DATA:

Data are known facts which can be stored or recorded.

Information : Processed data

DATABASE:

It is a collection of related data.

*) A database is a logically coherent collection of data with some inherent meaning.

A random collection of data can't be treated as database.

*) A database always represents some aspects of the real world.

*) A database is designed, built and populated with data for a specific purpose.

It has some intended group of users and some preconceived applications in which the users are interested.

*) Database is manual or computerised.

DBMS :

It is a general purpose software system which task is to → defining a database.

→ Constructing a database

→ Manipulating a database

→ Sharing the database among various users and applications.

→ Protecting of database

→ Maintaining the database over a long period of time.

⇒ Defining a database

It means specifying datatypes structures and constraints of the data to be stored in the database.

⇒ Constructing the database means storing of data on some storage medium which is controlled by the DBMS.

⇒ Manipulating a database includes function such as querying the database to retrive specific data and function like insert, delet- and update in the database.

⇒ Sharing a database allows multiple users and programs to acces the database simultenio- usly.

⇒ Protection of database means to protect database from H/W & S/W malfunction and protection from unauthorized acces.

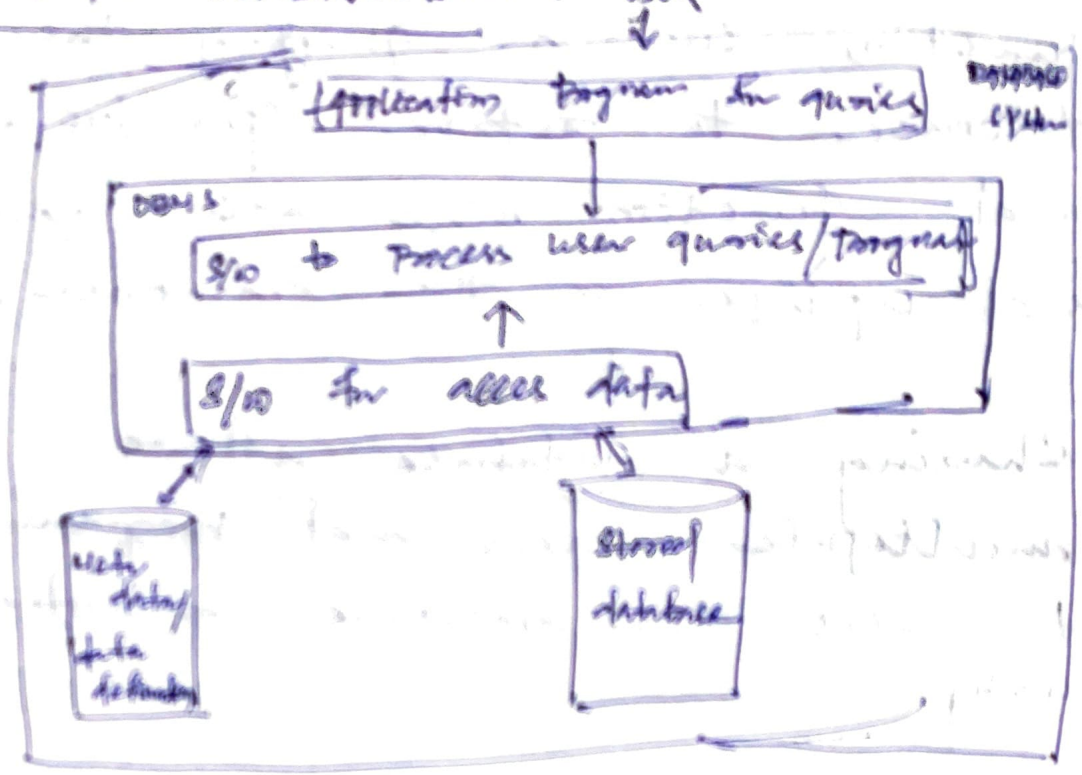
⇒ Maintaining a database means to maintain the requirement of the database belongs to the real world.

DATABASE SYSTEM:

Database system = DBMS + Database

Eg. of DBMS: {ORACLE, MYSQL, SQL SERVER, DB2, Postgre SQL, MS Acces} distributed. (Standard (not dist.))

DATABASE ENVIRONMENT :



CHARACTERISTICS OF DATABASE APPROACH OR ADVANTAGES OF DATABASE SYSTEM OVER TRADITIONAL FILE SYSTEM :

→ Self describing nature of the database system :

→ The database system contains not only the database itself but also a complete definition/description of the database structure and constraints.

→ This definition is stored in DBMS catalog which contains information such as the structure of each file, the type and storage format of each data item and

various constraint on the data.

This catalog is also known as data dictionary. The information is stored in the database catlog is known as meta data.

→ Insulation bet" program and data:

→ Support multiple views of data:

A view may be a subset of the database or it may contain virtual data that is derived from the database but not explicitly stored in the database.

→ Controlling Redundancy:

Data Redundancy means storing of same data multiple times in different locations. (is known as data Redundancy)

Redundancy leads to the following problems:

1) Duplication Effort
2) Wastage of storage space
3) Leads to inconsistant data.

Ex: (Meta data: The information stored in the catalog is called Meta data It describes the structure of Primary database)

Redundancy can't be zero, it only be reduce or minimize.

⇒ Restricting unauthorized access.

⇒ Providing storage structure for efficient query processing.

(Searching will be easy or take less time to search.)

⇒ Providing backup and recovery.

⇒ Providing multiple user interface.

⇒ Representing complex relationship among data.

⇒ Data integrity or enforcing integrity constraint on the data

(Some Rules enforcing in the database so that there ~~will be~~ will not occure any error.)

DATABASE LANGUAGES

tl: 26 DEC 21

① DDL (Data Definition Language)

→ It is used to define the conceptual schema.

→ DDL statments are used to specify the database structure or schema.

→ So it specifies relations, attributes, constraints and relationships.

N.B: After compilation of DDL statments database catalog or data dictionary is created which stores meta data.

② Storage definition Language (SDL)

SDL is defines internal schema.

(Storage structure)

It is used for defining Physical storage structures or accel methods to find ~~time~~ tune the performance of the database system.

③ VIEW Definition Language (VDL)

It is used to define external schema. i.e it is used to specify user views and there

mapping to the conceptual schema.

④ Data Manipulation Language (DML)

DML provides a set of operation to support the basic data manipulation operations on the data held in the database.

Typical manipulation includes
(i) Retrival (iii) Update (iv) Delete
(ii) Insert operations in a database.

There are two main types of DML

(i) High Level / Non-Procedural DML
(ii) Low Level / Procedural DML

⑤ Data Control Language (DCL)

DCL is used for granting and revoking user access on a database.

DCL statement includes GRANT and REVOKE statements / commands.

GRANT : It gives user access privileges to the database.

REVOKE : It withdraws users access privileges given by using GRANT command.

⑥ Transaction Control Language (TCL)

→ It used to manage transaction in a database.

→ These are used to manage the changes made by DML statements.

The following are the statements under TCL :

(i) COMMIT
(ii) ROLLBACK
(iii) SAVEPOINT

Schema :

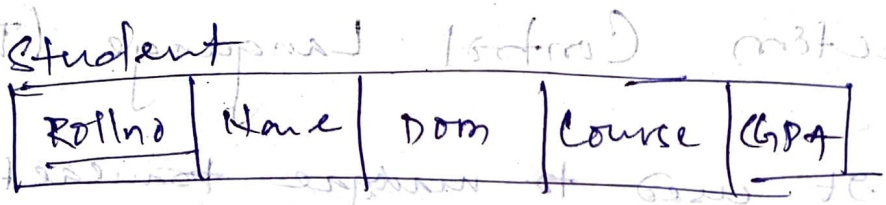
The description of a database is known as database schema.

A schema represents the names of data items, their formats or data types and some type of constraints.

The schema can be defined in any of the following ways.

Student (Rollno : Number
Name : string
DOB : date
Course : string
CGPA : Number)

Student (Rollno, name, DOB, Course, CGPA)



(constraint) (can't be duplic

N.B: Here the key fields are underlined.

DATABASE STATE :

The data in a database at a particular moment in time is known as database state.

It is also known as "Snapshot of a database" or "database instant".

N.B: The schema is sometimes called the intension and the database state is called extension of the schema.

DATABASE ARCHITECHTURE

DE: 9-Jan-2022

Data model determines the structure of data.

A data model defines the logical design and structure of a database and in which format the data stored, accessed or updated in a database.

The datamodel supports concepts of data abstraction in a database system.

The types of data model :-

(i) High level or Conceptual data model (ER model (Rel^n bet^n entities))
(ii) Low level / Physical data model
(iii) Representational or implementation data model (Eg: Relational Model)
(Logical data model)

~~(iv)~~

Database Architecture

Three schema Architecture / ANSI (American National Standard) Architecture for DBS

Goal of this Archi.

@ To ~~Access~~ achieve

→ Use of a Catalog to store the database description (schema) so as to make it self describing.

→ Insulation of program and data (data independence)

→ Support multiple user views

* In three schema architecture the schema can be defined in following ways.

→ Internal / Physical schema

→ Conceptual / logical schema

→ External schema (view)

* The internal level has an external schema, which describes the physical storage structure of the database.

The internal schema uses a physical data model and describes the complete ~~storage~~ details of data storage and access path for the database.

(The data structure are use to access data efficiently is known as access path)

* The conceptual level has a conceptual schema which describes the structure of the whole database

So it describes entities / relations, attributes, data types, constraints, & relationships and user operations.

So it hides details of the physical storage structure.

Usually, a representational datamodel is used to describe the conceptual schema.

the external or view level includes a number of external schemas or user views.

Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.

Dt 5 Jan 22

Mapping :

The process of transferring the requests and responds between schemas is known as mappings.

Data Independence :

Data independence can be defined as the capacity to change schema at one level of a database system without having to change the schema at the next higher level.

*) Physical data independence

*) Logical data independence

*) PDI

⇒ Physical data indep. has the capacity to change the internal schema without having to change the conceptual schema.

Hence the external schema need not be changed as well.

⇒ Changes to the external schema may be needed because

Some physical files were re-organized.

Ex: by creating additional access structures to improve the performance of the retrival or update.

⟹ If the same data as before remains in the database, the conceptual schema need not be changed.

↳ LDI

→ It is the capacity to change the conceptual schema without having to change external schema or application program.

→ We may change the conceptual schema to expand the database (by adding record type or data items)

adding new row / adding new columns

to change the constraints (Rules and regulation for data) or to reduce the database (by removing a record type or data item).

In these cases, the external schema should not be affected.

DATABASE USERS :-

Database users are the people who are involved in

→ design

→ use and maintenance of a large database

There are following catagory of users

→ Database Administrator (DBA)

→ Database Designer

→ End users

→ System Analyst / Application Programmers

→ DBA :-

DBA is a person or a group of people in an organization whose main task is to administer, oversee and manage the database, DBMS and related software.

→ Responsibilities of DBA :-

→ Schema definition

→ Storage structure and access

→ Ensuring data security (Prevent unauthorised users)
→ Schema modifications and performance monitoring
→ Assisting application programmers
→ Backup and Recovery

→ Database Designers :-

→ End Users :-

Diff. type of end users
→ casual end users
→ Naive or parametric
→ Sofesticated
→ Standalone users

7 Jan, 2022

E-R MODEL

(i) It is a high level conceptual data model.
(ii) It is used for conceptual design of database application or a database.
(iii) This model describes the basic data structuring concepts relationship and constraints.

(iv) Diagrammatic notation for E-R model is known as E-R model.

COMPONENTS :-

(i) Entity

It is a real life object or thing which has either physical existence or conceptual existence.

Entity is represented by a rectangular box ▭ :

Entity name is always a noun.

e.g. EMPLOYEE

(ii) Attributes

The properties which describe the entity is known as attribute.

e.g. An EMPLOYEE can be described by his/her id, name, designation, salary.

Attributes are represented by ovals in E-R diagram. 

eg.

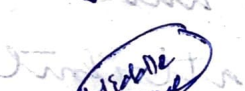
TYPES of Attribute:

(a) Simple attribute and composite attribute

→ Simple attribute: the attributes which can't be/ are not divisible.

eg Roll no., Id etc

→ Composite attribute: the attributes that are divisible to subparts or components attributs are known as composite attribute.



e.g. Name, Address, Contact

(b) Single valued and Multivalued Attributes.

→ Single valued attribute:

The attribute which has a single value for a particular entity.

eg: Id, Roll no, DOB

→ Multivalued attribute:

the attribute which has more than one value for a particular entity.

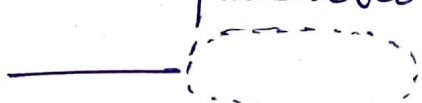
eg: Qualification, Department locations

In E-R diagram the multivalued attribute are represented by doubleline ovals.

(c) Stored attribute and Derived attribute

DOB is a stored attribute and AGE is a derived attribute.

• Derived attributes are represented by dotedline ovals.



(c) Key attributes

A set of attributes ~~of an entity~~ of an entity type is known as ~~entity~~ key iff it's values are distinct for each individual entity type in this entity set.

or// A set of attributes (one or more attribute) of an entity type is known as key if the value of this set of attribute uniquely identify each and every entities of that entity type.

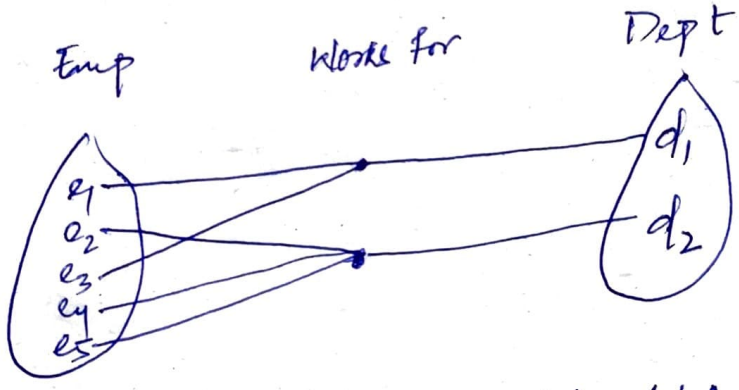
The key attributes are underlined inside the oval. — (c)

Emp

→ Constraints on Relationship type /
Structural Constraints

i) Cardinality ratio Constraints
ii) Participation Constraints /
Existence dependence

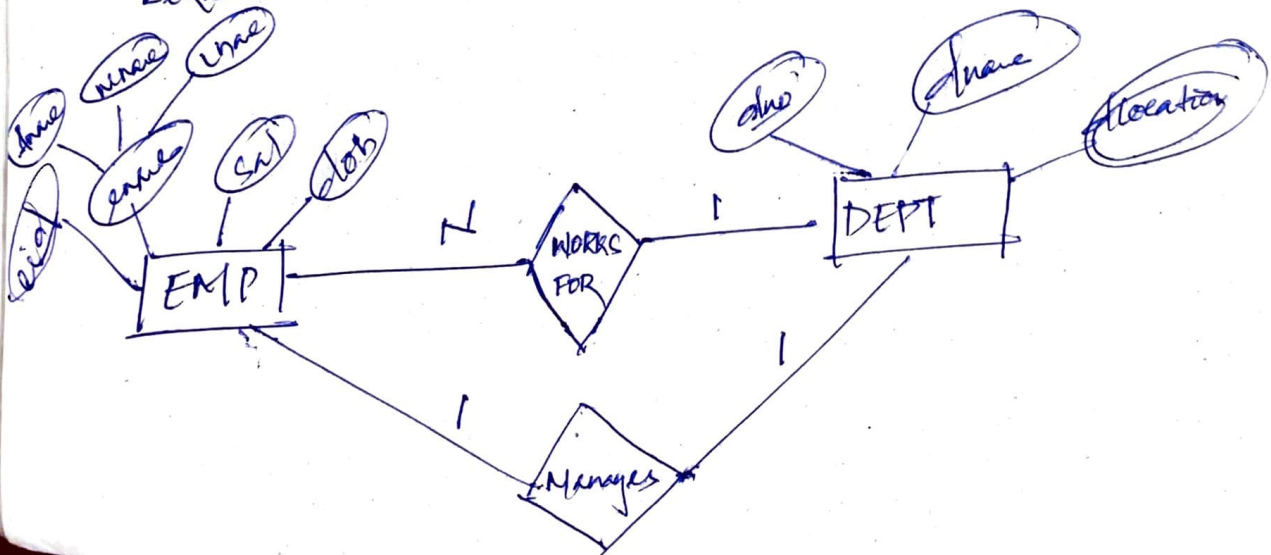
① Cardinality ratio Constraints :-



Many employes works for a department.

