Think **Exclusive** Think **ibex.**

Date: 21st Aug, 2019.

DBMS :

It allows to manipulate data.

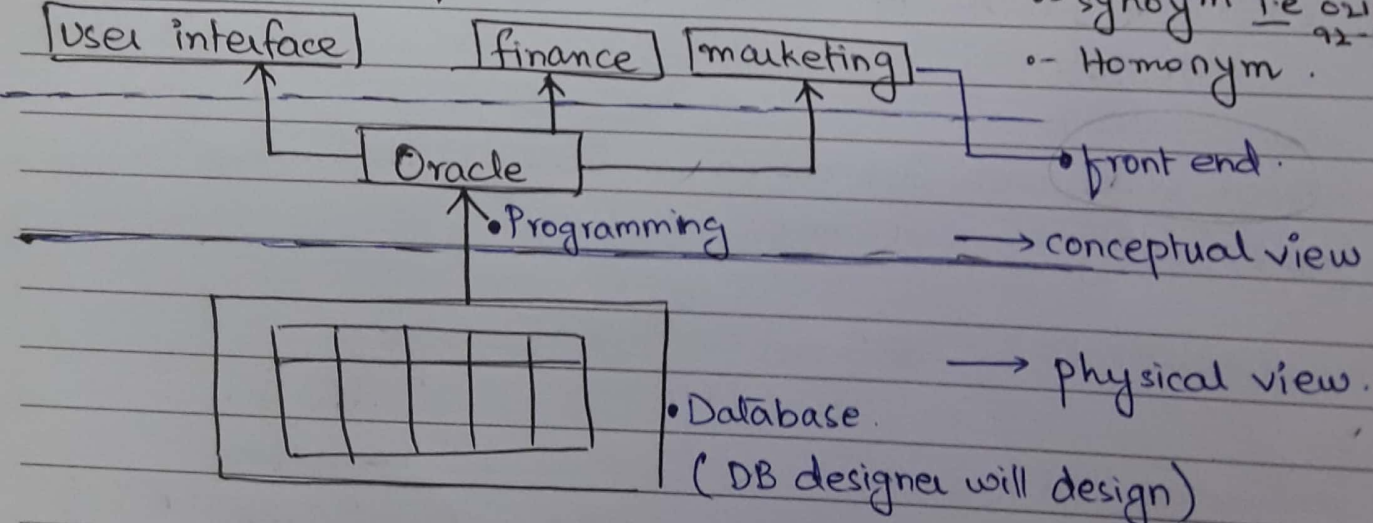
DB : where data is kept.

Advantages of DB

- Reduced data redundancy.
- Improved data integrity.
- Shared data
- Easier access.
- Reduced development time.

File disadvantages.

- Data duplication.
- Inconsistency.
- No relation b/w files.
- No sharing.
- Poor enforcement of standards
  - synonym i.e 3911202 021-3911202 92-21-3911202.
  - Homonym.



→ DB designer have the know how of SAN!

SAN
storage area network.

fault tolerance → those who cannot afford 'server-down' they duplicate their routers and severs as a backup, inorder not to compromise on data availability.

: Data about data is called metadata.

Date: _____

Every database has it's data definition (information of columns, data types, etc).

Transaction → isolated, atomic (critical kind of process)

DATA: Raw and isolated facts about an entity (recorded) e.g: text, audio, image, map, etc.

INFORMATION: Processed, meaningful, usable data.

DATABASE: Collection of similar/related data.

DBMS: Software used to create, manipulate and delete database.

---

* Paradigm shift from file to DBMS:

(Disadvantages) File
- Redundancy of data.
- Inconsistency of data → 021-3911202 / 92-21-3911202 / 3911202. → copies of single data.
- Difficult data access → unsorted (maybe) → searching is difficult.
- Unauthorized access → Access and change can be easily done in a file.
- No concurrent access → Access of same data by multiple users.
- No backup and recovery

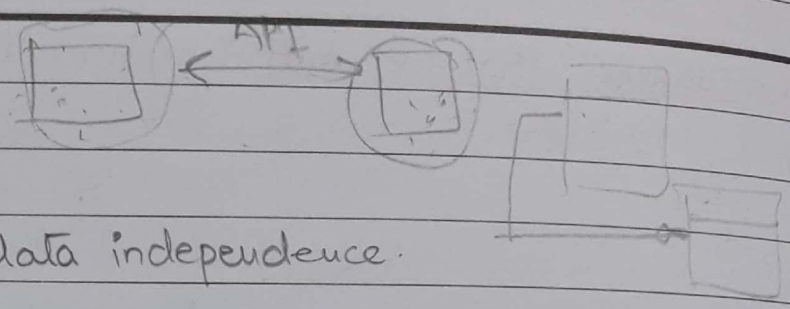
(advantages) DBMS
- Minimized redundancy & data consistency.
- Simplified data access.
- Data security.
- Multiple data views
- Backup & recovery mechanisms.

ibex.

TBS → lossless decomposition in DBMS.

Date: 22/8/19

DBMS architecture



External level → logical data independence.

Conceptual level → physical data independence.

Physical level ⟷ Data Store

Phases of DB design.

1- **Requirements** → checking the pattern of records and taking a survey to create a model (requirement analysis) (needed)

↓

2- **Conceptual Design** → ER modelling. (case tools are used)

↓

3- **Logical Design** → Pseudo code of DBMS. → Representing data in the form of ERD. data into → ERD

↓

4- **Physical Design** → DBMS.

Also called as implementation phase.

Backward engineering → Implementing DBMS first and then forming up an ERD.

* Operators : (used with "where").

- <> (not equal to)
- Between
- Like (search for a pattern)
- In (to specify multiple possible values for a column)

Select Query

Select * from table_name.

Select column from table.

Distinct Clause → help selecting unique values

select DISTINCT * from table → fetch distinct rows of table

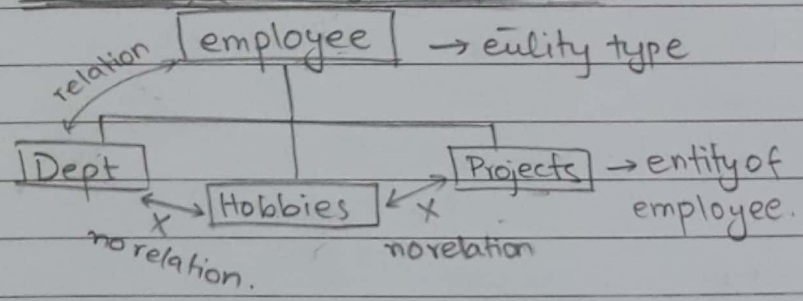
select DISTINCT rows from table

Think **Exclusive** Think **ibex.**

Date: _____

22/8/19.

★ Hierarchical Type of DBMS:

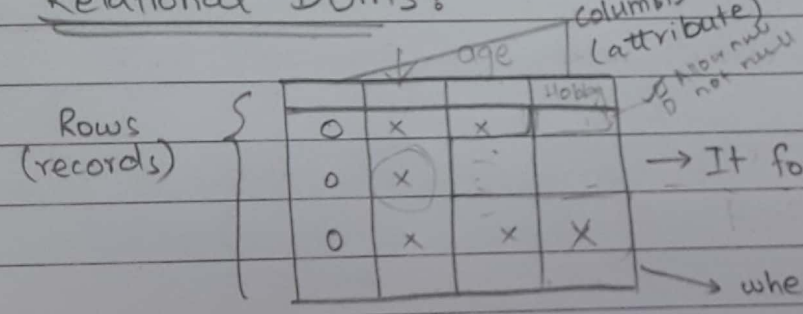


→ Each row in DB represents an entity.

