Database Management System

Drawbacks of File Systems

1. Data Redudancy and Inconsistency
2. Difficulty to access data
3. Data isolation
4. Security
5. Integrity
6. Atomicity of update
7. Concurrent Access by multiple users

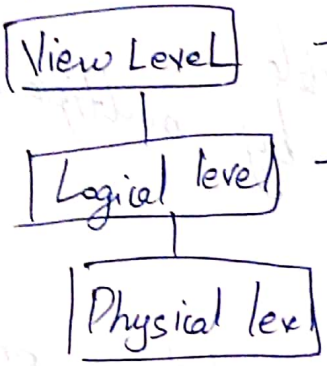
DBMS Vs File System

1. For DBMS minimal data redudancy problem
   In file system data redudancy problem exists
2. Data inconsistency does not exist
   File system data inconsistency exists

DBMS	File system
3. Accessing data base is easier	3. Accessing is comparatively difficult
4. The problem of data isolation is not found in database	4. Data is scattered in various files and files may be of different format so data isolation exists
5. Transactions like insert, delete, view, update are all possible	5. Transactions like these are not possible.

6. Concurrent access & recovery is possible in DB	6. No concurrent access & recovery
7. Security of data	7. Security of data is not good.
8. A DBMS administrator stores the relationship in form of structral tables	8. A file manager store all relationship in directories

Level of abstraction :-



→ the relevant information required by user

→ It describes the data stored and what relationships exist among them

→ how the data are actually stored.

Data base Schema :-

↓

Skeleton structre that represents the logical view of entire database

Data independence :-

↓

Capacity to chage information/scheme in one level of a database system without having to chage the schema at the next highest level.

(1) Physical data independence

(2) Logical data independence.

Instance :-

The actual content of data in database is called an database Instance.

How the data is stored

Physical level

Name string length 25 offset

USN integer 9 decimal digits offset 25

Logical level

Name string

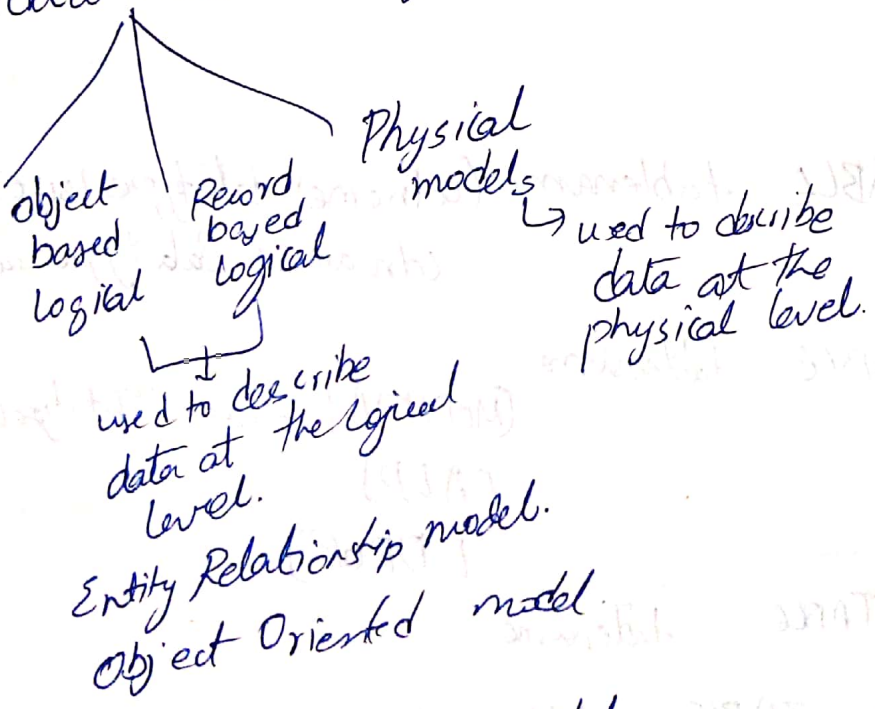
USN integer primary key

External level

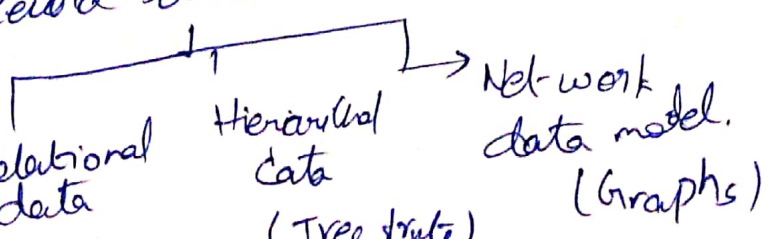
Name stry

USN integer

Data Model underlying the structure of DB



Record based model



Database languages:

DB provides DDL [Data Definition Language] - to the database schema

DML [Data Manipulation Language] - to data base queues & updates

TCL [Transaction control language]

DCL [Data Control language]

DDL commands	DML commands	TCL
CREATE	select	Commit
ALTER	Insert	Rollback
DROP	update	Savepoint
TRUNCATE	Delete	

DCL

GRANT

REVOKE

CREATE TABLE tablename (colname1 datatype (size
colname2 datatype(si

ALTER TABLE tablename
[MODIFY] / colname2 new datatype
[ADD]
[DROP]

DROP TABLE tablename

TRUNCATE TABLE tablename

INSERT INTO tablename( ) values( )

SELECT * FROM tablename;
SELECT stdNo,stdName from tablename;

Update

UPDATE tablename set colname = value2
where colname = value1;

DELETE

Delete from tablename where colname=value

Employee Database :-

CREATE TABLE Emp (Eid Varchar(5), ename char(5), dno Varchar(5), designation char(15))

CREATE TABLE Dept (dno varchar(5), dname char(10), loc char(20))

INSERT INTO Emp("12345", "Jana", "5", "Manger");

ALTER TABLE emp modify(designation char(5),
jobprofile char(5))

① Retrive details of all employees, ② employee id of all employee
Assign phoneno. to corresponding dept.
List name of dependant who is to pose to employee.

select * from emp;
select eid from emp;
alter table Dept add(phno number(10));
select name from dependat where relationship = "spouse";

Super keys ③ Foreign key
Candidate key

create table dependent(name, , relationship , eid)

NOT NULL constraints
↳ rules that are enforced on a table
can be either column level or table level

Most commonly used constraints

i) NOT NULL
ii) DEFAULT
iii) UNIQUE
iv) FOREIGN KEY
v) CHECK → ensures that all values in the column satisfies certain condition.

Alter table <tablename>
ADD CONSTRAINT
<constraintname>
Primary key (id)

Employee | ENAME | SSN | BDATE | PLACE | SALARY | SUPERSSN | DNO |

Department | DNAME | DNUMBER | MGRSSN | MGRSTARTDATE |

alter table <table name>
modify <column name>
<datatype>
default
<value>

DEPLOCATION | DNUMBER | DLOCATION |

PROJECT | PNAME | PNUMBER | PLOCATION | DNUM |

WORKSON | ESSN | PNO | HOURS |

DEPARTMENTS DEPENDENT | ESSN | | RELATION |

① Select details of employee with dNO-5
② Retrive all the details of all employee in the research department.