N : 1

Like that 1:N , 1:1 , M:N



ii) Participation Constraint 15 JAN 22

Existance of an entity depends upon it's relationship with other entities via relationship type or through relationship type.

In this constraint no entity can't be represent in an E-R digram independently.

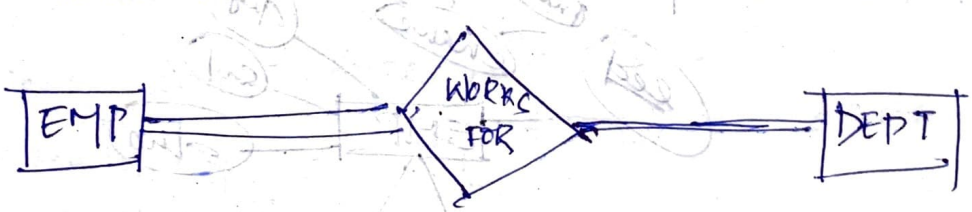
It specifies minimum no. of relationship instances that each entity can participate in and sometimes it is called as minimum cardinality constant.

a) Total participation Constant
b) Partial participation constant

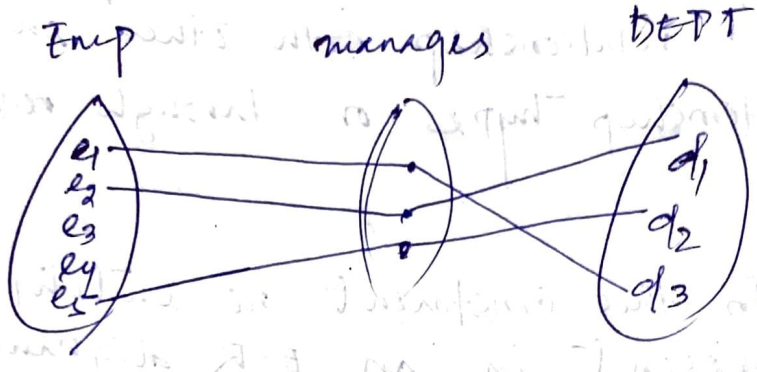
a) Total participation Constant

Here total participation constraint of each and every entity of an entity type participate in a relationship.

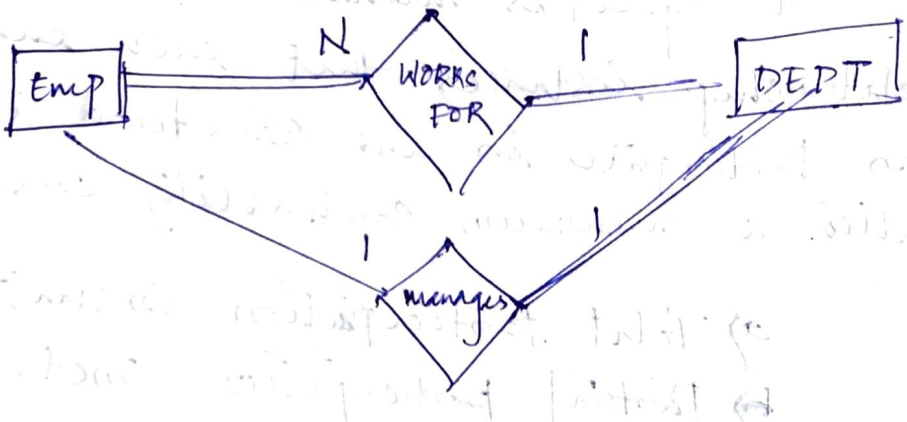
It is represented by double line.



b) Partial participation Constant.

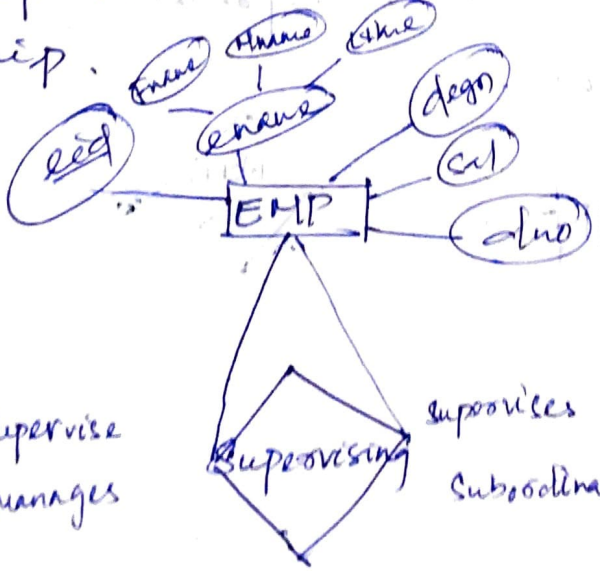


It is represented by single line.



Recursive Relationship :-

When some entities of an entity type participates more than once in a relationship type in different roles the relationship is known as recursive relationship.



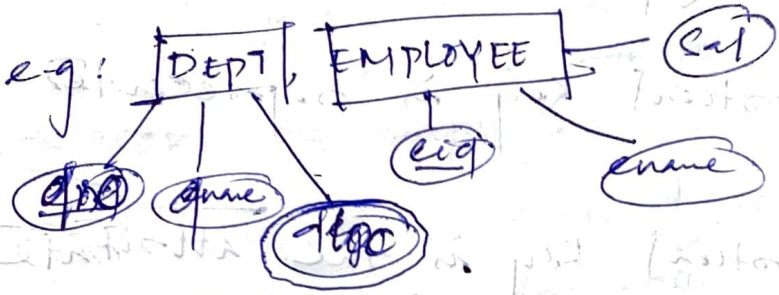
Role name: Role played by an entity type with respect to relationship type.

* W.r.t recursive relationship "Role name" are mandatory.

Dt: 19 JAN 22

Strong Entity type:

Entity type which has a key attribute of its own to identify its ~~entities~~ unique entities



imp

Weak entity type:

Weak entity type is an entity type that does not have key attributes of its own.

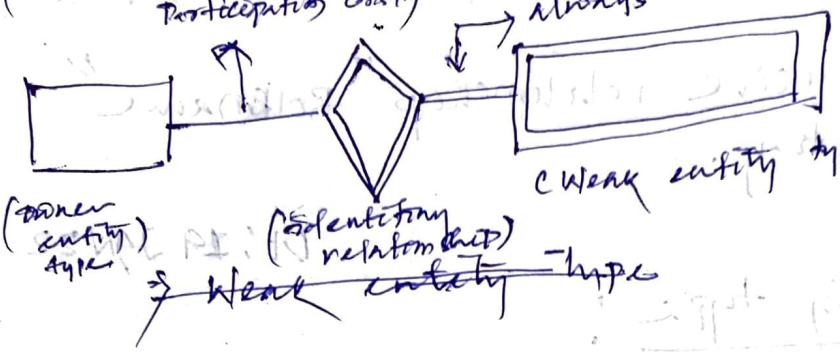
→ Entities belonging to weak entity type are identified by being related to specific entities of another entity type.

→ The entity type which identifies weak entity type is known as identifying entity type or owner entity type.

v v imp → The relationship that relates a weak entity type with its owner entity type is called identifying relationship.

This is represented by

(may be total or partial participation const.)



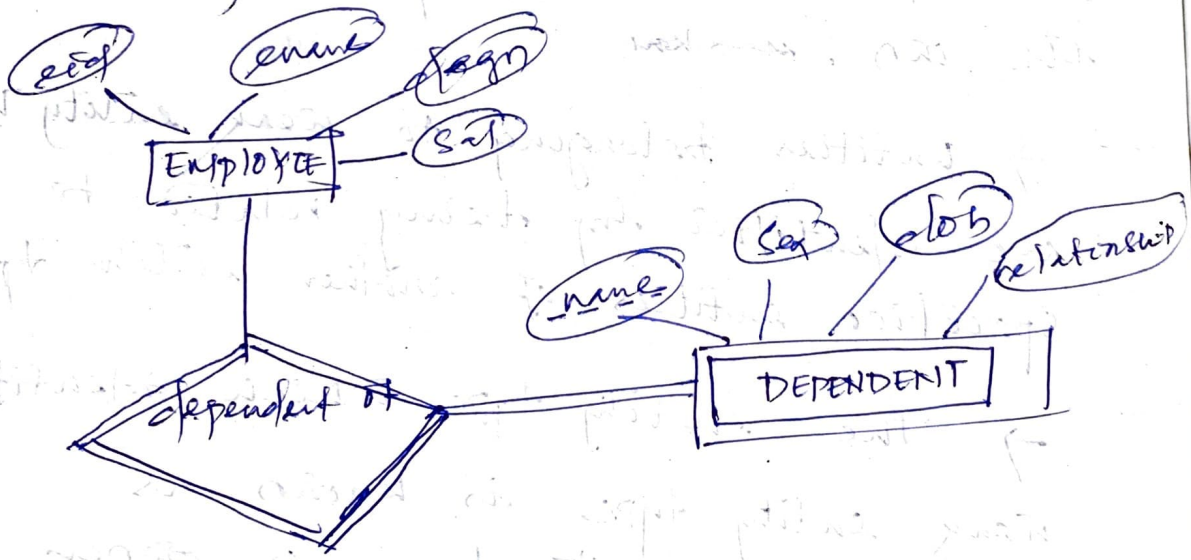
~~Weak entity type~~

Partial Key

→ A weak entity type normally has a partial key.

→ Partial key is represented by

→ Partial key is the attribute that can uniquely identify weak entities that are related to the owner entity.



Q. Consider a company database which keeps track of its employees, department and projects.

The company is organised into depart-ments. Each department has a

unique number, name and ~~the location~~ of the particular employee who manages the department. A dept. may have several locations.

A dept. controls no. of projects each of which has a unique name, unique number and a single location.

We store each employees name, id, address, salary, sex, and dob. An employee works for a department and works on different projects. We should track of managers of each department and direct supervisor of employees.

We want to keep track of dependents of each employee for insurance purpose, we keep each dependent name, sex, dob and rel'ship to the employee.

Based on the above information. Draw the E-R digram for the Company database.

20 JAN 22

Q: ER diagram of bank database

. E-R diagram of Hospital management

E-R diagram examples

- Library management system
- University
- Airline reservation system
- online food delivery
- online shopping

Relational Model

→ It represent a data-base as collection of relation

→ A relation is a table

→ A table is collection of rows and collumns with values.

	eid	ename	degn	sal	dno
t_1	e_1	xxx	mg		
t_2	e_2	xxy			
t_3	e_3	zzz			

Domain of attribute:

A set of values assign to an attribute is known as domain of the attribute.

dom(eid) = Varchar2(3) ← Any 3 alphanumeric characters

Relation Schema:

A relation schema $R(A_1, A_2 \cdots A_n)$ is made up of the Relation R and list of attributes $A_1, A_2, A_3 \cdots A_n$.

EMPLOYEE(eid, ename, dept, sal, dno)

↳ Relation Schema

EMPLOYEE (eid :
ename:
dob:
sal:
dept:
dno:

$R(A_1 A_2, A_3 \cdots A_n)$

Attribute A_i ($1 \le i \le n$) ∈ dom(A_i)

Relation State: (data is)

$R(A_1, A_2 \cdots A_n)$ → Rel^n schema

$t_1 = \langle$ 'e1', 'xxx', 'mgr', 35000, 1 $\rangle$

$t = \langle$ value of A_1, value of A_2 ---, value of A_n $\rangle$

$r(R) = \{t_1, t_2, t_3\}$

$r(R)$: a set of n-tuple

$t_i = \langle v_1, v_2, v_3 \cdots v_n \rangle$

v_1 → value of A_1, v_2 → value of A_2, v_n → value of A_n

*) A rel^n schema R is represented by

$R(A_1, A_2 \cdots A_n)$

where R = Relation name

$A_1, A_2 \cdots A_n$ = attributes

The relation state $r(R) = \{t_1, t_2 \cdots t_n\}$

$t_i = \langle v_1, v_2 \cdots v_n \rangle$

v_1 = value of A_1

v_2 = value of A_2

v_n = value of A_n

let $R(A_1, A_2 \cdots A_n)$ } A_2 of $R = R.A_2$

$S(A_1, A_2 \cdots A_n)$ } A_5 of $S = S.A_5$

R, S ← denotes Relation name

value of dno of tuple t_2 - 2

$t_2[A_2] = 2$; $t_2.A_5 = 2$ $t_2[dno] = 5$

The attribute A_j of tuple t_i is represented by $t_i[A_j]$ = value.

21 JAN 22

Degree of a Relation:

No. of attributes in a relation is known as / called degree of a Relation.

$R(A, B, C)$ → Degree = 3

$R(A, B, C, D)$ → Degree = 4

e.g: A relation with degree n can be represented by

$R(A_1, A_2, \cdots A_n)$

A relation with degree k can be represented by

$R(A_1, A_2 \cdots A_k)$

Schema based constraint:

① Domain constraint

The value of an attribute must come from it's domain.

② Key constraint

Basically in this constraint we discuss

ⓐ Super key
ⓑ key
ⓒ Candidate key
ⓓ primary key
ⓔ Alternate key

ⓑ key

→ uniqueness property
→ Distinct values

→ key is a set of attribute of a relation which is uniquely identifies tuples in that relation.

Mathematically, a key is a set of attribute of a relation $R(A_1, A_2 \dots A_n)$ iff for any two tuple t_i and t_j (if $i \neq j$) $t_i[K] \neq t_j[K]$.

② Super Key

→ A set of attribute Sk of a relation $R(A_1, A_2, \ldots A_n)$ is the super key of R iff for any two tuple t_i and t_j $(i \neq j)$

$$t_i[Sk] \neq t_j[Sa]$$

→ A superkey Sk specifies uniqueness constraint that no two distinct tuple in any state r(R) can have the same value for Sa.

→ A superkey can have redundant attribute i.e removal of attribute from the superkey set still has the uniqueness property.

eg: student(Rollno, name, cource, dob)

{Rollno, name, cource} - superkey

{Roll no, name} - superkey

Q/ What is the default super key of R(A,B,C,D,E)

Ans: {A, B, C, D, E}

③ Key

* Key is the minimal super key i.e you can't remove any attribute from the Key set.

(b) Key

→ set of attribute K of a relation $R(A_1, A_2 \dots A_n)$ is a key iff for any two tuple t_i, t_j $(i \neq j)$ then $t_i[K] \neq t_j[K]$.

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

What is the Key of R.

Ans:- can't be determined as value is not given.

eg: Student (Roll no, dob, name)

uniquely property — uniquely property

then {Roll no} {dob, name}

Key — Key

*) key are underlined.

$R(A_1, A_2, A_3 \dots A_i, A_j \dots A_k, A_{n-1}, A_n)$

Key = $\{A_2\}$, $\{A_j, A_k\}$

candidate key — candidate key

(c) Candidate Key :-

→ If more than one key present in a relation each key is known as a candidate key.

Primary Key

22 JAN 22

If one or more than one candidate key is present in a relation then one of the candidate key is choosen as as primary key.

*) It is choosen by database designer

Primary key has following properties

→ uniqueness property

i.e a set of attribute P_k of a relation $R(A_1, A_2 \ldots A_n)$ is a primary key if for any two tuples ti and tj $(i \neq j)$ $ti[P_k] \neq tj[P_k]$

→ Duplicate not allowed

(Primary

→ Primary key does not allow null values

→ Only one primary key is allowed for relations.

$R(A_1, A_2, \ldots A_i, A_j, A_p, A_{n-2}, A_{n-1}, A_n)$

Set keys of R are $\{A_i, A_j\}$, $\{A_p\}$, $\{A_{n-2}, A_{n-1}\}$

Candidate Keys

R(A, B, C, D)

Primary keys are {B}, {D} (Wrong)

Because for a relation only one primary key is possible.

Alternate key

The keys other than the primary key.

A

Composite Key

Relational schema

R(A1, A2, ... An) + Constraints

e.g. R(A1, A2 ... An) = R = {A1, A2 ... An}

Relational Database Schema :

Relational database schema is a collection of relational schema.

S = {R1, R2 ... Rn}

Valid state

A database state which satisfies all the constraints specified in the database.

③ INTEGRITY CONSTRAINT

INTEGRITY CONSTRAINT is two types

→ Entity Integrity

of table → Primary key value can't be null.

→ Referencial Integrity constraint

*) It is used to establish rel'ship among relations.

*) It is used to maintain consistancy among tuples in a relation.

*) It is described by the concept of foreign key.