Where Query

- Select * from table column = 20
- Select column from table where column between 20 And 30.
- name starting with letter S:
  Select * from table where Column like 'S%'
- IN operator is same as '=' but we can give multiple values
  Select column from table where column IN(10,20)

★ Relational DBMS:

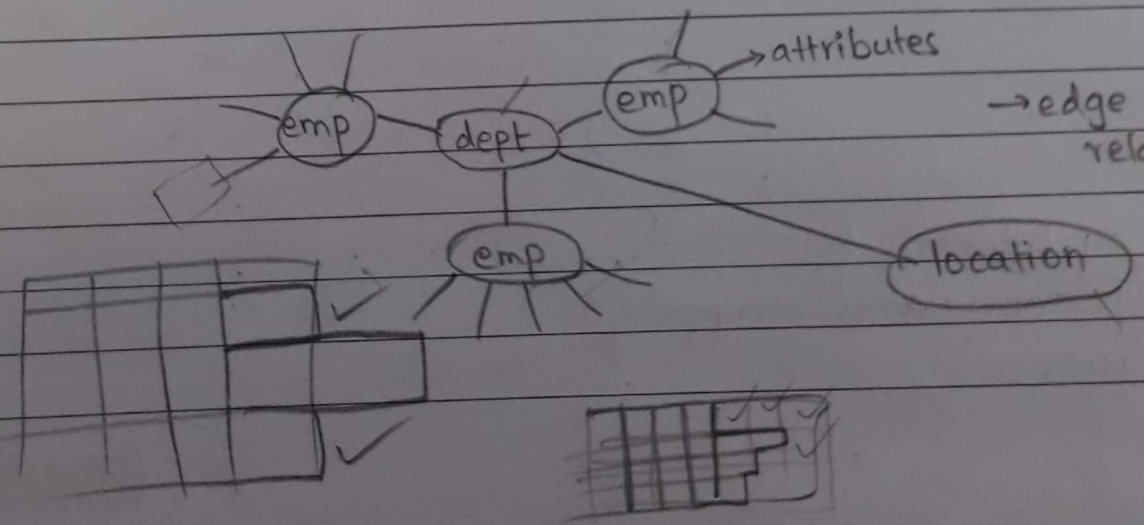


→ It forms a table or a matrix

→ when complete data of a record is not provided then there can be memory wastage (a solution to it is noSQL DBMS)

★ Network Models:

entities → represented by nodes.

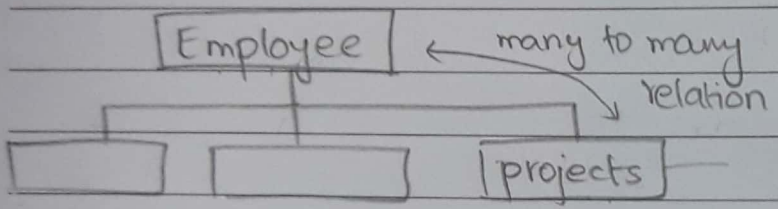


→ edge denotes the relation b/w entities.

ibex.

Date: _____

Jab bhi many to many relationship hoga to aik third entity bhi ban jati hai kyunke us k apne bhi attributes hosakte hain



E — P. → Third entity
(Here controlled redundancy applies)

AND, OR and NOT operator:

- Select * from table where col_1>10 AND col_1<20
- 'between' is alternate to 'And'
  select * from table where col_1 between val_1 and val_2.
- 'IN' is alternate to 'OR'.

ORDER BY:

select * from table order by column desc

Note:

→ DB has it's own buffers, baar baar use hone wala data us main hota hai.

→ Indexing is done in Database.

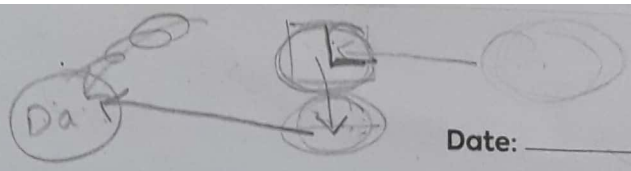
27/8/19.

Database architecture:

- Constructs → Elements of a DB (i.e: table, rows, attribute, records, constraints, relationship)
- Schema → Schema is the definition of constructs. (i.e: ERD)
- Instance → actual data
  → It changes / vaiable.
- DB State → Data present at any mome of time in DB is it's stat
- 3-schema Architecture.
- Database Independance mapping.
- DB languages.

DataBase

- arranged data collection
  - Tables, graphs, trees, class.
- Organized
  - records.
- logically related
- self describing
  - catalog
  - metadata
  - data dictionaries



Date: _____

Relationship b/w entities (3 types)
- → Unary
- → Binary
- → ternary.

insert, update, delete changes the instances and also DB state is changes.

* Schema (also called "intension")
- → it doesn't change frequently.
- → it " contain actual data.
- → it contain details of DB and it's data

* Instance
- → it changes frequently.
- → Actual data present in a table.

* DB State (also called "extension")
- → Instances / data present in a database at particular moment is DB state (i.e: state will change when we delete data)
- → 2 types ——— valid
  - invalid → when wrong data is entered.

* Independence Mapping → physical Data Independence (→ stored in memory)
  - → Logical Data Independence.

* Oracle → sql data modeler → schema is made.

Think Exclusive Think ibex.

Date: _____

* 3-schema Architecture:

External view → Frontend (HTML/CSS)
↓ ↓
Conceptual Model → ERD (relational model)
↓ ↓
Physical Model → storage structure (XML, hadoop, Mongodb etc)

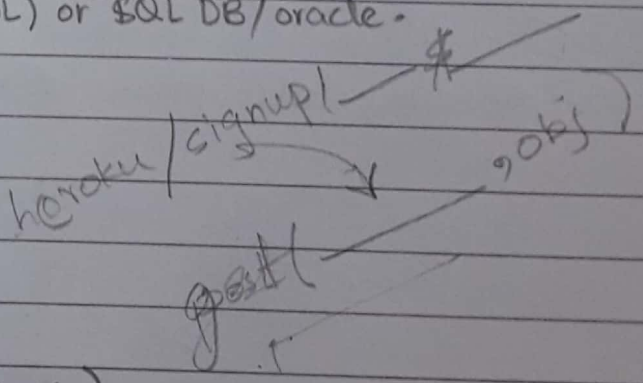
→ This schema supports the physical and logical data independence.

XML doesn't have a schema
→ data can be described in comments.
→ NoSQL is an example

SQL → non-procedural language

PL/SQL → procedural language

External view ← either no-sql (XML) or SQL DB/oracle.
↕ mapping (API)
Conceptual level view
↕ mapping (API)
Physical level view.



* DB Language (used to write DBMS).
- DDL (Data Definition Language).
i.e: Create DB HR.
Create table
Create view
Create users.
- DML (Data Manipulation language).
Insert, Update, Delete, Select, Modify.
↓
DRL

ibex

query compiler → then → query optimizer → then → DBM → manipulation.

Date:

- DCL (Data Control language).
  - → Data authentication & configuation.

28/8/19.

High level / Non-Procedural Language.
- → SQL relational language
- → large query cannot by written.
- → called as "declarative language"

tells 'what' data to be retreived rather than 'how'.

SDL and VDL
- SDL → storage definition language.
- VDL → view definition language

ODBC → Open Data Base Connectivity.

Only DBA give privilaged commands to DB.

~~Procedural Language~~.

DBMS Interface :
- → Standalone query language interface. → SQL.
- → User-friendly interface → menu, graph, form base.
  - Menu(web) base.
  - graphic based
    - Natual language → voice or text.
- → Parametric Interfaces → Bank teller using function keys.

ODBC → used by oracle to connect frontend to backend.

- → Interfaces for DBA (Data Base Administrator)
  - Setting system parameters.
  - Getting user accounts, authorization, etc.

DCL

DBMS utilities → required to perform certain utilities i.e:

- Maintaining servers → performance monitoring.
- load balancing ←
- Batkup DB periodically on tape.
- Performance matritoring utilities.
- Data compression, etc.

Indexing and inveted indexing (TBS)

featued indexing

Accuracy in seach

Decision (?) ← Miss ho gaya

seach with the help of the key word

TBS → DBMS component module.

→ Centralized and client-sewer DBMS.