3. Add a new column to the existing table (employee)

PHNO

4. Rename PHON.NO to PH_NO.

5. Drop column PH_NO.

6. Delete from

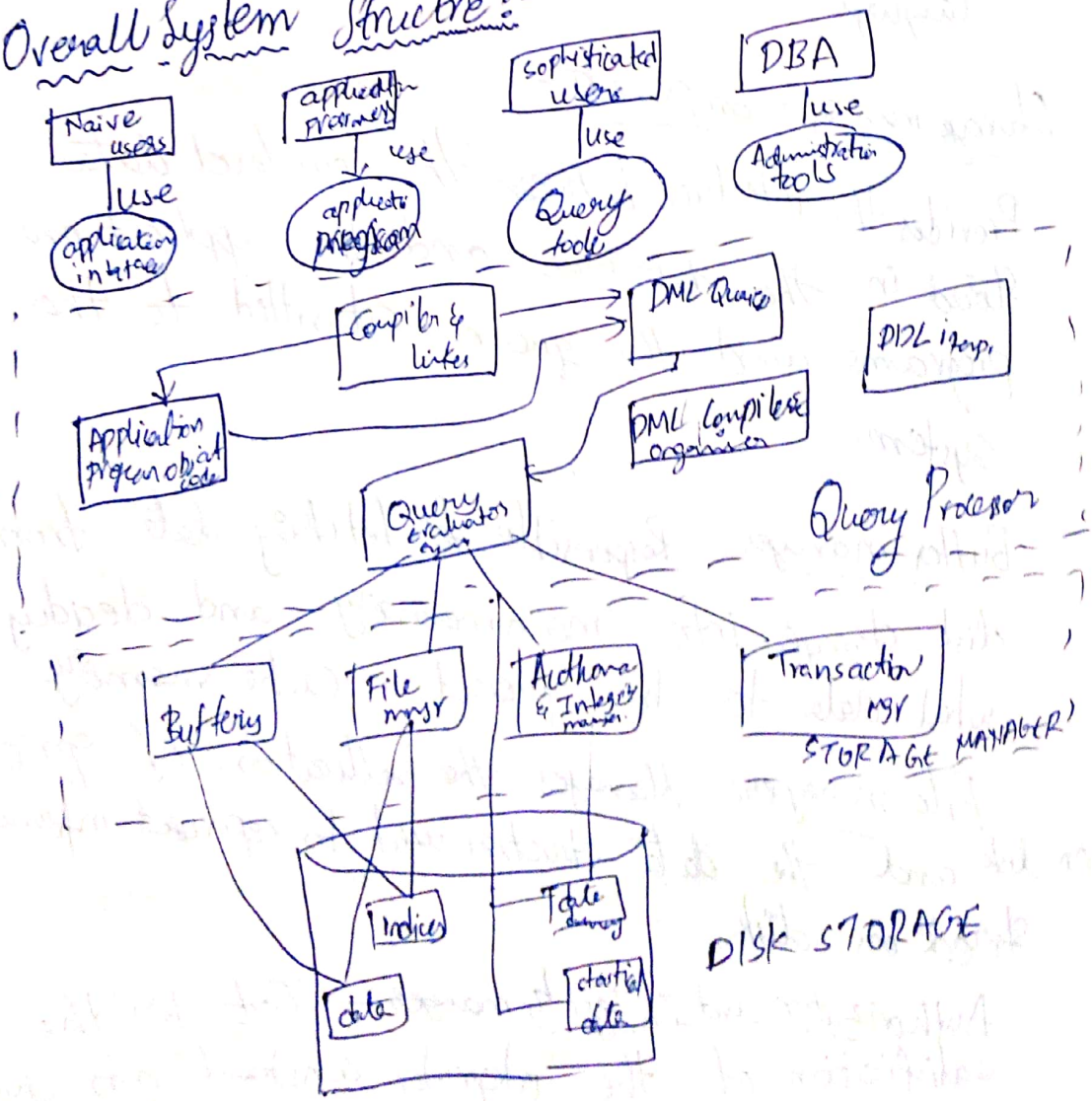
7. Chaye the details of project no: 10

8. For every project located in Bayarbre the project number the controlling department and dept manager name and BDATE

9. For each employee retrive the employee's name & name of his/her supervisor.

10. Update the salary of employee by 10 %.

Overall System Structure:-



Users:-

Naive users → Interact with system by invoking one of the permanent application program that has been written previously.

Sophisticated users → Interact with system through database languages.

Query languages.

Specalized users → Sophisticated users who write specalized data base application that do not fit into a Traditional data processing framework

Application programmers → Computer professionals who interact with system through DML calls which are embedded in a program written in a host language.

Storage manager components:-

Provides the interface between the low level data stored in the database and the application programs and the queries submitted to the system.

Buffer manager :- Responsible for fetching data from disk storage into main memory and deciding what data to be present in cache memory.

File manager:- Manages the allocation of space on disk and the data structures used to represent information stored on disk.

Authorization and Integrity manager :- Test for the satisfiation of the integrity constraint and check

the authority of user to access data

Transaction mayar: It ensures that the database remains in a consistent state despite system failures and the concurrent transactions execution proceeds without conflicting.

Query Processor Components:-

DML Compiler:- Translates DML statements to low level instructions that the Query evaluation engine

DML Interpreter: Interprets all the DML statements & records them in a set of tables containing meta data.

Query Evaluation Engine:- Execute low level instructions generated by DML compiler.

Embedded DML Precompiler: It converts DML statements in an application program to normal procedural language in host language.

The storage manager

Data structures

Filemanager:

Data Files: where the database is stored

Data dictionary: it stores meta data about the structre of the database

indices: Provides fast access to data items that hold particular data value.

statistical data: which stores statistical data/information in data.

Show the resulting salaries if every employee working on product X project is given 10%

Select ename, 1.1*salary from Emp, Workson, P
where PNO = Pnumber and Pname

Logical

AND, OR, NOT

Select name, place from employee where
Place = "Bombay" or place = "Delhi"

10000 to 20000

Select * from employee where
salary between 10000 and 20000

Pattern matching
using LIKE % → one or more _ → one character
list the employee information where first Ja

⇒ Select * from employee where
ename LIKE ('Ja%');

Select * from employee where
ename LIKE ('_e%'

or
ename LIKE ('_h%')

IN and NOT IN Predicate
list empname, place whose names are A, B, C, D

Select ename, place from employee
where ename IN ('A', 'B', 'C', 'D')

Oracle Functions

fnname (arg1, arg2, ..)

SUM, MAX, MIN, COUNT, AVG

select sum(salary) from employee;
Find sum of salaries of all employee of all employee of research department, max salary,
Select sum(salary), max(salary), min(salary) from employee, department from dNO = DNO and DNAME = 'research'.

