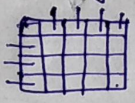
Database System [System that hold particular Database and perform Operations on it]

[Collection of Related Data] → Database

Database
- Structured
  - RDBMS [Table Format]
    - IRCTC
    - University
- Unstructured (90% data)
  - Webpages

DBMS
- SQL Server
- Oracle 9i, 11, 12c etc.
- MySQL
- DB2

• **DBMS:** A database management system is a computerized data keeping system consist of several operations like insertion, deletion, updatation etc. which helps user to operate the data and collect information from that data. [Father of DBMS – EF Codd]

⊛ In a structured database, all the datas are structured by RDBMS (Relational Data Base Management System), where R stands for Relation/table.  Relation/Table Format.

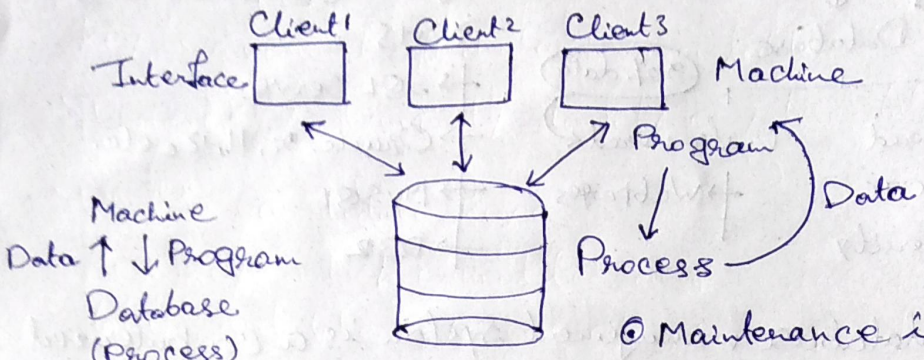
File System Vs. DBMS

[A way of arranging the data in a storage Medium like Hard disk] ← File System

DBMS → [A software that manages the collection of related data]

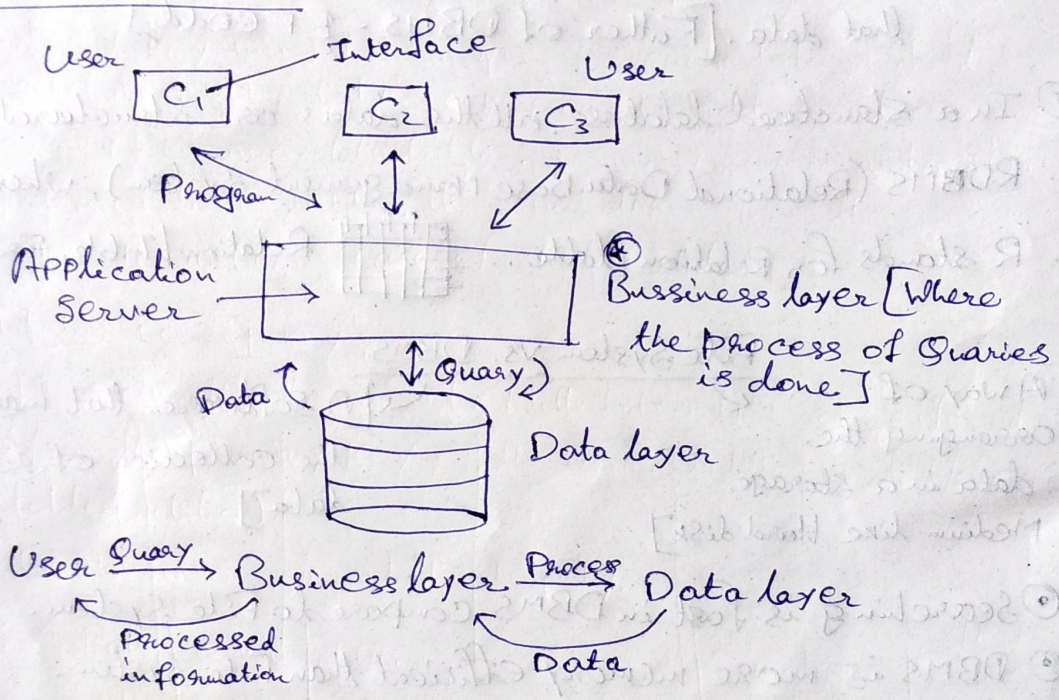
⊙ Searching is fast in DBMS compare to File System.
⊙ DBMS is more memory efficient than File System.
⊙ Information about the meta-data is needed to search a file from file System.
⊙ DBMS provides proper protocals for concurrency [RW, $^{RR}_{WR}$, WW]
⊙ DBMS provides Role based Security for accessing the data.
⊙ DBMS is more complex and costly than file System.
⊙ Multiple users can access data at a time in DBMS.

2-Tier Architecture: Also known as Client-Server architecture



- Maintenance is very easy.
- Main problem is Scalability.
- Another problem is Security.

3-Tier Architecture:



- Scalibility is achieved.
- Easy to handle.
- Security purposes are handled.
- System gets complex, as a result cost and maintenance is increased.