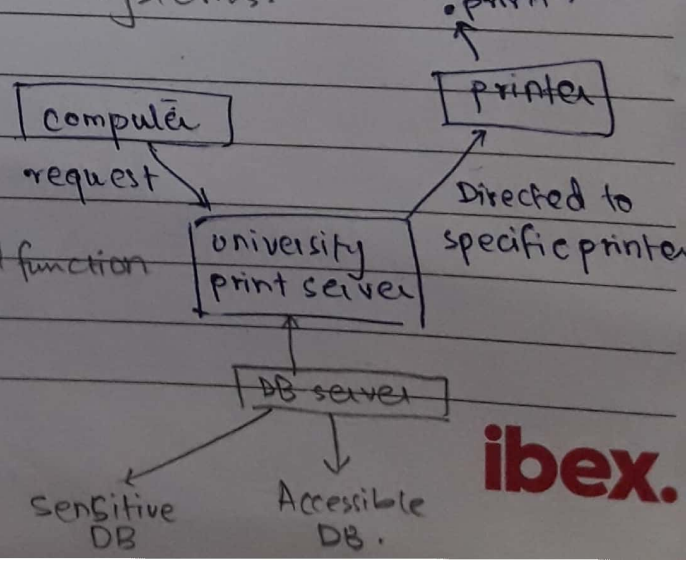
combine everything in single system but user can still connect through remote teminal.

Client and server are found in different systems.

Client sever Architecture ← 2-tier

Specialized server with specialized function

- Print server
- DBMS server.
- Web server
- Email Server.



Date:

3-tier server architecture.

client → App server → DB server / main server

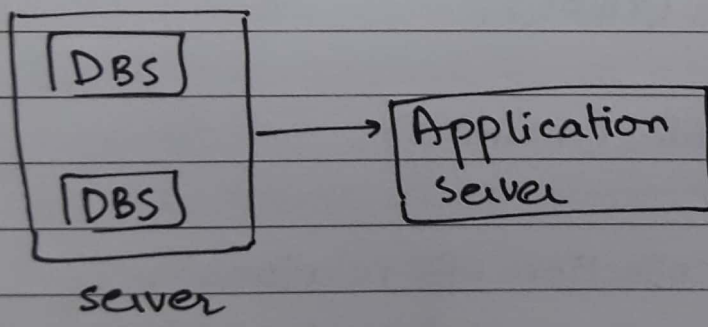
2-tier → direct client to server request.

2nd sep. 19.

3 tier architecture can be distributed or centralized
↙ ↘ ↓
multiple computers multiple DB. one DB.

DB is universe of discourse, which means collection of related data ~~for~~ by taking a real world example.
(kind of mini world or taking a real world facts for ~~DB~~ specific DB).

In order to avoid 'server down' situation when we have multiple clients for a server, we add application server in between. (load balancing)



Note :
3-tier archit
is implemen
to increase
security an
when the
client is
unknown.

← 3-tier architecture →

i.e
GUI web interface — Presentation layer.
Application program, web pages — Business logic layer.
DBMS — DB services layer.

row is document → in MongoDB.
table is collection →

Think **Exclusive** Think **ibex.**

Date: _____

Federated / MultiDB Systems → loosly coupled DBs with high degree of autonomy.

- Good for asynchronous calculations.
- no central security.
- Different time of computation in different nodes.
- Security policy are different for all clusters.

Chapter 5 → relational Model (ch 3 in 6th edition).

→ is main book ki exercise important hai.. dekh lijiye ga beta :)

Relation : 'Table of values'.
-OR-
set of tuples (rows).
-OR-
set of attributes (columns).

- DB is considered as "collection of relations".

Salary → Datatype integer → Domain (constraints) 30,000 to 80,000.
Domain of possible values (range).

Domain 'D' is set of atomic values. → i.e CNIC no/ age
Datatype must be specified for each domain.

13th Sep, 19.

Date:

Data Base Constraints:

Types:

1. Inherent Model-Based Constraints. → each cell has atomic values. i.e: one employee doesn't have 2 salaries.
2. Schema Based Constraints i.e → primary, foreign keys, etc.
3. Application Based or Semantic Constraints.
   ↳ i.e: manager salary > employee salary.

Schema Based Constraints

→ These are explicit constraints.

Application Based constraints.

→ manager ki salary employee se zyada hogi.

→ These are called 'Business Rules'.

→ Cannot be explained/directly expressed in schema, but they have to be enforced though application program.

→ Before inserting any new data business rules are being checked. i.e: manager salary ≮ employee's.

Data Dependency

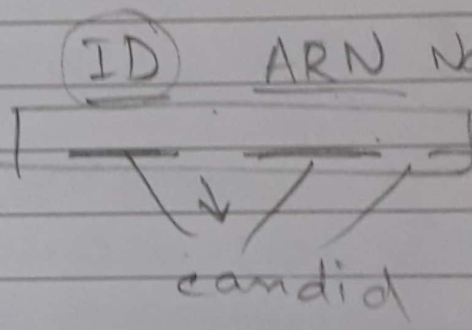
Types:

1. Functional → ~~tells~~ explains which attribute depends on other attribute.
2. Multi-valued. → we cannot put two addresses in single column so we make a column address 2.

primary key → key that uniquely identifies a row.

ID + N + Address → into → single key

super key.



* ID is a min ↑ super key but if a table doesn't have an ID we can combine few/all the attributes and form up a key (super) to uniquely identify the rows.

→ Combination of all non-unique attributes will give minimal 'Super key' which is also a key.
i.e

Serial no + registration no → Sk + key
(candidate keys)

Serial no + colour → Sk but not a key.

→ out of candidate key any one is chosen as 'primary key'
→ primary key attributes are 'underlined' in schema or ERDs.
→ primary key is never null.

Referential Integrity (TBS).

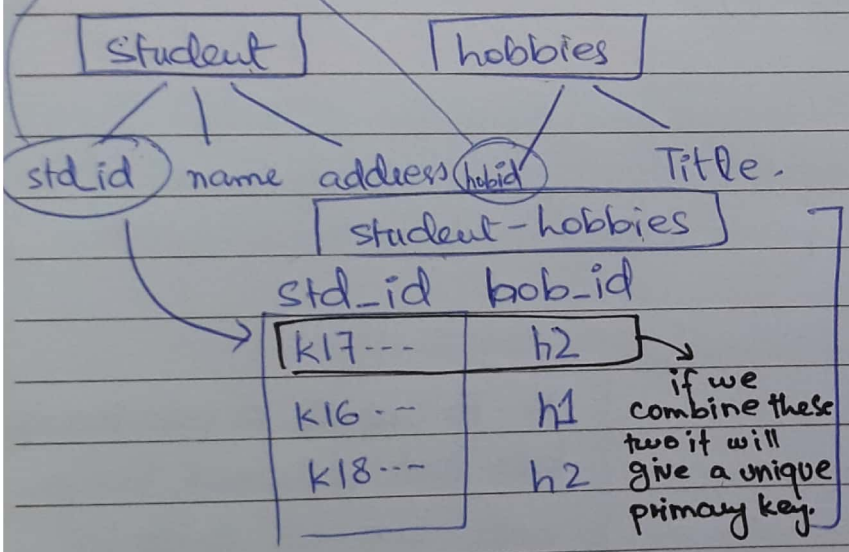
16th Sep, 19.

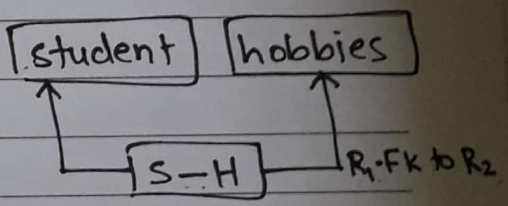
Primary key doesn't have null or duplicate value.

Operations that change relation state:
- Insert
- Update
- Delete.

Referential Integrity → it applies when there is a relationship b/w two tables.

