Siddhant Patil

DBMS

25.07.2022
Monday

understand the living needs of the society and create a novel / updated product.

Idea / Imagination

USP (Unique selling Population)

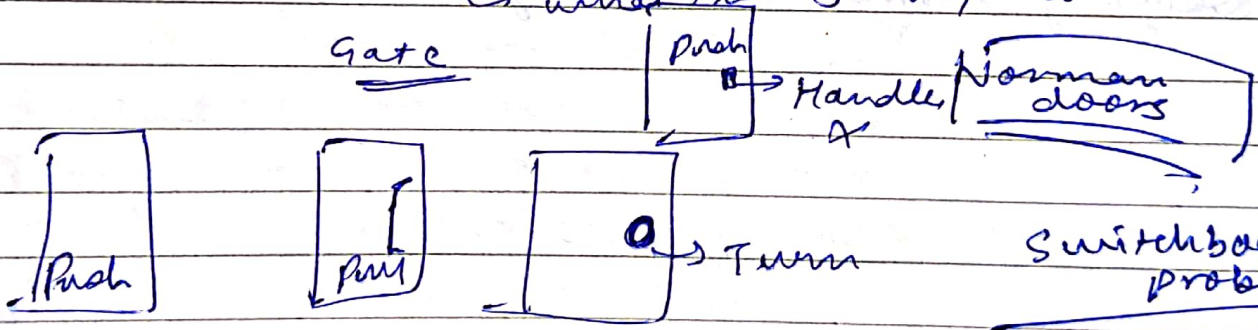
Creation → Designing is essential

Unit-1 Describe the design process
Constitute a design team.
Plan the design process.

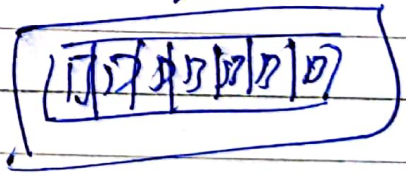
Psychologist
↓
Thinking of people.

Unit-2 Design Thinking
↳ what exactly requires

Gate



Switchboard problem



Dif. approaches
Specific problem statement

Apple, Pepsico, P&G, Nike, Shimano, Honda

Unit-3 Culture of design & innovation in org.

new sol^{n} to cater society.

26.07.2022
Tuesday

DBMS

– Data Base Management System

↳ Big Data.

Unstructured Data ↓
(i) FB's comments
(ii) Reviews.

Structure Data ↓

DB + DBMS = Database management Environment

Banking System was the earliest system to computerises
(using File Handling) Processing System)

Q) Student (Sch. No., Name, age, add^{r}, branch, Ph. No, email id).

Q) Text file → insert a new record
delete a existing student record
update existing ——"——
search a scholars No.

2 dept; maintains ex;

#) Problems with such systems :–

① Data Redundancy and inconsistency
↓ repetition of data.
↳ Not updated at all places

② Difficulty in accessing data.
Soln:
Modification of system or writing new code

querying → seeking a particular information

en: Bhopal leaving Student, Hostelers.

③ Difficulty in enforcing integrity constraints

en. Age limit

④ Data Isolation.

merging of 2 diff. data stored by 2 two diff. system.

⑤ Ensuring atomicity of update.

en. Transaction

Account A — Rs 50 → Account B

500 — 400

read(A)
A = A - 50
write(A)

read(B)
B = B + 50
write(B)

Either transaction takes place completely or not at all & not leaving it in bet^n.

only & only (A+B)$_{initial}$ = (A+B)$_{final}$ (= 900 (here))

else { NO }.

SQL → Structured Query Language.

⑥ Handling concurrent transactions.
ex. Two persons doing transactions from same a/c and at same time.

⑦ Security issue.
Enforcement access control of data
ex. Students can view only marks and not update

Example of DBMS. → SQL { MySQL., ; packages. }

~~Database~~

Database Management Systems

DBMS :- Set of programs to allow easy and efficient storage and management (mgmt) of the database.
Ex:- DBase, MySQL, MS-SQL, Oracle, MongoDB, MariaDB, SQLite, FoxPro, Cassandra

Database :- A collection of interrelated data / data pertaining to a particular organisation / enterprise application
Data of University only not of Bank employees.

Typical functionalities of a DBMS :- (SLMS)-(SVM)
① Define the database :- structure, constraints
② Construct or load the database (on secondary storage memory)
③ manipulating the database (inserting, deleting, updating, querying the database)

27.07.2022 Wednesday.

(first 2 to 3 lectures are theory)

④ Sharing the database among multiple users.

Other tasks:-

⑤ security against unauthorised access.

⑥ Presentation and Visualisation of data.

⑦ Maintainance of the database.

Books

DBMS by Korth and Sudarshan (IIT Bombay)] (Textbook)

DBMS by Elmasri Navathe] Reference

DBMS by C.J. Date

DBMS by Raghu Ramakrishna & Gehrke → Normalisation

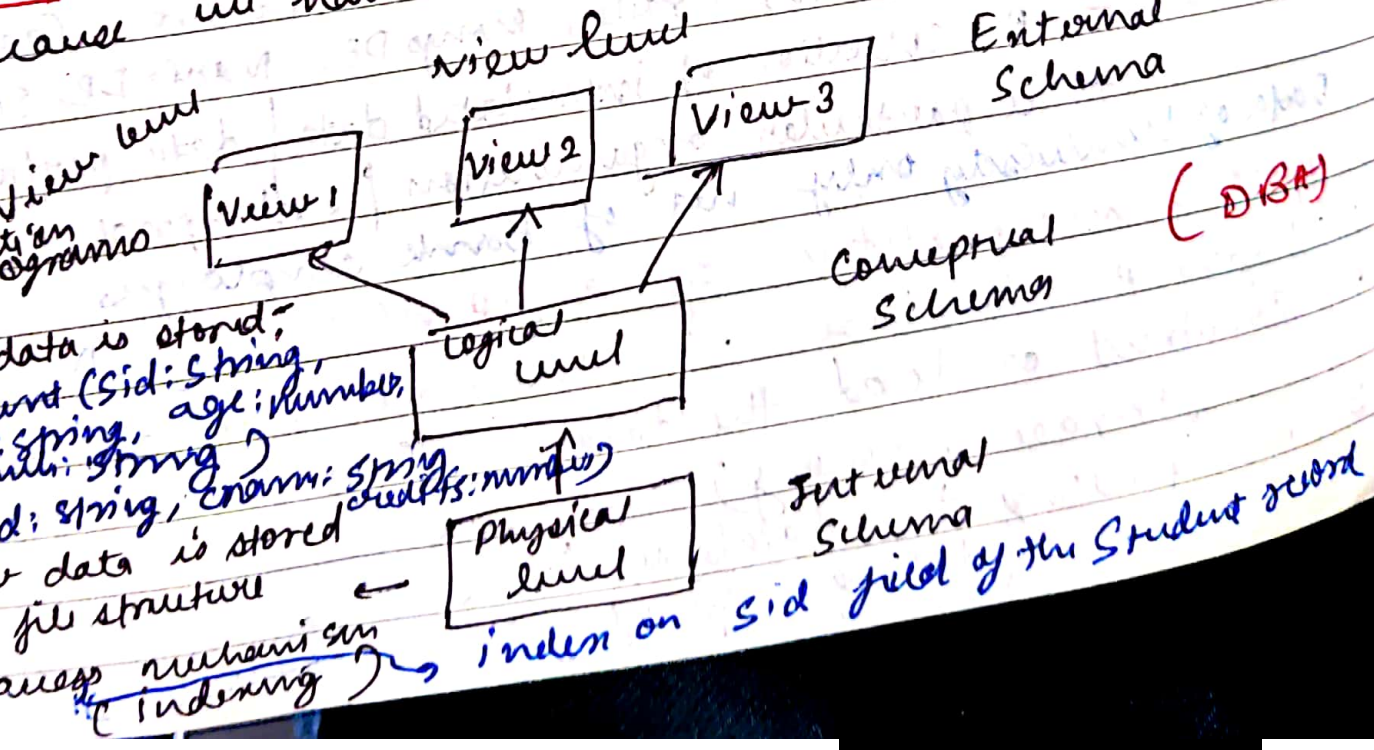
→ Mostly studied in ~~off~~ foreign

oldest

THREE LEVEL ARCHITECTURE

because we have 3 levels of abstraction.

Data Abstraction



Schema → structure
Instance → object

Schema :- describes the structure of the database and constraints (also known as Intension)

Instance :- data that is stored in the database at any particular instance of time. (also known as Extension)

Instance keeps on changing continuously but Schema changes hardly

View : course info(cid : string , enroll ~~num~~ no : number)

~~View~~

View level → multiple ~~level~~ views of the data for multiple users for security reasons.

Analogy

int sch-no. ; // definition → Schema
sch-no = 211112238 ; // Assigning → Instance

Data Abstraction : Hiding the data from users for easy access.

Data Independence / Insulation :
Insulation b/w structure of data and application programs

↳ ① Physical Data Independence :
Changes to internal schema & it does not affect the application program or the logical schema.

↳ ② Logical Data Independence :

Database plan
DBMS → software
Data models → to Relational Data Model
RDBMS → Relational DM DBMS

Changes to the conceptual schema does not affect the app programs.

LDI is more difficult than PDI

DATA MODELS

Data Models → Representation of something

known facts.

↳ Set of tools / concepts for describing the database.

Older Data Models :-

↳ (1) Network Data model

↳ (2) Hierarchical Data model

Relational Data Models :- Tables

(Strong mathematical foundation)

Represents data in the form of relations / tables (rows & columns)

Student

sid	sname	age	addr	emailid	Phno.	Branch
001	xx	2>	-	-	-	-
003	∝ ∝					

key

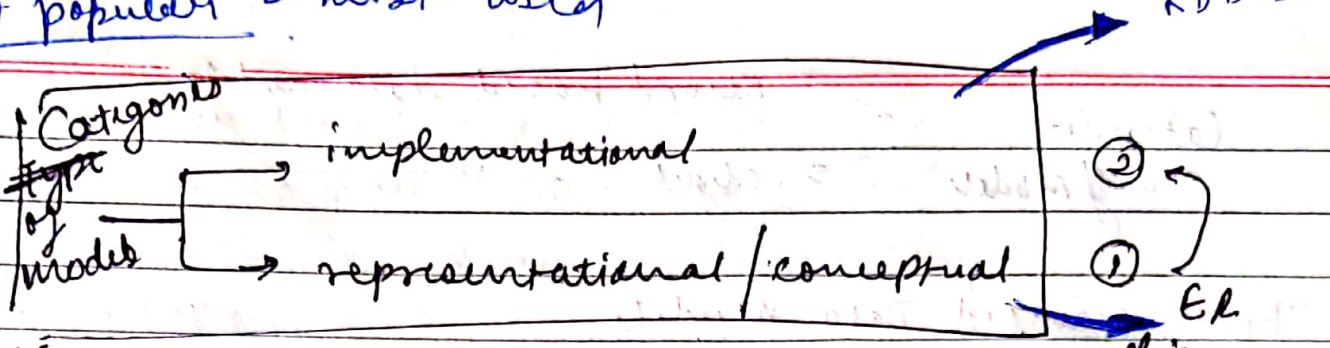
attributes (properties or characteristics of the entity of which data you want to store)

→ Tuple

Columns : attributes
Rows : tuples

key : unique / distinguishing field

Most popular & most used → RDBMS.



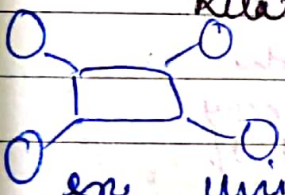
RDBMS. → Oracle, MySQL, DB2, MS-SQL, MS-Access

(used Relational Data Model)

Entity Relationship Model :- (ER diagram)

anything → real world object (Students)
→ abstract things (Courses)

Relationship bet^n entities. is shown by ER Model / ER diagram.

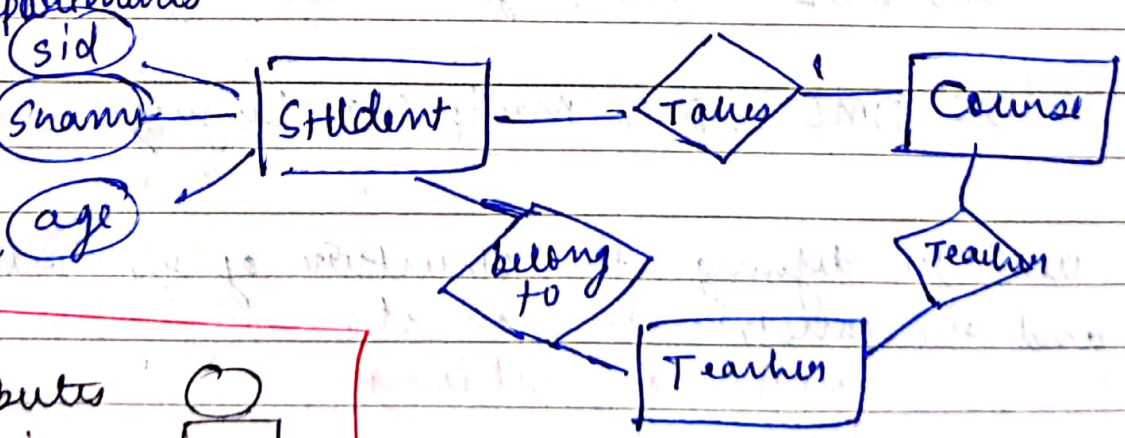


eg University

Student
Course
Teacher
Sections
Departments

Relation

Student takes courses,
Teacher teaches courses.
Student belongs to Departments.



data

attributes
Entities
Relations

Rhombus

Scanned with CamScanner

Objet based Data Model → stored as objet (C++, C#, Java)

Categories of Models → Record based data models
→ objet → ——

Objet-oriented Data Models.

Objet Oriented knowledge.

OODM was not that popular because RDBMS was popular.

Objet-Relation Data Model

Semi-Structured Data Model:

XML → (Semistructured data)

Nowadays it is of importance to store unstructured data

Implistentional ~~typ~~ Model

HTML type.

data items of the same type may have diff set of attributes

JSON, XML (Extensible markup language)

DBMS Languages

1. DDL - Data Definition Language
2. DML - Data Manipulation Language

(1)

DDL: used for defining the structure of the database and the constraints on it.

Works on Schema.

Data Dictionary | Catalog :-

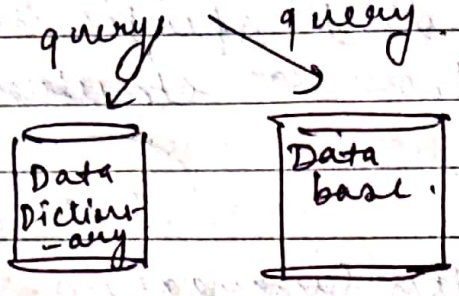
due to this are called DBMS as "Self Describing Nature".

Viva & Interview Questions

RDBMS is manipulate DDL & DML's commands

Verifies information

Any structure declared by us is stored in Data Catalog.



query 1st goes to Data Dictionary then goes to Data Base + many steps in betn..

SQL has both DDL & DML

SQL : Structured Query Language.

ex. of SQL DDL commands :- (i) Create Table
(ii) Drop Table
(iii) Alter Table

tables wise

(2) DML : manipulating the database

Quiz

- (i) Insert
- (ii) delete
- (iii) update
- (iv) query

Instance

ex. of SQL DML commands :- (i) Insert (ii) Delete
(iii) Update
(iv) Query

```
Select
From
where
```

28.07.2022
Thursday

Views are virtual tables.

↓

when you need specific tables from a structure of DBMS

Real table

③ SDL - (Storage Definition Language)
internal schema

④ CDL - (Control Definition Language)
Authorization of data

DML
(Data manipulation language)

Procedural

how to get the data & what data are needed

eg. Relational Algebra (RA)

SQL

Non Procedural.

SQL / Declarative

what data to retrieve without specifying how to get data

eg. Relational Calculus (RC)

Tuple (in proper order)

Domain (we just write) → no order mentioned

List of students who study in CSE branch

Student (sid, sname, ... , branch)

RA: $\pi_{sname}(\sigma_{branch="CSE"}(Student))$

RC: (Tuple) $\{t.sname \mid t \in student \wedge t.branch = "CSE"\}$

ex. SQL

Order. { ③ select sname
① from student
② where branch = "CSE" }

internal Order not clear

USERS / People Associated with DBMS / Database System

① End users

② Database Application Programmers :-
linking frontend with DBMS (backend work)

③ Database System Administration (DBA) (imp for gate)

↳ (i) designing of physical / logical schema
(Also can be done by DB designers)

↳ (ii) Authorization and Security protection from tampering of data
(who is going to view / edit which part)

↳ (iii) Monitoring & Controlling the efficiency of operation
↖ Turnaround time of query

↳ (iv) Database tuning & maintainence.
Hardware & Software requirements

↳ (v) Data Availability / Crash Recovery.
In case hardware / software failure ensure backup of data in regular intervals to prevent lost of data.

Types of Databases.

1. Textual / Numerical Database → only text & numbers (Student)
2. Multimedia DB → audio + video + image + text
   ex. Airtel Wynk, Gaana
3. Geographical Information Systems DB → spatial DB
   (relate to space - latitudes and longitudes)
   ex. Google Map.
4. Temporal DB → related to Time
   ex. Stock Market use this datatype
   Forecasting weather. ——→
5. Genomic DB → stores genes sequence
   ex. Bio informatics sequence of characters

RELATIONAL MODEL

TF Cood. (IBM)

Relations : Mapping the set of values from set-1 to set-2

informally known as **tables**.

collection of rows & columns.

Student (entity) → attributes → properties / characteristics of entity.

	sid	sname	age	branch
t1	001	xx	21	CSE
t2				

tuple

Attributes - Domain :- set of all possible values from which the attribute can draw a value.

sid = no. of a digits (0 - 9)

sname = string (25)

branch = {"CSE", "IT", "Null", "ECE", "Civil", "EC"}

↳ enumerated sets (known about the branches)

Tuple: Ordered set of related values.

t1 : {"001", 'xx', 21, 'CSE'}

Relation : set of tuple.

NULL

Special symbol

if we don't know field value

Represent

Relation Schema : $R(A_1, A_2, A_3, \ldots, A_n)$

- R → relation
- $A_1, A_2, A_3, \ldots$ → attribute names
- domain is known by the presence of mind / common sense.

Instance / relation state : Populated table

$r(R)$

if all constraints are fullfilled → relational state

else → relational state

Tuple

$t_1, t_2, \ldots, t_n$

$t.A$, $t_1[Sid] = 001$

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

$dom(A_1)$: is the domain of A_1,

$dom(A_n)$ is the domain of A_n

$r(R) \subseteq dom(A_1) \times dom(A_2) \times \ldots \times dom(A_n)$

- $\subseteq$ is the subset of
- $\times$ ~~cross~~ cartesian product

$S_1 = \{a, b, c\}$ $S_2 = \{1, 2, 3\}$

$S_1 \times S_2 = \{(a,1), (a,2), (a,3), \ldots (c,1), (c,2), (c,3)\}$

CONSTRAINTS

Integrity constraints : Validation pts. for data to be put in DB.

