DBMS - Notes

Database is collection of related data

DBMS is collection of operations which user can use to manipulate data

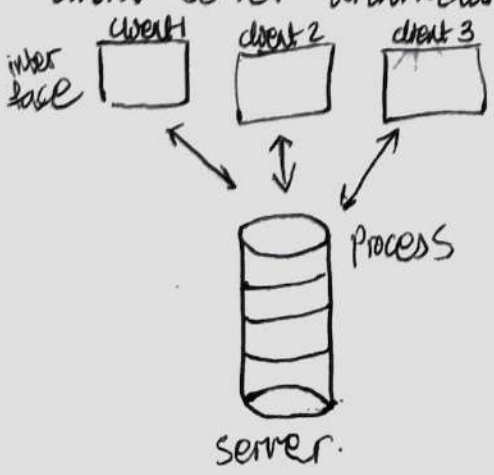
→ comparision between File system & DBMS

Filesystem	DBMS
• Searching is very slow and memory utilization is inefficient	• Searching is fast and memory utilization is efficient
• Attributes / metadata required to access data	• Data accessed through remote servers.
• Concurrency of data is not possible	• Concurrency is possible.
• No security protocol	• Security is Role based.
• Data redundancy	• No data redundancy.

DBMS - Architecture

2-tier

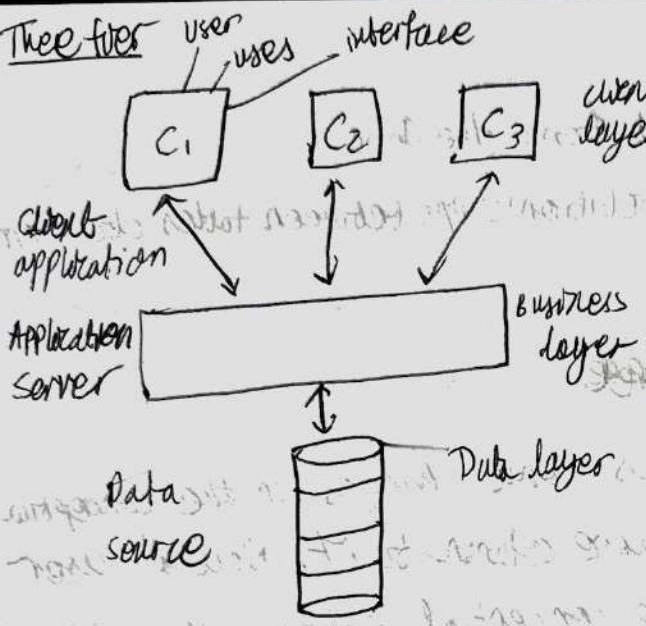
• Client server architecture.



Advantages:
- Maintainance is very easy

Disadvantages:
• Scalability is an issue
• Security - client interacts directly with the server which can cause problems

Three tiers



Advantages:

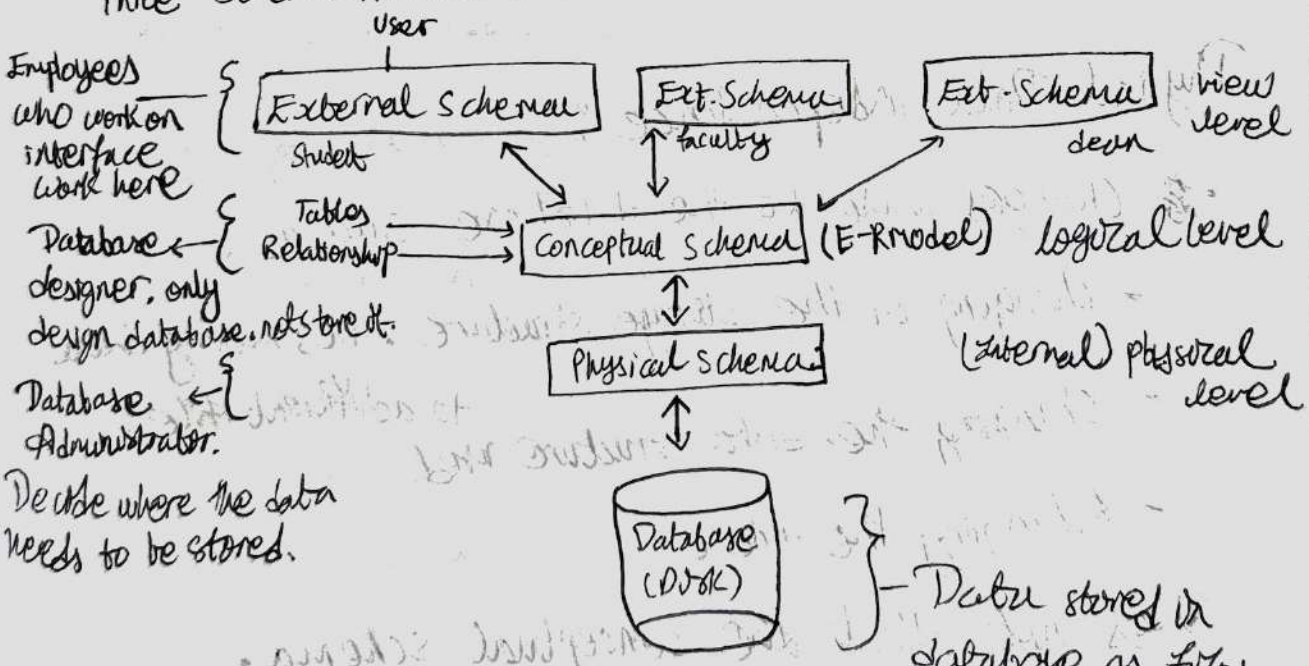
- Very easy to scale & it does not cause the server to bottleneck
- It is also very secure.

Disadvantages:

- Costly to maintain
- complex system

Schema → It is the logical Representation of the database.

Three Schema Architecture:



Data independence

- Making the user independent from the data.
- Hiding how data is stored, relationships between tables etc from user.

Logical data independence:

Lets say a user 1, makes some changes in the conceptual schema and adds one more column to it. Now if user 2 were to look at the conceptual schema, they would only find the original table which was present. Therefore, The data/table did not change for every one except user 1 at the view level.

Physical Data independence:

- Changes made to the database such as:
  - changing of the storage structure. i.e; moving data to a different file.
  - changing the data structure used.
  - changing the index.

Does not affect the conceptual schema.

Summary: changes in conceptual schema does not affect view level

changes in physical schema does not affect conceptual schema.

Integrity constraints

- Integrity constraints are rules defined on a database to maintain data accuracy, consistency & reliability.
- They assure that data stored in a database follows predefined rules and conditions.

Domain constraints: They define the set of valid values that a specific attribute in a table can hold.

eg. Data type, Range constrains (age ≠ negative), length constrains (name cannot exceed 50 char), regular expression.

Integrity:

Not null: It enforces a column to not accept null values

Foreign key: It is used to enforce referential integrity.

Set n tables in a relational database:

- It can contain null values
- Can be defined by create table or alter table statements

Characteristics:

- one or more per table
- Can be null
- one or more columns corresponding to the columns.

Database User :

- Sophisticated user : form request in a database query language
- Specialized user : write specialized database applications that do not fit into the traditional data processing framework
- Naive user : invoke one of the permanent application programs that have been written previously

Database Administrator

- Coordinates all the activities of the database system
- Has a good understanding of the enterprise information resources and needs
- Database Administrator's responsibilities include :
  - Schema definition
  - Storage structure & access method definition
  - Schema & physical organization modification
  - Granting user authority to access the database
  - Specifying integrity constraints
  - Acting as liaison with users
  - Monitoring performance & responding to changes in requirements

Transaction is a collection of operations that performs a single logical function in a database application

- Transaction management component ensures that the database remains in a consistent / correct state despite system failure.
- Concurrency - control manager controls the interaction among concurrent transactions, to ensure the consistency of database.
- A storage manager is a program module that provides the interface between low-level data stored in the database & the application programs & queries submitted to the system.
  - interaction with file manager
  - efficient storage, retrieving & updating of data.

Types of Databases :

1) commercial Database : Used in business sector to handle large volume of transactions & customer data. eg. CRM system
2) Multimedia Database : stores data types such as images, audio & video files. Eg. Youtube, Adobe Experience manager.
3) Deductive Database : Uses logic programming to derive information stored in database, allowing more complex & analytical queries
4) Temporal database : keeps track of changing data over time, allowing queries, concerning- time based data
5) Geological information system (GIS) : stores, organises & analyses geographical data. useful in mapping projects. Eg. google maps.