R_2 (Referenced value)





R_1 → referencing table

∴ R_1.FK to R_2

★ student_id in student table is primary key and table is referenced table (R_2) → same goes for hobbies.

★ std_id and hob_id in S—H table are foreign keys. and table is referencing table (R_1)

ibex.

Deleting data from student table is violation of the referential integrity b/c Pk has become Fk in other tables.

→ for this 'Delete cascade' query is executed (it delete data from other tables too). of same student

OR

→ Restrict method is used which ask to remove the referencing data from other tables manually.

OLTP → online transaction processing.

Queries → from chap 6 and 7. (TBS).

Sep/19

Chapter # 06.

SQL
- DDL
- DML
- DCL
- Transaction control
- Backup & recovery.

* To create a relationship b/w two schemas, they both should exist in same catalog.

CREATE Schema AUTHORIZATION 'john'

→ The one who has privilege to create schema.

```
Create table employee(
    Dno     Int         not null,
    name    vaucha(50)  notnull,
    ID      number      notnull,
    Salary  decimal(10.2)   ,
    PRIMARY KEY : ID,
    FOREIGN KEY : Dno references
                  DEPARTMENT Dnum.)
```

another table

attribute

* Dnum is taken as foreign key with the name of Dno.

Hotel

H	Name	H-L	H-Ph
a		2	
b		3	
c		2	

```
Select sum(sal)
from Emp_table
groupby DepID
having sum(sal) > 65000
       avg(sal).
```

```
select Dept_ID, sum(salary)
from emp_table groupby
groupby Dept_ID having
sum(sal) > 29,000.
```

```
select H.H_name, H_phone, L.L_name from Location, Hotel
groupby H.L_ID having count(H.H_name) > 2.
                (where H.L_ID = L.L_ID)
```

Think Exclusive Think ibex. Date:

Sub-query :

Select salary (from Emp_table) where Emp_name = Ali

Select * from Emp_table where salary = ( )

If subquery is returning multiple data then we use IN command-

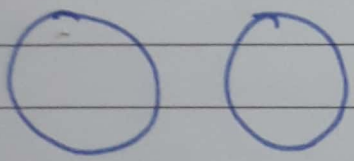
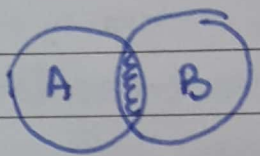
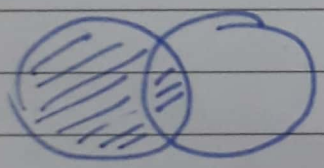


Table A Table B.



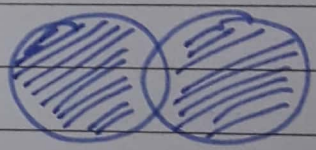
Inner join.

Select * from tableA objA
Innerjoin TableB objB.
On ObjA.key = ObjB.key.



left join

Right join



Selec department_ID from Department where D_name = 'Accounts'

Select * from employee where depatID = ( )

Date:

location table Hotel table

L_name H_name Phone

```
Select  Location.Lname, Hotel.H_name,
        Hotel.Phone
FROM    Location, Hotel
WHERE   Location.L_ID = Hotel.L_ID.
        AND Location.L_name='SWAT'
        AND Hotel.H_name LIKE 'M%'
```

→ give Hotel name, location and phone no of Hotels where hotel name is starting from 'M' and location is SWAT.

1 — 1

Depart 1 — M Student.

Student M — M Couse ID

1 → M Enrollment ← 1 ← M

Mid 2:

book pg # 212 of pdf:
(bring hardcopy in next class)

demobld.sql.
copy paste file and run.

* Select Ename from employee where salary > (select avg(salary) from employee).

Subquery in 'from':

* Select Ename, contact, Dname from emp, (select Dname from Department).

will get only required column from second table.

* Create table under_age_Employee(as
* Select E_name, emp_id, contact from employee where ~~DOB IN~~ Bdate IN (2003 to 2005).)

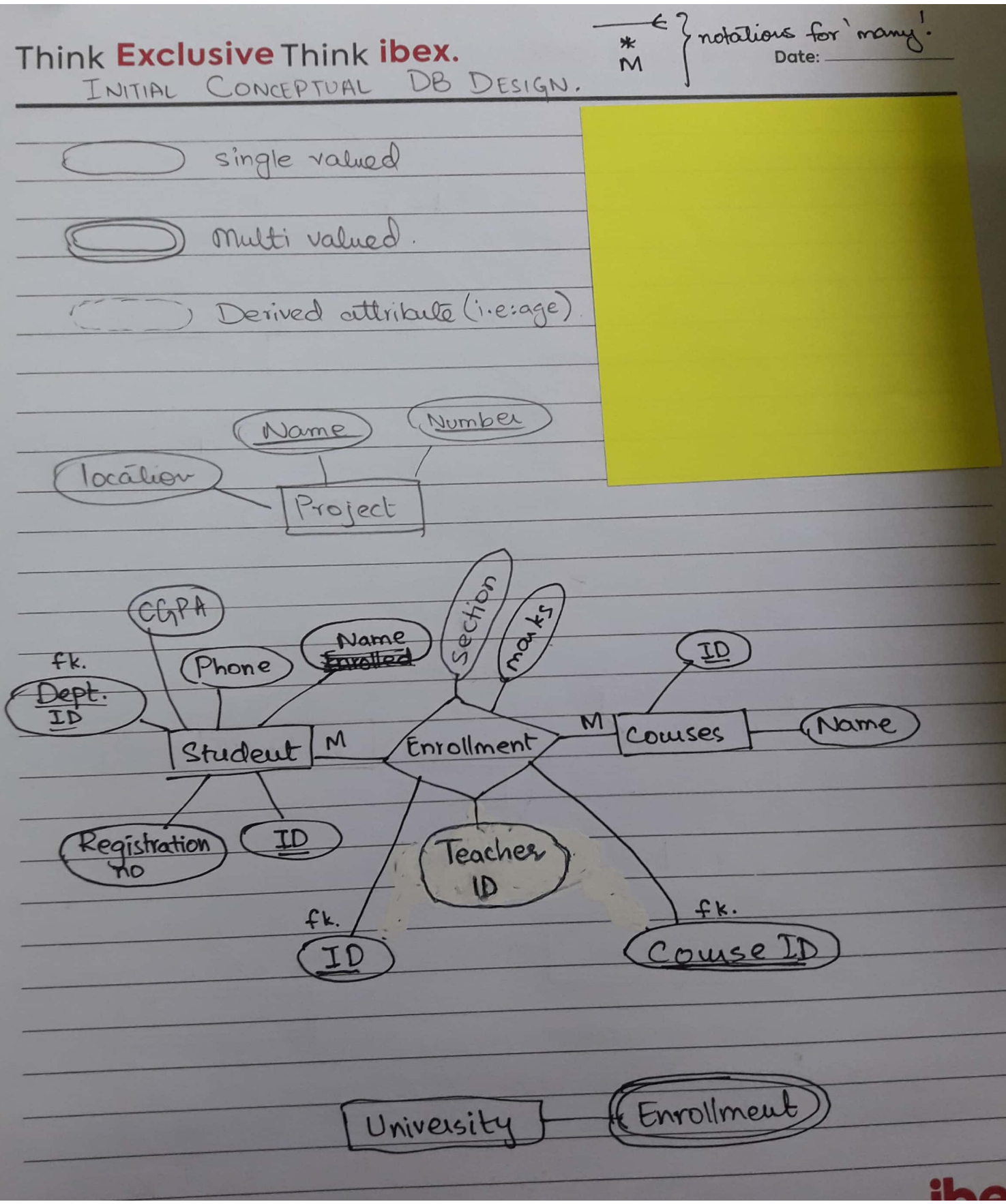
Note:

- Corelated subquery is ~~one~~ dependent on outer query attributes.

Think Exclusive Think ibex.

