Date: / / Page no:

Database Management System

Unit -1

→ DBMS is a collection of data & set of prog to access & store these data in an easy & efficient Manner.

→ DBMS is a software which is used to manage database.

Ex:- MYSQL, oracle etc

Purpose of DB system:-

Previously data is stored in files.

⇒ Drawbacks of using file sys to store data.

① data redundancy & inconsistency:- multiple file formats, duplication of information in diff. files.

② difficulty in accessing data:- Need to write a new prog to carry out each new task

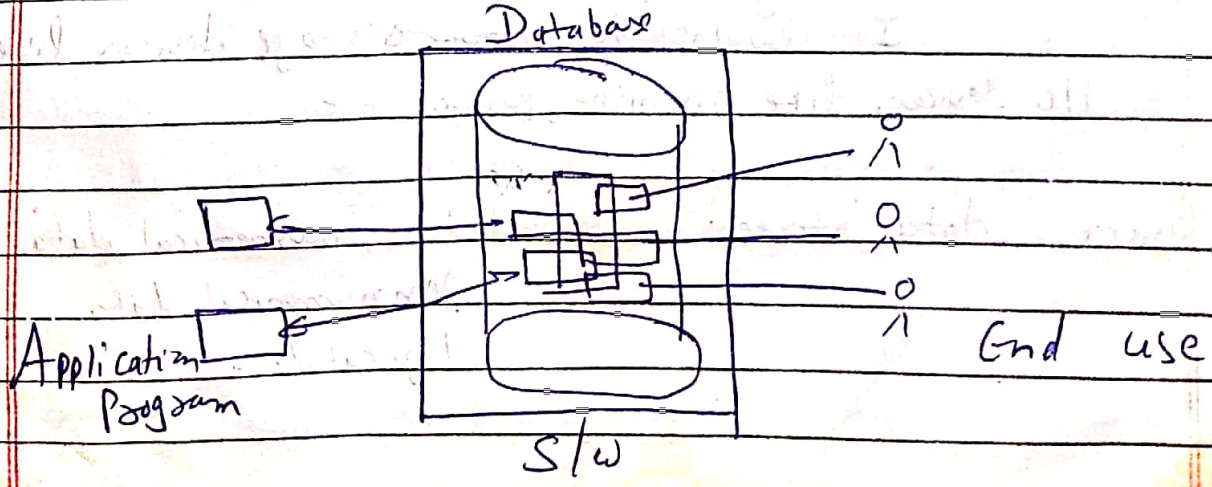
③ Data isolation:- multiple files & formats

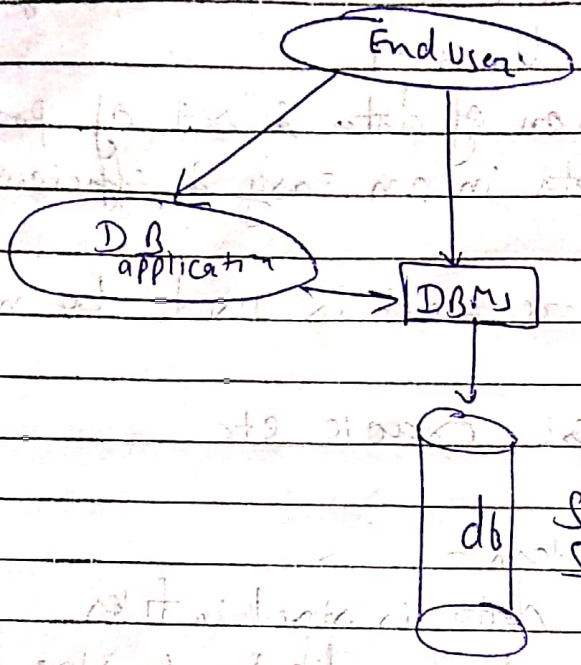
④ Data security:-

⑤ Transaction problems:-

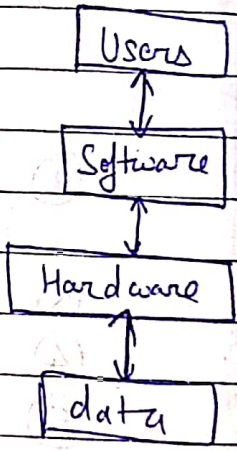
So, db system offer solutions to all these problems.

What is Database?





Components of DBMS



Users :- Application programmers, End users, Database Administrator

Software :- Controls the organization, storage, management & retrieval of data in db

Hardware :- H/w of a system can range from PC to a n/w of computers.
It also includes various storage devices like harddisk i/o devices like monitor, printer etc.

Data :- data stored in db includes, numerical data, Non numerical data, Logical data

Date __/__/__ Page no. ____

① Building Blocks of db
- ⇒ Columns / fields
- ⇒ rows / tuples / record
- ⇒ tables

② Advantages of db
- → data independence
- → Efficient data access
- → data integrity & security
- → data administration
- → Concurrent access & crash recovery
- → Reduce application development time

③ Application of DB:-
1. Banking :- (All transaction)
2. Airlines :- (Reservation, scheduling flight)
3. University :- (registration, grades)
4. Sales :- (customer, product, purchases)
5. Manufacturing :- (Production, inventory, order, Supply chain)
6. Human Resources :- (Employee Record, Salary, tax deduction)
7. Telecommunication :- (records of calls, Monthly bill)

④ Drawbacks of DB
- ⇒ The overhead cost of using dbms
  - → High initial investment, in h/w, s/w & training
  - → Cost of defining & processing data
  - → overhead for security, concurrency control, recovery.

→

Hence It may be more desirable to use regular files under

Date: / / Page no:

→ Simple well defined db app that are not expected to change.
→ no multiple user access to data.

DBMS Architecture

The DBMS ~~architecture~~ design depends upon its architecture. Database architecture depends upon how users are connected to the database to get their request done.

Types of DBMS Arch.

- → 1 Tier Architecture
- → 2 Tier —11— ____
- → 3 Tier —11— ____

Database architecture can be seen as single tier or multi tier But logically, database architecture is of two types like: two tier architecture & 3 tier.

1 Tier Architecture

→ In the 1-tier, database is directly available to the user. It means user can directly sits on the DBMS & uses it.
→ The 1-Tier architecture is used for development of local application, where programmers directly communicate with the database for quick response.

Date: / / Page no:

2-Tier Architecture:-

→ The 2-tier architecture is same as basic client-server. In the two tier architecture, application on the client end can directly communicate with the database at the server side. For this interaction, API's like: ODBC, JDBC are used.

→ The user interfaces and application programs are run on client side.

→ The server side is responsible to provide the functionalities like: query processing & transaction mgmt.

→ To communicate with the DBMS, client side app. establishes a connection with the server side.

3-Tier Architecture:-

→ The 3-tier architecture contains another layer b/w the client & server. In this architecture, client can't directly communicate with the server.

→ The application on the client-end interacts with an application server which further communicates with the DB system.

