tiny.cc/DPUBRACU

Attendance — 5%.

Assignment — 10%.

Quiz — 10%.

Midterm — 15%.

Final — 30%.

Project — 20%.

Lab — 10%.

LECTURE 1

THURSDAY DATE: 26/01/23

CHAPTER 1

INTRODUCTION: DATABASES AND DATABASE USER

CHAPTER - 1

BASIC DEFINITIONS:

DATA: Known facts that can be recorded and have an implicit meaning.

DATABASE: A collection of related data.

DATABASE MANAGEMENT SYSTEM (DBMS): A software package/system to facilitate the creation and maintenance of a computerized database.

DATABASE SYSTEM: The database software together with the data itself.

Database
- we create tables which are connected

CSE370

ID	
1001	A

CSE331

ID	
1001	A

CSE421

ID	
1001	A-

* transaction: data is successfully inserted & stored
* Database management system: eg: windows facilitating to use chrome. window is a software.

Database system: software + data

MANAGING DATA

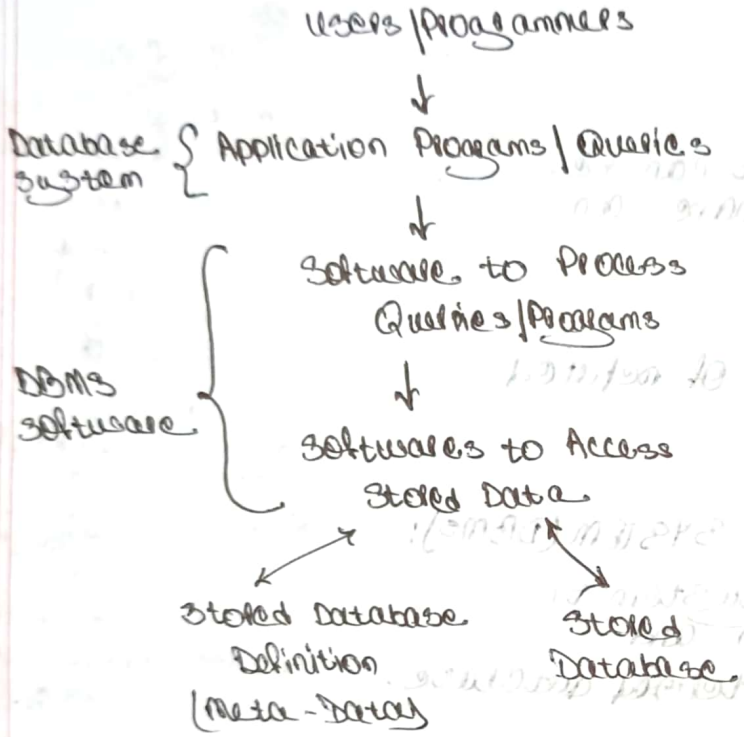
File based approach
- Each program defines and manages its own data.
- A change in one file is visible to the one who made the change only. others having the

Database approach
- Data is collected and manipulated using DBMS.
- Many programs share this data.

File based approach:-
- stored in files.
- is not centralised.

Database approach:- (imp)
- is centralised.
- less space is used.
- same view to all
- can edit and manipulation to the file and everyone can have the same view

SIMPLIFIED DATABASE SYSTEM ENVIRONMENT



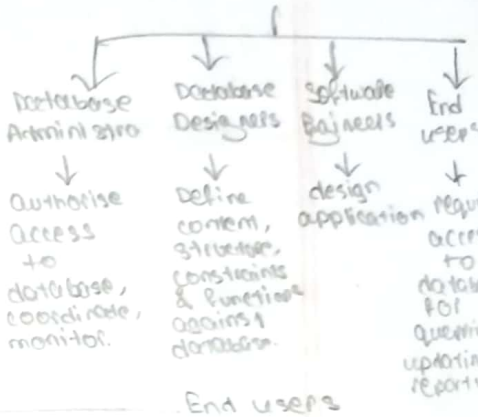
TYPICAL ~~DATA~~ DBMS FUNCTIONALITY

- Define a particular database in terms of its data types, structures and constraints.
- Construct or load the initial database contents on a secondary storage medium.
- Manipulating the database: Retrieving, Modification, Accessing the database through web applications
- Processing and sharing by a set of concurrent users and application programs → set keeping all data valid and consistent.

→ Protection or security measures to prevent unauthorized access.

→ Maintaining the database and associated programs over the lifetime of the database application.

Actors on the Scene



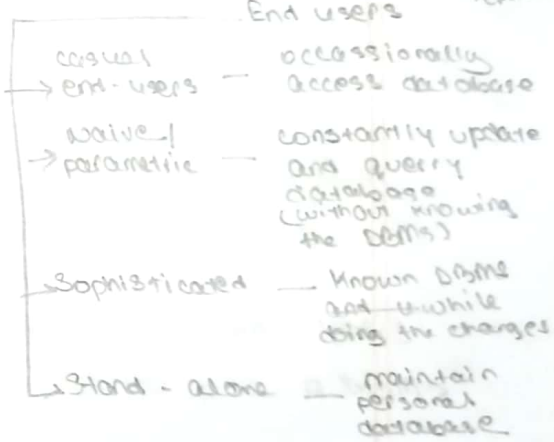
MAIN CHARACTERISTICS OF THE DATABASE APPROACH

→ Self-describing nature of a database system:
A DBMS catalog stores the description of a particular database (meta-data), allowing the DBMS software to work with different database applications

→ Insulation between programs and data:
Allows changing data structure and storage organization without having to change the DBMS access programs. (program data independence)

→ Support of multiple views of the data

End users



Advantages of using Database Approach:-
- share data between applications & users
- manage redundancy
- higher security & access control
- create/manage relationships between data

→ Sharing of data and multi-user transaction processing
- Concurrent users can retrieve from and to update the database
- Concurrency control with DBMS guarantees correct execution/abortion of transaction

* OLTP → Online Transaction Processing
- Recovery subsystem ensures completed transaction is permanently updated and failed transaction is removed.
- OLTP allows hundreds of concurrent transactions

LECTURE 2

SATURDAY DATE: 28/01/23

When not to use DBMS:

↳ expensive
[initial investment is very expensive ~~but it is not~~ and whenever it is returned back in future]
↳ unnecessary
↳ ~~no~~ DBMS may suffice
— some special operations are not supported by DBMS. so alternative is needed

CHAPTER - 1
File Based Approach
Basic Functionality
OLTP
Categories of End-users
When not to use a DBMS

CHAPTER 2

Abstraction — do not all details are expressed. Only the what is needed is expressed.

Data model — set of concepts that define the structure & constraints of database.

Categories of Data models:

* conceptual
* physical
* representational

Schemas versus Instances

→ Database Schema
Description of database
↳ Schema

→ Schema Diagram → illustrative display

→ Schema component → component of schema

***
→ Database state : The data at a particular moment.

When a constraint is present in schema and also in the table, it is called a valid state

*** Schema → state

STUDENT

Name	Section

} Schema

Student

Name	Section

} State

3 schema architecture

Schema * every part of schema uses data model

- Internal schema (physical level change)
- Conceptual schema (all the implementations are done)
- External schema (front end)

When a change is done in external schema, query is done in conceptual schema and permanent change is made in internal schema.

① external schema ↔ ② conceptual schema ↔ ③ internal schema

↳ mapping

level increases [1 < 2 < 3]

Logical + physical database examples

If we make a change in conceptual schema, the a specific part of the external schema is changed and the rest remains unchanged. — Data independence

***

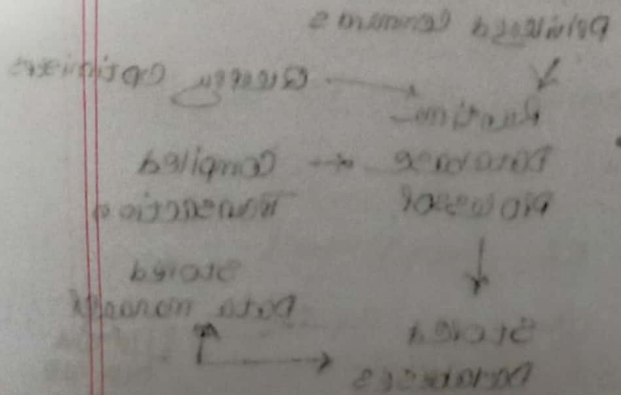
DBMS Languages

Data Definition Language (DDL) → define database schemas

Data manupilation Language (DML) →

* ~~DDL~~ In DDL, there is NO change in populated database.

* In DML, there is a change in populated database



LECTURE 3

THURSDAY

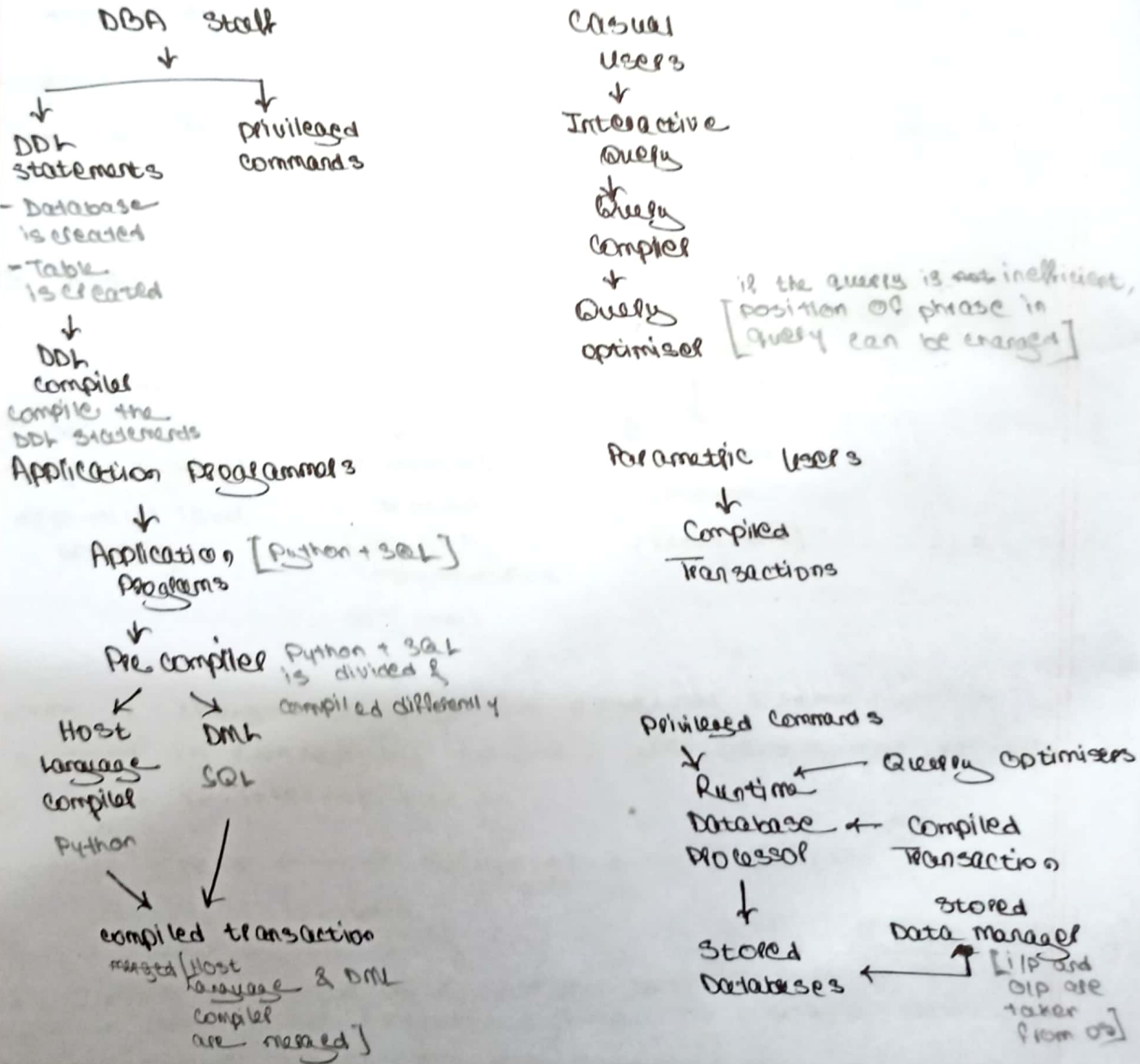
DATE: 02/02/23

DBMS Interfaces

* MariaDB is the database of mysql
* Shell is open and we write on it → stand - one query language languages.

menu based → users
parametric → bank teller

DBMS Component modules



Database System Utilities

Loading or import utility:

Backup utility

File reorganization utility

Performance monitoring utility

Data Dictionary

CASE tools

Application Development Environments

Classification of DBMS

* Data of cloud is distributed. When we do a query to a central part, the data is obtained from the distributed origin.

* Classify DBMS — 5 points

LECTURE 4

SATURDAY DATE: 04/02/23

CHAPTER 3 : DATA MODELING USING THE ENTITY-RELATIONSHIP (ER) MODEL

From miniworld, the path of requirements collection and analysing and functional analysis work together.

After conceptual design, we get a conceptual schema.
In conceptual schema, we have the entities.

In logical design, we implement the conceptual schema. We fix the datatype of the attributes of the entities.
The logical schema is fixed for the specific DBMS for which it is made.

RDMS	SQL
MySQL	MySQL

In physical ~~schema~~ design, we built the internal schema.

In functional analysis,

e.g. we do payment in an e-commerce, this is controlled by functional requirement.
how fast the payment is done, this is controlled by non-functional requirement.

High-level transaction specification of functional requirements is connected to the physical design of requirement collection.

Then application program design is build

Application program design and internal schema are mapped together in the transaction ~~desi~~ implementation. Then the application program is formed.

 → entity / regular entity / strong entity [at least one attribute is unique]

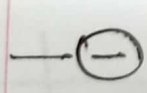
 → Attribute

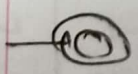
 → Relationship  → weak entity [at all the attributes are not unique]

If both the entities connected to  are strong / ~~weak~~ regular entities, then the  is relationship. 

If one of the entities connected to  are weak entities, then the relationship is identifying relationship. 

One weak entity has only identifying relationship. And, this relationship is formed with the entity

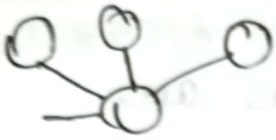
 → Key attribute [if the attribute is unique]

 → multivalued attribute

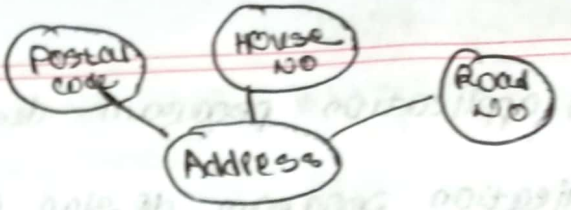
ID	Degree
1001	{0,9}
1002	

↓

multivalued attribute



Composite Attribute



simple attribute (—○) cannot be subdivided further

—◌— Derived attribute

A birth date is given and the age is derived. (example)

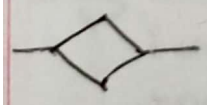
Entity → Relationship → Attribute

In "not unique" entities, we have to find out any possible unique attribute in that entity and this attribute is partially unique. —(...)

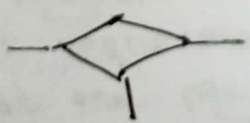
Employee	Name of children	
1001	A	B
1002	E	F

* Name of children of an employee are unique as one person will not give same name to both their children.

If one entity participates to any attribute more than once, that relationship is recursive relationship. In this case, we have to assign roles. ~~to the relati~~

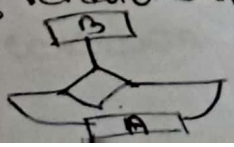


Binary relationship



Ternary relationship

Usually, ternary relationship is converted to binary relationship.



* Relationship can also have attributes.

LECTURE 5

THURSDAY

DATE: 09/02/23

Relationship Constraints — The lines that connect the relationship & entity

Cardinality Ratio — Denotes the participation of entities with another entities. (maximum participation)

Existence Dependency — minimum participation.

One employee can work for one department at max, so we place 1 on the line connecting ~~employee & works~~ (works-on) to department.

In one department, many employee work at max, so we place N on the line (works-on) connecting to employee

1	1
N	1
N	M

Cardinality Ratio [The value on the line]

* Double line — mandatory participation
* Single line — optional participation

existence dependency

Every employee must work for a department and every department must have employees. So, the relationship (works-on) connecting department & employee must have double line.

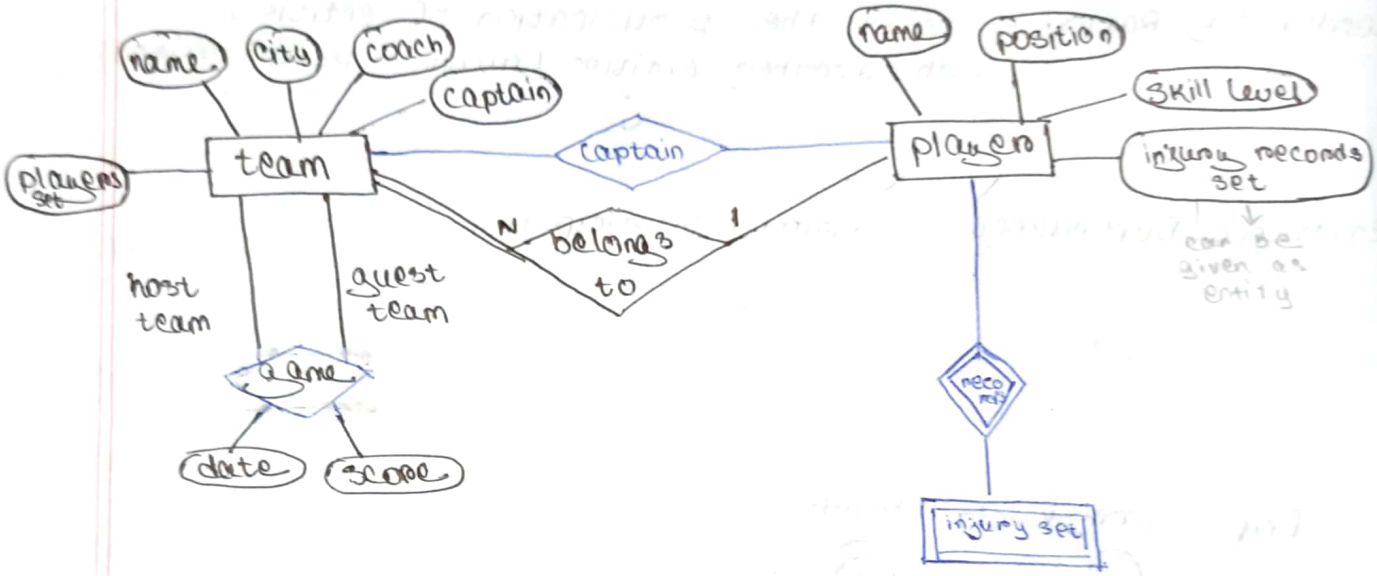
LECTURE 6

THURSDAY

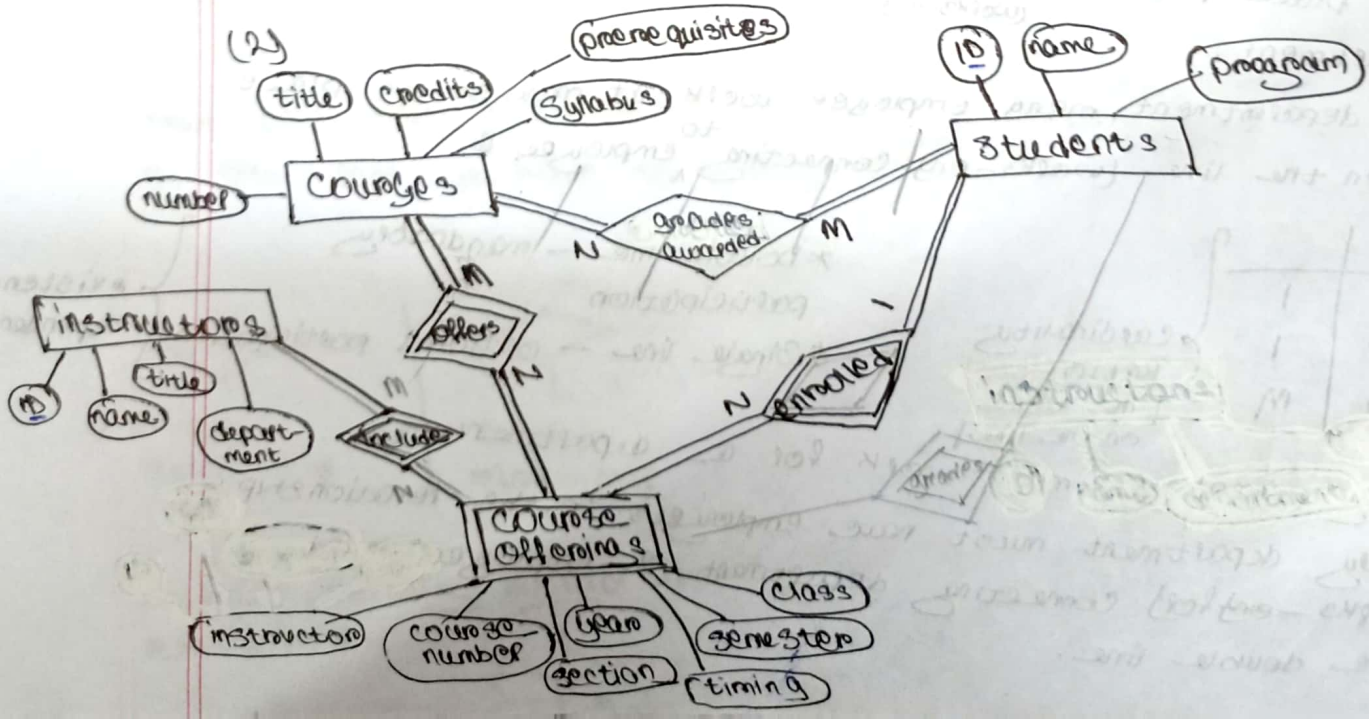
DATE: 16/02/23

ER DIAGRAM PRACTICE

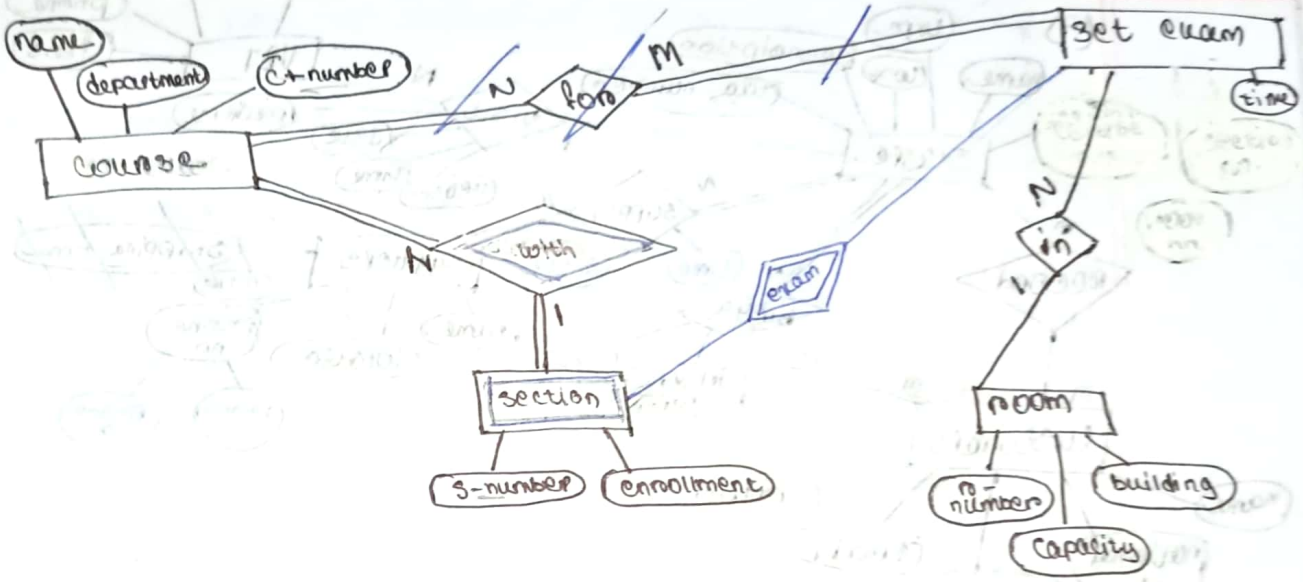
(1)