Select count(*) from employee;

ABS(), POWER(m,n), Round(n,m), LPAD(char1, n, char2), RPAD(char1, n, char2);

SQRT()

LOWER(char)

UPPER(char)
INITCAP(char)
LENGTH(char)
SUBSTR(char, n, c);

ER Model

Entity Relationship model:

- Entity sets
- Relationship sets
- Attributes

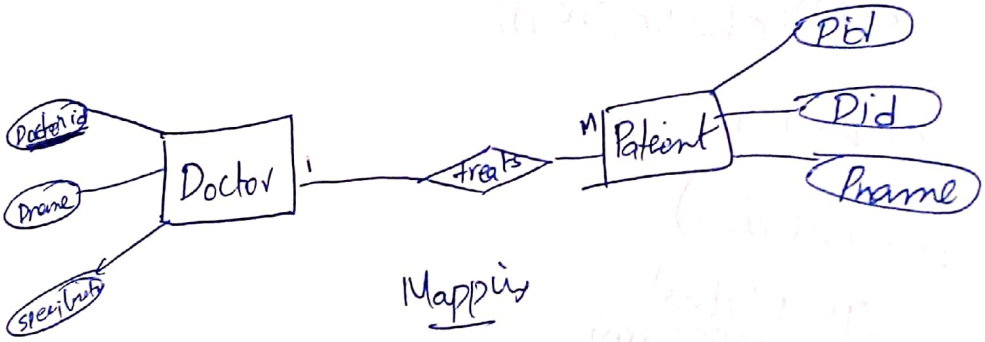
Type of Attributes

1) Simple vs composite
2) Singlevalued vs Multivalued
3) stored vs Derived
4) NULL
5) complex (combination of composite & multi

Person with more than one residence
↳ more than one

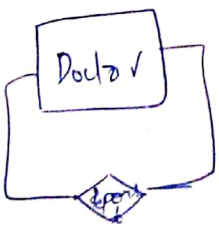
{Addressphone ({Phone (Area, Ph, ng}

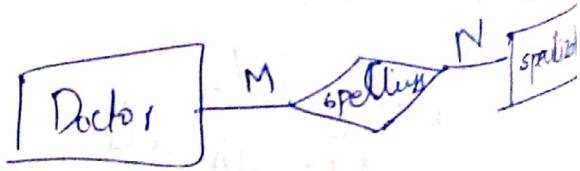
Addres (Street address (Steelnone, City,

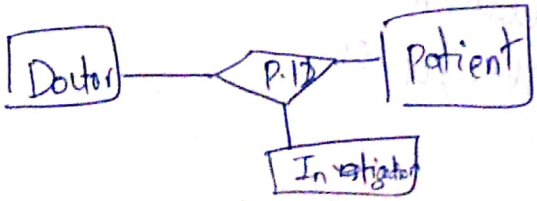


Unary
Binary
Teanary.

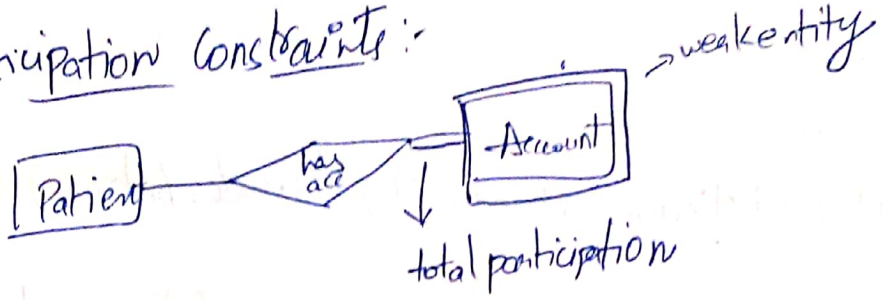
Mapping
1-1
1-M
M-M



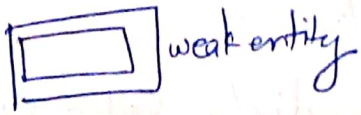


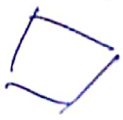


Participation Constraints :




Entity


weak entity

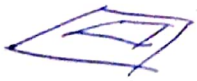

Relationship


Attribute


- multivalued attribute


Derived attribute


- Composite attribute


- Identifying relationship


key attributes.

→ A weak entity is an existence dependent i.e it cannot exist without the entity with which it has its relationship

→ It has a primary key which is totally or partially derived from parent entity

Grouping :-

clientMaster (client no, Name, city, Pincode, balance)

Product-Master (Pdt no, description, profit, unit, qty on hand, SP, CP)

Sales-master (sal-no, salname, salamt)

Sales-order (s-ordno, S-ord date, client no, Sal no, station)

Sales order details (s-ordno, Pdt no, qty-ordered, pdt-

Select pdtno, sum (Qty-ordered) QP from sales order details, Group by pdtno having pdtno col of pdtno

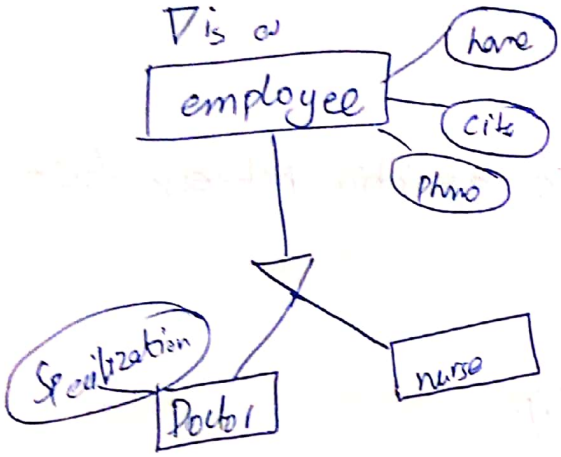
To_char
- number conversion
- date conversion

Display order information sordno, clientno, sorddate for all the orders placed by the client in the order of date

Select pdtno, description, sum (qty ordered) from Product master, sales order where P.pdtno = S.P dtno group by pdtno.

Extended ER features

specialization & generalization



1. Specialization
2. Generalization
3. Low level entity
4. High level entity
5. Attribute inheritance
6. Aggregation

Constraints on generalization & specialization

1. Attribute vs user defined (based on value)
2. disjoint vs overlapping (only one low-level / more than one)
3. total vs partial relates whether or not entities may belong to more than one low level entity set within a generalization

Constraints on Extended ER features

lower level entity set

- Disjoint → entity belongs to no more than one LL entity.
- overlapping

Completeness constraint:-

whether or not an entity in the higher level entity set must belong to atleast one of the lower level entity set

- total → each higher level belong to a lower level entity set
- partial → some higher level may not belong to lower level entity set.

Conversion of ER-model to relation model :-

1. mapping regular entities, composite attributes and multivalued attributes
2. mapping weak entity
3. mapping binary relationship existing between two entities
4. mapping associativity
5. mapping unary relationship
6. mapping supertype and subtype attributes.

Relational Algebra :-

Procedural Query language

→ instance of relation

→ Basic operations

- Select
- Project
- Rename
- Union
- Set difference
- Cartesian product

Division
Intersection
join operation
Assignment

Select (σ)

$\sigma_{p(R)}$

$\sigma_{name = 'ram'}$ (Student)

Project (Π)

$\Pi_{c_1, c_2}(R)$

Rename (ρ) :-

$\rho_x(R)$

Union (∪)

→ r ∪ s

→ r, s should have same no of attributes

→ Attributes domains must be compatible.

