M T W T F S

Page No.: YOUVA

Date:

# DBMS Sem Preparation Notes

## Unit-I

## Relational Database

Topics to cover:

- *) Purpose DBSystem
- *) Views of data
- *) Data models
- *) DB System Architecture
- *) Introduction to RDB
- *) Relational Model
- *) Keys
- *) Relational Algebra
- *) SQL fundamentals
- *) Advanced SQL features
- *) Embedded SQL - Dynamic SQL

## Purpose of Database

What is Database?

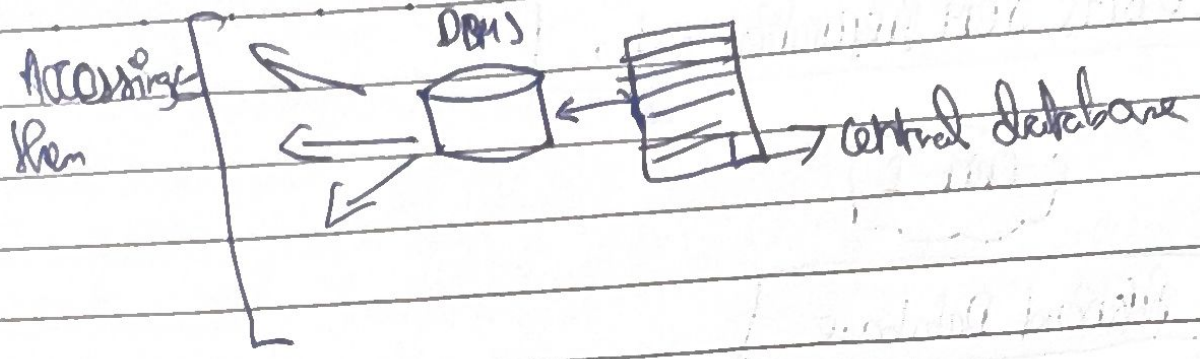
*) Collection of Data in a Sources, in a particular or desired format

*) Managing of Database or Data are called data base management System

*) Structure: rows x columns

*) Unstructured: Image files. mp3, mp4, MKV files

*) Tuple: Collection of related data




## Management System:

*) They organize, create, update, delete, sort and does those thing with database

## Application:

*) Banking
*) Student Administrations
*) University
*) Manufacturing and Selling

## Why Databases were created

*) To manage multiple data, informations easily but also well organized way, database were created, and management is used

## Problems of old file processing System:

V Data Redundancy & Inconsistency:

*) When same data are stored in different files
Ex: Avanish in music class and make, his data like name, address are stored in both
*) Inconsistency: When Avanish changes address and stored/updated only in music class file, then it leads inconsistency

M T W T F S S

Page No.:

Date:

YOUVA

2) Difficulty in Accessing Data:

*) Accessing particular data were hard
Ex: Ashis how many classes did Asmith attended
*) The program mostly print all data that are not asked for, or show error or asks for new set of program

3) Problem

*) Data is scattered in different files
*) files are in different formats
*) It becomes very hard, updated certain important data some times

4) Integrity problem

*) Integrity means keeping data correct and following rules
*) To follow the rule, programmers create rules inside many different programs
*) If new rule Added programs must be changed

5) Atomicity problems:

Ex: If A transfers money to B
500 from A to B
The money is reducted from A but not added to B due to external factor then these are called atomicity
*) All or nothing concept:

6) Concurrent - Access anomalies:

* Many people Accessing same data at a time
* Both updates the same time, the actual update will not be Reflected