Types of constraints

1. Key Constrains
2. Entity Integrity Constraints
3. Referential Integrity Constraints

KEY

key → Uniqueness

Super key : A set of attributes of a relation R such that for any 2 tuples t_1 & t_2

$t_1[SK] \neq t_2[SK]$ — follows uniqueness property

student (sid, sname, age, branch)

superkey SK = (sid, sname) or (sid, age)
or (sid, sname, age, branch)
or (sid, branch, age) or (sid) either one

Q) Why it is called Superkey?

Sol: Because it is superset of key attributes.

NOTE

Superkey must contains key ~~or~~ values like (sid)
+ Extraneous value

R can also be, one of the superkeys of the relation (relation)

Candidate Key – A minimal superkey is called as a Candidate Key.

- Uniqueness
- Minimality : – No proper subset of a candidate key can be a key by itself.

Student (sid, sname, age, branch)

(i) SK = (sid, sname) ⟶

Proper subset = {sid}, {sname}, {ϕ}

{sid} – key

{ϕ} – Not a proper subset

∴ SK is not a CK

(ii) CK = {sid}.

Q) Student (sid, sname, age, branch, address, Adhar No, Email Id)

★★★ underline to represent PRIMARY KEY

Sol:

Possible Candidate keys – (sid), (Adhar No), (sname, address)

(sid) – Primary key

(Adhar No), (sname, address) – Alternate key

both are unique

★ apply unique constraint in table in front of this

→ uniqueness

→ minimality prop.

2.08.2022
Tuesday.

Domain constraint

Any key is a super key but vice-versa is not always true

Student (Sch.No. , name , age , address , adhar no. , email.id)

Primary Key ↓ — Alternate Key ↓

(2) Entity Integrity constraints :- A PK attribute cannot take NULL value.

↓

The value which we don't know is assigned NULL

(i) Unique
(ii) Minimal
(iii) does not allow NULL values

Create Table Student (
Sch_no char(10) PRIMARY KEY,
name varchar(40) NOT NULL,
age int(2)
address ~~char(12)~~ varchar(100),
adhar-no char(12) UNIQUE, → Uniqueness & Not Null [Alternate Key]
email_id varchar(40) .) ;

Simple PK - single attributes (attribute level constraint)
Composite PK - multiple attributes (table level constraints)

Create table Stud1 (;
// No Primary Key above.
PRIMARY KEY (name, address)) ;

either data redundancy

(3) Foreign Key / Referential Integrity Constraint

Referencing Relation R,
Student (Sch_No, ..., D_No)

Referenced Relation R
Department (D_No, D_name, HOD, location)

spurious information → Information without any meaning

∴ Relation is required → use FK

Student

S.No.		D_No
001		1
002		5
.		6
.		

Department

D.No.	DName	HOD
1		
2		
3		
4		
5		

throw error as no 6 in Dept. table.

NOTE: FK can take NULL value if it is not part of Referencing Relation.

A FK attribute in a tuple t_1 of referencing relation R_1 can either be

→ a PK value of a tuple t_2 in referenced relation R_2
i.e., $t_1[FK] = t_2[PK]$

or

→ it can take NULL value (if FK attribute is not part of the PK)

Student (Sch_no, ..., DNo)

Course (CNo, name, ..., credits)

Enrolment (Sch_No., CNo., grade)
FK1, FK2

Enrolment

Sch_no.	CNo	grade
001	C1	A
001	C2	A+
002	C1	B

(X) When the referential integrity constraint is violated and how to deal with it?

(1) When a tuple in the referred relation is deleted a tuple in the referencing relation may get affected and it may violate the referential integrity constraint.

(i) RESTRICT

(ii) CASCADE - Delete all the tuples in the referencing relation which refer to the deleted tuple in referred relation

(iii) SET NULL - Set the value of FK to NULL

(iv) SET DEFAULT value - Set the value of FK to a default value.

(2) ——— " ——— updated (PK)

——— " ———

———

(i) RESTRICT ——— " ———

(ii) CASCADE - update ——— " ———

(iii) SET NULL - ——— " ———

(iv) SET DEFAULT value ——— " ———

No change is due to deletion from referencing relation.

(ii) due to addⁿ to referred relation.

assertion triggers.

FK

Student (Sch-No, ..., DNO)

Department (D-NO, ...)

Create Table Student (

—
—
—

DNo int(2) FOREIGN KEY REFRENCES DEPARTMENT;
);
// for this Department table should be above

Create Table Department (

DNo. ...
—
—
—)

ALTER TABLE Student
ADD FOREIGN KEY (DNO) REFRENCES DEPARTMENT)

// Better way if there is cross referring

Student (SNo., Coordinator, ..., DNO)

Department (D-NO, ..., Coordinator)

03.08.2022 Wednesday

Table level Constraint

Date of birth ↑ Date of Joining ↑

Employee (ENo, Name, DOB, ... DOJ, ...)

Create Table Employee (...

—

—

attribute level ~~Sta~~ constraint ↑

DOB char(4) CHECK CONSTRAINT (DOB > 1950)

CHECK CONSTRAINT (DOJ > DOB)) ;

↓

table level constraint

RELATIONAL ALGEBRA

closed ↙ i/p and o/p same

Basic set of Operations :-

1. Unary operations (1 operand)
   - (i) SELECT (σ)
   - (ii) PROJECT (π)
   - (iii) Rename (ρ)
2. Set Operations (2 operands, Binary operations)
   - (i) Union ($\cup$)
   - (ii) Intersection ($\cap$)
   - (iii) Difference ($-$)
   - (iv) Cartesian Product ($\times$)

③ Binary Operations :
   (i) Join (⋈)
   (ii) Division (÷)

④ Other Operations :
   (i) Aggregate function
   (ii) Outer function

Either individually or set of operations according to desired o/p.

Company Database

Employee (Fname, Minit, Lname, SSN, Bdate, Address, Sex, Salary, Super_SSN, DNO)
(Super_SSN : FK, DNO : FK)

Department (Dname, Dnumber, Mgr_SSN, Mgr_start_date)

Dept_Locations (Dnumber, Dlocations)

Project (Pname, Pnumber, Plocation, DNum)

Works_on (ESSN, Pno, Hours)

Dependent (ESSN, Dependent_Name, Sex, Bdate, Relationship)

Minit → Middle Initial

Lname → Last Name

SSN → Social security No. (Aadhar No. type) — unique.

Bdate → Birthdate

Super-SSN → Supervisor SSN

Super-SSN referring to SSN in same table

It is called SELF REFRENCING

* 2nd way to add Foreign Key (after creating table)

Mgr_SSN → Manager

Mgr-Startdate → When Mgr started its work.

Dept Location

1	NYK
1	Houston
1	Texas
2	London

Pname : Project Name

Works-on → which employee is working on which project and for how many hrs on project/week.

Employee id.

ESSN	PNo.	Hours
E01	P1	5
E02	P1	6
E01	P3	8

→ Hours on project / week.

Dependent

Father Mother of ~~Not~~ Employee

ESSN	Dependent Name	...	Relationship
E01	Ankit		Father
E01	Asha		
E02	Ankit		Child

① Select (σ)

(i) $\sigma_{\langle conditions \rangle}(\langle R \rangle)$ — two brackets means it will be replaced.

↳ argument relation

(ii) Selects a subset of tuples of the relation which satisfy condition

Eg: Find all employees who work in Dept. No. 4

Sol: $\sigma_{DNO=4}(Employee)$

Employee

SSN	DNO		SSN	DNO
E01	1	X	E03	4
E02	2	X	E04	4
E03	4	✓		
E04	4	✓		
E05	3	✓		

(iii) Resulting Relation has same schema as the org. schema

$$Schema_{before} = Schema_{after}$$

(iv) No. of tuples in resulting relation $\leq$ No. of tuples in the org. relation

$$n(Resulting\ Relation) \leq n(Org.\ Relation)$$

(v) condition General Syntax

↗ $=, <, >, !=, >=, <=$

$\langle R.A \rangle \langle op \rangle \langle R.B \text{ or } value \rangle$

Combining relations using logical operations

→ $\wedge, \vee, !$

(vi) Commutative

$$\sigma_{<cond2>}(\sigma_{<cond1>}(R)) \equiv \sigma_{<cond1>}(\sigma_{<cond2>}(R))$$

$$\sigma_{<cond1>}(R) \cap \sigma_{<cond2>}(R)$$

↓

and

Q) Find all employees who work in dept. no. 4 & have salary greater than $10000.

Sol:

$$\sigma_{Salary \geq 10,000}(\sigma_{DNo=4}(Employee))$$

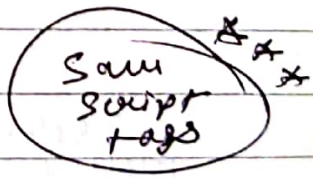
$$\equiv \sigma_{DNo=4}(\sigma_{Salary \geq 10,000}(Employee))$$

$$\equiv \sigma_{DNo=4 \text{ and } Salary \geq 10,000}(Employee)$$

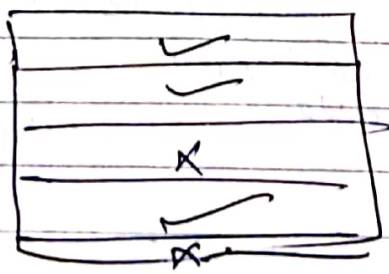
→ ∩ or and or AND or ~~AND~~ And all correct

In copy, capital letters, small letters, capital small AND, ∩,

In System, Remember these things



(vii) Horizontal Partitioning



either selects or not selects → in horizontal direct

04.08.2022 Thursday

② Project Operation (π)

π (R)
<attribute list>

(i) Vertical Partitioning

Q) Display the Fname, Lname and salary of all employees.

Sol:

π (Employee)
Fname, Lname, Salary.

π (<R>) → Relation
<Col1, Col2, Col3>
Columns you want
Print in order.

(Schema)$_{before}$ = (Fname, Lname, Salary, SSN)

(Schema)$_{after}$ = (Fname, Lname, SSN, Salary)

Case

Fname	Minit	Lname	Salary
A	B	C	5000
A	C	D	10000
A	D	C	5000

Fname	Lname	Salary
A	C	5000
A	D	10000
A	C	5000

✱ Removed as it is having unique values.

(ii)

n_{tuples} (before) $\geq$ n_{tuples} (after)

Sub cond^n = equality → when we include PRIMARY KEY field in Resulting Schema.

else $>$

(iii) Commutative ✗

$\Pi_{<list2>}(\Pi_{<list1>}(R)) \neq \Pi_{<list1>}(\Pi_{<list2>}(R))$

Cascading operation is valid iff $<list2> \subseteq <list1>$ → equivalent to $\Pi_{<list2>}(R)$

✗ List 1 = 1, 2, 3, 4, 5 List 1 = 2, 3, 4, 5 ✓
List 2 = 4, 5, 6. List 2 = 3, 4

Q) Find the Fnam, Lnam & SSN of all emp. who work in dept. no. 5.

Sol:

$\Pi_{(Fnam, Lnam, SSN)}(\sigma_{DNo=5}(Employee))$ ✓

DNo do not exist

$\sigma_{DNo=5}(\Pi_{(Fnam, Lnam, SSN)}(Employee))$ ✗

Correcting ↓

$\Pi_{(Fnam, Lnam, SSN)}(\sigma_{DNo=5}(\Pi_{(Fnam, Lnam, SSN, DNo)}(Employee)))$ ✓

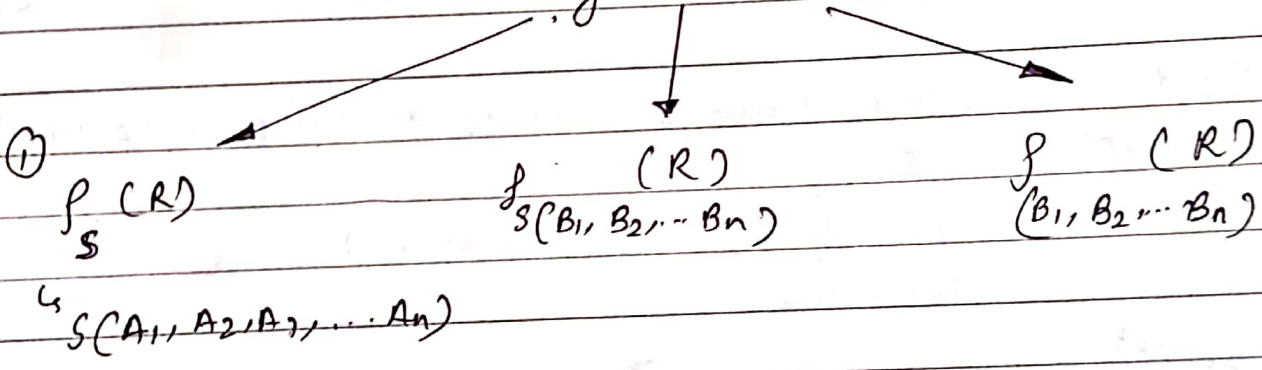
③ Rename (ρ)

(i) Creating a 2nd name for the relation name
"alias" name

Not copying

(ii) R ($A_1, A_2, A_3, \ldots A_n$)

Ways to Rename



↳ $S(A_1, A_2, A_3, \ldots A_n)$

(iii) Binary operations on same table :
(BO)

↳ Two unique Relation required

$\rho_{S_1}(R)$ BO $\rho_{S_2}(R)$

Set Operations

~~R = {a.~~

(1) Union (∪)

R = {a, b, c, d}

S = {c, d, e, f}

S ∪ R = {a, b, e, f, c, d}

unique only.

Union Compatible

To apply set (Union, Intersection and Set difference) on 2 relations R & S, they should be Union compatible

(i) The two unions relations should have same no. of attributes.

(ii) Domain of corresponding attributes of the 2 relations should be same.

~~dom (A1~~ R = (A1, A2, ... An)

S = (B1, B2, ... Bn)

dom (A1) = dom (B1), dom (A2) = dom (B2),

...., dom (An) = dom (Bn)

R ∪ S Union :- Tuples that are present either in R or S or in both.

R ∩ S Intersection :- Tuples that are present in both R & S.

R - S Set Difference :- Tuples ——→ in R but not in S.

Q) Find employees who work for DNo = 5 or who ~~supervise~~ manage dept. no 5.

Sol: ⇒ (a) → who work for DNo = 5

$\Pi_{SSn}(\sigma_{DNo=5}(Employee))$ ——→ (I)

(b) manage dept. no = 5

$\Pi_{Mgr\_SSn}(\sigma_{DNo=5}(Department))$ → II

SSN
E01
E03
E05

I ∪ II → Name of I

SSN
E01
E03
E05

Mgr SSN
E03

(9) Find employees who work on both project no. P_1 & P_2

Sol:

Work_On

ESSN	PNo.	Hours
E_1	P_1	5
E_2	P_2	6
E_3	P_2	4
E_4	P_2	8

tuple by tuple operation → only one project

Wrong: $\sigma_{PNo=P_1 \text{ and } PNo=P_2}(Work\_On)$

ESSN
E1
E2

$\Pi_{ESSN}(\sigma_{PNO=P_1}(Work\_On))$ → I

ESSN
E3
E2

$\Pi_{ESSN}(\sigma_{PNO=P_2}(Work\_On))$ → II

I ∩ II =

ESSN
E_2

Q) Find employees who work on either $pNo. = P_1$ or P_2 or both

Sol:- (I) ∪ (II) :

ESSN
E1
E2
E3

Q) Find employees who work on $pNo. = P_1$ but not on project $No. = P_2$

Sol:- (I) − (II) :

Essn
E1
E3

employees who work on $pNo. = P_2$ but not on $pNo. = P_1$

(II) − (I) :

Essn
E3
E1

Assignment Operator: give names to intermediate results.

$$R \leftarrow \pi\left(\sigma\left(\text{Rel}\right)\right)$$

or

$$R(SSN1) \leftarrow \pi\left(\sigma\left(\text{Rel}(SSN)\right)\right)$$

16.08.2022

Q) 1: $A \rightarrow \{B, C, D\}$ $B \rightarrow \{D, E\}$ CDE

$BC \rightarrow DB$

Sol: 5 Attributes → A, B, C, D, E R(A,B,C,D,E)

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

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

Q) $A \rightarrow B$, $BC \rightarrow D$, $B \rightarrow C$, $D \rightarrow A$

Sol: R(A, B, C, D)

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

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

$C^+ = \{C\}$ ✗

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

3 keys. AC^+, BC^+, DC^+

$E \rightarrow C$

Sol: $A^+ = \{A, B\}$ ✗

$B^+ = \{B, B\}$ ✗

$C^+ = \{C\}$ ✗

$D^+ = \{D, A, B, \}$ ✗

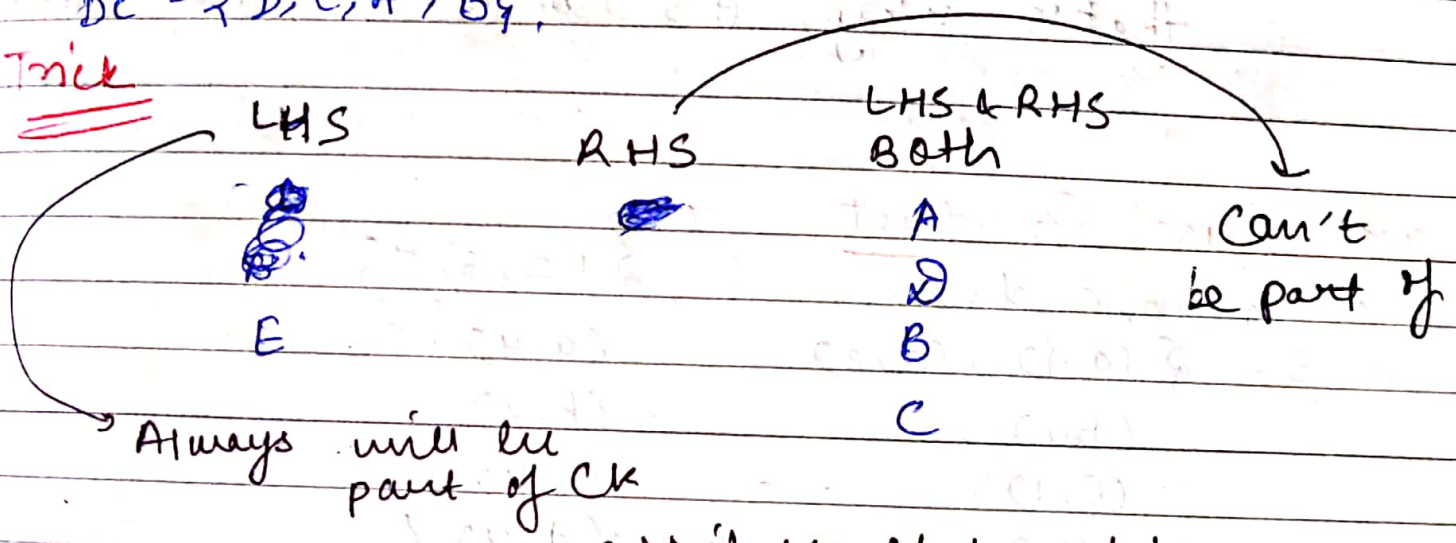
$E^+ = \{C\}$ ✗

$AB^+ = \{A, B\}$ ✗
$AC^+ = \{A, C\}$ ✗
$AD^+ = \{A, D\}$ ✗
$AE^+ = \{A, E, C\}$ ✗

$BC^+ = \{B, C, D, A\}$
BD^+ ✗
$BE^+ = \{B, E, C, D, A\}$ ✓
$CD^+ = \{C, D, A, B\}$
$CE^+ = \{C, E\}$ ✗

$DE^+ = \{D, E, C, A, B\}$ ✓
$DC = \{D, C, A, B\}$.