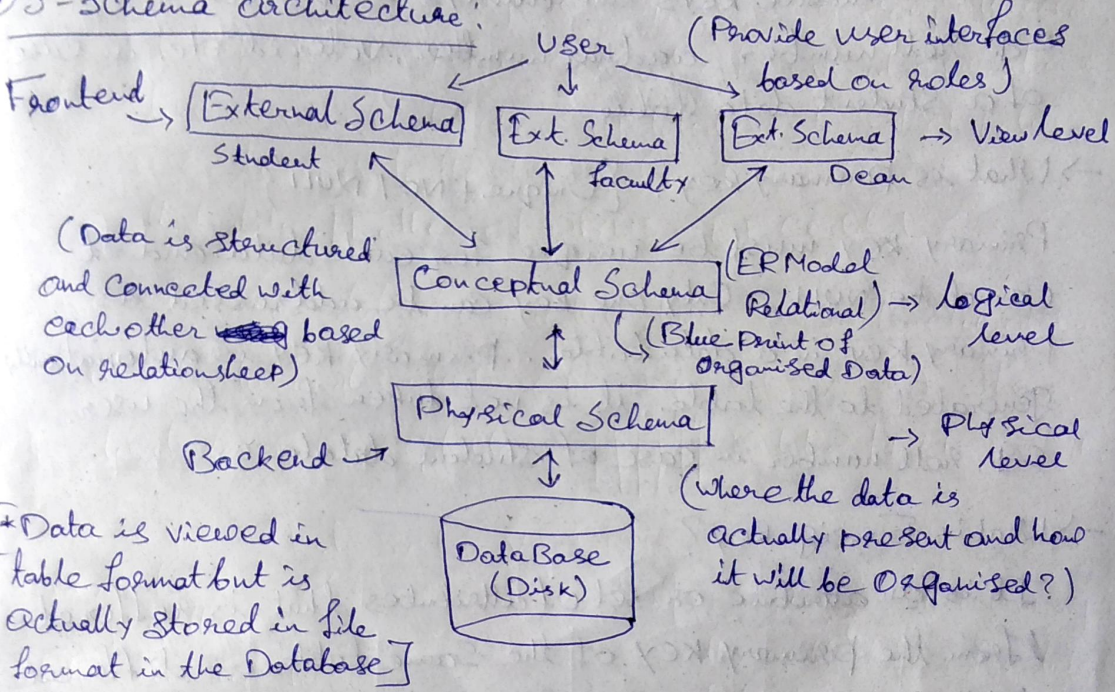
- **Schema:** Schema is a logical representation of a database. e.g RDBMS supports table format, E-R model supports entities-relationship format. These formats are actually a logical representation of data.

Schema Student → | Roll no. | name | addresses |

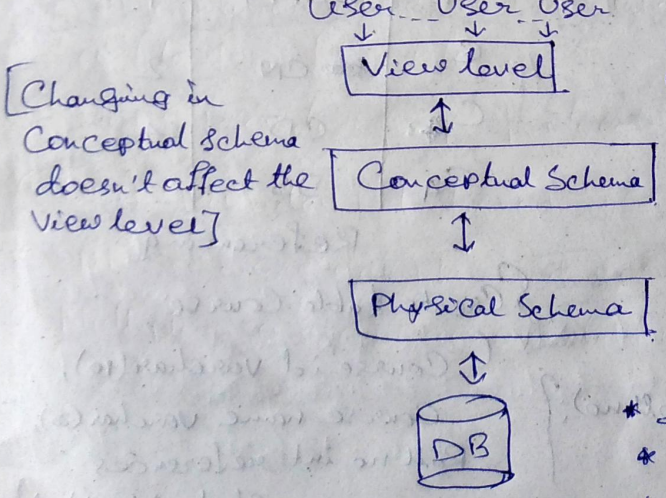
Schema Course → | Cid | name | Fname |

* Implementation of schema is done by SQL. In SQL, DDL (Data Definition language) is basically used to design the schema.

⊙ 3-Schema architecture:



⊙ Data independence: Data is independently stored from the user means all the internal working of storing a data is abstracted from the user.



Concept of ~~Candidate~~ Key

→ What is Key?

Key is a normal attribute from a table.

→ Use of key: Uniquely identify any two data from the table.

→ What is Candidate key?

Those keys which cannot be same in case of any two data. Candidate keys are always unique for each data. e.g. roll number, aadhar number, voter id etc. in case of a student data table.

→ What is primary key? {Unique + NOT NULL}

Primary key must be unique for each data and it cannot be NULL. Only one key can be addressed as primary key in a data table. Primary key is automatically generated to the table, it is not taken from the user. e.g. roll number in case of Student database.

→ What is Foreign key?

It is an attribute or set of attributes that is reffered from the primary key of the same table or different table (relation).

⊙ Maintains referential Integrity.

PK ← Roll | Name | addr.

Roll	Name	addr.
1	A	ab
2	B	abc
3	C	ed
4	B	fc

Referenced

fk

Courseid	Coursename	roll
C_1	DBMS	1
C_2	CN	2
C_3	CD	4
C_4	DS	

Referencing

{ After table Course
ADD Constraint fk
foreign key (Roll no)
references Student (Roll no); }

{ Create table Course
(Course id varchar(10),
Course name varchar(20),
Roll no int references
Student (Roll no)); }

Foreign key

Referenced table

Insert → No violation

Delete → May cause violation

Referencial Integrity →
- * On delete cascade
- * On delete set Null
- * On delete No Action

Update → May cause violation

Referencing table

Insert → May cause Violation

Delete → No violation

Update → May cause violation.

Super key: Super key is a combination of all possible attributes which can uniquely identify two tuples in a table.

** Super set of any candidate key is Super key.

Q) R($A_1 A_2 A_3 A_4 \dots A_n$) then how many super keys are possible?

if → A_1 is CK

Ans. 2^{n-1} ✓

→ A_1, A_2 Both CK

for A_1, 2^{n-1} for A_2, 2^{n-1}

for $A_1 A_2$, 2^{n-2}

$\therefore 2^{n-1} + 2^{n-1} - 2^{n-2}$ ✓

CK Roll	name	age
1	A	AB
2	B	CD
3	C	EF

Super:
Roll
Roll, name
Roll, age
Roll, name, age

$A_1 A_2$, $A_3 A_4$ both jointly CK

$A_1 A_2 \rightarrow 2^{n-2}$

$A_3 A_4 \rightarrow 2^{n-2}$

$A_1 A_2 A_3 A_4 \rightarrow 2^{n-4}$

Ans. $2^{n-2} + 2^{n-2} - 2^{n-4}$ ✓

Entity Relationship Model (ER Model)

↓

A object which has physical existance

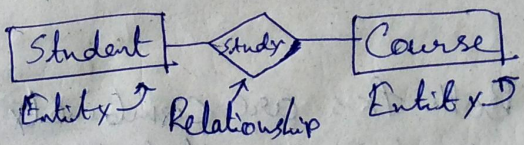
Entity type or Schema [Student (Roll, age, name)]

Attributes

Entity → ▭

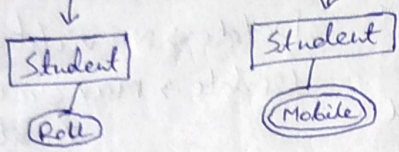
Attributes → ⬭

Relationship → ◇

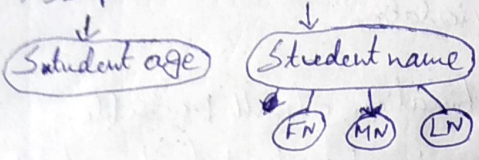


Types of Attributes:

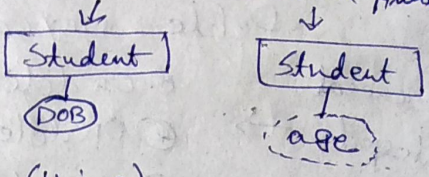
1) Single vs Multivalued Attributes



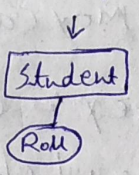
2) Simple vs Composite Attributes



3) Stored vs Derived Attributes



4) Key (Unique) vs Non-key Attributes



All other those are not unique

5) Required vs Optional attribute [Not the part of E-R Model]

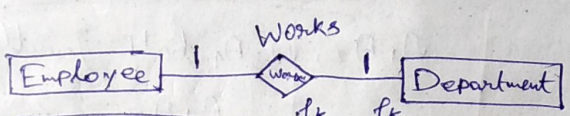
It have to be filled

It can be NULL

6) Complex Attribute -> (Composite + Multivalued)

Degree of Relationship (Cardinality): 1-1, 1-M, M-1, M-M

1-1



Eid	Name	age
E_1	A	20
E_2	B	25
E_3	C	28
E_4	A	24

fk Eid	fk Did
E_1	D_1
E_2	D_2
E_3	D_3

Did	Dname	Loc
D_1	IT	Banglore
D_2	Acc	Delhi
D_3	HR	Delhi

[Both can be merged] Pk = Either Eid Or Did

④ If any relationship has an attribute then that is known as Descriptive attribute.

[Video]

1-M / M-1 — Pk is the fk of many relationship ~~and~~ Entity.

[Lec-17 Gate Smashers]

M-M — Both fks are combinely make PK and the table can't be reduced or merged. [Video]

[Lec-18 Gate Smashers]

Normalization: It is a technique to remove or reduce Redundancy (duplicacy) from a table.

Sid	Sname	Age
→ 1	RAM	20
2	Varun	25
→ 1	RAM	20

Row level Duplicacy

[Resolved using Primary Key]

Sid	Sname	Cid	Cname	Fid	Fname	Salary
→ 1	RAM	C_1	DBMS	F_1	John	30000
2	RAVI	C_2	JAVA	F_2	Bob	40000
→ 3	NITIN	C_1	DBMS	F_1	John	30000
→ 4	AMRIT	C_1	DBMS	F_1	John	30000
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Column level

- Insertion Anomaly
- Deletion Anomaly
- Updation Anomaly

First Normal FORM: The table should not contain any multivalued attribute.

According to EF Codd →

[Multivalued attribute Not in FNF]

Student

Roll	Name	Course
1	Sai	C/C++
2	Harsh	Java
3	Onkar	C/DBMS

1st Representation

Roll	Name	Course
1	Sai	C
1	Sai	C++
2	Harsh	Java
3	Onkar	C
3	Onkar	DBMS

Roll & Course combinely can be the Primary key.

2nd Represen—

RollNo.	Name	C_1	C_2
1	Sai	C	C++
2	Harsh	Java	NULL
3	Onkar	C	DBMS

Roll is the PK.

3rd Repre—

Foreign key

Roll	Name
1	Sai
2	Harsh
3	Onkar

Base table.

Primary key: Roll

Roll	Course
1	C
1	C++
2	Java
3	C
3	DBMS

Foreign key: Roll

Primary key: Roll Course Combinely

• Closure Method:

R(ABCD)
FD{A→B, B→C, C→D}

A⁺ = ABCD, C⁺ = CD
∴ CK = {A}. B⁺ = BCD, D⁺ = D

* AB⁺ = ABCD [Using Transitive Property]
This can't be a ck, this is a Super key.
Prime attribute = {A}.
Non-" " = {B, C, D}

R(ABCD)
FD{A→B, B→C, C→D, D→A}
A⁺ = ABCD, B⁺ = BCDA,
C⁺ = CDAB, D⁺ = DABC.
CK = {A, B, C, D}
Prime Attribute = A, B, C, D
Non-" " = {ϕ}

* FD = Functional Dependency

R(ABCDE) FD = {A→B, BC→D E→C, D→A}

A⁺ ≠ AB D⁺ = ABD
B⁺ = B
BC⁺ = BCDA
E⁺ = EC
CE⁺ = CE
E⁺ = EC ~~CK = {BCE}~~
AE⁺ = ABECD ✓
BE⁺ = BCDEA ✓
DE⁺ = DEABC ✓
Prime = {A, B, D, E}
Non-" " = {C}
CK = {AE, BE, DE}

Functional Dependency

Reflexive: if y is subset of x then x → y [Y is determined by X]

Non-trivial → (x) Sid → Sname (x)
Not Reflexive → 1 Ranjit [Here name is determined by id]
2 Ranjit

* Trivial FD - X → Y, X ∩ Y ≠ ϕ [Sid Sname → Sid]
* Non-Trivial FD - X → Y, X ∩ Y = ϕ [Sid → Sname]

Augmentation: if X → Y, then XZ → YZ [Sid → Sname, then Sid Sphn → Sname Sphn]

Transitive: if X → Y, and Y → Z then X → Z

Union: if X → Y and X → Z then X → YZ

Decomposition: if X → YZ then X → Y and X → Z
* [XY → Z then X → Z, Y → Z] ✗ This is invalid

Pseudo transitive: if X → Y and WY → Z then WX → Z.

Composition: if $X \to Y$ and $Z \to W$ then $XZ \to YW$.

2NF
Second Normal Form:

→ Table or relation must be in 1st Normal Form

→ All the non-prime attributes should be fully functional dependent on Candidate key

⊗ There should be no partial dependency in the relation

Ex: R(ABCDEF)

FD: {C→F, E→A, EC→D, A→B}

CE⁺ = CFEADB

CK = {CE}

Prime att. = {C, E}

Non-" " = {A, B, D, F}

Here, C→F, E→A having partial dependency, so R(ABCDEF) is not in 2NF.