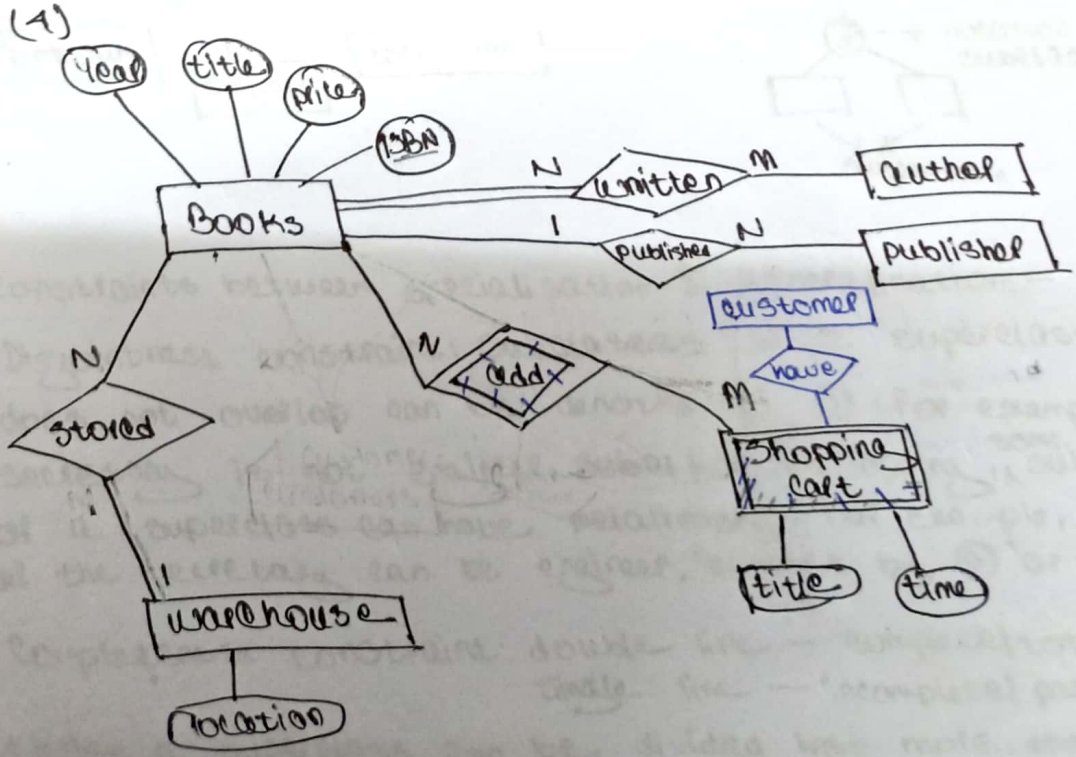
(2)

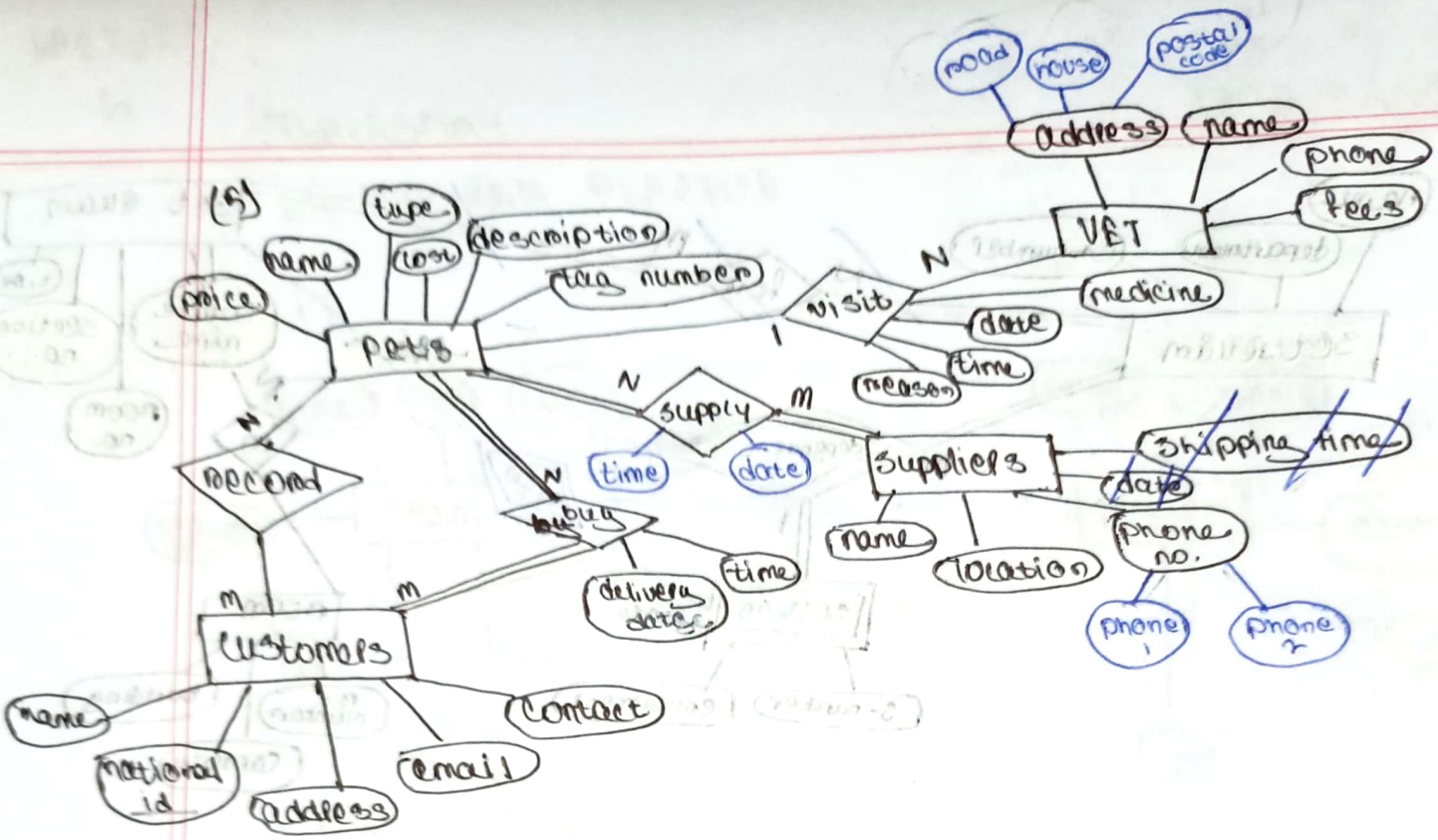


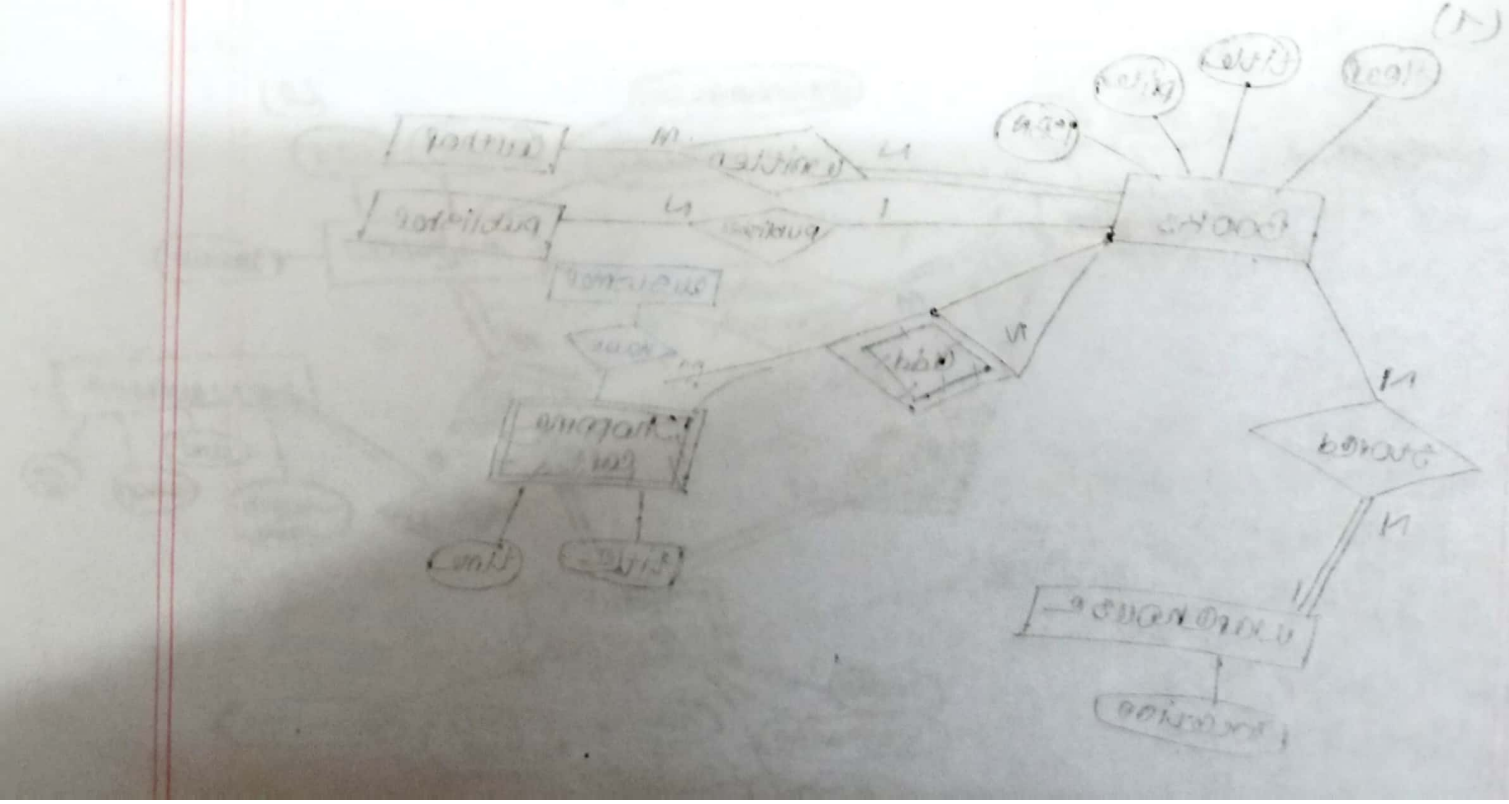
(3)


name
department
C-number
N
for
M
set exam
time
course
N
with
1
exam
N
in
1
section
S-number
enrollment
room
R-number
building
capacity

(4)


Year
title
Price
ISBN
Books
N
written
M
author
1
published
N
publisher
customer
have
M
N
stored
N
add
shopping cart
1
warehouse
title
time
location


(5)
type
name
cost
description
price
tag number
pets
visit
N
1
medicine
date
time
reason
vet
address
name
phone
fees
road
house
postal code
supply
N
M
time
date
suppliers
shipping time
date
name
location
phone no.
phone 1
phone 2
record
N
buy
M
M
customers
delivery date
time
name
national id
address
email
contact



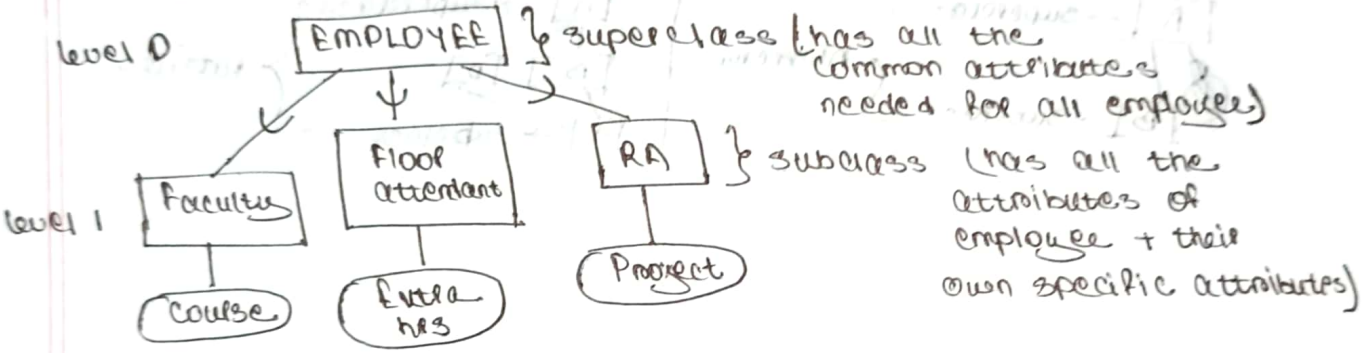
LECTURE 7

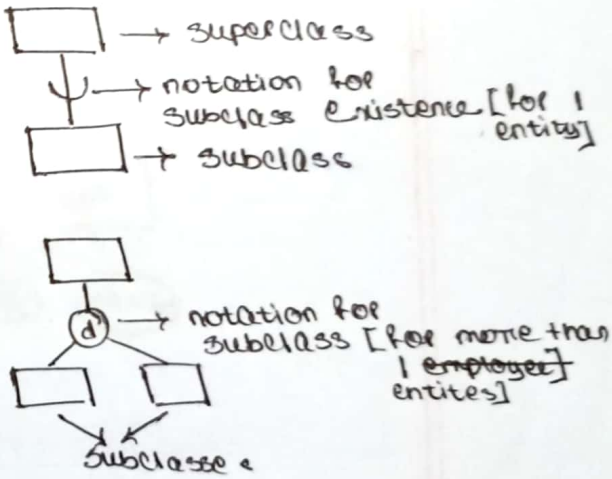
SATURDAY DATE: 18/02/23

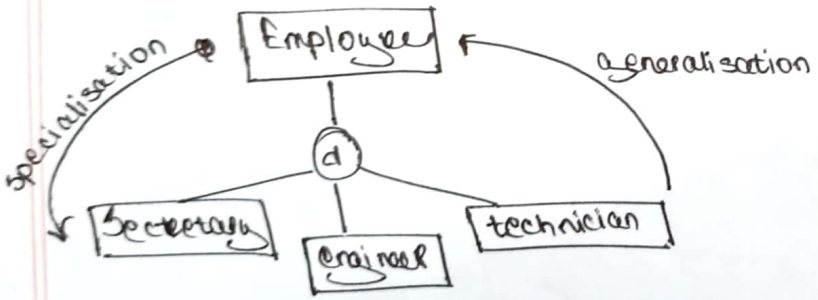
EER

EER = ER + something else

* Parent — child entity is used in EER







* Constraints between specialisation & generalisation: —

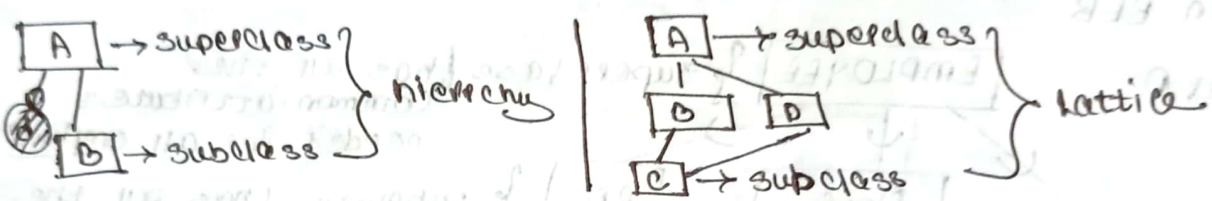
(1) Disjointness constraint: Subclasses of a superclass does not overlap can be denoted by (d). For example, ~~some of~~ some of the secretary is not engineer. ~~Sub~~ For overlapping subclasses of a superclass can have relationship. For example, some of the secretary can be engineer, Denotes by (O) or (O)

(2) Completeness constraint: double line — complete/total
single line — incomplete/ partial

* When a superclass can be divided ~~into~~ more, ~~than it~~ the ~~is~~ it is partial.
* When a superclass ~~is~~ cannot be divided ~~any~~ further, it is total.

* Hierarchy : If a subclass has only one superclass, then it is a hierarchy structure.

Lattice : If a subclass has more than one superclasses, then it is a lattice structure.



