DBMS (4th Sem)

DATE: __/__/__
PAGE:

Functional dependency

↳ Imp for normalization

- Conversion of E-R diagram into relational model
- Basics of relational model and functional dependency.
- Idea about keys / types

↳ For Normalization

SQL

R.A

Disadvantages of file system

① Data Redundancy:-

↳ एउटा data फरक फरक ठाउँ मा छ scattered छ।

② Data incosistency:-

↳ एउटा data फरक फरक ठाउँमा छ र value पनि ठाउँ ठाउँमा different छ।

③ Difficult in accessing data:-

④ Data Isolation

⑤ Security Problem

⑥ Atomicity problem

⑦ Concurrent - access anomalies

⑧ Integrity Problem

↳ An unintended changes to data as the result of a storage, retrieval or processing operation, including malicious

Good Write

intent, unexpected hardware failure, and human error, is failure of integrity.
The changes are the result of unauthorized access, it may also be a failure of data security.

ER Diagram:

- Intro and basics
- Entity and Entity set
- Attributes and types
- Relationship
- Strong and weak entity set
- Traps.
- Conversion.

ER ⇒ Entity Relation

DATE: __/__/__
PAGE: _____

ER

- ER Model defines the conceptual (or logical) view of a problem.
- Used for designing database.
- Works around the real word entities and relationships among them.

Why ER model are useful:-

- The construct used in the ER model can be easily transformed into relational table.
- ER model can be used by database designer to communicate the database to the user.
- It can be used as a design plan by the database developer to implement a data model in specific DBMS software.

* Entities
* Attributes
* Relationships

entity
attribute
relⁿ

DATE: __/__/____
PAGE: _____

Attributes:

↳ for each attributes there are some set of permitted values, called domain (or Range) of that attribute.

↳ Example:- a student name cannot be a numeric value. It has to be alphabetic.
A student's age can't be negative, etc.

Attributes
- Simple
- Composite

Simple and Composite attribute
Single-valued and multi-valued → attributes
Stored and Derived

Relationship:-

↳ Association among entities.
Ex:- an employee [works-at] a department
a student [enrolls]
relⁿ.

DATE: __/__/____
PAGE: _____

Relationship set :-

→ set of relationships of similar type is called relationship set.

Like entities, relationship ~~sets~~ too can have attributes. Those attributes are called descriptive attributes.

→ Descriptive Attribute

Degree of reln

→ The no. of different entity sets participating in a relationship set is called as Degree of reln.

- Unirary → 1 entity set participating
- Binary → 2 entity sets
- Ternary → 3 entity sets
- ⋮
- Nnary → N no. of sets

Imp Attributes in ER Model

DATE:
PAGE:

Attribute ⇒ Properties of an entity.

Types of Attributes

1) Simple and composite
2) Single-valued and Multi-value
3) Stored and derived
4) Key Attribute

1) Simple Attribute :-

→ Are atomic values, which cannot be divided further.

Ex:-

⇒ A student's mobile no. is an atomic value of 10 digits.

⇒ Birth date is an atomic value of date-month-year.

DATE: __/__/____
PAGE: _____

2) ~~Compound~~ Composite :

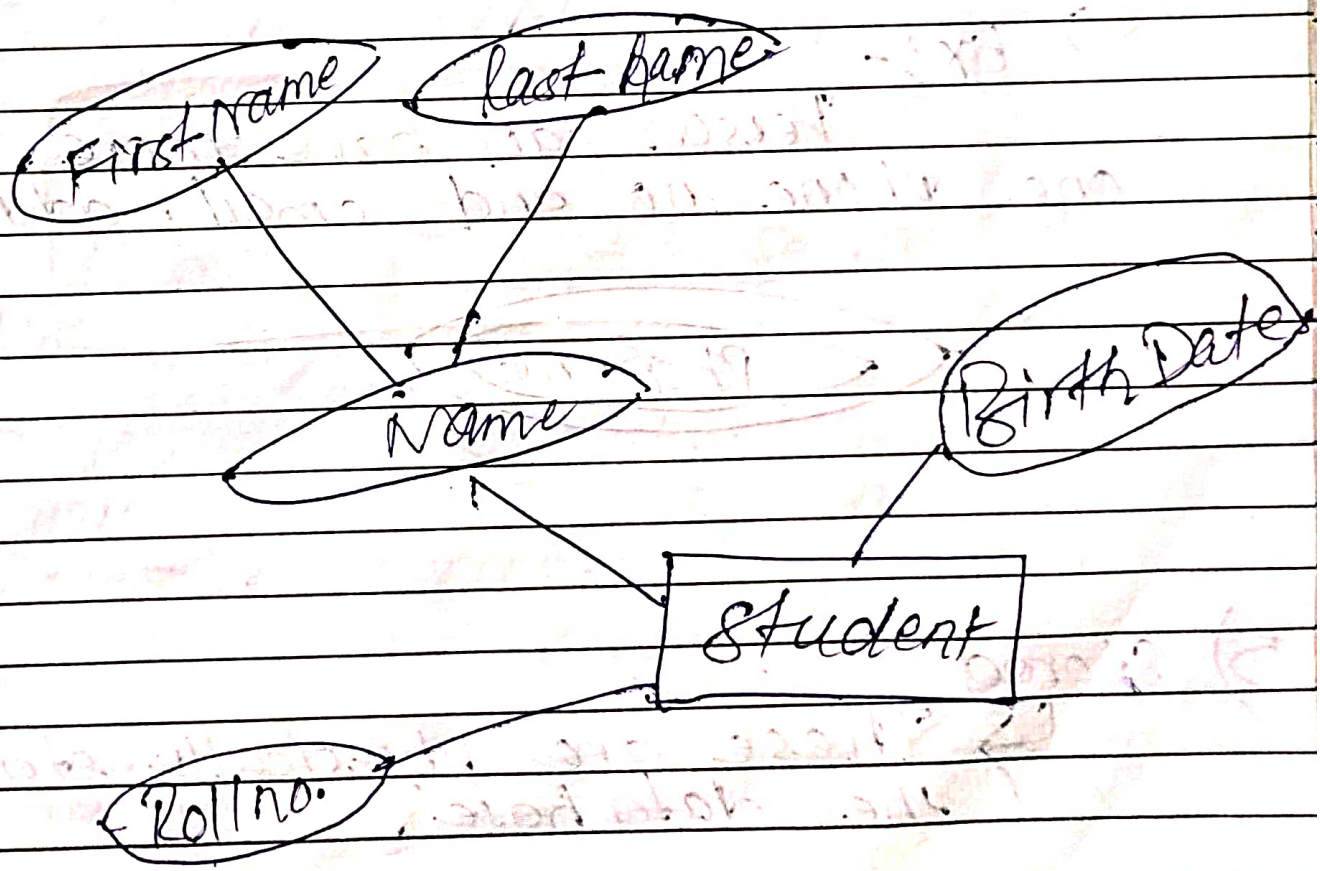
↳ made of more than one simple attributes.

↳ Composite attribute is divided in a tree like structure.

Ex:-

A student's complete name may have first-name and last-name.

An address may have street, city, state, country and pin code.

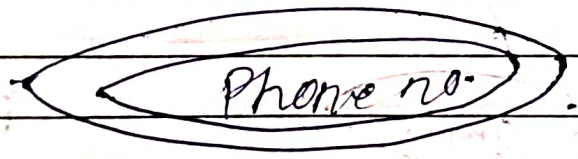


DATE: __/__/____
PAGE: _____

3) Single-valued
↳ Contains only single value

Ex:-
Social-security-no
Aadhar-card-no
Roll-no.

4) Multi-valued
↳ may contain more than one values
↳ Represented by Double-Ellipse.
↳ Ex:-
Person can have more than one phone no and email-address



5) Stored
↳ These are physically stored in the database

Mostly all the attributes are stored in database expect few ones.

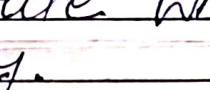
DATE: __/__/____
PAGE: _____

6) Derived :-

↳ Are not physically present in the database but they are derived from those certain stored attributes.

Ex:-

Age is not stored but derived from DOB.

↳  → Dashed ellipse.

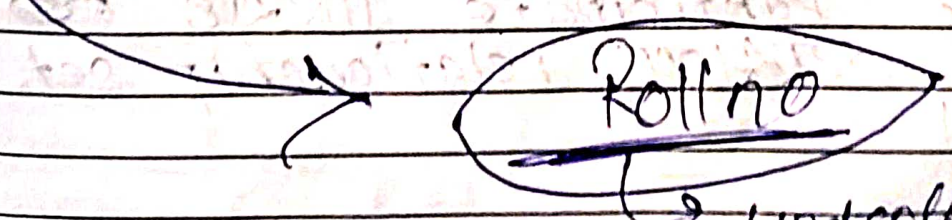
7) Key :-

↳ Attribute which uniquely identifies each entity.

Ex:-

Student को name same हुन सक्छ, address same हुन सक्छ if siblings छ भने Parents को Phone no same हुन सक्छ तर key attribute भयो भने त्यो different हुन्छ person by person.. Ex Roll no.

↳ It represents a primary key.



↳ underlined ellipse

Imp

DATE: __/__/____
PAGE: _____

Possible Combos

These attributes can come together in a way like :-

a) Simple single-valued
b) simple multi-valued
c) composite single-valued
d) Composite Multi-valued

Maping Cardinality and Participation Constraints

Maping cardinalities:-
(Cardinality Ratio) :-

Means एक entity set का कोई entity Participate कितने ~~कितने~~ relationship set बनाएगा।

PYQ defr

↳ Useful in describing binary rel^n.

Cardinality of types:-

a) One - one
b) One - many
c) Many - one
d) Many - many
(M - N)

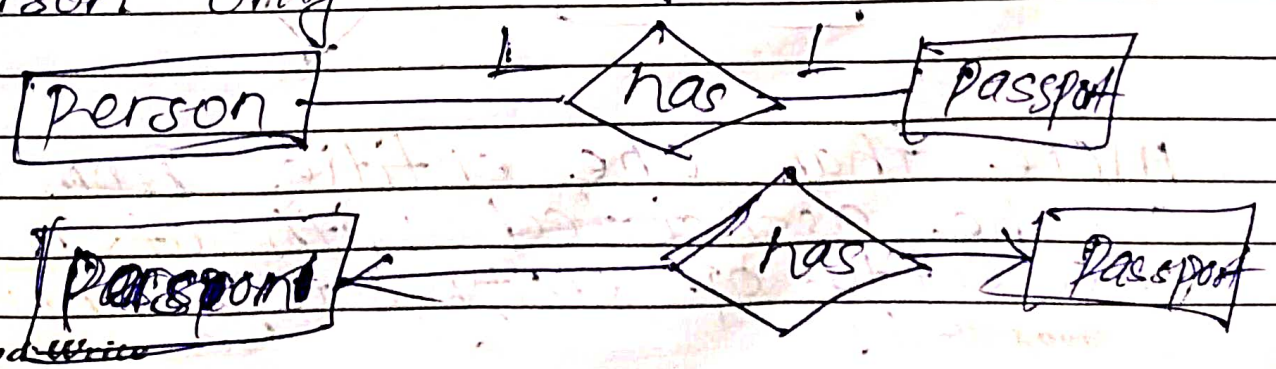
→ Cardinality can be of different types OR There are four types of rel^n

a) 1 - 1

↳ एउटा entity set को - एउटा entity at most एउटा मात्र entity sangd associated हुन्छ (एउटा set र अर्को set at most 1 - 1)

Ex:-

A person has only one passport and a passport is given to one person only.

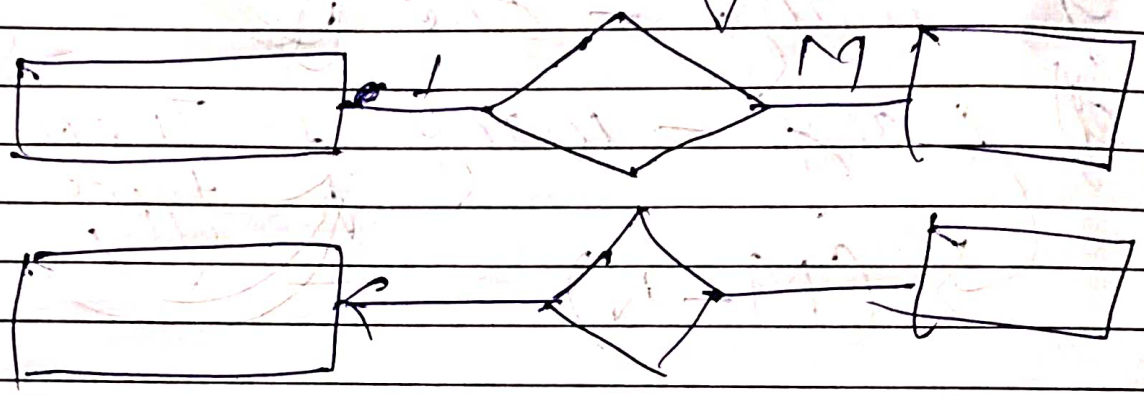


Good Write

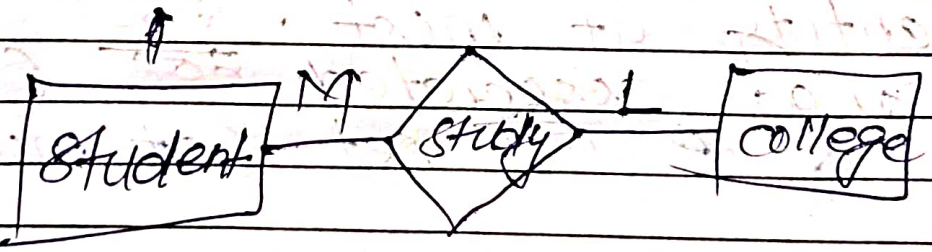
set A को एक entity को association set B की एक entities सेट हुन सकछ तर set B को एउटा - set A को एउटा सेट मात्र हुन्छ।

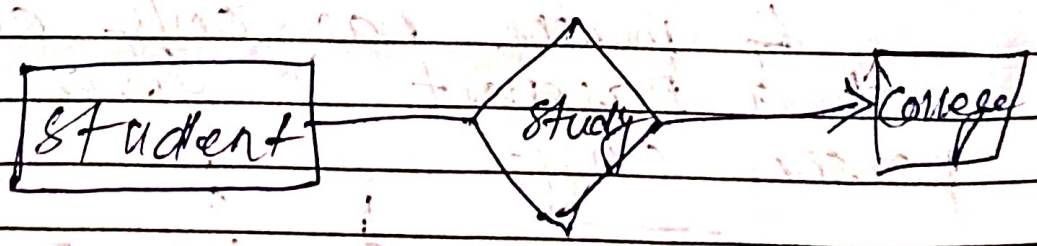
b) (1 - M) :-

→ one entity from entity set A is associated with more than one entities of set B. however entity of set B can be associated with at most one entity.



c) M - 1 :-





more than one entities from set A can be associated with at most one entity of entity set B, however

an entity from entity set B can be associated with more than one entity from entity set A.

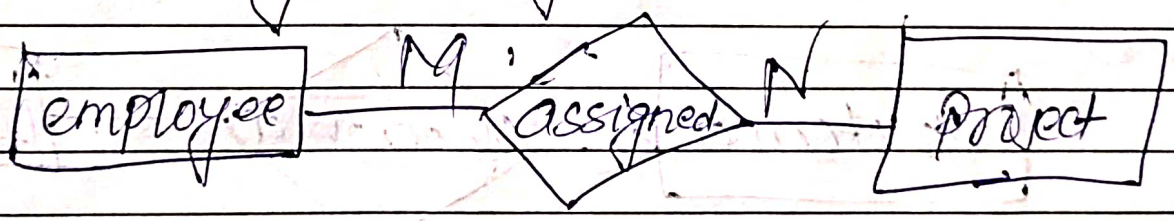
Multiple students can study in a college but a student can only study in one college.

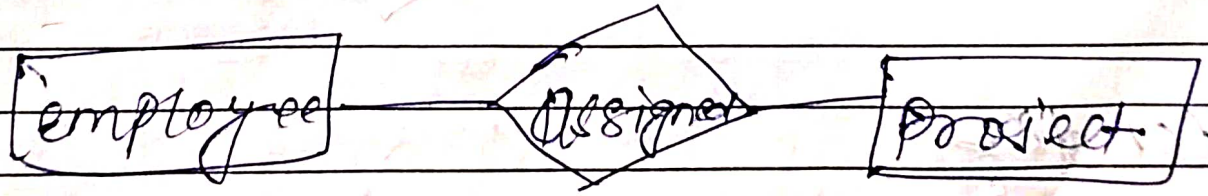
d) M – N

One entity from A can be associated with more than one entity from B and vice versa.

Ex:-

an employee can be assigned to many projects and project can too have many employees.





How to choose Relationship?

↳ Actually depends upon the real world situation :-

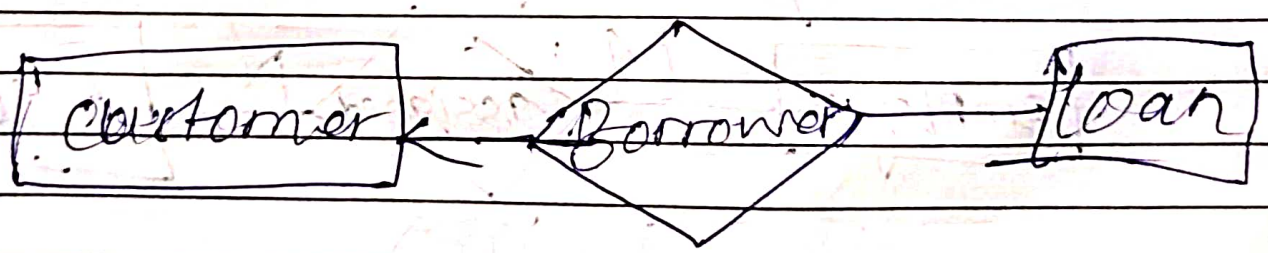
Ex:- Lets consider borrower rel^n b/w ~~be~~ costumer and loan.

cond^n 1 :-

if loan can only belong to only only costumer and vice versa.
i.e. 1 — 1

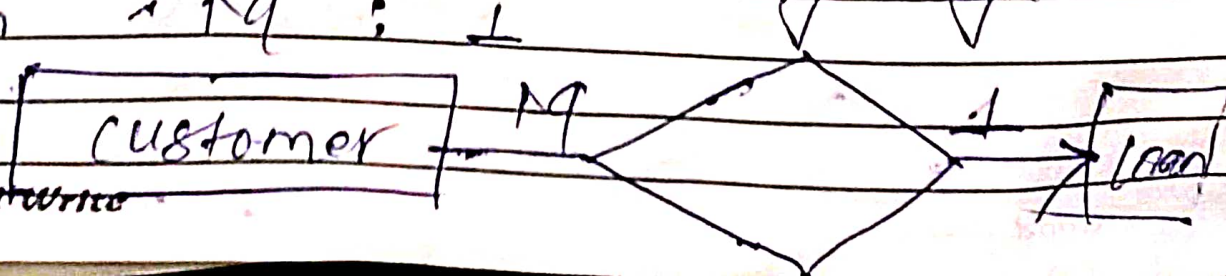
cond^n 2 :-

if loan can belong to only one customer and customer can have several loans then set from customer to loan is 1 : M



cond^n 3 :-

if loan to several customer and ~~sere~~ comtumer only only loan then M : 1

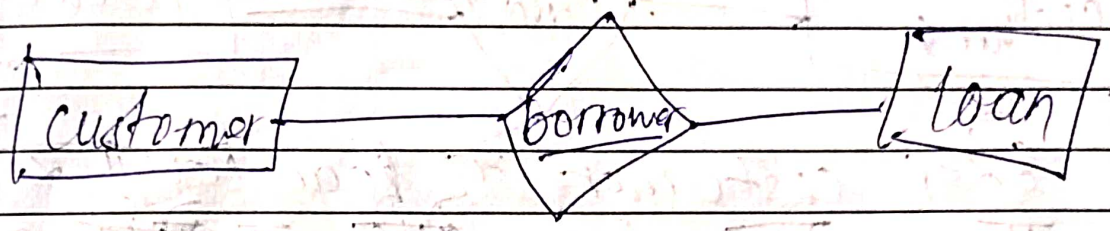


GoodWrite

Condn 4:-

loan can belong to several customers (as loans can be taken jointly by several business partners)

M : N



Participation Constraints:-

↳ Basically applied on the entity participating in the reln set.