Trick



Key Attribute :- Attributes which are part of the candidate key

Non key Attributes :- Attributes

ABF
$AB \rightarrow F$
Dependencies (IMP)

Key Attribute $\{A, B\}$
Non $\{C, D, E\}$

$AB \rightarrow CD$
$AB \rightarrow BC$
$AB \rightarrow B$ } Trivial
$AB \rightarrow A$ }

⑤ Normalisation (IMP)

Trivial fn depending

$AB \rightarrow A$ ✓

$AB \rightarrow BC$ ✗

$AB \rightarrow DE$ ✗

Completely Non-Trivial fn depending

17-08-2022
Wednesday.

Key Att. = B, D, E
Non key Att = C, A

E → A (or D → A)
Since these exists Partial dependencies
that's why it's not a 2nd Normal form.

Cartesian Product

R = {a, b, c, d} S = {1, 2, 3, 4}
R × S = {(a,1), (a,2), ... (a,4),
(b,1), ... (b,4),
(c,1), ... (c,4),
(d,1), ... (d,4).}

① Combining two data. (Multiple Relations)

② Not Union Compatible. ∴ Rarely used.

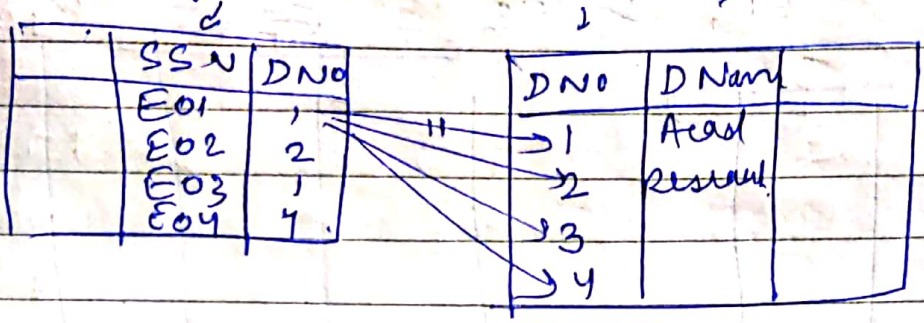
③ $R(A_1, A_2, \ldots, A_n) \times S(B_1, B_2, \ldots, B_n)$
n_R tuples n_S tuples

$\equiv T(A_1 A_2 \ldots A_n, B_1 B_2 \ldots B_m)$
$n_R \times n_S$ tuples.

⇒ SQL queries → Relational Algebra

Q) Find the name of employees who work in 'Research' department.

Sol: (Employee × Department)



= → meaningful

$$\pi_{fname, minit, lname}\left(\left(\sigma_{DNO=Dnumber \text{ and } Dname="Research"}(Employee \times Department)\right)\right)$$

Generally use select operation with C

Q) Find the name of employees who work project no 'P1'

Sol:

$$\pi_{fname, minit, lname}\left(\sigma_{PNO='P1' \text{ and } SSN=ESSN}(Employee \times Works\_on)\right)$$

Join Cond^n

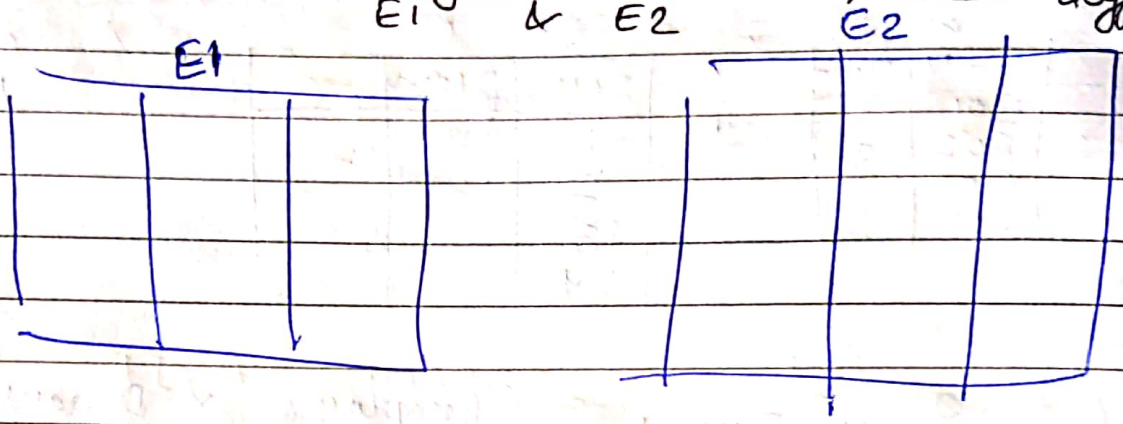
Name ⊗ ESSN ; SSN PNO

Method-II

Q) Find the employee who earns highest salary in company.

① $\rho_{E_1}(\text{Employee})$ $\rho_{E_2}(\text{Employee})$

// Rename (binary operations required different)
// aliases of employee table as E1 & E2



② $R_1 \leftarrow \pi_{E_1.SSN}(\sigma_{E_1.salary < E_2.salary}(E_1 \times E_2))$

③ $R_2 \leftarrow \pi_{SSN}(\text{Employee})$

④ $\text{Result} \leftarrow R_2 - R_1$

Q) Find the employees who earn highest salary in DNO = 5

Sol:

$\rho_{E_1}(\sigma_{DNO=5}(\text{Employee}))$ or $E_1 \leftarrow \sigma_{DNO=5}(\text{Employee})$

$\rho_{E_2}(\longrightarrow)$ $\rho_{E_2}(E_1)$

Q) Find the employees who earn the lowest salary.

just change sign ≥ to ≤

JOIN operations

Not a fundamental operation.
can't be expressed in terms of other operation

① $\sigma_{\langle R.A \rangle \langle op \rangle \langle S.B \rangle}(R \times S) \equiv R \bowtie_{\langle R.A \rangle \langle op \times S.B \rangle} S \equiv T$

join cond^n — join cond^n

ESSN = SSN

using relation name to qualify attribute names

op = !=, <, >, ≤, ≥, =

$R(A_1, A_2, \ldots A_n)$ — n_R tuples

$S(B_1, B_2, B_3 \ldots B_n)$ — n_S tuples

$T(A_1 A_2 \ldots A_n, B_1 B_2 \ldots B_m)$

② No. of tuples in T = ~~$n_R \times n_S$~~

No. of tuples in T $\leq n_R \times n_S$

Q) Find the name of employees who work in 'Research'

Sol:

$\pi_{Fname, Lname}(\sigma_{Dname = 'Research'}((Employee \bowtie_{(Employee.Dno = Department.Dnumber)} Department)))$

Join cond^n → Equi Join

or directly Employee

Dno = Dnumber // different name

(1) Theta Join $R \bowtie_\theta S$ $\theta = \langle R.A \rangle \langle op \rangle \langle R.B \rangle$ $\langle op \rangle = <, >, \leq, \geq, !=, =$

(Generalised Join)

(2) Equi Join $R \bowtie S$ $\langle R.A \rangle = \langle S.B \rangle$ ↓ equality only

(Special kind of Theta Join)

(3) Natural Join

(Special form of equi join where ~~equality~~ equality operation is performed only on common attributes

Same name & domain

NOTE

but the common attribute will appear only one in the Result.

(i) $R(A, B, C, D)$ $S(A, B, \cancel{E, F})$

$R \bowtie S \equiv R \bowtie_{R.A = S.A \text{ and } R.B = S.B} S = T(A, B, C, D)$

(ii) $R(A, B, C, D)$ $S(A, \cancel{B}, E, F)$

$R \bowtie S = R \bowtie$

18.02.2022 Thursday.

Q) Find a name of manager of each dept.

Ans: $\pi_{Dnumber, Fname, Minit, Lname}$ (Department ⋈ Employee)
MgrSSN = SSN

Department.MgrSSN = Employee.SSN

Q) Display the name & all locations of each deptt.

Sol: Dept_locations (Dnumber, Dlocation)
Dept (Dnumber, Dname, ...)

Same

Natural Join

$\pi_{Dname, Dlocations}$ (Department ⋈ Dept_locations)

Common attribute come one only

Dnumber Dname MgrSSN MgrR... Dlocation

Join cond^n Department.Dnumber = Dept_locations.Dnumber

OUTER JOIN (Natural Join).

R

A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_1	c_3

S

B	D	E
b_1	d_1	e_1
b_3	d_2	e_2
b_4	d_2	e_1

Inner Join = Natural Join

$R \bowtie S$ (Natural Join)

A	B	C	D	E	F
a_1	b_1	c_1	d_1	e_1	f
a_2	b_4	c_3	d_1	e_1	f

Information
- $R.b_2$ is lost
- $S.b_3$ is lost
- $S.b_4$ is lost

→ Left ~~two~~ will come

(I) Left Outer Join — $R \overset{L}{\bowtie} S$

If a tuple in the left relation is not having a matching tuple in the right relation then also it will be retained in the o/p relation.

$R \overset{L}{\bowtie} S$

A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_1	b_2	c_1	Null	Null
a_2	b_1	c_3	d_1	e_1

(II) Right Outer Join — $R \overset{R}{\bowtie} S$

$R \overset{R}{\bowtie} S$

A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_1	c_3	d_1	e_1
Null	b_3	Null	d_2	e_2
Null	b_4	Null	d_2	e_1

(IV) Full Outer Join — $R \overset{F}{\bowtie} S$

↳ Combo of Left OJ ~~and~~ Right OJ

A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_1	b_2	c_1	Null	Null
a_2	b_1	c_3	d_1	e_1
Null	b_3	Null	d_2	e_2
Null	b_4	Null	d_2	e_1

Q) Union vs Outer Join

Sol: (1) Union compatible — Not necessary.

(2) Adding ~~Count~~ of tuple — Comp^n of all possible

Division Operation { Comp^n of one attribute with all other attributes }

(Derived operator)

Student

SNo.	Sname
S_1	
S_2	
S_3	

Course

CNo	CName	Credit
C_1		
C_2		
C_3		

Enrollment

SNo	CNo	grade
S_1	C_1	
S_2	C_1	
S_3	C_1	
S_1	C_2	
S_2	C_2	
S_1	C_3	

Q) Find students which have taken one or other courses

Sol: Π_{SNo} (Enrollment)

Π_{sid}(Enrolled

Q) Find students who have enrolled in all the courses.

Sol:

$R \leftarrow \Pi_{SNo, CNo}$ (Enrollment)

$S \leftarrow \Pi_{CNo}$ (Course)

Result $\leftarrow R \div S$

CNo.

SNo → ( )

all

// SNo.

$T(Y) = R(Z) \div S(X)$

unknown — helper

where $X \subseteq Z$ and (X is subset of Z).

$Y = Z - X$

NOTE: Take necessary elements only while ÷ due to $X \subseteq Z$ cond^n.

Don't apply whole rel^n.

T will have tuple t such that tuples t_R appear in R $t_R[Y] = t$ and with $t_R[X] = t_S$ for every tuple t_S in S.

$t_R[SNO] \rightarrow t_S[CNo]$

$t_R[Y]$

$t_R[SNO] = S_1 \rightarrow t_S[CNo]$ c_1

c_2

Q) Find the employees who work on every project controlled by Dept No = 5

Sol:

$$R \leftarrow \pi_{ESSN, PNo.}(Works\_on)$$

$$S \leftarrow \pi_{Pnumber}(\sigma_{DNum=5}(Project))$$

$$Result \leftarrow R \div S$$

Q) Find the employee names for above query

Sol: Let $T \leftarrow R \div S$ (or $T \leftarrow Result$)

$$Result \leftarrow \pi_{SSN, Fname, Minit, Lname}(Employee \bowtie_{SSN = ESSN} T)$$

24.08.2022 Wednesday

AGGREGATE FUNCTIONS

(apply on attributes domain = Numbers)

max, min, count, sum, avg

count ↓ can be applied on any

max (R) Attribute

→ "single column, single row relation"

ex. Find the maximum salary of employees in the company.

Sol: $\rho_{(max\_sal)} \leftarrow max_{salary}(Employee)$

max_sal
50000

Q) Find the avg. salary of emp. in DNo = 5

Sol: $avg_{salary}(\sigma_{DNo.=5}(Employees))$

Q) Find employees who earn max salary (highest) in the company.

Sol:

$\sigma_{Salary = max(salary)}(Employee)$

→ This is not value, this is relation.

Write is as considering relation

① $\rho_{R(max\_salary)}(max_{salary}(Employee))$ // Renaming relation & attribute name

② $\sigma_{salary = max\_salary}(R \times Employee)$

Grouping functions $\mathcal{F}$ or G

max, min and

Q) Find the average salary of each department

Sol.

$$\mathcal{G}_{\langle \text{grouping functions} \rangle \mathcal{F}_{\langle \text{aggregate functions} \rangle}}(\langle R \rangle)$$

avg	DNO = 1
avg	DNO = 2
avg	DNO = 3
avg.	DNO = 4

$\mathcal{G}_{DNO}$ (Employee)
avg(salary), max(salary), min(salary)

DNO.	avg(salary)	max(salary)	min(salary)

Grouping attributes — Aggregate functions.

Q) Find ——— with depart name

Sol: method: 1

① $R\left(\begin{matrix} DNo., avg\text{-}sal \\ max\text{-}salary, \\ min\text{-}Salary \end{matrix}\right) \leftarrow {}_{DNo}\mathcal{G}$ (Employees)
avg(sal), min(sal), max(sal)

② $\pi_{Dname, avg\text{-}sal, max\text{-}sal, min\text{-}sal}$ (Department $\bowtie_{DNO = Dnumber}$ R)

Department Name	DNO	avg-sal	max sal	min sal

Method-2

$$\gamma_{Dept\_name,\ avg(sal),\ max(sal),\ min(sal)}\left(Employee \bowtie_{(DNo. = Dnumber)} Department\right)$$

NOTE

Grouping functions are always used with Aggregate functions.

Q) Find the ——— in each department of male & female.

Sol:-

$$\gamma_{DNo,\ Sex\ avg(salary),\ min(salary),\ max(salary)}(Employee)$$

DNo	Sex	avg(sal)	max(sal)	min(sal)
1	M			
1	F			
2	M			

Questions

(1) Find name of emp. of DNO = 5 (Employees) who works on more than 10 hrs a week on Project Product X (Work-on, Project)

(2) Find name of emp. who have dependent with same first name as themselves.

(3) Find name of employees who are directly supervised by Franklin Wong.

(4) For each project list project name and total no. of hours per week spend by all employees on that project.

Sol: (1) (i) R ← Employees ⋈ (SSN = ESSN) Work-On ⋈ (Pno = Pnumber) Project

Employees				Work-on			Project		
-	-	-	-	-	-	-	-	-	-

(ii)

$$S \leftarrow \sigma_{DNO=5 \text{ and } hours \geq 10 \text{ and } Pname = "Product-X"}(R)$$

(iii)

$$\Pi_{Fname, Minit, Lname, SSN}(S)$$

(2) (i) $R \leftarrow (\text{Employee} \bowtie_{SSN=ESSN} \text{Dependent})$

(ii) $S \leftarrow \sigma_{Fname = DependentName}(R)$

(iii) $\pi_{Fname, Lname, Minit, SSN}(S)$

(3) Emp(Fn.. SSN... SuperSSN
EO1 EOS

(i) $\rho_{E_1}(\text{Employee})$ $\rho_{E_2}(\text{Employee})$

// Employee // Supervisor

Emp
EO1
EOS

(ii) $R \leftarrow (E_1 \times E_2)$

(iii) $\sigma_{E_1.SuperSSN = E_2.SSN \text{ and } E_2.Fname = \text{'Franklin'} \text{ and } E_2.Lname = \text{'Wong'}}(R)$

$E_1 \bowtie_{SuperSSN} E_2 \bowtie_{SuperSSN} E_3$

(4) (i) $R(PNo, total\text{-}sum) \leftarrow {}_{PNo}\mathcal{G}_{sum(hours)}(Works\text{-}on)$

(ii) $\pi_{Pname, total\text{-}hours}(R \bowtie_{Pno = Pnumber} Project)$

Q) (5) Find name of all employees who work on every project.

(6) Find name of employees who do not work on any project.

Sol: (5) (i) $R \leftarrow \pi_{ESSN, PNo}(Work\_on)$

(ii) $S \leftarrow \pi_{Pnumber}(Project)$

(iii) $R \div S$

(6) (i) $R \leftarrow \pi_{ESSN, PNo}(Work\_on)$ // Employees who work on one or more project

(ii) $S \leftarrow \pi_{SSN}(Employee)$ // Employees who work 0 or more project

(iii) $S - R$

Q) (7) Find emp. who work on atleast 2 diff. projects

Sol: (7) Method 1.

$\sigma_{count(ESSN) \geq 2}(\rho_{R(ESSN, count(ESSN))}({}_{ESSN}\mathcal{G}_{count(ESSN)}(Work\_on)))$

ESSN	Pno.	hours
E1	P1	5
E2	P1	6
E1	P2	8
E3	P2	
E3	P4	
		10
E1	P5	9

Method-2

(i) ρ_{W_1} (Work-on) ρ_{W_2} (Work-on)

(ii) $\pi_{W1.ESSN}\left(\sigma_{W_1.ESSN = W_2.ESSN \text{ and } W_1.Pno! = W_2.Pno}(W_1 \times W_2)\right)$

(8) Find ——— atleast ——— 3 diff.

Sol:- (8) (i) ρ_{W_1} (Work-on) ρ_{W_2} (Works-on) ρ_{W_3} (Work on)

(ii) $\sigma_{W1.ESSN = W2.ESSN = W3.ESSN \text{ and } W1.pno! = W2.Pno.! = W3.Pno.}(W_1 \times W_2 \times W_3)$

(9) Find emp. who work on exactly on 2 diff. projects

Sol (9)

25.08.2022 Thursday

Q① For each dept retrieve dept-name & avg. salary of all female & male employees.