Foreign key

→ Primary key of one table can be used as foreign key in another table torelate dom(Pk) = dom(Fk)

relative k size — data type k size

Primary key does not allow null values but unique key allow null values.

dependent table / child table

e.g. EMP | eid | ename | degn | sal | dno |

DEPT | Dno | Dname | Dloc | master table / parent table

dom(DEPT.dno) = dom(Emp.dno)

27.01.22

SELECT eid, ename, degn FROM EMP } SQL
WHERE sal > 25,000;

Relational algebra { $\pi_{eid, ename, degn}(\sigma_{sal>25000}(EMP))$

SQL based on relational algebra

1st SQL translate to Relational algebra.

Relational Algebra:

→ Basic set of operations for a relational Model.

→ Specifies the retrival request.
(the way in which you can access the data from database)

→ The result obtained after performing relational algebra operation on a relation produced a new relation

SELECT operation:

→ It retrives or selects a set of tuples/rows, which satisfies a given condition.

→ General Form is

σ ↗ select operation
(R) → Rel" name
⟨cond⟩ → cond"

→ Then the condition is of the form:

attribute op value / attribute op attribute

→ op is one of the operator among
>, ≥, <, ≤, =, ≠. (Mathmatical symbols)

→ More than one operation can be combined by AND (∧), OR (∨), NOT (¬).

Q. Retrive the informations of employees of dno = 4.

⇒ $\sigma_{dno=4}^{(EMP)}$

Q. Find the managers who are getting more than 5000.

⇒ $\sigma_{degn='mgr' \wedge Sal > 5000}^{(EMP)}$

Imp $deg(\sigma_c(R)) = deg(R)$

deg(R) = no. of attributes in R.

Project Operation :

→ Retrive certain columns from a Relation.

→ General form is

$$\pi_{\langle columnlist \rangle}^{(R)} \;/\; \pi_{\langle attributelist \rangle}^{(R)}$$

Q. Find eid, ename, Sal of all ~~employees~~ mangers.

$R_1 \leftarrow \sigma_{degn='mgr'}^{(EMP)}$

⇒ $\pi_{eid, ename, Sal}^{(R_1)}$

or// $\pi_{eid, ename, Sal}\left(\sigma_{degn='mgr'}^{(EMP)}\right)$

Q. Find eid, dept & sal of all managers who are working under dno. 5.

(i) $R_1 \leftarrow \sigma^{(Emp)}_{dept=mgr' \wedge dno=5}$

$\pi^{(R_1)}_{eid, dept, sal}$

$deg(\pi^{(R)}_{attributelist}) \leq deg(R)$

OR//

$\pi_{eid, dept, sal}\left(\sigma^{(Emp)}_{dept=mgr' \wedge dno=5}\right)$

SET operations:

π, σ unary operator

$\cup, \cap$ binary operators

UNION

INTERSECTION

SET DIFFERENCE / MINUS

CARTESIAN PRODUCT

The set operations UNION (∪), INTERSECTION (∩) & MINUS can be applicable on rels if the relations are union compatible.

UNION COMPATIBILITY :-

Two operations $R(A_1, A_2, \ldots A_n)$ & $S(B_1, B_2 \ldots B_n)$ are union compatible iff

$deg(R) = deg(S)$ i.e $n = m$

$dom(A_i) = dom(B_i)$ $(1 \leq i \leq n)$

i.e domain of corresponding attributes of R & S are same.

i.e $dom(R \cdot A_1) = dom(S \cdot B_1)$

$dom(R \cdot A_2) = dom(S \cdot B_2)$ & so on.

Q. $R(A_1, A_2, A_3, A_4)$ & $S(B_1, B_2, B_3)$

that is $R \cup S$, $R \cap S$, $R - S$.

Ans: Can't be applied as R & S aren't union compatible.

Q. Emp (eid, ename, dgn, sal, dno)

dependent (eid, dgn, dname, dsex, drel)

a. EMP-dependent → Find the employees with dgn who does not have any dependent.

Ans: They can't perform any operations $(\cup, \cap, -)$ because then are not union compatible

Emp have 'sal, dno' in dependent have 'dsex, drel' different attributes.

$Emp1 = \Pi_{eid, dgn}(Emp)$

$dependent1 = \Pi_{eid, dgn}(dependent)$

Now domain and degree are same.

$Emp1 \cup dependent1$ | $Emp1 \cap dependent1$

Emp1 - dependent1

$r \cup s = \{ t \mid t \in r \text{ or } t \in s \}$

$r \cap s = \{ t \mid t \in r \text{ and } t \in s \}$

$r - s = \{ t \mid t \in r \text{ and } t \notin s \}$

28 JAN 22

RENAME Operation

↳ To rename a relation / Renaming the rel?.

Symbol: $\rho_{(rhs)}$

Renaming a relation

(I) $\rho_S(R)$ ← The relation R is renamed to relation S.

(II) To rename the attributes of a rel? $R(A_1, A_2, A_3 \ldots A_n)$ to $R(B_1, B_2, \ldots B_n)$

$$\rho_{(B_1, B_2 \ldots B_n)}(R)$$

(III) Change or renaming $R(A_1, A_2, A_3 \ldots A_n)$ to $S(B_1, B_2 \ldots B_n)$

$$\rho_{S(B_1, B_2, B_3 \ldots, B_n)}(R)$$

Emp(eid, ename, dno)

DEPT(dno, dname)

Q. Find the eid, ename of employees who are working under 'research' dept.

Cartesian product (Cross product)

Symbol : X

↳ Though it is a set operation, but union compatibility may not be satisfied.

(Union compatibility is not requered to be satisfied)

↳ Result of X is a new relation which contains combinations of every tuple from one rel with every tuple from other relation.

P (1, 2, 3) S (4, 5)

$R \times S = \{(1,4), (1,5), (2,4), (2,5), (3,4), (3,5)\}$

Fig-1

Emp

eid	ename	dno.	
e_1	X	1	t_1
e_2	Y	2	t_2
e_3	Z	1	t_3

DEPT

dno.	dname	
1	Research	t_4
2	finance	t_5

$Emp = \{t_1, t_2, t_3\}$

$DEPT = \{t_4, t_5\}$

$Emp \times DEPT = (t_1, t_4) \{e_1, X, 1, 1, Research\}$

EMP × DEPT

eid	ename	dno	dno	dname	
e_1	x	1	1	Research	(t_1, t_4)
e_1	x	1	2	Finance	(t_1, t_5)
e_2	y	2	1	Research	(t_2, t_4)
e_2	y	2	2	Finance	(t_2, t_5)
e_3	z	1	1	Research	(t_3, t_4)
e_3	z	1	2	Finance	(t_3, t_5)

$$\Pi_{eid,\ ename}\left(\sigma_{EMP.dno = DEPT.dno \wedge dname = 'Research'}\left(EMP \times DEPT\right)\right)$$

N.B:

① Let $R(A_1, A_2 \ldots A_n)$ and $S(B_1, B_2 \ldots B_m)$

So here $deg(R \times S) = n+m$

② If R has i no. of tuples and S has j no. of tuples

i.e $|R| = i$ and $|S| = j$

$$|R \times S| = i \times j$$

JOIN operation

- Join
  - Inner Join
    - theta Join
    - Equi Join
    - Natural Join
  - outer Join
    - left outer Join
    - Right outer Join
    - Full outer Join

Joen.

Symbol: ⋈

Let $R(A_1, A_2 \dots A_n)$ ~~is a relation~~ and $S(B_1, B_2 \dots B_m)$ are two relations, then the Joen operation on R and S is represented by

$$R \underset{\text{cond}}{\bowtie} S$$

The result of Joen operation contain tuple from both the relations which satisfies the cond.

$deg(R) = n$ and $deg(S) = m$

$$deg(R \times S) = n + m$$

$$|R \times S| \leq |R| + |S|$$

Theta Joen:

Let $R(A_1, A_2, \dots A_n)$ & $S(B_1, B_2 \dots B_m)$ be two relations.

The θ Joen $R \& S$ can be represented by $R \underset{\theta}{\bowtie} S$ where θ is the Joen condition.

So to apply Joen we have a joining attribute such that

$$dom(R.A_i) = dom(S.B_j)$$

Here, A_i & B_j are the joining attributes of R & S respectively.

Cond^n is of the form

~~$A_i \theta B_j$~~ A_i op B_j

op is one of the operator among

$<, \geq, >, \leq, =, \neq$

Equi Join

If equality is the only operator in θ Join it is known as equi Join.

Ex: Emp (eid, ename, dgn, sal, dno)

DEPT (dno, dname, dloc)

Q. Find the employee who are working under research department.

$R_1 \leftarrow$ Emp $\bowtie$ DEPT
Emp.dno = DEPT.dno

Emp $\underset{dno}{\bowtie}$ DEPT

Result $\leftarrow \sigma_{dname='Research'}(R_1)$

Q. Find eid, sal of employees who are working under research dept.

$\pi_{eid, sal}\left(\sigma_{dname='Research'}(\text{Emp} \underset{dno}{\bowtie} \text{DEPT})\right)$

Natural Join:

↳ Represented by the Symbol *.

Let $R(A_1, A_2, A_3 \dots A_n)$ and $S(B_1, m_2, A_3 \dots B_m)$ be two relations with ~~domain of~~

$dom(R.A_3) = dom(S.A_3)$

i.e A_3 is the joining attribute.

Natural join of R and S is represented by Q.

$Q = R * S$

i.e Q has attributes $(A_1, A_2, A_3 \dots A_n, B_1, B_2, B_4 \dots B_m)$

eg. Natural Join of EMP and DEPT

EMP * DEPT

eid	ename	dno.	dname

~~see~~ Q ~~has~~

If $deg(R) = n$ and $deg(S) = m$

then $deg(R * S) = (n+m) - 1$

Examples:

Given 3 relational schemas as

SAILOR (sid, sname, rating, age)

BOAT (bid, bname, bcolour)

RESERVE (sid, bid, day)

Q1. Find the names of SAILORS who have reserved boat 103.

$R_1 \leftarrow (SAILOR \underset{sid}{\bowtie} RESERVE)$

$R_2 \leftarrow (\sigma_{bid=103}(R_1))$

$Result \leftarrow \pi_{sname}(R_2)$

or

$\pi_{sname}(\sigma_{bid=103}(SAILOR \underset{sid}{\bowtie} RESERVE))$

Q. Find the id, names of sailors who have reserved the ~~boat~~ boat 'Interlake'.

$R_1 \leftarrow \sigma_{bname='Interlake'}(BOAT \underset{bid}{\bowtie} RESERVE)$

$Result \leftarrow \pi_{sid, sname}(SAILOR \underset{sid}{\bowtie} R_1)$

and rating

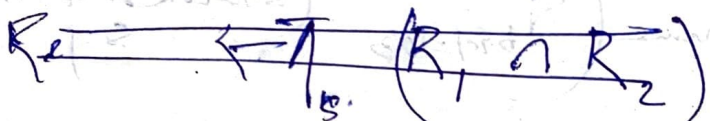
Q. Find the names and ratings of sailors who have reserved a red boat.

$\pi_{sid, rating}(SAILOR \underset{sid}{\bowtie} (\sigma_{bcolour='red'}(BOAT \underset{bid}{\bowtie} RESERVE)))$

Q. Find the names of sailor who have reserved a red boat and a green boat.

$$R_1 \leftarrow \sigma_{bcolor = 'red'}(BOAT \bowtie_{bid} RESERVE)$$

$$R_2 \leftarrow \sigma_{bcolor = 'green'}(BOAT \bowtie_{bid} RESERVE)$$



$$R_3 \leftarrow (R_1 \cap R_2)$$

$$Result \leftarrow \pi_{sname}(R_3 \bowtie_{sid} SAILOR)$$

Q. Find the names of sailor who have reserved a red or a green boat

Q. Find the names of sailor who have reserved a red boat but not green boat.

Q. Find the colour of boat Reserved by Mr. John.

(1) $\Pi_{bcolor}\left(BOAT \bowtie \left(\sigma_{sname=John}\left(SAILOR \underset{sid}{\bowtie} RESERVE\right)\right)\right)$

(2) $\left(\Pi_{sname=John}(SAILOR)\right) \bowtie RESERVE$

2 FEB 22

Q. Name of the sailor whose age > 20 who have not reserved a red boat.

$R_1 \leftarrow \sigma_{age>20}\left(SAILOR \underset{sid}{\bowtie} RESERVE\right)$

$R_2 \leftarrow \sigma_{bcolor=RED}\left(R_1 \underset{bid}{\bowtie} BOAT\right)$

~~$\Pi_{sname}(R) - R$~~

~~$\Pi_{sname}(R_2)$~~

$R_3 \rightarrow R_1 \underset{bid}{\bowtie} BOAT$

$\Pi_{sname}\left(R_3 - R_2\right)$ → Result

Q. Find the names of Sailor who have reserved at least one boat.

$$R_1 \leftarrow (SAILOR \underset{Sid}{\bowtie} RESERVE)$$

$$R \; \Pi_{Sname} \; \sigma_{bid \neq 0} (R_1)$$

$$\Pi_{Sname} (SAILOR \underset{Sid}{\bowtie} RESERVE)$$

OUTER JOIN Operation :-

1. Left outer Join ($\bowtie$)
2. Right outer Join ($\bowtie$)
3. Full outer Joint ($\bowtie$)

Left outer Join ($\bowtie$)

R

Col 1	Col 2
A	1
B	2
C	3
D	4

S

Col 1	Col 2
A	A1
C	C1
D	D1

R ⟕ S

Col 1	Col 2	Col 1	Col 2
A	1	A	A1
B	2	Null	Null
C	3	C	C1
D	4	D	D1

Q. Find the sailors who have reserved all the boats.

Division Operation

For phrases like all, every division operation is used.

Consider two relations R and S where R has two attributes X and Y (i.e R(X, Y)) and S has just attribute Y with same domain that of R i.e dom(R.Y) = dom(S.Y)

We define division operation by R/S or R ÷ S such that R/S contains set of all X values such that ∀ Y ∈ S ∃ a tuple ⟨a, y⟩ ∈ R.

eg.

R(Name, Pen)

Name	Pen
Ram	Black
Sita	Blue
Hari	Red
Sita	Green
Hari	Blue
Sita	Red
Ram	Green

S_2

Pen
Black
Green

S_1(Pen)

Pen
Black
Blue
Red
Green

Sita	black
Ram	Red

$R \div S_1 =$

Bname
~~Rsid~~
Sita

$R \div S_2 =$

Bname
Ram
Sita

(7)

$R_1 \leftarrow \pi_{sid, bid}$ (RESERVE)

$R_2 \leftarrow \pi_{bid}$ (BOAT)

$R_3 \leftarrow R_1 / R_2$ (sid of sailors who have received all the boats)

RESULT $\leftarrow \pi_{sname}(R_3 \bowtie$ SAILOR$)$ (sid)

Complete set of Relational Algebra operation.

Q. Find the SAILORS who have reserved all the boats named 'BLUE_HAVEN'.

$$R_1 \leftarrow \Pi_{sid,bid}\left(\sigma_{bname\ blue\_haven}\left(BOAT \bowtie RESERVE\right)\right)$$ maybe wrong

$$R_2 \leftarrow \Pi_{sname}\left(R_1 \bowtie SAILOR\right)$$

Sir //

$$R_1 \leftarrow \Pi_{sid,bid}(RESERVE)$$

$$R_2 \leftarrow \Pi_{bid}\left(\sigma_{bname='blue\_haven'}(BOAT)\right)$$

$$R_3 \leftarrow R_1 \div R_2$$

$$Result \leftarrow \Pi_{sname}\left(R_3 \bowtie SAILOR\right)$$

Dt: 4.2.22

Functional Dependency:

→ A functional dependency bet^n two sets of attributes x and y of a rel^n R is represented by $x \to y$.

Functional Dependency is a constraint which is specified on any rel^n state $r(R)$ and it is like for any two tuples t_1 & t_2

if $t_1[x] = t_2[x]$

& $t_1[y] = t_2[y]$

then $x \to y$

i.e the value of y-component of a tuple is determined by the value of x-component.

Otherwise we can say that x-component of a tuple uniquely/functionally determine the value of y-component of that tuple.

N.B.

(1) $x \longrightarrow y$

L.H.S of F.D = x

R.H.S of F.D = y

(2) If $x \to y$, we can't say that $y \to x$.

Inference Rules for Functional Dependencies:

Inference rules are used to derive new functional dependencies from a given set of functional dependences.