Customer

CID	SID	Loc
1	1	Delhi
1	3	Mumbai
2	1	Delhi
3	2	Bangalore
4	3	Mumbai

CK: CID SID

Prime Attributes: CID, SID

Non prime: Loc

2NF Convertion

CID	SID
1	1
1	3
2	1
3	2
4	3

CK

2NF

SID	Loc
1	Delhi
2	Mumbai
3	Bangalore

CK ← Full FD

2NF

3NF
Third Normal Form:

→ table or relation must be in 2NF.

→ There should be no transitive dependency in the table

R(ABCD) AB→C ✓ C→D ✗

⊗ CK ⟹ AB^+ = ABCD

C→D Not possible in 3NF coz L.H.S = R.H.S = NP

L.H.S should be a CK or R.H.S should be a prime attribute

Roll	State	City
1	Punjab	Mohali
2	Haryana	Ambala
3	Punjab	Mohali
4	Haryana	Ambala
5	Bihar	Patna

CK = {Roll}, Prime = {Roll}

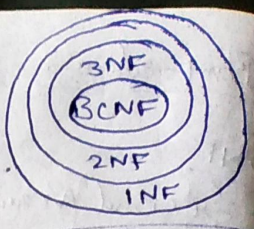
Non-Prime = {State, City}

Roll→State ✓

State→City ✗ [Both are non-prime att.]

Roll→City ✓

BCNF (Boyce Codd Normal Form):



→ Should be in 3NF. → No multivalued dependency.

→ L.H.S of each FD should be CK or SK.

Student

Roll	Name	Voter id	age
1	A	1110	20
2	B	2213	21
3	A	1212	21
4	C	2121	20
5	B	3223	19

CK = {Roll, Voter id}

FD: { Roll → name (CK), Roll → Voter id (CK), Voter id → age (CK), Voter id → Roll (CK) }

All are valid.

Lossy & Lossless Decomposition:

A	B	C
1	2	1
2	2	2
3	3	2

→ $R_1(AB)$

→ $R_2(BC)$

R_1

A	B
1	2
2	2
3	3

R_2

B	C
2	1
2	2
3	2

Find the value of C if the value of A = '1'

→ Select $R_2.C$ from R_2 Natural Join R_1 where $R_1.A = '1'$;

Natural Join:

R'

A	B	C
1	2	1
1	2	2
2	2	1
2	2	2
3	3	2

→ This type of data inconsistency after merging two tables is known as lossy decomposition.

R_1		R_2	
1	2	2	1
1	2	2	2
1	2	3	2
2	2	2	1
2	2	2	2
2	2	3	2
3	3	2	1
3	3	2	2
3	3	3	2

Natural Join selects those rows which have same values for the common attributes between both the tables.

* Common attribute should be CK or SK of either R_1 or R_2 or Both.

For Lossless Decomposition,

⑧ 5th Normal Form:

→ In 4th Normal Form

→ Lossless Decomposition

⑨ Minimal Cover: {A→B, C→B, D→ABC, AC→D}

✓A⁺ = A
✓C⁺ = C
D⁺ = BCD
✓D⁺ = CDB
D⁺ = ADB
AC⁺ = ACB

{A→B, C→B, D→A, D→B, D→C, AC→D}
→ {A→B, C→B, D→A, D→C, AC→D}
{A→B, C→B, D→AC, AC→D} ✓

A⁺ = AB
C⁺ = CB

Ex: R(ABCDEF), Check the highest Normal Form

FD: {AB→C, C→DE, E→F, F→A}

Step1: Find CK and relation
S2: Prime & Non-prime

✓AB⁺ = ABCDEF ×A⁺ = A ×B⁺ = B
×C⁺ = CDEFA ×F⁺ = AF
×D⁺ = D ×E⁺ = EFA
✓FB⁺ = FBACDE
✓EB⁺ = FEBACD
✓CB⁺ = BCDEFA

CK = {AB, CB, EB, FB}
Prime att. = {A, B, C, E, F}
Non- " " = {D}

S3: Check all the normal start from BCNF.--

	AB→C	C→DE	E→F	F→A
BCNF	✓	×	×	×
3NF	✓	×	✓	✓
2NF	✓	×	✓	✓
1NF	✓	✓	✓	✓

Imp.

1NF → No multivalued attribute

2NF → L.H.S should not be a proper subset of a CK or R.H.S should be a prime attribute.

3NF → L.H.S should be a CK or SK or R.H.S should be a prime attribute.

BCNF → L.H.S should be a candidate key or Super key.

R(ABCDEF)

FD's {AB→C, C→D, C→E, E→F, F→A}

CK={AB, FB, EB, CB}

This table is not in 2NF because of {C→D} attribute.

Let's Convert it into 2NF.

R(ABCDEF)

R1 (ABEF) ↓ R1 (ABCEF)

R2 (CD)

C→D C+=CD

∵ C is the CK in R2

[So, according to the lossless decomposition we can common C in both of the table.]

∴ Both the table is now in 2NF.

Next Target, Check for 3NF

Now,

R1	R2
(ABEF)	(CD)
{CK= AB, FB, EB, CB}	{CK=C}
{FD= AB→C, C→E, E→F, F→A}	{FD=C→D}

All are in 3NF

Next check for BCNF,

R1{FD= AB→C, C→E, E→F, F→A}, R2{FD= C→D}

R1 → R' (C,E,F), R'' (A,B,C)

R1 → R1 (AB→C, (ABC), CK→AB); R3 (C→E, (CE), CK→C); R4 (E→F, (EF), CK→E); R5 (F→A, (FA), CK→F)

R1, R2, R3, R4, R5

BCNF Satisfied

0% Redundancy

R(ABCDEFGH)

F={CH→G, A→BC, B→CFH, E→A, F→EG}

~~CH+={C, G, H,~~

ABCEFGH

={CH→G, A→B, A→C, B→C, B→F, B→H, E→A, F→E, F→G}

AD+ = ADBCFHEG

ED+ = EDABCFHG

FD+ = FDEGABCH

BD+ = BDCFHEGA

CK={AD, ED, FD, BD}

4 CK

Equivalence of FD:

$X = \{A \to B, B \to C\} \mid Y = \{A \to B, B \to C, A \to C\}$

- Check for X covers Y ($X \geq Y$)

→ closure of Y ⇒ $A^+ = ABC$

Check from X → $B^+ = BC$

- Check for Y covers X ($X \subseteq Y$)

→ Closure of X ⇒ $A^+ = ABC$

Check from Y → $B^+ = BC$

$\therefore X \equiv Y$

$X = \{AB \to CD, B \to C, C \to D\} \mid Y = \{AB \to C, AB \to D, C \to D\}$

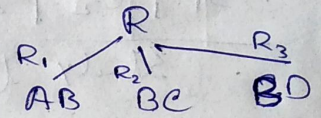
$X \geq Y$: $AB^+ = ABCD$ $C^+ = CD$

$X \subseteq Y$: $AB^+ = ABCD$ $B^+ = B$ $C^+ = CD$

$\therefore X \neq Y$.