$\Pi_{authors}(Books) \cup \Pi_{authors}(Articles)$

Set difference (−) :- (r − s)

$\Pi_{authid}(Books) - \Pi_{authid}(Articles)$

Cartesian Product (r × s) :-

$\sigma_{author='ABC'}(Books \times Articles)$

$(Books)$

subject = 'database' and price = '200'

Select details of employee where emp = 2

$\sigma_{emp=2}(employee)$

To select tuples for all employee who either work in dept = 4 and makeover ₹25000 per year or dept = 5 and makeover ₹30000

$\Pi_{name}(\sigma_{dept=4 \text{ and } makeover>25000}(employee)) \text{ or } (dept=5 \text{ and } makeover \geq 30000)$

$\Pi_{name, first, last, salary}(employee)$

$\Pi_{first,last,salary}(\leftarrow_{dept=5}(employee))$

$\rho_{EMP}(employee)$

↳ new relation name

$\rho_{enolno}(employee)$

(↳ old column name
→ new column name)

The rules for performing union, intersection

(i) Arity of both relations should be same

(ii) Domain of i^{th} attribute of first relation should be same as that of second relation

$\Pi_{fid}(faculty) \cup \Pi_{sid}(student)$

List the names shared between employee and dependent

$\Pi_{name}(employee) \cap \Pi_{name}(dependent)$

List the name of employee which is not name by any dependent

$\Pi_{name}(employee) - \Pi_{name}(dependent)$

List the name, gender irrespective whether he is employee or dependent

$\Pi_{name,gender}(employee) \cup \Pi_{name,gender}(dependent)$

Retrive the social security number (SSN) of all employees who either work dept = 5 or directly supervise an employee who dept = 5:

$r \leftarrow \Pi_{superssn}(\leftarrow_{dept=5}(employee))$

$\Pi_{ssn}(\leftarrow_{dept=5}(employee)) \cup r$

Retrive for each female employee a list of names and her dependent

$$\leftarrow \left(\leftarrow (\text{employee}) \times \leftarrow_{\text{gender='F' and employee.id=dependent.id}} \text{dependent} \right)$$

Insert:-

$r \leftarrow r \cup E$

$Emp \leftarrow Emp \cup \{("add", 1010, 10000)\}$

Delete

$r \leftarrow r - E$

↳ relational algebraic expression.

$stud \leftarrow stud - \leftarrow_{name = abc} (stud)$

Update

$\Pi_{f_1, f_2, \ldots f_n} (\leftarrow_P (R))$

Division :- $\div$

↘ for all

Divisor rule

① arity of numerator > arity of denominator

② denominator is a subset of numerator

③ o/p have the attributes in numerator but not in denominator.

Join

Natural Join $\bowtie$

Theta Join $\bowtie_c$

Equi join $\bowtie_{a1=a2}$

Outer join

⇒ left outerjoin ⟕

right outerjoin ⟖

full outerjoin ⟗

Emp

name	street	City
a	M	Ch
b	N	P
c	O	Ba

Full

name	branch	Salary
a		15000
b		15000
c		16000

Delete the details of whose name is smith $\leftarrow_{name=smith}(Employee)$

To modify the value of a salary of a employee whose name is 'Megha' $\Pi(\leftarrow_{name=Megha}(employee))$;

Retrive name of employe who works on all the projects worked of smith. $\Pi_{name}(\leftarrow_{projects} \cap \Pi_{name}(\leftarrow_{smith}))$

List all employee names and also the departments they manage if they happen to manage. (left outerjoin

$Temp \leftarrow Emplage \bowtie Deptno$

$\Pi_{name}(Temp)$ $SSN=MGRSSN$

Retrive name & city of all employe who works for the research dept. $\Pi_{name,city}$

For every project located in standard list PNo, control dept no, dept manager. $\Pi_{PNo, controldept, deptmanager}$

Find names of employee who works on all projects controlled by dept 5. $\Pi_{name}(\leftarrow_{dept=5}(Projects))$

Make a list of pno that projects that involves whose employe whose last name is smith or as a manager of the dept. $\Pi_{pno}(\sigma_{lname='Smith'}) \cup \Pi_{pno}$

Retrives name of employee who have no dependence

List of name of manager who have atleast one dependent

Subquery:-

↓

Set membership

Set comparison

Set cardinality

set membership test → testing tuples for mem

↓ ↓

in not in → test for the absence of set membership

test for set membership

Find all customers

select cname from borrower where cname in(select cname from depositer)

find all customers who have both account & a loan at the perryridge branch.

Select cname from borrower, loan l where b.loanno=l.loanno and branchname = 'perry ridge' and (branchname, custname) in select branchname, custname from depositer, account where d.accno = a.accno.

Set Comparison

>some
<some
>=some
<> some

find the name of all branches that have assets greater than those of atleast one branch located in 'b...

Select bname from branch which assets >some (select assets from branch where br.city='brooklyn';

Find the names of all branches that have an asset value greater than that of each branch in brooklyn

Find those branches that has highest average balance

Select bname from account where group by branchname having avg(balance) >all (select avg(balance) from account group by branchname).

Exists
not exist

Find all customers who have both an account & a loan at

aggregate fns:-

$\mathcal{G}_{sum(salary)}(Employee)$

$\mathcal{G}_{count(DNo)}(Employee)$

G (Employee)
Dno Sum(salary)

List the employees who salary is same as the salary A or B
List the result descending order

select ename from emp where salary in (select salary from emp where ename = 'A' or ename = 'B' order by salary desc;

select dept, avg(salary) from dept where

Display empid, mangerid, firstname, lastname who manage other side

selet empid, ~~superssn~~, firstname, lastname from emp where empid in (select sapperssn from emp)

Make a list of all project numbes that involve an employee whose name is smith either as a worker or a manger.

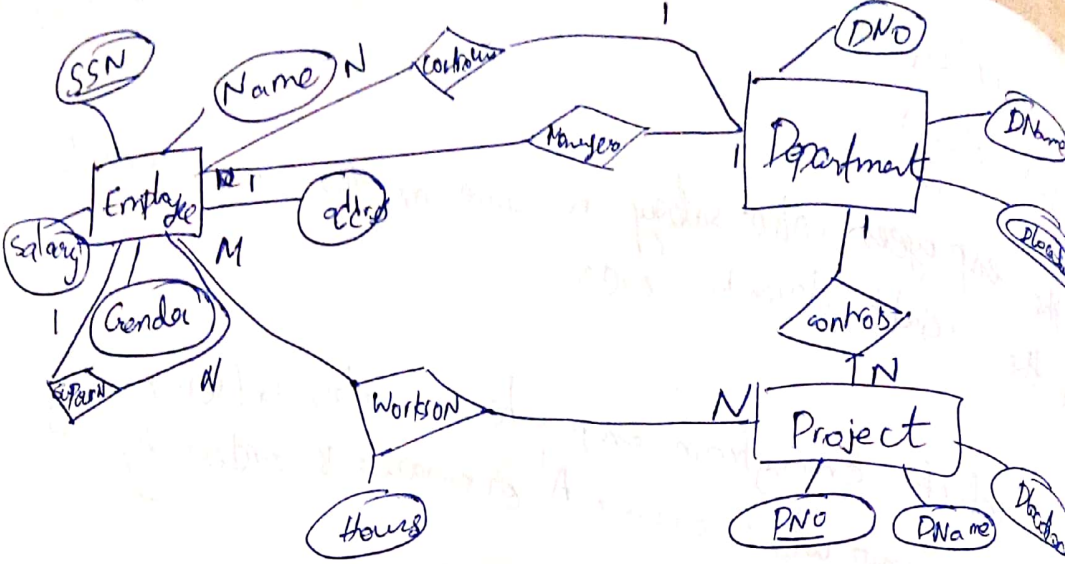
Employee(SSN, Name, address, Gender, Salary, SuperSSN, DNo);