IR 1: Reflexive Rule :-

If $y \subseteq x$ $(x \supseteq y)$

then $x \rightarrow y$, where $x, y \subseteq R$

IR 2: Augmentation Rule :-

$\{x \rightarrow y\} \models \{xz \rightarrow yz\}$ (If then)

where $x, y, z \subseteq R$

IR 3: Transitive Rule :-

$\{x \rightarrow y, y \rightarrow z\} \models \{x \rightarrow z\}$

where $x, y, z \subseteq R$

N.B

Reflexive rule, Augmentation rule & Transitive rule are known as "Armstrong's inference rule" for Fds.

* Armstrong's Inference rules are Sound and complete.

IR 4 : Decompositon Rule :

$\{x \to yz\} \models \{x \to y\}$ $x, y, z \subseteq R$

then also $\{x \to z\}$

IR 5 : Union or additive Rule :

$\{x \to y, x \to z\} \models \{x \to yz\}$

$x, y, z \subseteq R$

IR 6 : Pseudo transitive Rule :

$\{x \to y, wy \to z\} \models \{wx \to z\}$

$x, y, z \in R$

e.g: $F = \{A \to B, B \to C, CD \to E\}$

A/c IR3 $A \to C$, $F = \{A \to B, B \to C, A \to CD\}$

$AD \to A$, $A \subseteq AD$

$AD \to D$, $D \subseteq AD$

} A/c IR1 Reflexive

$A \to B$, $B \to C$

$A \to C$ → attributes closure the other fds thus can be derived from the attributes.

Closure set of Functional Dependency:

The set of all functional dependencies that includes F & all the functional dependencies that can be inferred from F is known as closure of F.

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

$= \{A \to B,\ B \to C,\ A \to C\}$ ⟵ inferred from F

Given set of F.ds

closure of F (Closure set of f.ds) is represented by F^+

$F^+ = F \cup \{$the set of F.ds that can be (derived) inferred from F$\}$

F^+ = closure set of f.ds

eg. $F = \{A \to B,\ B \to C\}$ $A^+ = \ldots$ $B^+ = \ldots$

A^+ → attribute closure of A

eg. $F = \{A \to B,\ B \to C,\ CD \to E\}$

$F^+ = \{A \to B,\ B \to C,\ CD \to E,\ AD \to BC\}$

A^+ = What other attributes can be derived from A.

Dt: 8 Feb 23

Attribute closure

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

Find A^+.

Sol: Consider $A \to B$

$A \subseteq A$

$\to A^+ = AB$

Consider $B \to C$

$B \subseteq A^+$

$\to A^+ = ABC$

A can derive A	$A \to A$
A can derive B	$A \to B$
A can derive C	$A \to C$

Consider $B \to C$

$B^+ = B$

$B \subseteq B$

$B^+ = BC$

Q. Given a set of F.ds $F = \{A \to B, B \to C, CD \to$

Find A^+, B^+, D^+, AB^+, AD^+, CD^+

A^+ ~~of~~ ans: Finding A^+

Consider $A \to B$

$A \subseteq A$

$\Rightarrow A^+ = AB$

Consider $B \to C$

$B \subseteq A^+$

$\Rightarrow \boxed{A^+ = ABC}$

Finding B^+

Consider $B \to C$

$B \subseteq B$

$\Rightarrow \boxed{B^+ = BC}$

Finding D ~~A~~ $^+$

~~Consider~~ $D^+ = D$

Finding AB^+

Consider $A \to B$

$A \subseteq AB$,

$AB^+ = AB$

Consider $B \to C$

$B \subseteq AB^+$

$AB^+ = ABC$

Finding AD^+

Consider $A \to B$

$A \subseteq AD$

~~$AD^+ = ABD$~~ ~~$ABCDE$~~

$AD^+ = ABD$

Consider $B \to C$

$B \subseteq AD^+$

$AD^+ = ABCD$

Consider $CD \to E$

$CD \subseteq AD^+$

$\Rightarrow AD^+ = ABCDE = R$ (Here AD is the Key attribute)

Finding CD^+

consider $CD \to E$

$CD \subseteq CD$

$\Rightarrow CD^+ = CDE$

Q. Given $F = \{A \to BC, CD \to E, E \to C, D \to AEH, ABH \to BD, DH \to BC\}$

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

Find $A^+, AE^+, AB^+, ABH^+, D^+$

Ans

Finding A^+

consider $A \to BC$

$A \subseteq A^+$

$\Rightarrow A^+ = ABC$

Consider D^+

~~Consider~~

~~Consider $CD \to E$~~

~~$D \subseteq CD$~~

~~$\Rightarrow D^+ = DE$~~

Consider

Finding D^+

Consider $D \subseteq D$ $D \to AEH$

$D \subseteq D$

$D^+ = DAEH$

Consider $A \to BC$

$A \subseteq D^+$

$D^+ = ABCDEH = R$

Finding AE^+

Initially $AE^+ = AE$

Consider $A \to BC$

$A \subseteq AE^+$

$AE^+ = ABCE$

Consider $E \to C$

$E \subseteq AE^+$

$\to AE^+ = ABCE$

Consider $ABH \to C$

$ABH \subseteq AE^+$

$\to AE^+ = ABCE$

Consider $DH \to C$

$DH \subseteq AE^+$

$\to AE^+ = ABCE$

Consider $D \to AEH$

$D \subseteq AE^+$

$CD \quad AE^+ = ABCE$

Finding AD^+

consider $A \to BC$

$A \subseteq AD$

$\to AD^+ = ABCD$

Consider $CD \to E$

$CD \subseteq AD^+$

$\to AD^+ = ABCDE$

Consider $D \rightarrow AEH$

$D \subseteq AD^+$

$\Rightarrow AD^+ = ABCDEH = R$

Finding BD^+

$BD^+ = ABDEH$

$BD^+ = ABCDEH$

Find ADH^+

$ADH^+ = ABCHDE = R$

$A^+ = ABC$

$B^+ = B$

$H^+ = H$

$AH^+ = ABCHDE$

$BH^+ = BH$

$AB^+ = ABC$

Dt: 12 FEB 22

COVER :-

A set of f.ds F is said to cover another set of f.ds G if all the f.ds in G is in F^+ i.e all the f.ds in G can be inferred from F.

$\Rightarrow$ Then we can say F covers G.

eg. $F = \{A \to B, B \to C, B \to D\}$

$G = \{A \to C, A \to D\}$

Check whether F covers G or not.

Ans: Consider $A \to C$ in G :-

Find A^+ under F

$A^+ = ABCD$

$\Rightarrow C \subseteq A^+$

$\Rightarrow A \to C$ can be inferred from F. — (1)

Consider $A \to D$ in G :-

Find A^+ under F.

$A^+ = ABCD$

$\Rightarrow D \subseteq A^+$

$\Rightarrow A \to D$ can be inferred from F — (2)

So from (1) and (2), all the f.ds of G can be inferred from F.

$\Rightarrow$ F covers G.

Equivalent set of functional dependencies :-

Two sets of f.ds F & G are equivalent if $F^+ = G^+$ i.e all the f.ds in G can be inferred from F and all the f.ds in F can be inferred from G.

i.e $F \equiv G$.

Q. $F = \{A \to B, B \to C, B \to D\}$

$G = \{A \to C, A \to D\}$

Check whether F & G are equivalent or not.

Ans: Let's check whether F covers G.

Consider $A \to C$ in G:

Find A^+ under F

$= ABCD$

$\Rightarrow C \subseteq A^+$

$\Rightarrow A \to C$ can be inferred from F — (i)

Consider $A \to D$ in G:

Find A^+ under F

$= ABCD$

$\Rightarrow D \subseteq A^+$

$\Rightarrow A \to D$ can be inferred from F. — (ii)

So from eq^n (i) & (ii), all the FDs of G can be inferred from F.

So F covers G — (iii)

Let us check whether G covers F :-

Consider $A \to B$ in F :-

Find A^+ under G

$= ACD$

$B \notin A^+$

$\Rightarrow A \to B$ can't be inferred from G.

$\Rightarrow$ G can't cover F.

$\Rightarrow F \not\equiv G$.

Q. Given $F = \{A \to B, AB \to C, D \to AC, D \to E\}$

$G = \{A \to BC, D \to AE\}$

Check whether F & G are equivalent or not.

Ans: Let us check whether F covers G :-

Consider $A \to BC$ in G:

Find A^+ under F

$= ABC$

$\Rightarrow BC \subseteq A^+$

$\Rightarrow A \to BC$ can be inferred from F. — (I)

Consider $D \to AE$ in G:

Find D^+ under F

$= DACEB$

$\Rightarrow AE \subseteq D^+$

$\Rightarrow D \to AE$ can be inferred from F. — (II)

As (I) & (II) can be inferred from F

$\Rightarrow$ F Covers G — (III)

Now let's check whether G covers F :-

Consider $A \to B$ in F

Find A^+ under G

$= ABC$

$\Rightarrow B \subseteq A^+$

$\Rightarrow A \to B$ can be inferred from G (iv)

Consider $AB \to C$ in F

Find AB^+ under G

$= ABC$

$\Rightarrow C \subseteq AB^+$

$\Rightarrow AB \to C$ can be inferred from G. (v)

Consider $D \to AC$ in F

Find D^+ under G

$= ACDEB$

$\Rightarrow AC \subseteq D^+$

$\Rightarrow D \to AC$ can be inferred from G. (vi)

Consider $D \to E$ in F

Find D^+ under G

$= ABCDE$

$\Rightarrow E \subseteq D^+$

$\Rightarrow D \to E$ can be inferred from G. (vii)

So eqn (IV), (V), (VI), (VII) can be inferred from G.

So G covers F ——— (VIII)

from eqn (III) & (VII),

$F \equiv G$.

(*) Non-Redundant covers:-

Given a set of fds F.

Let redundant set of fds in $F = F_{redundant}$

Non-redundant cover $F = F - F_{redundant}$

Ex:- $F = \{A \to B, A \to C, B \to C\}$

Find non-redundant cover of F.

Ans:-

Check whether $A \to B$ is redundant

Find A^+ under $F - \{A \to B\}$

$= AC$

$\Rightarrow B \notin A^+$ under $F - \{A \to B\}$

$\therefore A \to B$ is non-redundant.

Check whether $A \to C$ is redundant

Find A^+ under $F - \{A \to C\}$

$= ABC$

$\Rightarrow C \subseteq A^+$ under $F - \{A \to C\}$

$\therefore A \to C$ is redundant

$\therefore F = F - \{A \to C\}$

$\therefore F = \{A \to B, B \to C\}$

Check whether $B \to C$ is redundant:

Find B^+ under $F - \{B \to C\}$

$= B$

$C \notin B^+$ under $F - \{B \to C\}$

$\Rightarrow B \to C$ is non-redundant in F.

Hence the non-redundant Covers of F is $F = \{A \to B, B \to C\}$

15 FEB 22

eg. Given $F = \{A \to B, C \to D, AC \to E, BD \to E\}$

Find the non-redundant Cover of F.

Ans :- Check whether $A \to B$ is redundant

Find A^+ under $F - \{A \to B\}$

$\Rightarrow A^+ = A$

$\Rightarrow B \notin A^+$

$\Rightarrow A \to B$ is non-redundant.

Check whether $C \to D$ is redundant

Find C^+ under $F - \{C \to D\}$

$\Rightarrow C^+ = C$

$\Rightarrow D \notin C^+$

$\Rightarrow C \to D$ is non redundant

Check whether $AC \to E$ is redundant

Find AC^+ under $F = \{AC \to E\}$.

$AC^+ = ABCDE$

$\Rightarrow E \subseteq AC^+$

$\Rightarrow AC \to E$ can be inferred from $F = \{AC \to E\}$

$\Rightarrow AC \to E$ is redundant.

Now $F = \{A \to B, C \to D, BD \to E\}$

Check whether $BD \to E$ is redundant

Find BD^+ under $F = \{BD \to E\}$

$\Rightarrow BD^+ = BD$

$\Rightarrow E \notin BD^+$

$\Rightarrow BD \to E$ is non redundant.

Thus the non redundant cover of F is given by $F = \{A \to B, C \to D, BD \to E\}$

LEFT-Reduction For Removing extraneous attribute from the left hand side of F.d.

The Left-Reduction process is used to remove extraneous attribute from L.H.S of each F.D if the L.H.S contains more than one attribute.

Algorithm:

not necessary For each F.D $\alpha \to A$ in F:

For each attribute $B \in \alpha$

If $\{F - \{\alpha \to A\} \cup \{(\alpha - B) \to A\}\}^+ = F^+$

Then replace

e.g. ~~Given~~ Do Left-Reduction on the following set of F.D.s

$F = \{A \to D, BC \to AD, C \to B, E \to A, E \to D\}$

Check whether $BC \to D$ is redundant

Find BC^+ under F

$BC^+ = BC$

Sol: Consider $BC \to AD$

~~$B^+ = B$~~ Find B^+ under F.

$\Rightarrow B^+ = B$

$\Rightarrow AD \notin B^+$

$\Rightarrow$ ~~$B \to D$~~

Now Find C^+ under F

$C^+ = CBD$

After left reduction the set is given by $F = \{A \to D, C \to AD, C \to B, E \to A, E \to D\}$

THE CANONICAL COVER / MINIMAL COVER / MINIMAL SET OF FDs.

STEP-1 :

R.H.S of each F.Ds should contain single attribute.

i.e for each F.D $x \to \{A_1, A_2, \ldots A_n\}$ in F can be represented as $\{x \to \{A_1\}, \{x \to \{A_2\}$

$\ldots \ldots \ldots \{x \to \{A_n\}$

Step-2 :

Do left reduction from the set of F.D obtained from step-1.

Step-3 :

Find non-redundant cover from the set of F.D obtained form step-2.

Example

Q. Find the canonical / minimal cover of

$F = \{A \to D, BC \to AD, C \to B, E \to A, E \to D\}$.

Ans : Step-1 (Right hand side of each F.D should contain single attribute)

So Now $F = \{A \to D, BC \to A, BC \to D, C \to B, E \to A, E \to$

Step-2 (Left reduction on the set of F.D obtained from step-1)

Consider $BC \to A$

Find B^+ under F

$C = \langle BD \rangle$

$\to B^+ = B$

$\to A \notin B^+$

Find C^+ under F

$\to C^+ = ABCD$

$\to A \subseteq C^+$

$\to C \to A$ can be inferred.

Consider $BC \to D$

Again $B^+ = B$

$\to D \notin B^+$

And $C^+ = ABCD$

$D \subseteq C^+$.

$\to C \to D$ can be inferred.

It is clear that B is extraneous in $BC \to A$ and $BC \to D$.

So F can be

Now $F = \{A \to D, C \to A, C \to D, C \to B, E \to A, E$

Step-3 : (non-redundant cover of F obtained from step-2)

Check whether $A \to D$ is redundant.

Find A^+ under $F - \{A \to D\}$

$\to A^+ = A$

$\to D \notin A^+$

$\to A \to D$ is non-redundant.

Check whether $C \to A$ is redundant

Find C^+ under $F - \{C \to A\}$

$\to C^+ = CDB$

$\to A \notin C^+$

$\to C \to A$ is non-redundant

Check whether $C \to D$ is redundant

Find C^+ under $F - \{C \to D\}$

$\to C^+ = CABD$

$\to D \subseteq C^+$

$\to C \to D$ can be inferred from $F - \{C \to D\}$

$\to C \to D$ is redundant in F.

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

Check whether $C \to B$ is redundant

Find C^+ under $F - \{C \to B\}$

$\to C^+ = ACD$

$\to B \notin C^+$

$\to C \to B$ is non redundant.

Check whether $E \to A$ is redundant

Find E^+ under $F - \{E \to A\}$

$\Rightarrow E^+ = ED$

$\Rightarrow A \notin E^+$

$\Rightarrow E \to A$ is non-redundant.

Check whether $E \to D$ is redundant.

Find E^+ under $F - \{E \to D\}$

$\Rightarrow E^+ = EAD$

$\Rightarrow D \subseteq E^+$

$\Rightarrow E \to D$ can be inferred from $F - \{E \to D\}$

$\Rightarrow E \to D$ is redundant

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

Thus the non-redundant cover of F is given by $F = \{A \to D, C \to A, C \to B, E \to A\}$

So the canonical cover or minimal cover of $F = \{A \to D, C \to A, C \to B, E \to A\}$.

dt 16 FEB 22

N.B:- The canonical cover or minimal cover is also known as irreducable set of F.D.