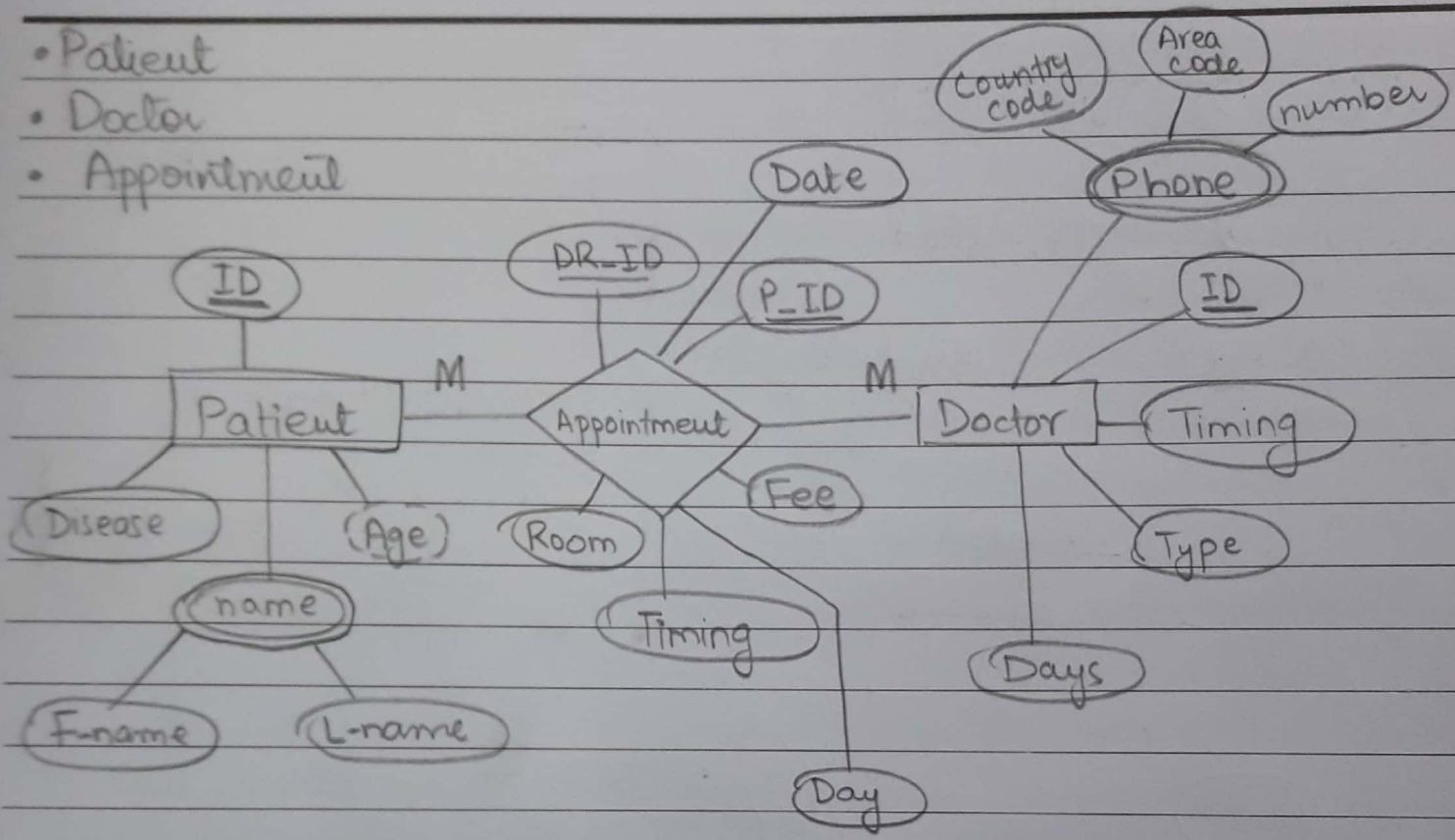
Date:

Initial Conceptual DB Design.



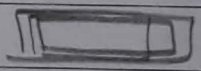
Date: 7th Oct, 19 Monday

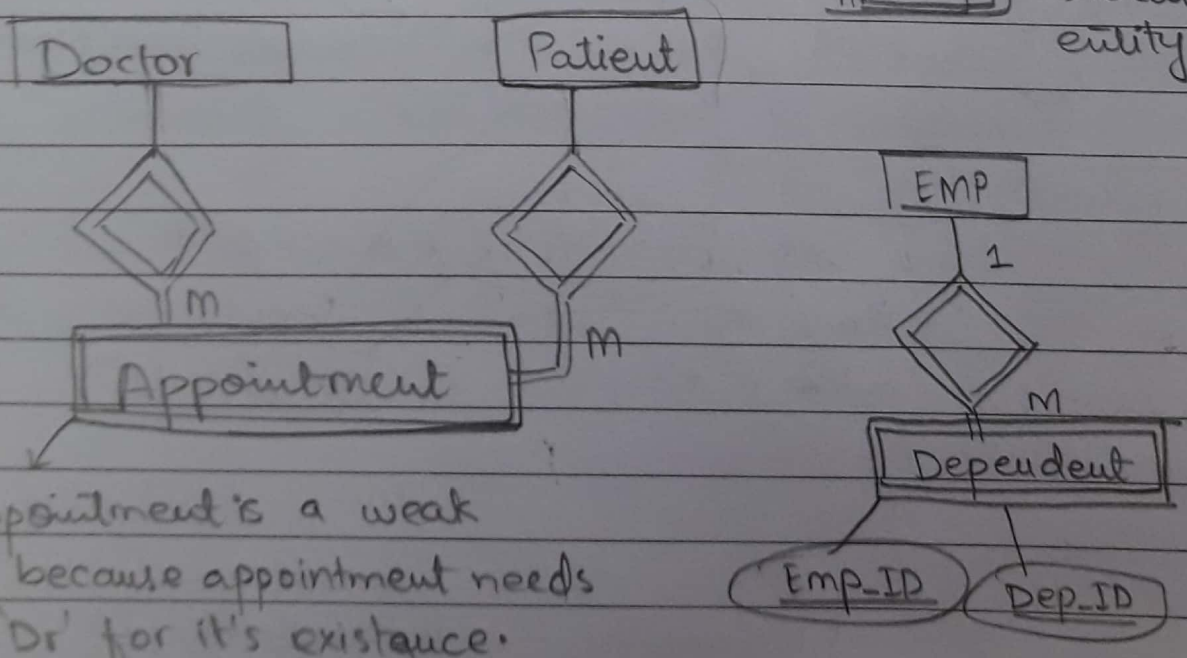
- Patient
- Doctor
- Appointment



Extended ERD:

Degree is 3, because of 3 entities


→ weak entity



Appointment is a weak entity because appointment needs entity 'Dr' for it's existance.

Here if we consider only 'Dr' and 'appointment', they have 1-M relation.

ibex.

* 1 depart has 1 HOD
  → cardinality ratio : 1 : 1

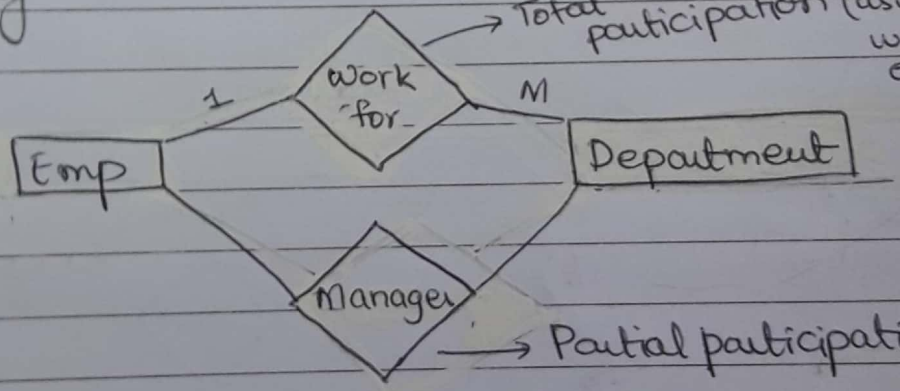
* Department : Employee
  → CR = 1 : N
  as depart has many employees.