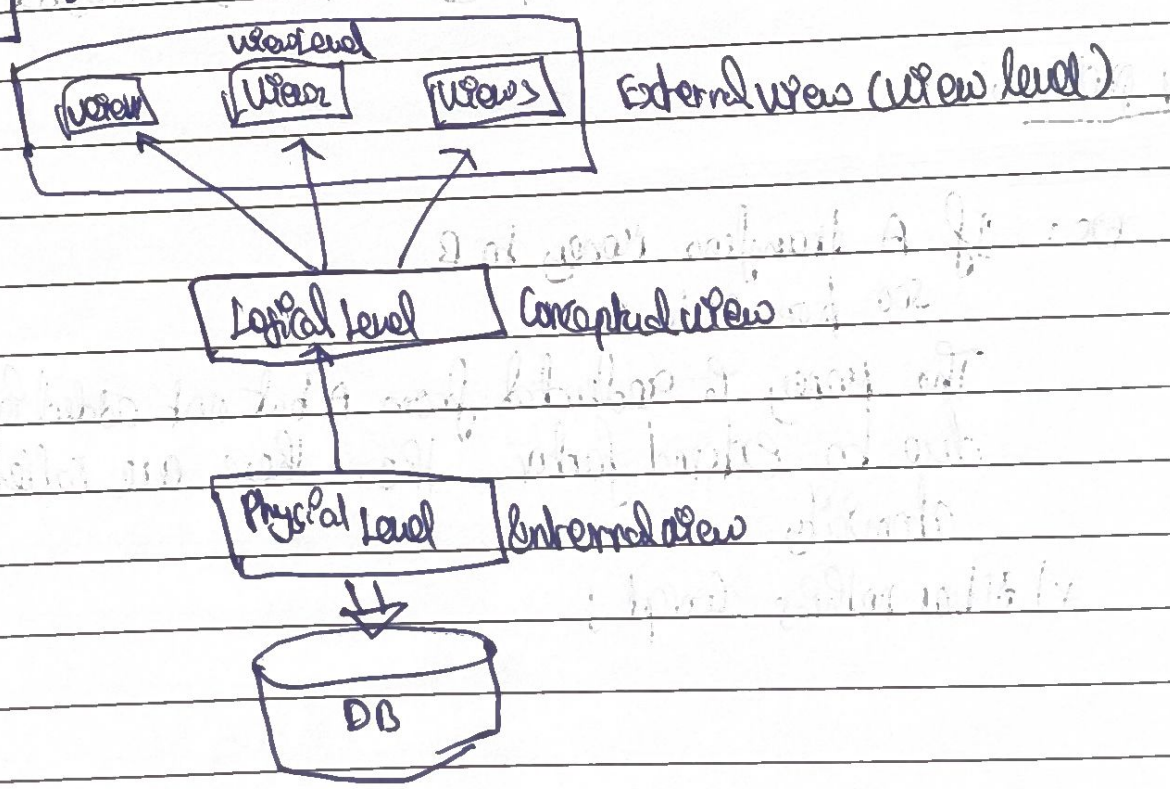
Ex: A has 10,000₹
C1 takes 500₹
C2 takes 100₹
updation right be: 9900₹ 9500₹
but actual is 9400₹

7) Security problems:

* The data might be easy for someone external to access, which is bad.

To avoid all these, DBMS is used.

Views of Data:



A DBMS shows data differently to different people:
Ex: student views (can see student can view their dashboard alone)
class incharge view (can see all students details of class)
Administration (can see all students, teachers data of college)

## Data Abstraction:

*) Hiding of certain, unnecessary data from certain ~~viewers~~ viewers, thus only those allowed will be viewed
So there are 3 main views:

*) Internal view
*) Conceptual view
*) External view

## Internal view or physical level:

*) It shows how and where data is physically stored inside the computer
*) It deals with files, blocks, indexes, and storage
*) Only database Engineers worry about this (the lowest lev)
*) Normal people never see this.
Ex: we only see YT videos but never care where it is stored

## Conceptual view or logical level:

*) Explains what, how data is stored in database, and how it is connected
*) They are just conceptual maps
*) used when certain organizing of data is needed or mapping before constructing.
ex: Table : Student has Rollno, class, dept, Name

M T W T F S S

Page No.: YOUVA

Date:

## External view (view level)

*) This is the topmost level
*) Simplest level
*) Just shows the front end of the data according to the people viewing them
Ex: If you buy something and bill is generated you can only see your bill, but Admin can see all bills.