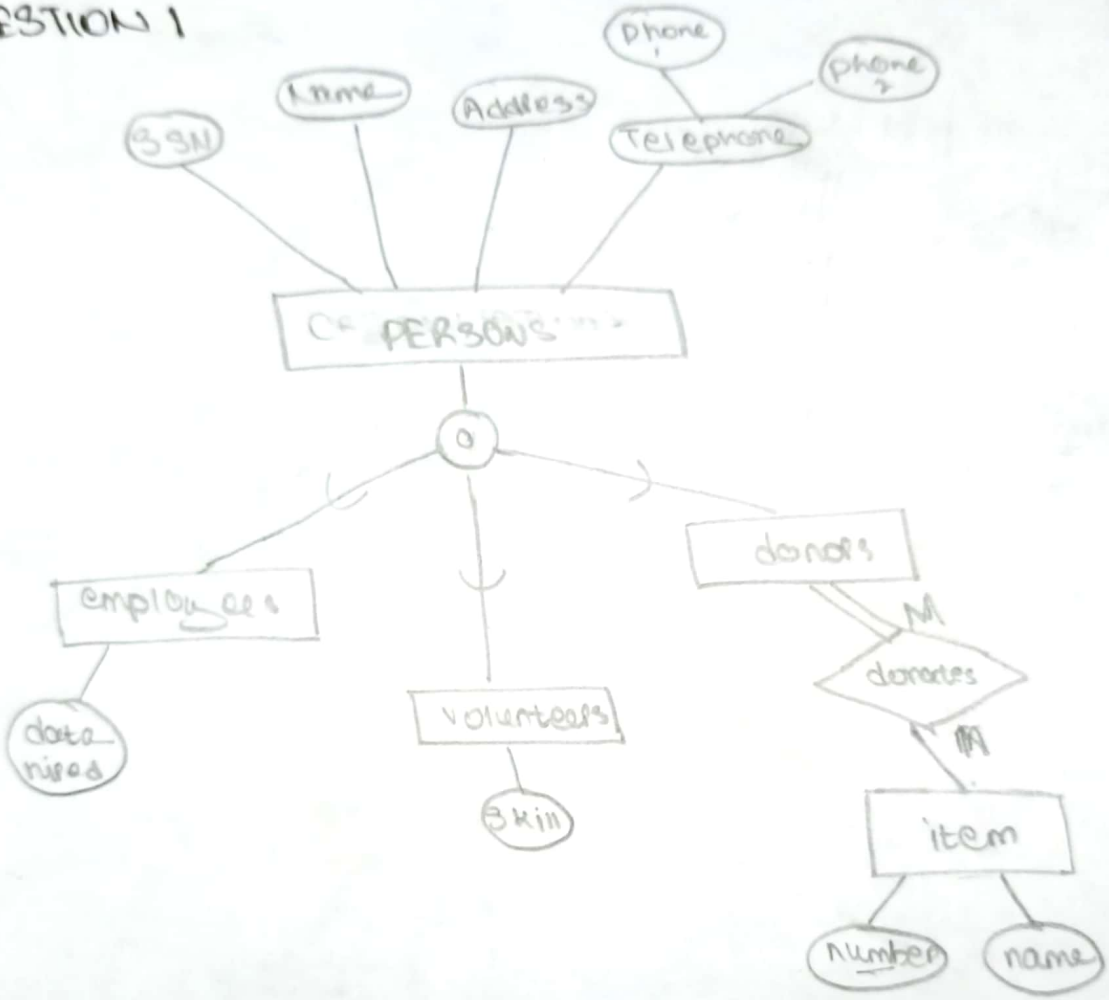
LECTURE 8

WEDNESDAY

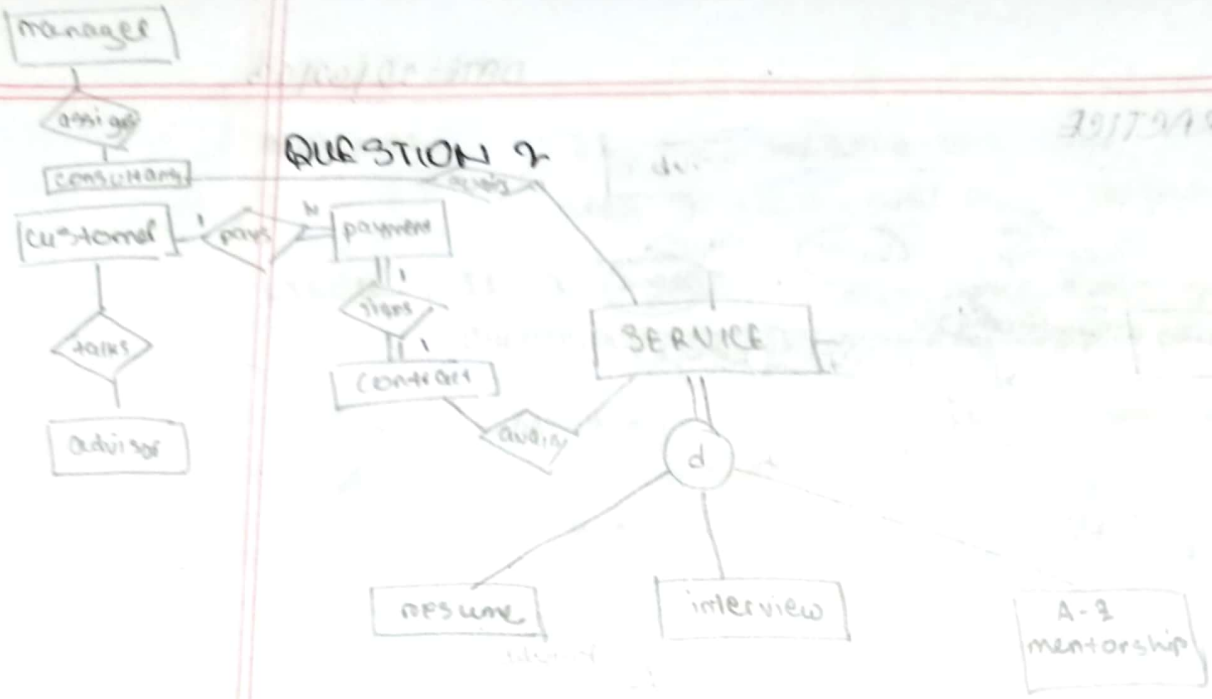
DATE: 22/02/23

EER DIAGRAM PRACTICE

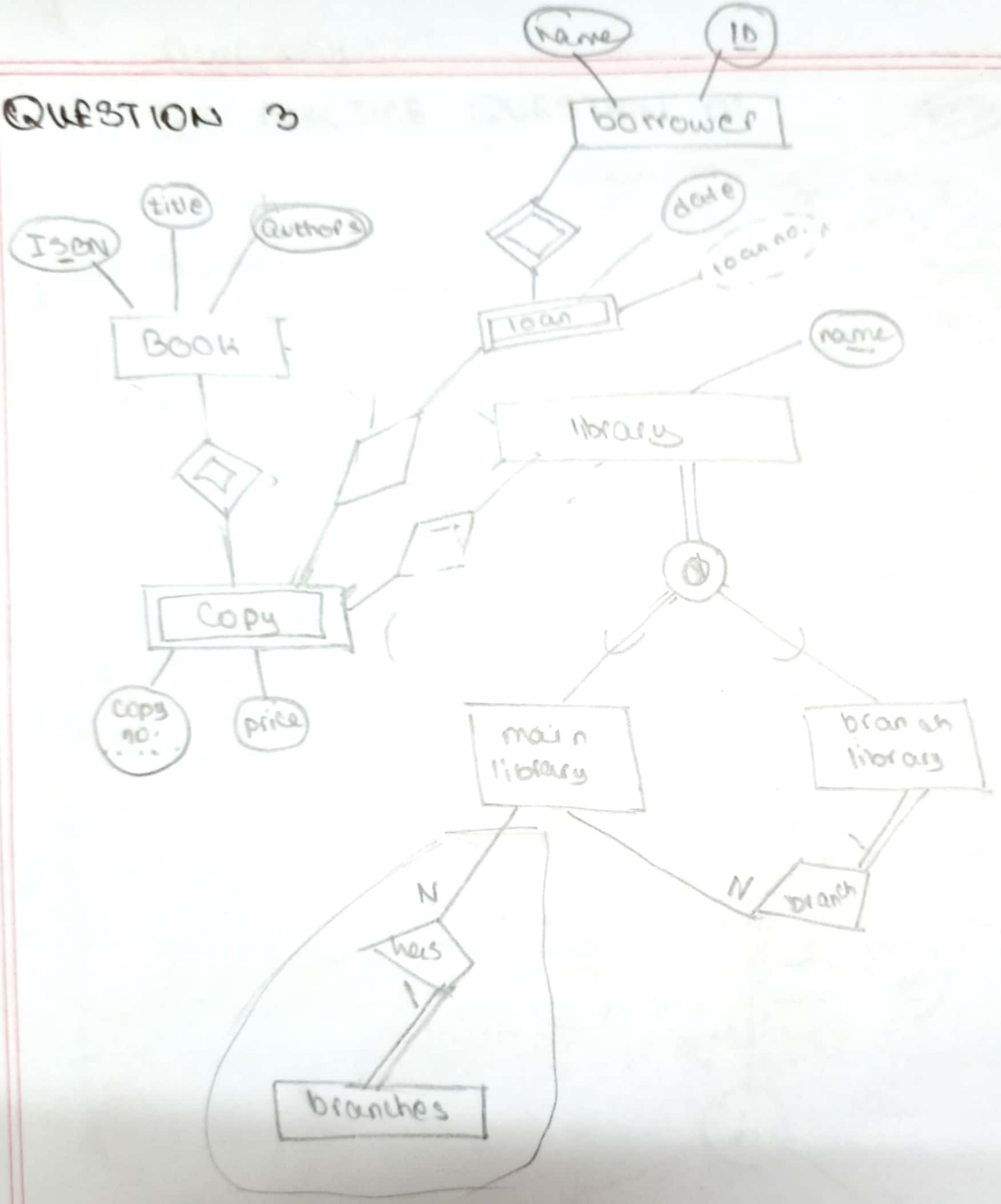
QUESTION 1

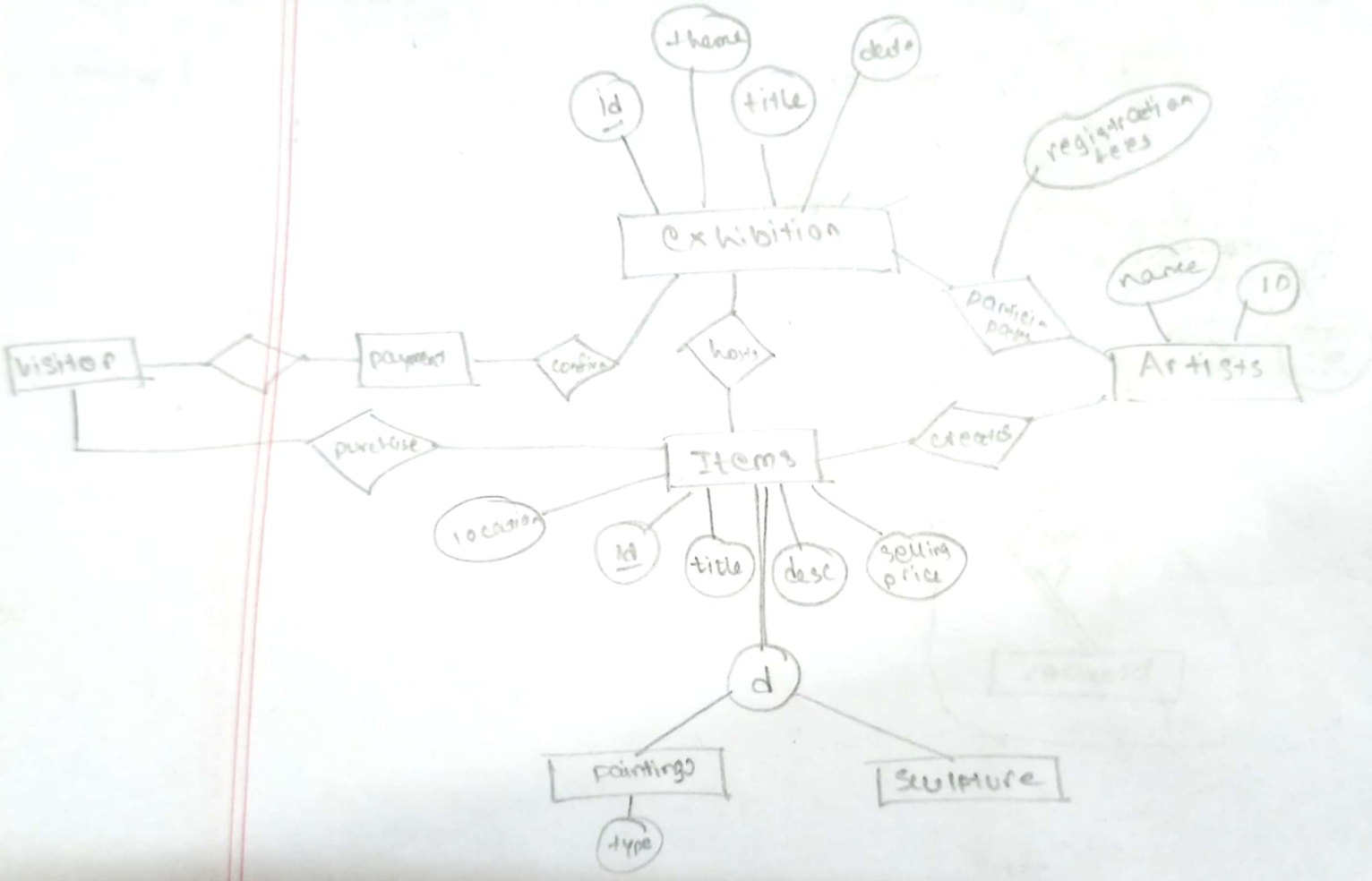


QUESTION 2


manager
assigns
consultants
customer
pays
payment
pays
contract
SERVICE
talks
advisor
audits
d
resume
interview
A-Z
mentorship

QUESTION 3


name
ID
borrower
date
loan no.
loan
name
library
title
ISBN
Authors
BOOK
Copy
copy no.
price
main library
branch library
N
has
1
branches
N
branch
1


theme
date
id
title
registration fees
exhibition
name
ID
participate
visitor
payment
confirm
hosts
Artists
purchase
creates
Items
location
id
title
desc
selling price
d
paintings
sculpture
type

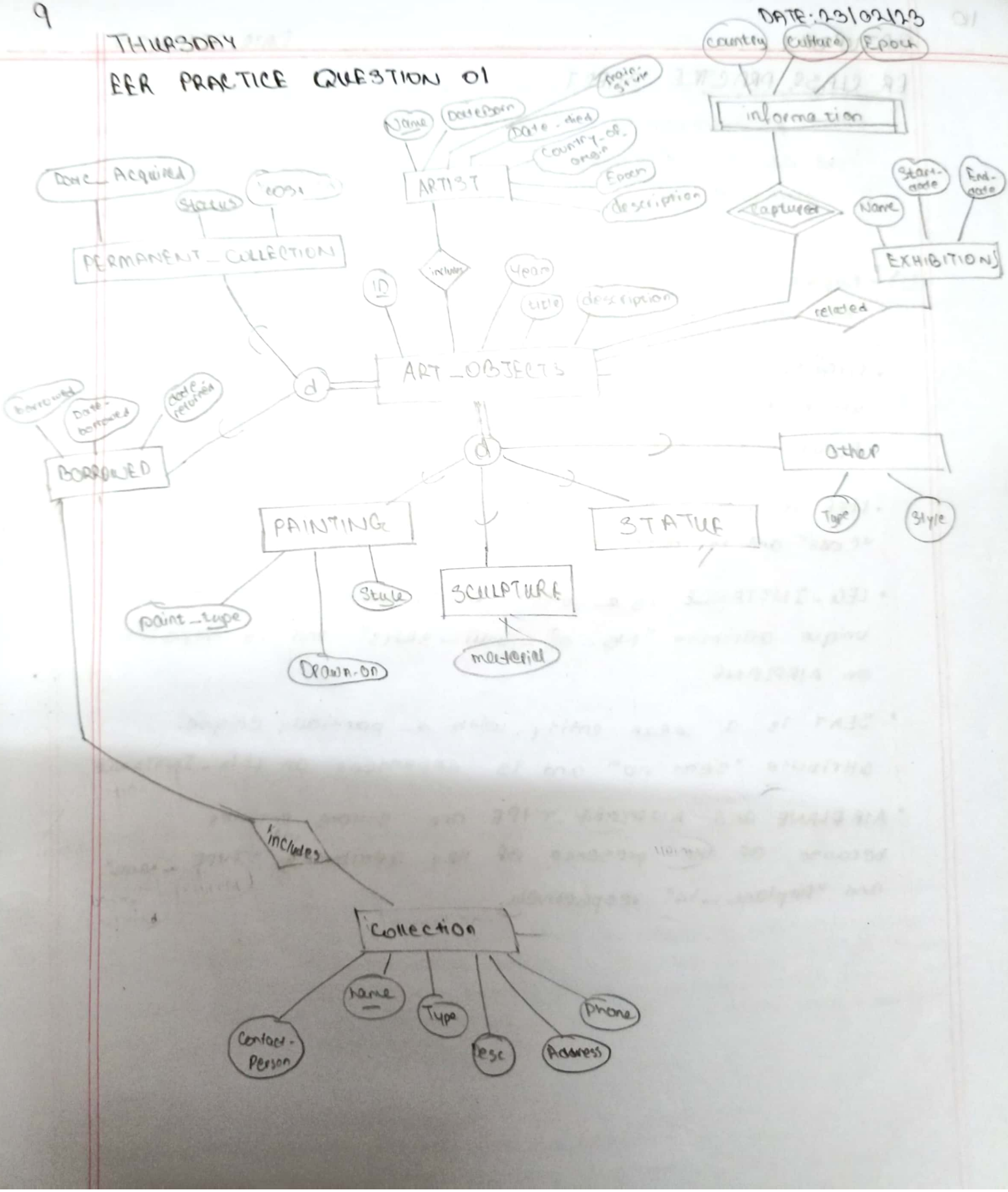
XV 6

LECTURE 9

DATE: 23/02/23

THURSDAY

EER PRACTICE QUESTION 01



LECTURE 10

SATURDAY

DATE: 25/02/23

ER CLASS PRACTICE SHEET

(1) • Airport is strong entity due to the presence of key attribute "Airport-code"

• FLIGHT-LEG is a weak entity dependent on FLIGHT, which is a strong attribute because of unique attribute "Number".

• FLIGHT-LEG has a partially weak entity, "Leg-no".

• FARE is a weak entity with a partially unique attribute "Code" and is dependent on FLIGHT.

• LEG-INSTANCE is a weak entity with no a partially unique attribute "No-of-avail-seats" and is dependent on AIRPLANE.

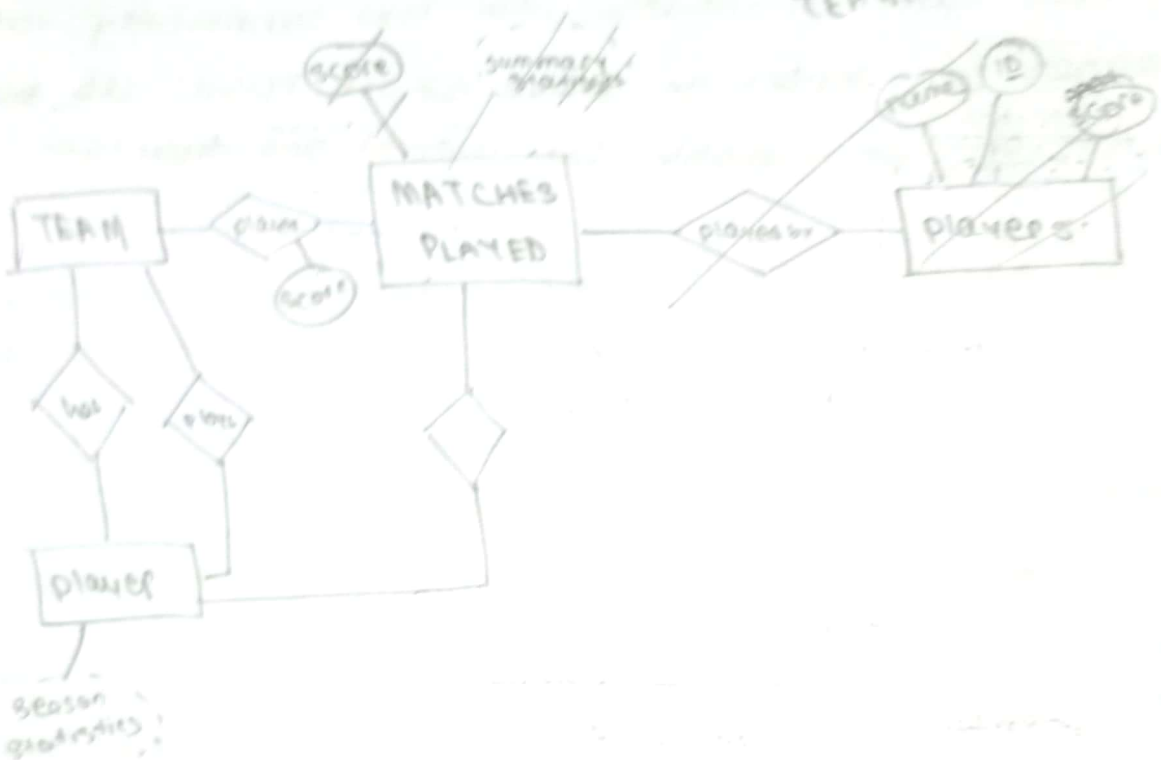
• SEAT is a weak entity with a partially unique attribute "seat no" and is dependent on LEG-INSTANCE.

• AIRPLANE and AIRPLANE_TYPE are strong entities because of the presence of key attributes "TYPE_name" and "Airplane_id" respectively.

- DEPARTURE_AIRPORT and ARRIVAL AIRPORT are the relationships between AIRPORT and FLIGHT_LEG with 1:N cardinality ratio. 1 or no AIRPORT can depart or arrive on the FLIGHT_LEG, but many FLIGHT_LEG are available for arrival and departure.

(2)

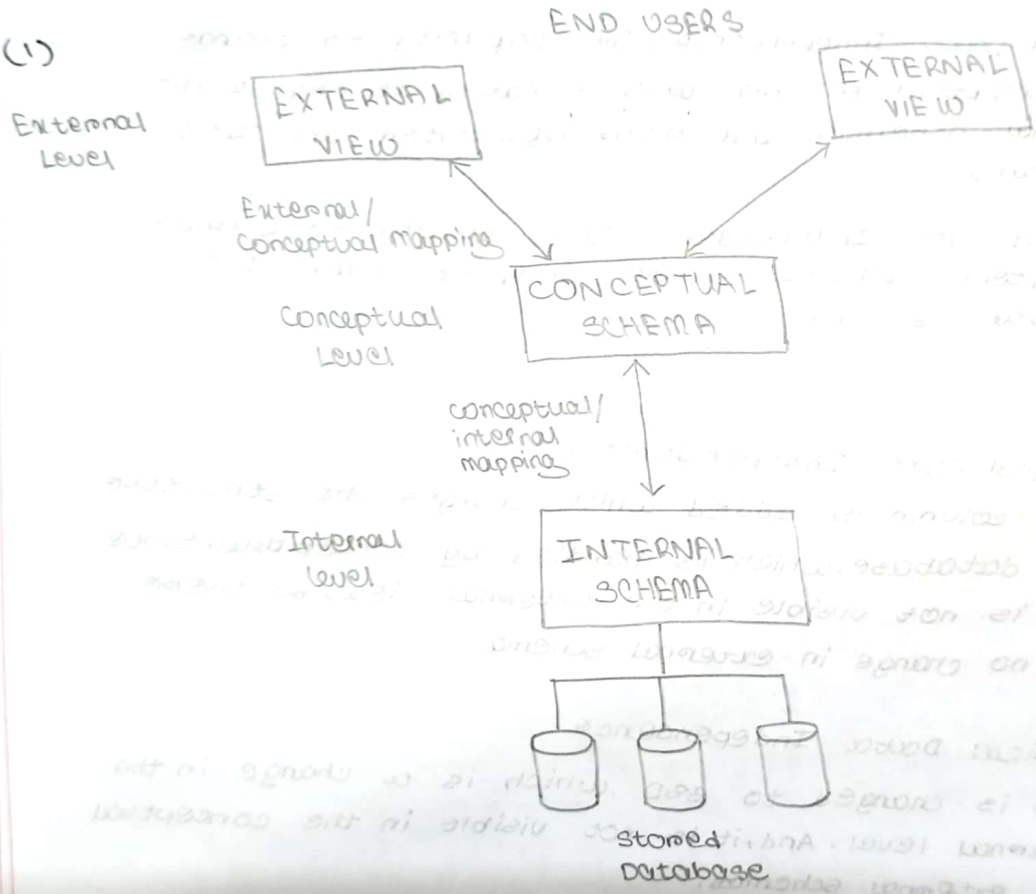
MATCH → player
player → MATCH
TEAM → MATCH



PRACTICE SHEET

CHAPTER 1 AND CHAPTER 2

(1)



INTERNAL SCHEMA: At the internal level, to describe physical storage and access paths.