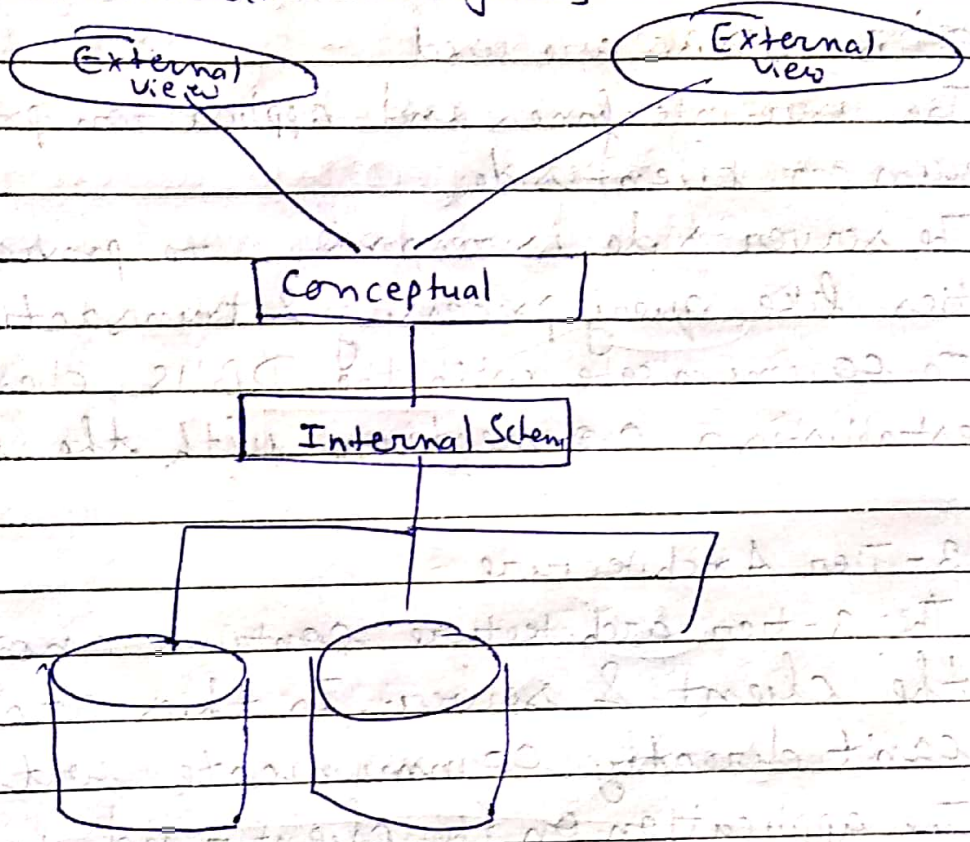
→ End user has no idea about the existence of the DB beyond the application server. The db also has no idea about any other user beyond the application.

→ The 3 tier architecture is used in case of large web application.

Date: / / Page no:

Three schema Architecture

The three schema architecture is used to separate the use application & physical database. The three schema architecture contains three levels. It breaks the database down in three diff. categories.



Internal Level

→ Internal level has an internal schema which describe the physical storage structure of the database.

→ Internal schema is also known as physical schema

→ It uses the physical data Model. It is used to define that how the data will be stored in a block.

→ The physical level is used to describe complex low level data structure in detail.

2. Conceptual level :

→ Conceptual schema describes the design of database at the conceptual level. Conceptual level is also known as logical level.

→ Conceptual schema describes the structure of whole db.

→ It describes what data are to be stored in the db & also describes what relationship exist among those data.

→ In this, Internal details such as implementation of DS is hidden.

→ Programmers & database administrators work at this level.

3. External level

→ At the external level, a DB contains several schemas that sometimes called as subschema. The subschema is used to describe diff. view of the database.

→ It is also known as view schema.

→ Each view schema describes the database part that a particular user group is interested & hides the remaining database from that user group.

→ The view schema describes the end user interaction with database system.

ER Model

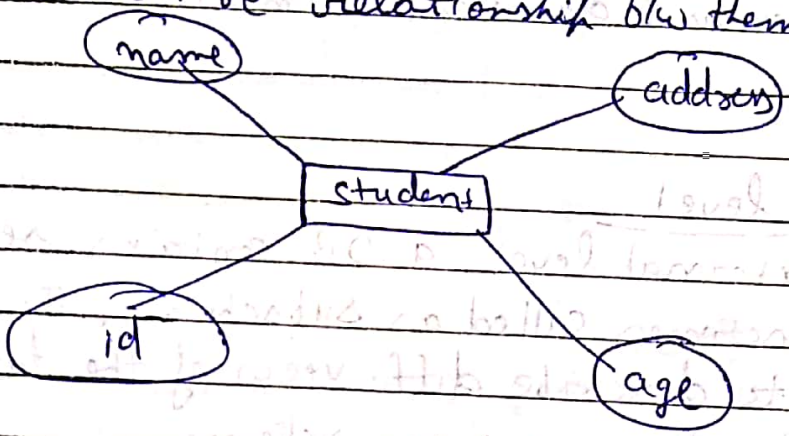
→ ER model stands for Entity-Relationship model. It is a high level data model. This model is used to define the data elements & relationship for a specified system.

→ It develops a conceptual design for the database. It also develops a very simple & easy to design view of data.

Date: / / Page no:

→ In ER modeling, the database ~~str~~ structure is portrayed as a diagram called an ER diagram.

for ex:- If we design a school database then student will be an entity with attributes like address, name, id, age etc. an address can be another entity with attributes like city, street name, pincode etc. and there will be relationship b/w them.



Component of ER diagram

- Entity
  - → Weak Entity
- Attributes
  - → Key Attribute
  - → Composite attribute
  - → Multivalued Attribute
  - → Derived Attribute
- Relation
  - → One to One
  - → One to Many
  - → Many to One
  - → Many to Many

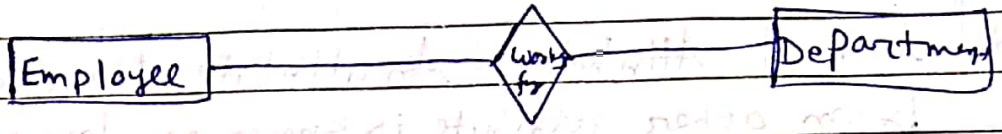
1) Entity :- An entity may be an object, class, person or place. In ER diagram, an entity can be ~~repre~~ represented as rectangle.

Tangible :- Physically exists object

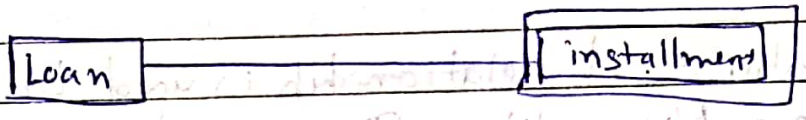
Nontangible :- logically exists

Date: / / Page no:

Consider an organization as an example manager product, employee, department, etc, can be taken as entity



Weak Entity :- An entity that depends on another entity called as weak entity. Weak entity doesn't contain any key attribute of its own.

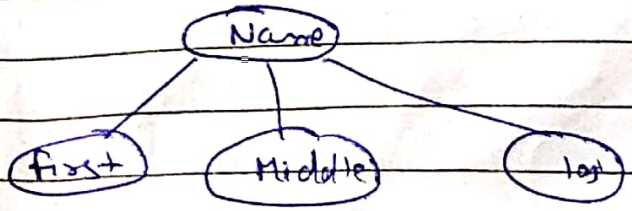


2 Attribute :- Attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

For ex. :- Id, age, contact number, name, etc can be attribute of a student

(a) Key attribute :- It is used to represent main characteristic of an entity. It represents a primary key. Key attribute is represented by eclipse with the text underlined.