↳ relation set मा सबै entity participate गरिराख्नु कुनै कुनै entities मात्र participate गरिराख्ने भनिन्छ।

* Total participation:-

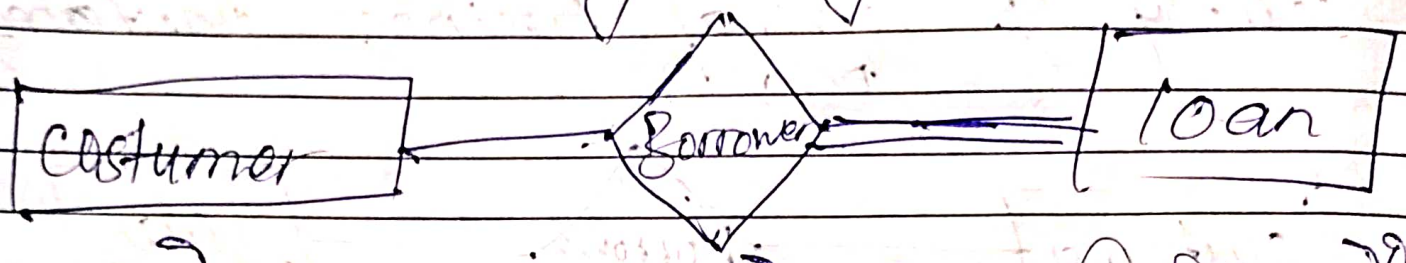
↳ Each entities involved. Represented by double lines.

↓↓

DATE: __/__/____
PAGE: _____

* Partial Participation:-

→ Not all the entities are involved denoted by single line.



हरेक costumer ले loan लिएको छैन तर हरेक loan कोहि न कोहि costumer ले लिएको छ।

Keys in DBMS

Key:-

Tabular form मा store भएको record मा दुई वटा same name भएको record हुन सक्छ र record हेर्न खोज्दा suppose हामीले Amrit search गरेमा र 2 वटा record देखियो र कुन Amrit हो थाहा भएन यसले key को case

⇓

So, key is an attribute or set of attributes that uniquely identifies a record (or tuple) from table.

DATE: __/__/____
PAGE: _____

Purpose :-

↳ Uniquely identify any record or row of data from table.

↳ It also sets, establish and identify the relⁿ betn relⁿ betn tables.

Imp

Types of Keys in DBMS :-

1) Super
2) Candidate
3) Primary
4) Alternate
5) Foreign
6) Composite

DATE: __/__/____
PAGE: _____

1) Super key :-

↳ Super key is the combination of all possible attributes that uniquely identifies the rows (or tuple) in the given relⁿ.

a) Is superset of a candidate key.

b) A table can have many super keys.

c) A super key can have additional attribute that are not needed for unique identity

⇓

Super key combination पति हुन सक्छ ।।

Ex:- मात्र Aadhar Id, Employee Id लाइएर → Aadhar Id — Name,
→ Employee Id — Address.
3, 4, 5 पनि पर्ने possible combos.

max s.k : If there are n attributes and each of them are super key then:
max s.k $= {}^nC_1 + {}^nC_2 + \cdots + {}^nC_n = 2^n - 1$

2) Candidate key :-

↳ An attributes or set of an attribute which can uniquely identify a tuple.

Candidate keys are not allowed to have Null values.

DATE: __/__/__
PAGE: _____

> A candidate key is a minimal super key or a super key with no redundant attributes.

→ Called minimal key because we select a candidate key from a set of super key such that selected candidate key is the minimal attribute required to uniquely identify the table.

> They are defined as distinct set of attributes from which primary key can be selected.

↳ Candidate keys select गर्दा यसरी गर्ने कुन set of candidate key भने त्योँ set लाई खोजेर एउटा एउटा candidate key बनाएम भने हरेक tuple लाई uniquely identify गरोस् र set र element element को content repeat नहोस्

DATE: __/__/____
PAGE: _____

3) Primary Key :-

A primary key is one of the candidate key choosen by the database designer to uniquely identify the tuple in the reln.

- It's value can never be zero. (never null)
- value must always be unique (not duplicate) → not repeated (primary key)
- It's values can never be changed (rarely updated) i.e updation is impossible
- It must be assigned when inserting a record.
- A relation is allowed to have only one primary key.

4) Alternate Keys :-

- Out of all candidate keys, only one gets selected as primary key, remaining keys are known as alternate keys.

Referential Integrity

DATE: __/__/__
PAGE: _____

Imp

5) **Foreign keys** :-

- A key used to link two tables together
- An attribute (or sets of attributes) in one table that refers to the primary key in another table.

The purpose of foreign key is :-

- To ensure (or maintain) **referential integrity** of the data.

मतलब दुइटा table लाई अर्को table संग link गर्न र Relation maintain गर्न ।

Foreign key भएको table ⇒ Referencing table → child / Foreign table

Primary key वाला table ⇒ Referenced table
⇓
Master table
⇓
Primary table
⇓
Parent table

DATE: __/__/____
PAGE: _____

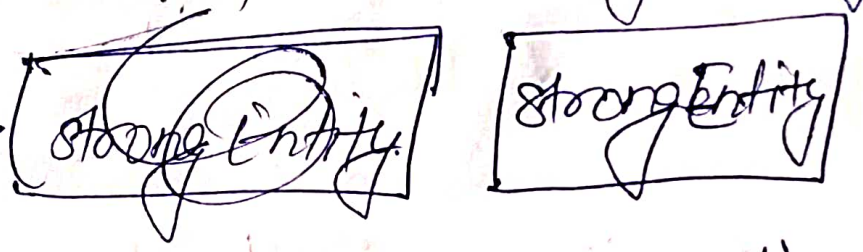
Foreign key :-

- References the primary key of the table
- It can only take those values which are present in the primary key of the referenced relation ⇒ referencing integrity maintain होता
- Foreign key may have a name other than that of a primary key.
- It can take the NULL value
- There is no restriction on a foreign key to be unique
- In fact, foreign key is not unique most of the time
- Referenced relⁿ may also be called as the master table or primary table.
- Referencing relⁿ may also be called as the foreign table

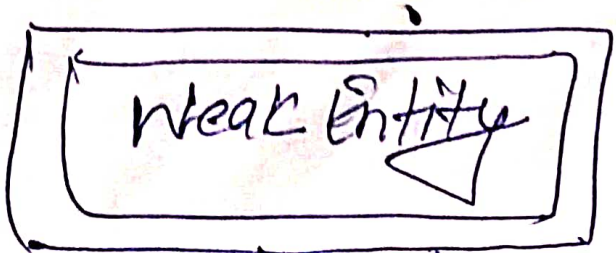
6) Composite key:-
A key that has more than one attributes is called composite or compound key.

Weak and strong Entity

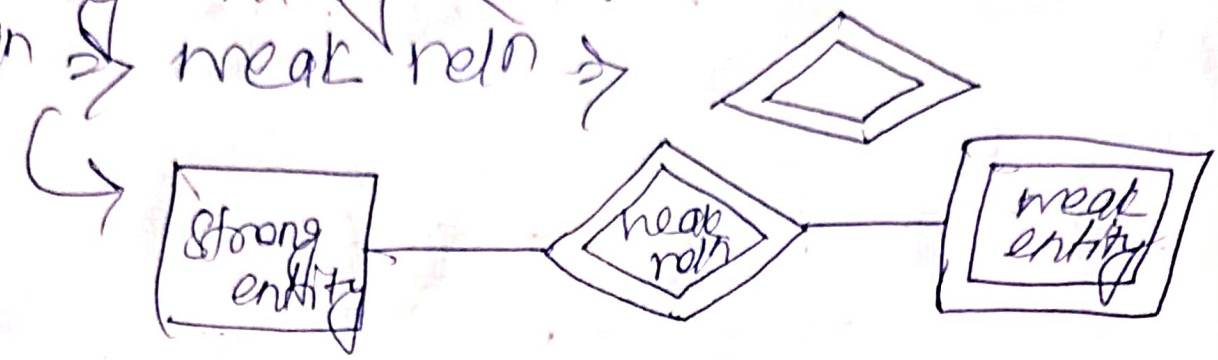
1) Strong entity (strong entity set):-
- It always has a primary key.
- It's existence is independent of other entity. i.e. it is independent entity.
- set of strong entities is called strong entity set
- Strong entity is represented by a single rectangle.



2) Weak Entity (Weak Entity set):-
- The weak entity doesn't have a sufficient attributes to form a primary key. i.e. It doesn't have a primary key.
- It is dependent on strong entity to ensure its existence
- A set of weak entities is called weak entity set
- weak entity is represented by double rectangle.



Strong entity र weak को बीच को reln ⇒ weak reln ⇒

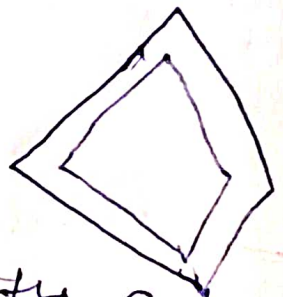


Strong entity ⇒ Identifying Entity set

weak Relⁿ ⇒ Identifying Relⁿ.

Strong — Weak Entity को cardinality 1 — M

Indentifying Relⁿ / weak Relⁿ



* Participation of weak entity set from the identifying set is always total.

* The discriminator (or partial key) of a weak entity is set of attributes that distinguishes among all the entities of a weak entity set

Discriminator / Partial key

DATE: / /
PAGE:

* The primary key of weak entity set ⇒ Primary key of strong + Partial key of weak

Primary key of strong मा weak entity को existence depend हुन्छ।

Extended Entity Rel^n

1) Generalization
2) Specialization
3) Aggregation

Generalization :-

→ Process of extracting common properties from a set of entities and create a generalized entity from it.

→ Generalization is a bottom-up approach in which two or more entities can be combined to form a higher

Bottom-up ⇒ Bottom मा low level entities कुछ एसलाई combine गरेर high level entity बनाउनु नुहुन up माथि कुछ।

level entity if they have some attributes in common.

↳ Subclasses are combined to ~~form~~ make a superclass.

⇒ Generalization is used to emphasize the similarities among lower-level entity set and to hide difference in the schema.

2 Specialization :-

Pyq

↳ Just opposite to generalization.

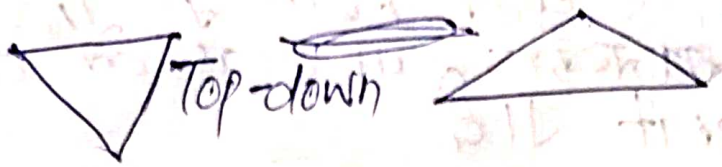
In specialization an entity is broken down into sub-entities based on their characteristics.

↳ It is a ~~top~~ Top-down approach where higher level entity is specialized into two or more lower level entities.

↳ Specialization is used to identify the subset of an entity set that shares some distinguishable characteristics.

↳ Specialization can repeatedly applied to refine the design of schema.

Good Write



DATE: / /
PAGE:

↳ Specialization is depicted by triangle component labeled ISA.

सुरु मा Superclass defined हुन्छ, त्यसपछि मात्र sub class defined हुन्छ।

Inheritance:-

↳ It is the important feature of generalization and specialization that allows lower level entities to inherit the attributes of higher level entities.

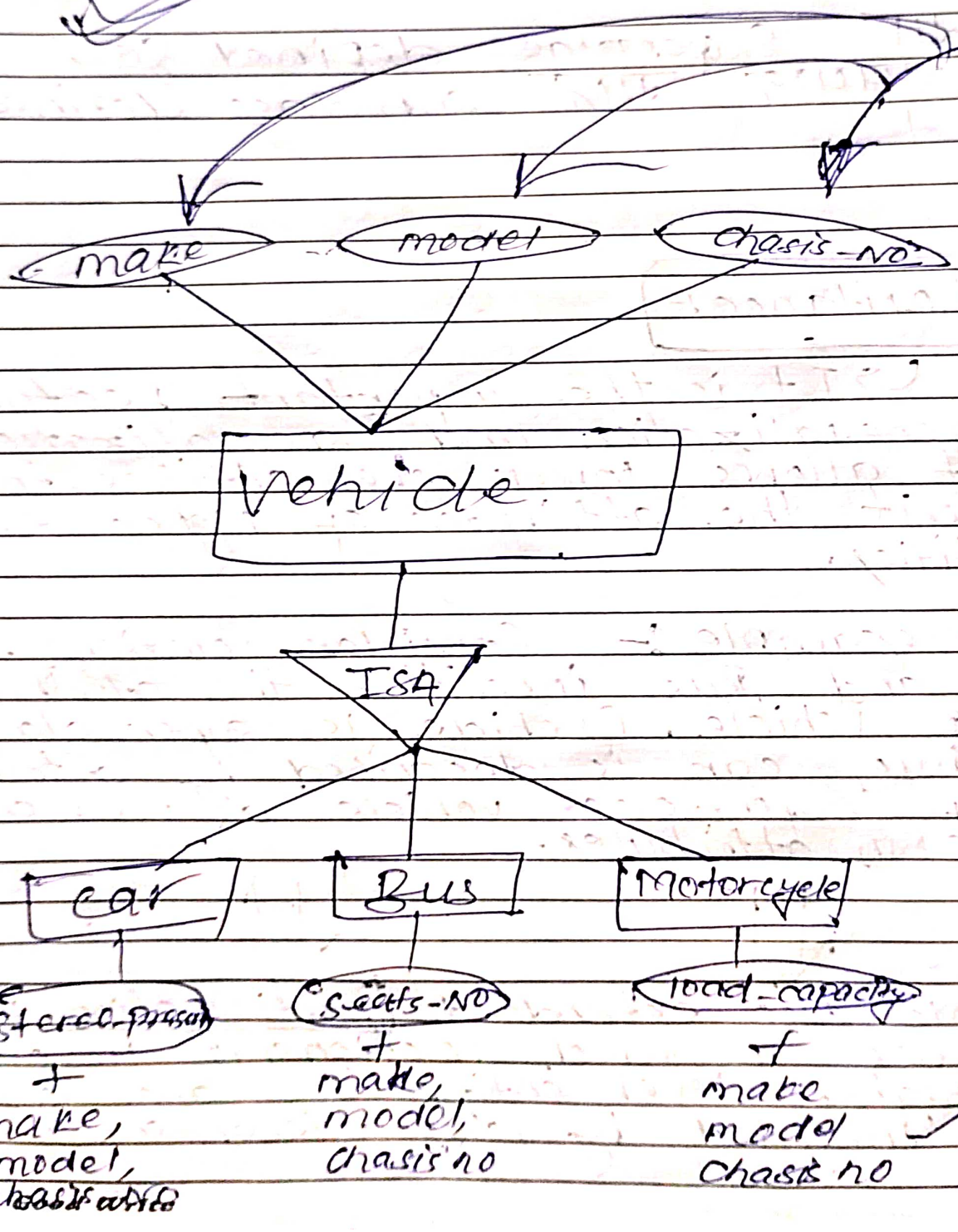
↳ For example :- Consider relations car and Bus inheriting the attributes of vehicle, (vehicle is super class). Thus, car is described by attributes of super-class vehicle as well as its own attributes.

This also extends to participation inheritance in which relationship involving higher-level entity-sets are also inherited by lower-level entity-sets.

higher level entities फुनँ रान सा involve
छ भनेर त्यस बाट भनेको subclass ले पनि
त्यो सबै रान inherit गर्छ

DATE: __/__/____
PAGE: _____

Participation Inheritance



How Schema or Tables can be formed ??

DATE: __/__/____
PAGE: _____

① पूरा leaf entities / Nodes है उसको मात्र Table बनाते।

5) Aggregation :-

Confusing है

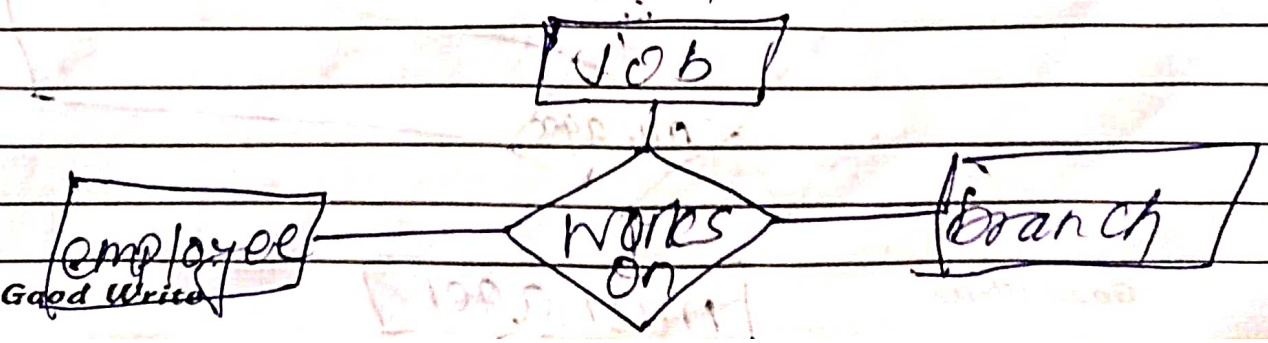
↳ Aggregation is used when we need to express ~~betn among~~ relationship among relationships.