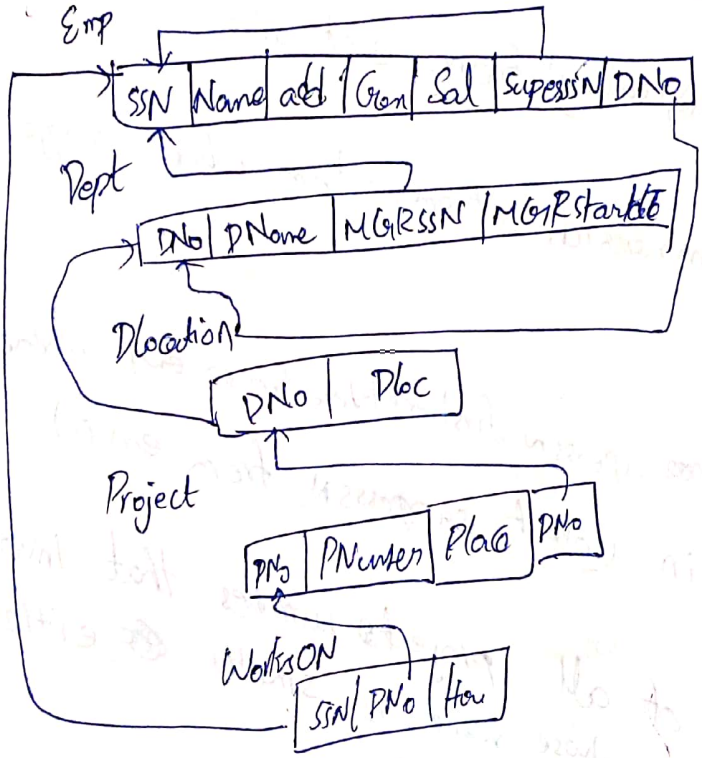
Department(DNo, DName, MGRSSN, MGRstartdate)

DLocation(DNo, DLoc)

Project(PNo, PName, Plocation, DNo)

WorksOn (SSN, PNo, Hours)

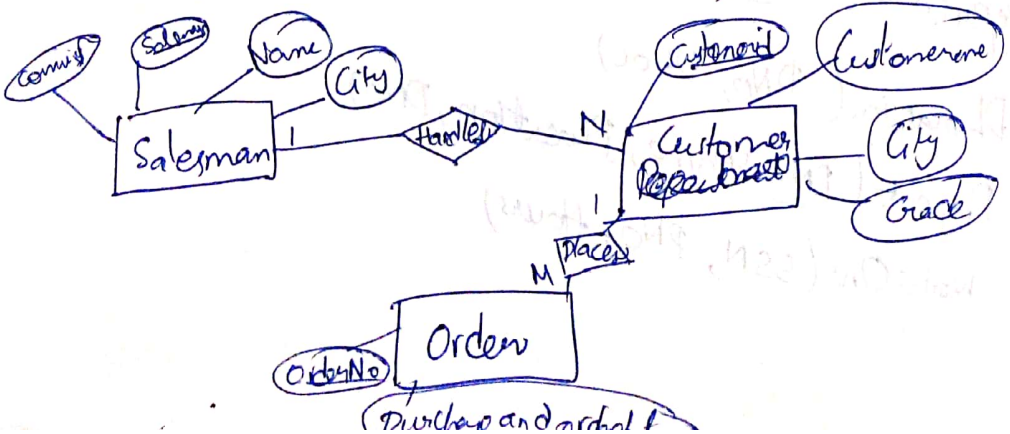




Salesman (salesmanid, Name, city, commission)

Customer (Custnerid, Custname, City, Grade, Salesmanid)

Order (OrdNo, purchase amount, order date, Customerid, salesmanid)



Functional dependency:-

The attribute of a table is said to depend on each other when an attribute uniquely identifies another attributes of same table.

col A uniquely identifies col B

col A → col B → dependent

↓

determinant

Stud_id	Stu_name
Each value of A is associated precisely with B value	

Advantage

It avoids data redudancy where sayig data should not be repeated at multiple location in same database. It mains quality of data base it allows clearly defined meanings and constraints of data bae. It helps in identifying bad designs.

Axiom rules:-

Armstrongs Axiom

Primary rules:-

Rule 1: Reflexivity — A → B (A holds B)

2: Augmentation — A holds B & C ⇒ AC holds BC (AC→BC)

3: Transitivity. — A holds B & B holds C ⇒ A → C (A holds C)

Secondary Rules:

1 → Union $A \to B$ & $A \to C$

$A \to BC$

2. Decomposition

$A \to BC$, $A \to B$ then $A \to C$

3. pseudo-transitivity

$A \to B$ $BC \to D$ then $AC \to D$

Composition:-

$X \to Y$ and $Z \to W$

then $XZ \to YW$

$E = P\,Q, R, S, T, U$

$P \to Q$ $P \to R$

$QR \to S$ $Q \to T$

$QR \to U$ $PR \to U$

Calculate $P \to T$

$P \to Q$, $Q \to T$ $P \to T$ (Transitivity

$PR \to S$

$P \to Q$ $QR \to S$ $PR \to S$ (pseudo transitivity

$QR \to S$, $QR \to U$

$QR \to SU$ (union rule)

PR → SU

P → Q QR → S
PQR → S } pseudo transitivity

P12 → U

PR → SU (union)

R (street cityzip)

F (city street → zip zip → city

Show street zip → street zip city

BA → C
C → B
BA → B
AC → ABE
city, street → zip city
zip → city
street zip → zip city
street zip → city
zip → zip
street → city
zip → zip city

1) zip → city

street zip → street city augmentation

city student → zip

city street → city street zip

street zip → city street zip

P = (ABCDEGHIJ)

F = { AB → E AG → J BE → I E → G GI → H }

EB

AB → I ✓

GI → H

AB → G

AB → G CAB → H

AB → H

AB → GH

AB → I
ABG → H
G → AI
ABB → G
AB → G

AB → E , E → G (AB → G)
AB → E EB → I (AB → I)

Closure set of FD's

~~R = {A, ...~~

Closure set of is set of all functional dependencies the include F as well as all dependencies t can be inferred from F.

FD = { A→B A→C CG→H CG→I B→

F⁺ { A→B, A→C, CG→H, CG→I, B→H,

A→H, AC→BC, CG→HI, AG→H, AG→

find candidate keys

R(A,B,C)

A→B, B→C

A⁺ = {A, B, C}

B⁺ = {B, C}

C⁺ = {C}

R(A, B, C, D, E, F)

A→BC C→B D→E E→D

A⁺ = {A, B, C}

B⁺ = {B}

C⁺ = {B, C}

D⁺ = {D, E} E⁺ = {D, E}

$AB^+ \rightarrow \{ABC\}$

$AC^+ \rightarrow \{ABC\}$

$AD^+ \rightarrow \{ABCDE\}$

$AE^+ \rightarrow \{ABCDE\}$

$R = \{A, B, C, D, E, F\}$

$A \rightarrow C \quad C \rightarrow D \quad D \rightarrow B \quad E \rightarrow F$

$AE = \{A, B, C, D, E, F\}$

$R\ (A, B, C, D, E, F, G, H)$

$CH \rightarrow G \quad A \rightarrow BC \quad B \rightarrow CGH \quad E \rightarrow A \quad F \rightarrow EG$