CONCEPTUAL SCHEMA: At the conceptual level, to describe the structure for the whole database for the community of users. It hides the details of the physical storage structures and concentrates on describing entities, data type and constraints.

EXTERNAL SCHEMA: At the external level, to describe the part of the database that a particular group of users is interested in and hides the rest of the database from the user group.

(2) Data Independence: The ability to change the schema at one level of the database system without having to change the schema at the next higher level.

Logical Data Independence: The capacity to change the conceptual schema without having to change the external schemas and their associated application programs.

Physical Data Independence: The capacity to change the internal schema without having to change the conceptual schema.

(3) (a) Logical Data Independence.

The column is added which changes the structure of database, which is handled by conceptual. Since it is not visible in the external view, so there is no change in external schema.

(b) Physical Data Independence

HDD is changed to SSD which is a change in the internal level. And, it is not visible in the conceptual and external schemas.

(4) (a) schema

(b) state

(c) ~~schema~~ state

(d) state

(e) schema

(f) state

(g) schema

(h) schema

(i) state

Data State: The actual data stored in a database at a particular moment.

Data schema: The description of the database, including the database descriptions, constraints, database structures and data type.

The database schema changes infrequently. The database state changes every time the database is updated.

(5) → There is no data redundancy in database approach.

→ There is a centralised DBMS which controls the changes from any end user and updates it for everyone's view permanently at once.

→ Data is consistent in database approach.

→ Self Data management becomes difficult in file based approach, as any change needs to be placed in multiple places.

(6) → Self-describing nature of a database system.

→ Insulation between programs and data.

→ Support of multiple views of the data.

→ Sharing of data and multi-user transaction processing.

(7) (a) File based Approach

There is no need for multiple users access, so any changes to the data ~~does not~~ is visible to the owner.

(b) Data base Approach

Multiple user access is needed here. Only those who are interested for a particular part, that part is visible to them and the other parts remain hidden. For this, internal schema is needed.

(8) Program-Data Independence: Allows changing data structures and storage organization without having to change the DBMS access programs.

(9) Data Abstraction: The suppression of details of data organization and storage and the highlighting of essential features for better understanding.

(10) Data model: A set of concepts to describe the structure of a database, and certain constraints that the database should obey.

Helps to achieve data abstraction.

Conceptual Data model: Provide concepts that are close to the way many users perceive data.

Physical Data model: Provide concepts that describe details of how data is stored in the computer.

Representational Data model:

Provide concepts that may be understood by end users but that are not too different from the way data is organized within the computer.

Hide some details of data storage but can be implemented on a computer system in a direct way.

Used by many commercial DBMS implementations.

(11) (a) DML

(b) DML

(c) ~~DDL~~ DML

(d) ~~DDL~~ DML

(e) DML

(f) DML

(g) DDL

(h) DDL

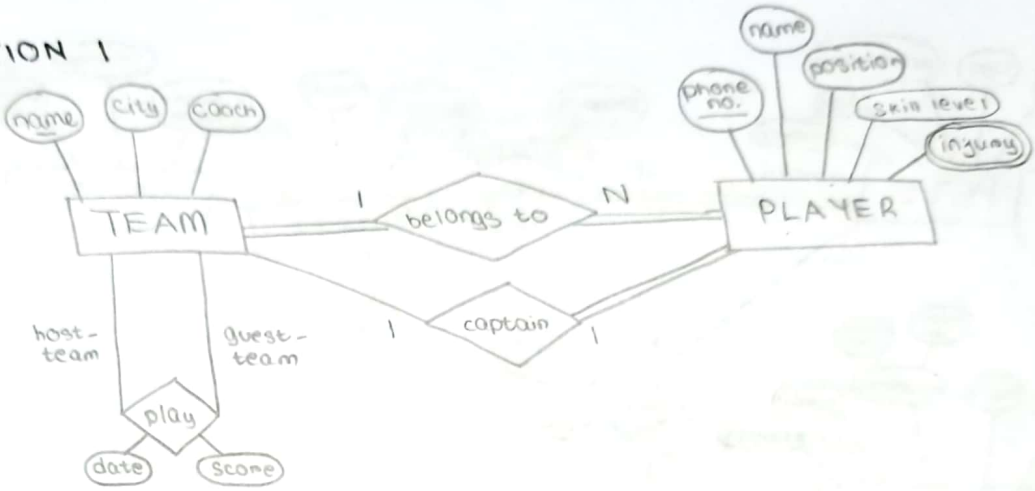
(i) DML

DDL: A language that is used to define database schemas.
[CREATE, ALTER, RENAME]

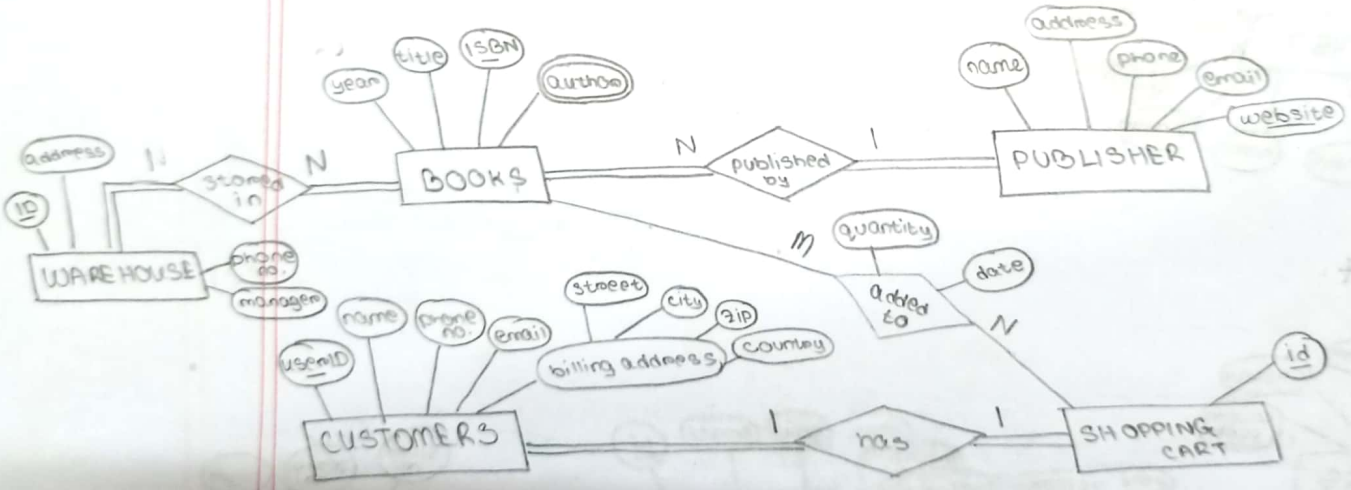
DML: A Language that is used to manipulate data
[SELECT, INSERT, UPDATE, DELECT, MERGE, RETRIVE, MODIFY]

CHAPTER 3 (ER MODELS)

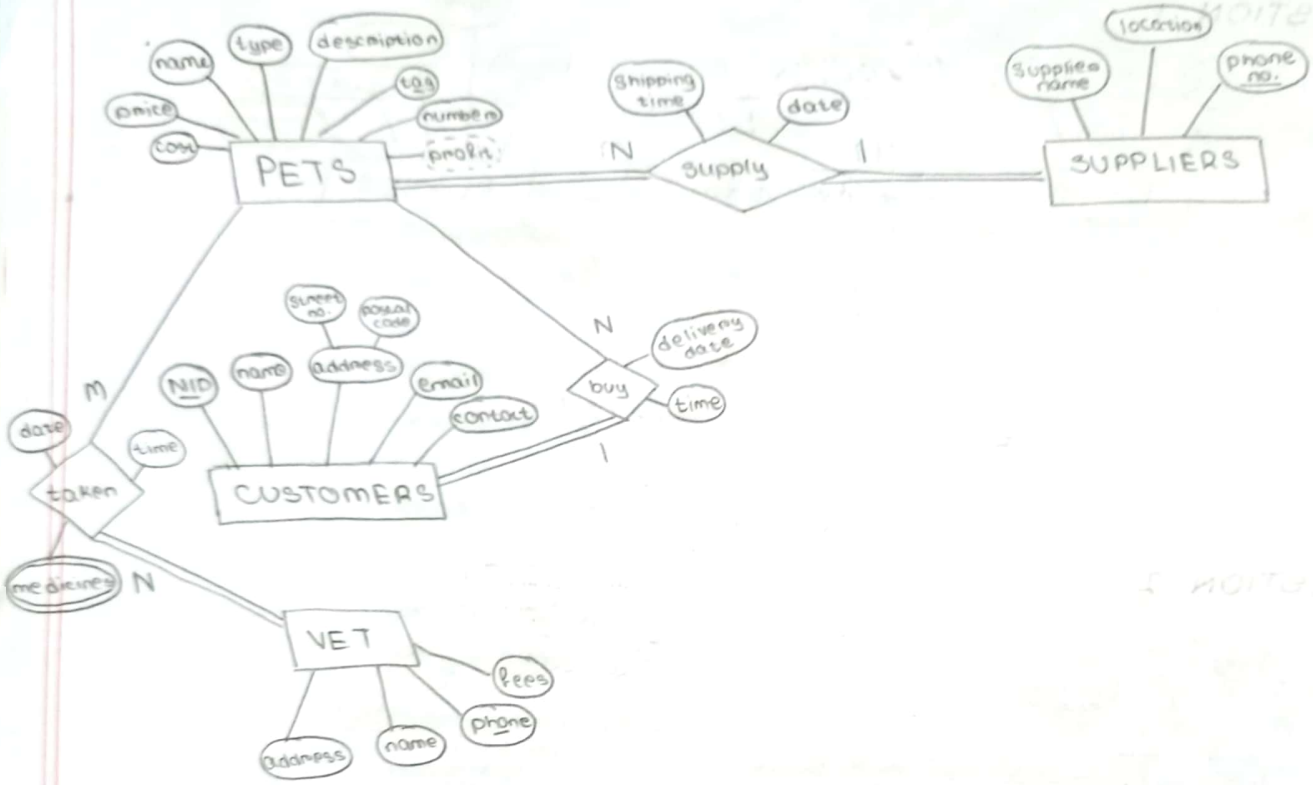
QUESTION 1



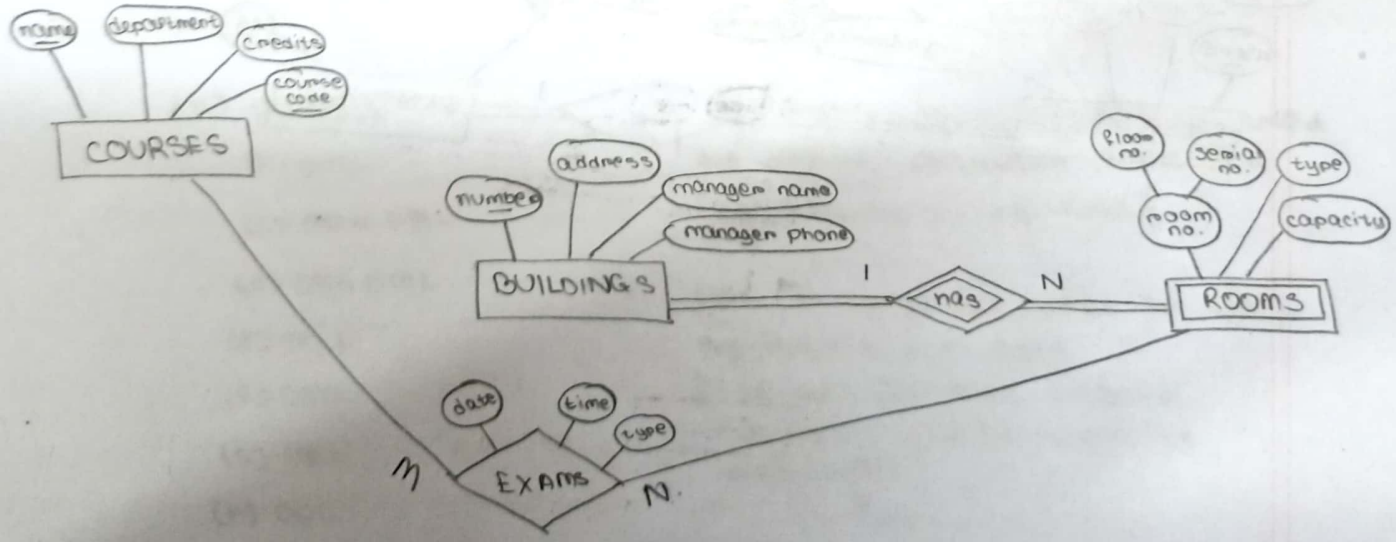
QUESTION 2



QUESTION 3

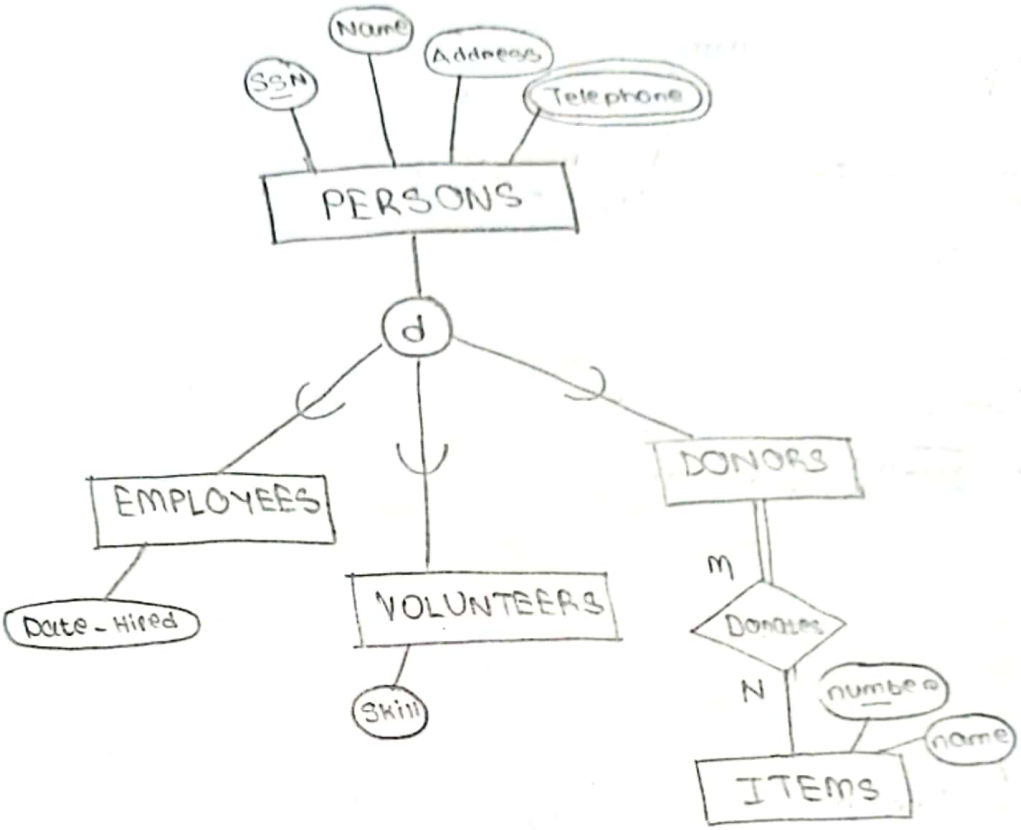


QUESTION 4

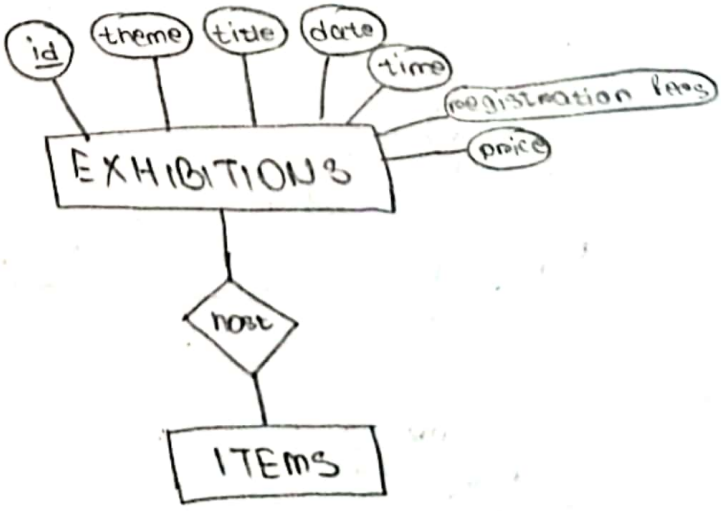


CHAPTER 4 (EER MODELS)

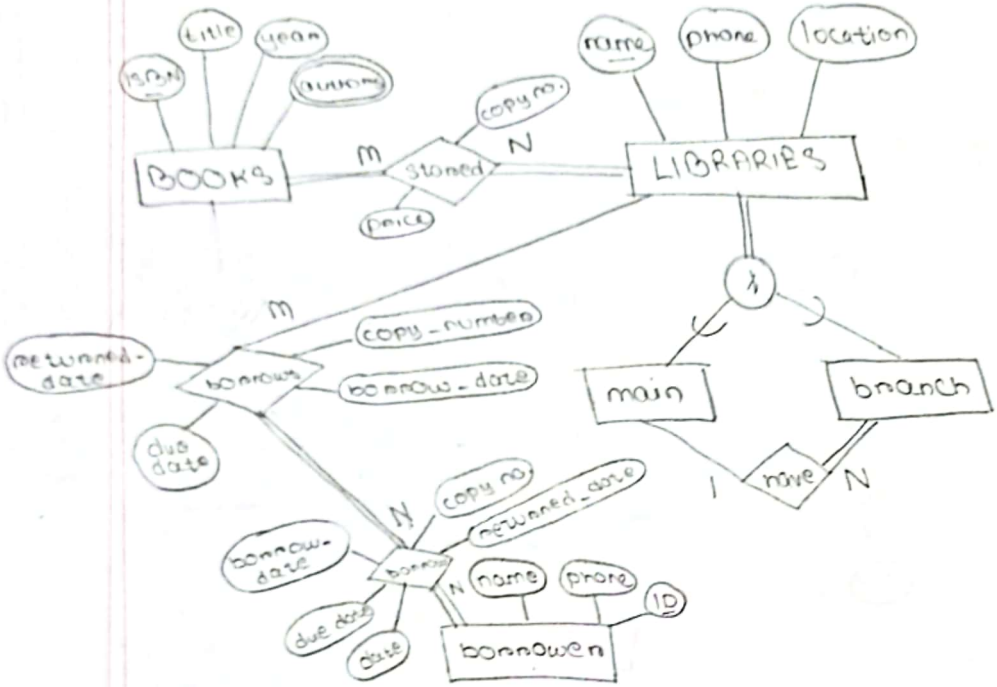
QUESTION 1



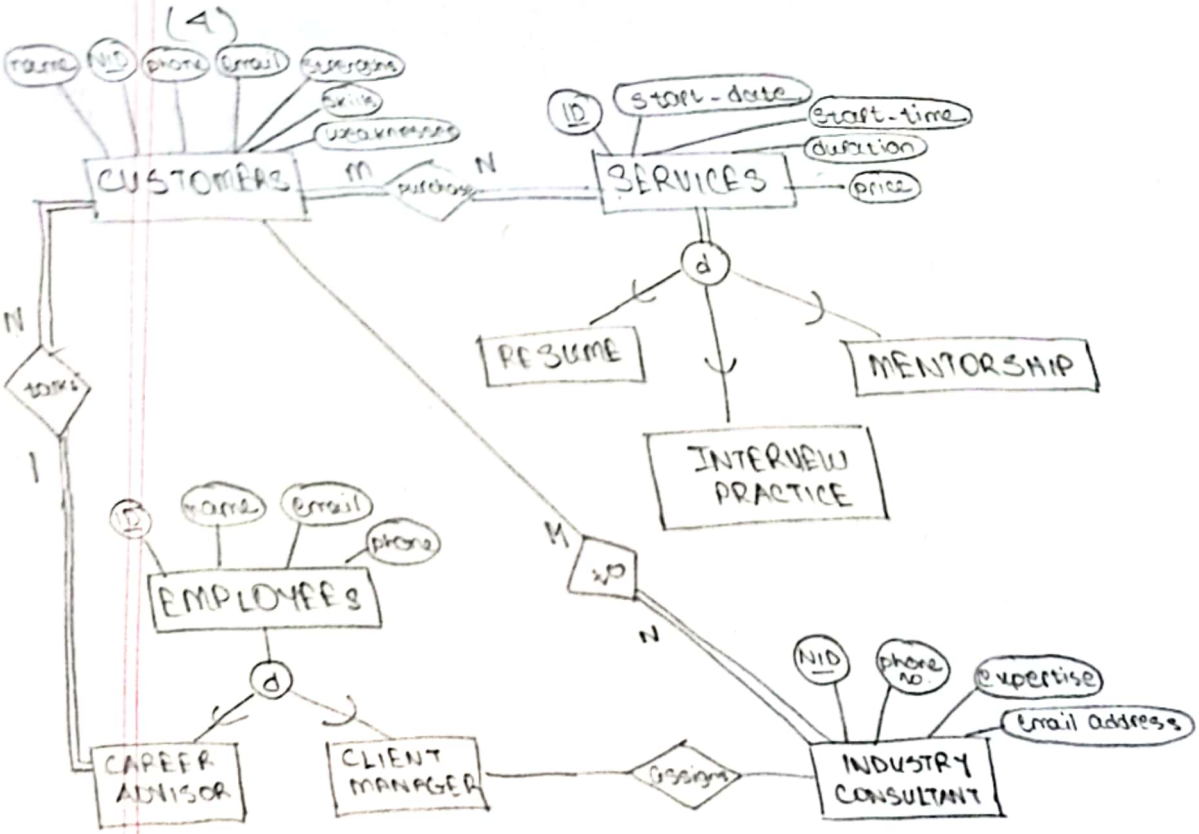
(2)



(3)


ISBN
title
year
author
BOOKS
m
stored
N
copy no.
price
name
phone
location
LIBRARIES
m
borrows
copy_number
returned-date
borrow_date
due date
main
branch
have
1
N
N
borrow-date
copy no
returned_date
borrow
N
name
phone
ID
due date
date
borrowen

(4)


name
NID
phone
email
expertise
skills
weaknesses
CUSTOMERS
m
purchase
N
ID
start_date
start_time
duration
price
SERVICES
d
RESUME
MENTORSHIP
INTERVIEW PRACTICE
N
books
1
ID
name
email
phone
EMPLOYEES
d
CAREER ADVISOR
CLIENT MANAGER
M
xD
N
NID
phone NO
expertise
email address
assigns
INDUSTRY CONSULTANT

FINAL

LECTURE 11

THURSDAY

DATE: 16/03/23

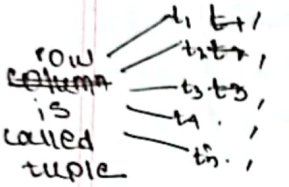
CHAPTER 5 : RELATIONAL MODEL

Entity is called "RELATION"

↓

STUDENT

name → ~~row is called~~ attribute.