↳ Aggregration is an abstraction through which relationships are treated as higher level entities.

↳ It is a process ~~in~~ when a relationship betn two entities is considered as a single entity and again this single entity has a relationship with other entity.

EX:-

Basic ER model ले relationship यानि relationship में involve भएको देखाउन सक्दैन।



Good Write

DATE: / /
PAGE:

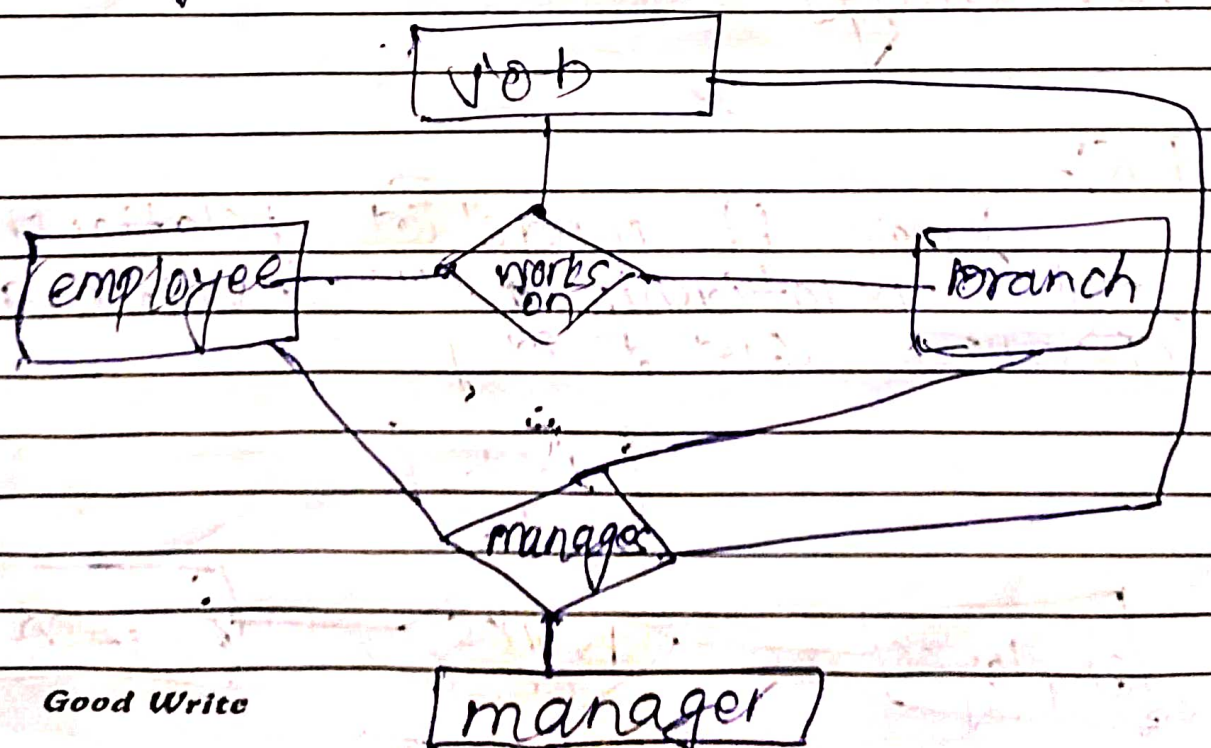
consider a ternary rel^n [works on] betn [employee], [branch] and [job]

An employee works on a particular job at a particular branch.

Then
Suppose we want to assign a manager for jobs performed by an employee at a branch (i.e. want to assign managers to each employee, job, branch combination)

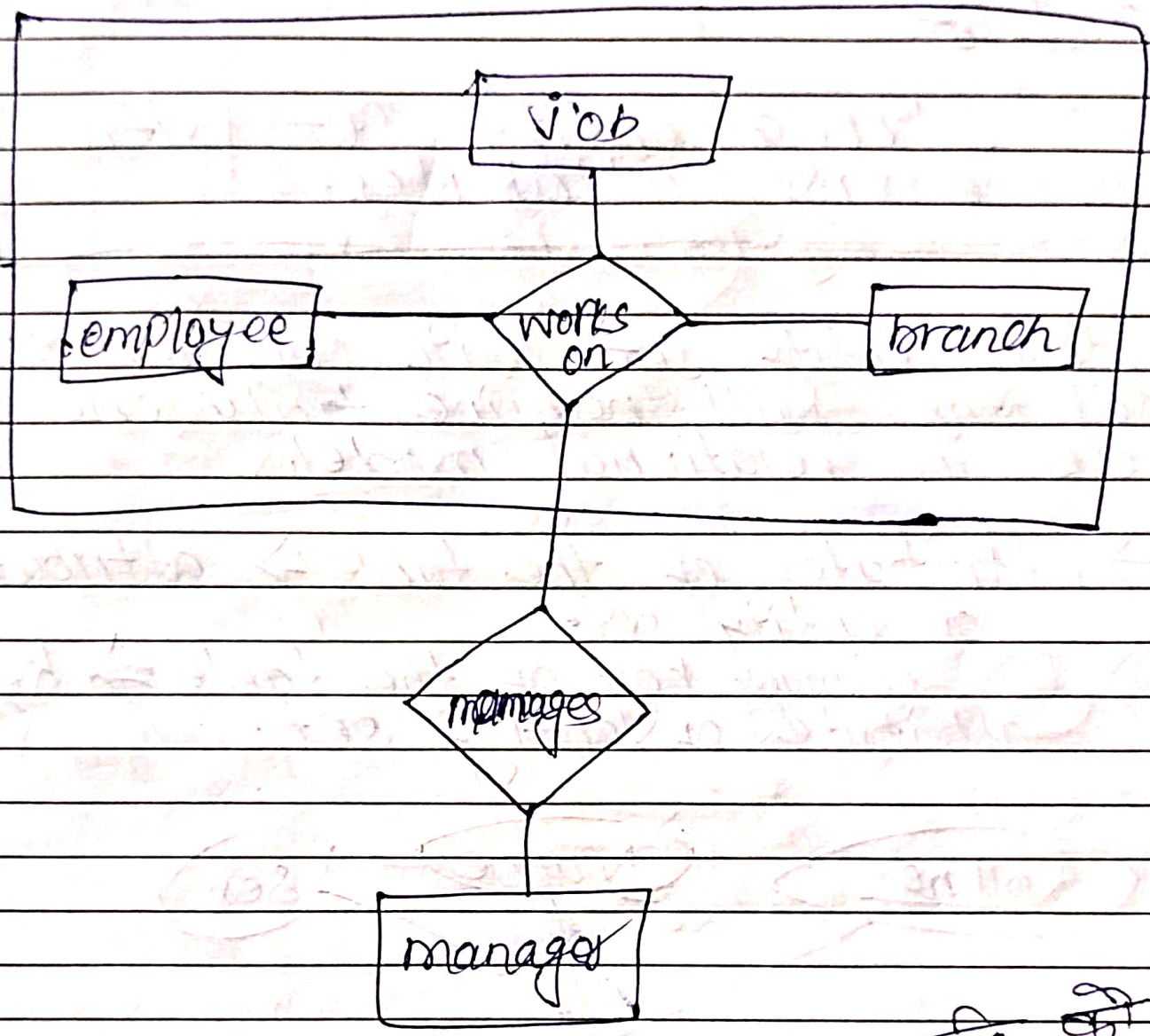
Needs a separate manager entity set.

Rel^n betn each manager, employee, branch and job entity.



DATE: __/__/____
PAGE: _____

After Aggregation :-



इसलिए ER Diagram को redundancy

Aggregation eliminates this redundancy

↳ Treats relⁿ as an abstract entity.

↳ Allows relⁿ both relⁿ.

↳ Abstraction of relⁿ into new entity.

Converting ER to Tables

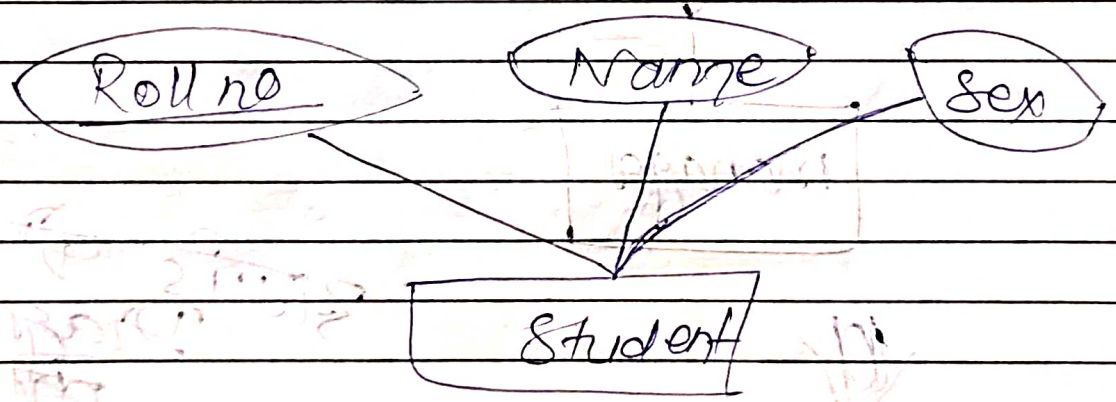
Rule 1 :-

↳ Strong entity set with only strong attributes.

A strong entity set with only simple attributes will require only one table in relational model.

↳ Attributes of the table ⇒ attributes of entity set.

↳ The primary key of the table ⇒ key attribute of entity set.



Rollno	Name	sex

~~Seet~~ Schema :- Student (Rollno, Name, sex)

Good Write

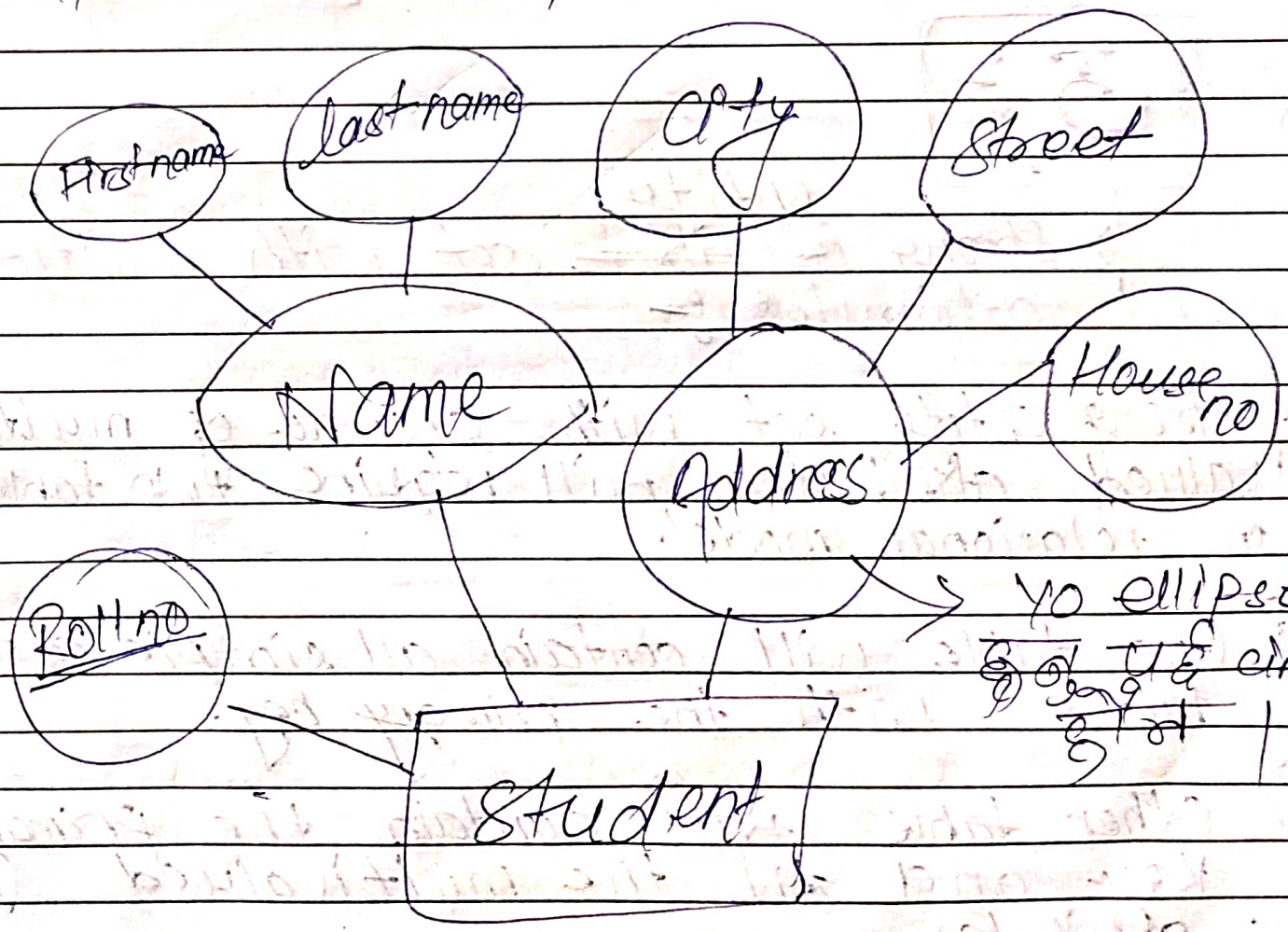
DATE: __/__/____
PAGE: _____

Rule 2:-

→ Strong entity with compound attributes.

1. A strong entity with compound attributes will require only one table in relational model.

2. While conversion, simple attributes of the composite attributes are taken into account and not the composite attributes itself.



DATE: __/__/____
PAGE: _____

Table :-

Rollno	Firstname	lastname	Houseno	street	city

Schema :- Student (Rollno, Firstname, lastname, Houseno, street, city)

Rule 3 :-

↳ Strong ~~attribute~~ entity set with multi-valued attributes :-

A strong entity set with any no. of multi-valued attributes will require two tables in relational model.

(a) One table will contain all simple attributes with the primary key.

(b) Other table will contain the primary key and all the multivalued attributes.

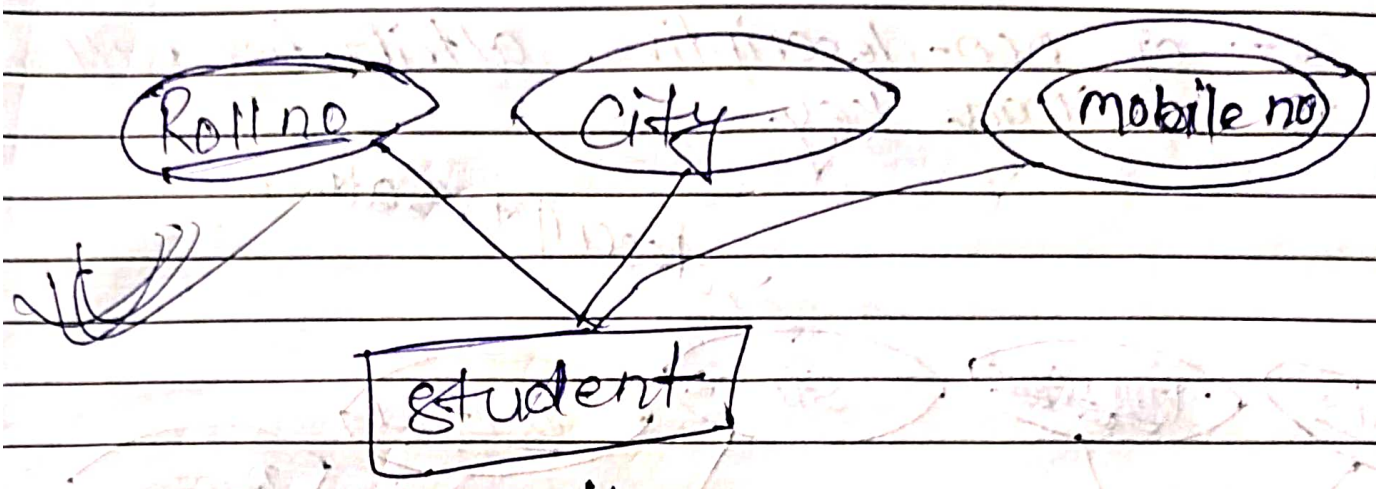


Table 1:-

Roll no	city

Table 2:-

Roll no	mobile no

Rule 4 :-

Translating Relⁿ set into a Table

↳ A relationship set will require one table in the relational model.

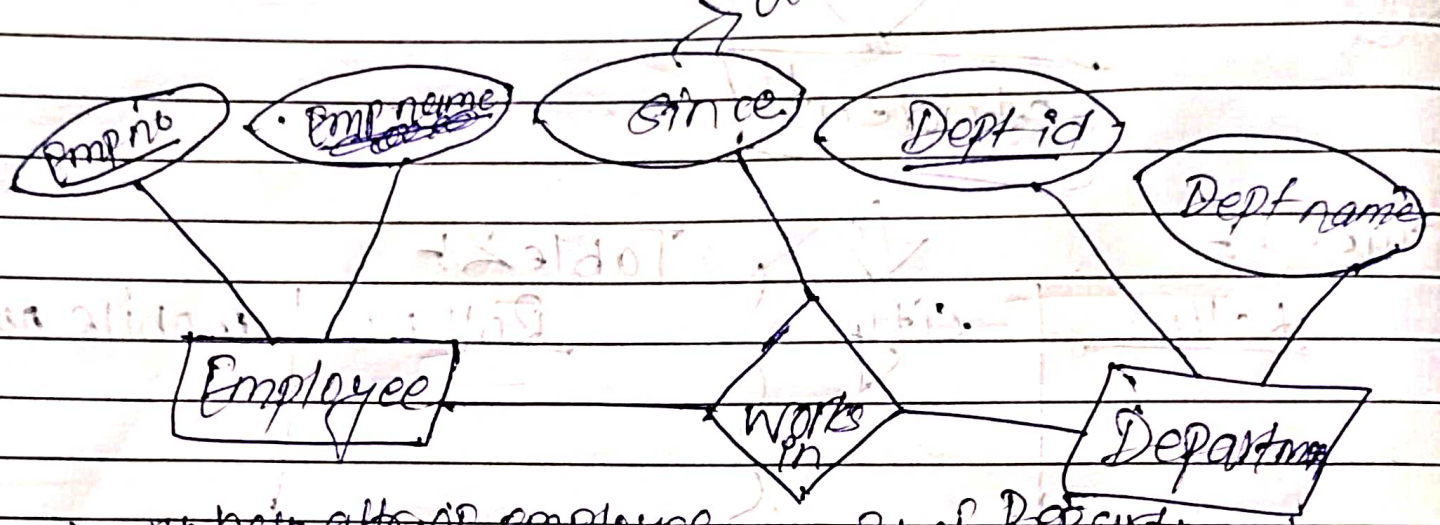
Attributes of the table are

ⓐ Primary key attributes of participating entity sets

ⓑ It's own descriptive attributes if any

DATE: __/__/____
PAGE: _____

↳ set of non-descriptive attributes will be the primary key.



primary key attr. of employee P.K of Department.