## Data Independence:

changing something in data without changing the program:

a) Physical Independence: changes physical storage without affecting logical table and concept
b) Logical Independence: changing tables or structures or relations without affecting users views

**Advantages**
*) Easy for users
*) Simple and fast

**Disadvantages:**
*) Hard for developers
*) More load, tricky

---

## Data models

*) It is blue print for a database
It provides tools to describe
a) Data
b) Data relationships
c) Data semantics
d) Consistency constraints

M T W T F S S
Page No.:
Date:
YOUVA

## Types of Data Models

| |
|---|
| (1) Relational model |
| (2) Entity-Relationship model |
| (3) Object based Model |
| (4) Semi-Structured Data model |
| (5) Historical Models |

(1) RM: uses table to store Data, tables are called relations, Columns are attributes and rows are their records, Every column has unique key, It is the most popular model used

(2) E-R Models: It used Entities which are Objects like Students table, And they uses those entities and find their relation with other and the structure is created
Ex: (Students = entity, Course = entity, enrolls - relationship)

(3) OBM: It is Similar to Object oriented programming, it has Objects, Methods, Encapsulation, Object identity, object-relational Model (Combine this model with relational Tables
Ex: Students Object Name + function like Course details

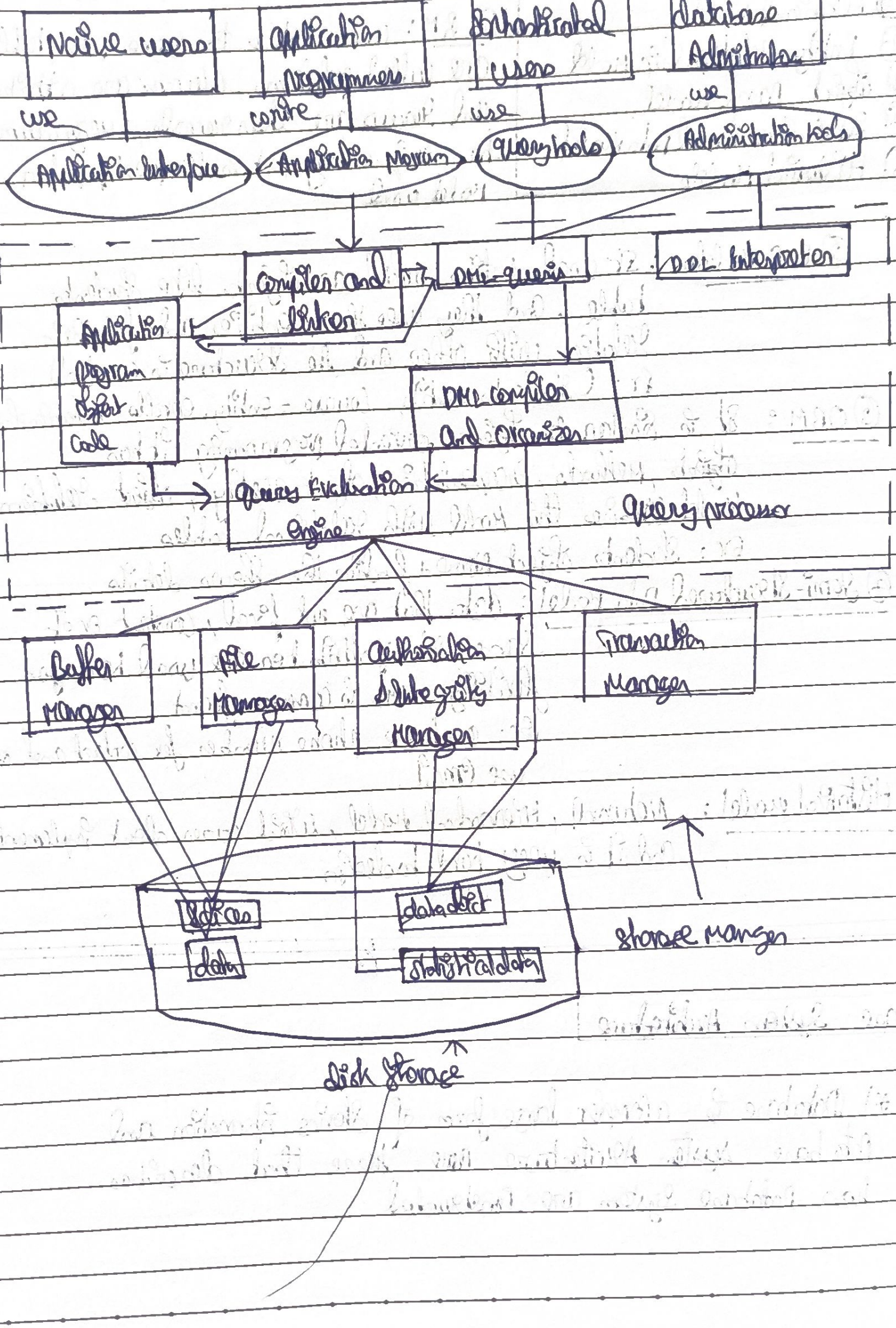
(4) Semi-Structured Data Model: data that are not fixed, constant or of same set of attributes are used here for flexibility, XML is common format
Ex: one have phone number for contact and other use Email