Entity - Relationship Model (ER model)

- Entity relationship diagram which helps the user to understand complex database easily using block diagrams. It also shows relationship between different datas in a database.

⇒ Entity:
smallest piece of information which can be stored in the database, which is also ~~stored~~ differentiable from other entity.

Types of entity:
(i) Tangible: Physically exists IRL, eg. docker
(ii) Intangible: Exists logically. eg. Account

⇒ Entity set: collection of same types of entities that share same properties or attributes.

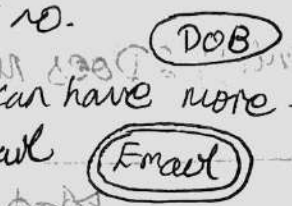
Entity → Representation

Attributes: The unit that defines the properties & characteristics of entities

Types of attributes:

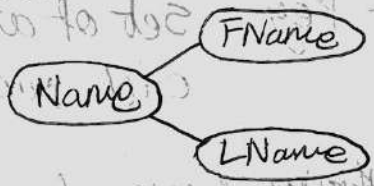
Single valued : Attributes having single value at any instance.
eg DOB, Aadhar no.

Multi valued : Attributes which can have more than one value
eg. Phone no. email



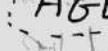
Simple : Attributes which cannot be further divided into sub-parts.

Composite : Attributes which can be divided into sub-parts
eg. Name



Stored : Attributes whose values are ~~permanently~~ Permanantly stored in db.

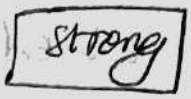
Derived : values of these Attributes can be derived from another attributes. Eg. Age can be derived from DOB



Relationship : Association between two or more entities of same or different entity set.

Mapping cardinalities:

(i) one to one (1:1)
(ii) one to many (1:M)
(iii) Many to one (M:1)
(iv) Many to Many (M:M)
or
(M:N)

Strong entity: Has a primary key 

weak entity: Does not have a primary key 

Keys

Super key: Set of attributes using which we can identify each tuple uniquely is called super key
eg Roll no, Age / Roll no, Name

Candidate key: Minimal set of attributes using which we can identify each tuple uniquely is called candidate key.
eg. Aadhar card, id, roll no etc. we choose P.key from it.

Primary key: It is a unique identifier by which you can access a particular record. It should not contain null values. There should be exactly one primary key per table.

Foreign key: It acts as a reference to the primary key of same or other table to represent relationship.

Secondary key: It can also act as a primary key to speed up data retrieval. It does not necessarily contains unique values.

Composite key: Key composed of more than one attribute.

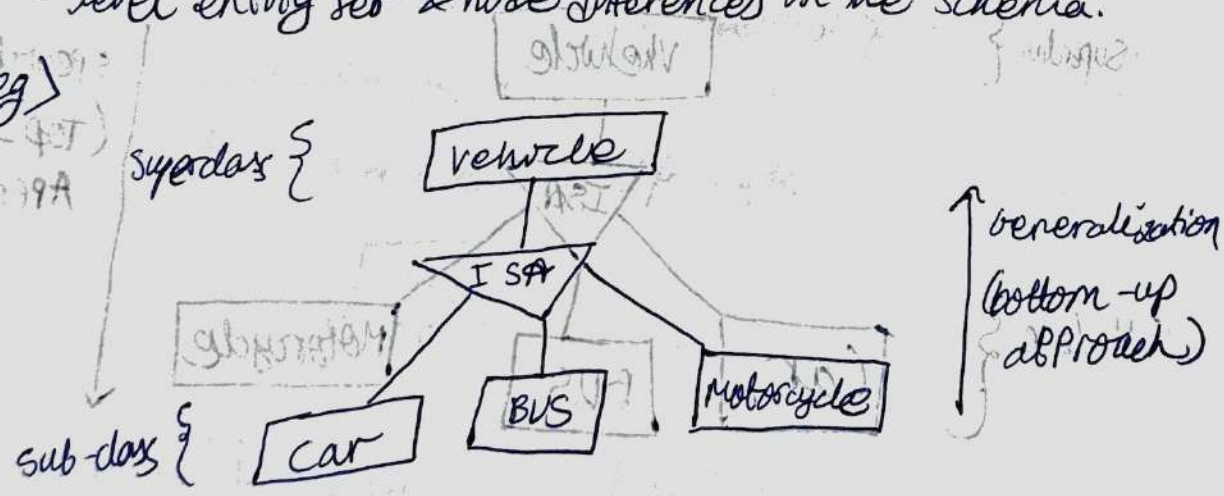
Module: 2

Enhanced ER Model

Generalization

- Generalization is the process of extracting common properties from a set of entities & create a generalized entity from it.