Schema for works in Reln:-

emp no	Dept id	since
Foreign key	Foreign key	

descriptive attr.

⇓

Schema:- works in (emp no, Dept id, since)

⇓ 1 + 2 वटा अन्य tables पनि बन्छ।

⇒ For the given ER diagram, three tables will be required in relational model:-

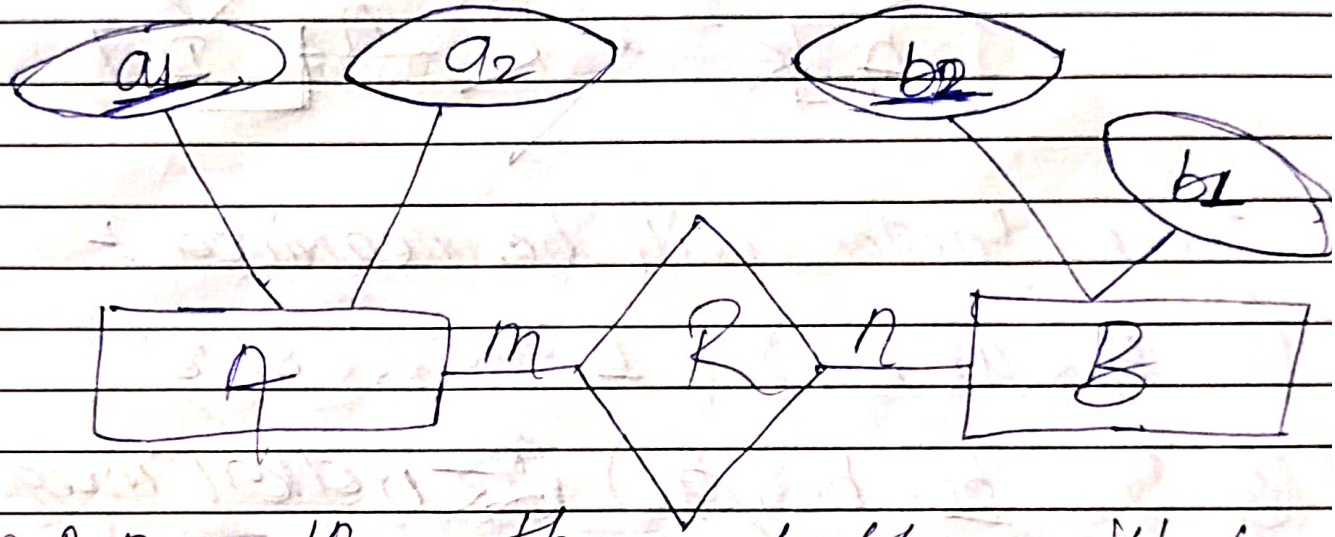
1. ⇒ 1 table for entity set employee
2. ⇒ 1 table for entity set Department
3. ⇒ 1 table for reln set works in.

DATE: __/__/____
PAGE: _____

Rule 5 :

Binary reln with cardinality ratio

(I) $\boxed{m : n}$



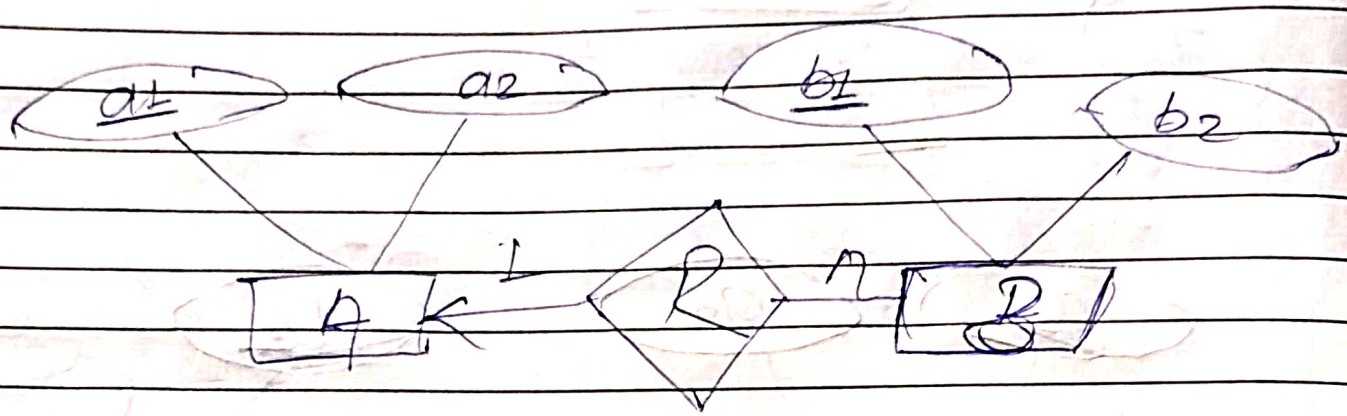
m : n reln, three tables will be required :-

1. A ($\underline{a_1}$, a_2) → Primary key
2. B (b_1, $\underline{b_2}$) → Primary key
3. R ($\underline{a_1}$, $\underline{b_2}$) → Primary keys of A and B.

DATE: __/__/__
PAGE: _____

(II) → one - many

$\boxed{1 : n}$



Two tables will be required :-

(a) A (a_1, a_2) (1 वाला side)

(b) B (b_1, b_2, a_1) (R 1:n वाला merge हुआ)

→ a_1 acts as a Foreign key

Combined table will be drawn from entity set B and rel^n set R.

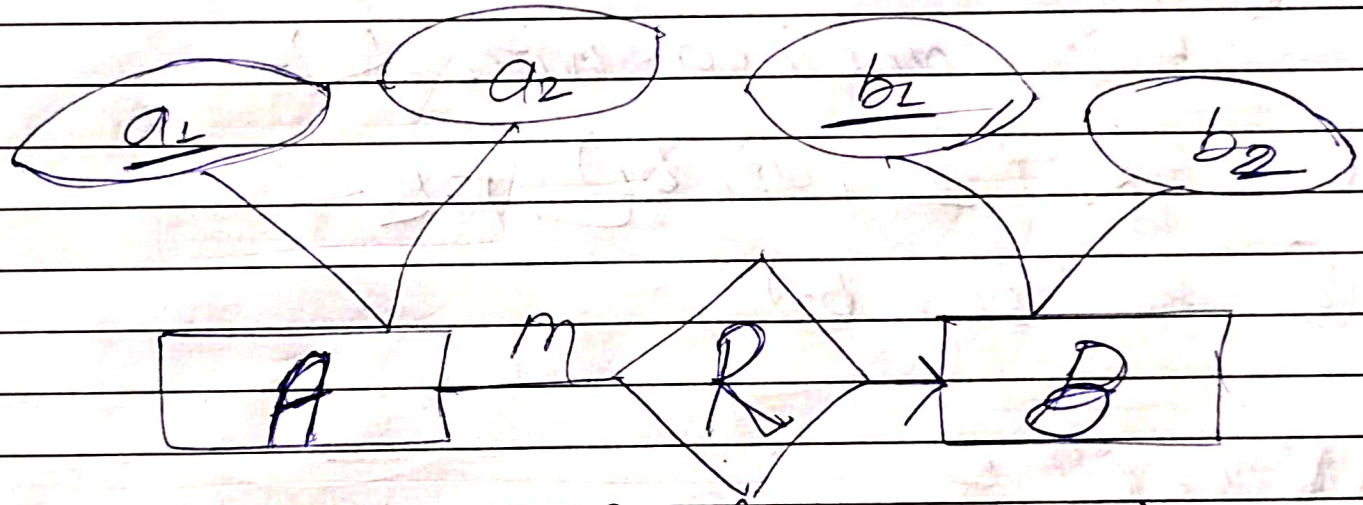
DATE: __/__/____
PAGE: _____

III. $m : 1$

Same as $1 : m$,

⇒ 1st table combined table हुएर Relation set र many side (m' वाट) का हाम्रो set

⇒ 2nd ~~set~~ table 1 side.



① 1st table AR (a_1, a_2, b_1) → foreign key

2nd table B (b_1, b_2)

DATE: __/__/____
PAGE: _____

IV

1:1

कुनै एउटा side लाई R sanga combine गर्दिने।

⇓

2 वटा table हुन्छ तर way पनि 2 वटा नै हुन्छ।

Way 01 :-

A is combined with B :-

① AR (a_1, a_2, b_1) → F.K

② B (b_1, b_2)

Way 02 :-

B is combined

① A (a_1, a_2)

② BR (a_1, b_1, b_2)
P.K.

DATE: __/__/____
PAGE: _____

Rule 6 :-

For Binary Relationship with both the cardinality constraints and participation constraints. → Imp

1:- Participation Constraints will be implemented as discussed in the rule 5.

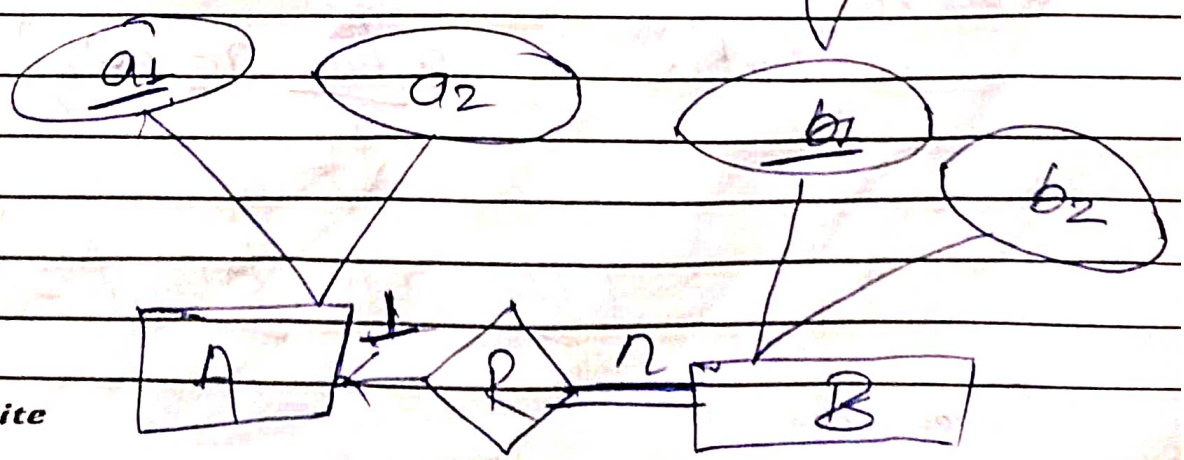
2:- Total Participation के लिए Foreign key null हुए सकते है।

Two cases :-

(1) Cardinality Constraints + Participation Constraints (Total participation only from one side.)

(2) Cardinality Constraints + Participation constrains (Total Participation from both sides.)

(1) Total participation from Only one side



m:n + Total From both

DATE: __/__/__
PAGE: _____

Two Tables are:-

(a) BR($\underline{b_1}$, b_2, a_1) → Foreign key

(b) A($\underline{a_1}$, a_2)

because of total participation, foreign key a_1 has acquired NOT NULL constraints, so can't be null.

Case II

1:1 + Total From Both

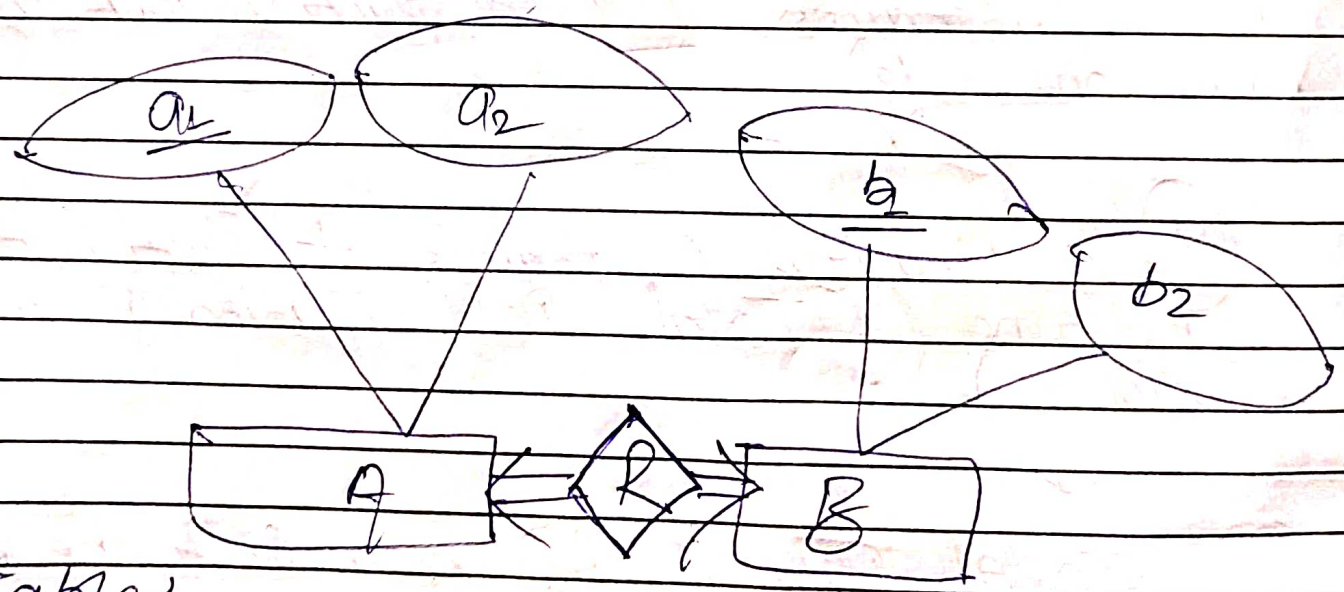


Table:-

ABR($\underline{a_1}$, $\underline{b_1}$, a_2, b_2)

⇓

($\underline{a_1}$, a_2, $\underline{b_1}$, b_2)

weak entity set always in 1 : n

DATE: __/__/__
PAGE: _____

Rule 7 :

Weak entity set

→ weak entity set always appears in association with identifying rel^n with participation constraint and there is always 1:n relationship from identifying set to weak entity set.

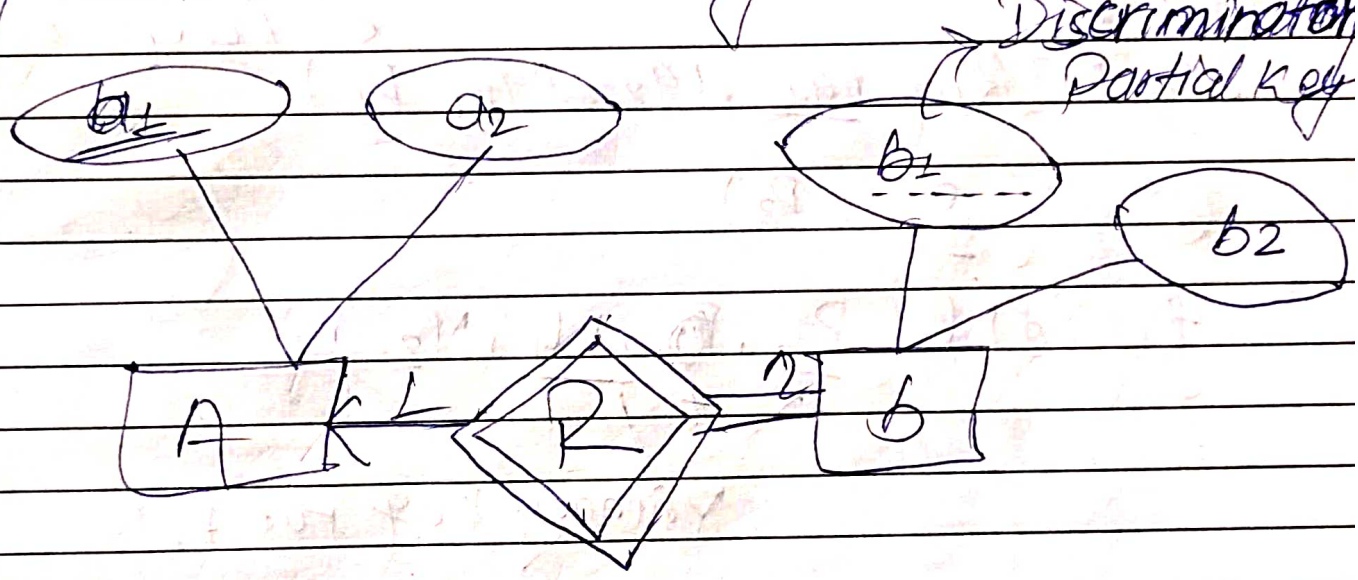


Table :-

A (a1 , a2)

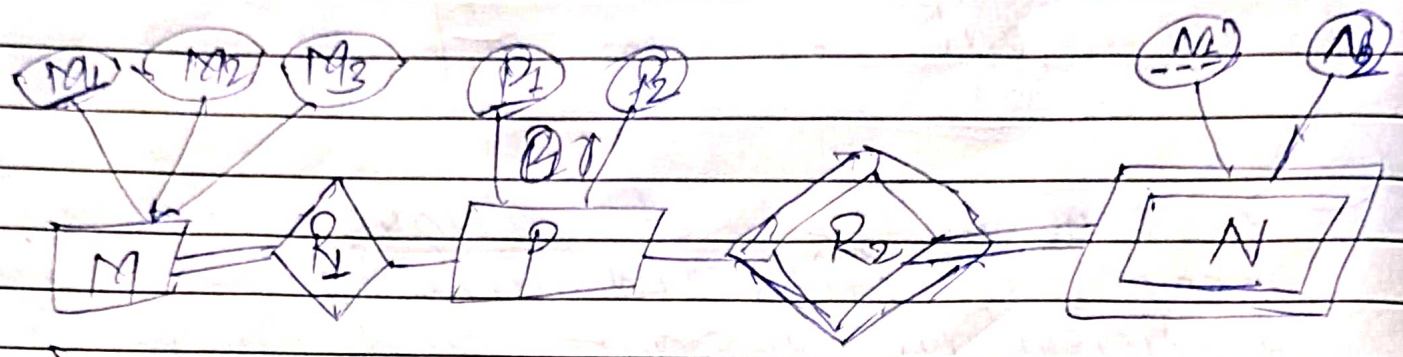
B (b1 , b2 , a1)

↓

a1 ≠ b1 मिलके Primary Key.