Dependency Preserving Decomposition:

$R(ABCD)$ $FD = \{A \to B, B \to C, C \to D, D \to B\}$



$R_1(AB)$	$R_2(BC)$	$R_3(BD)$
$A \to B$	$B \to C$	$B \to D$
$B \to A$	$C \to B$	$D \to B$

$A^+ = AB$ $B^+ = BCD$

$C^+ = CDB$ $D^+ = DB$

~~$A \to B$ $B \to C$ $B \to D$~~

$R_1 \cup R_2 \cup R_3 \Rightarrow A \to B, B \to C, B \to D, C \to B, D \to B$

$A^+ = ABCD$, $B^+ = CD$, $C^+ = CBD$, $D^+ = DBC$.

Satisfied Depending Preservedcy.

- R(ABCD) FD = {AB→CD, D→A}

$R_1(AD)$	$R_2(BCD)$
×A→D	×B→CD, ×BC→D
✓D→A	×C→BD, ×CD→B
	✓D→BC, ✓BD→C

$AB^+ = ABCD$

$D^+ = DA$

$DB^+ = DABC$

- D→A
  BD→C

$D^+ = DA$ $BD^+ = BDC$ ×

Dependency Preserving not Satisfied.

- Joins: (i) Cross Join, (ii) Natural Join, (iii) Conditional Join, (iv) Equivalent Join, (v) Self Join, (vi) Outer Join < left, right, full

- Self Join: A table is Joined with itself.

Find Student who is enrolled in at least 2 Courses.

S_id	C_id	Since
S_1	C_1	2016
S_2	C_2	2017
S_1	C_2	2017

Study (L_1)

S_id	C_id	Since
S_1	C_1	2016
S_2	C_2	2017
S_1	C_2	2017

Study (L_2)

Select L_1.S_id from Study as L_1, Study as L_2, where

$L_1.S\_id = L_2.S\_id$ and
$L_1.C\_id <> L_2.C\_id$;

- Equi Join: It is like natural Join but here we can equalize any two attributes of the tables.

Find emp name who worked in a dept having location same as their address?

Eno	Ename	Address
1	Ram	Delhi
2	Varun	Chd
3	Ravi	Chd
4	Amrit	Delhi

Emp

Dept No.	Loc	Eno
D_1	Delhi	1
D_2	Pune	2
D_3	Patna	4

Dept

Select Ename from Emp, Dept where Emp.Eno = Dept.Eno and Emp.Address = Dept.Loc;

- **Left Outer Join:** It is a join which returns the matching rows which are in left table but not in right table

Eno	Ename	Dno
E_1	Varun	D_1
E_2	Amrit	D_2
E_3	Ravi	D_1
E_4	Nitin	—

Emp

Dno	Dname	Loc
D_1	IT	Delhi
D_2	HR	Hyd
D_3	Finance	Pune

Dept

ENO	Ename	Dname	Loc
E_1	Varun	IT	Delhi
E_2	Amrit	HR	Hyd
E_3	Ravi	IT	Delhi
E_4	Nitin	—	

LOJ

select eno, ename, dname, loc from emp left outer join dept on (emp.Dno = dept.Dno)

- Right outer join is vice-versa of left outer join.

⊙ Relational Algebra (1970)

→ Procedural Query Language

OR

Formal Query Language

Operators

Basic Operator
- → Projection (π)
- → Selection (σ)
- → Cross Product (×)
- → Union (∪)
- → Rename (ρ)
- → Set Difference (−)

Derived Operator
- → Join (⋈)
- → Intersect (∩) (x∩y) = x − (x − y)
- → Division (/, ÷)

1) Projection (π) – Query: Retrieve the rollno from table (Student)

RollNO	Name	Age
1	A	20
2	B	21
3	A	19

Ans → π_{RollNO}(Student)

RollNO
1
2
3

Gives the column wise Answer.

π_{Name}(Student)

Name
A
B

Ans.

2) Selection (σ) – Query: Retrieve the name of student whose RollNO = '2'.

RollNO	Name	Age
1	A	20
2	B	21
3	A	19

Ans = B

$\pi_{name}(\sigma_{RollNO='2'}(Student))$

Projection = Column
Selection = Row

Gives Row wise Answer

3) Cross Product :-

R_1

A	B	C
1	2	3
2	1	4

R_2

C	D	E
3	4	5
2	1	2

$R_1 \times R_2$

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

Column = m + n = 3 + 3 = 6
Row = m × n = 2 × 2 = 4.

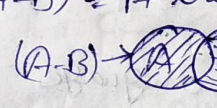
4) Set Difference (−) : (A − B) = A but not B = A ∩ B'

Roll	Name
1	A
2	B
3	C

(Student)

Empno.	Name
7	E
1	A

(Employee)



(Student − Employee)

=

Roll	Name
2	B
3	C

Column name will always be the 1st column of the table

5) Union: (1,2,3)S_1 ∪ (3,4)S_2 = (1,2,3,4)$S_1 \cup S_2$

Roll	Name
1	A
2	B
3	C

Emp no.	Name
7	E
1	A

Roll	Name
1	A
2	B
3	C
7	E

6) Division (Derived Method): ×, −, π, σ Combinly used

Sid	Cid
S_1	C_1
S_2	C_1
S_1	C_2
S_3	C_2

En

Cid
C_1
C_2

Co

Query: Retrieve Sid of students who enrolled in* every course.