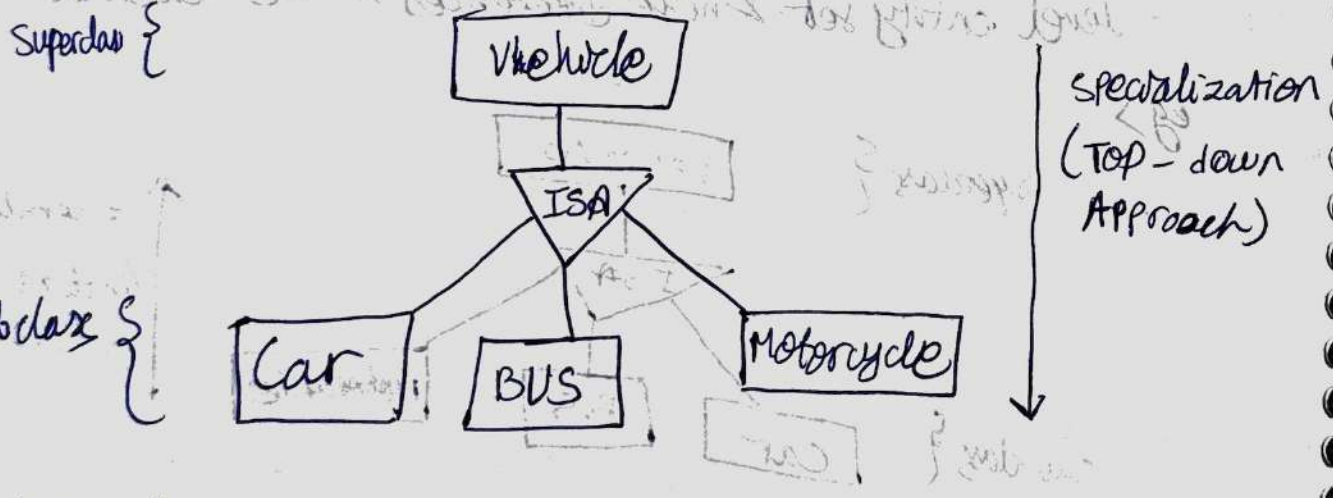
- It is a bottom up-approach (two or more sub-classes come together to form a superclass)
- It is used to emphasize the similarities among lower-level entity set & hide differences in the schema.



Specialization

- In specialization, an entity is broken down into sub-entities based on their characteristics.
- It is a top-down approach. (where superclass gets divided into multiple subclasses)
- It is used to identify the subset of an entity set that shares some distinguishing characteristics.

eg) Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set is then added.



Inheritance

- It is an important feature of both generalisation & specialization
- Attribute inheritance allows lower level entities to inherit the attributes of higher level entities.

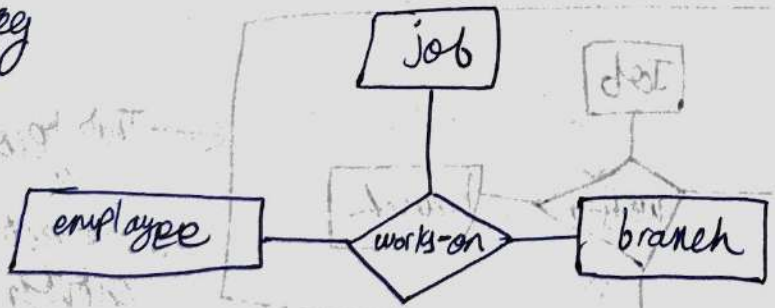
eg) Consider relations car & bus inheriting the attributes of vehicle. Thus, Car is described by attributes of super-class vehicle as well as its own attribute.

- This also extends to Participation inheritance in which relationships involving higher-level entity sets are also inherited by lower-level entity sets.