(Q) Find the minimum no. of tables required for the following ER diagram in relational model.

DATE: __/__/____
PAGE: _____



⇒ 3 Tables minimum

Tables:-

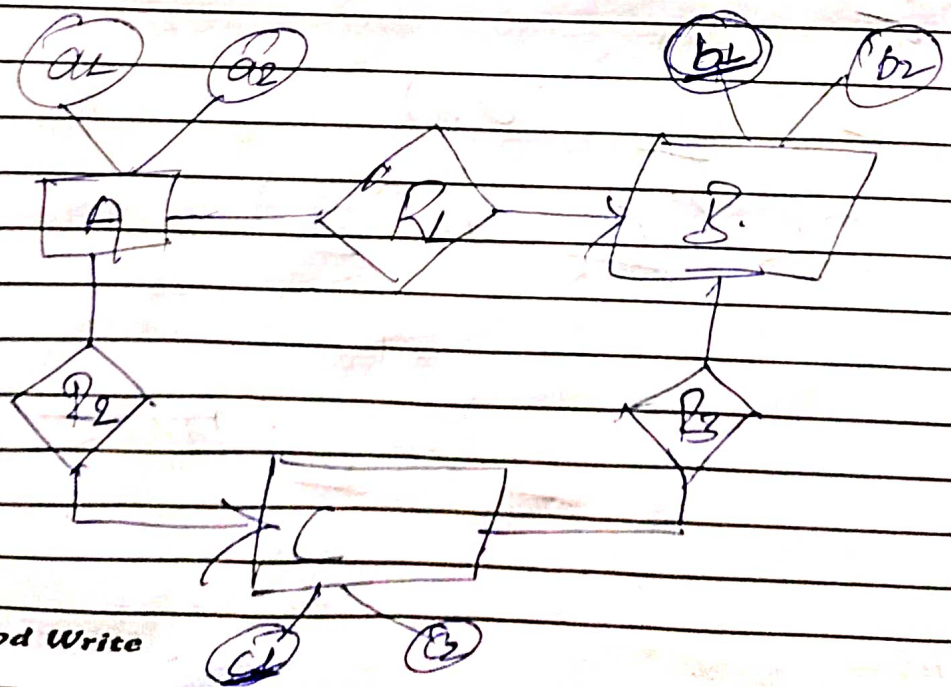
① $MR_1(\underline{M_1}, M_2, M_3, P_1)$ ↗ Foreign key

② $P(\underline{P_1}, P_2)$

③ $R_2N(\underline{P_1}, \underline{N_1}, N_2)$

Primary key for R_2N

(Q2)



DATE: __/__/____
PAGE: _____

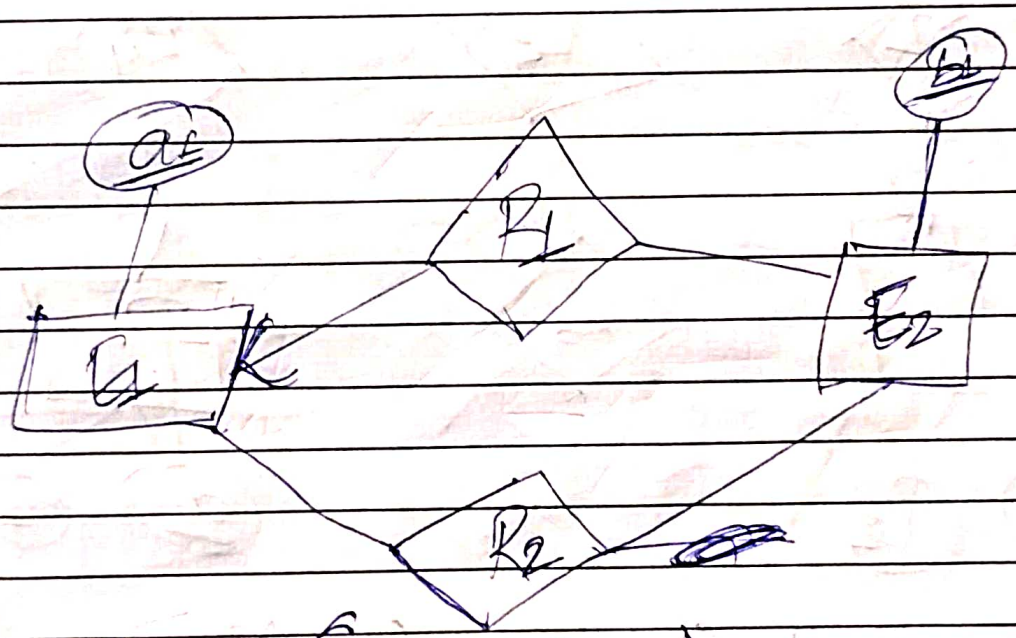
Q2 ⇒ 4 Tables Minimum

⇓

① AR_1R_2 ($\underline{a_1}$, a_2, b_1, c_1)
② B ($\underline{b_1}$, b_2)
③ C ($\underline{c_1}$, c_2)
④ R_3 ($\underline{b_1}$, c_1)

→ M:n ⇒ मात्र 3 बना पाऊँगे।

Q3



⇒ 3 Tables (Minimum)

⇓

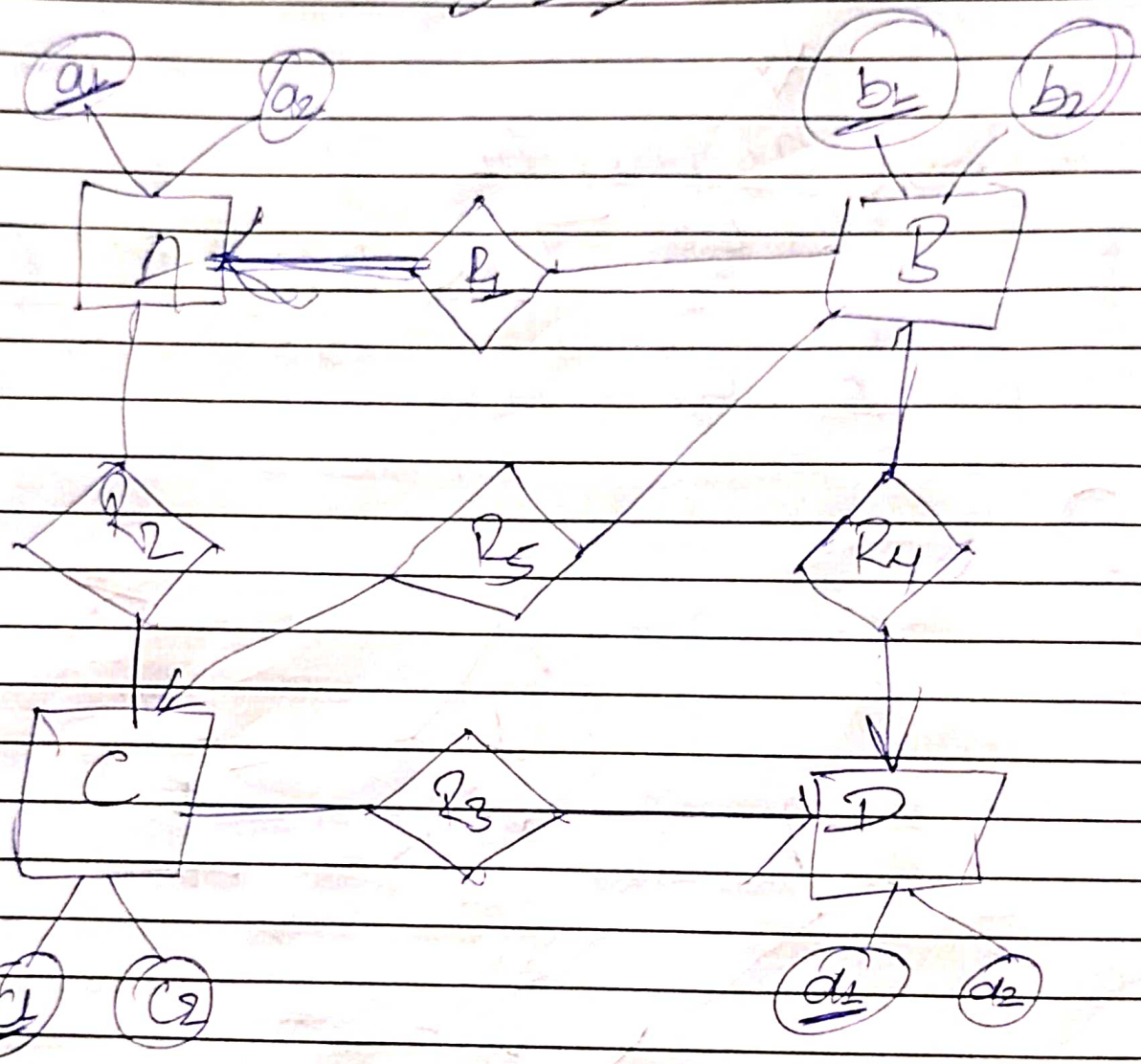
E_1 ($\underline{a_1}$)
E_2R_1 ($\underline{b_1}$, a_1)
R_2 ($\underline{a_1}$, $\underline{b_1}$)

BR_1R_5, A, R_2, CR_3, D

DATE: / /
PAGE:

Q4

Main table



① $BR_1R_4R_5$ (b_1, b_2, a_1, d_1)

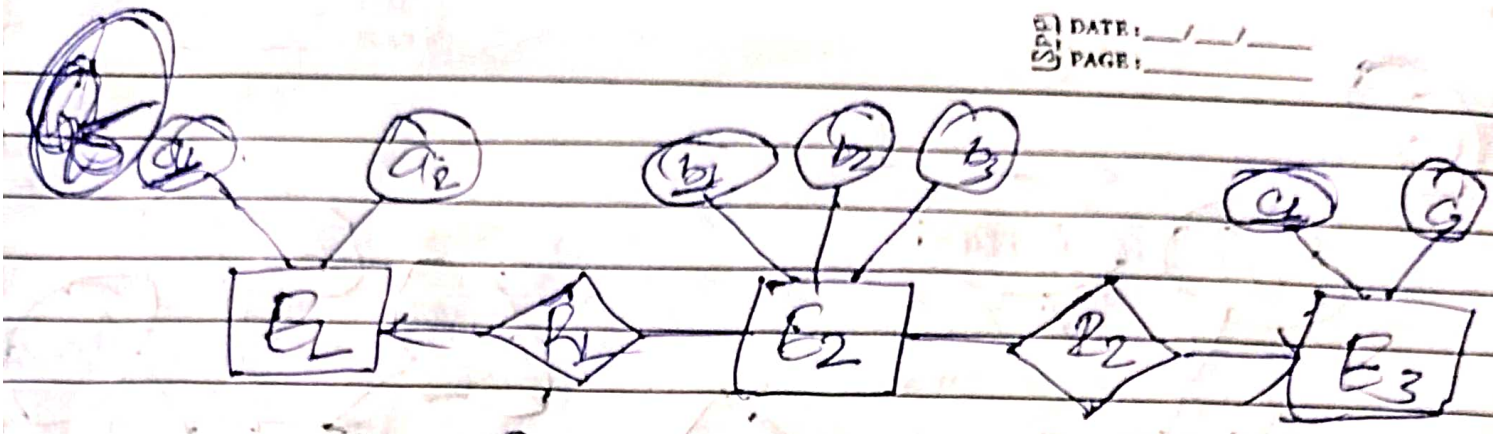
② A (a_1, a_2)

③ R_2 (a_1, c_1)

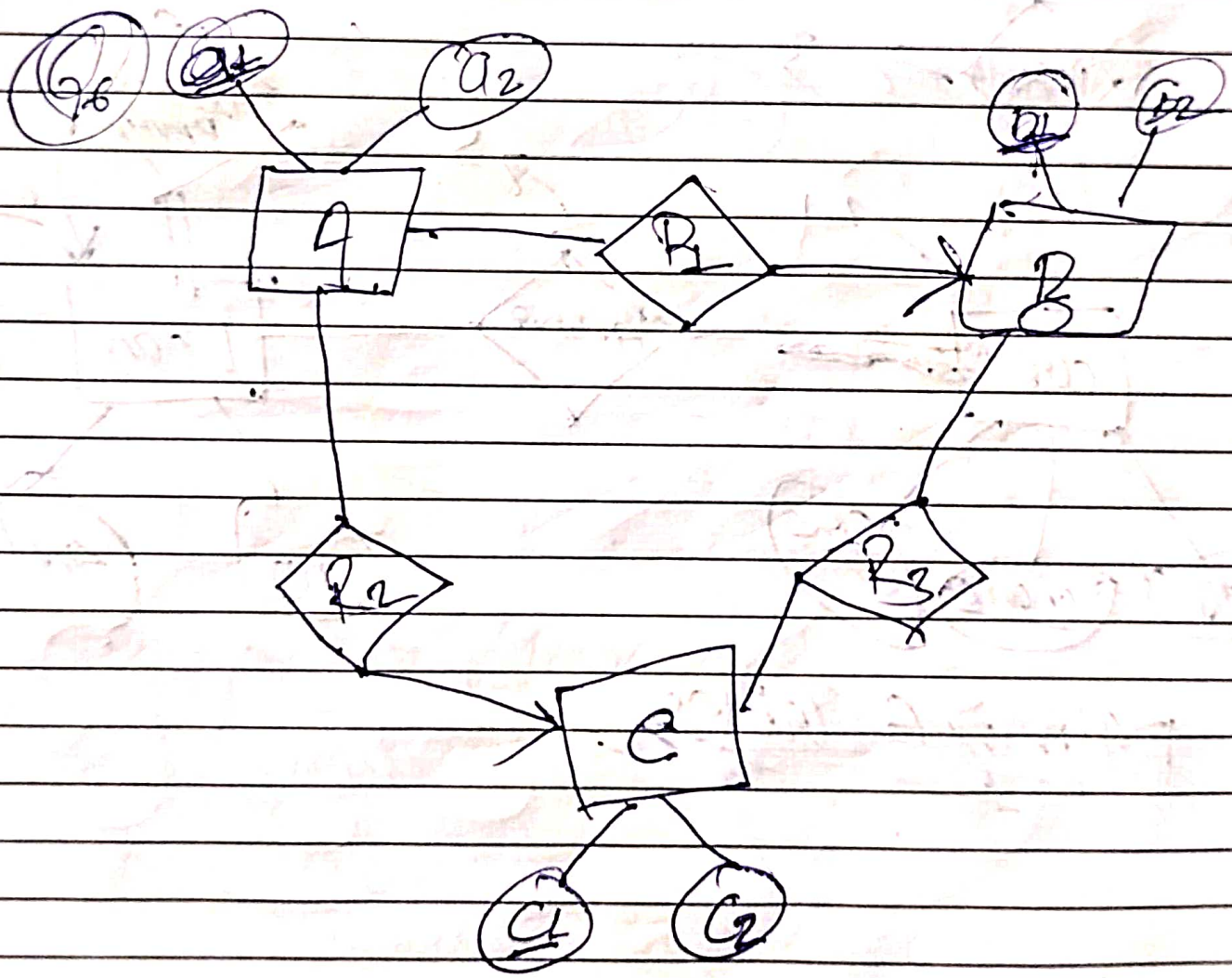
④ CR_3 (c_1, c_2, d_1)

⑤ D (d_1, d_2)

DATE: __/__/____
PAGE: _____



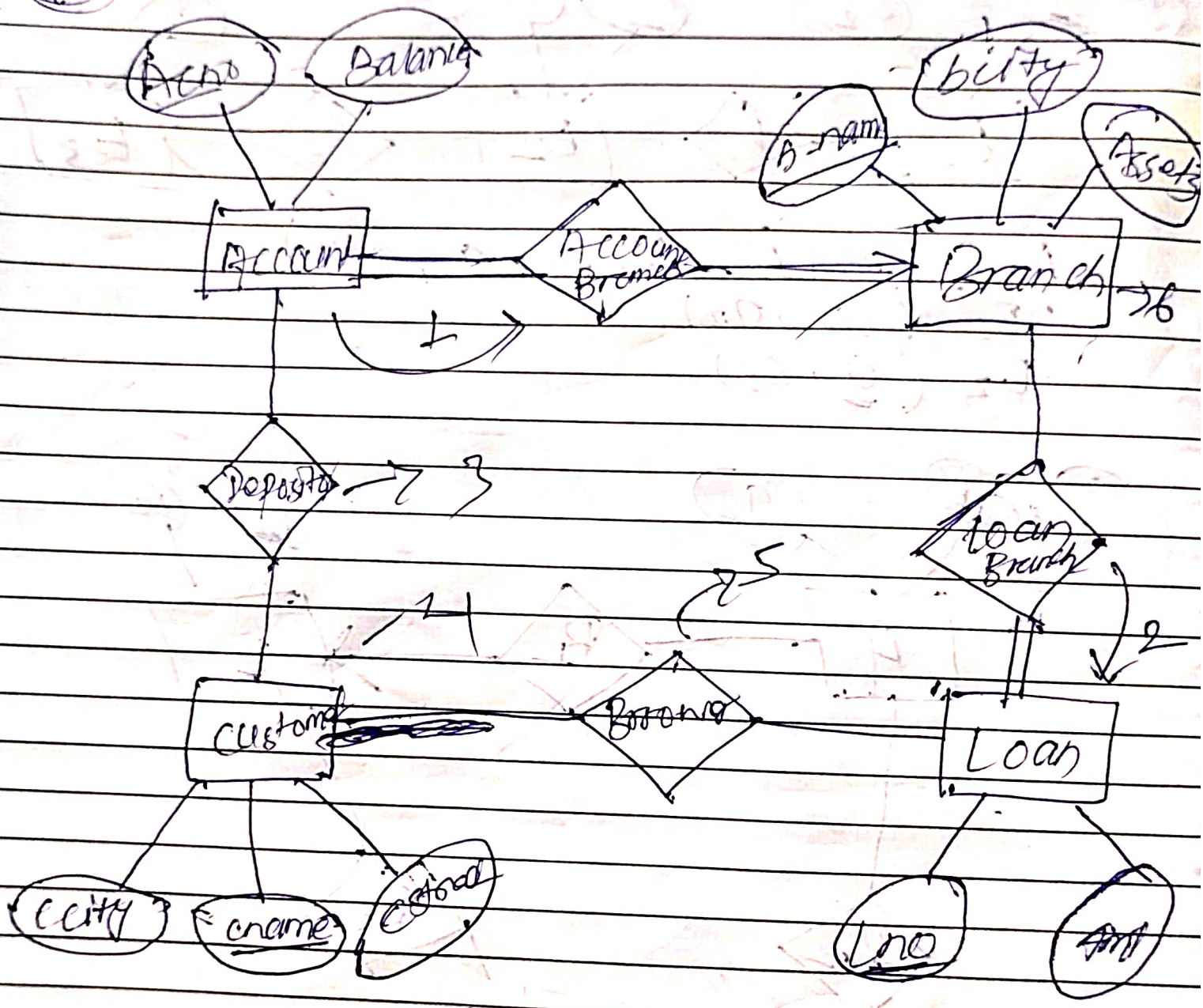
$E_2 R_1 R_2 (\underline{b_1}, b_2, b_3, a_1, c_1)$
$E_1 (a_1, a_2)$
$E_3 (\underline{c_1}, c_2)$