$R(A_1, A_2, \ldots\ldots, A_N)$ → Relation is denoted

STUDENT (Name, SSN, ... GPA)

<'Benjamin Bayer', '305-61- ', ... > → Tuple is denoted

$r(R) = \{t_1, t_2, \ldots\ldots, t_5\}$

r(R) → set of tuple

$R(A_1, A_2) = dom(A_1) \times dom(A_2)$

dom(A1) = 0, 1

dom(A2) = a, b, c

A1	A2
0	
1	

* domain → the possible values which a specific attribute can hold.
~~Eg. dom(A1) = int~~
* we get domain from data type of attribute

$R(A_1, A_2) = dom(A_1) \times dom(A_2)$

$= \{0, 1\} \times \{a, b, c\}$

$= \{<0,a>, <0,b>, <0,c>, <1,a>, <1,b>, <1,c>\}$

↳ all the tuples present in the super set are the possible outcomes of R

* Ordering of tuple does not matter.

* Order of attribute matters.

Notation:

$t_1[A_i]$ = ...

- A_i → value from attribute
- t_1 → tuple no.
- A_i → name of attribute

Relational Integrity Constraints

(1) Key constraint
(2) Entity Constraint
(3)
(4) Domain constraint

Student

ID	Name

SK1 = {ID}
SK2 = {ID, Name}
SK3 = {ID, Name, Course Code, Grade}

removing ID from SK2 as no unique attribute is present ∴ key is only.

Key → one or more unique attributes
non-unique attributes
super key → key + non-unique/unique attributes

Grade Report

ID	Course Code	Grade

* Key is a set of one or more unique attribute through which we can identify the set uniquely.
* more set of attributes can be added as

* Key and superKey are set of attributes.
* Key is the subset of superKey.

Key = {Id, Course code}

SK1 = {Id, Course code, Grade}

SK2 = {Id, Course code, GP}

Id	Course code	Grade	GP
1001	CSE110	A	4.0
1002	CSE111	A+	4.0

Key Constraints

* These are 2 unique attributes (key) which are the candidates of primary key. These are called candidate key.
* Primary key is used to uniquely identify the rows.

Schema

S = {R1, R2, R3}

↓ name of whole database

↘ ↙ ↙ Relations

Company = {Employee, Department}

Entity Entigrity Constraint

* Primary Key values cannot be NULL. It can be repetitive.
* In a table, there must be ONE primary key & multiple foreign keys.

Referential Entigrity Constraint

LECTURE 17

SATURDAY DATE: 18/03/23

Referential Integrity

Customer (PK)	
Customer id	account no.

Depositor (F-K)	(F-K)
C_id	a_num

* Domain of customer is used as a in depositor using foreign key to form relation between two tables.
* The table which contains the foreign key. It is the referencing table.
* The table which contains the primary key. It is the referenced table.

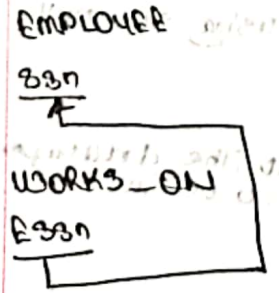
foreign key (a_num) references customer (account no.)

(a_num): attribute from referencing table

(account no.): attribute from referenced table

On update
On delete
* If one
cascade
restrict
set null

* Foreign keys have values which are already present in the primary keys which it refered or are null values.



* The arrow goes from WORKS_ON to EMPLOYEE.
So, Essn is FK and
Ssn is PK.

Populated Database State

The state of the database at present. The rows which are present at a particular time. It is the populated database state

Insert operation

- Domain constraint
- Key constraint
- Referential integrity
- Entity integrity

→ are taken into consideration.
→ One or more constraints can be violated

PK	FK
101	101
102	102
103	104

→ this value is not in PK, so it violates the referential integrity constraint. It is either restricted or the new value is inserted into PK so that FK is updated as well.

PK	FK
101	101
102	101
103	

→ inserting this value ~~can~~ violates key integrity constraint as primary values cannot be present more than once. It can be updated using set null.

PK	FK
101	101
102	101.5
103	103

→ This violates the domain constraint. The datatype of PK is int and FK also needs to be int. But here it is float.

Delete Operation

↳ Referential Operation can be violated.

Update operation

- ↳ Updating the primary key — similar to delete
  — similar to delete followed by insert
- ↳ Updating a foreign key — may violate referential constraint
- ↳ Updating an ordinary attribute — domain constraint can be violated

A constraint violation can be eliminated by using:

- (update/delete) cascade.
  [inserting/deleting a row in one relation inserts/deletes the row containing the FK in other relations]
- restrict
  [insert/delete/modify operations are aborted]
- set null
  [values are NULL]

LECTURE 13

THURSDAY

DATE: 23/03/23

CHAPTER 7

ER TO SCHEMA

STEP 1 : Mapping of Strong Entity

- Seperate tables are used
- ~~A~~ Simple attributes are used as attributes in the tables
- Composite attribute name is not used. The sub-division simple attributes of composite attribute are used as attributes.
- Multi-valued attribute is not used.
  Rather, a new seperate table is created with the name of multi-valued attribute as one of the attribute & a primary attribute which is the foreign key referenced from the main table.

STEP 2 : MAPPING WEAK ENTITY

- A new seperate table is created with all the attributes of the weak entity.
- The primary key of the table, on which the weak entity is dependent, is used as the reference of the foreign key in weak entity.

* mapping of M:N relationship is called cross-relationship.
  New relationship is created. & then foreign key is taken

* For 1:N relationship,
  1 → primary key.
  N → foreign key.
  (foreign key approach)

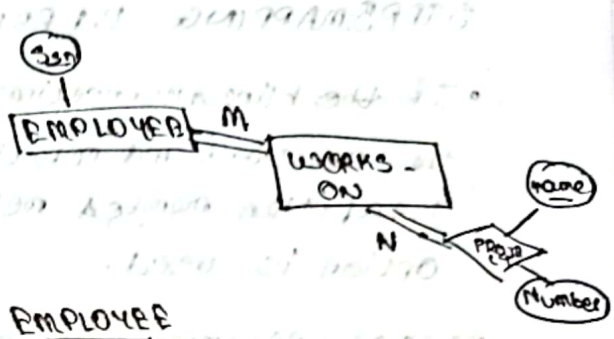
* For 1:1 mapping

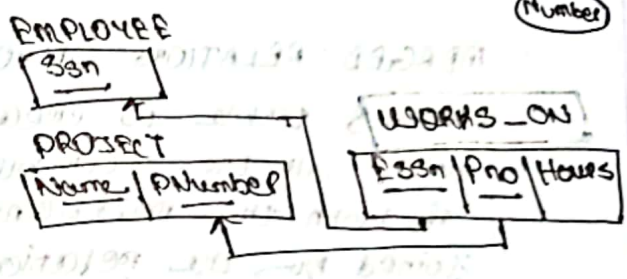
In foreign key approach, the relation which is for ~~m~~ partial relationship, that a primary key is taken from it ~~the one with not~~ to the one with total relationship as foreign key

[total side ✗ and partial side ✗ primary key]

STEP 3: MAPPING M:N RELATIONSHIP

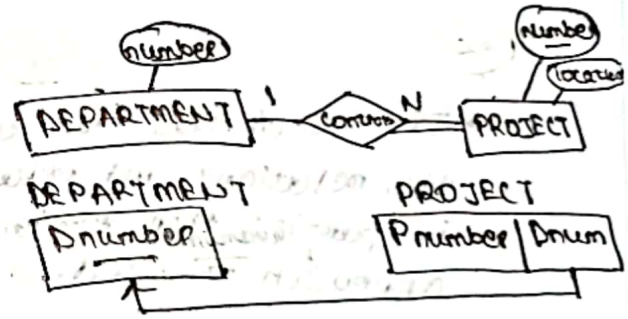
- A new table is created for the relationship and any relational attributes are is used as the attribute in the table.
- Primary keys from both the table with M & N lines are referenced into the tables as foreign key.
- This type of mapping is called cross-reference relationship





STEP 4: MAPPING 1:N RELATIONSHIP

- The relation with the line 1 is the primary key and the relation with the line N is the foreign key.
- The foreign key is used as the attribute in the relation with the line N.



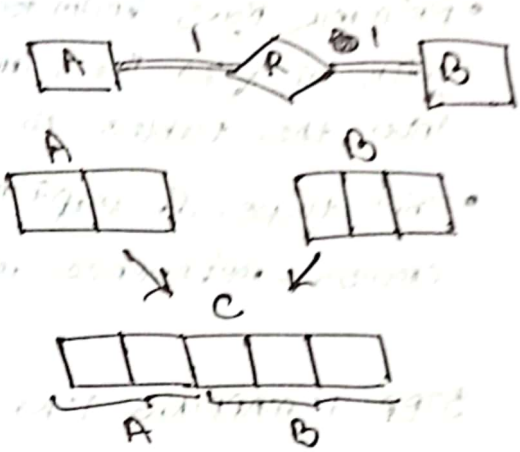
*FOR RECURSIVE RELATION, same as 1:N RELATIONSHIP

STEP 5: MAPPING 1:1 RELATIONSHIP

- If the both the line connecting the relations and relationship is 'total', then merged relation option is used.

MERGED RELATION OPTION

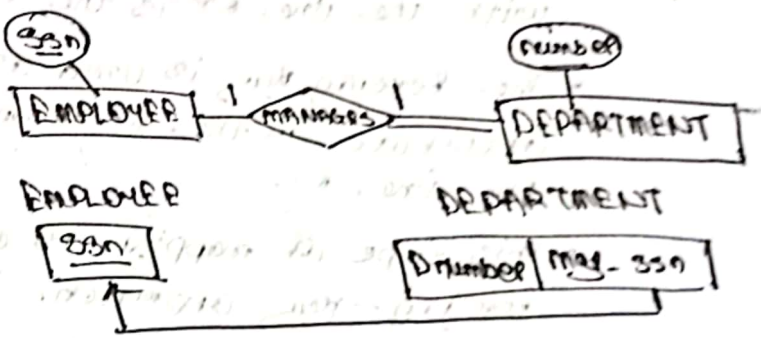
- A new table is created with all the attributes of both the relations joined by the relationship.



- If one of the line connecting the relations and relationship is partial, then foreign key approach is used.

FOREIGN KEY APPROACH

- The primary key of the relation with partial line is added to the table for the relation with total line as foreign key.



LECTURE 14

SATURDAY DATE: 25/03/23

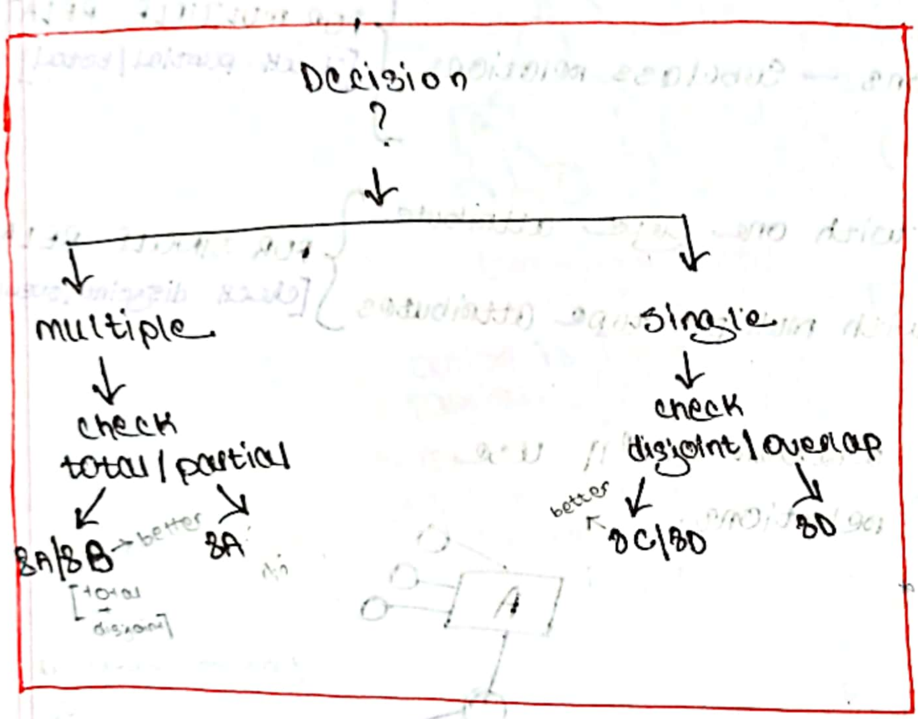
EER TO SCHEMA

* All the steps as ER to Schema + the steps for EER = EER to Schema

* Decide whether option 8A/8B/8C/8D is allowed or not.

* multiple relations map → total/partial

* single relations map → disjoint/overlapping



(1) 8A – multiple relations – superclass & subclasses. [parent + child tables]

(2) 8B – multiple relations subclass [no parent table, only child tables]

* For total + disjoint, 8B is better as superclass table is not needed as all the attributes in superclass are stored in the subclass table.

(3) 8C – single relation with only one type attribute.

* In a single relation, all the attributes of superclass, subclasses and job type [works for disjoint]

(4) 8D – single relation with multiple type attribute

* Each type has a seperate attribute

[For both overlapping & disjoint + total]

* Subclass এ attribute অনেক বেশি থাকলে, multiple relation (8A/8B) better as single relation use করলে একটা table এ অনেক columns হবে which is difficult to manage.

* Subclass এ কম attribute থাকলে, single relation use করা better.

STEP 8: OPTIONS FOR MAPPING SPECIALIZATION OR GENERALIZATION

8A: Multiple relations — Superclass and subclass

8B: Multiple relations — Subclass relations only

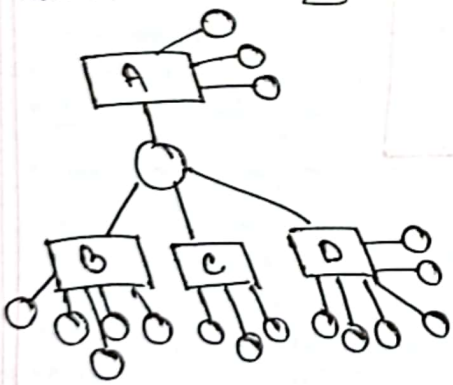
} FOR MULTIPLE RELATIONS [check partial/total]

8C: Single relation with one type attribute.

8D: Single relation with multiple type attributes

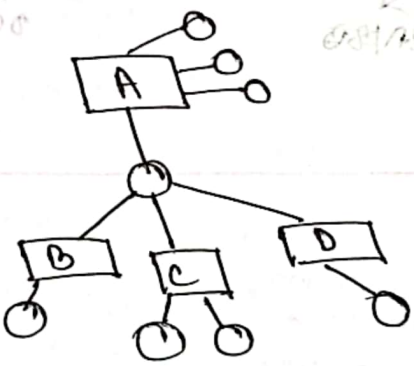
} FOR SINGLE RELATION [check disjoint/overlapping]

➤ First of all, check whether we'll use multiple or single relations.



* There are more attributes in the subclasses. So, there will be more columns in the relation. (if single relations is used)

So, MULTIPLE RELATIONS is used.

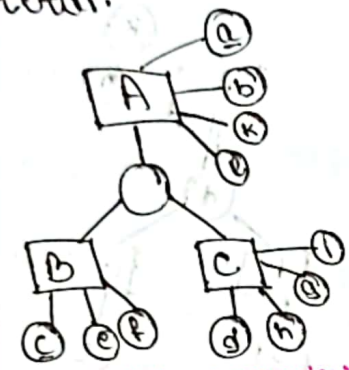


* There are less attributes in the subclasses.

So, SINGLE RELATIONS is used.

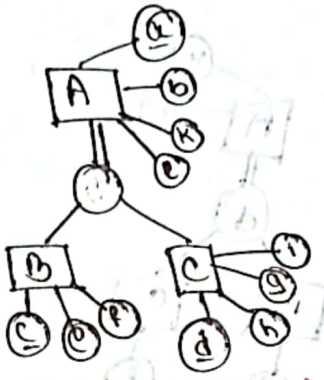
If we choose MULTIPLE RELATIONS, then we have to check partial / total.

{ 8A total/partial के लिए use करते हैं,
8B total के लिए use करते हैं, }

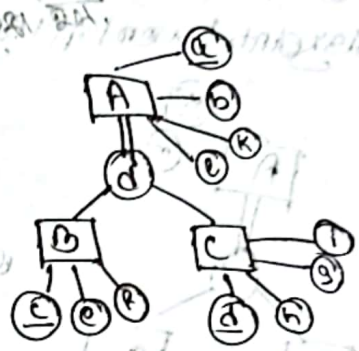


(partial + overlapping)

OR

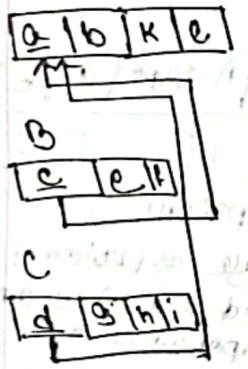


(total + overlapping)



(total + disjoint)

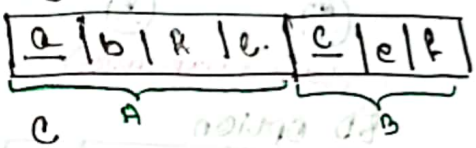
A



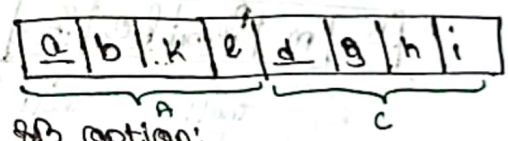
* 8A option is used for partial.
* 8A option can be used for total, if it is overlapping.

8A Option:
- seperate relation with all attributes of A, B and C are created. [relations for both superclass + subclasses]
- Primary key of A (superclass) is taken as foreign key of B and C (subclasses)

B



C

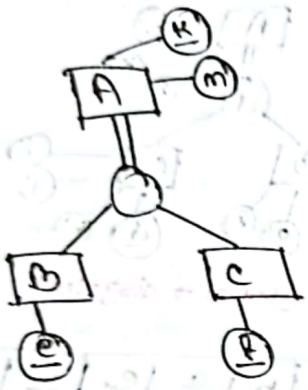


8B option:
Seperate relation with the attributes of A+B and A+C are created. No relation is created for A. [relations for subclasses are created]

* 8B option is used for total, if it is disjoint.
* 8B option cannot be used for total & overlapping, as entities of B can be repeated in C; [B and be C, e can be B + overlapping]
* 8B option will be better for total.

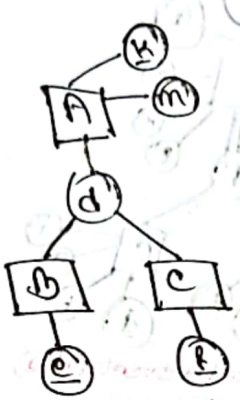
If we choose single relations, then we have to choose disjoint/overlapping.

8D disjoint/overlapping হলেও, আমরা use করতে পারি।
8C শুধু overlapping এর, আমরা use করতে পারি।

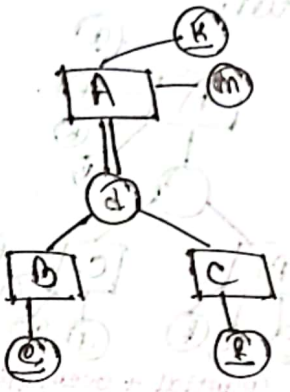


(total + overlapping)

OR



(partial + disjoint)



(total + disjoint)

A

K	m	B-type	e	C-type	f

A — True/1 or False/0 — B — True/1 or False/0 — C

* 8D option is used for both disjoint & overlapping.