(b) Composite attribute :- An attribute that composed of many other attributes is known as composite attribute. Composite attribute is represented by eclipse & those eclipses are ~~con~~ connected with an eclipse

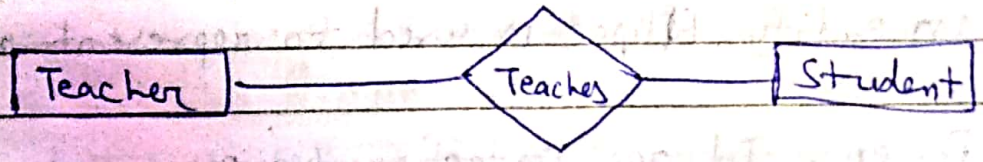


(c) Multivalued Attribute :- An attribute can have more than one value. These attributes are known as multivalued attribute. Double oval is used to represent multivalued attribute.

(d) derived Attribute :- An attribute than can be derived from other attribute is known as derived attribute. It can be represented by dashed eclipse

For ex:- A person's age changes over time and can be derived from another attribute like: DOB.

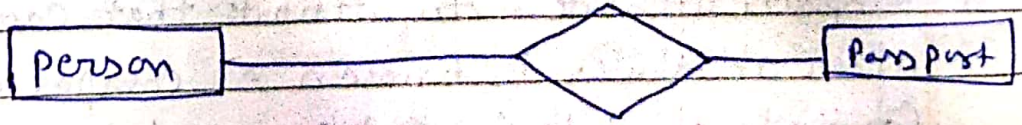
3 Relationship :- A relationship is used to describe the relation b/w entities. Diamond is used to represent the relationship.



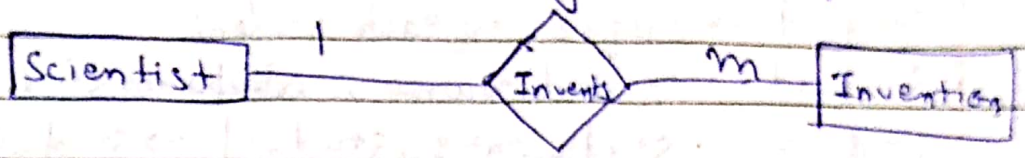
Types of relationship :-

(a) One to One Relationship :- When only one instance of an entity is associated with the relationship then it is known as one to one relationship

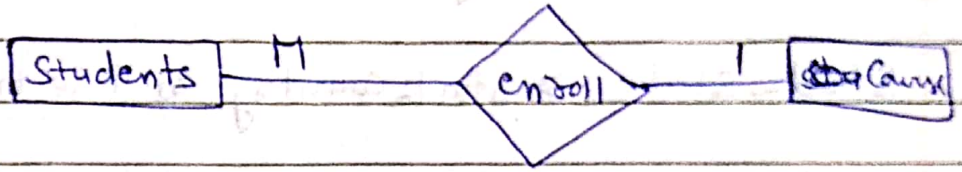
For ex :- A



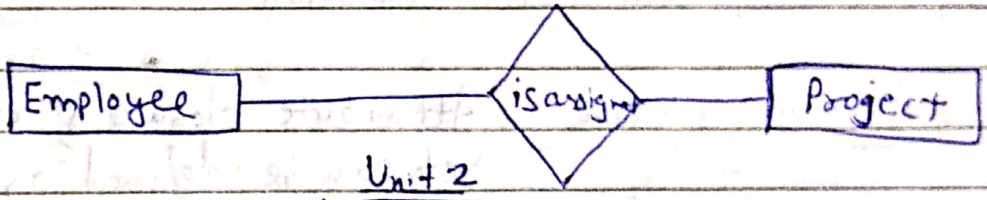
(b) One to Many Relationship :- When only one instance of entity on the left & more than one instance of an entity on the right associates with the relationship then this is known as one to many relationship.



(c) Many to One



(d) Many to Many Relationship



Unit 2

Functional Dependency

A functional dependency $A \rightarrow B$ in a relation holds if two tuple having same value of attribute A also have same value for attribute B

Stud_no → Stud_Name, Stud No → stud_Addr hold

but stud_Name → Stud_Addr do not hold

Stud_id	Stud_Name	Stud_Phone	Stud_State	Stud_Ctry	Stud_Age
1	Ram	123	Haryana	In	20
2	Ram	12345	Punjab	In	19
3	Sujit	456	Raj	In	18
4	Suresh		Pun.	In	21

Date: / / Page no:

Functional Dependency in a relation are dependent on the domain of the relation. Consider the Student relation given

Stud_id → unique for each student
Stud-id → stud-name, ~~stud-no~~ stud-id → stud-Ph
stud-id → stud-state, stud-id → stud-Country &
stud-id → stud-age.

Stud-State → stud-country will be true

FD Set :- FD set of a relation is the set of all FDs represent in the relation.

Attribute Closure :- Attribute closure of an attribute set can be defined as set of attributes which can be functionally determined from it.

Find Attribute Closure:
→ Add elements of attribute set to the result set.
→ Recursively add elements to the result set which can be functionally determined from the elements of the result set.

$(Stud\_no)^+ = \{stud\_no, stud\_name, stud\_Ph, stud\_state, stud\_country, stud\_age\}$

$(Stud\_state)^+ \Rightarrow \{stud\_state, stud\_country\}$

R(A, B, C)

$A \rightarrow B$
$B \rightarrow C$

$A \rightarrow BC$

Pseudo transitive Rule

$X \to Y$ & $YZ \to W \to XZ \to W$

Reflexive Rule :- if $\alpha \supseteq \beta$ α is a set of attribute & b. β is subset of α then alpha holds beta.

Augmentation rule :- If $\alpha \to \beta$ holds & y is attribute set then $ay \to by$ also holds.

Transitive rules :- if $\alpha \to \beta$ & $\beta \to c$ holds then $\alpha \to c$ also holds. $\alpha \to \beta$ is called as a functionally that determines b.

Trivial Functional Dependency :-

Trivial :- If a ~~fo~~ FD $X \to Y$ holds where Y is a ~~subj~~ subset of X, then it is called a trivial FD.

Non Trivial :- If an FD $X \to Y$ holds where y is not a subset of X then it is called a non-trivial FD.

Completely non-trivial :- if an FD $X \to Y$ holds where X intersects $Y = \phi$, it is said to be completely non-trivial FD.

Normalization :- If a database design is not perfect, it may contain anomalies, which are like a bad dream for any database administrators, Managing a database with anomalies is next to impossible.

→ Update :-
→ Insert
→ Delete

Date: __/__/____ Page no: ____

~~UDF~~

Closure set of Attribute

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

$F = F_1 + F_2$

$A \to B \quad A \to C$

$A \to BC$ (Total Dependency)

$A^+ = \{A, B, C\}$
$B^+ = \{B, C\}$
$C^+ = \{C\}$

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

$A \to B$
$BC \to DE$
$AEG \to G$

$(AC)^+ = AC$
ABC
$ABCDE$

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

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

$B^+ = B$
$= BD$

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

$AB \to C$
$BC \to AD$
$D \to E$
$F \to B$

$(AB)^+ \to (AB)$
(ABC)
$(ABCD)$
$(ABCDE)$

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

$A \to BC$
$CD \to E$
$E \to C$
$D \to AEH$
$ABC \to BD$
$DH \to BC$
$BCD \to H\ g$

~~$BCD^+ \to H$~~
$BCD^+ \to BCD$
$\to BCDE$
$\to ABCDEH$

2 NF

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

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

$(AB)^+ = ABCD$

↳ C.K.