Q. Find the minimal set of F.D

$F = \{ B \to A, BD \to A, AB \to D \}$

Soln:

Step-2: (left-reduction)

Consider $AB \to D$

Find A^+ under F

$\Rightarrow A^+ = A$

$\Rightarrow D \notin A^+$

Find B^+ under F

$\Rightarrow B^+ = AB$

$\Rightarrow D \subseteq B^+$

$\Rightarrow B \to D$ can be inferred.

It is clear that A is extraneous in $AB \to D$

Now $F = \{ B \to A, D \to A, B \to D \}$

Step-3 (non-redundant)

Check wheather $B \to A$ is redundant:

Find B^+ under $F - \{B \to A\}$

$\Rightarrow B^+ = BDA$

$\Rightarrow A \subseteq B^+$

$\Rightarrow B \to A$ can be interred from $F - \{B \to A\}$

$\Rightarrow B \to A$ is redundant in F.

Now $F = \{D \to A, B \to D\}$

Check wheather $D \to A$ is redundant

Find D^+ under $F - \{D \to A\}$

$\Rightarrow D^+ =$ ~~ADD~~ D

$\Rightarrow A \not\subseteq D^+$

$\Rightarrow D \to A$ is non-redundant.

check whether $B \to D$ is redundant

Find B^+ under $F - \{B \to D\}$

$\Rightarrow B^+ = B$

$\Rightarrow D \not\subseteq B^+$

$\Rightarrow B \to D$ is non-redundant.

So the connical cover or minimal cover of $F = \{D \to A, B \to D\}$

22 FEB 22

Key Finding Algorithm :

→ T

→ Take different combination of attribute present.

(Provided the set should'nt contain any extraneous (attribute).

Q. Given R(a, b, c, d, e, f)

F = {A → BC, CD → E, E → C, D → AEH, ABH → BD, BH → BC}

Find the candidate keys of R / keys of R.

Ans:

Consider A → BC

$A^+ = ABC$

$CD^+ = ACDEHB = R$

$E^+ = EC$

$D^+ = AEHDBC = R$

⇒ D is a key.

$ABH^+ = ABHCD$

$A^+ = ABC \neq R$ $\quad H^+ = H \neq R$

$B^+ = B \neq R$

$AB^+ = ABC \neq R$ $\quad AH^+ = ABCDEH = R$

$BH^+ = BCH$

$\Rightarrow AH$ is a key.

Hence candidate keys of R is D, AH.

Q. Given a Relation BOOK(Book_titel, author_name, Book_type, list_price, Author_affile, Publisher)

And a set of fds is given by,

$F = \{\{Book\_titel\} \rightarrow \{Book\_type, publisher\}, \{Book\ type\} \rightarrow \{list\_price\}$

$\{Authore\_name\} \rightarrow \{Author\_affil\}\}$

Find the key.

Ans. $\{Book\_titel\}^+ = \{Book\ titel, Publisher, Book\_type, listprice\}$ ≠ Book

$\{Book\_type\}^+ = \{listprice, Book\ type\} \neq$ Book

$\{Authore\_name\}^+ = \{Authore\_name, Auth\_affil\}$

≠ Book

METHOD #2

→ Find the attribute present in L.H.S of each F.Ds in F.

→ Find the attribute present in R.H.S of each F.Ds in F.

→ Find the attribute present in both the sides of F.Ds in F.

→ Find the attributes which are present only in the R.H.S.

(R.H.S - Both the sides)

→ The attributes which are present only in the R.H.S can't be part of any key.

→ Find the attribut which are present only in the L.H.S.

(L.H.S - Both the sides)

There two cases arises

Case-1

Take diffferent combinations of attribute present only in the L.H.S.

of the closer of these attributes = R, these set of attributes are the Key.

Case-2

(If case-1 is not satisfied)

Take different combinations of attribute LHS and both the sides of F.Ds in $F.

If Closer of the attributes = R, then these set of attributes are keys.

Ans: LHS : {Book_titel, Book_type, Author_name}

Both: {Book_type}

only LHS = {Book_titel, Author_name}

{Book_titel}$^+$, Author_name}$^+$

=

Q2. R(M, Y, P, MP, C)

$F = \{ \{M\} \to \{MP\}, \{M, Y\} \to \{P\}, \{MP\} \to \{C\} \}$

Find keys.

Soln:

$\{M\}^+ = \{MPC\} \neq R$

$\{MY\}^+ = MPYC = R$

$\{MP\}^+ = MPC \neq R$

$\{Y\}^+ = Y \neq R$

$\Rightarrow$ MY is a key (Primary key)

Q3. R(A, B, C, D, E, F)

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

Find keys?

Soln:

$C^+ = \{CF \neq R$ $EC^+ = EABCDF = R$

$E^+ = EAB \neq R$ $A^+ = AB \neq R$

$C^+ = CF \neq R$

Hence ECt is the key.

Q4. Book (Book_titel, Author_name, Book_type, List_Price, Author_Affil, Publisher)

$F = \{ \{Book\_titel\} \rightarrow \{Book\_type, Publisher\}, \{Book\_type\} \rightarrow \{List\_Price\}, \{Author\_name\} \rightarrow \{Author\_Affil\} \}$

Soln:

~~$\{Book\_titel\}^+ = Bo$~~

L.H.S

$\{ Book\_titel, Book\_type, Author\_name \}$

Both

$\{ Book\_type \}$

Only LHS

$\{ Book\_titel, Author\_name \}$

$\{Book\_titel, Author\_name\}^+$

$= R$

Q5. $R(A, B, C, D, E, H)$

$F = \{ A \to B,\ BC \to D,\ E \to C,\ D \to AH \}$

Soln:

$A^+ = \{AB\} \neq R$

$BC^+ = BCDAH \neq R$

$E^+ = C \neq R$

$D^+ = AHDB \neq R$

LHS

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

Both

$\{B, D, E, AH\}$ $\{D, B, A, C\}$

Only LHS

$\{A\ BC\ E\}$

$\{A, BC\}^+ = ABCDH \neq R$

$\{BC, E\}^+ = BCD \neq R$

$\{A, E\}^+ = ABCDH = R$

So $\{A, E\}$ is the key.

Q.6 $R\{A, B, C, D, E, H\}$

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

$(EH)^+ = ABCDEH = R$

AEH is the key

EDH

BEH

Q5. $R(A, B, C, D, E, H)$

$F = \{ A \to B, BC \to D, E \to C, D \to AH \}$

Soln:

LHC : $\{ A, B, C, E, D \}$

Both the side : $\{ A, B, C, D \}$

$A^+ = AB \neq R$

$AC^+ = ACBDH \neq R$

$AE^+ = ABCDHE = R$

$AB^+ = ADH \neq R$

$BC^+ = ABCDH \neq R$

$BD^+ = AHBD \neq R$

$CD^+ = ABDCH \neq R$

$ED^+ = ADHEBC = R$

$BE^+ = ABCDEH = R$

$CE^+ = CE \neq R$

So the keys are $\{AE\}, \{ED\}, \{BE\}$

23 FEB 22

NORMALISATION :

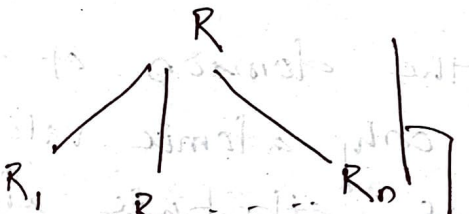
Process of decomposition of complex relations into smaller relations.

Assignment

① What are the anomalies discuss each anomali with example.

⇒ Normalisation is the process of decomposing a complex rel" into smaller relation based on normal form cond's

→ The goal of normalisation is to minimize
i) ~~minimising~~ redundancy
ii) Avoid anomalies
*) Insert anomali
*) update anomali
*) delet anomali



↳ This process of decomposition is known as normalisation.

→ The reverse process of normalisation is denormalisation.

DENORMALISATION :

The reverse process of normalisation is denormalisation.

i.e getting the original rel^n by joining the decomposed rel^n is known as denormalisation.

Different Normal forms are :-

Based on functional dependencies.
- → First normal form (1NF)
- → Second Normal Form (2NF)
- → Third Normal Form (3NF)
- → Boyce codd Normal Form (BCNF)
- → Fourth Normal Form (4NF) } Based on multivalued dependencies
- → Fifth Normal Form (5NF) } Based on Join dependencies

FIRST NORMAL FORM (1NF)

~~(It does not allow~~ .

It states that the domain of an attribute must include only atomic values and the value of each attribute in a tuple must be a single value from its domain in a relation R.

1NF does not permit composite and multivalued attributes or their combinations for a tuple in a relation R.

⇒ Prime attribute :-

An attribute of a rel^n R is said to be a prime attribute (candidate) if it is part of any key of R.

eg R (A, B, C, D)

i.e {A, B} is the key

then prime attribut are {A}, {B}

⇒ Full functional dependency :-

A function dependency $X \to Y$ is known as Full F.D if removal of any attribute {a} from X does not hold the F.D any more

i.e $X - \{a\} \to Y$ does not hold F.D

(Here X is a set)

⇒ Trivial Functional Dependency :-

A F.D $X \to Y$ is said to be trivial if $Y \subseteq X$.

It is non trivial if $Y \not\subseteq X$.

SECOND NORMAL FORM (2NF)

It is based on the concept of prime attribute and Full F.D.

Def^n: A Relation is said to be in 2NF if i) It is in 1NF.

ii) Every non-prime attribute of R is fully F.D on any key of R.

e.g. Given a Relation EMP_PROJECT (eid, Pno, hours, ename, Pname, Ploc)

and a set of F.Ds F is given by

{eid, Pno} → {hours}

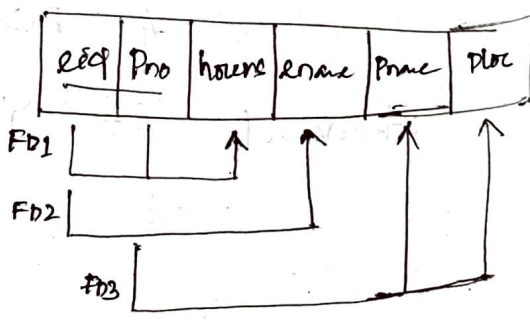
{eid} → {ename}

{Pno} → {Pname, Ploc}

decompose the above rel^n to 2NF rel^n.

Ans:

Key {eid, Pno}



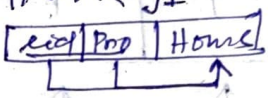
First you have to find out the key by applying finding algorithm, we have the key = {eid, Pno}

→ Here FD_1 i.e $\{Eid, Pno\} \rightarrow \{Hours\}$ satisfies 2NF conditions as the non-prime attribute $\{Hours\}$ is fully functional dependent on the key $\{Eid, Pno\}$

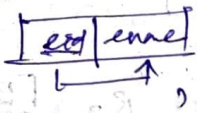
→ FD_2 i.e $\{Eid\} \rightarrow \{Ename\}$ violates 2NF conditions as $\{Ename\}$ partially dependent on the key.

→ FD_3 i.e $\{Pno\} \rightarrow \{Pname, Ploc\}$ violates 2NF conditions as $\{Pname, Ploc\}$ partially dependent on the key.

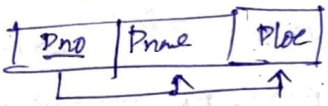
By decomposing we have EMP-PROJ1



EMP-PROJ2



EMP-PROJ3



So EMPPROJ

emp_proj1 emp_proj2 emp_proj3

Q. $R(A, B, C, D, E, F, G, H, I, J)$.

$F = \{AB \rightarrow C, A \rightarrow DE, B \rightarrow F, F \rightarrow GH, D \rightarrow IJ\}$

decompose the above rel^n in 3NF rel^n?

Sol^n:

$AB^+ = ABFDEGHIJC = R$

$A^+ = ADEIJ \neq R$

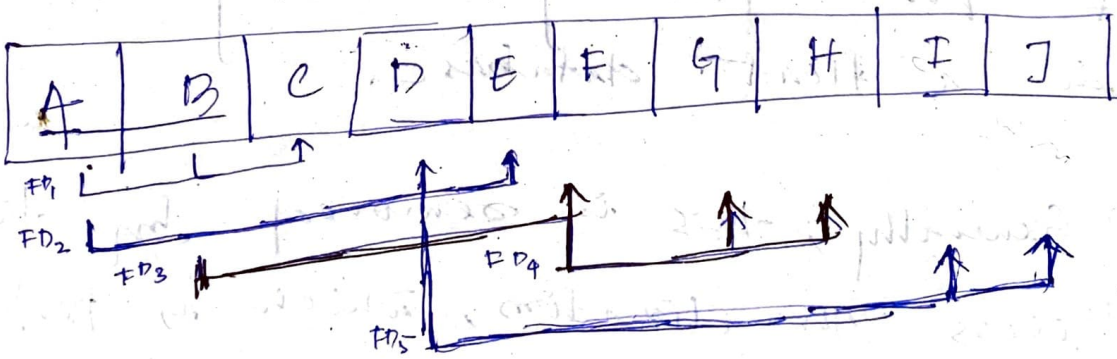
$B^+ = BFGH \neq R$

$F^+ = FGH \neq R$

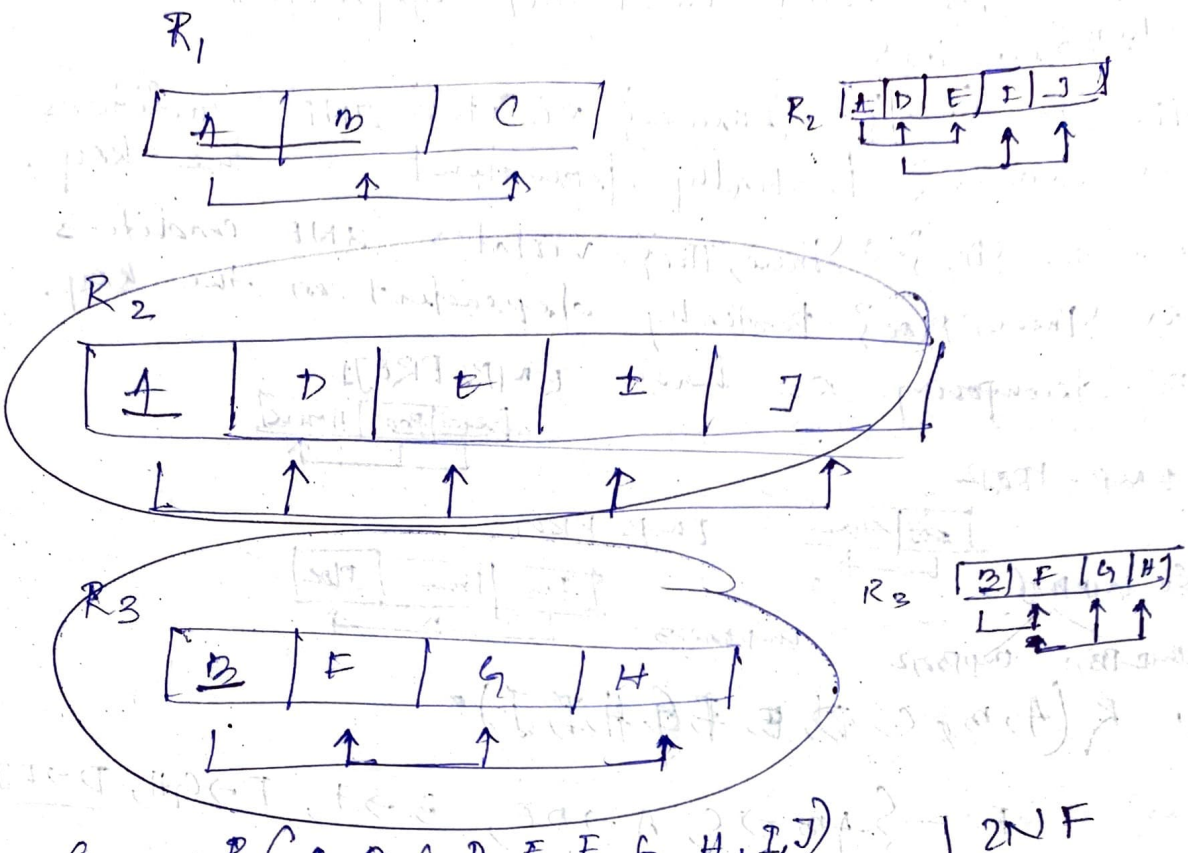
$D^+ = DIJ \neq R$

FD_1, $AB \rightarrow C$ does not violate 3NF conds.

AB^+ is the key.



By decomposing, we have



S_1 $R(A, B, C, D, E, F, G, H, I, J)$ ↓ 2NF

$R_1(A, B, C)$

$R_2(A, D, E, I, J)$

$R_3(B, F, G, H)$

Q. what

Ans: Database anomaly is normally the flaw in databases which occurs because of poor planning & storing everything in a float database.