* 8D option is used mainly for total +

8D Option:

- A single relation is created for A, B and C (superclass + subclasses) attributes + the subclass types.
- The subclass type attribute is boolean (True/1 or False/0)
- If B is present, then B-type is True/1 and values for e is stored. C-type is False/0 and values for f is NULL.
- If B and C are both present, B-type & C-type both are True/1 and values for e and f are stored.

A

K	m	A-type	e	f

A — B/C — B — C

8C Option:

- A single relation is created for A, B and C (superclass + subclasses) attributes + one type attribute
- If B is present, A-type is B and values for e is stored and f is NULL.
- If C is present, A-type is C and values for f is stored and e is NULL.
- It is used for total & disjoint.
- It cannot be used for total/partial + overlapping. If B and C are both present, then A-type will be B & C which corresponds to multi-valued, and it is not possible here.

LECTURE 15

THURSDAY

DATE: 30/03/23

CHAPTER 10

INFORMAL DESIGN GUIDELINES FOR RELATIONAL DATABASES

* Each tuple in a relation should represent one entity or relationship instance.

* Attributes of different entities should not mix.
(Only foreign key can be referred to primary key)

* Entity – relationship should be kept apart.

* A simple schema relation needs to be formed.

* Relations should not have insertion, update & delete anomalies, and have few or no NULL values.

* Relations should be made in such a way that it is easy to form join.

FUNCTIONAL DEPENDENCIES

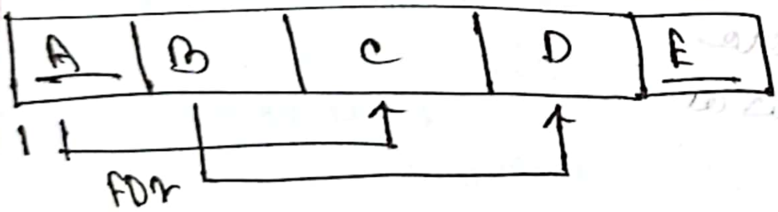
* Functional dependencies are constraint holding the relationships between attributes.

* $X \rightarrow Y$

* X & Y containing set of attributes.
* X derives Y. Y is dependent of X.

If values of attributes in Y are the ~~top~~ same values of attributes in X, then Y is dependent on X. X derives Y.

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



~~FD1 — shows that does not follow functional dependency~~

Here, partial functional dependencies exist as the attributes in FD1 does not involve ~~in both~~ with all the keys in primary key {A, B}

transitive dependencies exist as attributes are dependent on non-primary key attributes.

* Functional dependencies is a constraint.
* Derived from the meaning & relationships of the data attributes.

set of attributes
$X \rightarrow Y$ → set of attributes
(X derives Y)
→ Y depends on X

$X = \{A, B\}$

$Y = \{C, D, E\}$

[If values of Y for respective values of X is constant, then it holds fundamental dependencies.]

~~left side of~~

* One horizontal line represents one fundamental attribute.

Arrow Head has arrow → RHS

no Head has no arrow → LHS

FD1 : {SSN, Pnumber} → {hours}

NORMALIZATION OF RELATIONS

Normalization: Decomposing "bad" relations into smaller "good" relations.

* 1NF → 2NF → 3NF
- Disallows for 1NF
  - composite attributes
  - multivalued attributes
  - nested relations

* To resolve multivalued attribute, we need to create another table with primary key of main table.

1NF

Disallows:—

(1) composite attributes.

Table 1

| A | B | C | Table 1 (D, E) |

composite attributes

↓ decomposed

| A | B | C | D | E | (1NF)

(2) multivalued attributes

Table 1

| A | B | {C, D} |

multivalued attributes

↓ decomposed

Table 1

| A | B | C | D |

OR

Table 1

| A | B |

new

| A | C | D |

(3) Nested relation

Table 1

| A | B | {Table 1 (C, D)} |

composite multivalued

decompose from nested to multivalued

| A | B | {C} | {D} |

| A | B | C | D |

decompose from multivalued to simple.

* nested relation: composite or simple attribute of multivalued.

LECTURE 16

SATURDAY

DATE: 01/04/23

2NF

2NF

* Attributes fully depend on primary key.

* In 2NF, ordinary attributes depend on all the primary keys.

* If there is a partially dependent, then the FD is not in 2NF, we have to break the attribute to make into 2NF.

2NF normalization
- check for dependency
  - full dependency
  - partial dependency → break down

* By default, key attributes are on the LHS & normal attributes are on the RHS.

* If there is only 1 primary key in a relation, then it is already in 2NF.

3NF

* In 3NF, we see transitive dependency

$X \to Y$

$Y \to Z$

X is primary key
Y & Z are non primary keys.

when it is not a part of primary key, then transitive dependency issue arises.

* Here, Y is present in left of dependency & transitive dependency

- For transitive dependency, the LHS attribute is taken as primary key.

LECTURE 17

THURSDAY

DATE: 6/9/23

BOOK

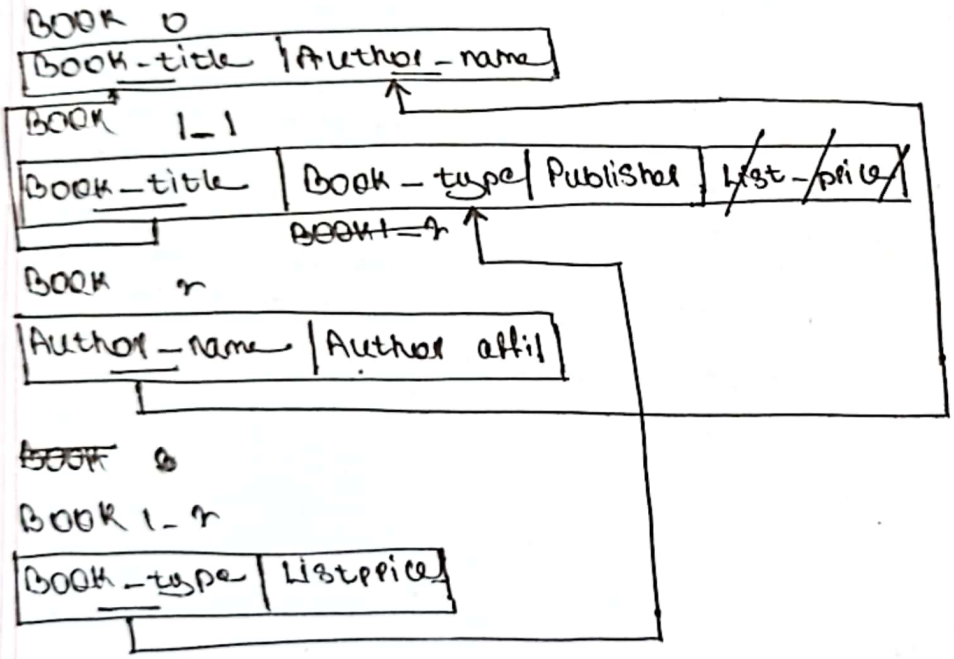
(Book_title, BookAuthor_name, Book_type, List_price, Author_authfill, Publisher)

* multivalued attribute

FD1

FD2

FD3



TURE
18

DATE: 8/4/23

SATURDAY

CHAPTER 8: MYSQL

SELECT
INSERT

* SELECT & FROM are important clauses

* FROM & SELECT are mandatory clauses

SELECT
INSERT
UPDATE
DELETE
} Data manupilation.

* WHERE

* Group by, Having filter out 4B.

* Having — sum, count functions use

* order by A, B DESC
major sorting, minor sorting

* Select A, max(B) ... from A, B ...

INSERT :—

* Insert into table_name (attribute names .....) values (data inserted);

→ used for single data input

* Insert into table_name values (data stored), (data stored);

→ used for single or multiple data inputs.

[24—28, 72—74 } excluded]

UPDATE :—

* Alter table table_name add new_attribute (column) data_type;

→ new attribute can be added.

* Alter table table_name modify column column_name new_data_type;

→ using modify column, data type is changed.

* Alter table table_name drop column old_column_name;

→ Using drop column command, a column is removed

* Alter table table_name change column old_column_name new_column_name datatype;

→ Using change column, a column name is changed.

* Update table_name set attribute = "data" needs to be changed" where condition;

→ Changes are made for specific rows where the condition applies.

→ After set, the changes need to be made are written.

DELETE:—

* Delete from table_name where condition;

→ Specific rows which lies within the condition is deleted from the table.

* Drop table table_name;

→ A table is removed from DB.

* Drop database database_name;

→ The entire DB is removed.

SELECT:—

* Select * from table_name;

→ All the columns and rows are shown.

* Select attribute 1, attribute 2 from table_name;

→ All the rows of attribute 1 & 2 are shown.

* Select distinct attribute_name from table_name;

→ The repetitive values are not shown again & again.

→ The repetitive values are shown only once.

WHERE clause
- → Comparison [= | <> | < | > | != | <= | >=] → comparison operators (AND, OR)
- → Range [between | between ... AND] (& NOT)
- → set membership [IN | NOT IN]
- → Pattern match [LIKE, NOT LIKE]
- → NULL [IS NULL | IS NOT NULL]

* IN is used instead of OR when more than two values are checked.

SELECT * from table_name where attribute_name in (value 1, value 2);

OR

Select * from table_name where attribute_name = value 1 or attribute_name = value 2;

* SELECT * from table_name where attribute_name like "_ _ a _ _ _";
  ↳ in the middle of the string, there is an "a".
  "a _ _ _" → string starts with a
  "_ _ _ a" → string ends with a
  "a _ _ _" → string starts with a and length of string is 4.

[_ _ _] → length of string is not fixed
[_ _ _] → length of string is fixed.

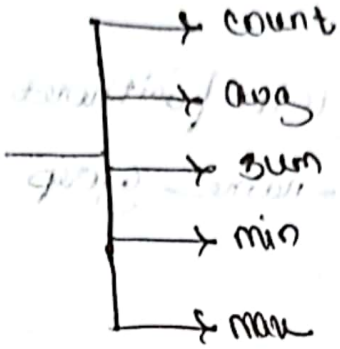
ORDER BY clause:

SELECT * from table_name order by attribute_name ;
  ↳ if nothing is given, table is sorted in ascending order wrt attribute_values in attribute_name.
  asc; → sorted in ascending order
  desc; → sorted in descending order

* If using a single column does not provide enough info for proper sorting, multiple columns are used.

SELECT * from table_name order by attribute_name 1 , attribute_name 2 desc;
* the first attribute (attribute_name1) → major
* the last attribute (attribute_name2) → minor

Aggregate Functions
("Having" clause is important)



* count, min & max = for numeric & non-numeric
* sum, avg ≠ " " " "
* All functions eliminates nulls at first and then do the operations. [Except count(*)]
* To eliminate repetitive values, distinct is used before the attribute_name on which the aggregate functions need to be applied. [count (distinct attribute_name)]
* No effect of distinct on min, & max

Select count(*) from table_name where condition;
count (attribute_name)
[min/max/sum/avg]

GROUP BY Clause

* A single value per column group is present.
* Where clause does not works first, and then group by clause works. Thus, where clause is not used after group by. Instead of where, having clause is used.

SELECT * from table_name group by attribute_name;

Having clause

* Used after group by with/without aggregate function.
* Select * from table_name group by attribute_name having condition.

SUBQUERY

* When values in one row needs to be checked with values of another rows bearing some conditions.

SELECT * from table_name T1 where attribute_name = (SELECT attribute_name from table_name T2 where condition);

↳ values in T1 = values in T2 → True

ANY </>

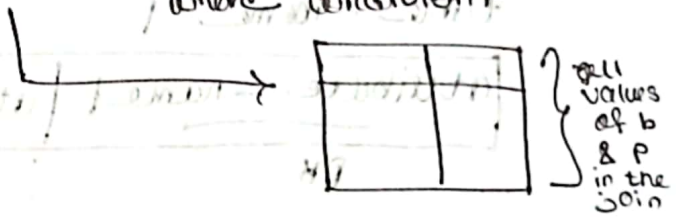
↳ At least one value in T1 </> values in T2 → True

ALL </>

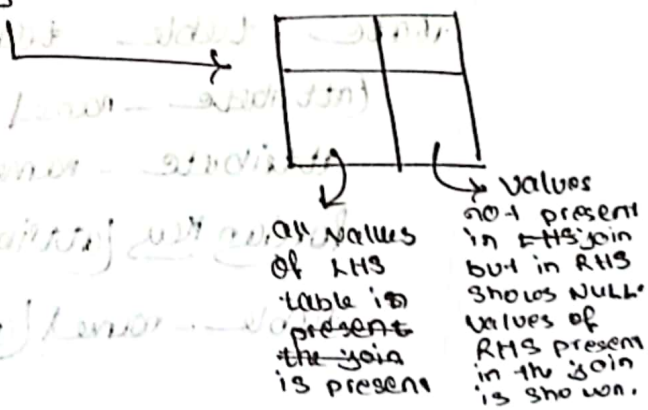
↳ All values in T1 </> values in T2 → True
At least one value in T1 </> values in T2 → False

JOIN

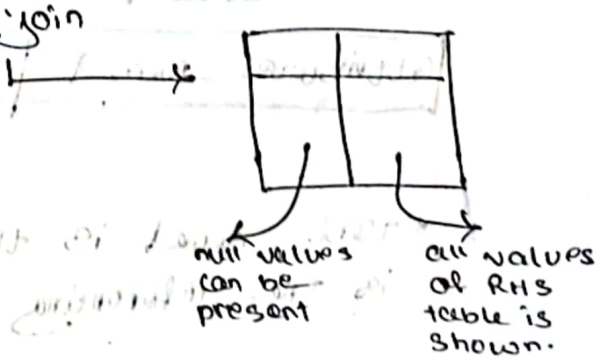
* Two or more tables are taken into consideration.
* Inner Join — All values from LHS table & RHS table are taken.
* Left Join — All values from LHS table is taken
* Select *b, *p from b inner join p on p.name = b.name where condition;



left join



right join



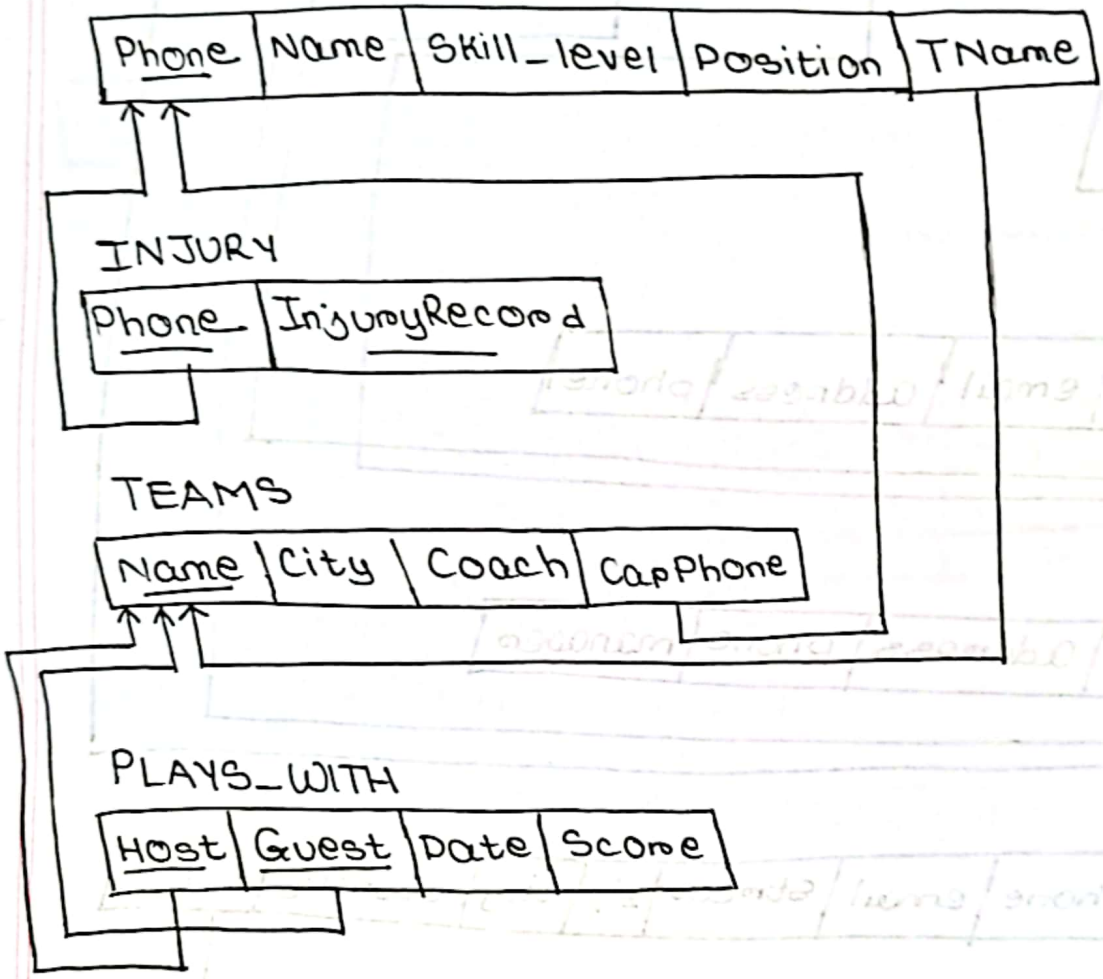
* select b.*, p* from b,p where b.name = p.name and condition;
  ↳ where clause can also be used.

Sequence to follows while writing SQL:

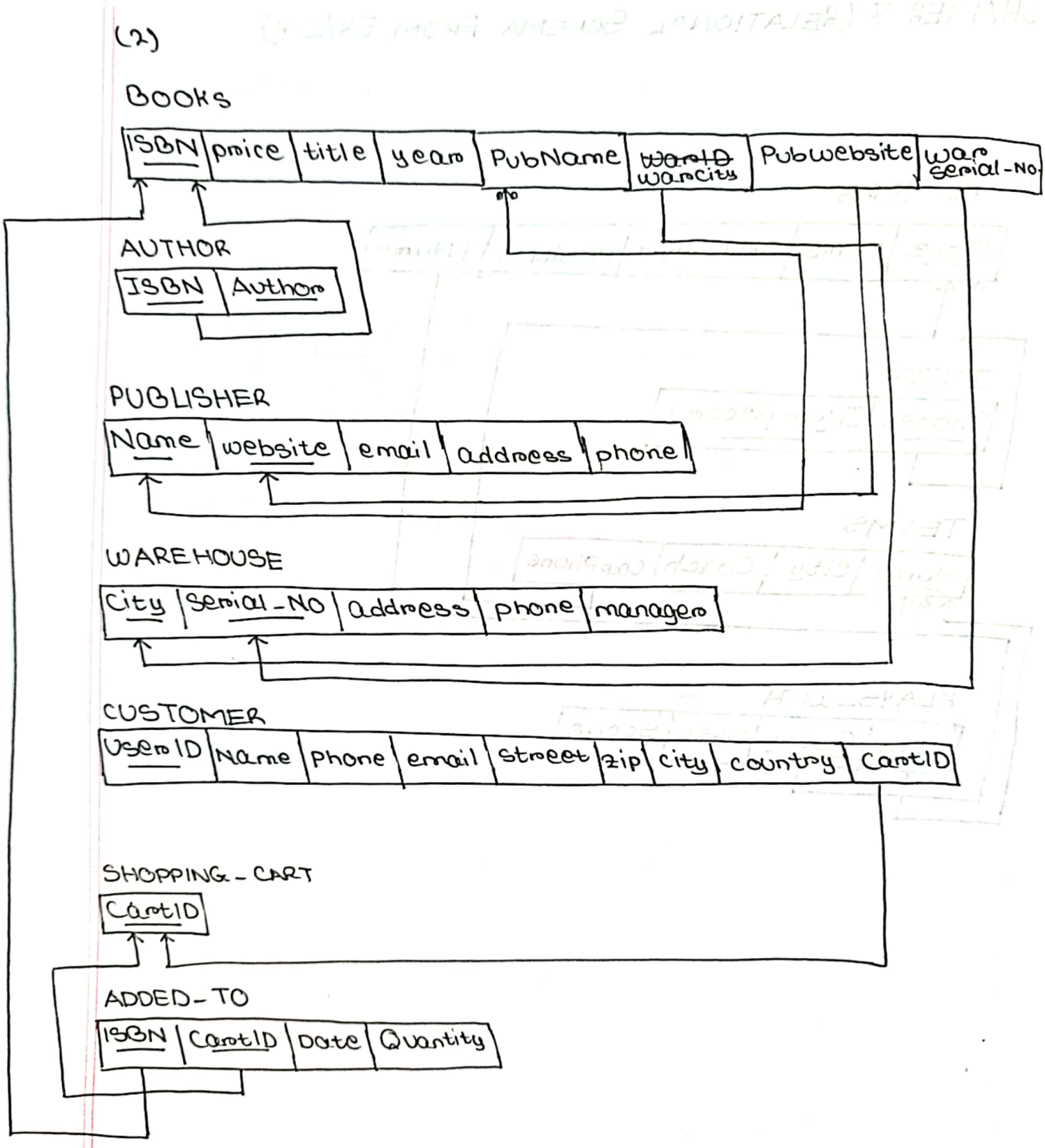
SELECT <attribute list>
FROM <table list>
[WHERE <condition>]
[GROUP BY <grouping attribute(s)>]
[HAVING <group condition>]
[ORDER BY <attribute list>]

CHAPTER 7 (RELATIONAL SCHEMA FROM ER/ERR)