② Find last name of all dept managers who have no dependents

③ Find name & addresses of all employees who work on atleast one project located in Houston but who department has no location in houston.

Sol: ① (i) $R \leftarrow$ Employee $\bowtie_{DNO=DNUMBER}$ Department

(ii) $\mathcal{G}_{Dname, Sex, avg(salary)}(R)$

② (i) $R \leftarrow$ Department $\bowtie_{MGRSSN=ESSN}$ Dependent

// Managers who have dependents

(ii) $S_1 \leftarrow \Pi_{MGR\text{-}SSN}(R)$ // Manager with dependents

$S_2 \leftarrow \Pi_{MGR\text{-}SSN}(Department)$ // Total Manager

(iii) Ans $\leftarrow S_2 - S_1$

(3) (i) $R \leftarrow \pi_{ESSN}(Works\_on \bowtie_{Pno = Pnumber} Project)$

(ii) $S_1 \leftarrow \sigma_{Plocation = 'Houston'}(R)$

(iii) $S_2 \leftarrow \pi_{ESSN}(\sigma_{Dlocation = 'Houston'}(Employee \bowtie_{Dno = Dnumber} Dept\_locations))$

(iv) Result $\leftarrow R - S$

8) (4) For each dept. whose avg. employee salary is more than 80,000 retrieve department name & no. of employees working for that department.

Sol: (4)

(i) $R(Dname, avg\_sal, noof\_emp) \leftarrow {}_{Dname}\mathcal{G}_{avg(salary),\ count(*)}(Employee \bowtie_{Dno = Dnumber} Department)$

Dname	avg(salary)

(ii) $\pi_{Dname,\ no\_of\_emp}(\sigma_{avg\_sal > 80,000}(R))$

(5) For every project located in Stanford list Pno. controlling dept No, dept Managers last name & address.

Sol: (5)

Method-1

(i) $R \leftarrow (Department \bowtie_{Dnumber = Dnum} Project)$

Dep	Project
D01	1
D01	2
D01	3

(ii) $S \leftarrow (R \bowtie_{mgrSSN = SSN} Employee)$

(iii) $\Pi_{Pno, DeptNo, lname, address} (\sigma_{Plocation = 'Stanford'} (S))$

Method-2

$\Pi_{Pno, DeptNo, lname, address} (\sigma_{Plocation = Stanford} (Project \bowtie_{DNO = Dnumber} Department \bowtie_{mgrSSN = SSN} Employee))$

ER Diagrams

Peter Chan (1960)

Entity Relational Models.

Physical / Abstract

Entity Type

Entity Set

Entity ~~Relation~~ Instance

Student Name	Id.
Ram	01
	02

← Collection of actual

→ (actual info)

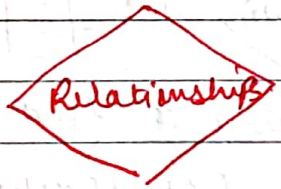
One instance of student entity.

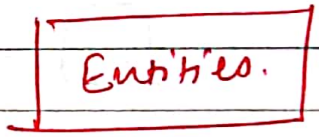
NOTE: Names of entities should be meaningful

Relationships

→ Association bet^n diff. entities.

→ Foreign Key type.



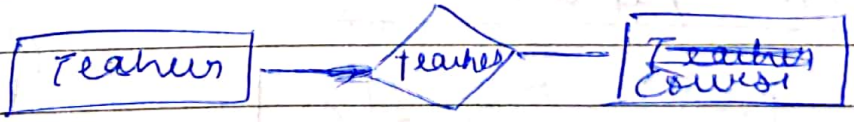


Entities part of Relationships are called Participating Entities

Nouns : Entities

Verbs : Relationship

Degree



Binary Relationship

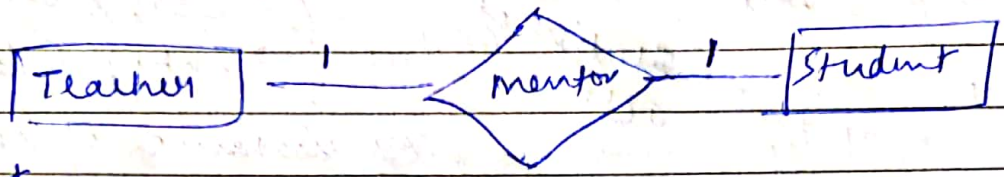
When there are 2 Participating Relationships,

Cardinality Constraints

Relationship Cardinalities

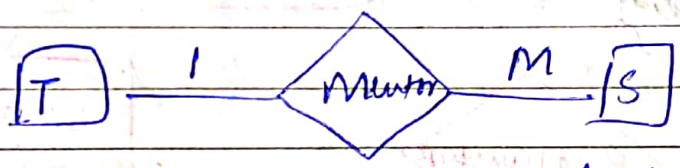
- One to One
- One to many
- many to One
- many to many

One to One



One ~~teacher~~ student has only one mentor.
One teacher can be mentor of one student only

One to many



M/N

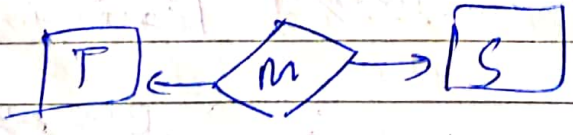
One teacher can mentor many student
~~A~~ Student can have only one Teacher as mentor

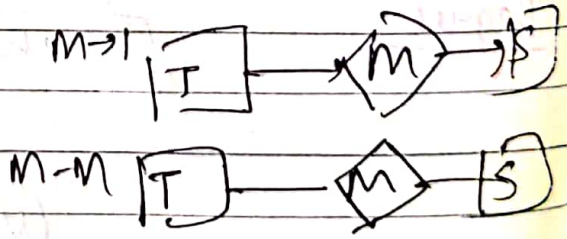
Trick

Use arrow → when 1
& nothing → when M

for Arrow representation

One to one



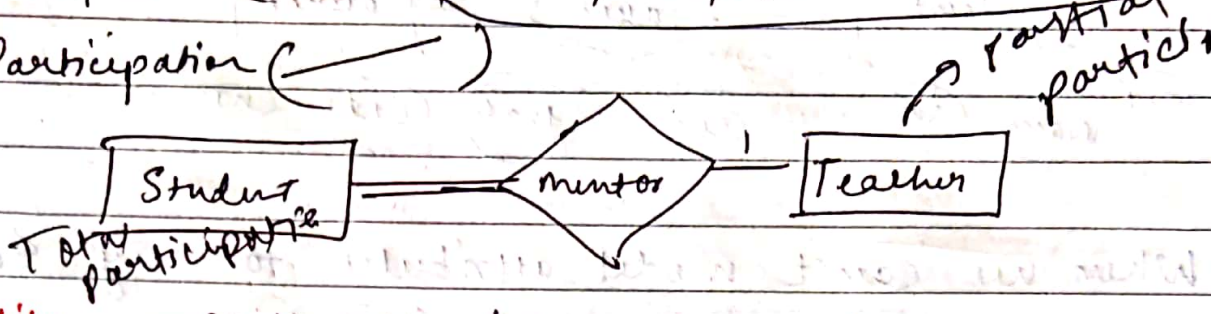


One to many

27.08.2022 Saturday.

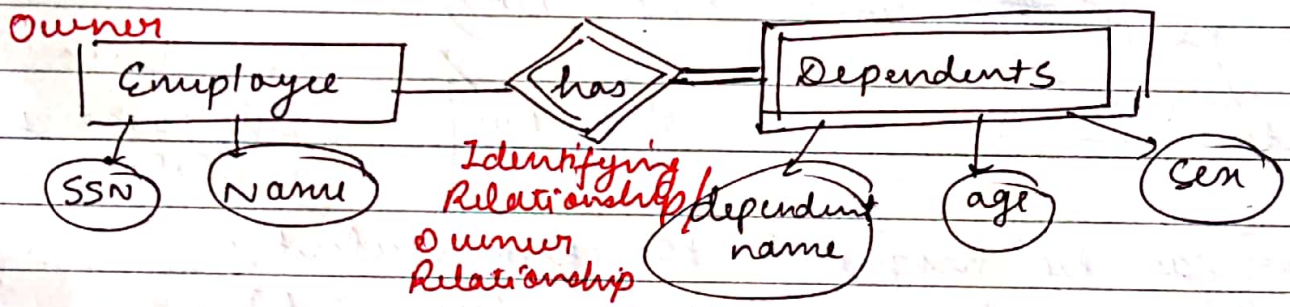
Participation

- Total Participation (=) → In General, entity on many side is total participation.
- Partial Participation (-)



Strong Entity → Have primary key of its own.

Weak Entity Type → Not PK



Partial Key or Discriminator.

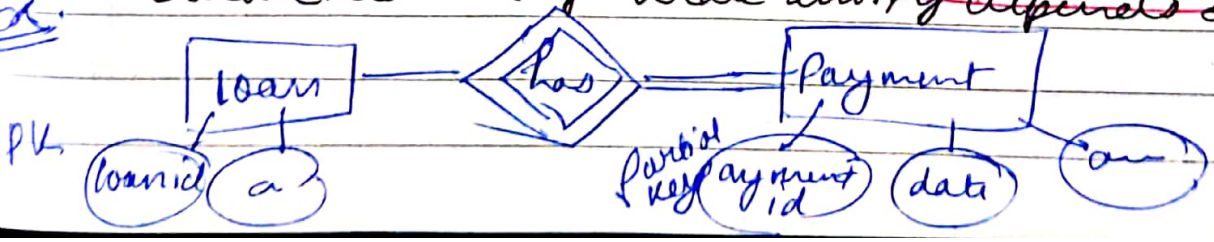
SSN + Dependent Name. → uniquely defines
(PK) (Partial Key)
→ but vice versa not true.

Weak Entity is always in Total Participation with Strong Entity.

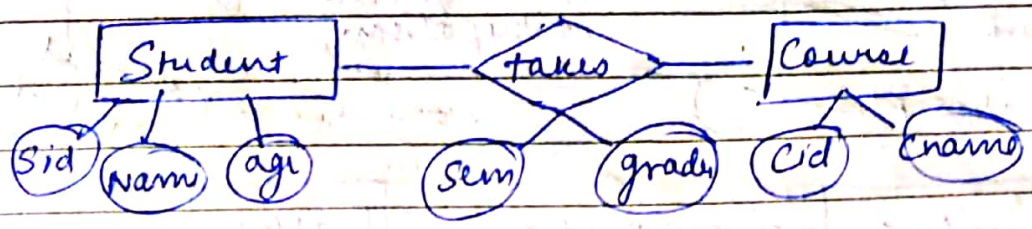
PK of weak entity type = Partial Key (of Weak Entity type) + PK of owner entity type

~~Each~~ Existence of weak entity depends on strong entity.

Ex.

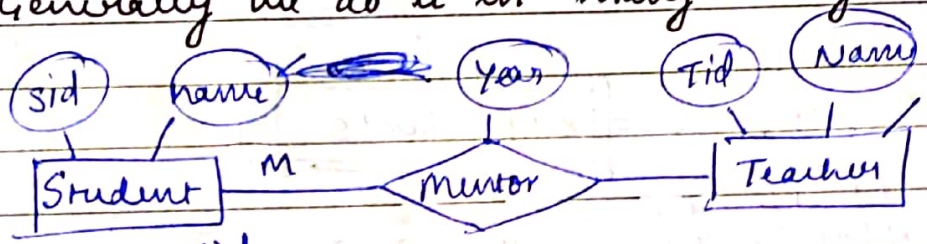


Relationships can also have attributes :-



(1) When we can't model attributes to any Entity then we consider it in Relationships
[Properties of Student - Course Relationships]

(2) Generally we do it in Many - many relationships.



∆ One to many Rel^n.

Year can be migrated to either student.
Student (Sid, name, Year, Tid)
↑ Tid

? Teacher (Tid, name, Sid, ~~Year~~)
↳ redundancy of data.

NOTE
→ In One to many Relation →
Relationship attribute can be migrated to many side entity

TYPES OF ATTRIBUTES

① Simple vs Composite :-

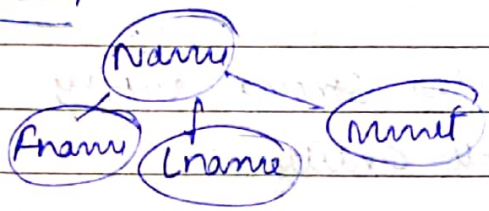
↓ Simple: atomic

↓ Composite: Attributes which can be broken further

ex. Name - Fname, Lname, Mname

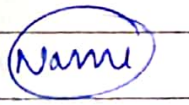
Address - House No, Street No, City, State, Country, Pin code

Represent



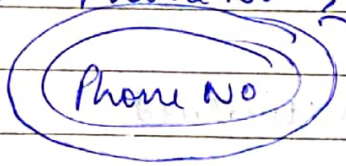
② Single Valued vs Multivalued :-

ex. Scholar No, Name



↓ Multivalued: Attr. which can take many values

ex.: Phone No., email id



One person can has many phone no.

③ Stored vs Derived :-

↓ Derived: "age" which can be derived from stored attribute DOB

(curr Date - DOB)

④ Null Valued Attributes :-

unknown / unavailable / unapplicable

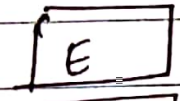
↓ phone No. not known. then NULL

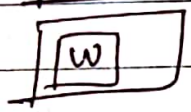
↓

↓ Salary slab

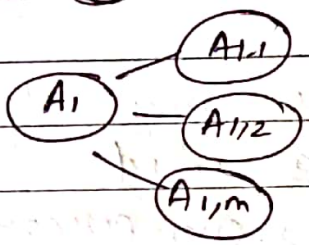
No Representation

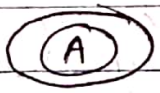
Representation (Quick).

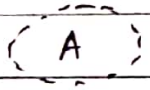
①  Regular Strong entity

②  Weak Entity

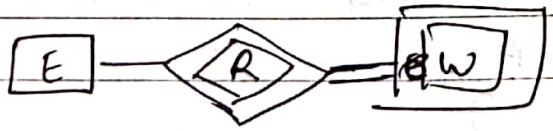
③  Simple Attributes

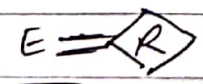
④  Composite Attr.

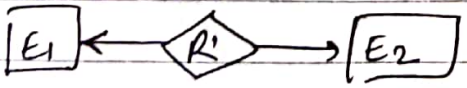
⑤  Multivalued

⑥  Derived

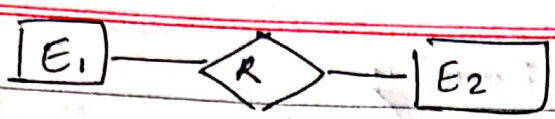
⑦  Relationships

⑧  Identifying Relationships

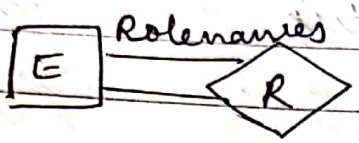
⑨  Total Participation

⑩  One to One

⑪  One to many

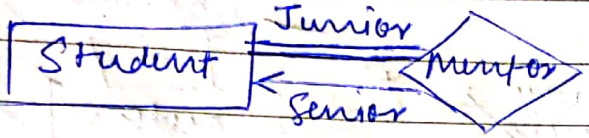
(12) 

many to many

(13) 

Self Relationships / Recursive —

ex. each junior will have a senior mentor

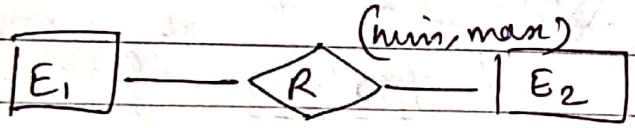


There is only one senior for

= Every junior student must be mentored by one senior only

many to one

Role name → Junior , Senior

(14) 

(min, max) constraint

Each student can take min 1 or max 5 courses

↓

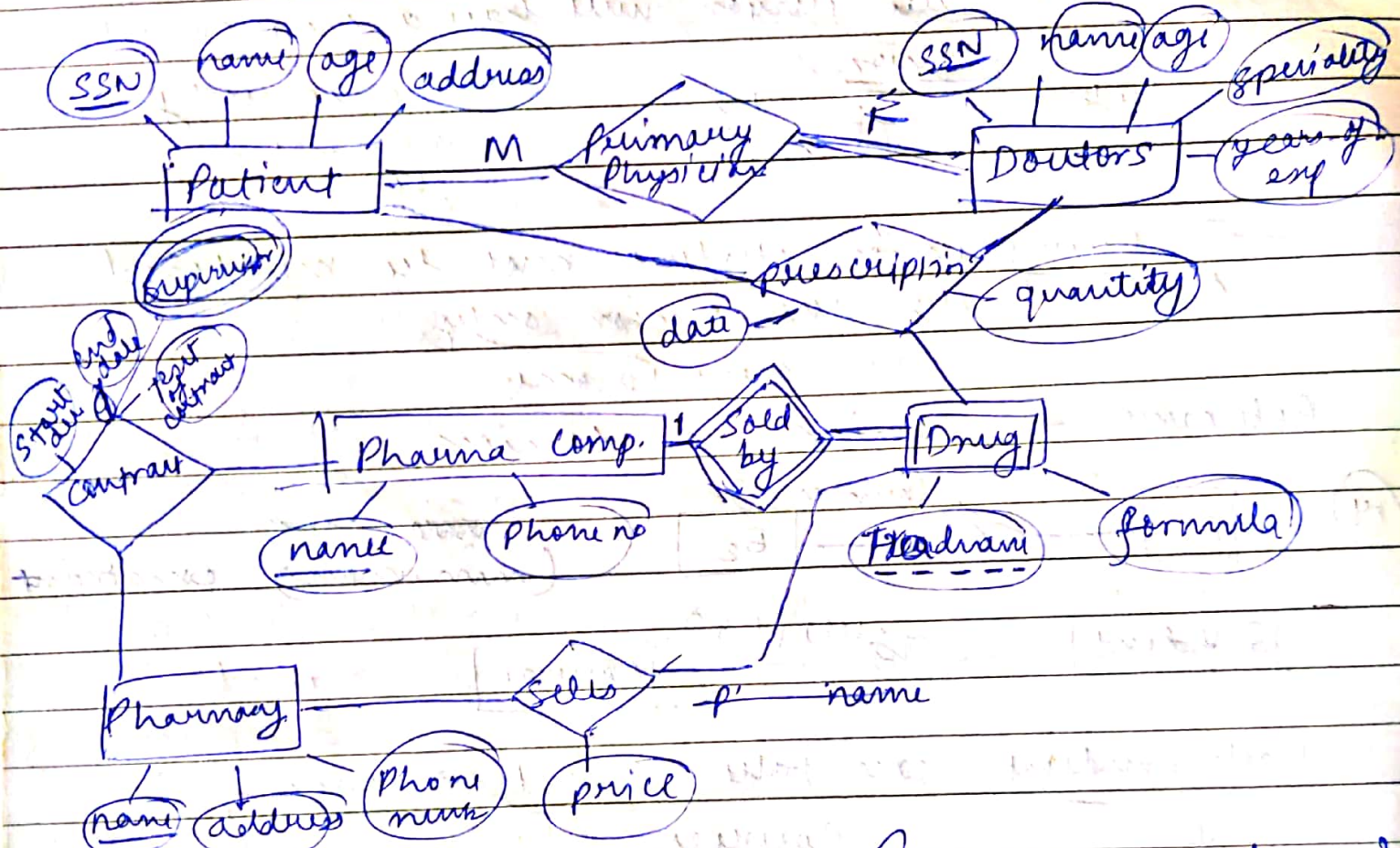
total participation

Whatsapp

Common Nouns → Entity

verb → Relationship (Sometimes can be nouns)

9) Prescription R-x chain



For some Cardinality we use our common sense

(Pharma name, Tradname)
↓
uniquely identifies

Prescription
↓
Ternary Relationship

* (You can assume that, if a doctor prescribes the same drug for same patient more than one . . .)