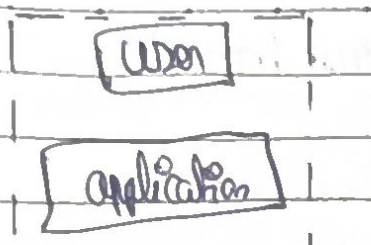
(5) Historical model: Network, Hierarchical model, which causes about implementation and it is very hard to design

## Database System Architecture

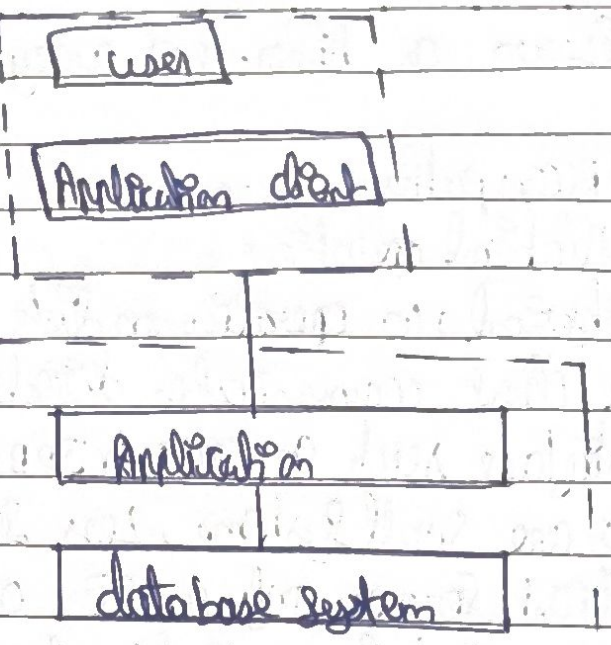
#1 Database is a complex large form of storing information and Data base system Architecture are those that describes how Database System are constructed

## Diagrams:


Naive users
Application programmers
Sophisticated users
Database Administrators
use
write
use
use
Application interface
Application Program
query tools
Administration tools
compiler and linker
DML queries
DDL interpreter
Application program object code
DML compiler and organizer
query evaluation engine
query processor
Buffer manager
file manager
authorization & integrity manager
transaction manager
indices
data
data dict
statistical data
storage manager
disk storage




a) Two tier Architecture




b) Three tier architecture

Database System Architecture:

a) They depends largely upon the computer system environment in which it operates Based on this

a) Centralized System: Entire db stored on single system
a) client-server systems: A server performs on operations for multiple client machines
a) parallel database system: use multiple processors at same time to improve performance
a) Distributed database : DB is stored on multiple geographically seperated location

a) Database System is divided into multiple components so that different responsibilities are handled efficiently and independently. Broadly the Main components are:

1) Storage manager : Manage how data is stored
2) query processor : Converts queries to understandable machine instruction
3) User : Different kinds of users

o| When user issues a higher-level query, the system must translate it into:
  o| Access paths
  o| Relational operations
  o| Physical I/O operation on disk
o| Two tier: The client communicates directly to server, they use interface such as ODBC or JDBC to send SQL queries they are small systems, less secure and hard to scale
o| Three tier: client server only as UI or front end, the application server contains application logic, they are high scalability and great performance, centralized business logic, used for large and web-based applications, easy to update

## Introduction to Relational Databases

*| The Relational model represents data and relationships using Tables
*| Each table has rows and columns
(tuples) (attributes)
*| It describes data at logical and view levels, hiding physical storage details.