→ Generally, this is removed by the process normalization, which is performed by splitting / joining of table

There are 3 types of anomalies

(i) Insertion anomaly
(ii) Update anomaly
(iii) Deletion anomaly

(i) Insertion anomaly

→ An insertion anomaly occurs when we are not able to insert certain attribute in the database without

Third Normal Form (3NF)

25 FEB 22

It is based on the concept of transitive dependency.

Def": A relation is in 3NF

~~if~~

i) if it is in 2NF

ii) No non-prime attribute is transitively dependent on the Primary key.

Q. R(A, B, C, D, E, F, I, J)

& F = { AB → C, A → DE, B → F, F → GH, D → IJ }

decompose R into 3NF.

Ans: Key = {A, B}

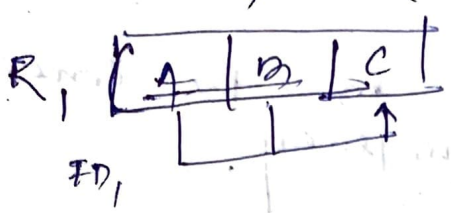
decomposing R into 2NF

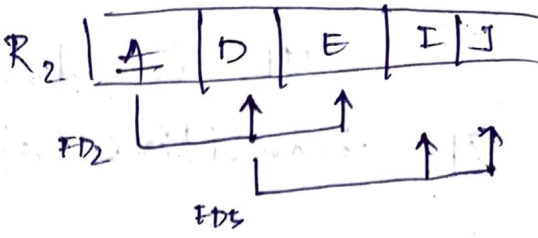
FD_1 AB → C does not violate 2NF Condition.

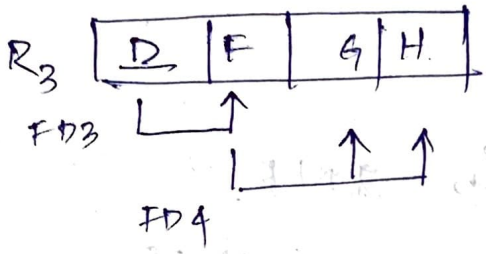
FD_2 A → DE ~~does not~~ violate 2NF condition.

FD_3 B → F violate 2NF Condition

Decomposing R into 2NF we have



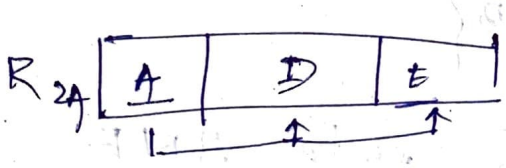


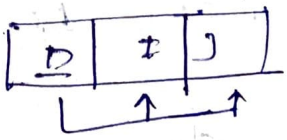


R_1 $AB \rightarrow C$ so $R_1(A, B, C)$ satisfies 3NF.

$R_2(A, B, E, I, J)$ does not satisfy 3NF as transitive dependency because of $A \rightarrow DE$, $D \rightarrow IJ$

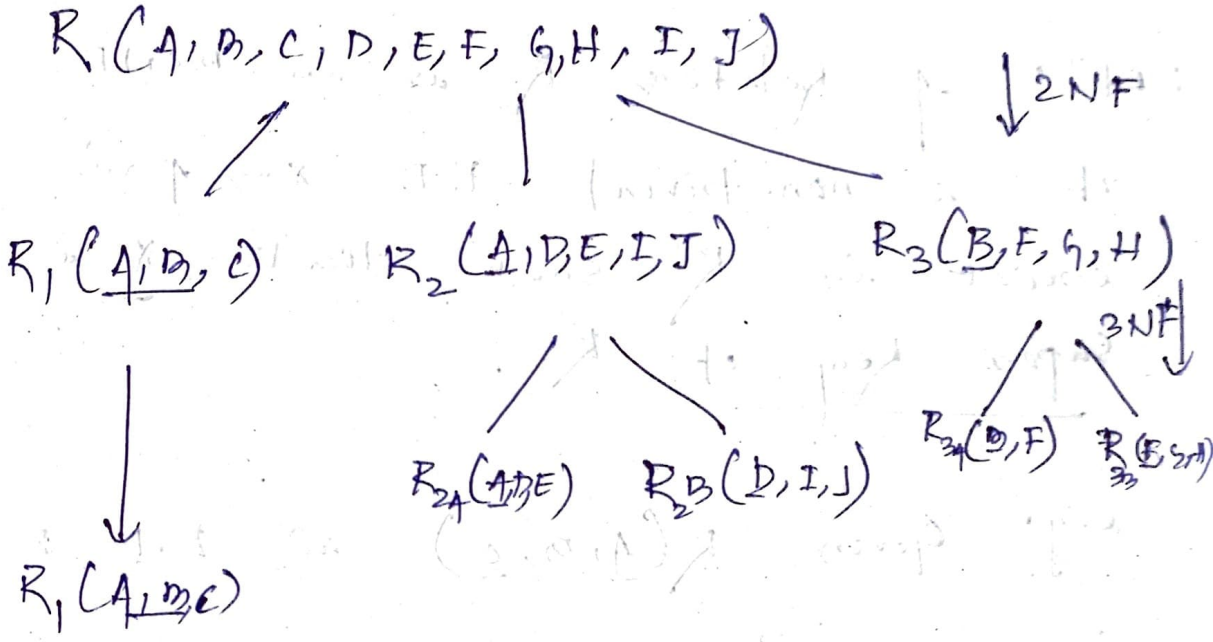
decomposing R_2 into 3NF we have





$R_3(D, F, G, H)$ does not satisfy 3NF as transitive dependency because of $D \rightarrow F$, $F \rightarrow GH$

Decomposing R_3 into 3NF we have



General Def' of 3NF

8 MAR 22

A relation R is in 3NF if a non trivial F.D $X \to A$ exists in R such that-

either (i) X is a Super key
or (ii) A is a prime attribute

BCNF (Boyce Codd Normal Form)

It is most strictest form than 3NF and it is simpler than 3NF also.

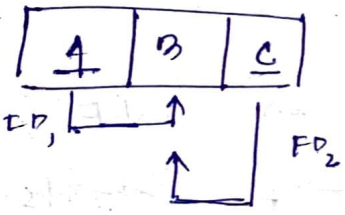
Every BCNF is 3NF but every 3NF is not in BCNF

Assignment 2

Discuss 3NF and BCNF and show that BCNF is stronger than 3NF.

Def: A Relation R is in BCNF if a non-trivial F.D $X \to A$ is exist in R such that X is a Super Key of R.

e.g: Given $R(\underline{A}, B, \underline{C})$ and F.D F is given by $F = \{A \to B, C \to B\}$



Here FD_1 i.e $A \to B$ does not violate 3NF Condition as A is a key.

FD_2 $C \to B$ does not violate the 3NF Condition because C is a key.

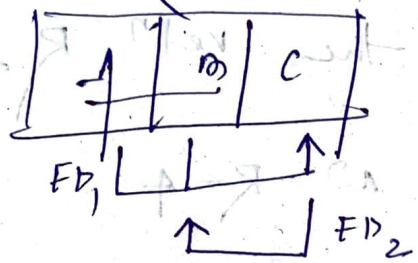
So the rel? is in 3NF.

FD_1 and FD_2 does not violate BCNF Cond? because A is a key and C is a key so that the rel?

is in BCNF.

eg. Given $R = (A, B, C)$ & F.D is given by $F = \{AB \to C, C \to B\}$

check it is in BCNF or not.



→ The rel^n is not in BCNF because FD_2 violates BCNF condition as C is not a key.

→ But the rel^n is in 3NF because here FD_1 satisfy the 3NF cond^n as AB is the key & FD_2 satisfies 3NF cond^n as B is a prime attribute.

So here this rel^n is in 3NF but not in BCNF..

BCNF DECOMPOSION PROCESS :-

Given a Relation R & a set of F.D. We have to decompose R in BCNF

- Step-1 : Identify the dependencies which violates the BCNF conditions in R and consider that as $X \to A$.

Step-2 : Decompose the relⁿ R into XA (i.e $\{X\} \cup \{A\}$) and $R - A$.

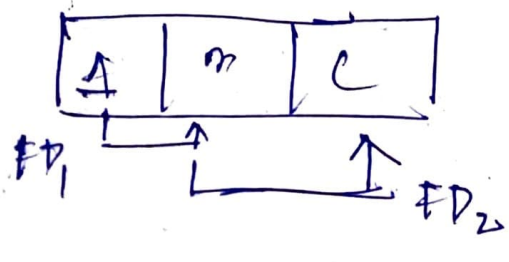
Step 3 : Validate if both the decomposition are in BCNF or not. If not reapply the algorithm ~~i.e step-1~~ on the decomposition that is not in BCNF.

Q) Given R(A, B, C) and F is $R = \{A \to B, B \to C\}$ is R in BCNF if not decompose it into BCNF.

Soln :

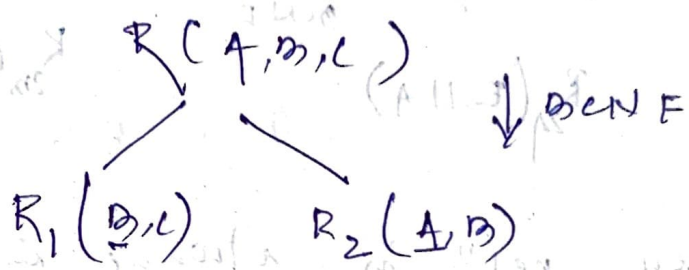
$A^+ = \{A, B, C\} = R$

Key = {A}



FD_1 does not violate BCNF as A is the key.

FD_2 violates BCNF cond^n as B is not a key.



④ Rel^n having two attribute is in BCNF.

Set of F.D.

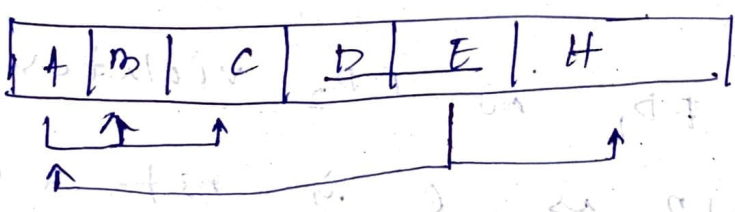
Q// Given R(A, B, C, D, E, H) rel^n, F on R is given by $F = \{A \to BC, E \to HA\}$

So whether the rel^n is in BCNF if not decompose it into BCNF.

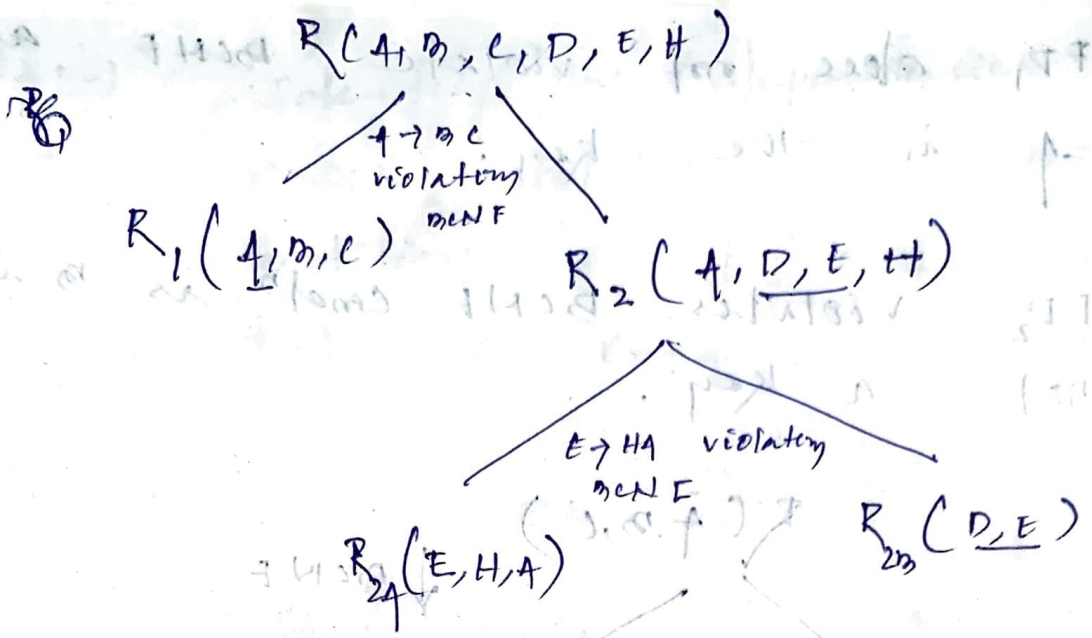
$A^+ = \{A, B, C\}$

$E^+ = \{A\ E\ H\ BC\}$

So the key is DE.



FD_1 i.e $A \to BC$ violates BCNF Condition as A is not a key.

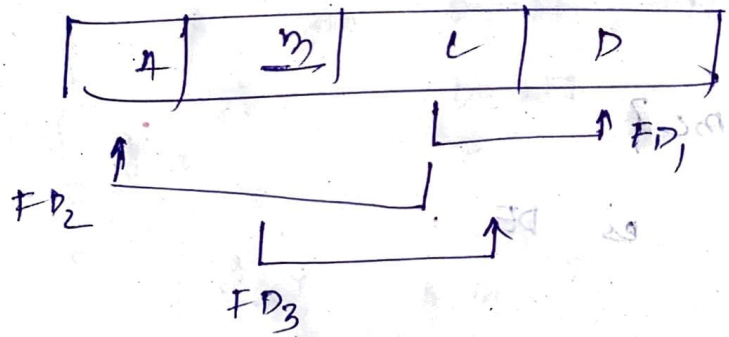


*) All key rel^n is always in BCNF (means all the attributes are key)

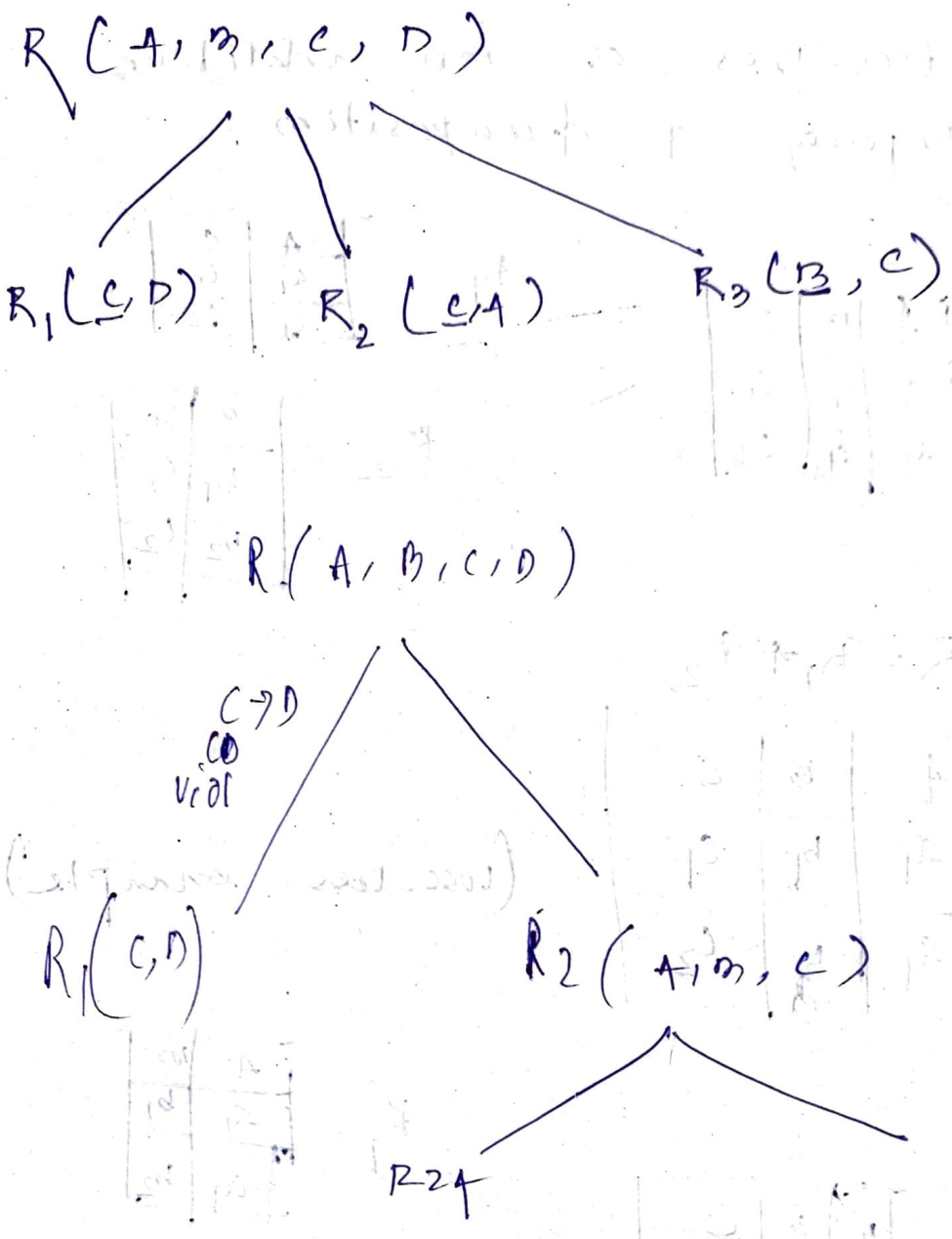
Q) R(A, B, C, D) and $F = \{C \to D, C \to A, B \to C\}$ is R is in BCNF.