(1) $A R_1 R_2 (\underline{a_1}, a_2, b_1, c_1)$
(2) $C (\underline{c_1}, c_2)$
(3) $R_3 (\underline{c_1}, \underline{b_1})$, $B (\underline{b_1}, b_2)$

Good Write

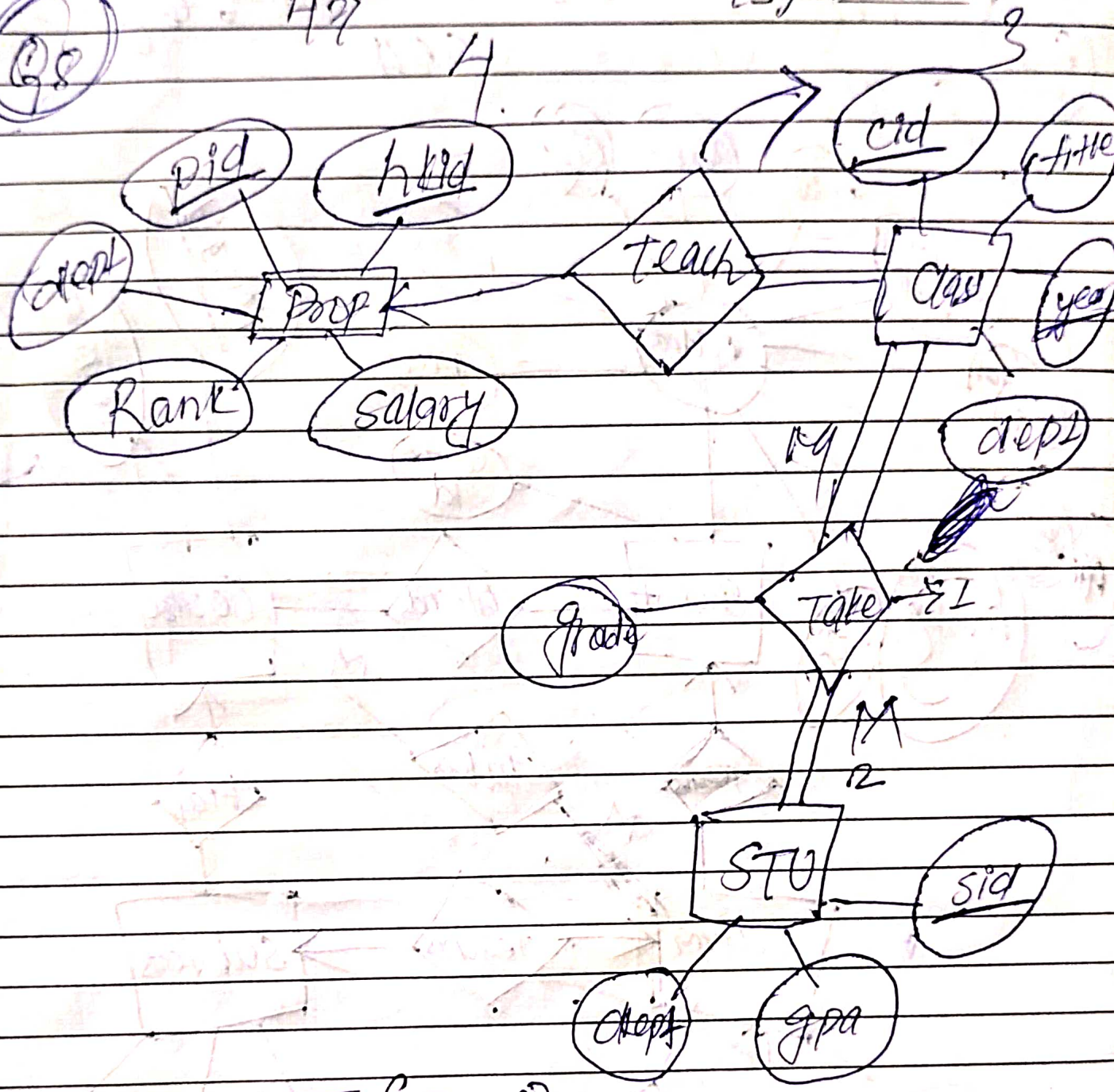
97



Main 7 6 Tables

3) 3 को attributes #
4 को लाएंगे F.K.

Q8



4) M मात्र (Prof)

3) class i.e. (Teach class) (Class की कोड + 3 को लाएंगे F.K)

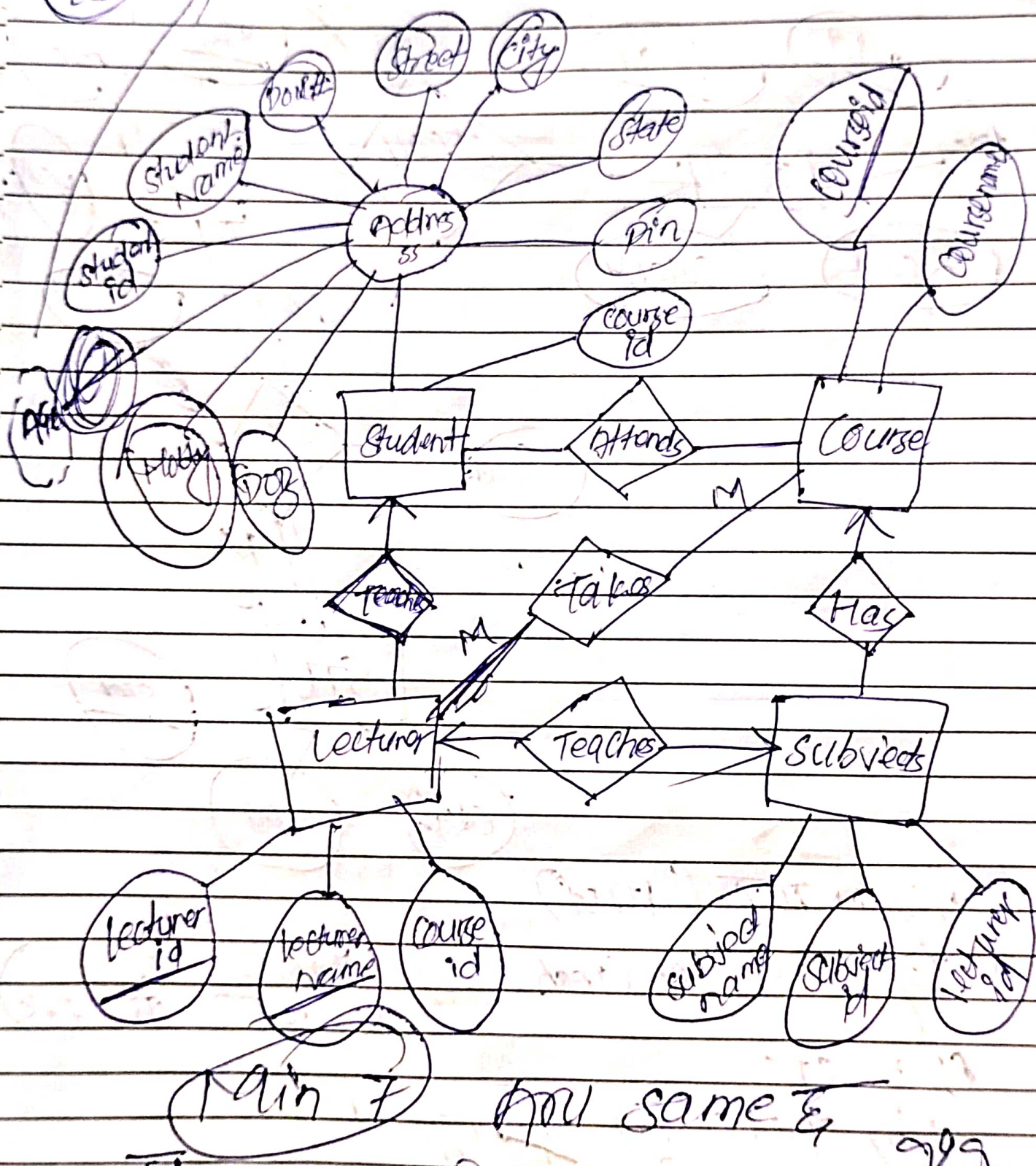
F2 → STO

1) Take(sid, ~~year~~, grade)

Derived attribute

Age > [illegible]

comp



Pin 7 [illegible] same है

⇒ तो Address [illegible] sab है [illegible] address [illegible] attributes

⇒ PTO

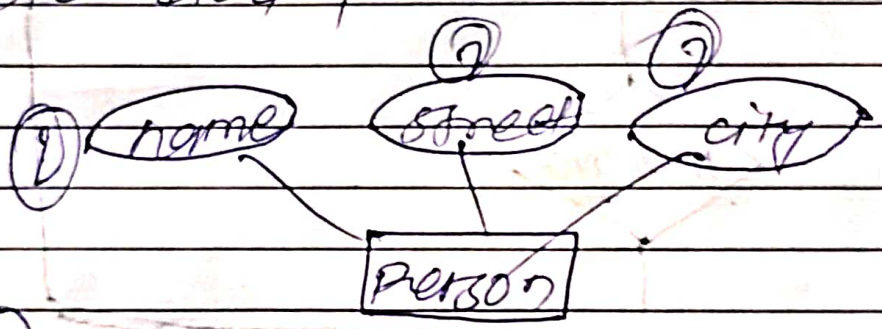
Good Write

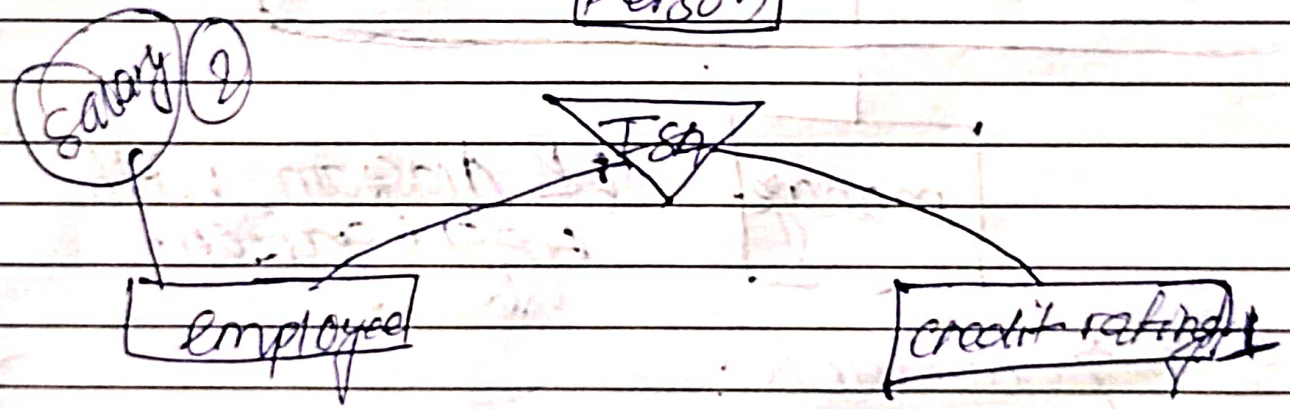
Multivalued Attribute

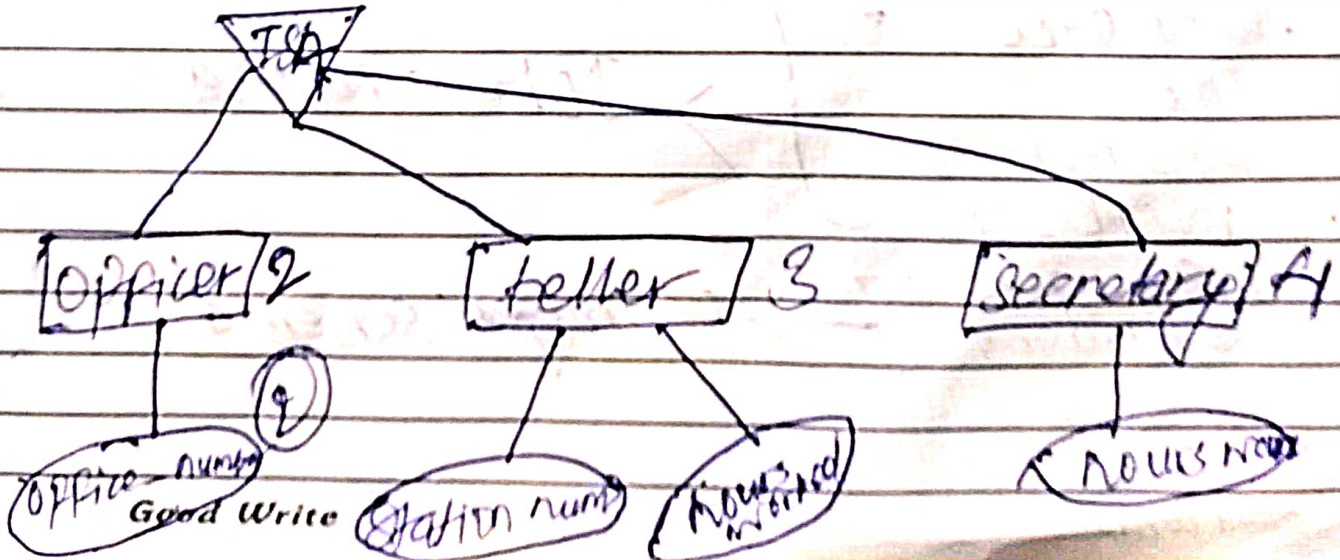
⇒ HOBBY (studentid, hobby)

(AGE) derived attribute है। इसलिए save नहीं रखते हैं।

Q10 :- Generalization and specialization को case में leaf nodes को मात्र table बनते।



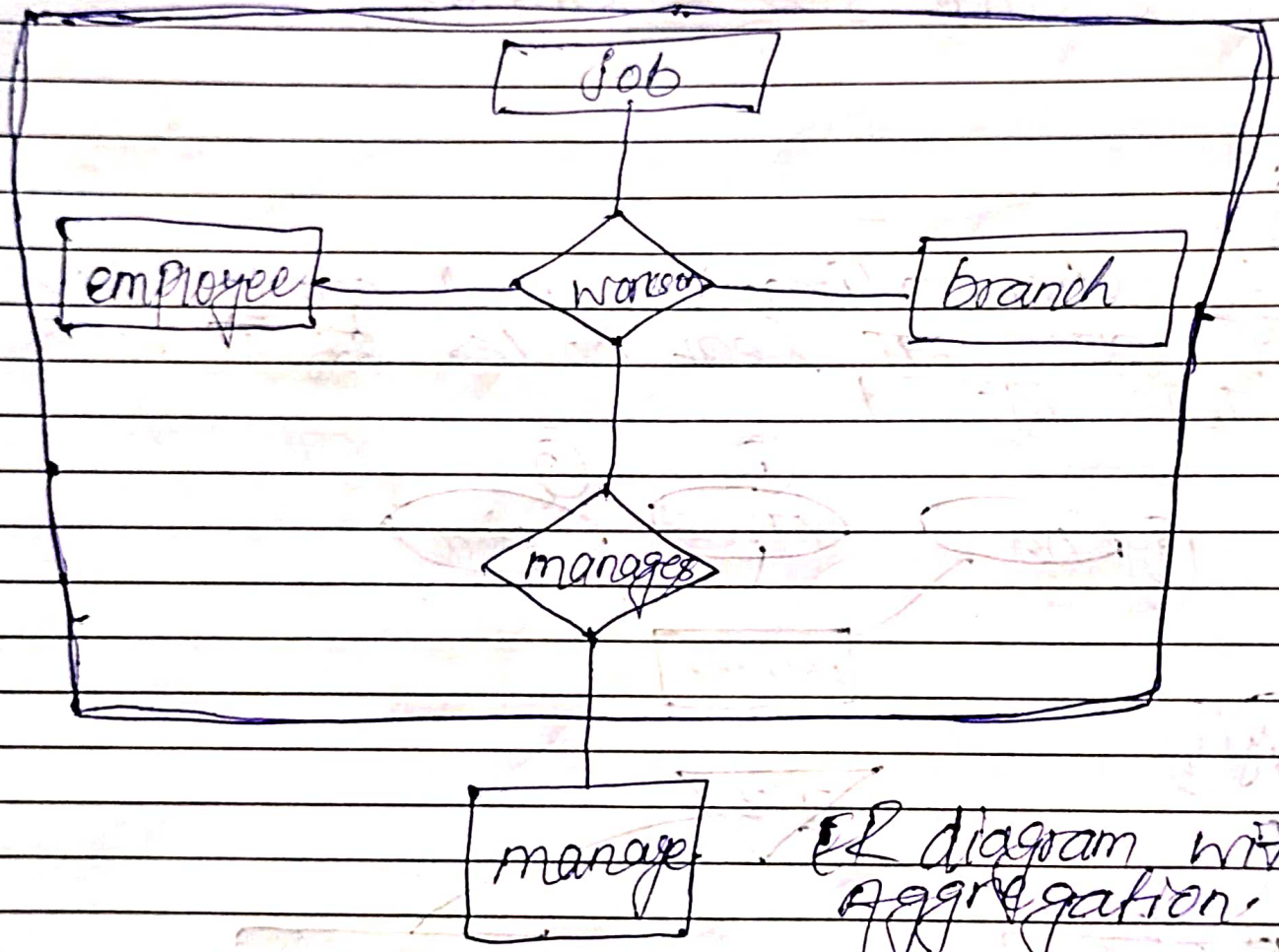




Good Write

??? Confussion

Q.10 Aggregation



ER diagram with Aggregation.

Min 7 6 Table.

1. employee
2. job
3. Branch
4. Works_on
5. Manager
6. Manages

→ Job schema

→ Manager schema.

ER Diagram for Banking System

DATE: __/__/____
PAGE: _____

Steps to draw an ER diagram:-

Identify:-

1. Entities
2. Attributes
3. Primary key
4. Relns and participation constraints.

SQL

DATE: __/__/____
PAGE: _____

SQL :-

↳ language to specify queries in structured manner.

structured माने relational data

↳ SQL is a language to specify queries in a relational database

⇒ mainly SQL भनेको computer language हो जसले store data, storing, manupulating and retrieving मा help गर्छ।

↳ access गर्ने (computer मा tabular form मा stored भएको data access गर्ने।)

SQL allows users to communicate with relational databases and retrive data from their tables.