## Key:

### Super key:

*| A Super key is one or more attributes that can uniquely identify a tuple in a relation
*| Example (Instructor(ID, name, salary)):
(ID, name)
↓ ↑ can be duplicated
Super key

## Candidate key:

*| A minimal superkey (no attributes can be removed)
*| Example {ID} and {name, dept name} are candidate keys
*| {ID, name} are not, but {ID} alone is a candidate key

## Primary key:

*| In which no duplicates are allowed just like candidate key
*| Should never change
*| primary key attributes are underlined

## Foreign key:

*| An Attribute in relation R1 that is primary key of relation R2
*| R1 → Referencing relation
*| R2 → Referenced relation
*| Ex: instructor (ID, name, dept-name) - ID is primary key
department (dept name, instructor name) - dept-name is primary key
dept name in instructor is a foreign key referencing department.

## Relational Algebra:

*| It is a procedural query language
*| Input → Relations, output → relations
*| Use operators Unary (one relation) Ternary (more than two)
Binary (Two relation)
*| Executed immediate and the results are also relations

M T W T F S S

Page No.: YOUVA

Date:

# Fundamental operations

| Operator | purpose |
|---|---|
| project | $\pi$ |
| select | $\sigma$ |
| rename | $\rho$ |
| union | $\cup$ |
| Intersection | $\cap$ |
| Assignment | $\leftarrow$ |
| Set difference | $-$ |
| Cartesian product | $\times$ |
| Join | $\bowtie$ |
| Left outer join | $\overset{\leftarrow}{\bowtie}$ |
| right outer join | $\overset{\rightarrow}{\bowtie}$ |
| Semi join | $\ltimes$ |

**Explanation:**

**Select ($\sigma$):** Syntax: $\sigma_{<Condition>}(R)$

$\sigma_{DNo=4}(employee)$ → Employee in DNo 4

$\sigma_{Salary>3000}(Employee)$

Select rows based on condition

**Project ($\pi$):** Select only required Columns

Syntax: $\pi_{<attributes>}(R)$

$\pi_{Name, Fname, Salary}(Employee)$

**Union ($\cup$):** Rows present in R or S or both

Syntax: $\pi_{Author}(Books) \cup \pi_{Author}(Articles)$

**Set difference:** Rows in R but not in S

Syntax: $\pi_{author}(books) - \pi_{author}(Articles)$

Cartesian Product (×):

Combines each Row in R with each Row in S
Example : Books × Articles

Rename (ρ):

Rename Relation or attributes
Syntax : ρ NewName (Relation)
ρ x(E)

Intersection (∩):

Rows that appear in both (Common only)
Example: π Author(Books) ∩ π Author(Articles)

Assignment (←):

Stores result in variable
Ex: Temp ← DNO = 4 (Employee)

Natural Join (⋈):

Combines table based on common attributes: names, removes duplicates
Automatically

Emp:

| Name | EmpID | Dept |
|---|---|---|
| A | 3 | F |
| AB | 34 | S |
| ABC | 345 | S |

Dept:

| Dept | Manager |
|---|---|
| F | AC |
| S | HE |
| P | C |

Emp ⋈ Dept =

| Name | EmpID | Dept | Manager |
|---|---|---|---|
| A | 3 | F | AC |
| AB | 34 | S | HE |
| ABC | 345 | S | HE |

## Semi Join(⋉):

*) Its the left Semi Join
*) Returns only tuple of R that Match with tuple of S on common Attributes

Emp:

| Name | EmpID | Deptname |
|---|---|---|
| A | 3 | F |
| B | 2 | S |
| C | 1 | F |
| D | 0 | P |

Dept

| Deptname | Manager |
|---|---|
| S | Bob |
| S | Thomas |
| P | K |
| P | M |

Emp ⋉ Dept =

| Name | EmpID | Deptname |
|---|---|---|
| B | 2 | S |
| D | 0 | P |

## Division:

*) Used when one relation contains pairs and another contains single attributes
Returns values to values in divisor

$R \div S$

Completed

| Student | Task |
|---|---|
| F | DB1 |
| F | DB2 |
| F | C1 |
| E | C1 |
| E | DB1 |
| S | DB1 |
| S | DB2 |