29.08.2022
Tuesday

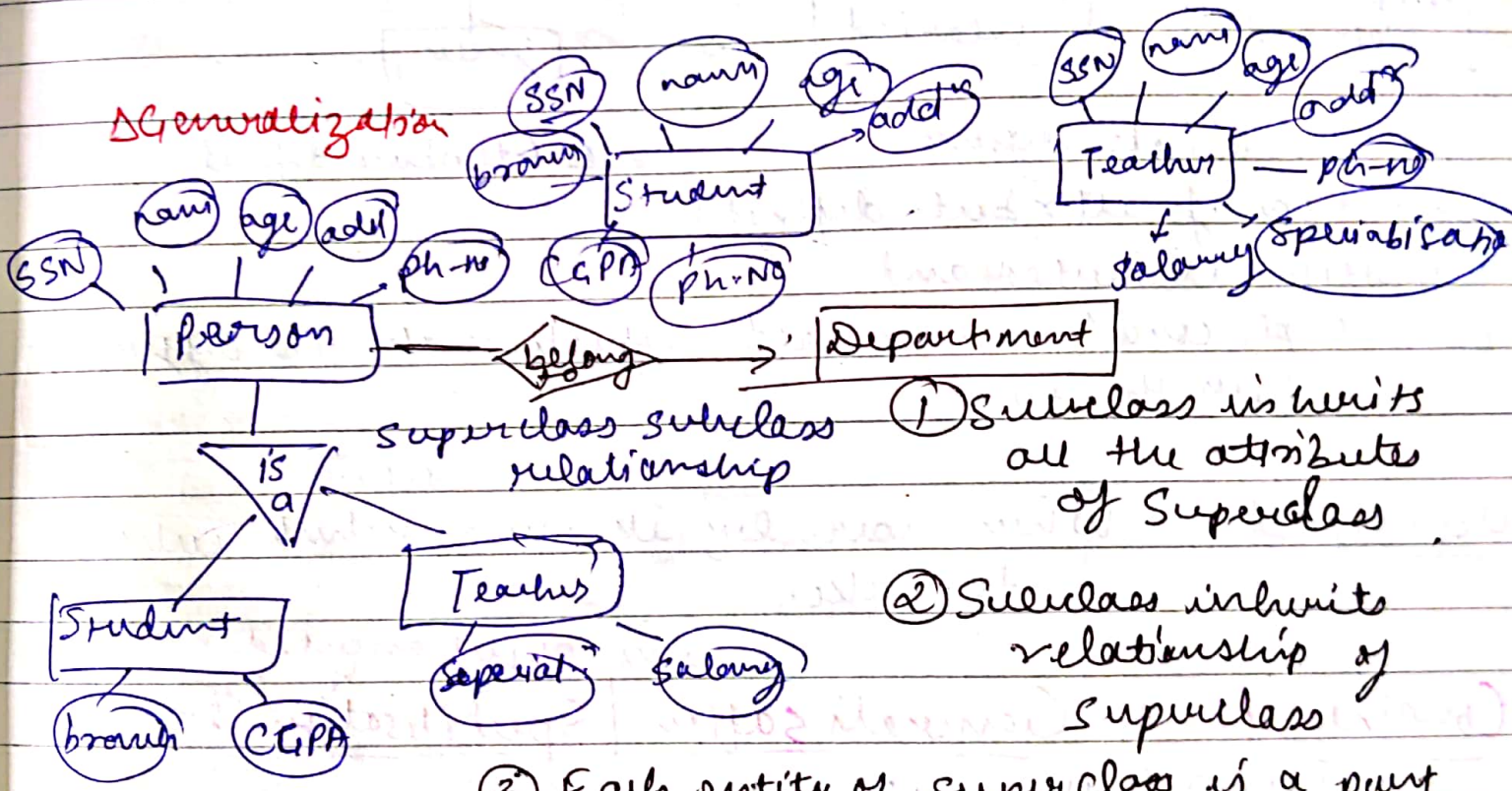
Fundamentals.
Keys: Relational Algebra.
ER models (Concepts)

Existence Dependency

Existence of Weak Entity depends on Strong Entity.

Generalization / Specialization

(Bottom up) (Top-Down)

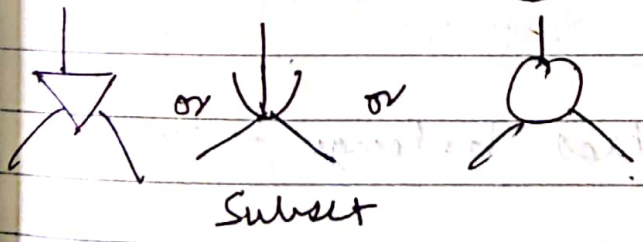


superclass subclass relationship

① Subclass inherits all the attributes of Superclass.

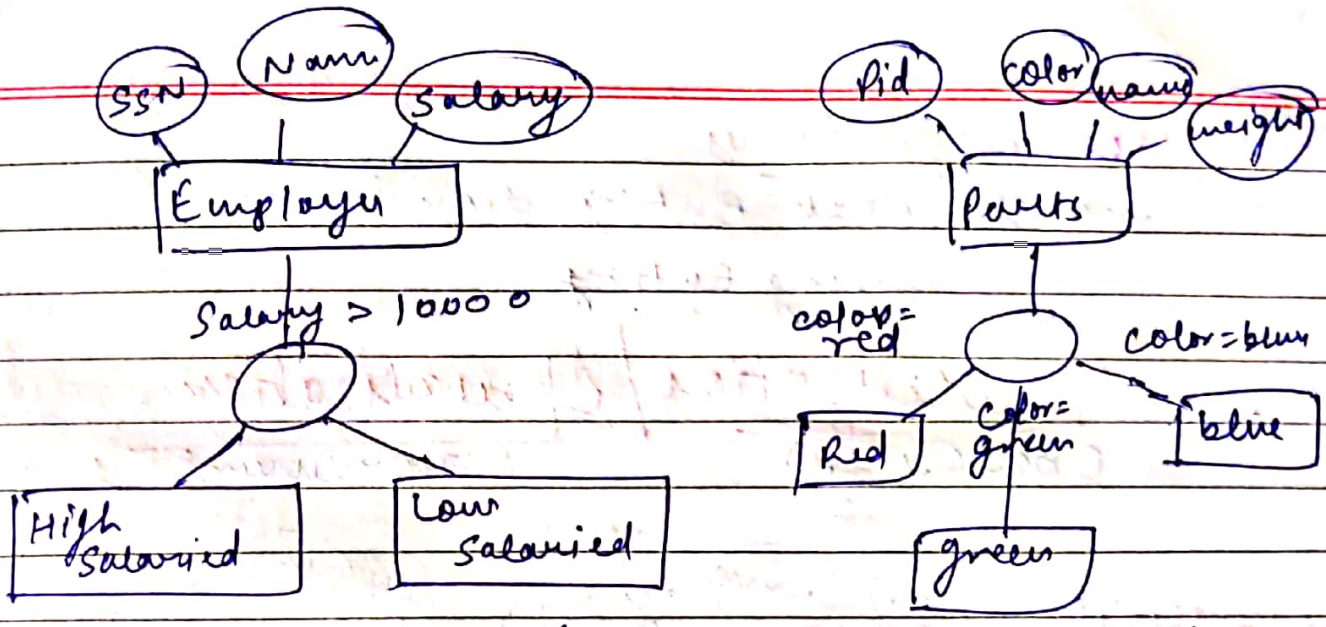
② Subclass inherits relationship of superclass

③ Each entity of superclass is a part of subclass but reverse may not be true.



Q) How to define generalisation / specialization.

Sol:
Predicate defined
Attribute defined
User defined



△ Predicate defined.

△ Attribute defined

(is a type of attribute defined)

[Multiple attributes and multiple cond^n can be used which make the difference btw the two.

User defined : When user by its own defines which subclass to take.

Constraints on Generalisation | Specialisation :-

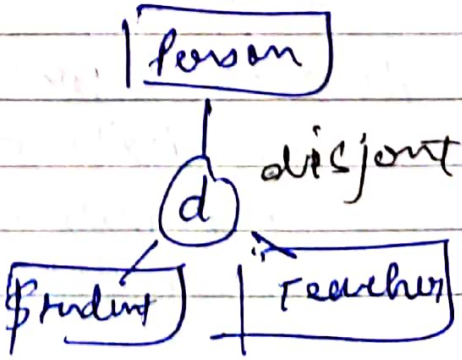
(some object oriented model here)

① Disjoint (only one) / Overlapping (atleast one)

each of entity of superclass belongs to only one of the subclass

one or more of the subclass.

Not a disjoint when PhD

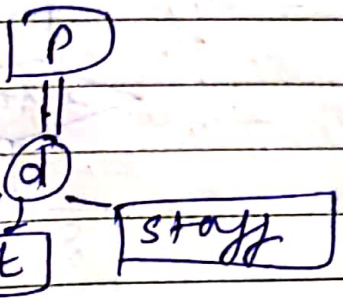


(must) (may)

(=) Partial

② Total / ~~Participation~~

↳ each entity of superclass ~~is a part of~~ belong to any one subclass.



each person must belong to either belongs to S, E or staff

↓ total

↓ disjoint "d"

total

Total + Disjoint → each entity of the superclass must belong to only one of the subclasses.

total

Total + Overlapping → each entity of sup. class must belong to atleast one of subclass.

Partial + Disjoint → each entity of superclass may belong to only one of the subclass

Partial + Overlapping → each entity of superclass may belong to atleast one of the subclass.

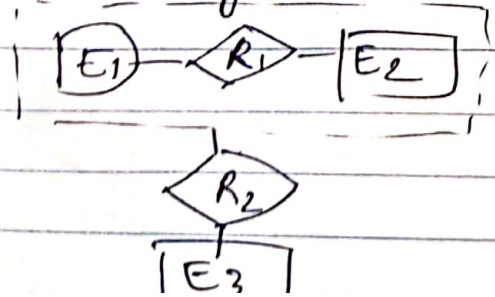
overlapping.

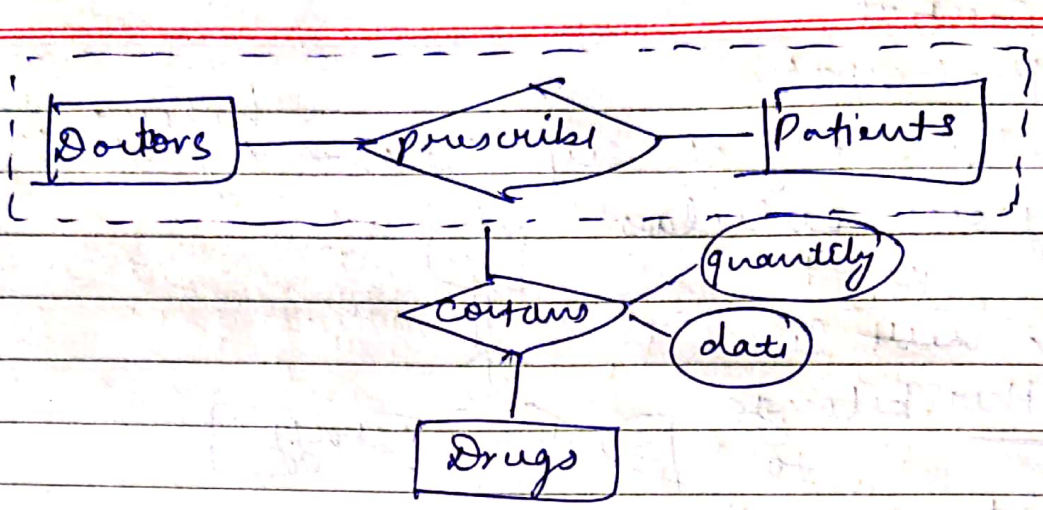
symbol - g

AGGREGATION (Diff. in object oriented model & here)

→ Relationships bet^n relationships.

→ Modeling relation bet^n Relation & Entity





- Each prescription contains Drugs of same quantity on some date

M to M (many to many)

- A doctor can prescribe many patients
- A patient can be prescribed by many doctors

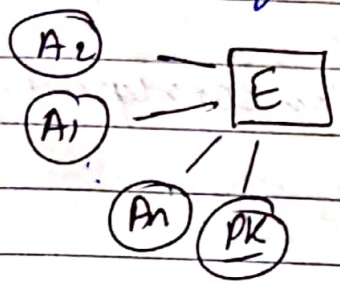
Prescription & Drugs can be linked through (Pid, ~~Date~~ Did)

06.09.2022
Tuesday

er diagram
draw: ID
Rationale

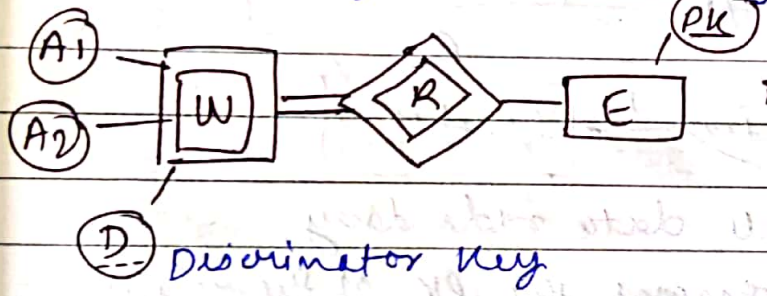
ER to Relational Mapping

① Regular / Strong Entity
(having Primary key)



⇒ R (A1, A2, ..., An, PK)

② Weak Entity
(does not have PK of its own)
Has Partial key / Discriminator key



Strong Entity (E)
S (~~PK~~ ..., PK)

R (A1, A2, ... D, PK)
(FK)

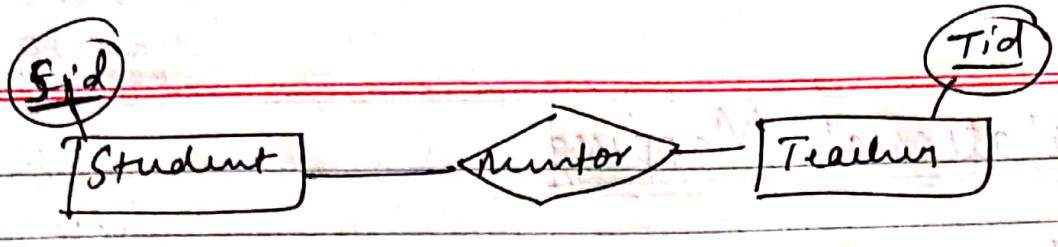
Weak Entity (W)

~~Disc~~

Relationships

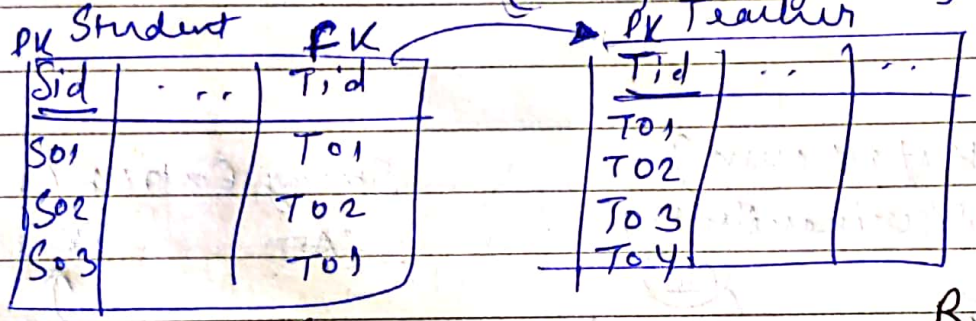
① One to One Relation

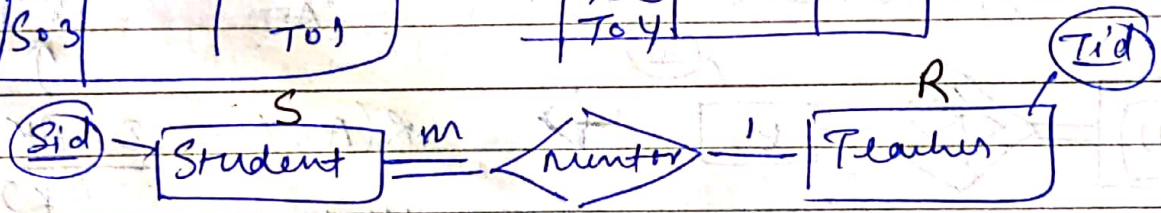
a) Merged relation approach:
Suppose R → E1 and S → E2
Combine / merge the relation R and S and the PK of either of the relations can act as the PK of the merged relation. It is valid only if E1 and E2 participates only in the relation R and no other relation.



b) Foreign Key Approach: Import the PK of either of the relation as FK in the other relation.
Generally the FK is imported in the relation corresponding to the entity having total participation.

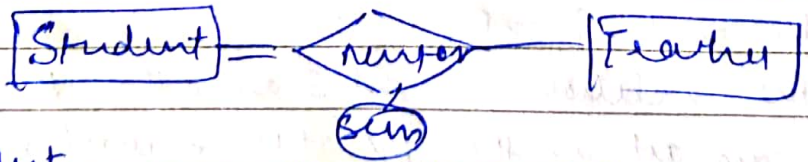
(compulsory side)





In opp. fasion it will creat data redundancy.

② One to Many Relation Import the PK of the relationship of the relation R corresponding to the entity on the one side as FK in the relation S corresponding to the entity on the many side of the relationship S.
Also include any simple attributes of the relation-ship in relation S.