1–M ← Peter Chan

← Crow feet notation (M, 1)

* ← Total participation is also called "existence dependency".

Total participation :
Atleast 1 association necessary



Every employee will work for a department which is "Total participation". Not every employee is manager in a department which is "Partial participation.

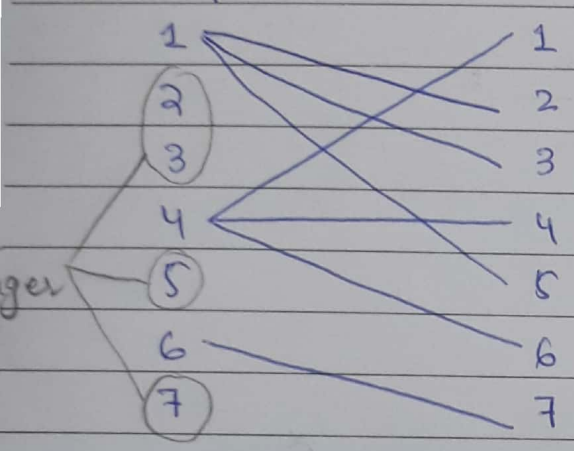
* Dependent to employee will have total participation

Hm ke bas ka nhi hai bhai sab 😅

palindrome.

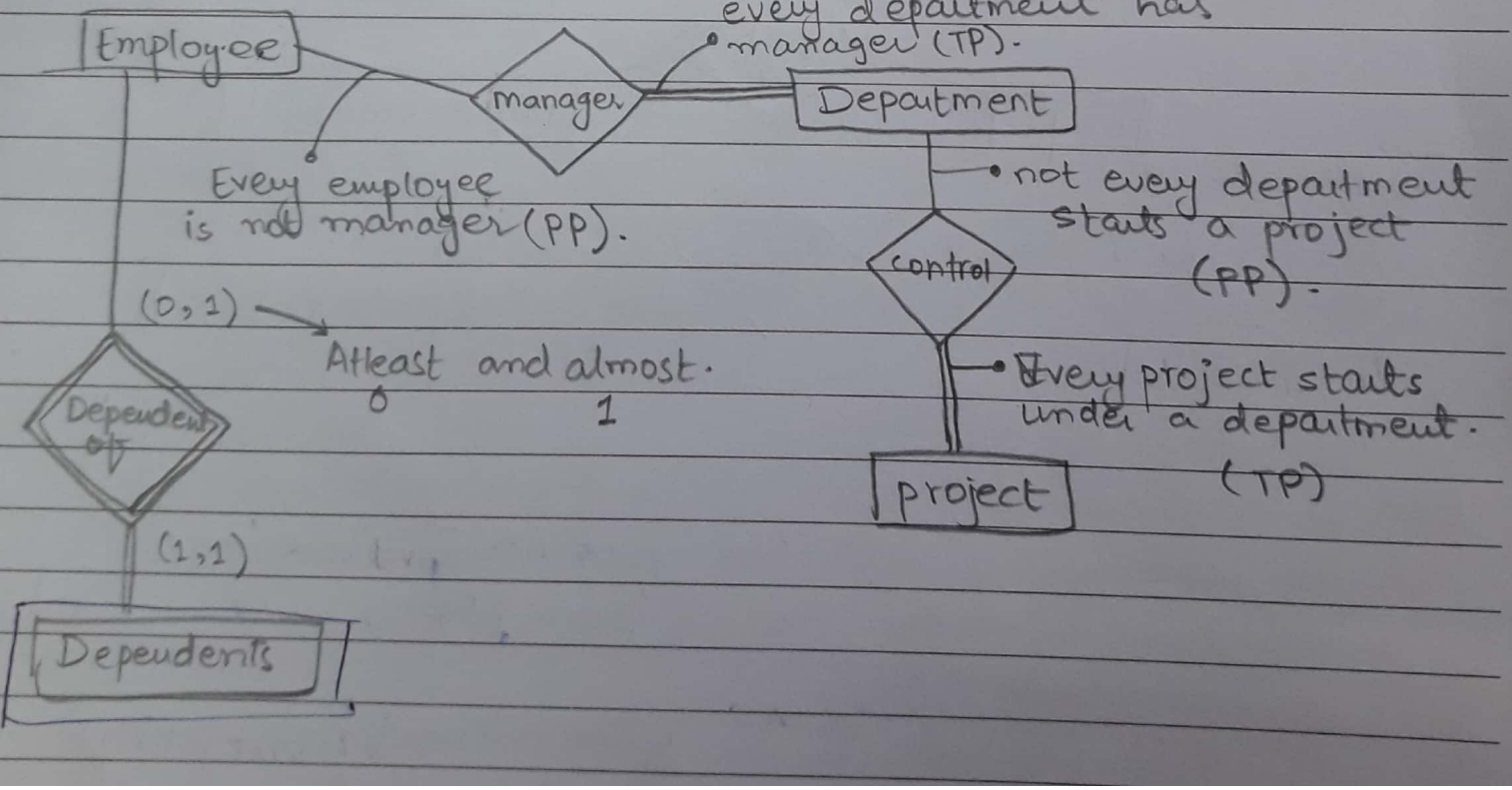
(partial participation) (Total participation)

Manager emp — managed by emp.



Cardinality Ratio → aik entity dusri entity se maximum kitne instances bna skti hai - i.e 1:1

Participation Relation → minimum kitne instances baneu ge.



Relational Algebra.

Select : σ → filter rows → selection is commutative.

$\sigma_{(selection\ criteria)}(R)$

(selection criteria) → where condition.

(R) → table to be selected.

Select is a unary operator that is applied on one table.

Project : π → filter columns.

list of attributes/columns that we want to show after selecting.

$\pi_{(A_1, A_2)}(R)$

* Select * from Emp where salary > 30,000

$Emp\_Sal \leftarrow \sigma_{(salary > 30,000)}(Emp)$ → with project → $\pi_{(name, age)}(\sigma_{(salary > 25000)}(EMP))$

Emp_Sal → A new table formed

Select * from Emp where (D_No = 4 and salary > 25,000) OR (D_No = 5 and salary > 30,000).

we can also write as:

$\sigma_{(DNO=4\ AND\ salary>25000)\ OR\ (DNO=5\ AND\ Sal=30,000)}$

$Dep\_4 \leftarrow \sigma_{(DNO=4)}(EMP)$

$Dep4\_Sal \leftarrow \sigma_{(salary>25,000)}(Dep\_4)$

$Dep\_5 \leftarrow \sigma_{(DNO=5)}(EMP)$

$Dep5\_Sal \leftarrow \sigma_{(salary>30000}(Dep\_5)$

Dep4_5 → Dep4_sal ∪ Dep5_Sal.

ibex.

Commutative property holds in σ

$$\sigma_{(cond1)}\left(\sigma_{(cond2)}(R)\right) = \sigma_{(cond2)}\left(\sigma_{(cond1)}(R)\right)$$

i.e

$$\sigma_{(sal>25000)}\left(\sigma_{(DNO=4)}(EMP)\right) = \sigma_{(DNO=4)}\left(\sigma_{(sal>25000)}(EMP)\right)$$

$A \cup B = B \cup A$ → in select case.

Select name, D_Birth from Employees where salary > 40,000 AND gender = 'male' OR gender = 'female'.

$$\pi_{(name, D\_Birth)}\left(\sigma_{(salary>40,000 \text{ AND } gender=male) \text{ OR } (gender=female)}(EMP)\right)$$

Select name, id from Emp where salary > (select avg(Sal) from Emp)

$$\pi_{(name,id)}\left(\sigma_{(salary > \sigma_{(avg(sal))}(EMP))}(EMP)\right)$$

— OR —

$$\pi_{(name,id)}\left(\sigma_{(salary > avg(sal))}(EMP)\right)$$

$\rho_s(R)$ ← relation. (source)

destination relation.

