>→ Composite can be single or multivariable.</p> <p>→ Multivariable can be simple or composite.</p>	<p>Relationship Degree</p> <p>Binary relationship (which means it has <u>two entities</u>).</p> <p>Recursive/ Unary relationship (The relationship between the <u>entity and itself</u>).</p> <p>Ternary Relationship (A relationship includes <u>three entities</u>). </p> <p>→ The relationship between the strong entity and weak entity is called Identifying relationship and it's <u>represented in a double line</u>.</p> <p>Relationship-Participation represents the <u>minimum number of relationships</u> that can occur with these instances.</p> <p>can be represented using MUST (double line) or MAY (single line)</p> <p>Cardinality Ratio Specifies the <u>maximum number of relationships</u>.</p>



Identifiers Keys		
Primary key It must contain a <u>unique value</u> for each row of data, It <u>cannot contain Null values</u> .	Foreign key is a value that refers to a primary key in another table.	Candidate key When an entity type has more than one unique key. Those are candidate keys.

ERD Mapping To Tables Converting the conceptual design into a logical design representing the relationships we have.

- relation database is a **set of tuples** or a **set of columns**
- the intersection between column and tuple called **DOMAIN**(single value)
- **Step 1:** Mapping of **regular entity types** (create table for each entity type)
- **Step 2:** Mapping of **weak entity types** (Add foreign key that correspond to the owner entity type)
- **Step 3:** Mapping of **relationship types Binary/ Unary 1:N** (Take the Primary key of one side and put it to as a Foreign key to N-side table)
- **Step 4:** Mapping of **relationship types Binary/ Unary M:N** (Take the Primary keys of participating entities and put it as Foreign keys to a new table)
- **Step 5:** Mapping of **relationship types Binary/ Unary 1:1**(There are three cases according to the participation)
 - ❖ May-Must : Take the Primary key of "May" side and put it as a Foreign key to "Must" Table
 - ❖ May-May : Take the Primary key of any side of both and put it as a Foreign key to the table of the other side.
 - ❖ Must-Must : Merge between the two tables.
- **Step 6:** Mapping of **ternary relationship types** (Take the Primary keys of all the parents and put it as a Foreign keys to a new table. (Like M:N relationship)



Structured Query Language (SQL): SQL is a programming language that helps us to use databases.

- **DDL** (Data Definition Language)

- **DML** (Data Manipulation Language)
 - **DCL** (Data Control Language)
-

Database Schema: A group of related objects in the database

- There is **one owner** of a schema who has access to manipulate the structure of any object in the schema
 - A schema doesn't represent a person, although the schema is associated with a user that doesn't represent him.
-

Data Types: determines the **type of data** that can be stored in a database column (Alphanumeric, Numeric, Date and Time, Characters, Integer, Float data type, etc..).

Database Constraints: Restrictions on database table or object to help Maintain integrity of data (Primary Key, Not Null Constraints, Unique Key, Referential Integrity (FK), Check Constraint).

Referential Integrity (FK)

- During **insertion** I insert in the parent record first, then I insert in the child record.
 - During **deletion** I delete from the child record, then I delete from the parent record.
-

SQL - **Data Definition Language** (DDL) : They are commands responsible for the structure of a database. This helps to **Create, Edit, or Delete** tables But not to manipulate the data itself.

→ **CREATE command**

```
CREATE TABLE Students
(ID PRIMARY KEY, First_Name CHAR(50) NOT NULL, Last_Name CHAR(50) NOT
NULL, Address CHAR(50), City CHAR(50), Country CHAR(25), Birth_Date
DATE)
```

→ **ALTER command** (To Add or remove a column in a table)

```
ALTER TABLE Students ADD Postal_Code
```

→ **DROP command** (To remove the whole table)

```
ALTER TABLE Students REMOVE Postal_Code
```

→ **TRUNCATE command**

Identical to DELETE statement with no WHERE clause.
remove all data but keep the structure of the table.