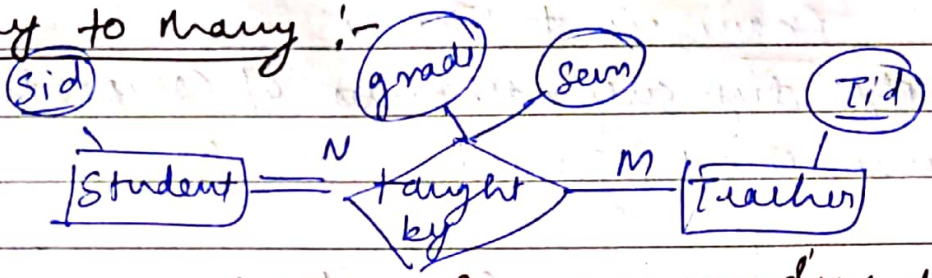
Student

Sid	..	Tid	Sem

Simple attributes
Not derived,
multivalued,
composite

(3) Many to One :- /* Sams do ~~no~~ one to many */

(4) Many to Many :-



Create a new Relation S corresponding to many-to-many relationships R.
Import PK of the relations corresponding to both the participating entities as FK in the relation S. Also include any simple attributes of relationship in S.

FK1 FK2
Taught_By (Sid, Tid, Sem, grade)

Student (Sid, ~~Tid~~ ...)
Teacher (Tid, ...)

PK of S will be combination of PK of all participating entity type

Attributes :

(1) Stored vs Derived
DOB age

R (DOB, age, ...)

Get ~~Store~~ both the attributes in relations as entities

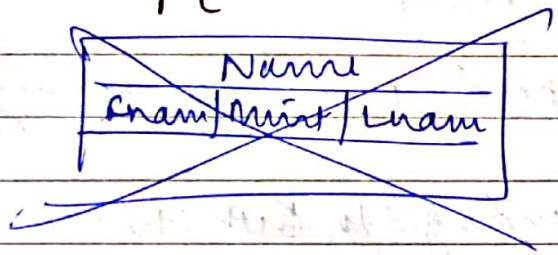
② Simple vs Composite

Sch.No. Name

Fname, Minit, Lname

Stored it simply by sub attributes of composite attr.

Emp (Fname, Minit, Lname, ...)

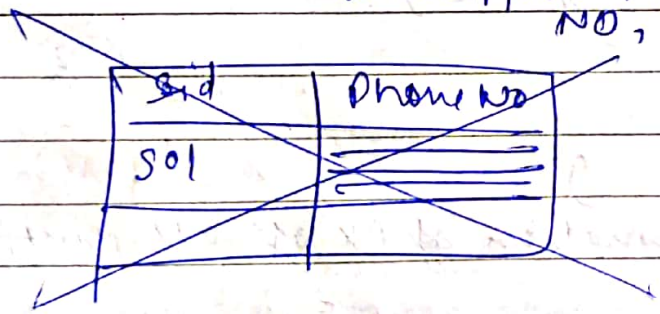


Not Possible in Relational Algebra

* Nested Table is not allowed

③ Multivalued Attr.

Student - (multivalue) multiple Phone No,



Vertical Splitting is not allowed

Not allowed Horizontal Partioning

and

Tuple should be atomic

// Can't store many values

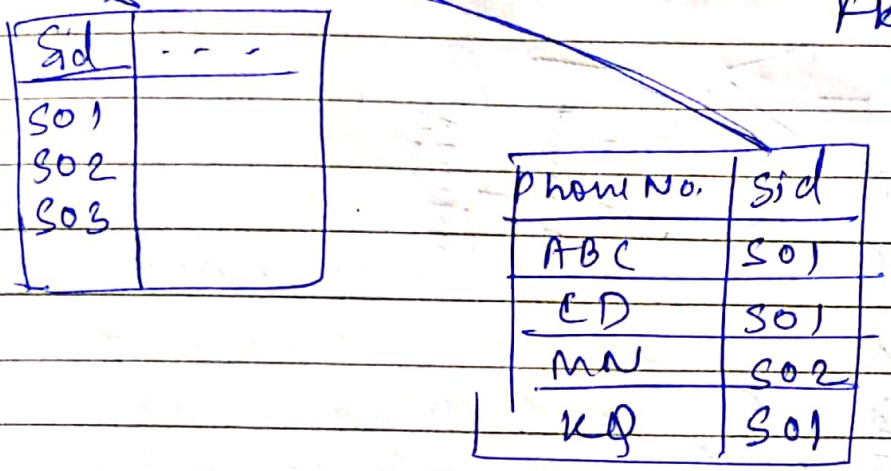
For a Multivalued attri A corresponding to an entity E we create a new relation S having the attribute A plus the PK of the relational R corresponding to the owner entity is included as FK in S.

PK

① Student (Sid ....) // Without Phone No)

② Phone No. (Phone No, Sid) // Multiple Phone No. is referred to Student table through FK of Sid

FK

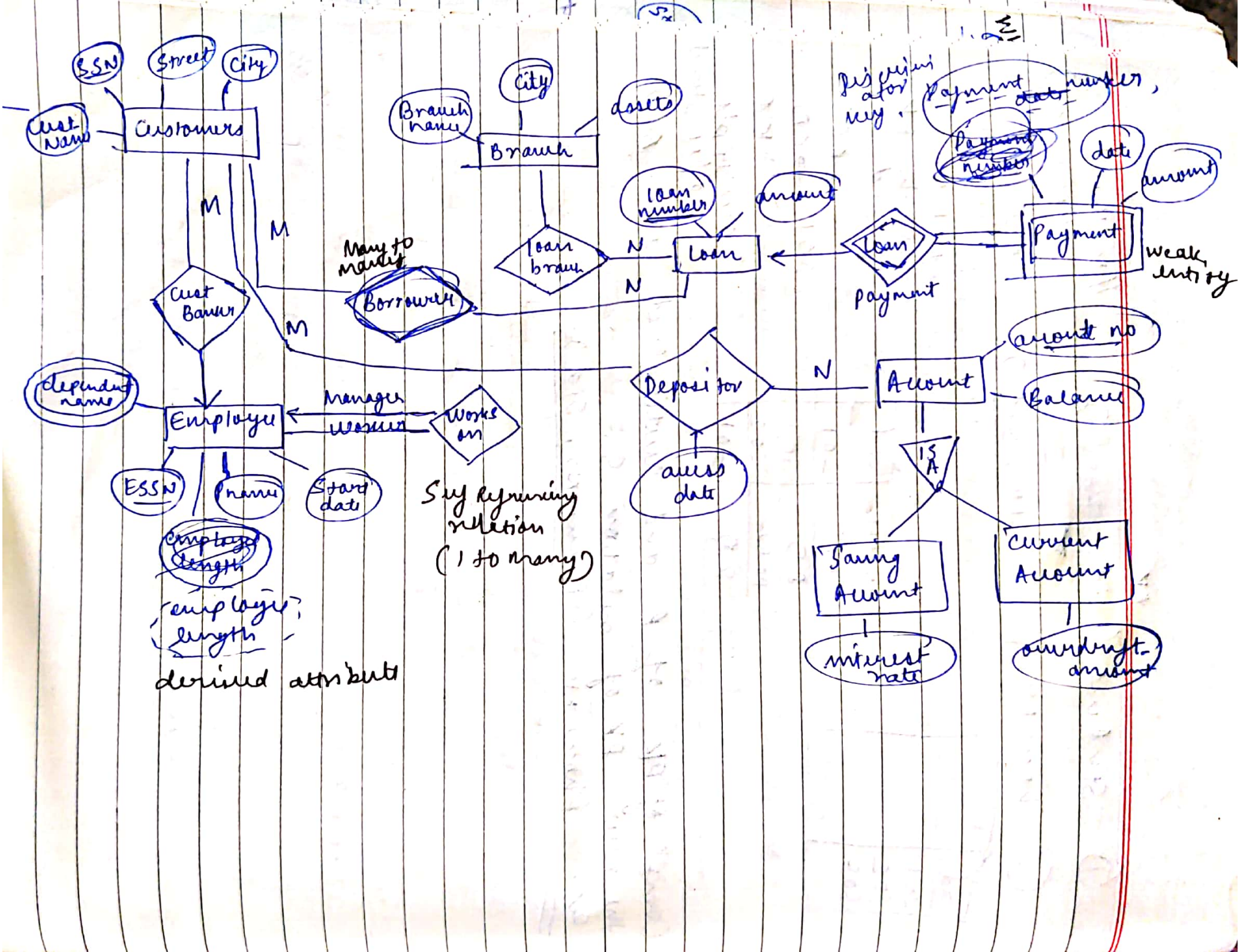


S01 → ABC, CD, KQ

Multiple value Stored

Comb^{n} of PK of R and multivalued attr. A is the new PK of relation S.

Ex. Phone No (Phone No, Sid)


SSN
Street
City
Cust-Name
Customers
M
M
Many to many
Cust Banker
Borrower
M
dependent name
Employee
Manager
Works on
ESSN
name
Start date
Employee length
derived attribute
Self Referring relation
(1 to many)
Branch name
City
assets
Branch
loan branch
loan number
amount
N
N
Loan
Loan Payment
Payment
Payment number
Payment date
number
date
amount
weak entity
Depositor
N
Account
account no
Balance
IS A
Saving Account
interest rate
Current Account
overdraft-amount
assn date

or Regular

(1) First write strong entity (Simple Attributes) FK

Customer (SSN , ~~Sp~~ Cust-name , street, city), ESSN)

Employee (ESSN , ename , Start-date , employee-length),

MgrSSN) ~~MGR~~

Branch (branch name , city , assets)

Loan (loan-number , amount), branch-name)

FK

FK1 FK2

Borrower (SSN, Loan-number)

FK1

Depositor (SSN, acount-number), accessdate)

~~Payment (payment number~~

(2) Write Weak Entities

FK

Payment (payment-no , amount , date , Loan number)

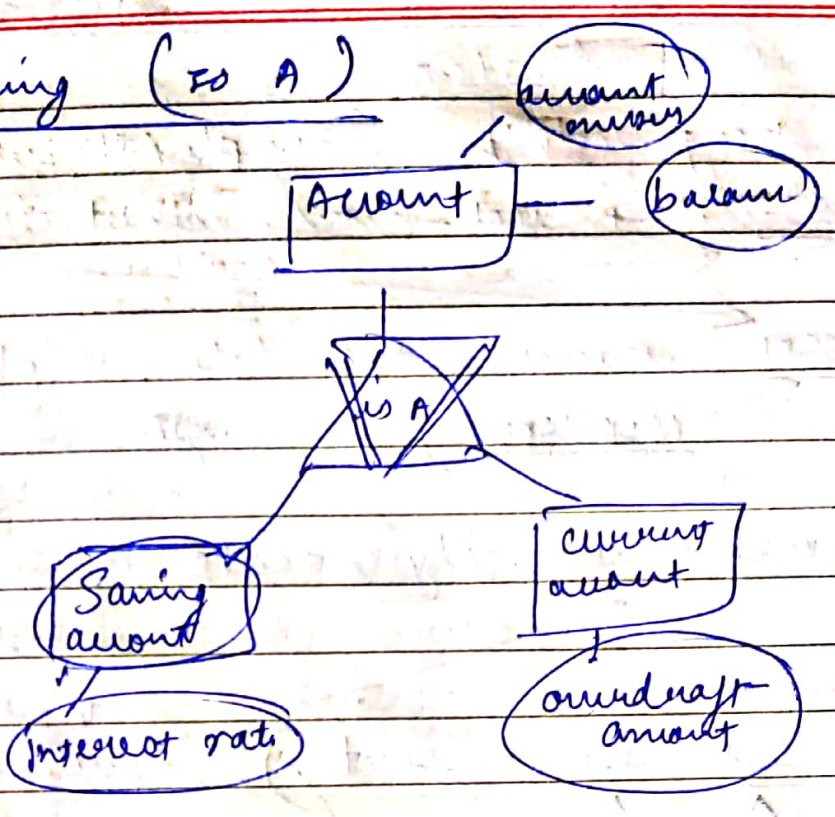
(3) Add Relation update (if required) entities,

(i) In 1 to many Relation take PK of ~~One~~ 1 side to many side as FK

(ii) 1 TO 1 No change

(iii) many to many create new relation PK of both side referring respective relation

(4) Handling (IS A)



(G) Total Disjoint Specialisation.

(i) If superclass does not participate in any relation create subclass + superclass attributes in one
Saving account (account-number, balance, interest-rate)
Current account (account-Number, balance, overdraft amount)

(ii) But here Account is a part of relation in Ques.
Method 1. ∴ superclass
[create subclass with only PK of SuperClass & make FK in both.

Account (account number, balance)
Saving account (account Number, interest rate)
current account (account Number, overdraft amount)

Method 2

Make entities as in part (5) but ~~change~~ make relations from Dependent to both the SA & CA

Reason: Subclass inherits all the relationships of Superclass

(5) Handling Multivalued Attribute :

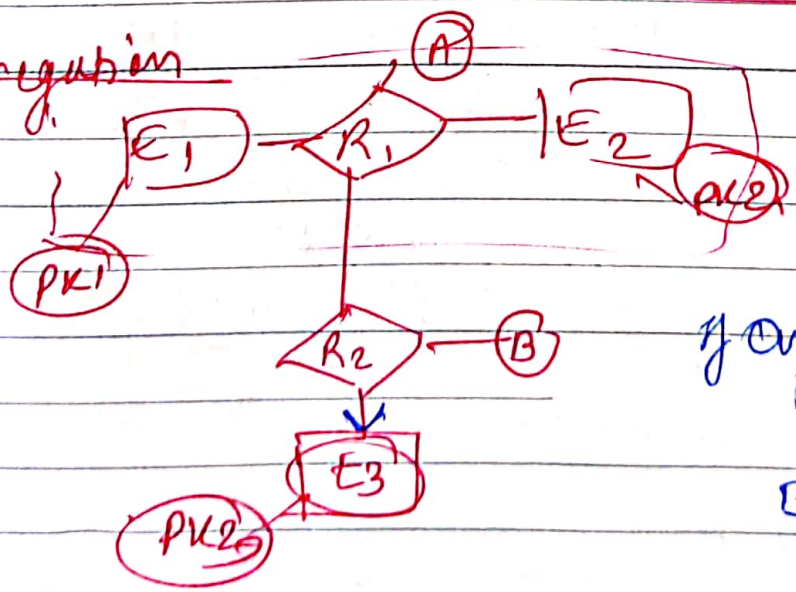
Dependents (dependent_name, ESSN)
PK

cust (SSN - - - -)

NOTE: FK can take NULL values if it is not part of PK of Entity

If it is part of PK then it can't take NULL values (due to Entity Integrity constraints)

Aggregation



R1 (PK1, PK2, A)

R2 (PK1, PK2, PK3, B)

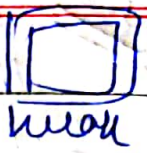
E3 (PK3, - -)

If One to many

R1 (PK1, PK2, A)

E3 (PK1, PK2, PK3, B)

Need ER Exercise

① (a) Non weak Entities →  

(b) Bank branch Partial key = Branch No, (- - -) (double line)
Identifying relationship = Branches =

(c)

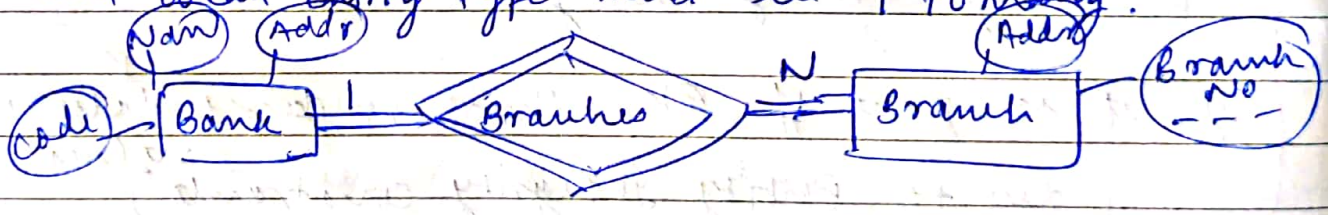
Q) What does the Partial key & the identifying relationship of weak entity type specify in this diagram?

Sol: Constraint of the Partial key: We need to combine Partial with Primary key of owner entity type key to uniquely identify weak Entity

Constraint of the identifying relationship:-

(i) Weak Entity ~~type~~ set must have total participation in the identifying set.

(ii) The identifying relationship set in strong Entity type and weak Entity type must be 1 to many.



08.09.2022 Thursday

SQL

Basic SQL Query

Select <attribute list>
From <relation list>
Where <condition>

Q) Retrieve the Bdate and address along with the name of employee with DNO = 5

SOL: ③ select Bdate, Address, Fname, Minit, Lname
① From Employee
② where DNO = 5

① → order of evaluation

$\pi_{Bdate, Address, Fname, Minit, Lname}(\sigma_{DNO=5}(Employee))$

Q② ——→ employee & its department name of employee

SOL: Slect Bdate, Address, Fnam, Minit, Lname, DName (Dnumber)
From Employee, Department
where Employee.Dno = Department.Dnumber

$\pi_{Bdate, Add^{n}, Fnam, Minit, Lname, Dname}(\sigma_{Employee.Dno = Department.Dnumber}(Employee \times Department))$

Q.③) Retrieve Bdate & address along with the name of employee in the company.

Sol: Select Bdate, address, Fname, Lname, minit
From Employee

Unspecified "where" clause where can be left if not required.

Q.④) Display the complete details of each employees in the company.

Sol: Select *
From Company Employee

NOTE
Select & From will always be there

Q.⑤) write a query, retrieve each Fname, Lname, minit along with its supervisors name.

Sol- Creating alias name of a table we use AS Clause (also can be changed attribute name)

Select E.Fname, E.Lname, S.Fname, S.Lname
From Employee AS E, Employee AS S Validation
where E.SuperSSN = S.SSN

$$\Pi_{E.Fname, E.Lname}\left(\sigma_{E.SuperSSN = S.SSN}(E \times S)\right)$$

Q.⑥) Show incrementing salary of each employee if it is incremented by 10%.

Sol: NewSalary = Salary + 10% Salary
= Salary + $\frac{10}{100}$ Salary = 1.10 * Salary

Does not change in original table only visible when runs this query

```
Select 1.10 * Salary AS Inc_Sal, SSN
From Employee
```

Renaming attributes with AS

SQL vs Relational Algebra

SQL → Multiset Result (Same value can be there)

Relational Algebra → can't have multiple tuples with same value

Reason: Because deletion & checking operation in database is costly (expensive cost)

∴ distinct tuples will be there

DISTINCT and ALL

Q(7) Display all the distinct salary value in the company

Sol:
```
Select DISTINCT Salary
From Employee
```

NOTE: Distinct is always used in Select.

Q(8) Display all the salary of all the employees in the company

Sol:
```
Select Salary
From Employee
```
or
```
Select ALL salary
From Employee
```
↳ same operation in Join

Set operations

Union, Intersection, Set Difference

UNION INTERSECT MINUS

↓ this is not in MySQL

(9)

Q) Find the list of Project No. who involve an employee whose Lname is Smith either as a worker or manager of the dept that controls that project.