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

Key is $\{B\}$



So here FD_1 and FD_2 violates BCNF concept as C is not a key.



11 MAR 22

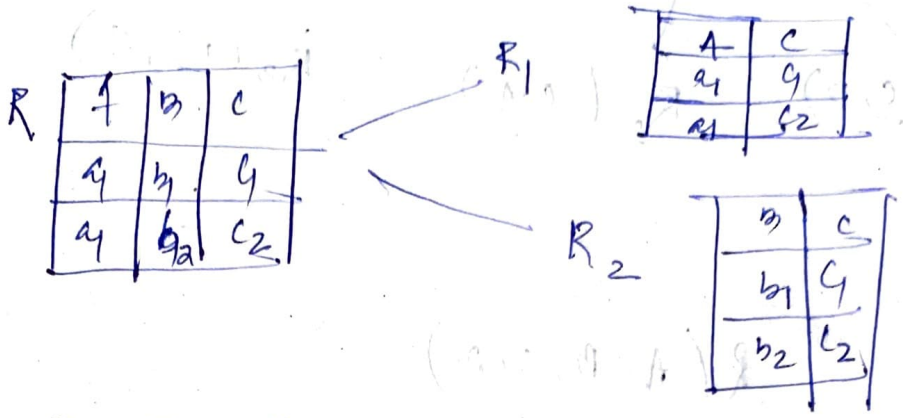
Properties of Decomposition

Let R be a relation and R can be decomposed into a set of relations $D = \{R_1, R_2, \ldots R_n\}$ w.r.t a set of f.d and some decomposition algorithm.

The decomposition D should satisfy the following properties

(1) Dependency preserving scheme or decomposition.

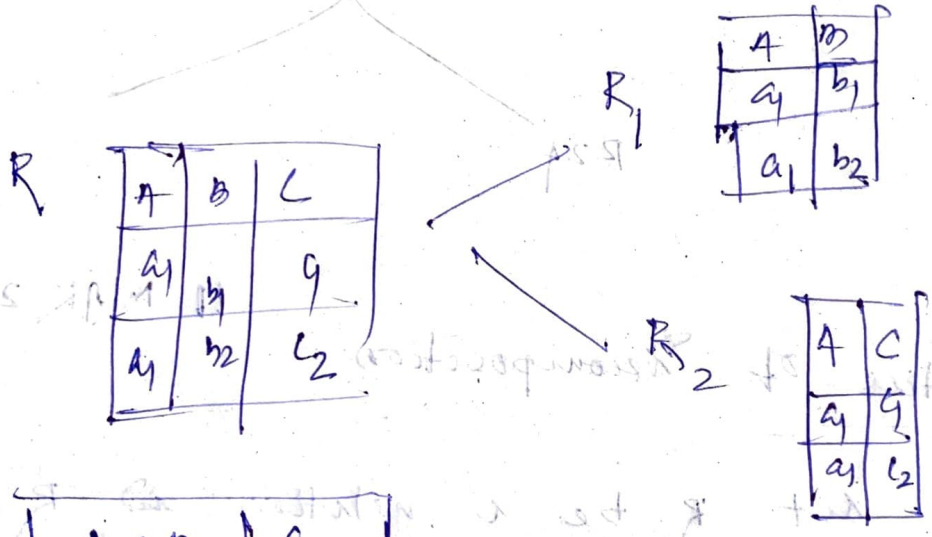
(11) loss-less or non-additive property of decomposition:



$R = R_1 * R_2$

A	B	C
a_1	b_1	c_1
a_1	b_2	c_2

(loss-less example)



A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_1	b_1	c_2
a_1	b_2	c_2

@V.V. gmp

① Whether a decomposition is preserving or not.

② Whether a decomposition is lossless.

Lossless join property or non-additive join property.

① Every 3NF is dependency preserving but may not satisfy loss-less.

② Every BCNF is loss-less but may not satisfy dependency

Generally a decomposition D should causes a non-additive or loss less join property if no spurious tuple (i.e extra tuple that is not in the original rel^n) are generated when a natural join operation is applied to the decomposed relations.

Formally, a decomposition D which is $D = \{R_1, R_2, \ldots R_n\}$ of R has the loss less join or non-additive join property w.r.t a set of F.

if for every rel^n state r(R) that satisfy the following where * is the natural join operation

$$\Pi_{R_1}(r) * \Pi_{R_2}(r) * \cdots * \Pi_{R_n}(r) = r(R)$$

VVV group

Ex: Given $R(eid, ename, Pno, Ploc, hours)$

$F = \{ eid \rightarrow ename, Pno$

$\{Pno\} \rightarrow \{Pname, Ploc\}$

$\{eid, Pno\} \rightarrow \{Hours\} \}$

and the decomposition $D = \{R_1, R_2, R_3\}$ is given by

$R_1(eid, ename)$

$R_2(Pno, Pname, Ploc)$

and $R_3(eid, Pno, hours)$.

Check whether the decomposition D is loss less.

Sol: Initial matrix

	eid	ename	Pno	Pname	Ploc	hours
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	b_{32}	a_3	b_{34}	b_{35}	a_6

FINAL MATRIX (by considering F.DF)

	eid	ename	Dno	Dname	Dloc	hours
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	~~b_{32}~~ a_2	a_3	~~b_{34}~~ a_4	~~b_{35}~~ a_5	a_6

N.B ⟹ If any one row contains all the a values, the decomposition is loss less otherwise, the decomposition is lossy.

Here the last row contains all the a values ∴ this implies the decomposition is loss less.

Q) R(ABCDEG)

$F = \{AB \to C,\ AE \to B,\ AD \to E,\ B \to D,\ BC \to A,\ E \to G\}$

$D = \{AB,\ BC,\ ABDE,\ EG\}$

Initial matrix

	A	B	C	D	E	G
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{22}	a_2	a_3	b_{24}	b_{25}	b_{26}
R_3	a_1	a_2	b_{33}	a_4	a_5	b_{36}
R_4	b_{41}	b_{42}	b_{43}	b_{44}	a_5	a_6

Lossless decomposition

$B \to D$
$AB \to E$
$E \to G$

	A	B	C	D	E	G
R_1	a_1	a_2	b_{13}	b_{14} a_4	b_{15} a_5	b_{16} a_6
R_2	b_{22}	a_2	a_3	b_{24} a_4	b_{25}	b_{26}
R_3	a_1	a_2	b_{33}	a_4	a_5	b_{36} a_6
R_4	b_{41}	b_{42}	b_{43}	b_{44}	a_5	a_6

It is lossy.

If R(A B C D E G)

$\{AB \to C,\ AC \to B,\ AD \to E,\ B \to D,$
$BC \to A,\ E \to G\}$

$D = \{ABC,\ ACDE,\ ADG\}$

	A	B	C	D	E	G
R_1	a_1	a_2	a_3	b_{14}	b_{15}	b_{16}
R_2	a_1	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	b_{32}	b_{33}	a_4	b_{35}	a_6

A	B	C	D	E	G
a_1	a_2	a_3	a_4	a_5	a_6
a_1	b_{22}	a_3	a_4	a_5	a_6
a_1	b_{32}	b_{33}	a_4	a_5	a_6

~~It is loss-less.~~

A	B	C	D	E	G
a_1	a_2	a_3	a_4	a_5	b_{16}
a_1	b_{22}	a_3	a_4	a_5	a_6
a_1	b_{32}	b_{33}	a_4	a_5	a_6

It is loss less.

Sheet. 9

Multivalued Dependency

22 MAR 22

TRANSACTION PROCESSING

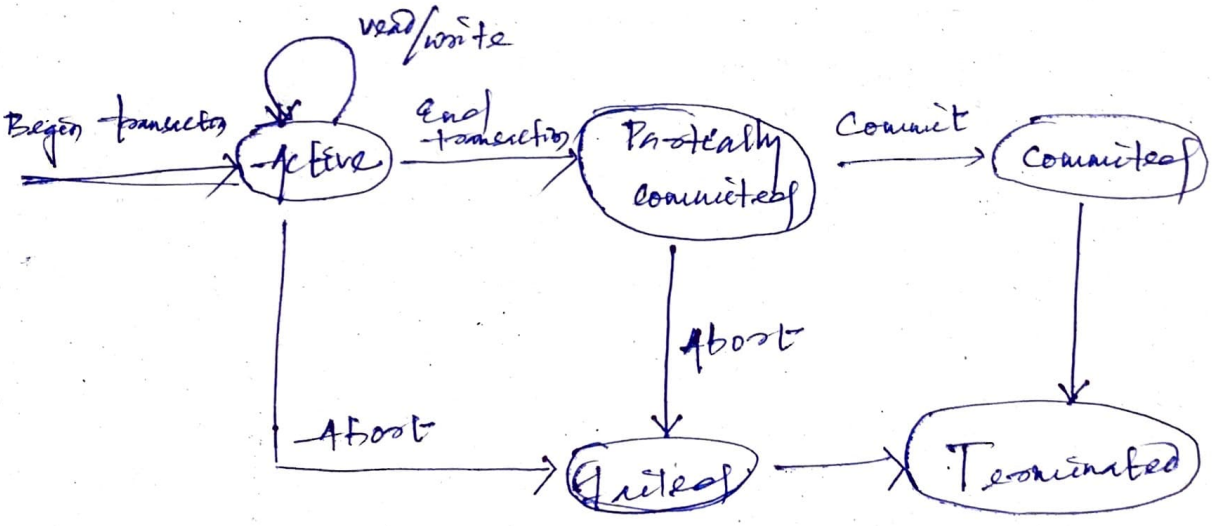
22 MAR 22

*) Transaction is a logical unit of database processing.

The operations that characterises the transaction are

i) Read ()
ii) Write ()
iii) Commit
iv) Abort

Transaction states :



Active state :-

When the instructions of the transaction are running then the transaction is in active ~~stage~~ state. If all the 'read and write' operations are performed without any error then

it goes to the "partially committed state, if any instruction fails it goes to the failed state.

Partially committed state :-

After completion of all the read and write operation the changes are made in main memory. If the changes are made permanent on the data base the state will change to "committed state" and in case of failure it will go to the "failed state".

Failed state :-

When any instruction of the transaction fails, it goes to the "failed state" or if failure occurs in making a permanent change of data on database.

Committed state :-

It is the state when the changes are made permanent on the database and the transaction is complete and therefore terminated in the "terminated state".

Terminated state :-

If there is not any roll back or the transaction comes from the "committed state", then the system is consistent and ready for new transaction and the old transaction is terminated.

Properties of transaction :

A - Atomicity

C - Consistancy Preserving

I - Isolation

D - Durability

Atomicity :-

A transaction is an atomic unit of processing it should either be performed in its entirety or not performed at all.

Consistancy Preserving :-

A transaction should be consistancy preserving meaning that it is completly executed from begning to end and it should take the database one consistance state to another.

Isolation :-

A transaction should appear as though it is been executed in isolation from other transactions even those many transactions are executed concurrently that means execution of a transaction

should not be interfered ~~with~~ by any other transactions executed concurrently.

Durability: The changes applied to the database by a committed transaction must persist in the database: these changes must not be lost because of any failure.

25 MAR 22

Schedule/History of Transaction:-

In which order operations of the transactions are executed.

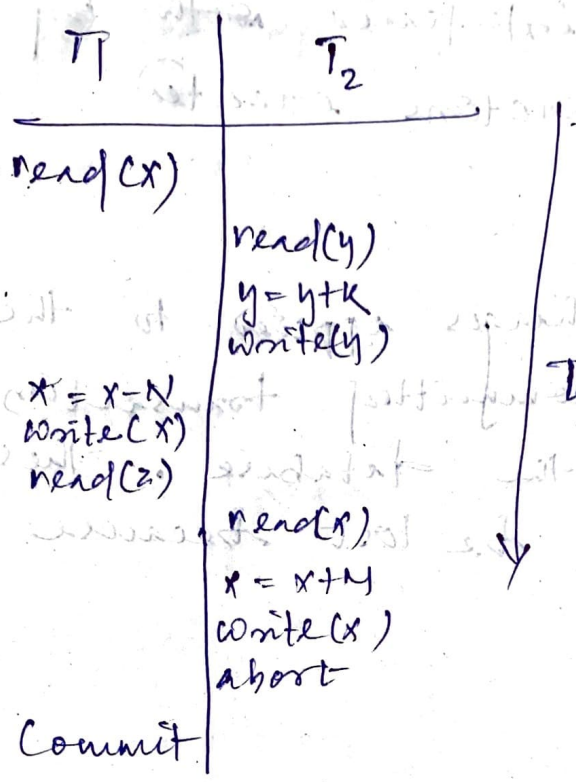
T_1

```
read(x)
x = x - N
write(x)
read(y)
y = y + N
read(z)
write(y)
Commit
```

Time ↓

Symbols for operations:
- read() → r
- write() → w
- Commit → c
- abort → a

S: $r_1(x)$; $w_1(x)$; $r_1(y)$; $r_2(z)$; $w_1(y)$; c_1



$S : r_1(x), r_2(y) ; w_2(y) ; w_1(x) ; r_1(z) ; r_2(x) ; w_2(x) ; a_1 ; a_2$

CONCORENT TRANSACTION :-

→ When the transactions are executed symoltaniously, it is known as Concorent transactions

→ When concorent transactions are executed in an uncontrolled manner, different problems arises such as

- Lost update problem (WW conflict)
- Dirty read / Temporary update problem
- Incorrect Summary problem

LOST UPDATE PROBLEM :-

T_1	T_2
read(X);	
X = X-N;	
	read(X)
	X = X+M;
	write(X);
write(X);	
read(Y);	

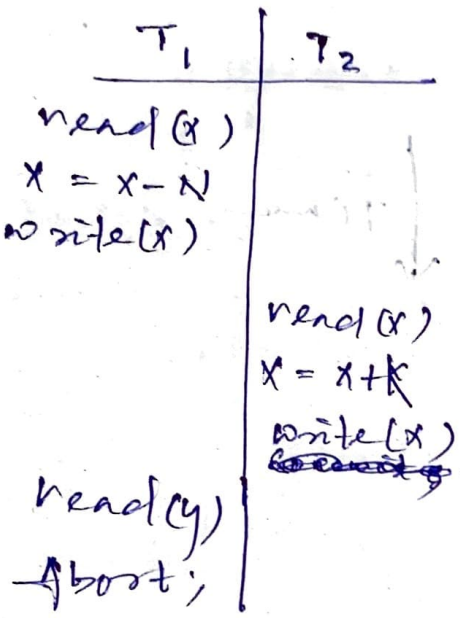
Time ↓

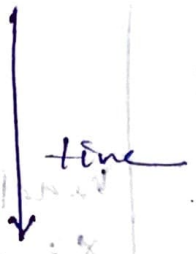
Consider X = 50
N = 10
M = 5

According to transaction the actual answer will be 45 but in this case that is when the transaction is executed in this manner, the output is 55 which is wrong.

In the above case the item X has inconvert value because the updation of T_1 is lost.

Dirty READ / TEMPORARY UPDATE PROBLEM!





~~When one~~ The problem occurs when one transaction updates a database item, then the transaction fails for some reason.

The updated item is accessed by another item before it changed back to its original value.

Instead of getting a correct data a dirty data is updated

Incorrect Summery Problem :- 29.3.22

If one transaction is calculating an aggregate summary function of a number of records while other transactions are updating some of these records, the aggregate function may calculate some values before they are updated and others after they are executed updated.