- When we use TRUNCATE it deletes the data and deallocates the physical memory assigned to data so we cannot **roll-back** (Undo) the command.

```
DROP TABLE Students
```

SQL- Data Control Language DCL (GRANT, REVOKE)

→ GRANT command

- give the User the access to table Employee and to only select/ view the table but he cannot add or remove data.
- (INSERT, UPDATE, DELETE, SELECT)

→ REVOKE command

- Remove the update command from the user on department table
- Remove all commands from the users on department table

SQL- Data Manipulation Language (DML)

→ INSERT

If you know the attributes of the table and there order

```
INSERT INTO Students
VALUES(211, 'Ahmed', 'Abdallah', '20 el-haram
st', 'Giza', 'Egypt', '20/11/2001')
```

OR If you insert specific data to only specific columns

```
INSERT INTO Students(ID, First_Name, Last_Name)
VALUES(211, 'Ahmed', 'Abdallah')
```

→ UPDATE

We use WHERE to choose a specific record

```
UPDATE Students
SET Birth_Date = '20/11/2001'
WHERE ID = 211
```

→ DELETE

The DELETE command deletes the data but keeps the physical memory assigned to data until we do **COMMIT** which saves the query we have issued or roll-back is issued

```
DELETE FROM Students
WHERE ID = 211
```

→ SELECT

```
SELECT First_Name FROM Students
WHERE First_Name = 'Ahmed'
```

```
SELECT *
from Students
WHERE First_Name = 'Ahmed'
```

```
SELECT ID, First_Name, Last_Name
FROM Students
```

Comparison & Logical operators

WHERE

=: We use '=' operator when we compare exact values

```
SELECT ID, First_Name, Last_Name, GPA
FROM Students
WHERE GPA > 2
```

```
SELECT ID, First_Name, Last_Name, GPA
FROM Students
WHERE GPA BETWEEN 2.23 AND 2.4
```

//OR by using single row operator

```
SELECT ID, First_Name, Last_Name, GPA
FROM Students
WHERE GPA >=2.23 AND GPA <= 2.4
```

```
SELECT ID, First_Name, Last_Name, GPA
FROM Students
WHERE GPA IN (2.22,2.4)
```

//OR by using single row operator

```
SELECT ID, First_Name, Last_Name, GPA
FROM Students
WHERE GPA =2.22 OR GPA = 2.4
```

LIKE: We use "like" operator when we don't know the exact value we are looking for

```
SELECT *
FROM Students
WHERE Last_Name LIKE '?b*'
```

```
SELECT *
FROM Students
WHERE First_Name LIKE 'Ahm?d'
```

ALIAS: we use it when we do a calculation on a specific column and we want to display the result in a new column (just for displaying)

```
SELECT First_Name + ' '+Last_Name AS [Full Name]
FROM Students
WHERE GPA >2.3
```

ORDER BY: arrange the data (By default it arranges the values ascendingly)

```
SELECT *
FROM Students
ORDER BY First_Name ASC, Last_Name DESC
```

DISTINCT: remove duplicate value

```
SELECT DISTINCT First_Name , Last_Name
FROM Students
```

JOIN

A condition showing how DBMS can link more than one table together and it's always a relationship between the Primary key and Foreign key.

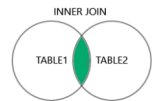
- The number of JOIN conditions is always equal to the number of tables.

Equi Join

```
SELECT First_Name , SNAME
FROM Students as Stud , Subjects as Sub
WHERE Stud.SNO = Sub.SNO;
```

INNER JOIN

```
SELECT First_Name , SNAME
FROM Students as Stud INNER JOIN Subjects as Sub
ON Stud.SNO = Sub.SNO;
```



LEFT Outer JOIN: It returns the left table and the intersection data with the right table.

```
SELECT First_Name , SNAME
FROM Students as Stud LEFT JOIN Subjects as Sub
ON Stud.SNO = Sub.SNO;
```



RIGHT Outer JOIN: It returns the full right table and the intersection with the left table.

```
SELECT First_Name , SNAME
FROM Students as Stud RIGHT JOIN Subjects as Sub
ON Stud.SNO = Sub.SNO;
```



FULL Outer JOIN

```
SELECT First_Name , SNAME
FROM Students as Stud FULL JOIN Subjects as Sub
ON Stud.SNO = Sub.SNO;
```



Sub-query: it's a query inside the main query.

- we use it when we want to get an info that isn't available directly and need to access a specific table to get such info

```
SELECT *
FROM Students
where GPA > (SELECT GPA FROM Students WHERE First_Name='Saeed' and
Last_Name='Abdallah')
```

Aggregate Functions: Functions that take multiple inputs and return only one result, aggregate functions ignore the NULL values.

→ Such as (**MAX, MIN, SUM, COUNT , AVERAGE**)

```
SELECT MAX(GPA) AS MAX, MIN(GPA) AS MIN
FROM Students;
```

GROUP BY & HAVING: Group by : we use it to divide data of a table based on the attribute you input in "**GROUP BY**".

```
SELECT SEX,AVG(GPA) AS [GPA AVERAGE]
FROM Students
GROUP BY SEX
```

→ if we want to put a condition based on aggregate function so we use "**HAVING**" clause

```
SELECT SEX,AVG(GPA) AS [GPA AVERAGE], SNO
FROM Students
GROUP BY SEX
HAVING MAX(SNO) > 1104
```

Other DB objects

→ **View:** It's a **logical table** based on a table or another view.