Sol:
```
Select W.PNo                          // cartesian product
From Works_on AS W, Employee AS E
where E.Lname = 'Smith' and W.ESSN = E.SSN
        UNION
Select P.Pnumber
From Project AS P, Department AS D, Employee AS E
where P.Dnum = D.Dnumber and D.MgrSSN = E.SSN
      and E.Lname = 'Smith'
```

PNO
P1
P1
P2
P3
P4
P4

∪

Pnumber
P1
P2
P3
P3
P4

Project Pnumber DNumber

↓

PNO
P1
P2
P3
P4

→ Name of resulting table = Name of first table to be unioned

Remove duplicate tuples

(10) Q) Find name of employees who work on both projects P1 & P2

(11) Q) ——————— either P1 or P2

(12) Q) ——————— work on P1 but not on P2

Sol (10)

M
```
Select E.Fname, E.Lname, E.Minit
from Employee AS E, Works-on AS W
where W.PNO = 'P1' and E.SSN = W.ESSN
```

~~UNION~~ INTERSECT

N
```
Select E.Fname, E.Lname, E.Minit
from Employee AS E, Works-on AS W
where W.PNO = 'P2' and E.SSN = W.ESSN
```

(11)

M

~~UNION~~

~~INTERSECT~~

N

(12)

M

MINUS

N

// We use "contains" Clause in some Query language where MINUS is not there.

Way-2

(13)

```
Select E.Fnam, E.Lname, E.Minit
from Employee AS E, Works-on AS W
where E.SSN = W.ESSN and W.PNO = 'P1' or W.PNO = 'P2'
```

// Sometimes ~~OR~~ set intersection can be replaced by OR

13.09.2022
Tuesday.

Check How to search " in string

STRING COMPASION (char check)

LIKE used in "where" clause (where we have character domain)

wild card characters (%), (_) → replacing single character
(replace → by) 0 or more character

Q13) Select employee whose Fname starts with A

Sol:
```
select Fname
from Employee
where address LIKE 'A_ _ _ _ _ _ _ _ _ _';
```
// 10 character long string start with 'A'

Q14) Select employee whose lives in TEXAS

Sol:
```
select fname
from Employee
where address LIKE '%TEXAS%';
```
% → There can be some other / Then can be some other string

If we want to JOSEPH'S

Then use '' to use it

'JOSEPH''S'

If we want TEXAS is a sport and underscore

any character which is followed by this should be considered as baddy part

'%TEXAS_%' ESCAPE '\'

we can use other char also

NOT LIKE

just opp. of LIKE

finding string which are not having

Change in From Clause:

Q.(5) Display the new salaries of employee if they are given 10% ~~bida~~ increment in salary.

Sol: Salary' = Salary + 10% of Salary

= 110% of Salary

or 1.1 * Salary

```
Select 1.1 * Salary AS INC_SAL, SSN
From Employee
```

NOTE: Only display incremented salary but not change this in original schema.

Order By

(4) select
(1) From
(2) Where
(3) Order By // order the tuples on some attribute of tuples.
↳ Order

Q(16) Display the Employees in order of their Salaries.

Sol:

```
Select Fname, Minit, Lname, SSN
From Employee

Order By Salary DESC
```

Order By
- ASC — Ascending order
- DESC — Descending order

Q(17) If salary is equal then order the employees on basis of Fname.

Sol:

```
Select Fname, Minit, Lname, SSN
From Employee

Order By Salary DESC, Fname ASC;
```

Salary DESC — first ordering by this; if we have multiple equals

then

Fname ASC — sorting alphabetically

JOIN OPERATION

Q(18) Display SSN of employee and SSN of manager of the department for which an employee belongs.

Sol:
```
Select SSN, MgrSSN
from Employee, Department
where DNo. = Dnumber;
```

Employee × Department Cartesian Product

After applying cond^n

↓

It becomes JOIN

In SQL we have 5 types of "Join" operation.

JOIN // Inner Join
NATURAL JOIN
LEFT OUTER JOIN
RIGHT OUTER JOIN
FULL OUTER JOIN

check whether true or not

```
Select SSN, MgrSSN
From Employee JOIN Department ON (DNo. = Dnumber);
```

Q(19) Display name of the department along with all its locations.

Sol:
```
Select SSN, MgrSSN : DName, DLocation
from Department NATURAL JOIN Dept_Location
```

// Same attribute name: Dnumber = Dnumber

Outer Join is a special case of Natural Join ∴, NO need to Spify equal cond^n

discard NULL values

AGGREGATE FUNCTIONS

SUM, AVG, MAX, MIN, COUNT

Q.20) Display the average, max & minimum salary of the company.

Sol:-
```
Select avg(salary), max(salary), min(salary)
From Employee
```

avg(Salary)	max(Salary)	min(Salary)
xx	xx	xx

To give name use 'AS'

```
Selet avg(salary) AS AVG-SAL, max(salary)
AS MAX-SAL, min(salary) AS MIN-SAL
From Employee ;
```

AVG-SAL	MAX-SAL	MIN-SAL
xx	xx	xx

(20) (21)

Eg: (21) count all the employees in company.

Sol:
```
Select count(*)
From employees
```
→ counting all the tuples in the relation

count (salary)
↳ count total no. of salary values including duplication, discarding NULL values.

count (DISTINCT Salary)
↳ it will only count distinct no. of salary values.

Grouping Functions

Group By – grouping fn of Relational Algebra.

Q: (22) Display the avg, max & min salary of each dept. in the company.

Sol: NOTE: It is compulsory to mention the grouping attribute in the select clause along with the aggregate functions.

```
Select DNo, avg(salary), max(salary), min(salary)
From Employee
Group By Dno;
```

Q (23) ——— a ——— of female employees of each dept in the company.

Sol: (4) Select DNo, avg(salary), max(salary), min(salary)

(1) From Employee
(2) where Sex = 'F'
(3) Group by DNo;

Reason for NOTE:

It will generate max, min & avg values but will get confused on how to plan.

NOTE:

Group By can be only in conjuction with aggregate function

Can't use Group By alone.

Q. (28) Display no. of male & female employees each Sex.

Sol: Select DNo, avg(salary), min(salary), max(salary)
From Employee

Group By DNo, Sex;

DNo.	Sex	avg	max	min
1	M			
1	F			
2				
2				

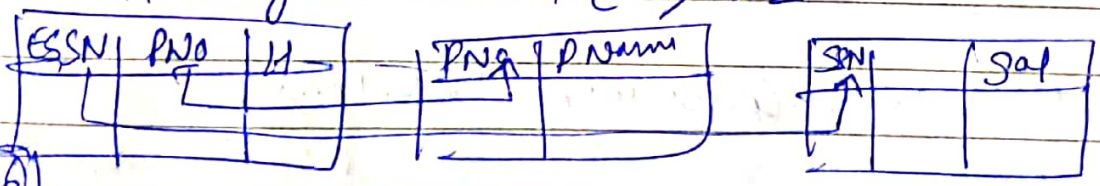
14.09.2022 Wednesday

"Having" Clause.

(5) Select
(1) From
(2) Where
(3) GroupBy ⇒ GroupBy can only be used when we
(4) Having → only can be used when Aggregate functions
there is GroupBy
↑ Order [To apply f^n on Groups individually

Q(25) Find the departments for each project having more than 2 employees working on it. Display Project Name, No. of employees working on it and the avg. salary of employees working on it

Sol: Select P.Pname, count(*), avg(salary)
From Works-On AS W, Project AS P, Employee AS E
where W.PNo = P.Pnumber & W.ESSN = E.SSN
Group By P.Pname
Having count(*) > 2



Q(26) For each project retrieve PNo., Pname & no. of employees who work on that project from Dept. No 5 having count > 5

Sol: Select P.Pnumber, P.Pname, count(*), ~~avg(Salary)~~
From Works-On AS W, Project AS P, Employee AS E
Where W.PNo = P.number and W.ESSN = E.SSN
and E.DNo = 5

DNo
↓
P1 ---2
EO1 P1 8

EO1 ---5
↓
DNo

Group By P.Number, P.Pname
Having count(*) > 5;

Q(21) For each department, count total No. of employees whose salary exceed 40,000 and with more than 5 such employees.

Sol: Select DNo, count(*)
from Employee
where Salary > 40,000
Group By DNo
Having count(*) > 5

// Applying cond^n on Subgroups of Tuples.

NESTED QUERIES

Query inside Queries.

→ Not allowed in Relational Algebra but allowed in SQL

comparing attributes to a set of values:

{ Bhopal, Indore, Gwalior, Jabalpur }
↑ enumeration form

cond^n of where is given { and } LS computed first & then apply.

* Unrestricted case
Nested Query in From clause is used many less in Recent SQL's.

Nested Queries in Where Clause

Q(28) Find the name of employee whose salary is same as employee & SSN is 123456789

Sol: IN / NOT IN → Set comparison operator

```
Select E1.SSN
From Employee AS E1
where E1.Salary IN ( Select E2.Salary
                     From Employee AS E2
                     where E2.SSN = '123456789' )
```

scope of inner is within brackets

we can use = operator since we get only one

If an attribute's value belongs to the set / present in the set it will return "True" or otherwise "FALSE" (if not present).

(1) For each tuple of E1 Query E2 will be executed for each.

Inner Queries will run each time for every Outer tuple.

(2) We can write without using E1 & E2 because both have different scope.

Q.29) Find employees whose salary is greater than the employee whose SSN is '123456789'

Sol:
```
Select SSN
From Employee
where salary > ( Select SSN
                 From Employee
                 where SSN = '123456789' )
```

We can use operator since inner query gives only one value to compare rather than enumerated list → (1)

Q.30) Find name of employees who work on both Project P1 & P2

Sol: M.1. Set operator INTERSECT

M.2. Set comparision

```
Select E.SSN, E.Fname, E.Lname
From Employee AS E, Works_On AS W
where E.SSN = W1.ESSN and W1.PNo = 'P1' and
      W1.ESSN IN
      ( Select W2.ESSN
        From Work_On AS W2
        where W2.PNo = 'P2' )
```

Q(5) Find employees who work on P1 but not on P2.

Sol: M-1: MINUS Method Set Operator

M-2: Use NOT IN ∅ instead of IN

M-3: Double Nesting

```
Select E.Fname, E.Lname, E.SSN
From Employee AS E
Where E.SSN IN (Selut W1.ESSN From Work-On AS W1
                Where W1.Pno = 'P1' and
                      W1.ESSN
                NOT IN
                (Selut W2.ESSN
                From Works-On AS W2
                Where W2.PNO = 'P2' ));
```

Both give the result same.

Q(3) <comparison operator> ANY

<comp op> SOME

<comp op> ALL

=, >, <, ≥, ≤

<, >

(ALL =) is not valid operation

A: <comp op> =, >, <, ≥, ≤ used in where clause

Q(32) Find employee whose salary is greater than some employee of department No. = 5

Sol:
```
Select Fname, Lname, SSN
From Employee
where Salary > SOME (Select salary
                     From Employee
                     where DNo = 5);
```
↓
Also we can ANY

Q.(33) Find the employees who are have the ~~max~~ highest salary.

Sol:
```
select fname, SSN
from employee
where Salary = (Select max(salary)
                from employee );
```

Q.(34) Find the employee who earns 2nd largest salary in the company.

Sol: Write it in ~~copy~~ From Clause
```
select max(salary) As 2nd_max_sal
from Employee
where salary Not in (select max(salary)
                     from employee);
```

15.09.2022
Thursday

Correlated Nested Queries

every time result of inner query changes according

(35) Q) Find the name of employees who work on all projects controlled by dept NO = 5

EXISTS
NOT EXISTS

```
select
from
where EXISTS (subquery)
```

returns TRUE if the result of subquery is not an empty set
Otherwise it returns FALSE

NOT EXISTS → returns TRUE if the result of inner query is an empty set
Spau
Otherwise FALSE

In Relational Algebra, it can be solved by ÷ operator

(16) ÷ operator can be solved this

```
select Fname, Lname, addr
from Employee
where NOT EXISTS ( select Pnumber
                   from Project
                   where Dnum = 5
                   MINUS
                   select Pno
                   from Works_on
                   where SSN = ESSN );
```

E01
E02
E03
E04

change ↓ E02

DNO=5

Pnumber
P1
P2
P6
P1

−

Pno
P1
P2
P6
P8
P10

NOTE: ① Outer query attributes will have scope on inner query.

② Inner Query attributes will

Q 36) Find employees who have at least one dependent.

Sol: M-1 Can be done by Join operation

M-2 Select Fname, Lname, SSN
From Employee
where EXISTS (Select *
From Dependent
where SSN = ESSN)

	SSN	dependent
E01 →	E01	
	E02	
	E02	
	E04	
	E04	
	E04	

UNIQUE → returns true if query returns unique value
NOT UNIQUE

37) Q) Find employees who work on two different projects.

Sol: Select Fname, Lname
from Employee
where NOT UNIQUE (Select ESSN
from Work-on
where SSN = ESSN)

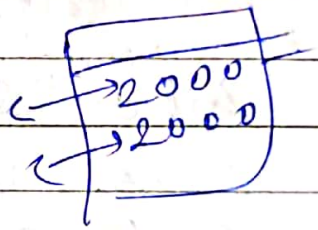
Q(38) Find Projects which involve one or more employees on it.

Sol: Same as (37)

Q.(39) Select the employees who have the highest salary.

Sol: M.1. previous page

M.2: Greater than all




```
Select
from Employee AS E1
where salary > ALL ( select salary
                     from Employee AS E2
                     where E1.salary != E2.salary );
```

LIMIT Clause: Limit the no. of rows in the clause. To use it efficiently use order by

(It is

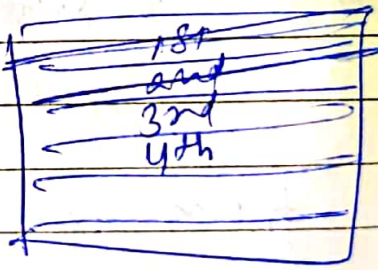
```
Select column-list
From table
Order By column list
LIMIT row-count OFFSET offset
```

Q (59) Display the Top 5 salaries of the company

Sol- Sulit Fname, Lname, Salary
From Employees
Order By Salary Desc
LIMIT 5

Q (60) Find the 3rd highest salary in the company

Sol: Select Salary, Fname, Lname
From Employee
Order By salary DESC
LIMIT 1 OFFSET 2;



Select max(Salary)
from
where
LIMIT

From Nested Query

(11) 8) Find the departments for which the avg. salary of employee is min. over all the department

Sol: * Most SQL does not allow from Nested Query.

```
Select DNO
From (Select DNO, avg(salary)
      From Employee
      Group By DNO)
AS DEPT-AVGSAL (Dno, AVG-SAL)
Where AVG-SAL = (Select min (AVG-SAL)
                 From Dept-Avgsal);
```

Queries

Q(1) Find the employees who have same first name as their dependents first name.

Sol:
```
Select Fname, Minit, Lname
From Employees, Dependent
where SSN = ESSN and Fname = dependentname;
```

Q(2) Find name of employees who work in DNO. 5 & who work for more than 10 hrs a week on a project called Project X.

Sol:

Sol: Select Fname, Minit, Lname
From Employee, Project, Work-on
where SSN = ESSN and Pnumber = Pno
and DNo = 5 and Pname = 'Project X'
and hours ≥ 10;

Select Fn ——————
From (Employee Join Work-on ON (SSN=ESSN)
JOIN Project ON (Pno = Pnumber))
where DNo = 5 and ——————

Q.3) Name of all employees who are directly supervisor Franklin T Wong.

Sol: (Join Employee with Employee such that SSN = MgrSSN)

Sol

Select E.Fname, E.Minit, E.Lname
From Employee AS E, Employee AS S
where E.SuperSSN = S.SSN and
S.Fname = 'Franklin' and
S.Minit = 'J' and S.Lname = 'Wong';

Supervisor

E

E01	E05

S

E05	Franklin

Q.4) For each project list project name & total no. of hours ~~for the~~ per week spent on that project by all employees who are working on that project.

Sol:
```
Select Pname, sum(hours)
From Works_on JOIN Project ON (PNo. = Pnumber)
GroupBy Pname;
```

P1	E1	5
P2	E2	4
P1	E7	3

→ Group

sum (5+6+3) = 14

Q.5) Find the name of all employees who work on every project.

Sol: ~~List~~ NOT EXISTS MINUS

```
Select Fname, Lname, Minit
From Employee
Where NOT EXISTS ( Select Pnumber
                   From Project
                   MINUS
                   Select Pno
                   From Works_on
                   Where SSN = ESSN );
```

Q.6) ——— who do not work any project

M.1:

Sol:- Select SSN
From Employee
MINUS
Select ESSN
From Works_ON ;

M.2 NOT IN Clause

Select SSN, Fname, Lname
From Employee
where SSN NOT IN (select ESSN
From Work_ON);
if not present
then true

For each department

Q.7] For each department retrieve the department name & avg. salary of all female employee working in department.

Sol: Select Dname, avg(Salary)
From Employee JOIN Department ON (Dno=Dnumber)
Where Sex = 'F'
Group By Dname.

Q.8] Find the last name of all department managers who have no ~~names~~ dependents

Sol: M.1. Select E.Lname
From Employee AS E

where E.SSN IN (select D.MgrSSN
from Department AS D
where D.MgrSSN NOT IN
(select ESSN
from Dependents));

(ESSN not in department mangers but not in Dependents)

ESSN : not in ~~department~~ Dependent ~~but~~ . but department in

M:2 select E.Lname
from Employee AS E ~~IS~~ JOIN Department AS D
ON (ESSN = SSN).
where D.MgrSS

Q.9) Find name & address of all employees who work on atleast one project located in Houston ~~but~~ but who's department has no location in Houston.

Sol: select ESSN
Q1 { from Works-On, Project
where PNo = Pnumber and Plocation = 'Houston'

INTERSECT

Q2 { select SSN
from Employee
where DNO IN (select Dnumber
from Department
MINUS

Select Dnumber
From Dept_Locations
Where Dlocation = 'Houston';

M-2.

Q2
Select SSN
From Employee
Where Dno NOT IN
(Select Dnumber
From Dept_Locations
Where Dlocation = 'Houston');

Q) 10 Find emp. who work on atleast 2 diff. projects

Sol M-1 Select Fname, Lname
From Employee
Where NOT UNIQUE (Select ESSN
From Works_ON
Where SSN = ESSN);

M-2.