T_1	T_2
	read(a)
	sum = 0;
	sum = sum + a;
	write(a)
read(x);	⋮
x = x - N;	⋮
write(x)	⋮
	read(x)
	sum = sum + x;
	read(y)
	sum = sum + (y);
read(y)	
y = y - K	
write(y)	

time ↓

Serial Schedule :

A schedule S is serial if for every transaction T participating in the schedule, all the operations of T

are executed ~~consq~~ consecutively ~~consecutively~~ in the schedule.

Otherwise the schedule is non serial.

eg: S: $r_1(x)$; $w_1(x)$; $r_1(y)$; $r_2(x)$; $w_2(x)$

Therefore in a serial schedule only one transaction is active at a time.

The commit or abort of the active transaction initiate the execution of the next transaction.

*) No ~~interleaving~~ interleaving occurences in a serial schedule.

*) A serial schedule is always assume to be correct.

dup e.g: For n-transactions, How many serial schedule can be possible.

Ans: n!

Serializable schedule :-

A schedule S of n transaction is said to be serializable if it is equivalent to some serial schedule of same n transaction.

N.B —

* If a non-serial schedule S is serializable, then it is assumed to be correct.

VV imp

Conflict Equivalent :-

Two schedules are said to be conflict equivalent if the order of any two conflicting operation is same in the schedule.

Conflict operations in a schedule :-

Two operations in a schedule are said to be conflict if the following conditions hold.

(i) They belong to different transaction

(ii) They access the same data item.

(iii) Atleast one of the operation is write operations.

e.g: Given a schedule S

S: $r_3(y)$; $r_3(z)$; $r_1(x)$; $w_1(x)$; $w_3(y)$; $w_3(x)$; $r_2(z)$; $r_1(y)$; $w_1(y)$; $r_2(y)$; $w_2(y)$; $r_2(x)$; $w_2(x)$;

Find the conflict pair?

Ans: $r_3(y)$; $w_2(y)$;

$r_3(y)$; $w_2(y)$;

~~$r_1(z)$;~~ $r_1(x)$; $w_2(x)$;

$w_1(x)$; $r_2(x)$

$w_1(x)$; $w_2(x)$;

$w_3(y)$; $r_1(y)$;

$w_3(y)$; $w_1(y)$;

$w_2(y)$; $r_2(y)$;

$w_3(y)$; $w_2(y)$;

$w_3(z)$; $r_2(z)$;

~~$r_2(z)$~~;

$r_1(y)$; $w_2(y)$;

$w_1(y)$; $r_2(y)$;

$w_1(y)$; $w_2(y)$;

30 MAR 22

Serializability :-

The concept of serializability is used to identify which non-serial schedule are correct.

Serializability Test :-

① Conflict Serializability test :-

This algorithm takes only the read and write operations in a schedule to construct a precedence graph (serialization graph) which is a directed graph.

Step - 1 :

For each transaction T_i participating in the schedule S, create a node, labelling T_i in the precedence graph.

Step - 2 :

Choose the conflict pairs in S.

For each conflict pairs i.e.

$r_i(x)$; $w_j(x)$

$w_i(x)$; $r_j(x)$

$w_i(x)$; $w_j(x)$

Create an edge $t_i \rightarrow t_j$ in the precedence graph.

Step - 3 :

The Schedule S is serializable if the precedence graph has no cycle.

N.B :

*) If a Schedule S is serializable the corresponding equivalence serial Schedule can be obtained by topological ordering of nodes in the graph.

Consider the following schedules

1) S_1 : $r_3(y)$; $r_2(x)$; $r_1(x)$; $w_1(x)$; $w_3(y)$; $w_3(z)$; $r_2(z)$; $r_1(y)$; $w_1(y)$; $r_2(y)$; $w_2(y)$; $r_2(x)$; $w_2(x)$;

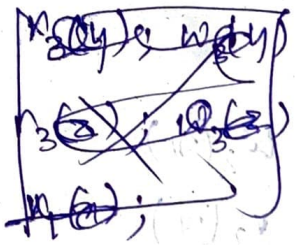
2) S_2 : $r_2(x)$; $r_2(y)$; $w_2(y)$; $r_3(y)$; $r_3(z)$; $r_1(x)$; $w_1(x)$; $w_3(y)$; $w_3(z)$; $r_2(x)$; $r_1(y)$; $w_1(y)$; $w_2(x)$

check whether the given Schedule is Serializable.

If Serializable, find the corresponding Serial schedules.

Ans 1. ~~Checking~~ :

The conflicts pairs in S_1 are



$r_3(y)$; $w_1(y)$ $w_2(y)$; $w_2(y)$

~~$r_3(z)$~~ $r_3(y)$; $w_2(y)$ w_3~~(x)~~; $r_2(z)$

$r_1(x)$; $w_2(x)$ $r_1(y)$; $w_2(y)$

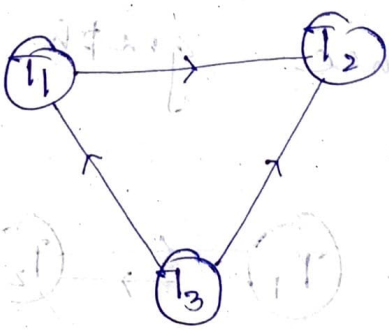
$w_1(x)$; $r_2(x)$ $w_1(y)$; $r_2(y)$

$w_1(x)$; $w_2(x)$ $w_1(y)$; $w_2(y)$

$w_3(y)$; $r_1(y)$

$w_3(y)$; $w_2(y)$

The Precedence graph is as follows



The Precedence graph doesn't contain any cycle. ~~graph~~

So the Schedule S_1 is Serializable.

Corresponding serial schedule by topological ordering is

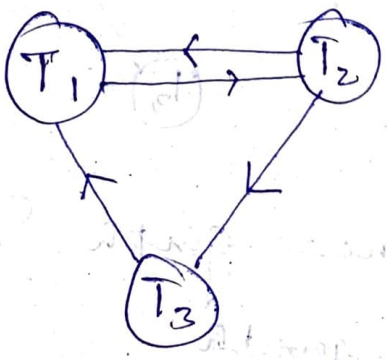
$$T_3 - T_1 - T_2$$

Ans 7.

The conflict pairs in S_2 are

$r_2(z)$; $w_3(z)$	$r_1(x)$; $w_2(x)$
$r_2(y)$; $w_3(y)$	$w_1(x)$; $r_2(x)$
$r_2(y)$; $w_1(y)$	$w_1(x)$; $w_2(x)$
$w_2(y)$; $r_3(y)$	$w_3(y)$; $r_1(y)$
$w_2(y)$; $w_3(y)$	$w_3(y)$; $w_1(y)$
$w_2(y)$; $r_1(y)$	~~$w_3(z)$; $r_1(z)$~~
$w_2(y)$; $w_1(y)$	
$r_3(y)$; $w_1(y)$	
~~$r_3(z)$; $r_1(z)$~~	

The precedence graph is as follows



If the graph contains a cycle, the schedule is not Serializable.

5 APR 22

LOCKING TECHNIRQUE IN CONCURRENCY CONTROL

LOCK: Lock is a variable associated with a data item that describes status of the data item with respect to possible operations.

Shared lock / Exclusive lock :

↳ Read lock / Write lock

In this scheme there are 3 operations

(i) Read_lock(a) / S_lock(a)

(ii) Write_lock(a) / X_lock(a)

(iii) Unlock(a)

Shared lock / Read_lock / S_lock :

→ A transaction must issue a S_lock(a) operation before any read(a) operations is performed in T.

→ Multiple transaction can hold a S-lock on a database item simultaneously.

Write-lock / Exclusive-lock / X-lock :

→ A transaction T must issue the operation write-lock(x) / X-lock(x) before any write(x) operation performed in T.

→ Only one transaction can hold write-lock(x) / X-lock(x) on a data item on any given time.

→ No other transactions can hold any lock (not even read lock) of a transaction has a write-lock / X-lock.

N.B —

A transaction that holds a write-lock on a database item can read and write the data item where as in case of S-lock operation the transaction can only read the data.

Convert transaction in lock based protocol

T_1

```
read(x)
x = x - N
write(x)
read(y)
y = y + N
write(y)
```

T_1

```
S_lock(x)
read(x)
x = x - N
X_lock(x)
write(x)
S_lock(y)
read(y)
y = y + N
X_lock(y)
write(y)
Unlock(x)
Unlock(y)
```

T_1 (another way)

```
S_lock(x)
read(x)
x = x - N
Unlock(x)
X_lock(x)
write(x)
Unlock(x)
S_lock(y)
read(y)
y = y + N
Unlock(y)
X_lock(y)
write(y)
Unlock(y)
```

2 Phase Locking Protocol:

2PL (basic 2PL)

Each transaction must obtain a shared lock on an object before reading and an exclusive lock on object before writing.

A transaction can release its lock once it has performed its desired operation (read, write). A transaction can't request additional locks once it releases any lock.

If a transaction holds exclusive lock on an object no other transaction can get a lock (S or X) on that object.

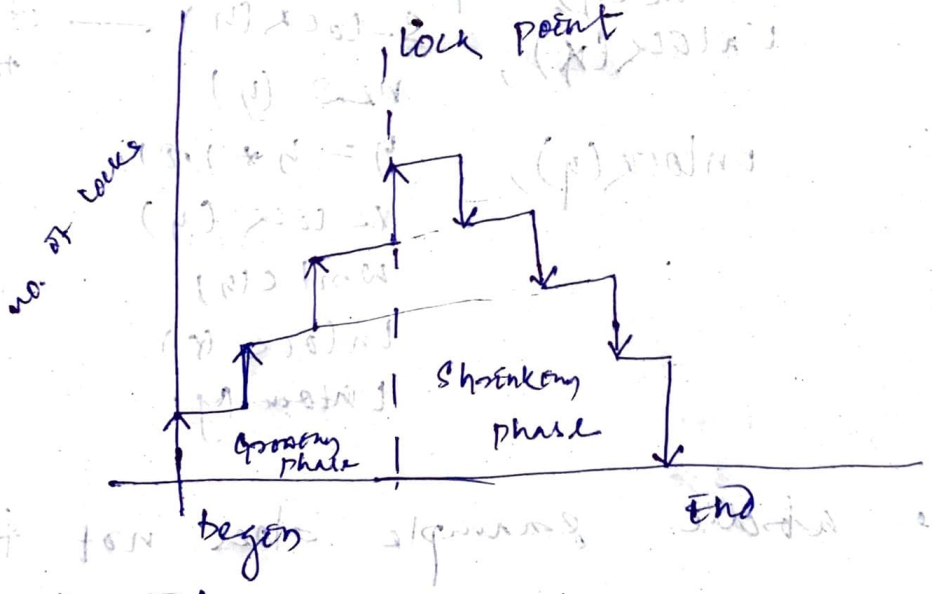
transaction
2PL is devided into 2 phases

- Growing Phase:

New locks can be obtained but none of the locks can be released.

- Shrinking Phase:

Existing locks can be released but no new locks can be obtained.



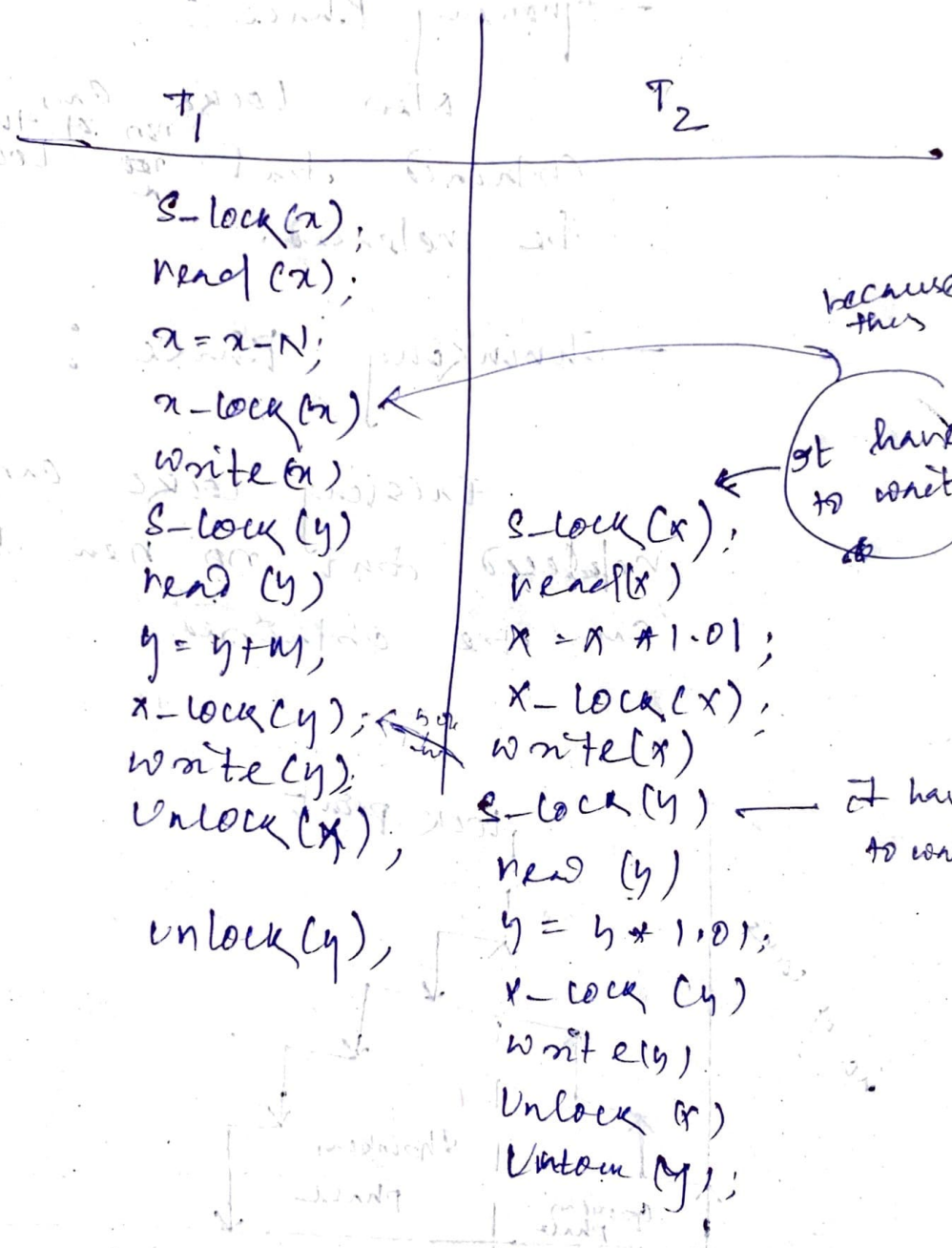
LOCK POINT:-

The point where the transaction obtains its last lock. It is the end of growing phase.

Two phase locking protocol ensure serializability (Conflict serializability)

Transactions can be serialize in the order of their lock points.

e.g.



The above example does not follow two phase locking protocol.

In two phase locking protocol the transaction T_1 & T_2 can be specified as follows

T_1	T_2
s_lock (x);	
read (x)	
x = x - N;	
x_lock (x)	
write (x)	
s_lock (y)	
read (y)	
y = y + M	
← x_lock (y)	
write (y)	
Unlock (x)	s_lock (x)
	read (x)
	x = x * 1.01;
	x_lock (x)
	write (x);
Unlock (y);	s_lock (y)
	read (y)
	y = y * 1.01;
	x_lock (y) ← Lock point for T_2
	write (y)
	Unlock (x);
	Unlock (y);

Lock point for T_1

wamp

Q/ Prove that 2PL is serializable.
or 2PL ensures serializability.
(Proof by method of Contradiction)

Proof: Statment:
If S is a schedule containing the interleved operations from a number of transactions $T_1, T_2, \ldots, T_n$ and all the transactions are using 2PL Protocol schedule S is serializable.

Assume that the schedule S is not serializable.

⇒ The Precedence graph for S will have a cycle made up of a subset of $\{T_1, T_2, T_3, \ldots, T_k\}$

Assume that the cycle consist of $T_a \to T_b \to T_c \to \ldots \to T_x \to T_a$

i.e a lock operation in T_b is followed by an unlock operation of T_a, a lock operation by T_c is followed by

an unlock operation in T_b and so on.

Finally a lock operation by T_a is followed by an unlock operation in T_a.

However, this is a contradiction of the statement that T_a is using 2PL Protocol, once T_a is unlocked it can't again acces any lock.

Thus, the assumption that there is a cycle in the precedence graph is incorrect and S is serializable.

Hence 2PL ensures Serializability.