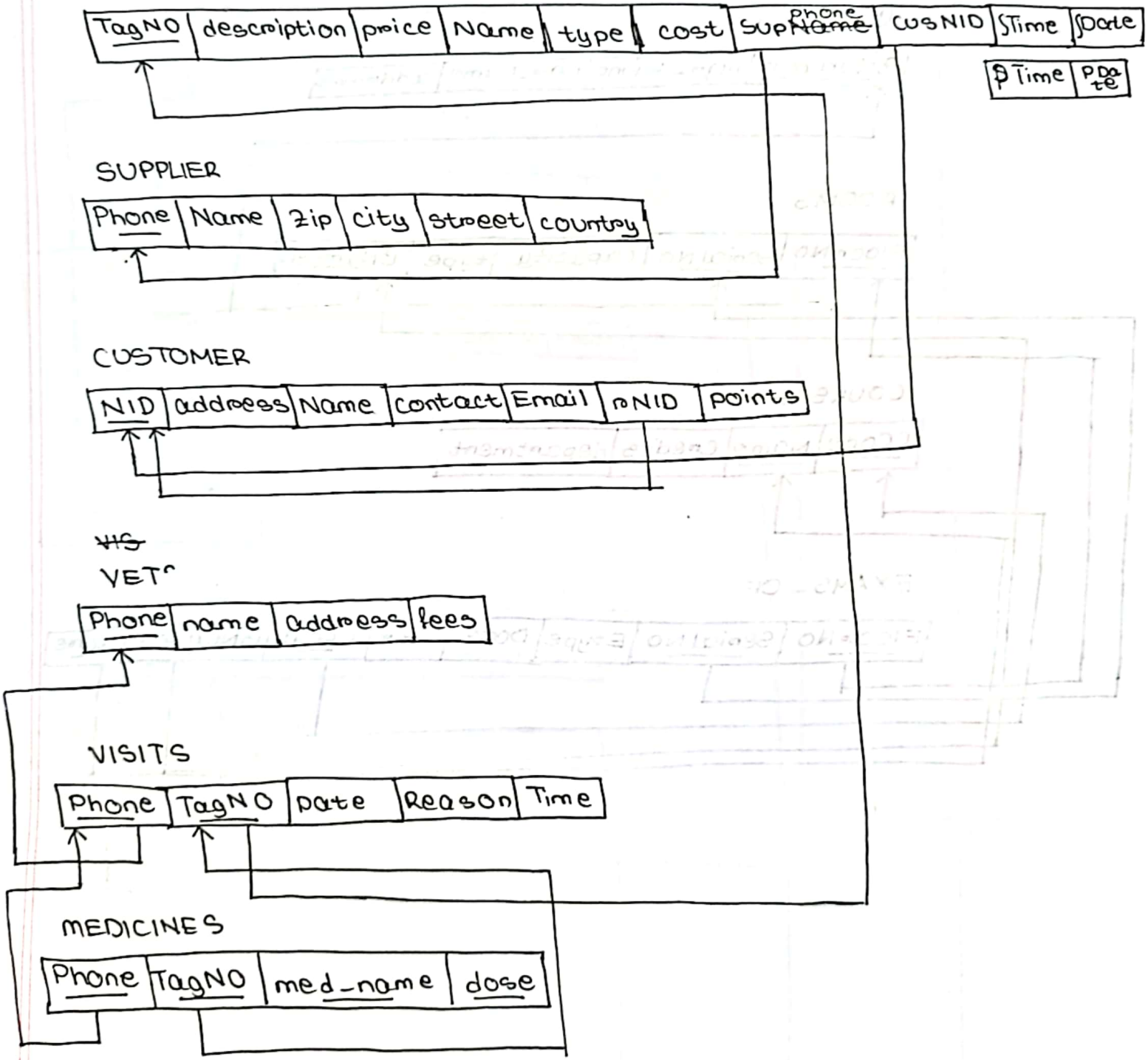
(1) PLAYERS



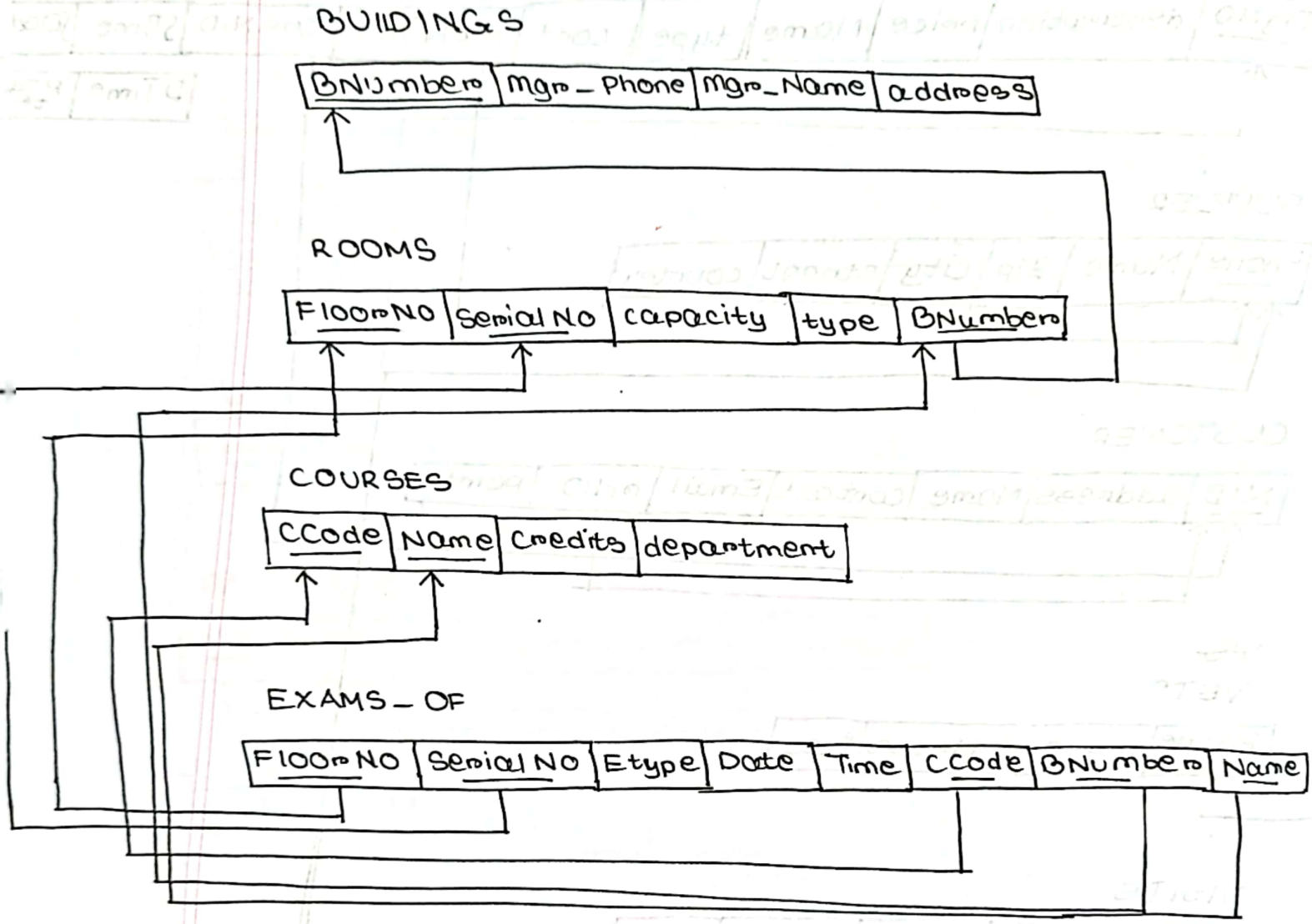
(2)



(3) PET


TagNO
description
price
Name
type
cost
supPhone
cusNID
sTime
sDate
pTime
pDate
SUPPLIER
Phone
Name
Zip
City
street
country
CUSTOMER
NID
address
Name
contact
Email
rNID
points
VETs
Phone
name
address
fees
VISITS
Phone
TagNO
pate
Reason
Time
MEDICINES
Phone
TagNO
med_name
dose

(4)



(5) SERVICES

ID	start Time	start Date	Duration	Price	SType

CUSTOMER

NID	name	phone	skills	email	weakness	CaID

INDUSTRY_EXPERT

NID	name	expertise	phone	email

PURCHASE

ID	NID

EMPLOYEE

employee-id	email	name	phone

career_Advisors

ID

client_Managers

ID

ASSIGNED

NID	ENID	CliD

(6)(A)

8A

Student

ID	dept	Name	cgpa

Undergrad

ID

Graduate

ID	major

8B

Undergrad

ID	dept	Name	cgpa

Graduate

ID	dept	Name	cgpa	major

8C

Student

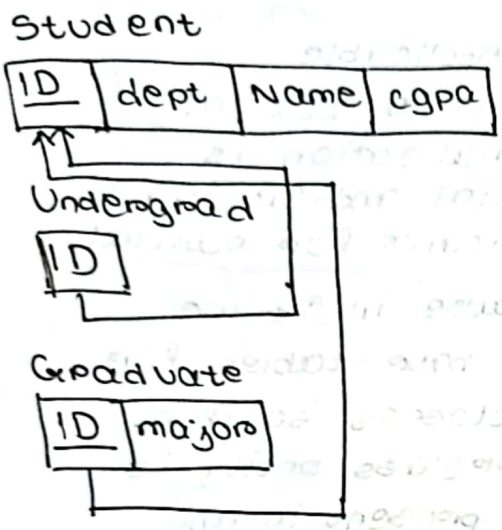
ID	dept	Name	cgpa	SType	major

8D

Student

ID	dept	Name	cgpa	U Type	major	G Type

(B) 8A



8B

Not Applicable

Specialization/generalization is partial and 8B is not applicable for partial. Because in 8B we only have tables for subclasses, so if a superclass entity does not belong to any of the subclasses the data for the entity will be lost.

8C:

Student

ID	dept	Name	cgpa	SType

8D

Student

ID	dept	Name	cgpa	UType	major	GType

(c)

8A

Student

ID	dept	Name	cgpa

TeachingAssistant

ID	course

ResearchAssistant

ID

8B

Not Applicable

The specialization/ generalization is partial and 8B is not applicable for partial. Because in 8B, we only have tables for subclasses so if a superclass entity is not present in any subclasses, the data for the entity is lost.

8C

Not Applicable

The specialization/generalization is overlapping and 8C is not applicable for overlapping. As only 1type attribute is not sufficient to store complete information if an entity belongs to multiple subclasses.

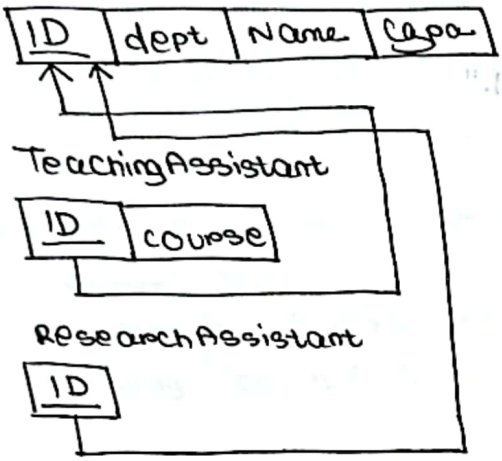
8D

Student

ID	dept	Name	cgpa	TType	course	RType

(8) ~~Stu~~ 8A

Student



8B

~~Stu~~

TeachingAssistant

ID	dept	Name	Cgpa	course

ResearchAssistant

ID	dept	Name	Cgpa

8C

Not applicable.

8C is not applicable for overlapping as only 1 type attribute is not enough to store information if an entity belongs to multiple subclasses.

8D

Student

ID	dept	Name	Cgpa	TType	Course	RType

CHAPTER 8 (MYSQL)

(1)(a) Select * ~~from~~
From TEAMS
where Name like "%.City.%"
Or Name like "A.%.";

(b) Select *
From PLAYERS
where Skill_level
(Select max(Skill_level)
From PLAYERS);

(c) Select *
From PLAYERS
where Skill_level
(Select Team_Name,
max(Skill_level)
FROM PLAYERS
group by Team_Name);

(d) Select Host_Name, Avg(host_score) as ~~average~~ host_average
FROM GAMES
group by Host_Team;

(e) Select Distinct City
From City;

(f) Select City, count(*)
From Teams
Order by City;

(g) Select *
FROM GAMES
where Guest_Score >= Host_Score
and Date between "01-01-2023" and "31-12-2023";

(h) Select Guest_Team, max(Guest_score) as guest_max
From GAMES
group by Guest_Team
having count(*) >= 3;

(i) Select *
from GAMES
Order by Date desc limit 10;

(j) Select *
From GAMES
Where Date =
(Select min(Date)
From GAMES)
Group by Host_Name;

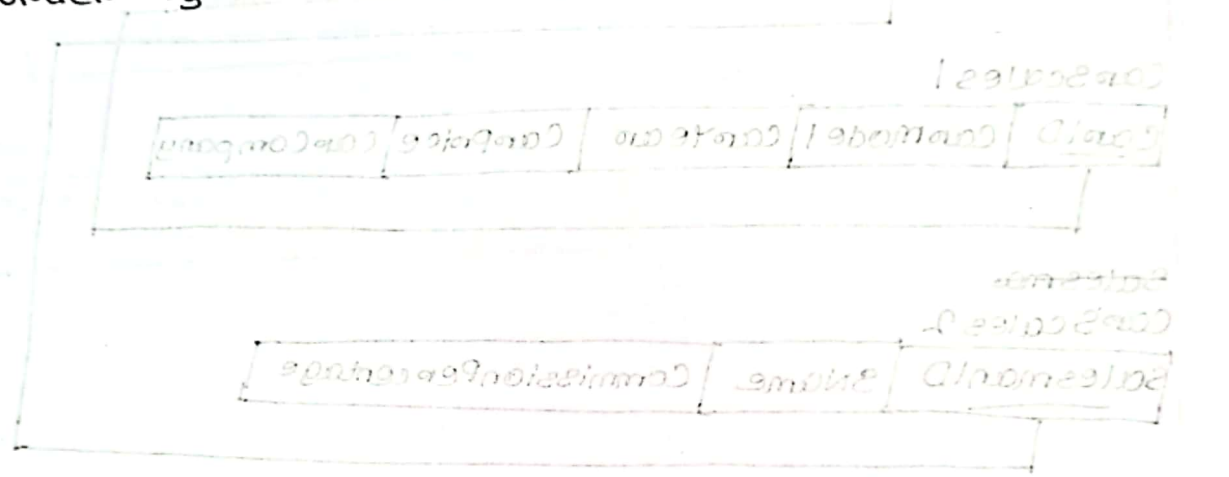
(k) Select *
FROM PLAYERS
where position = "centre forward"
and skill_level >=
(select skill_level
From PLAYERS
where position = "centre back");

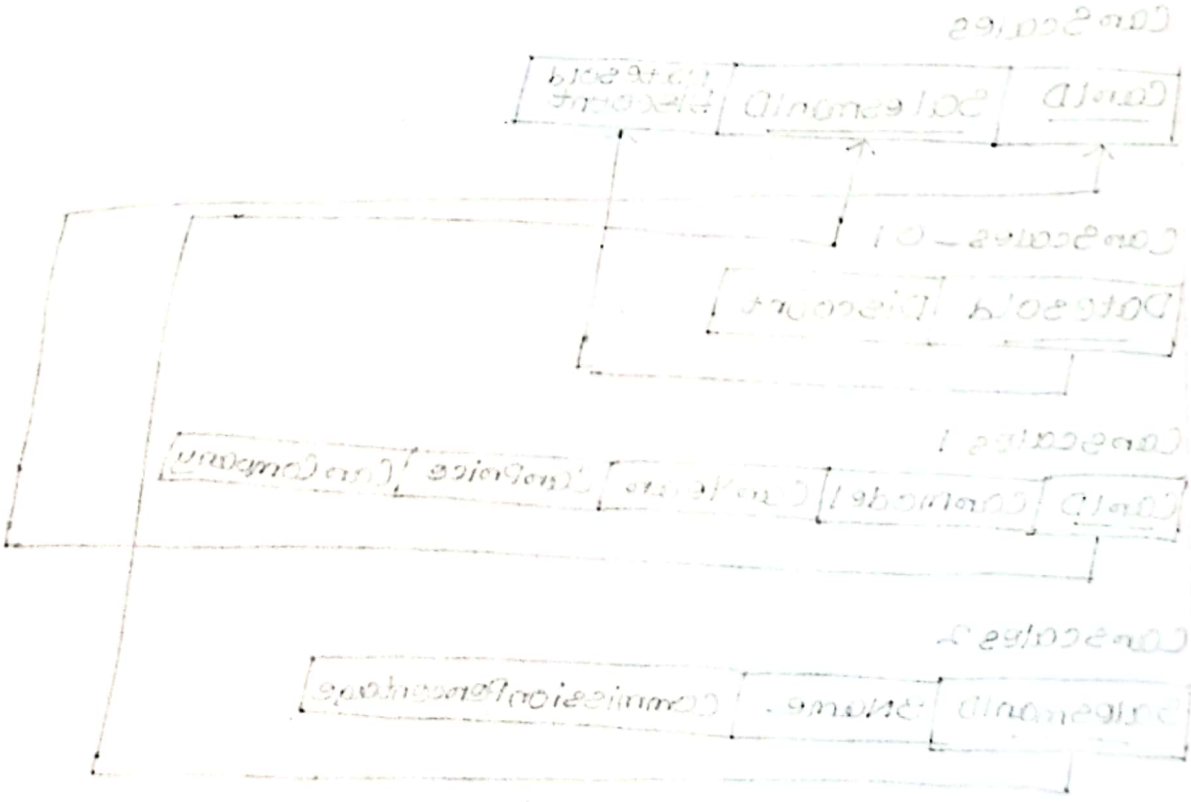
(2) (a) Select *
FROM TEAMS
INNER JOIN PLAYERS
ON TEAMS.Captain_Phone = PLAYERS.Phone;

(b) SELECT Games.Host_Team, Team1.city,
Games.Host_Score, &Host.Guest_Team,
Team2.city, Games.Guest_Score
From Games, Team Team1, Team Team2
where Games.Name = Team1.HOST_Team
and Games.Name = Team2.HoGuest_Team
and Date between "01-01-2022" and
"31-12-2022";

(c) SELECT Players.Name, Players.Position,
Players.Skill_level
FROM Players
INNER JOIN Team
ON Team.Name = Players.Team_Name
INNER JOIN Games
ON Games.Host_Team = Team.Name
and Skill_level =
(Select max(Skill_level)
From Players);

(d) Select Teams.Name, Players.Name, Teams.Captain-Phone,
Players_InjuryRecords.InjuryRecords
From Players, Teams, Players_InjuryRecords
where Players.Phone = Teams.Captain-Phone
and ~~Phone~~ Players.Phone = Players_InjuryRecords.Phone
Order by Teams.Captain_Phone ;



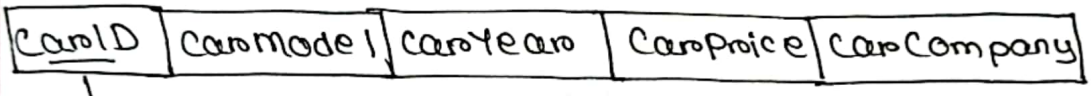


CHAPTER 10 (NORMALISATION)