SQL is standard language for RDBMS.

↳ MySQL, MS Access, Oracle, sab le SQL use गर्छ।

Particular domain का लागि यो language बन्यो. For Ex:- structural data. tabular

DATE: / / PAGE:

→ SQL is a domain specific language.

→ It is declarative. i.e. a non-procedural language??

→ SQL allows use to declare what we want to do but not how to do.

In SQL, we can specify through quires that what data we want but doesn't specify how to get those data.

How the query is actually performed in SQL?

→ It is done by data SQL engine. When we write an SQL queries, it first parses it; then it figures out its internal algorithms and it tries to select an algorithm which will be the best for that particular query. So, then it applies that algorithm, finds the answer and returns it.

→ SQL is based on Relational Algebra and Tuple Relational calculus.

DATE: __/__/__
PAGE: _____

SQL can do the following:-

- Execute queries against database.
- Retrive data from database
- Insert data in.
- Update records in.
- delete records in.
- Create new databases.
- Create new table in database
- Create views in a database
- Set permissions on tables, procedures and views.

SQL sub-languages:-

1) Data Defination language (DDL)
2) Data manupulation (DML)
3) Data control (DCL)
4) Transaction control (TCL)

Types of SQL commands.

DATE: / /
PAGE:

① Data Defination language (DDL) commands

↳ Are used to define the structure and schema of database.

CREATE → creates new table or database.

ALTER → to alter the structure of table. → alter/Modify

DROP → delete table from database.

TRUNCATE :→ to delete all the records from database.

RENAME :- to rename the table.

(CATRDC) comment

② DML (siu delmc)

delete, explain/plan, lock, merge, call

↳ These commands are used for accessing and manipulating the data record stored in the database.

SELECT :- to retrive data from.

INSERT :→ to insert new row in the table.

UPDATE :→ to update existing row in table.

DATE: __/__/____
PAGE: _____

DELETE :→ to delete a row from the table

DML operations ⇒ CRUD

C ⇒ create ⇒ INSERT

R ⇒ Read ⇒ SELECT

U ⇒ Update ⇒ UPDATE

D ⇒ Delete ⇒ DELETE

SELECT is also called Data Query Language (DQL)

→ Select को through database मा stored भएको data मा query गर्ने सक्छन।

(5) DCL

→ Mainly security को लागि use हुन्छ।

→ Commands to control the access of the data stored in the database

TCL commands are only used for DML comm while DDL and DCL comm are auto commited.

DATE: __/__/____
PAGE: _____

GRANT :- grant permission to user for database access.

REVOKE :- take back granted permission from user.

(4) TCL

↳ commands to manage the changes made by the DML statement.

COMMIT :→ permanently save the transaction.

ROLLBACK :→ to undo transaction.

SAVEPOINT :→ temporarily save the transaction so that you can Rollback to that point whenever necessary.

DATE: __/__/__
PAGE: _____

SQL Rules:-

- Not case sensetive (imp)
- (imp) But names of database, tables and columns are case sensetive
- Can use multiple lines but ends with semi-colon (;)

DDL

define garxa data लाई

- DDL is used for creating and modifying the database objects such as ~~creating the~~ tables, views and users.
- DDL commands are used define structure and schema of the database
- All the DDL commands are auto committed means permantely saves all the changes in the database

DATE: __/__/____
PAGE: _____

CREATE :-

↳ to create database schema and to define the type and structure of the data to be stored in the database

↳ can be used for creating a data base

↳ Tables too.

CREATE DATABASE ⇒ create database in RDMS.

syntax :-

CREATE DATABASE database_name

To check whether the data is created or not, we use SHOW statement.

imp **SHOW** :- To see existing databases and tables.

SHOW database.

SHOW tables.

USE :-

Too use or select any existing database.

DATE: __/__/__
PAGE: _____

Syntax:-

USE database_name

USE Daxi:-

CREATE TABLE:-

↳ create new table in database

It specifies the column names of the ~~da~~ table, its datatypes (eg:- VARCHAR, integer, date, etc) and can also specify integrity constraints (eg. Primary key, Foreign key, not null, Unique)

imp

Syntax:-

CREATE TABLE:-

(

column_name1 datatype,
column_name2 datatype,
|
column_namen datatype,
(Integrity constraints)

);

DATE: __/__/____
PAGE: _____

Examples :-

```
CREATE TABLE Emp(
    Emp-ID int,
    Name VARCHAR(20)
    Age Int
    Address VARCHAR(20)
    Salary numeric(10,2)
);
```

→ varchar(20)

```
CREATE TABLE branch(
    branch-name varchar(15),
    branch-city varchar(30),
    assests integer,
    Primary key (branch-name)
    Check(assets >0)
);
```

Table ko structure OR details देखने DEC or DESCRIBE statement use करते है।

Syntax :-

```
DEC table_name;
```

Ex :- DEC Emp;

DATE: __/__/____
PAGE: _____

* Each column in a database requires to have name and datatype.

SQL Constraints

→ Used to specify rules for the data in the table.

→ Used to limit the type of data that can go into the table.
→ Ensures the accuracy and reliability of the data in the table.

Constraints can be table level or column level

→ Column level constraints apply to column.
→ Table level to table

Constraints can be defined / specified:-

↳ When a table is created with the CREATE TABLE statement, OR

↳ we can use ALTER TABLE statement to create constrains even after the table is created.

DATE: __/__/____
PAGE: _____

NOT NULL :- column can't have null value

DEFAULT :- Provides default value for a column when none is specified

UNIQUE :- Ensures that all the values in a column are different/unique.

PRIMARY KEY :- A comb^n of NOT NULL and UNIQUE. Uniquely identifies each row in a table

FOREIGN KEY :- Uniquely indentifies a row/record in another table

CHECK :- Ensures that all the values in a column satisfies certain cond^n.

INDEX :- Used to create and retrive data from database very quickly

Ex :-
imp

```
CREATE TABLE Emp
(
    EmpID int NOT NULL,
    Name varchar(20) NOT NULL,
    Age int NOT NULL,
    Address varchar(100) DEFAULT 'India',
    salary numeric (10,2) NOT NULL,
    DeptID int
```

DATE: __/__/____
PAGE: _____

Primar Key (Emp-ID),
Foreign Key (Dept-ID) References Dep (Dep-ID)
Check (Age >= 18)

ALTER ALTER TALE

↳ modify the structure of the existing Table.

↳ Used to add, delete, modify or rename columns in existing table

↳ Used to add and drop various constraints on an existing table

1. ALTER TABLE - ADD COLUMN
   ↳ Adds new column in the table
2. ALTER TABL - DROP COLUMN
   ↳ Removes existing column from table
3. ALTER TABLE - MODIFY COLUMN
   ↳ modifies existing - - -
4. ALTER TABLE - RENAME COLUMN.

DATE: __/__/____
PAGE: _____

1) ALTER TABLE – ADD COLUMN

Syntax:-

```
ALTER TABLE table_name
ADD column_name(s) datatype(s);
```

multiple table add garna milxa

Ex:-

```
ALTER TABLE Emp
ADD Email varchar(100);
```

Email → column_name, varchar → datatype

2) ALTER TABLE – DROP COLUMN

Syntax:-

```
ALTER TABLE table_name
DROP column_name(s);
```

Ex:-

```
ALTER TABLE Emp
DROP Email;
```

3) ALTER TABLE – MODIFY COLUMN

Syntax:-

```
ALTER TABLE table_name
MODIFY column_name datatype;
```

Ex:-

```
ALTER TABLE Emp
MODIFY Name varchar(100);
```

DESC Command le changes rayo bina देख मिलेंगे।

DATE: __/__/____
PAGE: _____

4) ALTER TABLE - RENAME COLUMN

↳ Syntax :-

ALTER TABLE table-name
RENAME old column-name to
new column-name;

Ex :-

ALTER TABLE Emp;
RENAME Address TO Location;

DROP

↳ Completes removes table from database.

↳ This command will destroy the table structure and data stored in it.

Syntax :-

↳ DROP TABLE table-name;

Ex :-

DROP TABLE Emp;

DROP COMMAND CAN ALSO BE USED ON DATABASE.

DROP DATABASE :- Delets complete database

DATE: __/__/____
PAGE: _____

Syntax:-

DROP DATABASE database_name;

Ex:-

DROP DATABASE my_dp;

TRUN CATE (Truncate)

→ Used to remove all rows (complete data from Row is deleted) from a table. It is similar to DELETE Statement with no WHERE Clause.

TURNCATE VS DROP TABLE

→ Drop table command can also be used to completely delete the table but it deletes table structure too but TURNCATE TABLE doesn't delete the structure but deletes all the rows.

Syntax:-

TURNCATE table_name;

Ex:-

TURNCATE Emp;

~~the~~ Truncate deletes only records not the column names and their constraints.

DATE: __/__/____
PAGE: _____

Syntax:-

DROP DATABASE database_name;

Ex:-

DROP DATABASE my_dp;

TRUNCATE (Truncate)

→ Used to remove all rows (complete data from Row is deleted) from a table. It is similar to DELETE Statement with no WHERE Clause.

TURNCATE VS DROP TABLE

→ Drop table command can also be used to completely delete the table but it deletes table structure too but TURNCATE TABLE doesn't delete the structure but deletes all the rows.

Syntax:-

TURNCATE table_name;

Ex:-

TURNCATE Emp;

~~ate~~ Truncate deletes only records not the column names and their constraints.

To check the changes use the SHOW tables [SHOW TABLES command]:-

DATE:
PAGE:

RENAME TABLE

Used to change the name of the table.

Syntax:-
RENAME TABLE old_table_name TO new_table_name;

Ex:-
RENAME Emp to Employee;

Date .....

ddl → (cat rdc)

c → create, a → alter, t → truncate,
r → rename, d → drop, c → comment

SELECT → Also called the data query ~~language~~ language because used to query the database. (DQL)

Date .....

SQL DML Commands

DML :- (s i u del mc)

↳ Used for accessing and manipulating the data stored in database.

↳ These commands are not auto commited means they can be Rollback

SELECT :→ To retrive data from database

INSER :→ To insert a new row in the table.

UPDATE :→ Update the existing row.

DELETE :→ to delete a row from the table

CRUD

C → Create → INSERT
R → Read → SELECT
U → Update → UPDATE
D → Delete → DELETE

s → select, i → insert, u → update
d → delete, e → explanplan, l → lock table
m → merge, c → call

→ Mostly used SQL Query / SQL Comm

Date .....

SELECT

↳ It is used to select a set of data from a database table.

↳ SELECT is used to retrive data FROM database table.

↳ It returns data in the form of Result Table

↳ These result table are called Result-sets.

DQL because it used to query information from a database table.

It specifies Column names, and FROM specifies Table names

This command is used with different Conditions and CLAUSES

यसको मतलब specific column हुन सक्छ। Ex:- Name, Age....

Date .....

Basic Syntax :-

To retrive selected fields from the table:-

```
SELECT column1, column2 ---- ColumnN
FROM table-name(s);
```

To retrive all the fields from the table-

```
SELECT *
FROM table-name(s);
```

→ यसले sabbai field ko mean table ले हुन सक्छ।

→ multi table select गर्यो भने दुवैको cartesian product मिलाएर show हुन्छ। र सबै table show हुन्छ।

Ex 1:-

To retrive column Emp-ID, Name from Emp Table

```
SELECT Emp-ID, Name
FROM Emp;
```

Date .....

Ex2:-

To retrive all the fields from the Emp Table

```
SELECT *
FROM Emp;
```

WHERE CLAUSE

↳ Used to select some particular record based on the condition. It is used to filter Records.

↳ It is used to specify a condition while fetching the data from a single table or joining with multiple condition

↳ NOT only Used in SELECT but in UPDATE and DELETE statement.

Syntax:-

SELECT with WHERE :-

```
SELECT column1, Colum2, ---
FROM Table_name
WHERE condition;
```

Date .....

Ex:-

To retrive Emp-ID, Name from Emp table whose salary is greater than 2000

⇓

```
SELECT Emp-ID, Name
FROM Emp
WHERE salary > 2000;
```

BASIC SQL STRUCTURE

SQL is based on set and Relational operations with certain modifications and enhancements.

A typical SQL Query has the Form:

```
SELECT A1, A2, --- An
FROM r1, r2, ---- rm
WHERE P;
```

Ai's Represents set

ri's Represents Relations

P is a Predicate (or condn)

Spiral

Projection (π), selection (σ)

Date .....

The above SQL query is equivalent to the relation algebra expression.

$$\boxed{\pi_{A_1, A_2, A_n}(\sigma_p(r_1 \times r_2 \times \cdots \times r_m))}$$

The result of an SQL query is a relation (or table)

Optional CLAUSES in SELECT statement

1) WHERE clause :→ Specifies which rows to retrieve by specifying conditions.

2) GROUP BY clause :- organize similiar data into groups
Groups rows that share a property so that the aggregate function can be applied to each group.

3) HAVING :- Selects among the groups defined by the GROUP BY clause by specifying condns.

Spiral

Where 1st → Group By sequence → Having 3rd → Order 4th

Date .....

4) ORDER BY :-

It specifies the order in which to return the rows.

DISTINCT :-

↳ It is used to remove duplicates from results set of a SELECT statement. (SELECT DISTINCT)

→ Distinct clause select सँगै हुन्छ।

Generally select clause ले duplicate value पनि show गर्छ तर हामीले duplicate सँगै select गरेमा duplicate value हटाउँछ।

SQL Operators

→ Select statement सँगै हामीले धेरै operators पनि Use गर्छौं Which are:

1. Arithmetic
2. Comparison
3. Logical
4. Special

Date .....

ARITHMETIC

+, -, *, /, % → modulo

Ex:-

To retriere Emp-ID, Name, Salary Plus 3000 from Emp table

⇓

SELECT Emp-ID, Name, Salary +3000
FROM Emp;

→ पहिले को हामको table बाट यो Sab Emp-ID, Name, Salary show गर्दै तर Salary चाहि 3000 + गरेर imean पहिले भएका 3000 थपेर देखाउछ।

↓
+

COMPARISION

not equal to

=, <, >, >=, <=, <> or !=

↳ Basically where clause सँग add गरेर comparision operator use गर्दछ

Ex:-

SELECT Emp-ID, Name, salary
FROM Emp
WHERE Salary > 2000;

Spiral

Date .....

LOGICAL

AND ⇒ It displays a record if all the condns seperated by AND are TRUE.

OR ⇒ It displays a record if any of the conditions seperated by OR is TRUE

NOT ⇒ Displays a record if the conditions is NOT TRUE

- It reverses the meaning of any operator with which it is used.
- This is a negate operator.
- Eg :- NOT EXISTS, NOT BETWEEN, NOT IN, IS NOT NULL etc.

Ex:-

```
SELECT*
FROM Emp
WHERE salary > 2000 AND Age < 25;
```

Retrives all the records from Emp table whose salary is > than 2000 and age is less than 25.

Spiral

Date .....

SPECIAL

ALL ⇒ Compares a value to all the values in another value set (in subqueries)

ANY ⇒ compares values in the list acc to the condn (in subqueries)

BETWEEN ⇒ Used to search for values that are within a set of values (given min and max values).

EXISTS ⇒ Used to search for presence of a row in a specified table (in subqueries)

IN ⇒ Compares a value to that specified list value (and in subquery)

LIKE ⇒ Compares a value to similar values using wildcard operators (%, _).
underscore

IS NULL ⇒ Checks for missing data or NULL values.

Date .....

Ex:-

```
SELECT *
FROM Emp
WHERE Salary BETWEEN 2000 AND 5000;
```

INSERT (INSERT INTO)

↳ to insert new records in the table.

2 तरिकाले statement लेख्न Possible छ।

Syntax 1:-

↳ Specify both the column names and the values to be inserted.

```
INSERT INTO table_name(column1, column2
VALUES( value1 , value2 , ~~valu~~ ---);
```

Syntax 2:-

↳ Specify only values. But needed to member the column order.

```
INSERT INTO table_name
VALUES (value1, value2 ---);
```

Spiral

Date .....

Ex:-

INSERT INTO Emp (Emp_ID, Name, Age, Address, Salary)
VALUES (1, "Rahul", 32, 'Delhi', 2000);

OR

INSERT INTO Emp
VALUES (1, 'Rahul', 32, 'Delhi', 2000);

→ Column को order अनुसार।

⇓

Insert हुआ कि नहीं अनेर
Check गर्न

⇓

SELECT *
FROM Emp;

Date .....

UPDATE

↳ Used to modify the existing records in the table

NOTE ▷:
Always use the WHERE clause with the UPDATE statement to update the selected rows, otherwise all the rows would be affected.

Syntax :-

```
UPDATE table_name
SET column1 = value1, Column2 = Value2
    - - - -
WHERE Condition;
```

↳ Where Imp &

Example :-
↳ To update the address of Emp-ID 3 to Chennai :-
⇓

```
UPDATE Emp
SET Address = 'Chennai'
WHERE Emp-ID = 3;
```

DELETE & UPDATE मध्ये पहिले पण WHERE-

Date .....

DELETE

↳ to delete the existing record in a table.

WHERE नसे all the records would be deleted.

Syntax:

```
DELETE FROM table-name
WHERE Cond^n;
```

Ex:-

To delete the employee records of pune location.

⇓

```
DELETE FROM Emp
WHERE Address = 'Pune';
```

⇓

Check करूं

```
SELECT*
FROM Emp;
```

SELECT DISTINCT Address
FROM Emp;

Date .....

SELECT DISTINCT

Select distinct is used to return only distinct (different) values OR simply used to return only unique values.

Syntax:-

```
SELECT DISTINCT column1, column2--
FROM table_name;
```

TO COUNT DISTINCT VALUES

Ex:-
To show number of distinct (different) addresses in Emp table

```
SELECT COUNT(DISTINCT Address)
FROM Emp;
```

```
SELECT COUNT(DISINCT ---
FROM ---;
```

Spiral

Date .....

SELECT DISNTICT

↳ One or more than one columns select गर्या छ भने unique combinations for the columns.

↳ NULL value छ भने त्यसलाई ignore गर्दैन, result will include NULL as a distinct value.

AND, OR, NOT

↳ The WHERE clause can be combined with AND, OR, NOT operatos
↳ to make precise condns for fetching data from database by combing more than one condn together.

AND र OR पहिले पढेको same

NOT condn displays the record if the condn is not true.

NOT reverses the meaning of any operator with which it is used.

NOT ⇒ Negate operator.

Spiral

Date .....

Eg:- NOT IN, NOT BETWEEN, NOT EXISTS, IS NOT NULL, etc.