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

$(AB)^+ = AB$
$ABCD$

Teacher_id	Subject	Teacher_age

3 NF

Empid	Emp_Name	Emp_Zip	Emp_State	Emp_City

Candidate key Empid
Non prime Attribute

Empid, Empname, Empzip	Emp Zip	Emp State	Emp City

State, city → zip zip → Empid

$AB \rightarrow C$

$C \rightarrow D$ → Transitive Dependency

If Non Primen attribute finds another nonprime attribute then it is called as transitive dependency

i) It must be in 2NF.

ii) It should not have any transitive dependency.

$R_1(A, B, C)$

$R_2(C, D)$

BCNF

$R(A, B, C)$

$(A)^+$

$(AB)^+$ ✓

$(AC)^+$ ✓ } CK

$AB \rightarrow C$

$C \rightarrow B$ } → Neither Partial nor transitive

It's in 3NF.

$\alpha \rightarrow B$

P/NP P

$\alpha \rightarrow B$

SK

for BCNF α must be a super key.

Fourth Normal Form (4NF)

→ A relation will be in 4NF if it is in BCNF & have no multivalued dependency.

→ For a dependency A → B if for a single value of A multiple value of B exists then the relation will be multivalued dependency.

Ex:

Stud id	Course	Hobby
21	Comp.	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

This given Student table is in 3NF but the Course & Hobby are two independent entity. Hence, there is no relationship b/w Course & Hobby.

In the student rel^n, student with stu-id, 21 contains two courses, Comp & Math and two hobbies, Dancing & Singing. So there is a Multivalued dependency on stu-id which leads to unnecessary repetition of data.

So to make the above table into 4NF, we can decompose it into 2 table

Stu-id	Course
21	Comp.
21	Math
34	Chem.
74	Bio.
59	Phy.

Stu-id	~~Dancing~~
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

Data: __/__/____ Page no: ____

5NF

→ A relation is in 5NF if it is in 4NF & Not contains any join dependency & joining should be lossless

→ 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.

→ 5NF is also known as Project-join normal form.

Sub	Lecturer	Semester
Comp.	Anshika	Sem 1
Comp.	John	Sem 1
Math	John	Sem 1
Math	Akash	Sem 2
Chem.	Praveen	Sem 1

P1 →

Sem	Sub
Sem 1	Comp.
Sem 1	Math
Sem 1	Chem.
Sem 2	Math

P2 →

Sub	Lecturer
Comp.	Anshika
Comp	John
Math	John
Math	Akash
Chem.	Praveen

Sem	Lecturer
Sem 1	Anshika
1	John
1	John
2	Akash
1	Praveen

Date: / / Page no:

Relational Algebra

Relational Algebra is a procedural query lang. It gives a step by step process to obtain the result of query. It uses operators to perform queries

It is a collection of op. on relations. Each opr^n takes one or more rel^n as its operand(s) & produces another relation as its result.

Relational operation

Operation
- Unary (select, project)
- Binary (operate on pair of relation) cartsesian, Union, Intersection

Select Operation :- It select tuples (rows) from a relation (table). Denoted by σ

Syntax :- $\sigma_{(\text{Select Cond}^n)}(R)$

↳ filtering Criteria

Ex :-

Student → Relation (Sid, Sname)

Sid	Sname
1	A
2	B

$\sigma_{Sid=1}$ Student

In Select op all relational operator may be used.

Cond^n may be Combined using and($\wedge$) or ($\vee$)

$\sigma_{Sid=1 \text{ or } Sid=3}$ (Student)

Date: __/__/____ Page no: ____

Solved Ex.

Student

Sid	Sname	Teacher	Marks
1	A	T1	90
2	B	B	80
3	C	T1	85
4	D	B	75

Q. Find the details of students whose name is same as teacher

$$\sigma_{Sname = Teacher}(Student)$$

Q. Find the details of students whose ~~marks~~ marks are greather than or equal to 85.

$$\sigma_{(marks \geq 85)}(Student)$$

Project Operation :- It yields column (attribute) of a relation. Using project op^n duplicate values are automatically removed

Syntax:- $\Pi_{A1, A2 \ldots An}(R)$

↓ Attribute of a Relation

Ex:- Student

RollNo	SName	Class
1	A	11
2	B	12
3	C	11
4	B	12

$\Pi_{Name}(Students)$

O/P

A
B
C

Project & select op^n can be used simultaneously

Solved Ex:-

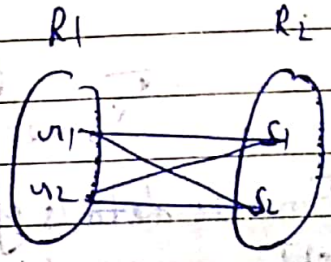
$$\Pi_{Name}\left[\sigma_{Class=12}(Student)\right]$$ — o/p B

Cartesian Product:- It produces new relation which has a degree (no of attributes) equal to the sum of the degree of two relations operated upon the no of tuples is product of no. of tuples of two relations.

Ex:- R_1 R_2 $(R_1 \times R_2)$

Attributes x y (x+y)

Tuples n_1 n_2 $(n_1 \times n_2)$



R

A	B
α	2
β	1

S

C	D
x	101
y	102

$R \times S \Rightarrow$

A	B	C	D
α	2	x	101
α	2	y	102
β	1	x	101
β	2	y	102

Date: / / Page no:

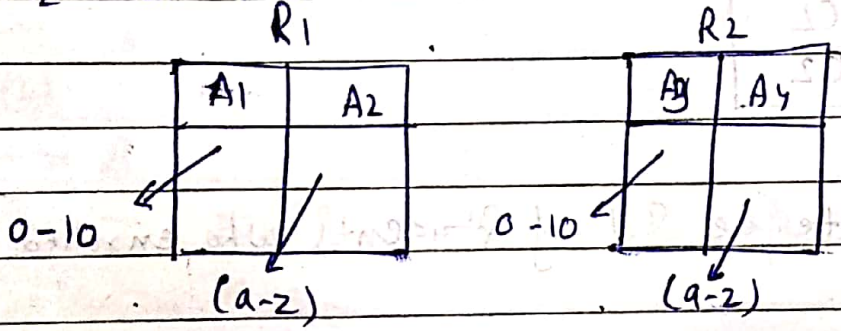
Set op^n

Note:- Set operations are applicable if the relations are Union Compatible

Union Compatibility:

i) The relation R_1 & R_2 must be of the same arity (degree) (no of column / attributes)

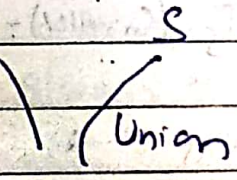
ii) The domain of ~~the~~ i^{th} attribute of R_1 & ~~is~~ i^{th} attribute of R_2 must be same for all i.



Union op^n

→ denoted by U.

→ duplicate tuples are removed.



R ∪ S

→ tuples from either R & S or both.