$D^+ = \{D\}$

$A \rightarrow \{A, B, C\}$

$B \rightarrow \{B, C, F, H, E, G\}$

$C \rightarrow \{C\}$

$D \rightarrow \{D\}$

$E \rightarrow \{A, E, B, C\}$

$F \rightarrow \{E, F, G\}$

$G \rightarrow \{G\}$

$H \rightarrow \{H\}$

$DA = \{A, B, C, D, E, F, G, H\}$

DB

DE

DF

AB DEF → prime attributes

CGH → non-prime attributes

(i2 = { A, B, C, D, E, H }

A → B
B → D
E → C
D → A

A → { A, B, D }
B → { B, D, A }
E → { E, C }
H → { H }
C → { C }
D → { D, A, B }

EH → { E, H, C }

AEH = { A, B, D, C, H, E }

BEH = { A, B, D, E, C, H }

DEH = { A, B, D, C, E, H }

R { A, B, C, D, E }

A → B A → C CD → E B → D
E → A

A : { A, B, C, D }

B → : { B, D }

C → { C, A, B, D }

D → { D }

E → { A, E, B, C, D }

CD → AC ✓ BC → CD
BD → CD × AC → BC

Canonical cover/Minimal cover:-

A canonical cover (F_c) for F is a set of dependencies such that F logically implies all dependencies in F_c and F_c logically implies all dependencies in F. Further F_c must have the following properties

(i) No functional dependency in F_c contains an extraneous attribute.

(ii) Each left side of functional dependency in F_c is unique ie there are no two dependencies such that if $\alpha_1 \to \beta_1$ and $\alpha_2 \to \beta_2$ such that $\alpha_1 = \alpha_2$

Extraneous attribute:-

An attribute of a functional dependency is said to be extraneous if we can remove it without changing the closure of set of functional dependencies.

$F_c = F$

repeat

use the union rule to replace any dependencies in F_c of the form $\alpha_1 \to \beta_1$ & $\alpha_1 \to \beta_2$ with $\alpha \to \beta_1 \beta_2$

Fist a FD $\alpha \to \beta$ in F_c with an extraneous attribute either α or in β

(test for extraneous attribute is done using F_c not F)

if an extraneous attribute found delete it from $\alpha \to \beta$ until F_c does not change.

1. set F_c of

2. For each FD $X \to A$ in F_c

for each attribute B in X

if $(X - B)^+$ with respect to F_c co

then replace $X \to A$ with

$X - \{B\} \to A$ in F_c

$F = \{ A \to B, AC \to B \quad A \to A \quad AD \to CE \quad B \to$

$F_c = \{ A \to B \quad AC \to B \quad A \to A \quad AD \to CE \quad B \to D$

$\downarrow$ $\downarrow$

$A \to B$ $AD \to C$ & $AD \to E$

$AD \to C$ $A \to E$

$F_c = \{ A \to B, AD \to C, B \to DE \}$

$F : \{ X \to Z, XY \to WP, XY \to ZWQ, XZ \to R \}$

$F_c : \{ X \to Z, XY \to W, XY \to P, XY \to Z, XY \to W, XY \to Q, XZ \to R \}$

(the $XY \to Z$ and $XY \to W$ are crossed out)

$F_c \{ X \to Z, XY \to WP, XY \to ZWQ, XZ \to R$

Find Minimal cover

$F: \{AB \to CD, B \to C, BC \to D, CD \to EF, E \to F\}$

$AB \to C, AB \to D, B \to C, BC \to D, CD \to E, CD \to F, E \to F$

$B \to C \quad B \to D \quad B \to D$

$B \to CD \quad CD \to EF$

$B \to EF$

Equivalence of FD

R(set of attributes)

FD_1 & FD_2 are two FD for relation R

1) $FD_2 \subseteq FD_1$
2) $FD_1 \subseteq FD_2$

① & ② case true $FD_1 = FD_2$

R(A, B, C, D)

$FD_1 = \{A \to B, B \to C, AB \to D\}$

$FD_2 \{A \to B, B \to C, A \to C, A \to D\}$

FD_2 : $AB^+ \{A, B, C, D\}$ $FD_2 \supseteq FD_1$

$FD_1 \supseteq FD_2$

$\therefore FD_1 = FD_2$

$R(A, B, C, D)$

$FD_1 = \{A \to B, B \to C, A \to C\}$

$FD_2 = \{A \to B, B \to C, A \to D\}$

$FD_1 \subset FD_2$

$FD_2 \not\subset FD_1$

$\therefore FD_1 \neq FD_2$

Lossless join and Dependency Preservation:-

Decomposition of a relation is done when a relation in relational model is not in appropriate normal form. Relation R is decomposed into two or more relations if decomposition is lossless join as well as dependency preserving.

Decomposition is lossy if $R_1 \bowtie R_2 \supset R$

Decomposition is lossless if $R_1 \bowtie R_2 = R$

Check for lossless join Decomposition:-

1. $Att(R_1) \cup Att(R_2) = Att(R)$
2. $Att(R_1) \cap Att(R_2) \neq \phi$
3. $Att(R_1) \cap Att(R_2) \rightarrow$ by for $Att(R_1)$ or $Att(R_2)$

$R(A\ B\ C\ D)$

$R_1(A\ B\ C)$ $R_2(A\ D)$

$R_1 \cup R_2 = (A\ B\ C\ D) = R$

$R_1 \cap R_2 = A \neq \phi$

$R_i \cap R_2 = A$

Dependency Preserving Decomposition:-

If we decompose a relation R into relations R_1 & R_2 all the dependencies of R are either must be a part of R_1 & R_2 or must be derived from R_1 & R_2 or FD's of R_1 & R_2.

~~Dependencies of R must b~~

$R(A,B,C,D)$ FD set $\{A \to BC\}$

is decomposed $R_1(ABC)$ $R_2(AD)$

$A \to AB$
$A \to AC$

$R(ABCD)$

$FD = \{A \to B, C \to D\}$

decomposition of R into $R_1(AB)$ $R_2(CD)$

$R_1 \cup R_2 = (ABCD)$

$R_1 \cap R_2 = \phi$ (No lossless join)

$R(A,B,C,D)$

$(A \to B \quad B \to C \quad C \to D \quad D \to B)$

$R_1(AB)$ $R_2(BC)$ $R_3(BD)$

Lossless decomposition

dependy Preserving.

Trivial FD $X \rightarrow Y$ Y is a subset of X

Non Trivial FD $X \rightarrow Y$ Y is not a subset of X

Normalization :- — redundancy
— data consistency

anomalies :-

↓

Deletion
Updation
Insertion

If a design of database is not perfect it may contain anamolies.

deletion → If we try to delete a record but parts of it was left deleted because of unawareness. If two different information is stored in one table at the end of the academic year if student record is deleted we lose the branch information also.

Insertion → If we try to insert data in a record that does not exist

Update :- If data items are scattered and are not linked to each other properly then it could lead to update anamoly. For example when we try to update 1 data item having its copies scattered over several places a few instances get updated properly while few others remain with old values leading to data consistency.

First Normal Form (1NF) :-

It is defined in tables itself.

It defines that all the attributes in a relation must have atomic domains i.e the values in the domain should be indivisible.

Second Normal Form :- (2NF)

i) it should be 1NF

a) For every 2NF every non prime attributes should be fully functional dependent (~~dependent on the prime~~) i.e $X \rightarrow A$

Third Normal Form :- (3NF)

i) it should be 2NF ~~i.e more noprimes~~

ii) no non prime attributes is transivitely dependent on primary key attributes.

For any non trivial FD.

$X \rightarrow A$

X is a super key or A is prime attribute.

stud_det

stuid id → Primary key attribut

stuid → zip

zip → city.