* every/all means division method

$A(x,y) / B(y) \Rightarrow En(Sid, Cid) / Co(Cid) = Sid$

$$\left(\pi_{Sid}(En) - \pi_{Sid}\left(\left(\pi_{Sid}(En) \times \pi_{Cid}(Co)\right) - (En)\right)\right) = S_1$$

◉ Structure Query Language (SQL) Founded by EF Codd in 1970

A language used to ~~interact~~ connect user with the database. SQL is used in RDBMS (Relational DB)

→ SQL is domain-specific & declarative language
→ DDL, DML, DCL, TCL
→ Keys & Constraints
→ Operators (Like, between, in*, not-in*, Conditional)
→ Clauses (distinct, order by, group by, from, having)
→ aggregate functions
→ Joins & Nested Queries
→ PL SQL (Triggers, function, Cursor, Procedures)
↓
Procedural SQL

SQL Commands

- **Data Definition language (DDL)**
  - Creat
  - Alter
  - Drop
  - Truncate
  - Rename
- **Data Manipulation language (DML)**
  - Select
  - Insert
  - Update
  - Delete
- **Data Control language (DCL)**
  - Grant
  - Revoke
- **Transaction Control language (TCL)**
  - Commit
  - Roll back
  - Save point
- **Constraints**
  - PK
  - FK
  - Check
  - Unique
  - Default
  - Not Null

* create table employee → Table name
(
  id int,
  name Varchar2(10),
  Salary number(10)
); → Column names.

desc emp;

* **Alter Command:**
  - → Add Columns
  - → Remove Columns
  - → Modify datatype
  - → Modify datatype length
  - → Add Constraints
  - → Remove Constraints
  - → Rename Column / table

Alter table Student Add address varchar(20);

ID	name	address

Student

Alter	Update
DDL	DML
Change in Structure	Change in Data
e.g. Add / Remove Column	e.g. update <tablename> Set Salary = Salary * 2;
Change the table.	Change in data under table

* Select works Columnwise
  where " Row wise

Delete	Drop	Truncate
DML	DDL	DDL
Delete rows from a table	Whole table is deleted.	Delete rows from a table.
Delete from Student;	Drop table Student;	Truncate Student;
Delete the rows one by one		Whole table rows deleted at a time
where condition is there		no condition
Rollback is possible		Rollback is not possible
Slower but flexible		faster but not flexible.

⊙ Constraints in SQL: Restrictions in database -

1) Unique (Duplicacy not allowed)
2) Not Null (The particular column can't be NULL)
3) Primary key => Unique + Not NULL
4) Checked (To fix some value in a column)
5) Foreign key
6) Default (If any value is not given then a default value will be put there).

⊙ SQL Queries & Sub-Queries:

Emp

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

→ Write a SQL Query to display maximum Salary from Emp table

Ans Select Max(Salary) from Emp;
[Aggregate function] = 50000

→ Write a SQL Query to display Employee name who taking maximum Salary.

Outer Query ↗ Inner Query ↖

Ans: Select Ename from Emp where Salary = (Select Max(Salary) from Emp);
= Varun

→ Write a SQL Query to display Second Highest Salary from Emp table?

Ans Select Max(Salary) from Emp where Salary <> (Select Max(Salary) from Emp);
= 10000 (2nd Highest)

→ Write a SQL Query to display Employee name who is taking Second highest Salary?

Ans Select Ename from Emp where Salary = (Select Max(Salary) from Emp where Salary <> (Select Max(Salary) from Emp));
= Nitin.

→ Write a query to display all the dept names along with no. of Emps working in that. [* Group by Clause]

Ans Select dept, Count(*) from Emp group by dept.

HR	2
MRKT	2
IT	1

* Aggregate functions: Max, Min, Sum, Average, Count.

→ Write a query to display all the dept. names where no. of emps are less than 2.

Ans Select Dept from Emp ~~where Dept = (Select Dept, Cou~~
group by Dept having Count(*)<2. = IT

Select Ename from Emp where dept In(
Select Dept from Emp group by Dept having Count(*)<2);
= Varun.

* between => Salary between (5000, 25000).
Select * from emp order by Salary desc;

→ Write a query to display highest Salary department wise and name of emp who is taking that Salary.

Ans Select Max(Salary) from Emp group by Dept;

⊙ In / Not In

Emp

Eid (PK)	Ename	Add
1	Ravi	Chd
2	Varun	Delhi
3	Nitin	Pune
4	Robin	Bangalore
5	Anmay	Chd

Q1) Detail of Emp whose address is either Delhi or chd or Pune;

→ Select * from Emp where Add In ('Delhi', 'Chd', 'Pune');

Project

Eid (FK)	Pid (PK)	Pname	Loc
1	P_1	IOT	Bangalore
5	P_2	BIG DATA	Delhi
3	P_3	Retail	Mumbai
4	P_4	Android	Hyderabad

Q2) Find the name of Emp who are working on a project

→ Select Ename from Emp where Eid In (Select Distinct (Eid) from Project);

⊙ Exists / Not Exists: Find the detail of Emp who is working on at least one project?

⊛ Used in Correlated nested Queries.

→ Select * from Emp where Eid Exists (Select Eid from project where Emp.Eid = Project.Eid);

⊙ Aggregate function:

1) Select Max(Salary) from Emp
2) Select Min(Salary) from Emp
3) Select Count(*) from Emp [Count the tuples]
4) Select Sum(Salary) from Emp
5) Select Avg(Salary) from Emp
   ↳ Sum(Salary) / Count(Salary)

* Distinct = No Repeatation
Distinct Count(*)

① Correlated Sub Query (Synchronized Query):

→ It is a Subquery that uses values from outer Query.

→ Top to down approach.

Find all employee details who work in a department.

Ans. Select * from Emp where exists (Select * from dept where dept.Eid = Emp.Eid);

[Select * from Emp where Eid In (Select Eid from Dept);]

↙ Nested Subquery

Select attributes from Emp, Dept where Emp.Eid = Dept.Eid;

↙ Joins

Q) Find Nth highest Salary using SQL.

Select Salary from Emp e1 where N-1 =
(Select Distinct Count (Salary)
where e2.Salary > e1.Salary);

Q) Display the last name of employees who have 'A' as second character in their names.

→ To solve this type of quiries we basically use Like command in the Query.

Select last_name from emp where last_name like '_A%';

② [With like command we use '_' and '%', where % means any String/strings but _ means any Single character Strings.]

Q) A command to remove relation from SQL database?

→ Drop table <tablename>

Q) In the following, Schema R is R(a,b)

1) Select * from R

2) (Select * from R) Intersect (Select * from R)

3) Select distinct * from R

A) Q_1 Q_2 Q_3 produce same result

B) Only Q_1 Q_2 produce same result

C) Only Q_2 Q_3 " " "

d) Q_1 Q_2 Q_3 produce diff result.