R

id	name	age
1	A	20
2	B	21

S

id	name	age
3	C	23
4	D	24

R ∪ S →

id	name	age
1	A	20
2	B	21
3	C	23
4	D	24

$\Pi_{name}(R) \cup \Pi_{name}(S)$

→ A
B
C
D

Date: / / Page no:

Division Method (Derived Op)

A(x, y) / B(y) = if results x values for that there should be tuple <x, y> for every y value of relation

E(Sid, Cid) / C(Cid = C1)

Sid	Eid
S1	C1
S2	C1
S1	C2
S3	C2

Cid
C1
C2

Query :- Retrieve Sid of Students who enrolled in all every Class.

$$\Pi_{Sid}(Enrolled) - \Pi_{Sid}\left(\left(\Pi_{Sid}(Enrolled) \times \left(\Pi_{Eid}(Course)\right)\right) - (Enrolled)\right)$$

Joins

→ Used to Combine related tuples from two relations into single tuple.
→ Denoted by '⋈'
→ Similar to Cross / Cartesian Product

Join	Cartesian Product
Combination of tuples that satisfy the filtering / matich Condition.	all possible combinations of tuples from the relation.

Data: __/__/____ Page no: ____

Types of Joins

- Inner Join [contains only matching tuples]
  - → Theta (θ) Join $\bowtie_{(\theta)} R$
  - → Equi Join
  - → Natural Join
- Outer Join [all rows from either one or both]
  - → Left Outer
  - → Right outer
  - → Full outer

R S

4 6

(2)

→ 2

θ :— $\bowtie_\theta (R)$ → Comparative op $=, <, \leq, \geq, >, \neq$

Equi Join :— $\bowtie =$

Natural :— based on common attribute in both relation

Left outer :— left relation tuples will always be in result

Right outer :— RHS relation tuples will —//—

Full outer :— tuples of from both relation

Inner Join

Theta (θ) Join :—

Mobile

model	Price
Nokia	10,000
Samsung	20,000
iphone	50,000

Laptop

model	Price
Dell	30,000
Acer	20,000
HP	25,000

Want to purchase both mobile, laptop but mobile price should be less than laptop price

mobile.model	Laptop model
Samsung	Dell
Nokia	Dell
Nokia	Acer

$$\text{Mobile} \bowtie_{(\text{mobile.price} < \text{laptop.price})} \text{Laptop}$$

Equi Join

$$\text{Mobile} \bowtie_{(\text{mobile.price} = \text{laptop.price})} \text{Laptop}$$

Natural Join

Natural join op^n forms a Cartesian product of its two arguments performs selection forcing equality on those attribute that appear in both relation schemas and finally removes duplicate attributes.

Emp

Name	Emp	Dname
A	1	Fin
B	2	Sales
C	3	Sales

Dept

Dname	Manager
Fin	M1
Sales	M2

EMP ⋈ Dept

Name	Empid	Dname	Mgr
A	1	Fin	M1
B	2	Sales	M2
C	3	Sales	M2

Date: / / Page no:

Outer Join :- It is used when we want to keep all the tuples in either or both the relation in the result of the Join regardless of whether or not they have matching tuples in other relation.

Left outer Join :- keep every tuple in the first or left relation

⟕

R

ename	did
A	1
B	2
C	3

S

did	dname
1	d_1
2	d_2

R ⟕ S

ename	did	dname
A	1	d_1
B	2	d_2
C	3	—

Right Outer Join :- keep every tuple in the second or Right relation ⟖

R ⟖ S

ename	did	dname
A	1	d_1
B	2	d_2

R

ename	did
A	1
B	2
C	1

S

did	dname
1	d_1
3	d_3

R ⟖ S

ename	did	dname
A	1	d_1
—	3	d_3
C	1	d_1

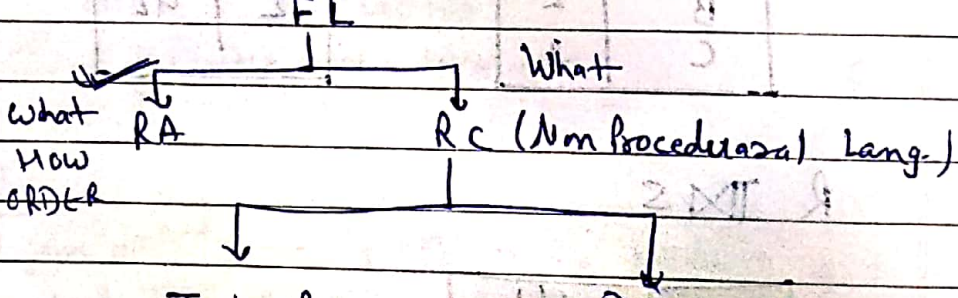
Date: __/__/____ Page no: ____

Full outer Join :- Keeps tuples from both left & right relation.

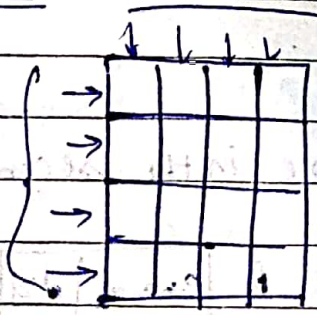
R ⟗ S

ename	did	dname
A	1	d1
B	2	–
C	1	d1
–	3	d3

Relational Calculus



TRC



DRC – (attribute)

Op^n on tuple

open Unsafe Operation

~t(Emp)

(negation)

Date: / / Page no:

Tuple Relation Calculus

Student

Fname	LName	Marks
a	b	100
b	c	40
c	d	60

tuple variable = t

find student those marks > 50

{ t | Student(t) & and t.Marks > 50 }

t.Fname, t.Lname

Syntax :- { $t_1.A_1, t_2.A_2$ | $A_1(t_1), A_2(t_2)$ ---- $t_1.M_1 > 50$ And ... }

Domain Rel. { $t_1.A_1, t_2.A_2$ | | }

- Atomic or Simple
- Complex or Composite

Emp

F	L	A
		20
		21
		22

Emp(t) & t.Age

{Emp} ≥ 20 (Simple)

- $A_1 \wedge A_2$ (and) opn
- $A_1 \vee A_2$
- $\neg A_2$ (negation)
- $\exists t(\_)$ (If there exist in t)
- $\forall t(\_)$ for all

T1

A_1	

T2

A_2	

$t_1(A_1) > t_2(A_2)$

Date: / / Page no:

t

Quantifier

A	B
10	11
20	12
30	13
40	14

⇒ $\exists(t).A > 50)$

↳ false

⇒ $\forall(t.B > 10)$

Relational Calculas :- Relational Calculas is non-procedu query lang. & has no description about how the query will work or the data will be fetched. It only focusses on what to do, and not on how to do it.

TRC :- In tuple relational calculas, we work on filtering tuples based on the given condition.

Syntax

$\{T \mid condition\}$

~~for ex :- In this~~

Q1 :- Relation is student & name is a attribute of Studen

: if want to get data for students with age so greater than 17

$\{T.name \mid Student(T) \text{ and } T.age > 17\}$

DRC :- In domain relational calculas filtering is done based on the domain of the attribute & not based on the tuple values

$\{C_1, C_2, C_3, \dots C_n \mid F(C_1, C_2, C_3, \dots C_n)$

Date: / / Page no:

Where C1, C2, . . . represents domain of attribute & F defines the formula including the condition for fetching the data.

$\{ \langle name, age \rangle \mid \in student \wedge age > 17 \}$

Composite Datatype :- A composite data type could be a record, table, nested table, varray. This is so because all of them are composed of multiple data types. Composite datatypes can be used to construct complex data type full filling different requirements.