Boyce Codd Normal form (BCNF) :- (3.5 NF)

$X \rightarrow A$

↳ superkey

Bookid Genre id Genre type Price

does Not satisfy 3NF so split the tables

Bookid Genre id Price

Genre id Genre type

$R(F_1, F_2, F_3, F_4, F_5)$

$F_1 \rightarrow F_3$

$F_2 \rightarrow F_4$

$F_1, F_2 \rightarrow F_5$

$F_1F_2 \rightarrow F_1F_2F_3F_4F_5$

↑ candidate key

(RNo, Name, DOB, AGE, Eligibility)

DOB → AGE

AGE → Eligibility

Name → RNo

RNo → Name

DOB, Name → DOB, Age, Name, R

Fourth Normal form :- 4NF

Multivalued Dependency (MVD)

↓

when more than one multivalued attribute present in the table.

student name	Discipline	Activity

R(A, B, C, D, E)

$A \to B$

$BC \to E$

$ED \to A$

$BCD \to ABCDE$

$ACB \to ABCDE$

$ECD \to ECDAB$

A

① List all key.

② What is highest NF that this relation holds?

R(ABCDEFGH)

$AB \to C$ ✓

$A \to DE$

$B \to F$

$F \to GH$

~~CDEFGH → ABCDEFGH~~

No BCNF

3NF ✗

2NF ✗

1NF holds

id → Phone, ...

Phone → id, ...

R(ABCDE)

$CE \to D$

$D \to B$

$C \to A$

$CE \to ADCEB$

$CE \to ABCDE$

3.5 ✗

3 ✗

2 ✗

1 NF

R(ABCDE)

$R_1(ABC)$ $A_2(ADE)$

$A \to BC$ $A \to BC$ $E \to A$

$CD \to E$ A, E, CD, BC A ABCDE

$B \to D$ B BD

$E \to A$ A, E, CD, BC

(i) Listedout all functional dependencies & find Candidate keys. (A & E)

2. Check → lossless decomposition & dependency preserving

3. Find minimal cover F_c. $A \to ABCDE$, $CD \to ABCDE$, $B \to BD$

$R_1 \cup R_2 \equiv R$

$R_1 \cap R_2 = A \neq \phi$

It is lossless decomposition

$A \to B$ $A \to C$ $CD \to E$ $B \to D$ $E \to A$

Using Armstrong rules prove the union rule.

$\alpha \to \beta$ $\alpha \to \gamma$

$\alpha \to \beta\gamma$

Use augmentation rule and transitive rule

↓ augmentation rule.

$\alpha \to \gamma$ $\alpha\beta \to \beta\gamma$

$\alpha \to \beta\gamma$

Sound :- It is said to be sound if any dependency that can be derived from F using rules holds good in every relational satisfies dependencies

Complete:- Using inference rules repeatedly you can infer new set of functional dependencies until no more dependencies can be derived.

R(ABCDEF)

AB → C
DC → AE
E → F

AABD → AB, D ∈ F
CBD → ABEDEF
BD

INF

R(ABCDE)

F (AB → C, C → D, B → E)

R1(ABC) R2(CDE)

as R1 ∩ R2 is not a key

Lossy

R(A, B, C) F = {A → B, B → C}

↓ does not hold.

R1(A, C) R2(B, C)

F (A → C, AC → D, E → AD, E → H}

G (A → CD, E → AH)

F → A → C AC → D E → A E → D E → H

G → A → C A → D E → A E → H

Decomposition into 3NF:-

Both lossless and Dependency Preserving

Given relation R, F set of FD's

decomposition of R into a set of 3NF relation R.

Eliminate redudant FD's resulting in a canonical cover F_c.

Create a relation $R_i = XY$ for each FD: $X \to Y$ in the canonical cover

If the key k of R does not occur in any relation R_i create one more relation

$R_i = k$.

Decomposition into BCNF:-

It may be lossless but not dependency preserving

For all the dependencies $A \to B$ in F^+ check if A is a super key. If not then choose a dependency F^+ that breaks BCNF

say $A \to B$

$R = (A, B)$

R_2 A $(R - B - A)$

R(A, B, C, D, E)

ACE

F {A → B, BC → D}

candidate key (ACE)

F+ = { A → B , BC → D , AC → D
C → C , D → D , B → B , E → E

(BC)+ = BC

R1(AB) R2(ACED)

A → B AC → D

R1(AB) R2(ACD) R3(ACE)

Transaction Management:-

+
handles all transaction property

ACID

Atomicity → all the operations of a must be completed or no operation should take place (aborted)

Consistent → permanence of database consistent state

Isolation → Data used during transaction cannot be used by second transaction until first is completed.

Durability
↓
Ensures that once transactions are committed they cannot be undone or lost.

Transaction log keeps track of all transac
that update the database. DBMS uses the
stored in the log to recovery requirement by
by a rollback statement and system fa

Concurrency control:-

Concurrency control:- Coordination of the simulta
transactions executed in a multiuser database
system. Objective of concurrency is it make
sure that serializibility of transactions
is used in a multi-user database system.

Problems in concurrency control are:-

1) LOST UPDATE:- It occurs in 2 concurrent transactions when some dataelement is updated and one of the update is lost.

2) UNCOMITTED DATA:- It occurs when two
(Dirty Read) transactions are executed concurrently first
transaction is rolled back after the second
transaction already accessed data.

3) inconsistent retrival:- It occurs when a
(Write blind) transactions access data before and after
one or more other transactions finish working
with such data.

T_1

A	B
1000	2000

T: read(A)

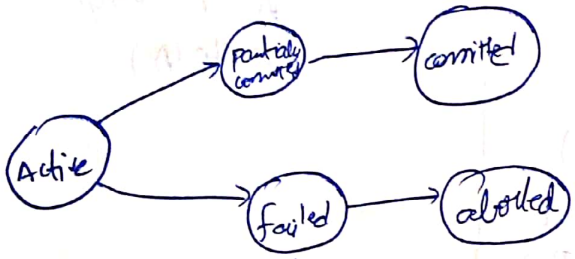
A: A - 50

write(A)

read(B)

B : B+50

write(B)



↓

internal logic error → kill the transaction

(Hw/sw errors) → restart

Concurrent transaction advantages

1. improved throughput
2. resource utilization
3. Reduce the waiting time

T_1

read (A)

A = A - 50

write(A)

read(B)

B : B+50

write(B)

T_2

read(A)

temp = A * 0.1

A = A - temp

write(A)

read(B)

B = B + temp

write(B)

Schedule:- T_1 T_2 } serial schedule

2 T_2 T_1

Schedule 3

T_1	T_2
read(A)	
A = A − 50	
write(A)	
	read(A)
	temp = A * 0.1
	A = A − temp
	write(A)
read(B)	
B = B + 50	
write(B)	
	read(B)
	B = B + temp
	write(B)

Schedule 4:- (Inconsistent)

T_1	T_2
read(A)	
A = A − 50	
	read(A)
	temp = A * 0.1
	A = A − temp
	write(A)
	read(B)
write(A)	
read(B)	
B = B + 50	
write(B)	
	B = B + temp
	write(B)

Serializability

(i) Conflict serializability

(ii) View serializability

A schedule is called conflict serializable if it can be transformed into a serial schedule by swapping non conflicting operations.

conflict operations:

Two operations are said to be conflict operations if the following conditions satisfy

(i) They belong to different transactions

(ii) They operate on the same data item

(iii) Atleast one of them is a right operation.