* [$Q_1 \cap Q_2$ means Common and unique att. from two tables]

- PL-SQL (Procedural-SQL) (What to do + How to do)

Function → Procedure → Trigger → Cursor

Block Code

Declaration	a int
Exe. Code	begin end
Exp. Handling	error detection

a int
b int
c int
begin
a := 10;
b := 20;
c := a+b;
end;

- Transaction: It is a set of operations used to perform a logical unit of work.

→ A transaction generally represent change in DB.

Major operation of Transaction:

DB → Read → RAM → Commit → DB

- Read [Read the data from DB]
- Write [Change the data from DB]

A=1000
B=2009

R(A) → 1000
A = A-500
W(A) → 500
R(B) → 2000
B = B+500
W(B) → 2500
← Commit ✓

④ Used to save the data in DB from RAM

- ACID properties:
  ↓
  Atomicity Consistency Isolation Durability.

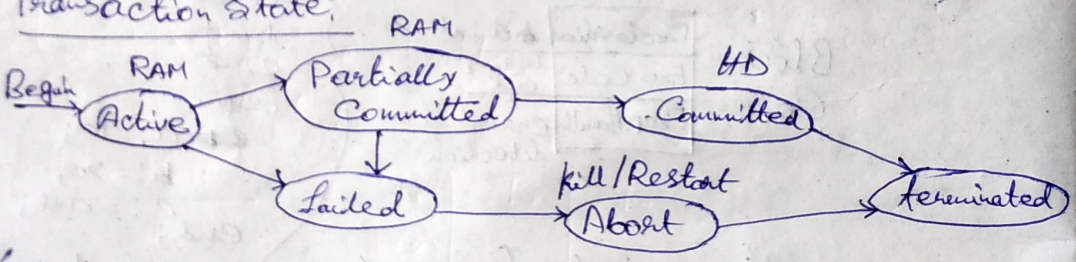
(i) Atomicity: Either all or None. If any kind of failure occurs in between some steps before Commit, then the roll back will occur means all the steps will start from the first.

(ii) Consistency: Before transaction start and after transaction completed the sum of total money should be same.

(iii) Isolation: Trying to convert parallel transactions into serial transactions.

(iv) Durability: All the changes in the database should be permanent.

- Transaction State:



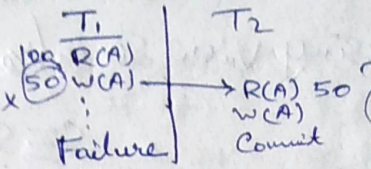
Schedule - It is a chronological execution sequence of multiple transactions.

(i) Serial - One after another transaction will occur. It is consistent for all time but the waiting time increases in this case.

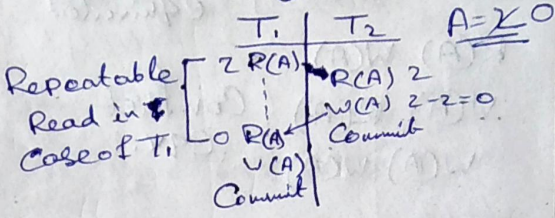
(ii) Parallel - Multiple transactions can be scheduled together by switching the CPU. (Similar as preemptive processes).
Throughput is very high.
Performance is very high.

Types of problems in parallel Schedule:

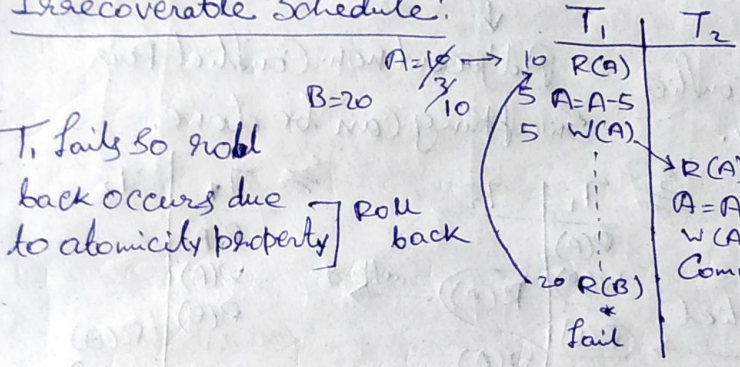
- **Dirty read** – Read-Write Conflict



- Incorrect Summary
- Fantom Read.
- Lost update
- Unrepeatable Read



- **Irrecoverable Schedule:**



- **Cascading Schedule:** In a parallel schedule if one transaction failed due to some reason the other schedules those are running ~~parally~~ parallely should also be aborted otherwise dirty-read problem might occur. This technique is known as cascading schedule.
- Disadvantage is performance, CPU utilization, throughput.
- **Cascadeless Schedule:** If one transaction is ~~running~~ writing, other transaction cannot work on the same data (Can't read the same data). After Committed or aborted of the prev. transaction the transaction can read the data.
- Disadvantage is Write-write problem cannot be solved.
- **Serializability:** Parallel Sched. → Serial Sched. Conversion.

The two method → Conflict
→ View

$T_1 \to T_2$ $T_1 \to T_2 \to T_3$
$T_2 \to T_1$ $T_2 \to T_1 \to T_3$
One after another

Ⓐ If loop is seen in conflict serializability then we have to check using view serializability.

• Conflict eqv.:

R(A) R(A) } Non-Conflict equivalent

R(A) W(A)
W(A) R(A)
W(A) W(A) } Conflict eqv.

R(A) R(B)
R(A) W(B)
W(A) R(B)
W(A) W(B) } Non-Conflict eqv.

$S \equiv S'$

S		S'	
T_1	T_2	T_1	T_2
R(A)		R(A)	
W(A)		W(A)	
	R(A)	R(B)	
	W(A)		R(A)
R(B)			W(A)

↓

Adjacent non-Conflict Pair Swapping can be done.

$S \equiv$	T_1	T_2
	R(A)	
	W(A)	
Adj. non-Conflict pair →	R(A)	
	R(B)	
		W(A)

Swapping →

$S \equiv$	T_1	T_2
	R(A)	
	W(A)	
	R(B)	
		R(A)
		W(A)

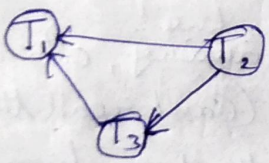
Ⓑ If $S \xrightarrow{CE} S'$ then Serializable also Consistent.