Primitive datatype are the basic units of a language each primitive value contains a single data and describe that data literally. Primitive data is very straight forward. Primitive datatypes are as their name suggests, simple. They can hold text, message, frame numbers, movie size value, & so on.
Ex: Number, String, boolean, undefined & null

Date: / / Page no:

Unit - 3

SQL :- Structure Query language (SQL) is a database query lang. used for storing & managing data in Relational DBMS. SQL was the first commercial language introduced for E. F. Codd's Relational model of database. Today almost all RDBMS (My sql, Oracle, Infomix, Sybase, MS Access) use SQL as the standard database query language. SQL is used to perform all types of data operation in RDBMS

SQL Commands:

SQL define following ways to manipulate data stored in RDBMS

DDL :- Data definition language

This includes changes to structure of the table like creation of table, altering table, deleting a table etc

All DDL commands are auto committed. That means it saves all the changes permanently in the Database

Command	Description
Create	to create new table or database
alter	for alteration
truncate	delete data from table
drop	to drop a table
rename	to rename a table.

Date: / / Page no:

DML :- Data manipulation lang.
DML commands are used for manipulating the data stored in the table & not the table itself.

DML commands are not auto committed. It means changes are not permanent to database, they can be rolled back.

insert :- to insert a new row
update :- to update existing row
delete :- to delete a row
merge :- merging two rows or two tables

TCL :- Transaction Control language
These commands are to keep a check on other commands and their affect on the database. These commands can annul changes made by other commands by rolling the ~~data~~ data back to its original state. It can also make any temporary change permanent.

Commit :- to permanently save
rollback :- to undo change
savepoint :- to save temporarily.

DCL :- Data Control lang.
DCL are the commands to grant & take back authority from any database user
Grant :- grant permission of right
revoke :- take back permission.

DQL :- Data Query lang.
It is used to fetch data from tables based on conditions that we can easily apply.

Select :- retrieve records from one or more table

SQL Constraints :- SQL constraints are rules used to limit the type of data that can go into a tables, to maintain the accuracy & integrity of the data inside table.
Constraint can be divided into the following two types.

1) Column level :- Limits only Column data.
2) Table level :- Limits whole table data

Following are the most used Constraint
→ Not null
→ Unique
→ Primary Key
Foreign key
Check
Default.

Not NULL :- Not null Constraint restricts a col. from having a NULL value, Once not null Constraint is applied to a column, you cant pass a null value to that column. It enforces a column to Contain a proper value.

Date: / / Page no:

Unique

Unique Constraint ensure that a field ~~or~~ or column will only have unique values. A unique constraint field will not have duplicate data. This constraint can be applied at column level or table level

Create table Student (S_id not null unique, Name - -

Alter table Student Add Unique (S_id),

Primary key :- Primary key constraint uniquely identifies each record in database. A primary key must contain unique value & it must not contain null value. Usually primary key is used to index the data inside the table.

Create table student (S_id int primary key - -)

Foreign key :- It is used to Relate two tables. Foreign key Constraint is also used to restrict actions that would destroy links b/w tables.

Customer_Detail

C_id	Cust_name	address
101	Alex	Noida
102	Aman	Delhi
103	Rohit	Jaipur

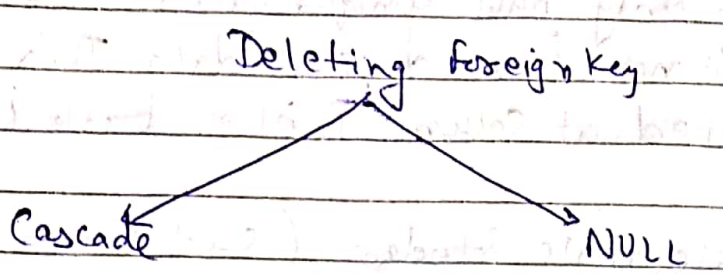
Order_details

O_id	Order_name	C_id
10	01	101
11	02	103
12	03	102

If you try to insert any incorrect data, DBMS will return error & will not allow you to insert the data

Data: / / Page no:

Create table Orderdetail (
C-id int Foreign key References Customer_Detail (C-id));



1) On Delete Cascade :- This will remove the record from child table if that value of foreign key is deleted from the main table.

2) On Delete NULL :- This will set all the values in that record of child table as NULL, for which the value of foreign key is deleted from the main table.

If we don't use any of the above, then we cannot delete data from the main table for which data in child table exists. We will get an error if we try to do so.

Check Constraint :- Check Constraint is used to restrict the value of a column b/w a range. It performs check on the values, before storing them into the database. Its like condition checking before saving data into a column.

Create table Student (S-id int not null check (S-id > 0)
);

Views

→ In some cases, it is not desirable for all users to see the entire logical ~~mode~~ model.

→ Consider a person who needs to know an instructors name & department, but not the ~~salary~~ salary. This person should see a relation described, in SQL by

select id, name, dept-name from instructor

⇒ A view provides a mechanism to hide certain data from the view of certain users.

→ Any relation that is not of the conceptual model but is made visible to a user as a "Virtual relation" is called a view.

→ A view is defined using the create view statement which has the form

create view v as <query expression>

→ Once a view is defined the view name can be used to refer to the virtual relation that the view generates.

Ex- create view faculty as select id, name, dept-name from instructor

select * from faculty.

→ Create a view of department salary totals.

create view dept_total_salary (dept-name, total-salary) as select dept-name, sum(Salary) from instructor group by dept-name.

Date: __/__/____ Page no: ____

Stored Procedure :

A stored procedure is a prepared SQL code that you can save, so the code can be reused over & over again.

So if you have a SQL query that you ~~to write~~ write over & over again, save it as a stored procedure and then just call it to ~~ex~~ execute it.

You can also pass parameter to a stored procedure so that the stored procedure can act based on the parameter value that is passed.

```
create Procedure procedure-name
as
sql-statement
Go;
```

Execute a Stored procedure

```
EXEC procedure_name;
```

Stored procedure with one parameter

```
create procedure p_name
@id int
as
Sql query
Go;
```

Date: / / Page no:

Trigger :-

It is a procedure that starts automatically if specified changes occur to the DBMS.

⇒ DB triggers has 3 parts :-

→ 1) Trigger event
2) Trigger Condition ⇒ optional
3) Trigger action ⇒

⇒ When an event occured, db trigger is fired and an predefined PL/SQL block will perform necessary action.

Syntax :-

Create or replace Trigger trigger-name
{ Before / After } trigger event on table name

[That can be done for each row]
[when condition]

→ Declare
declaration Stmts
Begin
executable statements
Exception
Exception Block
End;

Date: / / Page no:

Program

SQL → Select * from customers;

id	name	age	add	Salary

⇒ Trigger Displays the salary diff. b/w the old value & new values

```
Create or replace trigger salary_diff.
Before Deleting or insert or update on customer
for each row
when ( New.id > 0)

Declare
   Salary_diff number,
Begin
   Sal_diff := New.Salary - Old.salary;
dbms_output.put_line("old salary : " || : old.salary);
dbms_output.put_line("new salary : " || : new.salary)
——— || ——————— .("Salary.dif : " || :Sal_diff );
End;
```

Date: / / Page no:

PL-SQL Trigger:

→ Are invoked by oracle Engine automatically when a specified event occurs.

→ Are stored programs in database & are invoked repeatedly when specific condition matches.

Can be written in response of the following event: → DML, → DDL, → DB op (Login,)

Adv:

↳ Enforces Referential Integrity (PK, FK)

→ Triggers are used in auditing purpose

→ It is also used in synchronous replication of table.