→ Renaming relation.

$\rho_{S(\text{attributes if they need to get renamed})}(R)$ ← relation name that is going to be changed.

new name of relation.

ρ Result (Emp_fname, Emp_Lname, Emp_Salary) $\leftarrow \Pi_{(Fname, Lname, Sal)}(DEP5\_Emp)$

OR

$RESULT_{(Emp\_Fname, Emp\_Lname, Emp\_Sal)} \leftarrow \Pi_{(Fname, Lname, Sal)}(DEP5\_EMP)$

"−" or difference is represented by 'except' keyword in queries.

* Difference is non-commutative.

Cricketer ← $\sigma_{(hobby = cricket)}(Student)$

non-Cricketers ← Student − Cricketers.

ibex.

Transaction → included in

⋈ → join

$R \bowtie_{(ssn = Essn)} S$ ⊗ $R \bowtie_{(ssn = Essn)} S$

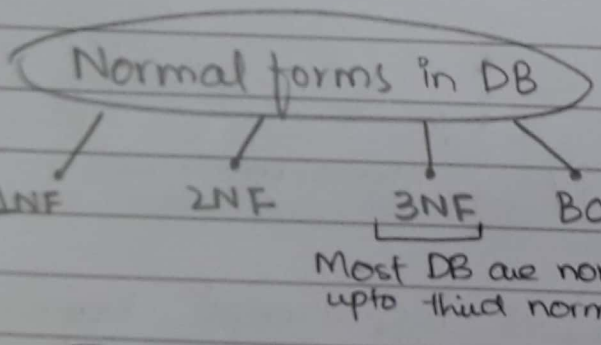
21st October, M.

Topics: Division query
Group functions.
Chap 14: Normalization
Anomalies.
Functional dependencies

Normalization:

Tells the goodness of our DB.

control the redundancy & inconsistency. Anomalies are removed



There are 3 types of anomalies:

1- Insert Anomalies.
2- Update Anomalies.
3- Delete Anomalies.

Advantages Of Normalization:

→ help produce cost-effective DB.
→ better " better security models.
→ Anomalies are remove.

SID	Hobbies
1	Cricket
	Tennis
	Football
2	Throwball.
	Hockey.

→ Update, Delete anomaly due to multivalued data

Assignment 2 :

Relational Algebra Queries from book + SQL form.

Topics for Mid 2 :

Advanced queries
Normalization
ERD
Relational Algebra (final)
Transaction (finals)

* 1NF (First normal form)

R(A,B,C,D) — A: PK, D: multivalued

S(A,B,C) S_1(A,D)

* 2NF (Second Normal form)

R(A,B,C,D) — (A,B): PK

→ Bring functionally dependent attributes into one table.

Functional Dependencies:

- Full dependency
- Partial dependency
- Transitive dependency.

A ← B.

if for each A there is exactly one value of B.

Full dependency.

composite pk should be a full pk.

2NF:

R(ABCD) ← partial dependency is found which needs to be removed in 2NF form.

S_1(A,C) — C is fully dependent on A

S_2(B,D) — D is fully dependent on B.

- Splitting should be lossless

UNF or ONF → tells that table is in unnormalized form.

Date:

paitial dependency:

A, X, C, n.

i.e out of A and X (keys), C is fully dependent on X only

Transitive dependency:

R (A, B, C)

DO → depend on.

$A \xrightarrow{DO} B \xrightarrow{DO} C$

$A \xrightarrow{DO} C$

3NF :

we have to remove transitive dependency

$X \longrightarrow Y \longrightarrow Z$

$X \longrightarrow Z$

we make [X Y] (fk.) and [Y Z] (PK)

3NF stricter form → Boyce Codd Normal form.

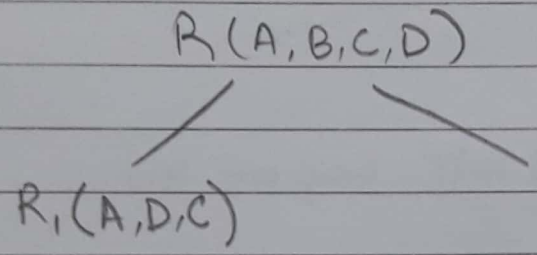
Boyce-Codd Normal Form : (BCNF) or (3.5 NF)

$A \longrightarrow BCD$

$BC \longrightarrow AD$

$D \longrightarrow B$. → it violates the condition as D is not key.

(D: nonkey)



Concept of normalization: One table must contain one idea, if a table has multiple values then it should be decomposed until it's based on 1 logic / idea.

1NF

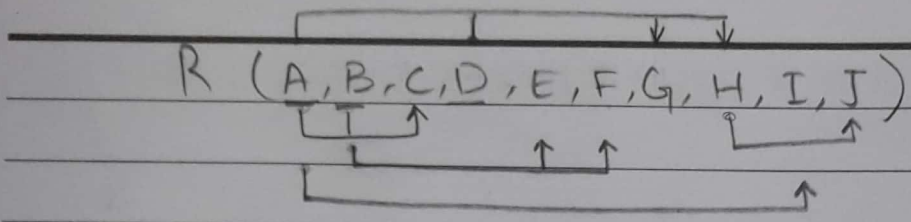
A	B	C
ID	Name	Couse
101	Eisha	OS, DS
102	Noor	CN, ITC

Decompose it to (A, B) and (A, C)

ID	name
101	Eisha
102	Noor

Couse–Student

ID	Couse
101	OS
101	DS



ABD ⟶ C, E, F, G, H, I

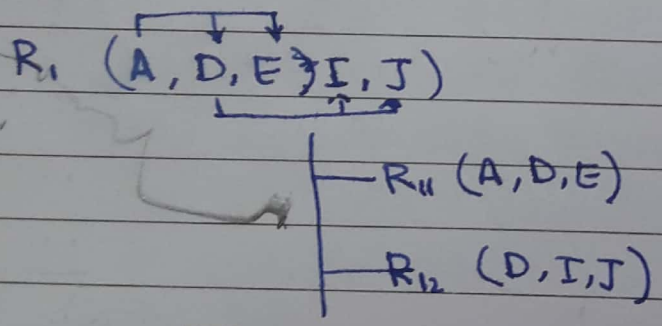
AB → C

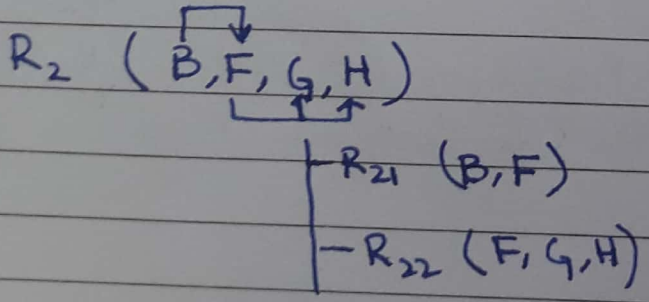
A → DE ✓ → A is candidate key of AB.

B → F ✓

F → GH

D → IJ





R3 (A, B, C)

Date:

Transaction

→ Multiple read and write operations ho sakte hain.

Concurrent system → multi user system

→ problems that occur:
- Atomicity
- consistency.

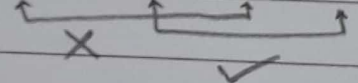
Need of concurrency control
1- Need of lost update problem
2- Temporary update problem → system will go to inconsistent state.
3- Unrepeatable read problem
4- Incorrect summary problem. (a change is interrupted due to another change i.e: update)

Note: Temporary update problem is also called "dirty read problem"

ibex.

* Schedules (histories) of transactions :

W, R X
R, W, R, W ✓
R, R, W ✓
W_2, R_1, R_2, W_1 → read should always come before write.



* Transactions :
Commit
Abort
Read
Write

Steps to read → 3 steps
" " write → 4 steps (include 3 of read and 1 of write).