→ The tables on which a view is based are called base tables

→ The view is stored as a **SELECT statement** in the data dictionary (metadata)

→ **Check Option** whenever you create any DML on this view to edit data or store data in it for example, you have to confirm that the Where condition is satisfied.

```
CREATE VIEW Suppliers
AS
SELECT*
FROM suppliers
WHERE status>15
WITH CHECK OPTION; //it must validate the where condition every
time when creating any DML to insert data or to update data
```

→ **Modifying View:** **CREATE OR REPLACE VIEW** view_name

→ **Removing view:** **DROP VIEW** view_name

→ **Advantages** of using Views:

- Restrict data access
- Make complex queries easy
- Provide data independence
- Present different view of the same data

→ **Types of Views:**

Feature	Simple Views	Complex Views
Number of tables	One	One or more
Contain functions	No	Yes
Contain groups of data	No	Yes
DML operations through a view	Yes	Not always

→ **Indexes:** We use indexes to solve the following problems

- the data is not sorted by default.
- the data is scattered in the physical memory.

→ **Why indexes?**

- They are used to speed up the retrieval of records responsible for certain search conditions.
- May be defined on multiple columns
- Can be created by the users by the DBMS
- Are used and maintained by the DBMS

→ The **disadvantages** of indexes: Indexes cause overhead on DML (insert, update ,delete)

→ **Create an index when:**

- **Retrieving data** heavily from table
- Columns are used in **search conditions** and joins
- Column contain **large number of nulls**

→ **Don't create indexes when:** Table is updated **frequently**

→ **Creating of indexes:** **CREATE INDEX** index_name **ON** table_name(column_name)

→ **Removing indexes:** **DROP INDEX** index_name

Normalization: A process that takes a table through a **series of tests** (Normal Forms).

→ To Certify the goodness of a design and thus to **minimize redundancy** and **insert, update, delete anomalies, frequent null values.**

→ To Use Normalization to another method for create a database design

Functional Dependency		
A <u>constraint between two attributes</u> (columns) or two sets of columns. for every valid instance of <u>A uniquely determines the value of B.</u>		
Fully dependency: It means there's a non key attribute which is fully dependent on key , it depends completely on key	Partially Dependency: Non key attribute <u>depends on part of the key</u>	Transitive Dependency: Non key attribute <u>depends on non key attribute depends on key attribute</u>

Normalization: the process of decomposing unsatisfactory "**bad**" relations (tables contain some issues) by breaking up their attributes into **smaller** relations(tables).

Normal Form Condition		
using keys and FDs of a relation to certify whether a relation schema is in a particular normal form(1NF, 2NF, 3NF)		
1NF first normal form When it doesn't contain: <ul style="list-style-type: none">→ Multivalued Attribute→ Repeating Group→ Composite Attribute We solve this problem by place it in a new table <u>carrying the PK as a FK</u>	2NF second normal form When: <ul style="list-style-type: none">→ It applies 1NF→ Doesn't contain partial dependency we solve this by <u>taking the non-keys carrying the key they depend on and place them in a new table</u>	3NF third normal form When: <ul style="list-style-type: none">→ It applies 2NF→ Doesn't contain Transitive Dependency We solve this by <u>taking the case of transitive dependency and separate it into a new table and make the dependent attribute a PK in a new table and FK in the old table</u>

● 1NF first normal form

⊗ Multivalued Attribute

⊗ Repeating group

⊗ Composite attribute

place it in a new table carrying the
PK as a FK

subparts each in a column when
necessary

● 2NF second normal form

✓ First Normal Form

⊗ Partial Dependency

non-keys carrying the key they depend
on and place them in a new table

● 3NF third normal form

✓ Second Normal Form

⊗ Transitive Dependency

non-key attributes carrying the
non-key attribute they depend on
and place them in a new table

Database Interview Questions And Answers

1. Define Database.

is a set of information about the data stored by BDMS (data about data).

2. Define DBMS.

Database Management System. It is a collection of application programs which allow the user to organize, restore and retrieve information about data efficiently and as effectively as possible.

3. Define DDL DCL and DML.

Managing properties and attributes of a database is called Data Definition Language(DDL).

Manipulating data in a database such as inserting, updating, deleting is defined as Data Manipulation Language. (DML)

4. Define DDL DCL and DML commands

-Data Definition Language (DDL) commands are used to define the structure that holds the data. These commands are auto-committed i.e. changes done by the DDL commands on the database are saved permanently.