→ It is also used for event logging.

Syntax:

Create or [Replace] Trigger trigger_name

trigger_timing → Before / After

event1 or event2 → insert, update, delete

On table name

PL/SQL Block; → Trigger type → Statement (executes only once (Default)) / Row (executes once for each row affected by triggering event)

Ex:

Create Trigger SQLchange

Before delete,

Unit 5

Date: / / Page no:

Transaction Management

→ Database transaction is an atomic unit that contains one or more SQL statements.

→ It is a series of opⁿ that performs as a single unit of work against a database.

→ It is a logical unit of work

→ It has a beginning & an end to specify its boundary

Let's take an simple ex of bank transaction, Suppose a bank clerk transfer Rs 1000 from x's account to y's accour

x's Account	y's Account
Open Account(x)	Open Account(y)
pre_balance = x.balance	Prev_balance = y.balance
Curr_balance = pre-balance - 1000	Curr-balance = Prev + 1000
x.balance = Curr-balance	y.balance = Curr.bal.
Close-account(x)	Close_account(y)

Transaction Properties:-

Following are the Transaction Properties, referred to by an acronym Acid Properties:-

1) Atomicity
2) Consistency
3) Isolation
4) Durability

→ Acid properties are the most imp. concepts of database theory.

→ A transaction is a small unit of program which contains several low level tasks.

→ These properties guarantee that the database transactions are processed reliably.

1 Atomicity :- It is ensures that a transaction will
→ run to completion as an individual unit, at the end of which either no changes have occurred to the database or database has been changed in a consistent manner.

ALL or none

Ex. Rs 500 from A → B

A = 2000 , B = 3000 ~~+~~ = 5000

if after successful completion

A = 2000 - 500 = 1500 ↓ 5000

B = 3000 + 500 = 3500 ↓

Ti R(A, a)

a - 500

write (A, a) → Power failure → A → 1500, B = 3000 ⇒ 4500

Read (B, b)

b = b + 500

write (B, b)

4500 ≠ 5000

Inconsistent

2 Consistency :- It is ensure that if the database was in a consistent state before the starts of a transaction then or termination the database will also be in a consistent state

→ Correctness or

Sum (A, B) = Sum (A, B)

before trans. after transaction.

→ Isolation :- It indicates that the actions performed by a transaction will be isolated or hidden from outside the transaction until it terminates.

T1	T2
R(A)	
a-500	
W(A)	R(A)
	R(B) Print(A, B)
R(B)	
B=B+500	
W(B)	

↓

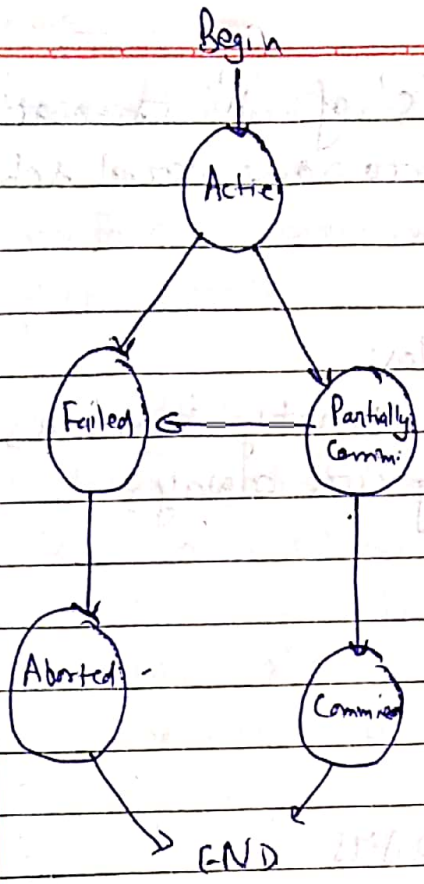
execute first this & after this T2 will be executed

Durability :- All updates done by a Transaction must become permanent. Or
Ensure that the commit action of a transaction on its termination will be reflected in the database.

It is a commit.

Transaction State :-

1) Active
2) Partially Committed
3) Failed
4) Aborted
5) Committed



1) Active :- Active is the initial state of every transaction. The transaction stays in active state during execution.

2) Partially Committed :- It defines that the transaction has executed the final statement.

3) Failed :- It defines that the execution of the transaction can no longer proceed further.

4) Aborted :- It defines that the transaction has rolled back & the database is being restored to the consistent state.

5) Committed :- If the transaction has completed its execution successfully then it is said to be Committed.

Date: / / Page no:

Serializability:- A schedule 'S' of 'n' transactions is serializable if it is equivalent to some serial schedule of the same 'n' transaction.

Property of Serializable Schedules:-
Serializable Schedule behave exactly same as serial schedules, So they ~~are~~ are always.

→ Consistent
→ Recoverable
→ Casacadeless
→ Strict

What is Serializability in DBMS

→ When multiple transactions run concurrently then it may give rise to inconsistency of the database
→ Serializability is a concept that helps to identify which ~~no~~ non serial schedules are correct & will maintain the consistency of the database.

Diff. b/w Serial Schedule & Serializable Schedules
The only diff. b/w serial schedule & serializable schedule is that

→ In serial schedule one one transaction is allowed to execute at a time ie no concurrency is allowed
→ Whereas in serializable schedules multiple transactions can execute simultaneously i.e. concurrency is allowed.

Types of Serializability

1) Conflict Serializability
2) View —//—

1) Conflict :- If a given schedule can be converted into a serial schedule by swapping its non-conflicting op^ns then it is called as a conflict serializable schedule

Conflicting op^ns:-
Two operations are called conflicting operations if all the following conditions hold true for them:
1) Both the op^ns belong to diff. transactions.
2) Both the op^ns are on same data item
3) At least one of the two op^ns is a write operation

Ex Consider the following schedule:-

T1	T2
R1(A)	
W1(A)	
	R2(A)
R1(B)	

In this W1(A) & R2(A) are called as conflicting op^ns.

Check if a given schedule is Conflic or not
1) Find and list all the conflicting op^ns.
2) Start creating a precedence graph by drawing one node for each transaction.

Date: __/__/____ Page no: ____

3) Draw an edge for each conflict pair such that if $X_i(V)$ & $Y_j(V)$ forms a conflict pair & then draw an edge from T_i to T_j which ensure that T_i gets executed before T_j.

4) Check if there is any cycle formed in the graph. If there is no cycle found, then the schedule is conflict serializable otherwise not.

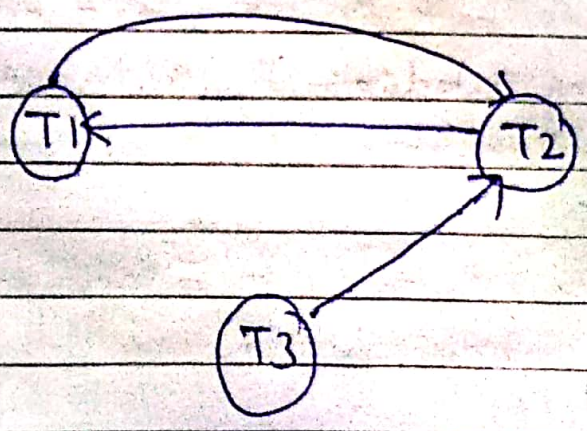
Ex: $S: R_1(A), R_2(A), R_1(B), R_2(B), R_3(B), W_1(A), W_2(B)$

Sol:

1)

$R_2(A)$ $W_1(A)$	$T_2 \rightarrow T_1$
$R_1(B)$ $W_2(B)$	$T_1 \rightarrow T_2$
$R_3(B)$ $W_2(B)$	$T_3 \rightarrow T_2$

2)



Since there exist a cycle in the precedence graph, therefore the given schedule S is not conflict Serializable

Date: / / Page no:

2

T1	T2	T3	T4
	R(x)		
		W(x)	
		Commit	
R(x)			
W(x)			
Commit	W(y)		
	R(z)		
	Commit		R(x)
			R(y)
			Comm.