Aggregation

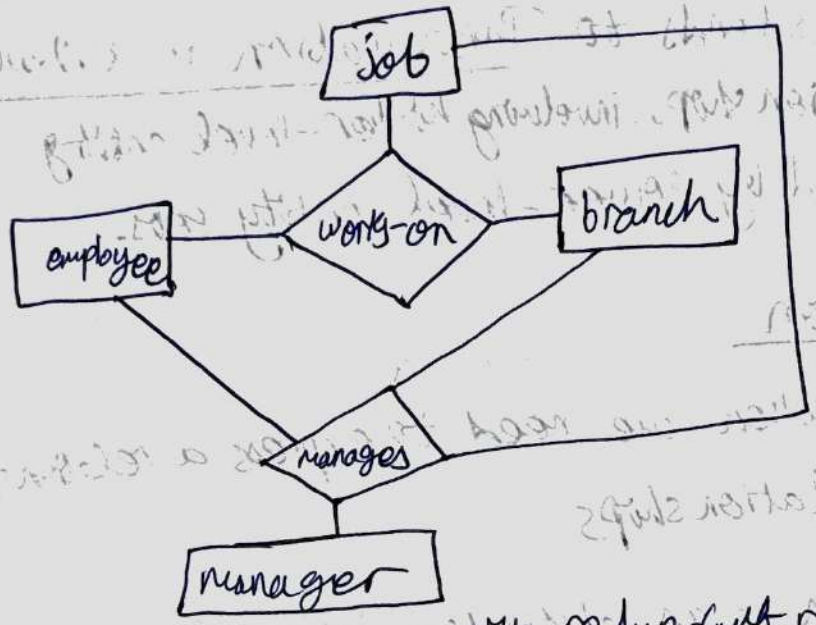
- It is used when we need to express a relationship among relationships
- Aggregation is an abstraction through which relationships are treated as higher level entities.
- Aggregation is a process when a relationship between two entities is considered as a single entity and again this single entity has a relationship with another entity.

eg



↔ suppose we assign a manager for jobs performed by an employee at a branch.

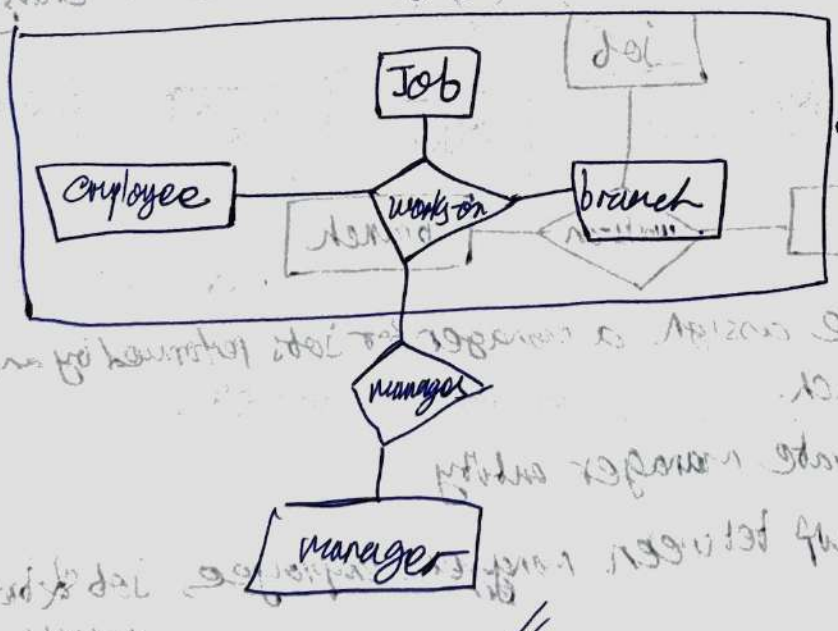
- Need seperate manager entity.
- Relationship between manager, employee, job & branch entity.



E-R diagram with redundant relationship

* Relationship sets works-on & manages are overlapping (redundant) information.