$R_1 W_1 R_2 W_2$ → this is called serial execution.
(commit point)

Conflicting Operations :
Conditions (all should be true for a conflict to occur)
1) 2 transactions
2) Same
3) at least one write.

pair main aik read aur aik write karni hogi shared variable pe tou conflict aye ga.

1 2 3 4 5 6

Sa : $r_1(x)$ $r_2(x)$ $w_1(x)$ $r_1(y)$ $w_2(x)$ $w_1(y)$

Conflicting	Non-conflicting.
1–5	1–2
2–3	1–3
3–5	1–4
	2–5
	2–6.

conflict may occ
W–W
W–R
R–W

Transaction Schedules

- Serial → no interleaving.
- Non-serial → interleaving (conflicts occur).
  - Cascade less
  - Non-recoverable
  - Cascading rollback.

→ jis schedule main hai read se pehle commit hona must hai.

T_1	T_2
$W_1(X)$	
(first) → Commit	then ↓
	$R_2(X)$ (Read)
	$W_2(X)$
	commit
Abort	

T_1	T_2
$W_1(X)$	
	$R_2(X)$
	$W_2(X)$
	commit
Abort	

→ It was not committed yet so it's non-recoverab (pehle isko commit karna chahiye tha).

Cascading rollback → when one transaction cause failure and roll back has to be performed.

1 2 3 4 5 6

→ $r_1(X), w_1(X), r_2(X), r_1(Y), w_2(X), w_1(Y), c_1, c_2$

Non-conflicting	Conflicting
1-2	1-5
1-3	2-3
1-4	2-5
3-4	
3-5	

In order to remove non-recoverable condition, move c_1 before $r_2(X)$ which is '3'

Book 20.5 and 20.4.

Date:

Schedules of transactions → (Study their graphs as well)

- Serial (no-conflicts)
- Non-serial (conflicts occur).
  - Serializable
    - Conflict based serialization
    - View based serialization
  - non-serializable
    - Recoverable
      - Cascading Schedule — 2nd transaction can't read until 1st commits itself.
      - Cascading rollback scheduling
      - Strict scheduling → 2nd transaction cannot read or write before 1st commits it. → Every strict schedules is cascadeless.
    - Non-recoverable.

* If you increase concurrency, recovery become more complex. There is a tradeoff b/w recovery and concurrency.

* cascadeless schedules are recoverable and we can do cascading to strict scheduling on it.

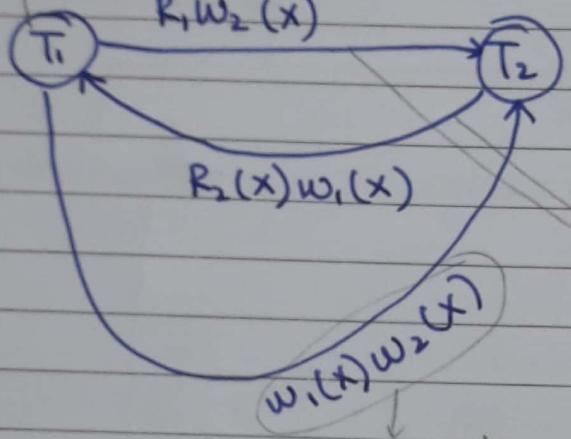
Precedence Graph $(R_1(x), R_2(x), W_2(x), R_1(y), W_2(x), W_1(y)$

Conflicts :
$R_1 W_2(x)$
$R_2(x) W_1(x)$
$W_1(x) W_2(x)$

Schedule

Transaction node

(x) or X



cycle means, the conflict is non-serializable.

conflict (we can allow write it as X)

variable

View Equivalence → ye us DBMS ke liye hoti hai jo 'blind write' allow karte hain-

$S_1 \longrightarrow S$

initial read T_1
Final write T_2
Update x by T

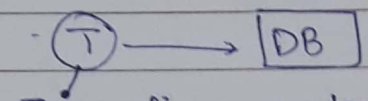
Read k baghair write karwaen tou blind write hota hai.

ibex.

★ ACID Properties Of Transaction :

1- Atomicity

2- Consistency → DB should remain consistent before and after the transaction.



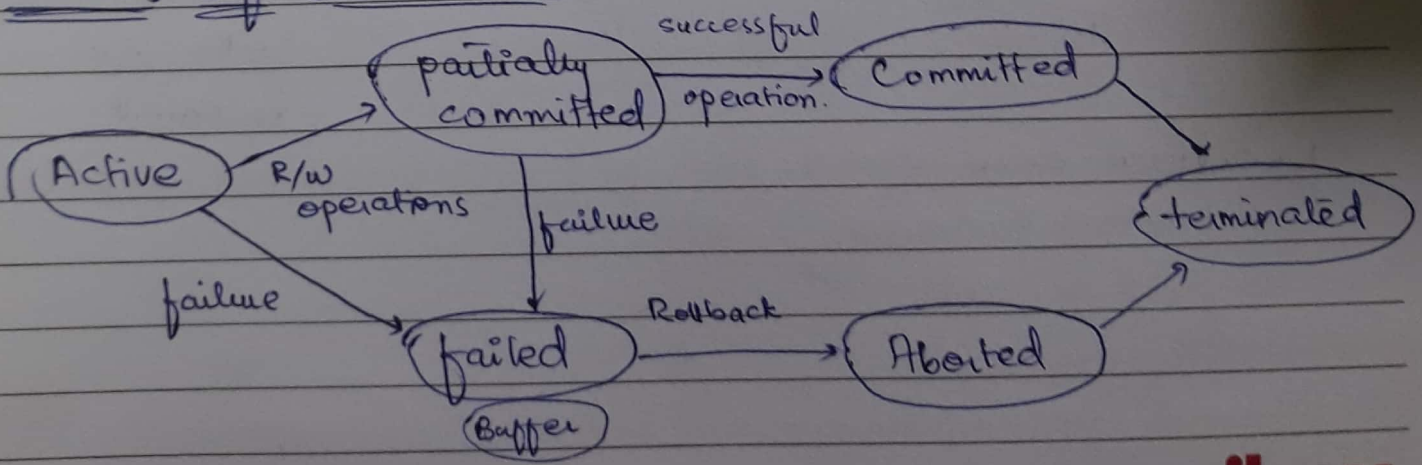
3- Isolation → There should be a logical isolation b/w the transactions.

Note :
→ Concurrency control unit is a module that takes care of isolation.

4- Duability → The changes we have made must reside ~~duab~~ with duability in the system -

Note
→ Recovery management component is responsible to maintain duability.

★ States Of Transactions



ibex.

* Advantages Of Concurrency :

1) Waiting time ↓
2) Response time ↓
3) Resource utilization ↑
4) Efficiency ↑

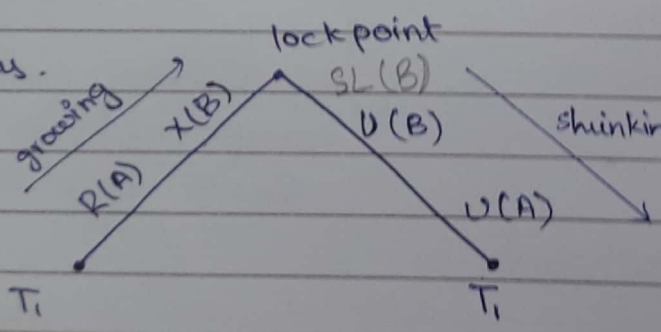
TBS:
- Shared lock (SL)
- Exclusive lock (XL)

Two Phase Locking Protocols

R → S-L ⇒ R & W → X-L

Basic
Conservative
Strict
Rigorous.

2 phase → Expanding (Growing)
→ Shrinking (X-L ⇒ SL)



Deadlock may occur if shared locks are not released.

* Deadlock can arise in following situations:
  - Mutual Exclusion.
  - Hold & Wait
  - Circular Wait
  - No preemption.

Time Stamp Protocol :

* 2PL rigorous -

- Basic -
- Strict -
- Thomas Write Rule -