$\therefore S \equiv S'$ (Proved)

• Another Method:

T_1	T_2	T_3
× R(x)		
		R(y) ×
		R(x) ✓
	R(y)	
	R(z)	
		W(y) ✓
	W(z)	
R(z)		
✓ W(x)		
✓ W(z)		

1) Draw Precedence graph:



2) Check cycle:

If no cycle then Conflict Serializable.

∴ Serializable and Consistent.

3) Find Serializable Seq.

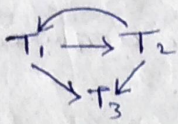
$T_1 \leftarrow T_2 \leftarrow T_3$ $\quad T_2 \rightarrow T_3 \rightarrow T_1$

$T_1 \leftarrow T_3 \leftarrow$ Check Indegree = 0.

• Another example:

S

	T_1	T_2	T_3
100	R(A)		
		W(A) 60	
	W(A) 20		
			W(A) 0



Cycle is present here. So, we can't say if it is Serializable or not.

Here, we use view Serializability.

1) Try to represent the table by some swapping.

S'

	T_1	T_2	T_3
100	R(A)		
60	W(A)		
		W(A) 20	
			W(A) 0

[Check if this table is equivalent with the prev. one or not by using some values]

If they are eqv. then we can say it is Serializable and consistent by view Serializability.

• Concurrency Control protocol: How some parallel T can be serialized. We use locking protocol.

(i) Shared - Execlusive Locking:

→ Shared lock (S) => if T locked data item in Shared mode that allowed to read only.

→ Exclusive lock (X) => Read and write both.

* Problems in S/X locking:

T_1	T_2
S(A)	X(A)
R(A)	R(A)
U(A)	W(A)
	U(A)

1) May not sufficient to produce only serializable Schedule.

	S (R-R)	X
S	Yes	NO (R-W ×)
X	NO (W-R ×)	NO (W-W, R-W, W-R ×)

2) May not free from Irrecoverability.

3) May not free from deadlock.

4) May not free from Starvation.

2-Phase locking (2PL):

→ Growing Phase: locks are acquired and no locks are released.

→ Shrinking phase: locks are released and no locks are acquired.

T_1	T_2
lock S(A)	
	lock S(A)
lock X(B)	
Unlock(A)	
	lock X(D)
Unlock(B)	
	Unlock(A)
	Unlock(B)

Once a growing phase of a transaction is started, all the locks are aquired at that time and once a Shrinking phase is started growing phase can't be start anymore.

Serializability is surely achieved here

*Lock point: Before starting the Shrinking phase (1st unlock) the particular point is known as lock point.

Advantage: Always sure Serializability and Consistency.

Disadvantage: May not free from irrecoverability.
May not free from deadlock.
May not free from starvation.
* Not free from cascading deadlock.

Strict 2PL: It should satisfy the basic 2PL and all exclusive locks should hold untill Commit/Abort.

Rigorous 2PL: It should satisfy the basic 2PL and all shared, exclusive locks should hold untill commit/Abort.

Cascadeless Strict Recoverable

But problem of deadlock and starvation cannot be removed using these.

Time Stamp Ordering Protocol-

→ Unique value assign to every transaction.
→ Tells the order (when they enters into system)
→ Read TS (RTS) - Last (latest) transaction no. which performed Read successfully.
→ Write TS (WTS) = Last (lastest) transaction no. which performed write successfully.

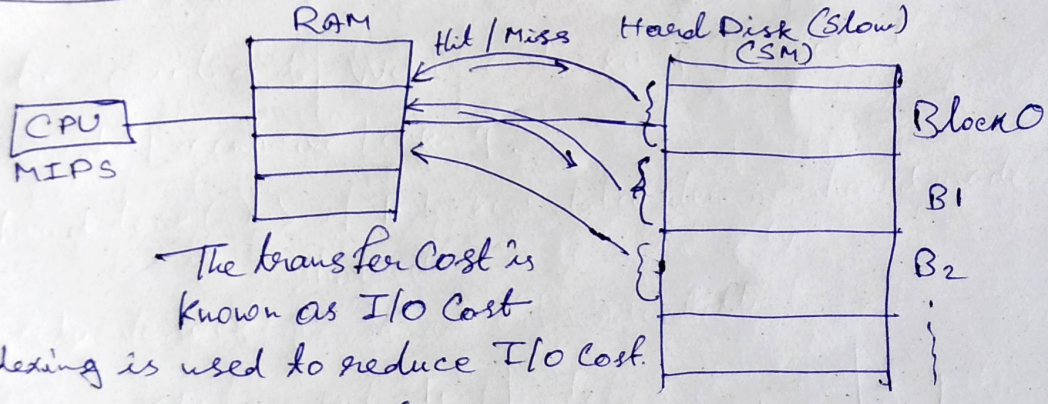
Which transaction comes first, executes first.

Oldest T_1 (100)	T_2 (200)	T_3 (300) youngest	
R(A)			0 > 100
	R(B)		0 > 200
W(C)			0 > 100
		R(B)	0 > 100
R(C)			0 > 300
	W(B)		100 > 100
	Roll back	W(A)	Roll back ← 300 > 200
			100 > 300
			0 > 300

	A (100)	B (200)	C (300)
RTS	ϕ	ϕ	ϕ
WTS	ϕ 300	0	ϕ 100

Indexing:

Select * from Student where Roll no = 1;



Indexing is used to reduce I/O cost.

Index has two parts (key + Pointer)

Two Searching methods are used → Dense, Sparse

- Dense - When data is unordered, Each entry record will be included in the index table
- Sparse - When data is ordered, Each entry of first record of a block is included in the index table.

- Types of Indexes:

1) Primary 2) Clustered 3) Secondary

✓ Ordered file and key → Primary
✓ Ordered file and non-key → Clustered
✗ For Non-ordered Secondary index is used.

As the Primary indexing is used for ordered file and key values, we generally use Sparse method here and complexity becomes $\log_2 N + 1$ [Where N = no. of blocks in the index table]

- Clustered Index:

Clustered is also in Sparse Searching method and the complexity is $\log_2 N + 1 + 1$
↳ For Block Hanker.