AND OPERATOR

↳ Used to combine multiple conditions with where clause, in ~~sele~~ SELECT, UPDATE or DELETE statements.

↳ यदि all condns must be satisfied.

Syntax:-

```
SELECT column1, column2 ----
FROM table_name
WHERE cond^n1 AND cond^n2 AND cond^n3 ---
```

Ex:-

```
SELECT*
FROM Emp
WHERE salary >2000 AND Age<25;
```

Spiral

Date .....

OR OPERATOR

→ combine multiple conditions with WHERE clause in SELECT, UPDATE ~~and~~ or DELETE statements.

Syntax:-

```
SELECT column1, column2 - - - - Column
FROM table-name
WHERE condition1 OR condition2 - - - -
```

Ex:-

```
SELECT salary
FROM Emp
WHERE Salary >2000 OR Age >25;
```

NOT Operator

→ Negate the condⁿ with the WHERE clause

→ Displays record only when the condⁿ is not TRUE.

Eg:- NOT IN, NOT BETWEEN, NOT EXISTS, IS NOT NULL, etc

NOT reverses the meaning of any operator with which it is used.

→ This is a negate operator.

Spiral

Date .....

Syntax:-

```
SELECT column1, column2, --- ColumnN
FROM table_name
WHERE NOT condition;
```

Ex:-

```
SELECT*
FROM Emp
WHERE NOT Address='Mumbai';
```

⇓

Selects all the records from Emp table which address is not Mumbai.

Combing AND, OR Together

→ The SQL AND condition and OR condition can be combined to test for multiple conditions WHERE clause

→ When combining these condns, it is important to use parenthesis ().

⇓

Parenthesis determine the order that the AND and OR conditions are evaluated.

✓ Precedence order NOT, AND ~~FOR~~ Special

Precedence → NOT, AND, OR

Date .....

Ex:-

```
SELECT*
FROM Emp
WHERE (salary>2000 AND Age<25)
      OR (EmpID=2);
```

IN OPERATOR

→ Allows us to specify mutiple values in WHERE clause

→ IN is a shorthand for multiple OR conditions. So it used to replace many OR conditions.

Syntax:-

```
SELECT colum-name(s)
FROM table-name(s)
WHERE column-name IN (values,2,3...)
```

Subquery :-

A subquery is a select-from-where expression that is nested within another query.

Subquery :- select-from-where expression that is nested ~~to~~ within another query.

Date .....

OR (in Subqueries)

inner select statement.

```
SELECT column_name
FROM table_name
WHERE column_name IN (SELECT STATEM
```

Ex :-

```
SELECT *
FROM Emp
WHERE Address IN ('Delhi', 'Bilaspur',
                  'patna');
```

Selects all Employees located in Delhi, Bilaspur and Patna.

Ex :-

```
SELECT *
FROM Emp
WHERE Address IN (SELECT Address
                  FROM customers);
```

Selects all the employees that are from the same address as the customers.

Date .....

NOT IN

→ Just opposite of IN operator.

Syntax :-

```
SELECT column-name(s)
FROM table-name
WHERE column-name NOT IN (value1 ---)
```

Example :-

```
SELECT *
FROM Emp
WHERE Address NOT IN ('Delhi', 'Bilaspur',
                      'Patna')
```

⇓

It selects all Employees not located in Delhi, Bilaspur and patna.

BETWEEN

Between ___ And is used to select values within given range. (given minimum and maximum value)

Date .....

The values can be numbers, text or dates.

→ Inclusive :- begin and end values are included.

Syntax :-

```
SELECT column-name(s)
FROM table-name
WHERE column-name BETWEEN value1
      AND value2;
```

Ex:-

```
SELECT*
FROM Emp
WHERE salary BETWEEN 2000 AND
      7000;
```

NOT BETWEEN

Syntax:-

```
SELECT column-name(s)
FROM table-name
~~NOT BETWEE~~
WHERE column-name NOT BETWEEN
      value1 and value2;
```

Date .....

LIKE OPERATOR

↳ It is used in a WHERE clause to search for a specified pattern in a column

↳ LIKE operator is used to match string pattern values using two wildcard characters.

Wild Card Character:-

↳ It is used to substitute one or more characters in a string.

2 wildcard characters used.

% ⇒ zero, one or multiple characters

_ ⇒ Represents only a single character.

(%) and (_) can also be used in combinations.

Syntax:-

```
SELECT column1, column2, ---
FROM table-name
WHERE columnN LIKE pattern;
```

Spiral

Date .....

Ex1:-

```
SELECT*
FROM Emp
WHERE NAME LIKE "K%";
```

⇓

Slects all the records ~~wh~~ from employee table whose name starts with K.

Ex2:-

```
SELECT*
FROM Emp
WHERE NAME LIKE "_a%";
```

Ex3:-

```
SELECT*
FROM Emp      LIKE
WHERE NAME "M____%";
```

starts with M and atleast 4 characters

Ex4:-

```
SELECT*
FROM Emp
WHERE salary LIKE "200%";
```

salary starts with 200....

LIKE

'A-I-N' → starts with A and ends with N.

-I-oh-I- → oh in any position

Spiral

→ missing, unknown, non-existent (just like people who truly care about you :))

Date .....

NULL values are different than a zero value or a field that contains space

→ Result of any arithmetic operations or expression involving null is null.

→ Result of where clause condition is False if it evaluates to null.

→ AND, OR and NOT operators handles null as follows:-

↳ missing, non-existent, unknown, non-applicable (just like people who love you)

AND

And	true	False	null
→ true	true	False	null
→ null	null		
→ False			

true AND null → null
null AND null → null
False AND null → null
False AND False → False

OR

True OR NULL → True
null OR NULL → NULL
False OR NULL → NULL

Spiral

Date .....

NOT

NOT True ⇒ False
NOT False ⇒ True
NOT NULL ⇒ NULL

HOW to Test NULL values??

↳ comparision operator like =, <, <> can't be used to test NULL values so,

SQL Uses IS NULL and IS NOT NULL operators instead.

IS NULL :-

↳ Allows us to find set of records in which value for a particular column is NULL.

Returns TRUE if a NULL value is found, otherwise returns FALSE.

SELECT, UPDATE or DELETE statement with WHERE clause.

Date .....

IS NULL Syntax :-

```
SELECT column-name
FROM table-name
WHERE Column-name IS NULL;
```

IS NOT NULL

Opposite of IS NULL and this operator checks for NOT NULL values (or non-empty values).

Returns TRUE if ~~not~~ null value is not found ~~and~~ otherwise it returns FALSE

IS NOT NULL Syntax :-

```
SELECT column-name
FROM table-name
WHERE column-name IS NOT NULL;
```

SQL ALIASES

SQL Aliases (ALIASES) can be used to create a temporary name for colums or tables.

Date .....

- Aliases are create to make table or column easier to read.

- An alias only exists for the duration of the query. i.e. The renaming is just a temporary change and the actual table name doesn't change in database.

- Preffered when there are more than one table involved in a query.

Syntax For Aliases

Column Aliases

```
SELECT column-name As Alias-name
FROM table-name;
```

TABLE Aliases

```
SELECT column-name(s)
FROM table-name As Alias-name
```

Date .....

Ex :

SELECT Emp-ID as ID, Name
FROM Emp;

Emp table की 2 वटा ~~table~~ Column देखाउ, Emp-ID र Name
↳ Emp-ID के ID मा temporarily changed भएर देखिन्छ।

Table Alias Example

SELECT E.Email-ID, E.Name, D.Dept-ID, D.D-Name

FROM Emp As E, Dept As D

WHERE E.Dept-ID = D.Dept-ID;

imp

Date .....

AGGREGATE FUNCTIONS

Aggregate and scalar functions bothe return a single value.

Diff
Aggregrate functions operate on many records but scalar works on each record independently.

Aggregate Functions :-

COUNT()
SUM ()
AVG ()
MAX ()
MIN ()

→ Sabले work गर्ने group of data मा आएर single value return गर्ने।

These are often used by GROUP By clause of the SELECT statement to group multiple rows together as input to form a single value output.

IGnores NULL values expects COUNT(*)

Spiral

IGNORES NULL VALUE → ???

Date .....

COUNT ()

↳ Counts no. of rows that matches the specific condition.

Syntax :-

```
SELECT COUNT (column_name)
FROM table_name
WHERE condition;
```

Ex:-

```
SELECT COUNT(*)
FROM Student;
```

→ column name यदि ऐसे [count(*)]

COUNT (DISTINCT)

↳ Distinct value count करे

SUM ()

syntax:-

```
SELECT SUM (column_name)
FROM table_name
WHERE condition;
```

Date .....

Ex:-

20+22+20+20+60+22

```
SELECT SUM (Marks)
FROM student;
```

] sums duplicate marks as well

20+22+60

```
SELECT SUM (DISTINCT Marks)
FROM student;
```

Finds the sum of distinct marks from student Table.

Renaming the column

```
SELECT SUM (Marks) AS 'Sum Marks'
FROM student;
```

without as	using as
sum (marks)	summarks

AVG ()

Syntax:-

```
SELECT AVG (column_name)
FROM table_name
WHERE condition;
```

Ex:-

```
SELECT AVG (marks)
FROM student;
```

AVG (Marks) = SUM (Marks) / COUNT (Marks)

Spiral

SUM (DISTINCT Marks) / COUNT (DISTINCT Marks

Date .....

DISTINCT AVERAGE

```
SELECT AVG (DISTINCT marks)
FROM student;
```

RENAMING:

```
SELECT AVG (Marks) AS ~~SELEC~~ 'Average marks
FROM student;
```

MAX ()

Syntax:-

```
SELECT MAX (column-name)
FROM table-name
WHERE condition;
```

MIN ()

Syntax:-

```
SELECT (column-name)
FROM table-name
WHERE condition;
```

Spiral

Date .....

SCALAR FUNCTIONS

↳ operates on each record independently

- UCASE()
- LCASE()
- MID()
- LENGTH()
- ROUND()
- NOW()
- FORMAT()

UCASE ()

Syntax:-

```
SELECT UCASE (Name)
FROM Emp;
```

MID () ⇒ SUBSTR

↳ Used to extract substrings from column values of string type in (a table)

Syntax:-

```
SELECT MID(column_name, start, length)
FROM table_name;
```

Note:- length is optional.

start ⇒ starting position of string.

Date .....

Example MID()

→ Extract sub-string from column name ~~string~~ from 2nd position having length of 3.

~~SELECT MID (string 2, 3~~

SELECT MID (Name, 2, 3)
FROM Emp;

start, length

Rahul
→ ahu

LENGTH() OR LEN()

Syntax:-

```
SELECT LENGTH(column_name)
FROM table_name;
```

Ex:-

```
SELECT LENGTH(Name)
FROM Emp;
```

ROUND() गोल करने के लिए

→ Used to round a numeric column to the number of decimals specified.

Note :- Decimals represents no. of decimals to be fetched.

no. of decimal to be fetched

Date

Syntax (ROUND())

```
SELECT ROUND (column_name, decimals)
FROM table_name;
```

decimal parikhat'o ta digit

Example :- → ROUND()

```
SELECT ~~salary~~ ROUND (salary)
FROM Emp;
```

OR

```
SELECT ROUND (salary, 0)
FROM Emp;
```

यसलाई के गर्दा माथि salary column को हरेक value लाई Round off गरेर विन्छ (0 को ठाउँ मा 1 हुन्थ्यो भने

पहिले		पछि	
2000.50	→	2001	2000.5
1500.20	→	1500	1500.2
2000.00	→	2000	2000.0
6500.80	→	6501	6500.8
8500.00	→	8500	8500.0
4500.00	→	4500	4500.0
10000.00	→	10000	10000.0

* पछि केही

means sab values round off भयो हरएक हरेक लाई round off ~~गरेको~~ को।

SELECT Name, NOW() AS 'Date and Time'
FROM Emp; ⇒ FOR COLUMN

Date .....

NOW()

↳ Returns the current system date and time

Syntax:-

```
SELECT NOW()
FROM table_name;
```

Fetching current system date and time in a column:-

```
SELECT Name, NOW() AS 'Date and Time'
FROM Emp;
```

```
SELECT NOW();
```

↓

NOW()
Year/dd/mm time

FORMAT() → to format how column is to be displayed

```
SELECT FORMAT(column_name, format)
FROM table_name;
```

eg: SELECT FORMAT(NOW(), 'YYYY-MM-DD')
FROM Emp;

Date .....

FORMAT()

→ Format function is used to format how a column is to be displayed.

Syntax FORMAT Function:

Format माने कुन तरिकाले show गर्ने।

→ कुन column लाई Format गर्ने।

→ SELECT FORMAT (column_name, Format)
FROM table_name;

Example FORMAT()

→ Formatting current date as 'YYYY-MM-DD'

SELECT Name, FORMAT (NOW(), 'YYYY-MM-DD')
AS Date
FROM Emp;

⇓

NOW() ले current date and time दिन्छ

AS 'Date' ⇒ त्यो नयाँ बन्न आएको column को नाम Date मा change गर्छ।

Date .....

SQL ORDER BY CLAUSE

Generally हामीले SELECT query clause use गर्छ specific order मा Result देखिन्न त्यसैले ORDER BY use हुन्छ।

⇓

The ORDER BY CLAUSE allows us to specify order for the query output

ORDER BY clause is used with SELECT statement for arranging retrieved data in sorted order.

ORDER BY

→ ~~यसको~~ यसले by default ascending order मा sort गर्छ।

⇓

[ASC]

Descending को DESC keyword word with ORDER BY clause.

Date .....

Syntax :-

```
SELECT column-list
FROM table-name
[WHERE conditions]
ORDER BY column1, column2, ---column N
```

यो optional → based on the condn of output

→ [ASC] [DESC]

Example 1 :-

```
SELECT*
FROM Emp
ORDER BY salary;
```

→ यसले salary को हिसाबले assending order मा output show गर्छ।

→ जसको salary सबै भन्दा थोरै त्यो माथी अरु तल तल तल।

Example 2 :-

To sort details of Emp table in ascending order by name and salary.

```
SELECT*
FROM Emp
ORDER BY Name, Salary;
```

1st column ⇒ Primary Ordering Field

2nd column ⇒ Secondary Ordering Field, and so on.

Date .....

Suppose दुई जनाको Name same छ तर ordering way मा output निकाल्नु राख्न चाहिं Name column यदि कुनै column ORDER BY clause मा Use गरिराख्ने त्यसको ordering हुन्छ।

Example 3:-

To sort details of Emp table in descending order by name.

```
SELECT*
FROM Emp
ORDER BY ~~NAME~~ Name DESC;
```

GROUP BY

Used for organizing similar data into groups.

For Eg:- "Find the no. of customers in each country".

GROUP BY stament is often used with aggrograte functions to group the result set by one or more columns.

SELECT → FROM → WHERE
↳ GROUPBY → HAVING → ORDERBY

Date .....

⇒ Placed with the select statement in SQL query

⇒ After : Where

⇒ before order by.

Syntax:-

SELECT column-name(s), aggregate function (column-name)
FROM table-name
[WHERE condition]
GROUP BY column-name(s)
[ORDER BY column-name(s)]

जुन column Name हामीले GROUP BY मा दिएका, त्यहि column Name हामीले select मा दिने।

Example 1:-

↳ To Find the no. of employee in each department :-

```
SELECT Dept_id count(*)
FROM Emp
GROUP BY Dept_id;
```

Spiral

Date .....

Output:-

Dept_ID	count(*)
500	3
501	2
502	2

Example 2:-
GROUP BY सेंट ORDER BY

```
SELECT Dept_id, Count(*)
FROM Emp
GROUP By Dept_id
ORDER BY count(*);
```

→ count(*) वाला column को order by करेगा।

⇓

Dept_Id	count(*)
501	2
502	2
500	3

Example 3:-
GROUP BY with SUM
Write a query to find the sum of salaries of employees from each department

Date .....

```
SELECT Dept-id SUM(salary) as 'Sumsalary'
FROM Emp
GROUP BY Dept-id;
```

Dept-id	Sum salary	
500	5500.00	⇒ 500 को सब (3) बता
501	15000.00	⇒ 501 को सब (2)
502	14500.00	⇒ 502 को सब (2) बता दे

Example 4:-

```
SELECT Dept-id, AVG(salary) AS 'Avg salary'
FROM Emp
GROUP BY Dept-id;
```

Example 4:-

```
SELECT Dept-id, MIN(salary) AS 'MIN Salary'
FROM Emp
GROUP BY Dept-id;
```

Ex 5:-

```
SELECT Dept-id, MIN(salary), Max(salary)
FROM Emp
GROUP BY Dept-id;
```

Spiral

GROUP BY + subquireies

Date .....

Dept_id	Min(salary)	Max(salary)	
500			
501			
502			

Imp

Ex 8:- GROUP BY Multiple columns

→ Write a query to find the number of employees in each department location wise.

SELECT Dept_id, Address, Count(*)
FROM Emp
GROUP BY Dept_id, Address;

Dept_id	Address	count(*)
500	Delhi	1
500	Pune	2
501	Mumbai	2
502	Bilaspur	1
502	Patna	1

1st मा Dept_id अनुसार Group बनाउछ त्यसपछि त्यो Dept_id को corresponding Address check गर्छ Same छ सब Dept_id को corresponding address भाने 1 Row मात्र छुट्टा छुट्टै।

500 ⇒ Delhi मा (1)
500 ⇒ Pune मा (2).
502 Bilaspur (1)
502 Patna (1)
501 Mumbai (2)

Spiral

Difference Betn Having and WHERE

Date .....

HAVING Clause

- Used with groupby clause and filter the groups created by Groupby clause.

- Having clause enables us to specify condn that filter which group results appear in the results.

- Having clause was added to SQL because the aggrograte functions like SUM, AVG, MIN, MAX, COUNT cannot be used with WHERE clause.

- The WHERE clause places conditions on the selected columns where as HAVING clause places conditions on groups created by the GROUP BY clause.

Syntax:-

```
SELECT column-name, aggregat function(colm
FROM table-name
[WHERE condn]
GROUP BY column-name
HAVING conding
[ORDER BY column-name(s)]
```

Select → from → where → Group by
→ Having → Order by.

Date .....

Ex 1:-

Write a query to display departments where the no. of employee is greater than 2

⇓

```
SELECT Dept-id, count(*)
FROM Emp
GROUP By Dept-id
Having (count(*) >2);
```

Ex 2:-

To find the department for which the sum of salaries of employees is more than 1000.

```
SELECT Dept-id, Salary, SUM(salary)
FROM Emp
GROUP BY Dept-id, Salary
HAVING (SUM (salary) > 1000;
```

Date .....

Example 3:

To find the sum of salaries of employee that lives at same location

```
SELECT Address, Sum (salary)
FROM Emp
GROUP BY Address
HAVING COUNT (Address) >=2;
```