View serializable

Two schedules are said to be conflict equivalent when one can be transformed into another by swapping non-conflicting operations.

S_{11}

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

S_{12}

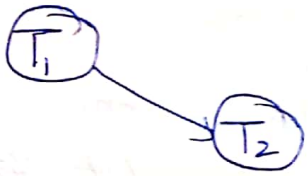
T_1	T_2
R(A)	
W(A)	
R(B)	
W(B)	
	R(A)
	W(B)
	R(B)
	W(B)

Dependency graph:-

(i) Create a node T in the graph for each pos in the schedule

(ii) For the conflicting operation read_item(x) write_item(x)

- If the transaction T_j executes a read_item(x) after T_i executes a write_item(x)



(iii) For the conflicting operation write_item(x) and read_item(x)

If the transaction T_j execute write_item after T_i executes read_item(x) then draw an edge between T_i and T_j in the graph

(iv) For the conflicting write_item(x) and write-item(x)

If the transaction T_j executes a write_item after T_i executes write_item(x) then draw an edge from T_i and T_j in the graph

A schedule is serializable if there is no cycle in the dependency graph.

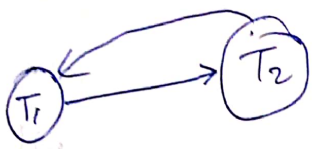
If means we can construct serial schedule

If there is no

S' which is conflict equivalent to S.

$S : r_1(X), r_1(Y), w_2(X), w_1(X), r_2(Y)$

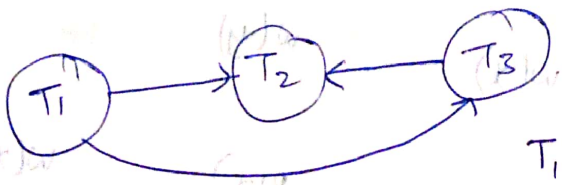
T_1	T_2
$r(X)$	
$r(Y)$	$w(X)$
$w(X)$	
	$r(Y)$



→ cycle exists

S_1 $r_1(X)$ $r_3(Y)$ $w_1(X)$ $w_2(Y)$ $r_3(X)$ $w_2(X)$

T_1	T_2	T_3
$r(X)$		
	$r(Y)$	
$w(X)$		
		$w(Y)$
	$r(X)$	
	$w(X)$	



$T_1 \rightarrow T_3 \rightarrow T_2$

$R_1(B)$ $R_3(C)$ $R_1(A)$ $W_2(A)$ $W_1(A)$ $W_2(B)$

$W_3(B)$ $W_1(B)$ $W_3(B)$ $W_3(C)$

T_1 T_2 T_3

R(B) R(C)

R(A) W(A)

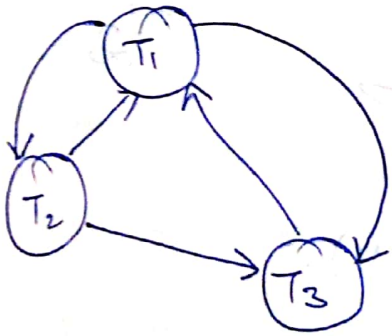
W(A) W(B)

W(B)

W(B)

W(B)

W(C)

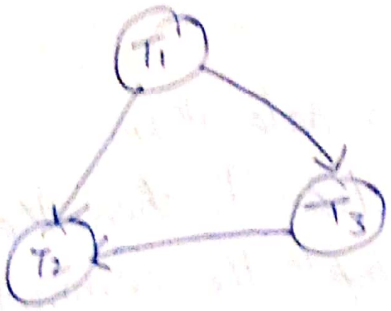


No-conflict serializability

T_1 T_2 T_3

R(x) R(y) R(y)

W(x) W(y)



$T_1 \rightarrow T_3 \rightarrow T_2$

View serializability :-

Two schedules are view equal if and only if

(1) Initial Read - If a transaction T_i reads data item from the database ~~then~~ in S_1, then in S_2 also T_i should read A from initial data base

(2) If T_i is reading A which is updated by T_j in S_1 then in S_2 also T_i should read(A) which is updated by T_j

T_1	T_2	T_3
W(A)		
	W(A)	
		R(A)

T_1	T_2	T_3
W(A)		
		R(A)
	W(A)	

Not equal

(3) Final write :- If transaction T_i updated A at last in S_1 then in S_2 also T_i should perform final write also

S_1

T_1	T_2
R(A)	
	W(A)
W(A)	

S_2

T_1	T_2
R(A)	
W(A)	
	W(A)

Recoverable Schedule:-

If a transaction T_j reads a data item previously returned by a transaction T_i then commit operation must appear before the commit of T_j

Schedule.

T_1	T_2
read A	
write(A)	read(A)
	read(B)

Lock based concurrency control protocols

A lock is a mechanism to control concurrent access to a data item, it can be locked in two different modes

1) exclusive (X) mode
2) Shared (S) mode

When it is in exclusive mode it can be both read & write and this X lock it is required using lock X instruction

In shared mode data item can only read lock S instruction

Lock compatibility Matrix

	S	X
S	true	fale
X	fale	false

A trasaction may be granted a lock on a data item if requested lock is compitable if the locks are comptiable already held on data item by other transaction.

Any no of trasactions can hold shared lock but if any trasaction holds exclusive lock then no other trasaction any lock on the data item. If a lock cannot be granted the requested transaction is made to wait till all incompatible locks held by other trasactions have been released.

The two phase locking protocol:-

ensures the conflict-serializable schedule

Phase 1 → growing phase

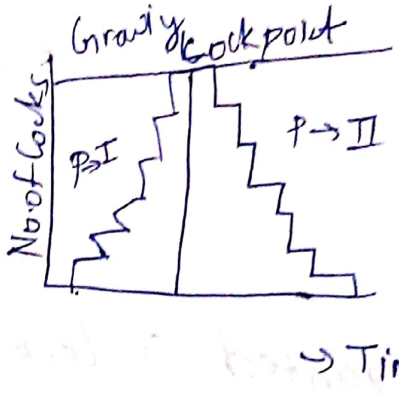
transaction may obtain lock

trasaction may not release lock

Phase 2 → shrinking phase

transaction may release locks

trasaction may not obtain locks



Phase locking does not ensure free from deadlock.

System is said to be in deadlock if there a set of transactions such that the transaction in the set is waiting for another transaction in the set.

Deadlock prevention.

Required that each transaction locks all its data items before it begins the execution

wait die scheme - non preemptive

wound-wait - preemptive

NP

Older transaction may wait for younger one to release the data item.

Younger transaction will not wait for older transaction to release its locked they will release back

Wound-wait

& older transaction wounds of younger transaction instead of waiting for it.

Time out based

A transaction may wait for a lock only a specified amount of time after that the wait times out and transaction is rolled back.

PL/SQL → used to Mix SQL with procedural constructs
↓
from various modules → to logic constructs
↓ procedures
functions
packages
Triggers

↓
block
↳ 3 - parts
Declaration
Execution section
Exceptional section

Cursor → strength of PL/SQL
↓
→ handle to an address in memory that stores the result of the executable SQL statement

→ useful to perform operation a single row returned from a SELECT stmt

Schema objects
View, sequences, indexes

Create View Viewname

from "SQL Query"