Sol

1) $R_2(x) \cdot W_3(x)$ $T_2 \rightarrow T_3$

$R_2(x)$ $W_1(x)$ $T_2 \rightarrow T_1$

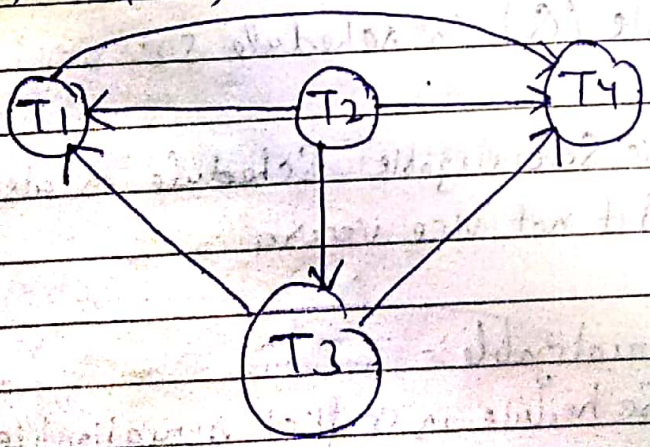
$W_3(x)$ $R_4(x)$ $T_3 \rightarrow T_1$

$W_3(x)$ $R_4(x)$ $T_3 \rightarrow T_4$

$W_1(x)$ $R_4(x)$ $T_1 \rightarrow T_4$

$W_2(y)$ $R_4(y)$ $T_2 \rightarrow T_4$

2



There is no cycle in the graph therefore the given schedule S is conflic ~~serial~~ serializable.

Date: / / Page no:

2 View Serializability :-

→ If a given schedule is found to be view equivalent to some serial schedule, then it is called as a view serializable schedule.

Consider two schedules S1 & S2 each consisting of two trans. T1 & T2. schedules S1 & S2 are called view equivalent if the following three condition hold true

1) For each data item Q, if T_i reads an initial value of Q in schedule S1 then T_i must in S2 also reads an initial value of Q.
2) If T_i executes Reads Q in S1 & that value was produced by T_j (if any) then T_i must in schedule S2 also reads the value of Q that was produced by T_j.
3) For each data item Q the transaction that perform the final write (Q) opⁿ in schedule S1 must also perform the final write (Q) in schedule S2.

Note :- Every Conflic Serializable Schedule is also view serializable but not vice versa

Check View Serializable :-

1) If the given Schedule is conflict serializable, then it is surely view serializable. If it is not conflict then go to step2.
2) Check if there exists any blind write opⁿ (writing without reading). If there does not exist any blind write then the schedule is not view serializable. If there exist then go to step 3.

3) In this method, try finding a view equivalent serial schedule.

Q1

T1	T2	T3	T4
R(A)			
	R(A)	R(A)	
			R(A)
W(B)			
	W(B)		
		W(B)	
			W(B)

Sol

$W_1(B), W_2(B)$ $T_1 \rightarrow T_2$

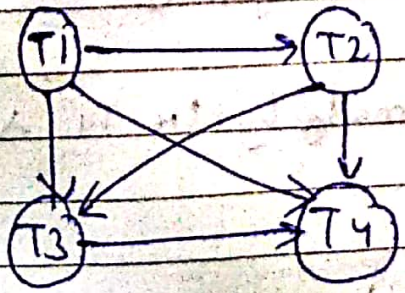
$W_1(B)$ $W_3(B)$ $T_1 \rightarrow T_3$

$W_1(B)$ $W_4(B)$ $T_1 \rightarrow T_4$

$W_2(B)$ $W_3(B)$ $T_2 \rightarrow T_3$

$W_2(B)$ $W_4(B)$ $T_2 \rightarrow T_4$

$W_3(B)$ $W_4(B)$ $T_3 \rightarrow T_4$



No cycle means it is conflict serializable therefore S is view Serializable.

Q

T1	T2	T3
R(A)		
	R(A)	
		W(A)
W(A)		

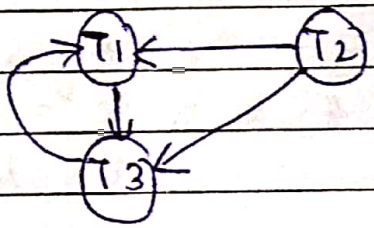
Sol

$R_1(A)$ $W_3(A)$ $T_1 \rightarrow T_3$

$R_2(A)$ $W_3(A)$ $T_2 \rightarrow T_3$

$R_2(A)$ $W_1(A)$ $T_2 \rightarrow T_1$

$W_3(A)$ $W_1(A)$ $T_3 \rightarrow T_1$



S is not conflict so go to check step 2

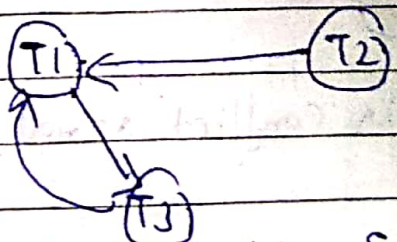
Blind write opn is available so go to step 3

→ Transaction T1 firstly read A & T3 firstly updates A Therefore T1 must execute before T3

$T1 \rightarrow T3$

→ final updation on A is made by the T1 so

$(T_2, T_3) \rightarrow T_1$



There is cycle so S is not view serializable

Date: / / Page no:

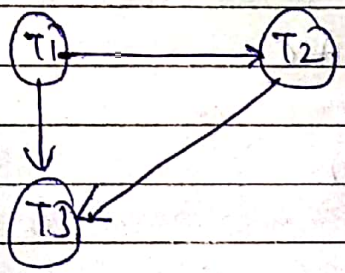
Q.

T1	T2	T3
R(A)		
	W(A)	
		R(A)
W(A)		
		W(A)

T1	T2	T3
R(A)		
W(A)		
	W(A)	
		R(A)
		W(A)

Sol No conflicting so check blind writes it is available so dependency

→ T1 → T2 (firstly read & firstly updation)
→ (T1, T2) → T3 final updation
→ T2 → T3 ~~Read write sequence~~ write read seq.



no cycle → View Serializable.

Q Check Conflict Equivalent T1 → T2

$R_1(A) \rightarrow W_2(A)$
$W_1(A) \rightarrow R_2(A)$
$W_1(A) \rightarrow W_2(A)$

S1

T1	T2
R(A)	
W(A)	
	R(A)
	W(A)
R(B)	
W(B)	
	R(B)
	W(B)

S2

T1	T2
R(A)	
W(A)	
R(B)	
W(B)	
	R(A)
	W(A)
	R(B)
	W(B)

$R_1(A) \rightarrow W_2(A)$

S1 ≡ S2