Completed ÷ DB project =

| Student |
|---|
| F |
| S |

DB project

| |
|---|
| DB1 |
| DB2 |

Left outer Join):

$\bowtie$

R-S

Returns only common values from both table, with common Attributes
That's all no need for drawing table in this Notes, Just remember
it Joins only the common, also it take Null value from the S

| value | id |
|---|---|

Right outer Join ( $\bowtie$ ):

*) Just opposite to left it returns all matching to the right
and Null value to the left side of the table

R => all value = table with matching from R, then
S => Two rows those unmatched returns null values
from R

Full outer join():

$\bowtie$

*) Returns all matching rows from both table + unmatched with Null on both side

## SQL Fundamentals

o) Structured query language is used to store, manipulate and retrieve data in relational database.
o) It is used in RDBMS such as:

- o) MySQL
- o) Oracle
- o) MS Access
- o) PostgreSQL
- o) SQL Server
- o) Sybase

## Different versions

| company | Pulect |
|---|---|
| Microsoft SQL Server | T-SQL |
| Oracle | PL SQL |
| MS Access | JET SQL |

## SQL Datatype:

| | |
|---|---|
| Char(n) | fixed length |
| Varchar(n) | variable length |
| Int | - |
| Small Int | Small number |
| Numeric(e,d) | decimal with fixed points |
| Real / double precision | floating point numbers |
| float(n) | Any Value |
| date | - |
| time | - |

## SQL Commands:

### a) DDL (Data Definition language):

* Create table [tablename( )]; used to create
* Desc [desttablename;] used to show table structure
* DROP deletes table
* Alter Modifies based on condition
* Truncate deletes all that are empty

### b) DML (Data Manipulation model):

* Insert into table
* UPDATE based on condition update name set
  @1 = value
  where condition;

*) Delete selected rows
*) Select retrieves data based on conditions or overall

## DCL (Data control language)

a) Grant to Give permission to user
Grant previlege on object to user
b) Revoke block from accessing

## TCL (Transaction control language)

a) Commit, Save changes permanently
b) Savepoint, Mark a point to rollback
c) Rollback undo changes

```
Save point1;
rollback to point1;
commit;
```

## Embeded SQL | Static

*) It means writting SQL commands inside a programming language
*) The language that is used is called host language
*) Example: C, Java, COBOL, FORTRAN.

How it works?

1) Program containing EXEC SQL statements
2) A precompiler / preprocessor reads and converts SQL seperately
3) Output of preprocessor goes to host language compiler
4) Final program can communicate with database

Syntax:

Every embedded SQL statement starts with

```
EXEC SQL <SQL Statement> End EXEC
```

Fetching Data:

```
EXEC SQL fetch cursorname into :variable1, :variable End EXEC
```

Closing query:

```
EXEC SQL close CursorName End-EXEC
```

Example in C:

```
EXEC SQL Begin Declare Section;
varchar fname [16], lname [16], address [50];
char SSN[10];
int Salary, Sqlcode;
EXEC SQL End Declare Section;
while(1){
    prompt("Enter SSN:", SSN);
EXEC SQL
Select FNAME, LNAME, Address, Salary
Into :fname, :lname, :address, :Salary
from Employee
where SSN = :SSN;
End-EXEC
if (SQLcode == 0)
    printf(fname, lname, Salary);
else printf("Error");
}
```

```
prompt("more SSN != 0
: 0 or NO", loop);
}
```

| Static Embedded SQL | Dynamic SQL |
|---|---|
| SQL Decide before program | queries are executed runtime |
| faster | Slower |
| Compiled at Compilation | execution |
| All task done in Compile time | It is done in runtime |
| no Immediate Execution | Executes Immediately |
| Less flexible | more |

**Advanced SQL features** Least important topic.

Ty Tuple variable:

- A tuple variable is a temporary variable used to rename tables in a query
- defined in FROM clause using AS
- useful when
  - Same table used twice
  - Shorten and readable queries
  - Comparing values

Ex:

```
Select B.customer_name, B.loan_no, L.amount
from borrower as B . loan as L
where L.loan_no = B.loan_no;
```

Other Advanced operation

% Matches any Sequence of characters

_ matches exactly one character