(1) 2NF

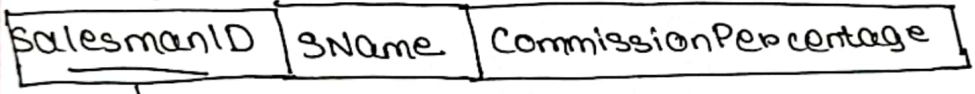
CarSales

CarScales1



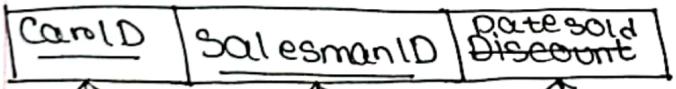
~~SalesMa~~

CarScales2

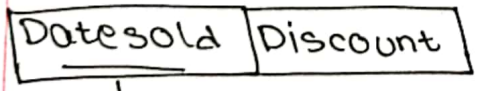


3NF

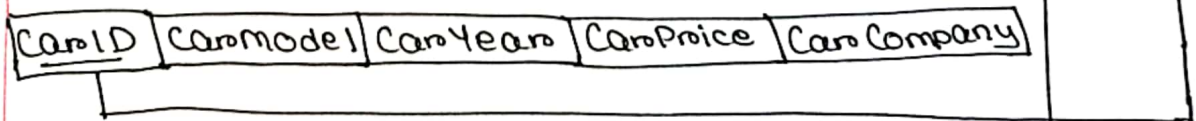
CarScales



CarScales_01

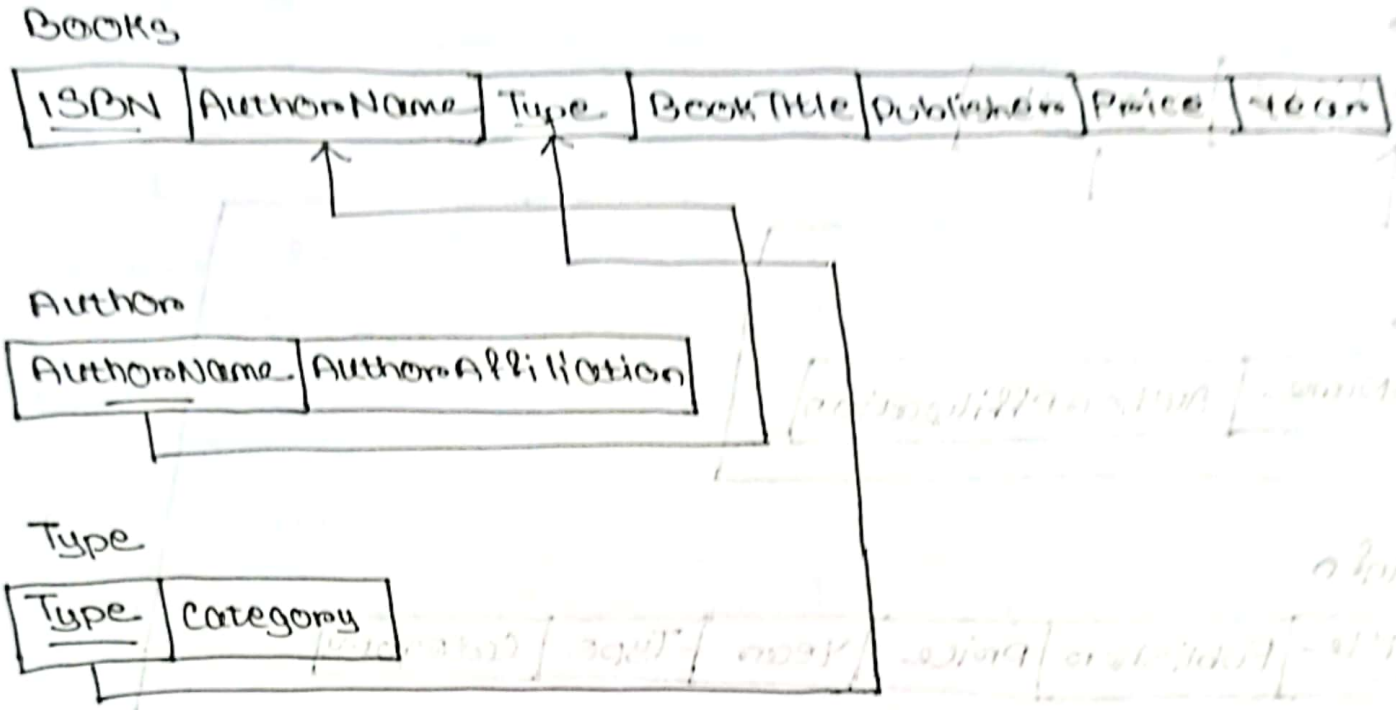


CarScales1



CarScales2

2) 3NF

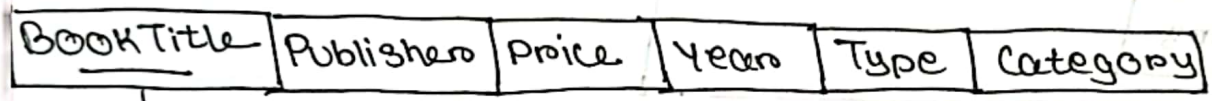

Books
ISBN
AuthorName
Type
BookTitle
Publishers
Price
Year
Author
AuthorName
AuthorAffiliation
Type
Type
Category

(3) 2NF

Books

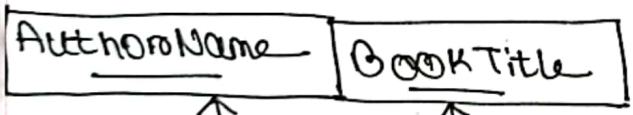
Author

Book_info

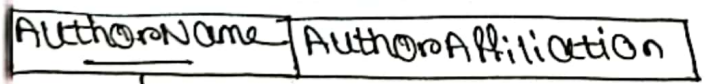


3NF

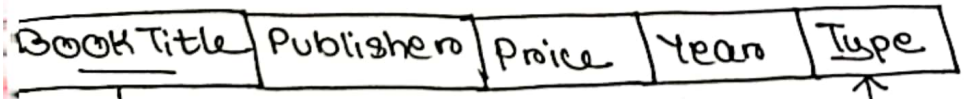
Books



Author



Book_info



Type_info

(৭) 2NF

Student_Project

StudentID	ProjectID	Score

Student_info

StudentID	SName	CGPA

Project_info

ProjectID	PName	Course Code	CTitle	Semester

3NF

Student_Project

StudentID	ProjectID	Score

Student_info

StudentID	SName	CGPA

Project_info

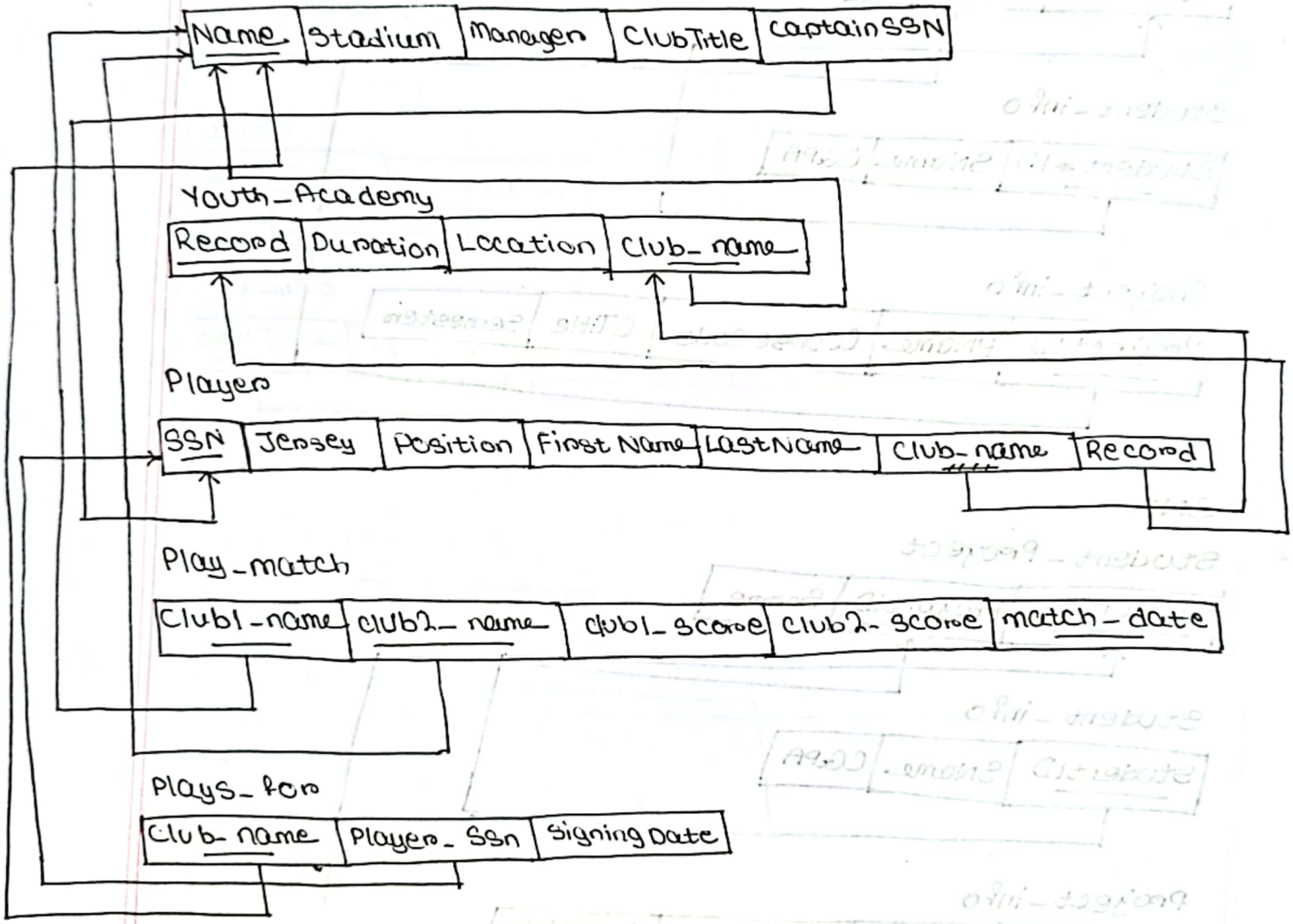
ProjectID	PName	CourseCode	Semester

Course

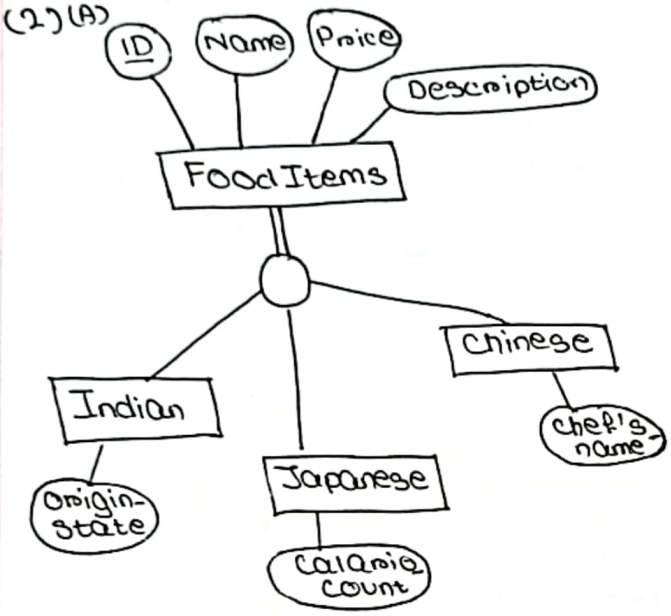
CourseCode	CTitle

Assignment 3 (NNC)

(1) Clubs



(2)(A)



(B) 8A

Food Items

ID	Name	Price	Description

Indian

ID	Origin_State

Japanese

ID	calorie_count

Chinese

ID	Chef's_name

8B

Indian

ID	Name	Price	Description	Origi-state

Japanese

ID	Name	Price	Description	calorie-count

Chinese

ID	Name	Price	Description	chef's-name

8C

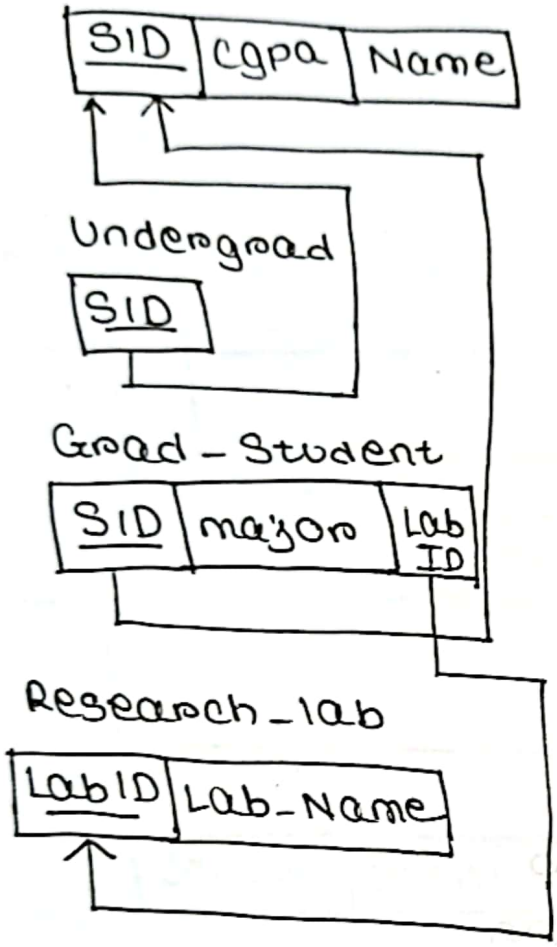
Not Applicable as ~~is~~ the generalisation/ specialisation is overlapping and there is only one type attribute so if an entity belongs to multiple subclasses, complete information cannot be stored.

8D

Food Items

ID	Name	Price	Description	Origin state	calorie-count	Chef's-name

(3) Student


SID
cgpa
Name
Undergrad
SID
Grad - Student
SID
major
Lab ID
Research_lab
LabID
Lab_Name

Assignment 4 (NNC)

(2) (i) 1NF. As there is no multivalued, composite or nested attributes.

(ii) Not in 2NF. As partial dependencies exists.

2NF

Computer_Repair

CompID	Engineer_ID	Date_Assigned	Date_Repaired

Engineer

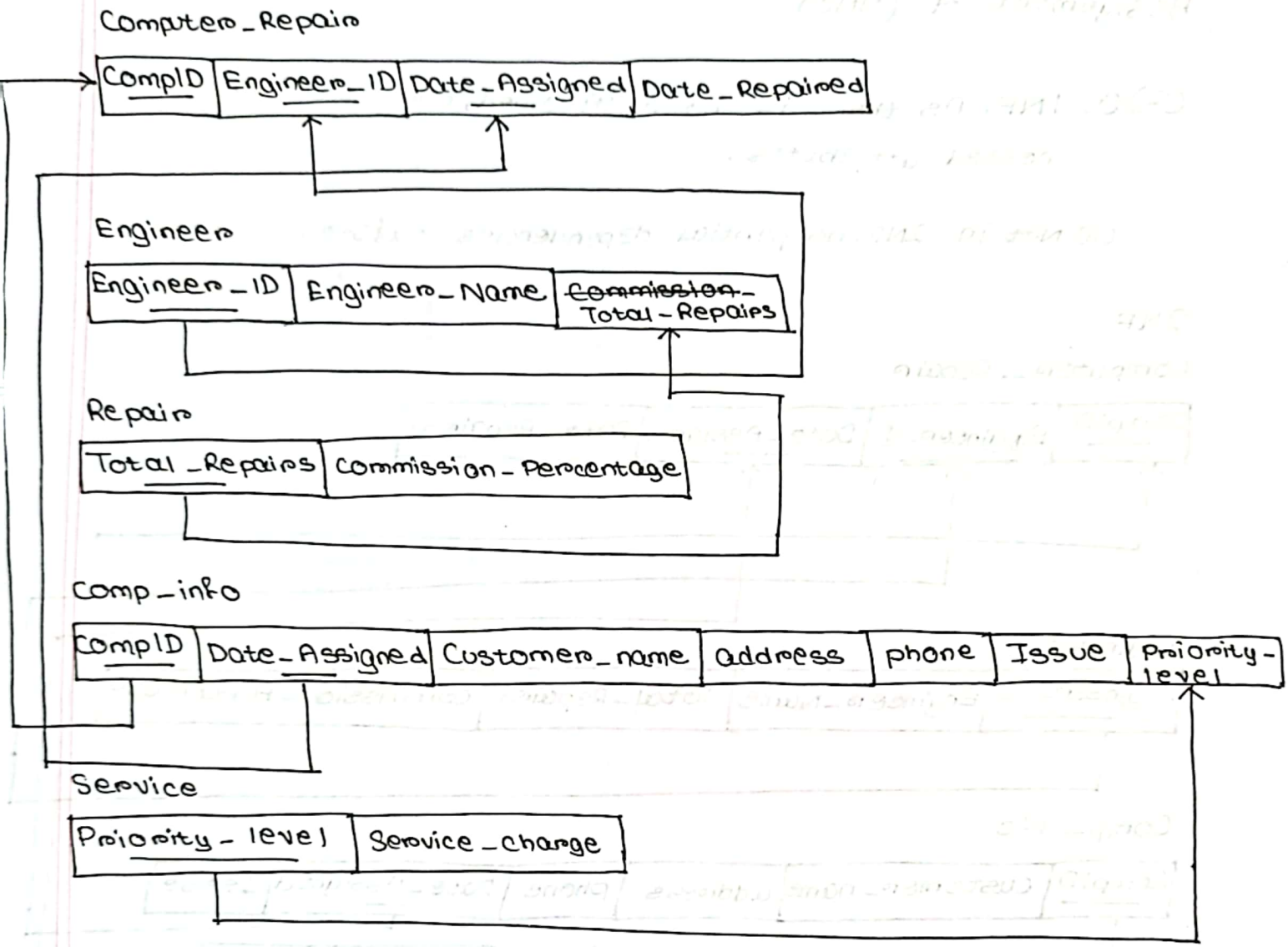
Engineer_ID	Engineer_Name	Total_Repairs	Commission_Percentage

Comp_info

CompID	Customer_name	address	phone	Date_Assigned	Issue

Priority_level	Service_charge

(iii) Not in 3NF. As transitive dependencies exist.


Computer_Repair
CompID
Engineer_ID
Date_Assigned
Date_Repaired
Engineer
Engineer_ID
Engineer_Name
~~Commission~~ Total_Repairs
Repair
Total_Repairs
Commission_Percentage
Comp_info
CompID
Date_Assigned
Customer_name
Address
phone
Issue
Priority_level
Service
Priority_level
Service_charge

CSE370 – Lab 7

(a) Select Heroes.Name, ~~Battles~~.avg(Battles.Hero-points)
From Heroes, Battles
where Heroes.Hero-id = Battles.Hero-id
group by Heroes.Hero-id;

(b) Select Heroes.Name, Villains.Name, Battles.Battle-date,
Battles.Battle-location
From Heroes, Villains, Battles
where Heroes.Hero-id = Battles.Hero-id
and Villains.Villain-id = Battles.villain-id
and Battles.Hero-points < Battles.villain-points;

(c) Select Villains.Name
From Heroes, Villains, Battles
where Heroes.Hero-id = Battles.Hero-id
and Villains.Villain-id = Battles.villain-id
and Battles.count(*) > 1;

(d) Select Villains.Name
from Villains, Battles
where Villains.Villain-id = Battles.villain-id
and villains.location != Battles.battle-location;

(e) Selupdate Battles.~~battle_point~~ se

```
Set Battles.battle_point = = 0.02 * Battles.battle_point
Inner Join Villains
On Battles.villain_id = Villains.villain_id
where Battles.Battle_date =
(select max(Battles.Date_date)
 From Battles);
```

(g) Select Villains.*

```
From Villains, Battles
where Villains.villain_id = Battles.villain_id
and Battles.~~battle~~villain_point =
(select ~~max~~min(Battles.~~battle~~villain_point)
 From Battles);
 Group by ~~He villains.villa~~ Battles.villain_id);
```

~~(g)~~ (f) Select Heroes.*

```
From Heroes, Battles
where Heroes.hero_id = Battles.hero_id
and Battles.hero_point =
(select ~~sum~~ max(Battles.hero_point)
 From Battles
 Group by Battles.hero_point);
```

(h) Select Villains.Name
From Villains, Battles
where Villains.villain_id = Battles.villain_id
~~and count (villBattles.villain_id)~~
group by ~~villBattles~~.villain_id
having count(*) > 1 ;

(i) Select Heroes.Name
From Heroes, Battles
where Heroes.hero_id = Battles.hero_id
group by Battles.~~battle location~~hero_id
~~having count(*) > 1 ;~~
having count (distinct (Battles.battle_location) > 1 ;

(j) Select Villains.Name
From Villains, Battles
where Villains.villain_id = Battles.villain_id
~~and~~ Order by Battles.villain_point desc limit 3;

(k) Select Heroes.Name
From Heroes, Battles
where Heroes.hero_id = Battles.hero_id
Order by Battles.hero_point limit 3;

370

LAB

Project — 5% + 15%

Assignment — 10% (4)

Attendance — 5%

LAB 1

MONDAY

DATE : 30/01/23

* char → specific length [length is known]

* varchar → length is unknown so large length is taken.

decimal(3, 2) → accept 3 digit
→ 2 digits are taken after decimal place

e.g. 4.56 , 4.567 → error

* Select * from ~~tbl~~ your_table → query [shows the

describe — shows the datatype and structure of table

select — retrive the entire ~~data~~ table.

BC : Single relation with only one type attribute

LAB-2

MONDAY

DATE : 06/02/23

CREATE, INSERT } Lab 1
MODIFY, DELETE, RETRIVE } Lab 2

Retrive query tells which particular data we want to retrive use

During facebook login, by we need only password and username, not the entire info of the user. For this, we cannot use select *. As * will give all the columns and thus information. So, by giving the name of particular columns after select, we can get that particular info only.

Select Name, Project_marks * Days_present * 5/17 as Total marks from Lab_Grades; — Aliasing

Sorting is done from left to right

select * from lab_grades order Name desc, submission_date.

→ Name is sorted in descending order at first and then submission_date is sorted in ascending order.

→ When no desc or asc is not mentioned, by default the table will be sorted in ascending order.

Regular expressions format the string.

start with 'a' → select * from lab_grades where name like 'a%'

ends with 'a' → '%a'

like '%a%' a%a → a is present anywhere in string

'%aa%' → 2 consecutive is present

[a---]
↳ length of string is 4 starting with a

LAB 3

MONDAY DATE:

Aggregate function → min, max, sum, count, average
() () () () ()

↓

when we consider multiple values and get one value.

group by → major wise আলাদা আলাদা group করবে

↓

first group, then apply condition

"where" cannot work with aggregate function as it works for rows, not for group by. "Having" works for group by. To work with "where", we have to use "where" condition b4 group by.

count(*) → counts the total number of rows

any/in/all → returns boolean (True/false) for a sub-query

any → any of the sub-query needs to be fulfilled to get True

all → all of the sub-query needs to be fulfilled to get True

Correlated Sub-queries

* Lab_Grades L2 — copied version of lab_grades
* Lab_Grades L1 — original version

L1 & L2 are imaginary instances

MONDAY DATE: 27/02/23

TAB.

primary or key — data is unique

RDBMS — Relational database management system.

ID	Name	Phone
1234		

Table 1

ID	Dept	CGPA
1234		

Table 2

For example, 1234 is the primary key. If we want to know about CGPA, we cannot find it from Table 1. So, we have to ~~find~~ map the primary key '1234' to Table 2; and if we find 1234 in Table 2, it is the foreign key.

Table 1 and Table 2 is handled ~~by~~ ~~differently~~ seperately.

Where there is foreign key — it is the child ~~of table~~.

Where there is primary key ~~which~~ — it is the parent ~~key~~.