-Data Manipulation Language (DML) commands are used to manipulate the data of the database. These commands are not auto-committed and can be rolled back.

-Data Control Language (DCL) commands are used to control the visibility of the data in the database like revoke access permission for using data in the database.

5. Define Join and enlist its types.

A condition showing how DBMS can link more than one table together and it's always a relationship between the Primary key and Foreign key.

-EQUI JOIN

-INNER JOIN

-LEFT Outer JOIN

-RIGHT Outer JOIN

-FULL Outer JOIN

6. Define Entity, Entity type, and Entity set.

-Entity can be anything, be it a place, class or object which has an independent existence in the real world.

-Entity Type represents a set of entities that have similar attributes.

-Entity Set in the database represents a collection of entities having a particular entity type.

7. Define a Weak Entity set.

an entity that doesn't have a key attribute, must be fully dependent on another entity.

8. Enlist the advantages of DBMS.

- Controlling Redundancy (The data is found in one place and many users can access this data).
- Restricting Unauthorized Access (By creating the users and giving them the permissions and privileges on this database).
- Sharing data .
- Enforcing Integrity Constraints (the data apply specific business rules and data logical rules).
- Inconsistency can be avoided (when data is updated we can all share the same updates at the same time).
- Providing Backup and Recovery (the DBMS can recover the data and avoid losing it).

9. Enlist the advantages of SQL.

- Simple SQL queries can be used to retrieve a large amount of data from the database very quickly and efficiently.
- SQL is easy to learn and almost every DBMS supports SQL.
- It is easier to manage the database using SQL as no large amount of coding is required.

10. Enlist some commands of DDL, DML, and DCL.

Data Definition Language (DDL) commands:

- CREATE
- ALTER
- TRUNCATE
- DROP

Data Manipulation Language (DML) commands:

- INSERT
- UPDATE
- DELETE
- SELECT

Data Control Language (DCL) commands:

- GRANT
- REVOKE

11. What are the different types of Normalization?and Explain Normalization.

the process of decomposing unsatisfactory "bad" relations (tables contain some issues) by breaking up their attributes into smaller relations (tables).

- 1NF
 - 2NF
 - 3NF
-

12. What is SQL?

It stands for Structured Query Language. A programming language used for interaction with relational database management systems (RDBMS). This includes fetching, updating, inserting, and removing data from tables.

13. How many SQL statements are used?

Answer: SQL statements are basically divided into three categories, DDL, DML, and DCL.

14. What is an index?

We use indexes to solve the following problems

- the data is not sorted by default.
 - the data is scattered in the physical memory.
-

15. What types of SQL operators do you know?

- Arithmetic (+, -, *, /, etc.)
 - Comparison (>, <, =, >=, etc.)
 - Compound (+, -, *, /, etc.)
 - Logical (AND, OR, NOT, BETWEEN, etc.)
 - String (% , _ , + , ^ , etc.)
 - Set (UNION, UNION ALL, INTERSECT, and MINUS (or EXCEPT))
-

16. What is the DISTINCT statement and how do you use it?

This statement is used with the SELECT statement to filter out duplicates and return only unique values from a column of a table. The syntax is:

```
SELECT DISTINCT col_1  
FROM table_name;  
POWERED BY DATACAMP WORKSPAC
```

17. Explain the Primary Key and Foreign and Candidate and Composite Key and Unique key.

- Primary key It must contain a unique value for each row of data, It cannot contain Null values.
 - Foreign key is a value that refers to a primary key in another table.
 - Candidate key When an entity type has more than one unique key. Those are candidate keys.
 - Composite Key is a form of the candidate key where a set of columns will uniquely identify every row in the table.
 - Unique key is the same as the primary key whose every row data is uniquely identified with a difference of null value i.e. Unique key allows one value as a NULL value.
-

18. Differentiate between 'DELETE', 'TRUNCATE' and 'DROP' commands.

- After the execution of 'DELETE' operation, COMMIT and ROLLBACK statements can be performed to retrieve the lost data.

- After the execution of 'TRUNCATE' operation, COMMIT, and ROLLBACK statements cannot be performed to retrieve the lost data.
 - 'DROP' command is used to drop the table or key like the primary key/foreign key.
-

19. How to select all columns from a table?

Using the asterisk * with the SELECT statement. The syntax is: SELECT * FROM table_name;.

```
SELECT DISTINCT col_1  
FROM table_name;  
POWERED BY DATACAMP WORKSPAC
```