Now, with aggregation



← This becomes a seperate entity now

Relational Algebra

"operators"

Basic operators:

- → Projection (π)
- → selection (σ)
- → cross product (×)
- → Union (∪)
- → Rename (ρ)
- → set difference (−)

Derived operators:

- → Join (⋈)
- → intersect (∩)
  $(X \cap Y) = X - (X - Y)$
- → Division (/, ÷)

Projection (π)

Roll no.	Name	Age
1	A	20
2	B	21
3	A	19

Query: Retrieve the roll no from table (student)

$\Pi_{name}(student)$ → (Rename name)

name
A
B

(No duplicate values, only distinct values)

$\Pi_{roll\,no,\,Name}(student)$

↳

Roll no.	Name
1	A
2	B
3	A

Selection (σ)

Roll no	Name	Age
1	A	20
2	B	21
3	A	19

Query: Retrieve the name of the student whose Roll no = '2'.

Ans ⇒

$\Pi_{name}(\sigma_{Roll\ no = '2'}(Student))$

Roll no.	Name	Age
2	B	21

→ B

Projection = columns
Selection = Rows (tuples)

Cross product (X)

$(R_1 \times R_2)$

A	B	C
1	2	3
2	1	4

x tuples, m columns

C	D	E
3	4	5
2	1	2

n columns, y tuples

A	B	C	C	D	E
1	2	3	3	4	5
1	2	3	2	1	2
2	1	4	3	4	5
2	1	4	2	1	2

T. not columns ⇒ m+n ⇒ 3+6 ⇒ 6

Why did we take 2 C's?
→ At least one column should be common for cross product

T. not of tuples ⇒ x×y ⇒ 2×2 = 4

Set difference (−)

$(A - B)$ = A but not B

$= A \cap B'$

$A - B \neq B - A$

$S_1 = 1, 2, 3$

$S_2 = 3, 4$

$S_1 - S_2 = 1, 2$

$S_2 - S_1 = 4$

$\therefore (S_1 - S_2) \neq (S_2 - S_1)$

(student) − (Employee)

Roll no	Name
1	A
2	B
3	C

EmpNO	Name
7	E
1	A

1) No. of columns must be same

2) Domains must be same

Roll no	Name
2	B
3	C

Retrieve name of a person who is a student & who is not emp:

Query : $(\Pi_{name}(Student) - \Pi_{name}(Employee))$

name
B
C

Union (U)

$S_1 = 1, 2, 3 \quad S_2 = 4, 5$

$S_1 \cup S_2 = 1, 2, 3, 4, 5$

Roll no	name
1	A
2	B
3	C

Empno	name
7	E
1	A

name
A
B
C
E

1) No. of colums must be same

2) Domains of every column must be same

Student ∪ Employee

Roll no.	Name
1	A
2	B
3	C
7	E

← $(\Pi_{name}(Student) \cup \Pi_{name}(Employee))$

Division Method (÷)

Alter → update
~~delete~~ delete → drop

Sid	Cid
S_1	C_1
S_2	C_1
S_1	C_2
S_3	C_2

'Enrolled'

Cid
C_1
C_2

'course'

course 'c'

Query: Retrieve sid of students who enrolled in every course
all

all / every → division method.

A(x, y) / B(y) = it results x values for that there should be tuple ⟨x, y⟩ for every y value of relation B.

$\neq$ (Sid, Cid) / C(Cid)

$\approx S_1$

$\Pi_{Sid}(Enrolle) - ((\Pi_{Sid}(Enrolled)) \times \Pi_{Cid}(course)] - (Enrolled)$

~~S_1 C_1~~
~~S_1 C_2~~
~~S_2 C_1~~
S_2 C_2
S_3 C_1
~~S_3 C_2~~

S_1 C_1
S_2 C_1
S_1 C_2
S_3 C_2

$=$ S_2 C_2
S_3 C_1

S_1 S_2
~~S_2~~ $-$ ~~S_3~~ $\Rightarrow$ S_1
~~S_3~~

SQL Queries

Data Definition language (DDL)

- → Create
- → Alter
- → Drop
- → Trun cate
- → Rename

Data manipulation language (DML)

- → select
- → insert
- → update
- → Delete

Constraints

- → Primary key
- → Foreign key
- → check
- → Unique
- → Default
- → Not null.

Create a Table :

~~Create a Table~~

```
CREATE TABLE STUDENTS (
    Student ID    INT Primary key,
    First Name    VARCHAR(50),
    Last Name     VARCHAR(50),
    Email         VARCHAR(100)
);
```

Alter Command :

→ Add columns
→ Remove columns
→ Modify ~~column~~ datatype
→ Modify datatype length
→ Add constraints
→ Remove constraints
→ Rename column / table.