Select Fname, Lname
From ~~Employee~~ Works_ON JOIN Employee
ON (SSN = ESSN)
Group By ESSN
Having count(*) > 2
ESSN
*

Q.11)

M.U

Find emp who — works — exactly 2 diff projects

distinct

Select W1. ESSN

From Works-On AS W1, Works-On AS W2

Where W1. ESSN = W2. ESSN

and W1. Pno ! = W2 Pno.

atleast 2

Select E

——»

——»

——

count(*) = 2 ;

21.09.2022 Wednesday

MINUS

Q) Find the employee who work on 'P1' and not on 'P2'

Sol:

```
Select
From Employee
where SSN IN (Select ESSN
              From Work_On
              where Pno = 'P1'
              and ESSN NOT IN
                  (Select ESSN
                   From Work_On
                   where Pno = 'P2'));
```

INTERSECT

Q) Find employees who work on both P1 & P2

```
IN
```

Data Base Design

(int)
1. Domain constrain
2. Table level constraints (DoB < DoJ)
3. Key constraints (PK, FK, SK, CK)

→ how data is arranged
Normalization
Testing your database Design
Use of keys

Functional Dependency (FD) → defines on the schema & Relationship / Association b/w 2 subsets of not on instance attributes of a Relation R

$X, Y \subseteq R$

X functionally determines Y
or
Y is functionally dependent on X

$X \longrightarrow Y$

If a FD $X \rightarrow Y$ holds on R, then for any 2 tuples t_1 & t_2 in $r(R)$

(small r → instance of Relation R
Capital R → Relation.)

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

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_2	c_3
a_3	b_3	c_1
a_3	b_3	c_2
a_4	b_1	c_1

$A \rightarrow B$

but $B \rightarrow A$ does not holds

$t_1[a_1] = t_2[a_1]$
$\Rightarrow t_1[b_1] = t_2[b_1]$
Same with a_3

It can be part of ⓐ property

Remarks

① FD is a property of relational schema and not of the instance.

② All Relational instances of R(R) that satisfy all the FDs defined over R are called valid/legal states of a Relation.

③ $X \to Y$ does not imply $Y \to X$

④ $XY = Z \not\Rightarrow X \to Z$ or $Y \to Z$

↳ for any 2 tuples t_1 and t_2

if $t_1[X] = t_2[X]$ and $t_1[Y] = t_2[Y]$

$\Rightarrow t_1[Z] = t_2[Z]$

⑤ $X \to Y$ and $X \to Z \Rightarrow X \to YZ$

⑥ $R(A, B, C, \ldots, \underline{K})$

then $K \to A, B, \ldots$

or $K \to R$

Primary Key can uniquely determine Relation (FD)

Trivial FD

Non Trivial FDs

$X = AB$

$AB \to A$ and $AB \to B$

$X \to Y$ where $Y \subseteq X$

R, (F) → initial set of FDs created by Database designers

We can also infer additional FD(s) from set F that hold on R using a set of rules.

ARMSTRONG'S AXIOMS

F^+ → Closure of FDs (original + all the FD's that can be derived)

(*) Inference Rules / Armstrong's Axioms :-

IR1 Reflexive Rule — If $Y \subseteq X$ then $X \to Y$

IR2 Augmentation Rule (adds) — If $X \to Y$ then $XZ \to YZ$

IR3 Transitive Rule — If $X \to Y$ and $Y \to Z$ then $X \to Z$

IR4 Decomposition Rule — If $X \to YZ$ then $X \to Y$ and $X \to Z$

IR5 Union Rule — If $X \to Y$ and $X \to Z$ then $X \to YZ$

IR6 Pseudo transitive Rule — If $WX \to Y$ and $Z \to W$ then $ZX \to Y$

④ & ⑤ are just opposite each other

2 on left side

① $WX \to Y$
$Z \to W$
$ZX \to Y$

① ② ③ are basic rules and other rules can be derived.

Set of FDs derived rules IR1 through IR3 are sound and complete.
↳ All the FDs can be derived.

R, F

r(R) satisfies F — # Instance

then all the FDs that are derived using the above rules will also hold on r(R)

22.02.2024 Thursday

Q) $R(A, B, C, D, E, F)$
$A \to BC$ $\quad CD \to E$
$B \to EF$

(Write Rule in bracket which you use)

Sol: $A \to BC$ → (I)

$\Rightarrow A \to B$, $A \to C$ [∵ Decomposition Rule] ← Union

$B \to EF$ → (II)

$\Rightarrow B \to E$, $B \to F$ [∵ Decomposition Rule]

$A \to B$ (From (I)) , $B \to E$ (From (II))

$\Rightarrow A \to E$ [∵ Transitive Rule]

$A \to B$ [From (I)] , $B \to F$ [From (II)]

$\Rightarrow A \to F$ [∵ Transitive Rule]

$CD \to E$ and $A \to C$ [From (I)]

$\Rightarrow AD \to E$ [∵ Pseudo transitive Rule]

Apply when left hand side is complex

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

Closure of an Attribute

$R(A, B, C, D, E, F)$
$A \to BC$
$E \to CF$
$B \to E$
$CD \to EF$

Find AD^+

$X^+ = \{X\}$ Reflexive Rule $X \to X$

Then if $X \to Z$ that holds on R & $Y \to Z$ holds on R

$X^+ = X^+ \cup Z$

$AD^+ = \{A, D\} \cup \{B, C\}$ [$\because A \to BC$]

$= \{A, D, B, C\} \cup \{E\}$ [$\because B \to E$]

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

$\to \cup \{F\}$ [$\because E \to CF$]

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

$\to \cup \{F\}$ [$\because CD \to EF$]

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

If left hand part is completely present then include Right hand side completely or partially in 'Closure'.

AD is the key of relation R because $K \to R$

$AD \to A\ B\ C\ D\ E\ F$

[Point ⑥ of Remarks]

Q) R (A, B, C, D, G, H, I)

$A \to B$

$A \to C$

$CG \to H$

$CG \to I$

$B \to H$

D is part of relation but not in dependency
∴ D is independent

↓
Not Related to any other attribute but they are related to themselves

NOTICE
A is always on LHS
∴ A is ~~always~~ independent

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

$AG^+ = \{A, G, B, C, H, I\}$ ← G

$D = \{D\}$

$ADG^+ = \{A, D, G, B, C, H, I\}$

↳ Candidate key "Minimal property"

$ABDG^+ = \{A, D, G, B, C, H, I\}$

↳ Superkey but not Candidate key.

#Algorithm for finding Candidate key.

Q) R (A, B, C, D, G, H, I)

$A \to BC$

$CG \to HI$

$B \to H$

Sol: $K = R = \{A, B, C, D, G, H, I\}$

(1) Consider whole relation as initial as candidate key

(2) Now try to remove elements & check.

$(K - A)^+ = \{B, C, D, G, H, I\}^+ = \{B, C, D, G, H, I\}$

∴ A cannot be removed from key K

Because closure of remaining set of attributes does not contain A.

$(K - B)^+ = \{A, C, D, G, H, I\}^+ = \{A, C, D, G, H, I\} \cup \{B\}$ [$A \to BC$]

$= \{A, B, C, D, G, H, I\}$

∴ B is a redundant attribute and can be removed from key K.

$(K - C)^+ = \{A, D, G, H, I\} \cup \{B, C\}$ [∵ $A \to BC$]

$K = \{A, D, G, H, I\}$

$(K - D)^+ = (A, G, H, I) \cup (B, C)$

(∵ D is not present

→ , D can't be removed.

$K = \{A, D, G, H, I\}$

$(K - G)^+ = \{A, D, G, I\} \cup \{B, C\}$

G cannot be removed [G is absent]

$K = \{A, D, G, H, I\}$

$(K - H)^+ = \{A, D, G, I\} \cup \{B, C\} \cup \{H\}$

$A \to BC \quad B \to H$

$K = \{A, D, G, I\}$

$(K-I)^+ = \{A, D, G\} \cup \{B, C\} \cup \{H, I\}$

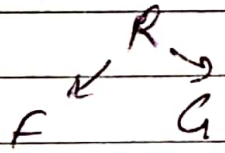
$CG \rightarrow HI$

I can be removed

$K = \{A, D, G\}$

ADG is the Candidate key.

Equivalence of Set of FDs



"Two database designers designed 2 diff. (designs) sets for relation R"

The 2 sets of FDs F and G are equivalent if F covers G and G covers F away

F covers G → if, FD of F can be inferred of FDs of set G

$R\{A, B, C, D, G, H\}$

$F = \begin{bmatrix} A \rightarrow C \\ AC \rightarrow D \\ E \rightarrow AD \\ E \rightarrow H \end{bmatrix}$ $G = \begin{bmatrix} A \rightarrow CD \\ E \rightarrow AH \end{bmatrix}$

Method:

FDs of F | FDs of G

① $A \to C$ ← $A \to CD$

$\Rightarrow$ $A \to C$, $A \to D$ [Decomposition Rule]

② $AC \to D$ ← $A \to CD$

$\Rightarrow A \to C$ & $A \to D$ [Decomposition Rule]
(I) (II)

$\Rightarrow AC \to DC$ [Augmentation Rule]

$\Rightarrow AC \to D$ & $AC \to C$ [Decomp. Rule]
[Trivial]
[$C \subseteq AC$]

③ $E \to AD$ ← $E \to AH$

$\Rightarrow E \to A$ & $E \to H$ [Decomposition Rule]
(III) (IV)

From (I) & (III) [Transitive Rule]

$E \to D$ → (IV)

From (III) & (IV):

$E \to AD$ [Union Rule]

④ $E \to H$ ← From (IV): $E \to H$

Every entries of F can be inferred from G

∴ G covers F → (A)

Now check for opposite / vice versa

FD of G

① $A \to CD$

FD of F

$A \to C \to$ (I)

$\therefore AC \to D$

$\because AA \to D$ [Pseudo transitivity Rule]

$\Rightarrow A \to D \to$ (II)

From (I) & (II)

$\boxed{A \to CD}$ [Union Rule]

② $E \to AH$

$E \to AD$

$\Rightarrow E \to A$ (I), $E \to D$ [Decomposition Rule]

$\because E \to H \to$ (III)

$\boxed{E \to AH}$ [(I), (III) Union Rule]

Every FD of G can be inferred from F

$\therefore$ F covers G $\to$ (B)

From (I) & (IV):

$\boxed{F \text{ \& } G \text{ are equivalent}}$

Method-2

$\boxed{\text{if } G^+ \equiv F^+ \text{ then } F \text{ \& } G \text{ are equivalent}}$

Union ↑ Decomposition ↖

Minimal Cover / Canonical Cover

Do not include Redundant FDs

① Standard Form

$X \to A_1, \quad X \to A_2, \ldots, X \to A_n$

$X \to A_1 A_2 \ldots A_n$

NOTE: ① In Minimal ~~for~~ Cover, we write by taking UNION

② In Cover, just opposite Decomposition

② Remove redundant attributes in FDs

↓

removing the element from LHS I can get the valid FD

(i) $A \to BC$ → (I)

$AB \to C$

↙ ↘

$(AB-B) \to C$ $(AB-A) \to C$

From (I) $B \to C$

B is redundant

(ii) $A \to BC$

$AB \to C$

$B \to C$

A & B are redundant but remove only one attribute

③ Remove redundant FDs from the set F

(i) $A \to BC$

$AB \to C$

$B \to C$

→ $AB \to C$ is redundant FDs

(ii) $A \to BC \Rightarrow A \to B$, $A \to C$

$B \to C$

Redundant FDs.

27.09.2021
Tuesday

Generally Transistive FDs form Redundant FDs.

But (i) is different

If multiple attributes on left and are dependent other then remove dependent one

$X \to Y$

$XY \to Z$

$\Rightarrow \boxed{X \to Z}$

Algorithm : Finding minimal cover G for F.

1. Set $G = F$.
2. Replace each FD $X \to A_1 A_2 A_3 \ldots A_n$ by n FDs $X \to A_1, X \to A_2, \ldots X \to A_n$
3. For each FD $X \to A$ in G (Redundant Attributes?)
   for each attribute B that is an element of X.
   { compute X^+ w.r.t. set of FDs { $F - (X \to A) \cup (X \to B) \to A$ }
   if X^+ contains A then remove B from X in FD $X \to A$ }
4. For each FD $X \to A$ remaining in G (Redundant FD)
   { compute X^+ wrt { $F - (X \to A)$ }
   if X^+ contains A, remove $X \to A$ from G }

Q) ① F = { A → BC
(1) A → B
(2) B → C
(3) AB → C
(4) AC → D }

Sol: ① (iii) AB → C ⇒ B → C
Remove A & look whether we

(4) (iii) AB → C ⇒ B → C

A → B
B → C
C → D

② R (A, B, D) {
(i) B → A
(ii) D → A
(iii) AB → D

② Step. No use CALL left single
(iii) AB → D
& B → A
∴ ~~BA~~ → D
⇒ B → D

A is an extra nous attr.

After step ③
B → A
D → A
B → D

Step ④ B → D
& D → A
∴ (B → A) Redundant ~~A1~~ FD

Finally:
B → D
D → A

③ R(A, B, C, D)
A → B
A → C
A → D
A → E
CD → E

Step ① Left single

Step ②
CD → E
& A → C
∴ AD → E
& ~~D~~ A → D
A → E

No change

Step ③
A → C
& A → D
∴ A → CD
& CD → E
∴ A → E
∴ Redundant FD

Finally
A → B
A → C
A → D
~~A → E~~
CD → E

Used in Normal forms.

Database Designs

Base Relations. Virtual View

↳ Our design is good / not on basis of Normalization.

Informal Design Guidelines

→ Semantics & Attributes.

→ Removal of Redundant Data → leads to ~~Insertion~~ updation anomalies
- ~~Emp dept~~ Emp_data
  - Name
  - DNo
  - Dname
  - again & again

↳ insertion
↳ deletion
↳ modification

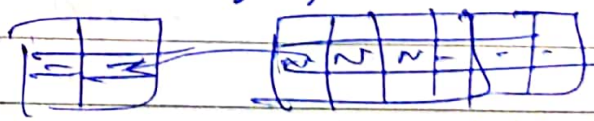
Any Dept Name changes then change in every field where it is present

If I have only one employee then removing that person will remove Dname of that Dept also

[If u miss at any tuple then it will lead to inconsistency]

To avoid this use FK (Foreign key)

→ Minimal use of NULL value



If your tuple has ~~many~~ most entry as NULL avoid entry of such tuple.

Create small separate table to look for that.

Not related

→ Avoid generation of spurious tables

Emp_Proj

SSN	Ename	PNo	Hours	Pname	Plocation
E01	John	P1	8	-	Houston
E01	John	P2	8	-	NewYork
E02	~~John~~ Peter	P3	9	-	NewYork

Emp_Loc

Ename	Plocation
John ~~E01~~	Houston
John ~~E01~~	NewYork
Peter	NewYork

Emp_Proj

SSN	Pno	Hours	Pnam	Plocation
E01	P1	8	-	Houston
E01	P2	8	-	New York
E02	P3	9	-	New York

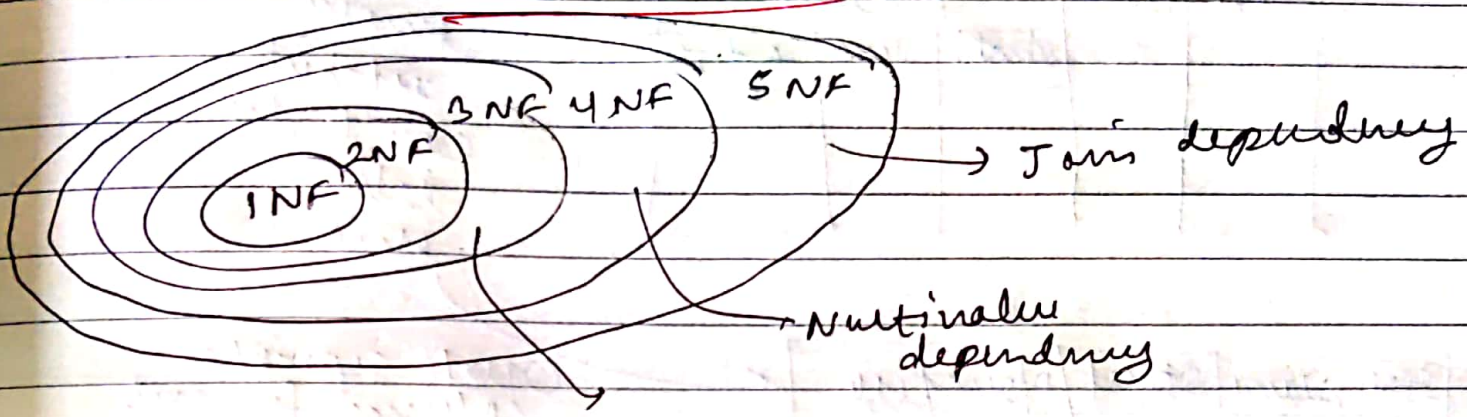
To avoid spurious ⋈ Natural Join

→ After Join we will not get this plocation

SSN	Ename	PNO	Hours	Pnam	Plocation
~~E01~~	John	P1	8		Houston
E01	John	P2	8	-	New York
✗ E02	John	P3	9	-	NewYork

To avoid this use
P.K in any table
& PK as FK in 2nd

Normalization



5NF should satisfy all the cond^n of all inner NFs

∴ Venn diagram

3NF ——— 2NF, 1NF

Normal Form Highest NF a relation is satisfying

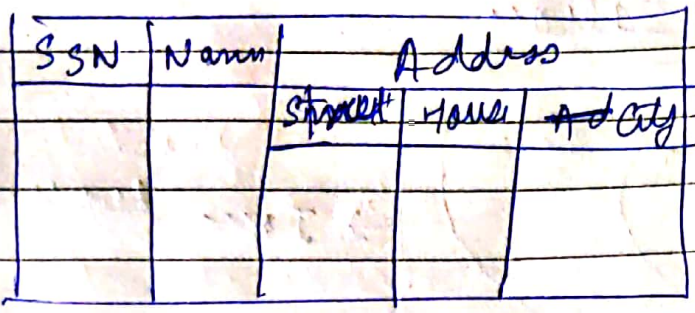
1NF : Relation should contain only Atomic values

SSN	Ename	PNo	Hours
E01	ABC	P1	5
		P2	3

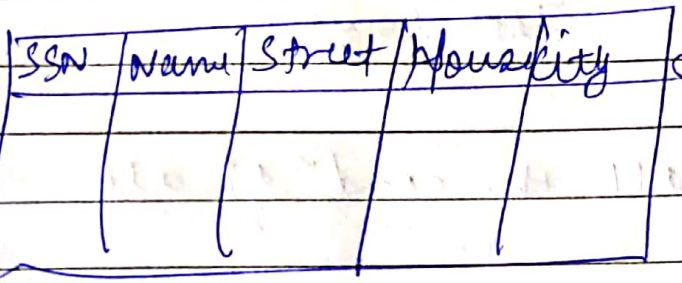
→ A tuple can't have multiple values ~~at~~.

↙ create as many tuples as values

SSN	Ename	PNo	Hours
E01	ABC	P1	5
E01	ABC	P2	3



Composite attribute can't be like this



create separate column for each ∴ Atomicity

Our Relation is in By dyault in 1NF

$R(A, B, C, D)$ { $A \to B$, $B \to CD$ }

Already in 1NF

Start chec