ADDing :

eg.

```
ALTER TABLE STUDENTS,
    ADD PhoneNumber VARCHAR(15);
```

Drop / Delete a column :

```
ALTER TABLE STUDENTS
DROP COLUMN PhoneNumber;
```

Modifying an existing column:

```
ALTER TABLE STUDENTS
MODIFY COLUMN PhoneNumber(20);
```

Renaming a column:

```
ALTER TABLE STUDENTS,
RENAME COLUMN PhoneNumber TO ContactNumber;
```

Renaming a Table:

```
ALTER TABLE STUDENTS,
RENAME TO Staff;
```

Drop a table:

```
DROP TABLE STUDENTS;
```

ADDING A foreign key:

```
CREATE TABLE orders(
    OrderID     INT Primary key;
    CustomerID  INT NOT NULL,
    FOREIGN KEY(CustomerID) REFRENCES
        Customer(CustomerID);
```

Inserting values:

```
INSERT INTO STUDENTS (StudentID, FirstName, LastName)
VALUES
(1, 'Ojas', 'chatur'),
(2, 'Boka', 'choda');
```

Deleting values:

```
DELETE FROM STUDENTS
WHERE studentID= 1;
```

Show table values

```
SELECT * FROM STUDENTS
WHERE studentID= 1;
```

Difference between Alter & Update

Alter	Update
• DDL	• DML
• ADD, Remove, modify data length, datatype • etc	•

Update:

```
UPDATE STUDENTS
SET FirstName = 'soma', LastName = 'bokaa'
Where STUDENTID = 2;
```

Delete	Drop	Truncate
DML	DDL	DDL
DELETE FROM STUDENTS	DROP TABLE STUDENT;	TRUNCATE STUDENT;
ID | Name		ID | NAME
DELETE FROM STUDENT WHERE STUDENTID = 1;		DELETES all the rows in the table.
• Rollback possible • slower		• No roll back option • faster

Practice questions on SQL Queries

EMP

E_id	E_name	Dept	Salary
1	Ram	HR	10000
2	Amrit	MRKT	20000
3	Ravi	HR	30000
4	Nitin	MRKT	40000
5	Varun	IT	50000

Maximum Salary

~~SELECT * FROM EMP~~

```
SELECT MAX(SALARY) FROM EMP;
```

-> 50,000.

MAX. Salary + Name

outer query →

```
SELECT * E_name FROM EMP
    WHERE SALARY = (SELECT MAX(Salary) from Emp);
```

↑ inner query

2nd highest salary

```
SELECT * (MAX(SALARY) FROM EMP
    WHERE SALARY <> (SELECT MAX(SALARY) FROM EMP);
```

=> 40,000

Employee name → 2nd highest

```
SELECT * E_name FROM EMP
    WHERE SALARY = (SELECT * MAX(SALARY) FROM
        EMP WHERE SALARY <> (SELECT * MAX(SALARY)
        FROM EMP);
```

Display all dept names & no. of emp working in it

```
SELECT * DEPT, FROM EMP GROUP BY DEPT
```

```
SELECT * DEPT, COUNT(DEPT) FROM EMP GROUP BY DPT.
```

=>

MR	2
MRKT	2
IT	2

Dept names less than 2 employees

```
SELECT * DEPT FROM EMP GROUP BY DEPT
    HAVING COUNT(DEPT/*) < 2);
```

↓ now add name of employee who works in this dept

```
SELECT E NAME FROM EMP WHERE DEPT IN ( );
```

Dept wise highest salary & name of employee

SELECT E.NAME FROM EMP WHERE SALARY
IN (SELECT MAX(SALARY) FROM Emp Group By deptt);

Details of employee whose address is Delhi, Chandigadh or pune

SELECT * FROM EMP WHERE ADDRESS IN('DELHI',
'CHANDIGADH', 'PUNE');

~~Name of emp~~

Details of employee working on Atleast one project

SELECT * FROM Emp WHERE EID Exists (SELECT
EID FROM
PROJECT WHERE
EMP. EID = PROJECT. EID)

SELECT MAX SALARY FROM EMP;
" MIN " " " ;
SELECT COUNT (*) FROM EMP
= 6
" AVG
" SUM FROM EMP