CSE370 Quiz 01

Chapter 01

Total Mark -15

I. What is the difference between database and database management system? Mark 02

- DB is a organized collection of interrelated data
- DBMS is a software for managing data stored in a DB.

II. What does it means database system's data has implicit meaning? Mark 02

III. Write down some major disadvantages of managing data with File based approach? Mark 02

Mark 05

IV. Explain the following terms

1. Program-data independence.

2. Multiple Views

3. Meta data a set of data that describes ~~other~~ and gives info about other data.

4. Roll Back and OLTP

5. Actors

Mark 02

V. Explain the term

1. data redundancy

2. Indexing.

Mark 02

When Database management system may become unnecessary

DB is an organized collection of interrelated data.
↳ examples: MySQL, Oracle

Database: A database is an organized collection of data, generally stored and accessed electronically from a computer system.

A structured set of data held in a computer, especially one that is accessible in various ways.

It is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently. It is also used to organize the data in the form of a table, schema, views and reports, etc.

Using the database, you can easily retrieve, insert, and delete the information

---

The main purpose of database is to operate large amount of info by storing, retrieving and managing.

A database is a collection of information that is organized so that it can be easily accessed, managed and updated.

Data is organized into rows, columns and tables, and it is indexed to make it easier to find relevant information.

SQL - Structured Query Language

It is used to perform operations on the records stored in the database such as updating records, deleting records, creating and modifying tables, views, etc.

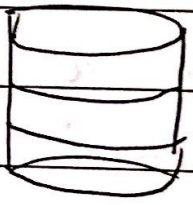
SQL is a query language, it is not a database.

To perform SQL queries, you need to install any database, for e.g. Oracle, MySQL

SQL is designed for managing data in a relational database management.

SQL is a database language.

A cylindrical structure is used to display the image of a database.



A relational database is a tabular database in which data is defined so that it can be reorganized and accessed in a number of different ways.

It is made up of a set of tables with data that fits into a predefined category.

It is easy to extend, and a new data category can be added after the original database creation without requiring that you modify all the existing applications.

Accessing the database: DBMS and RDBMS.

A DBMS is a type of software that allows you to define, manipulate, retrieve and manage data stored within a database.

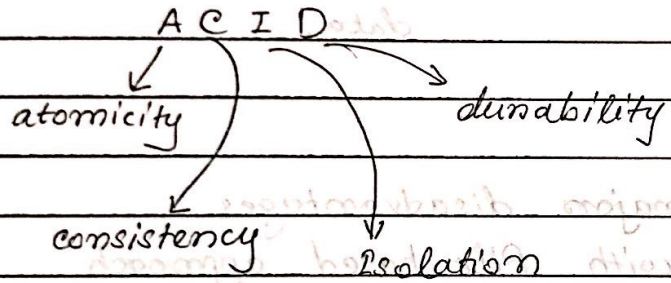
DBMS is also responsible for data transaction.

In a relational database, the structure of the table also belongs to the database.
It is also called a Schema of the database.

Implicit Properties of database → consistency, non-redundancy, ease of accessing data, ease of storing and retrieving in real world application.

Advantages of DBMS over File System.

1. Concurrency
2. Rich Query Set
3. Indexing
   → It is the process of ordering database entries. It helps to improve the speed of data retrieval operations.
   Indexing makes search quicker.
4. Transaction



5. Privileges
6. Remote Data access
7. Client Supportive
8. Database management tools → backup, replication
9. Scalability

Managing Data

File-based approach

Database approach

1. Share data betⁿ applications and users
2. manage redundancy
3. higher security and access control
4. create / manage relationships betⁿ data.

3Q. Write down some major disadvantages of managing data with file-based approach.

Ans 3:

- data redundancy (results in memory wastage)
- data inconsistency
- difficulty in accessing data / data isolation
- limited data sharing
- integrity problems
- atomicity problems (it must happen in its entirety or not at all)
- security problems
- concurrent access anomalies.
- program maintenance
- incompatible file-format

Advantages of file-oriented system

1. Backup
2. Compactness
3. Data retrieval
4. Editing
5. Remote Access
6. Sharing (shared among multiple users at a same time.)

A modern DBMS has the following properties:

1. Real-world entity (realistic)
2. Relation-based tables
3. Isolation of data and application
4. Consistency
5. Query Language
6. ACID properties (Atomicity, Consistency, Isolation, Durability)
7. Multiuser and Concurrent Access
8. Multiple views
9. Security

* One of the major advantages of a file-based approach is that the file-system has various access methods e.g. random, sequential, indexed

Advantages of DBMS

1. Backup
2. Multiple user interfaces
3. Data sharing
4. Controls redundancy (It stores all the data in a single database file)

Disadvantages of DBMS

1. Size (occupies large disk space and large memory to run efficiently)
2. Cost (requires a high-speed data processor and large memory to run DBMS software)
3. Complexity

Advantages of Using the Database Approach.

- Controlling redundancy
- Multiple users / Sharing of data
- Restricting unauthorized access to data
- Backup and recovery
- Multiple user interfaces

Key differences betⁿ Schema and Database

1. DB is an organized collection of interrelated data.
   Schema is a structural representation of DB.

2. DB updates frequently.
   Once you declare schema for a DB, it won't get frequently modified as it disturbs the organisation of data in the DB.

3. DB consists of a schema, records for the table.
   Schema contains the structure of tables and attributes inside the tables, their types and constraints.

4. The DML command specifies the changes in the data of a DB.
   The DDL (Data Definition Language) ~~spec~~ command specifies the schema for a DB.

Lecture - 01

- Data: known facts that can be recorded and have an implicit meaning.

- Database: A collection of related data.
It has the following implicit properties.
  (i) A database represents some aspect of the real world, sometimes called the miniworld.

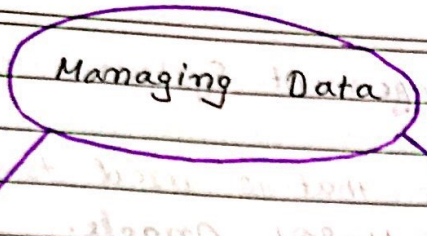
  (ii) A database is a logically coherent collection of data with some inherent meaning.

  (iii) A database is designed, built, and populated with data for specific purpose.

Example: Airline reservation system,
Students' registration system.

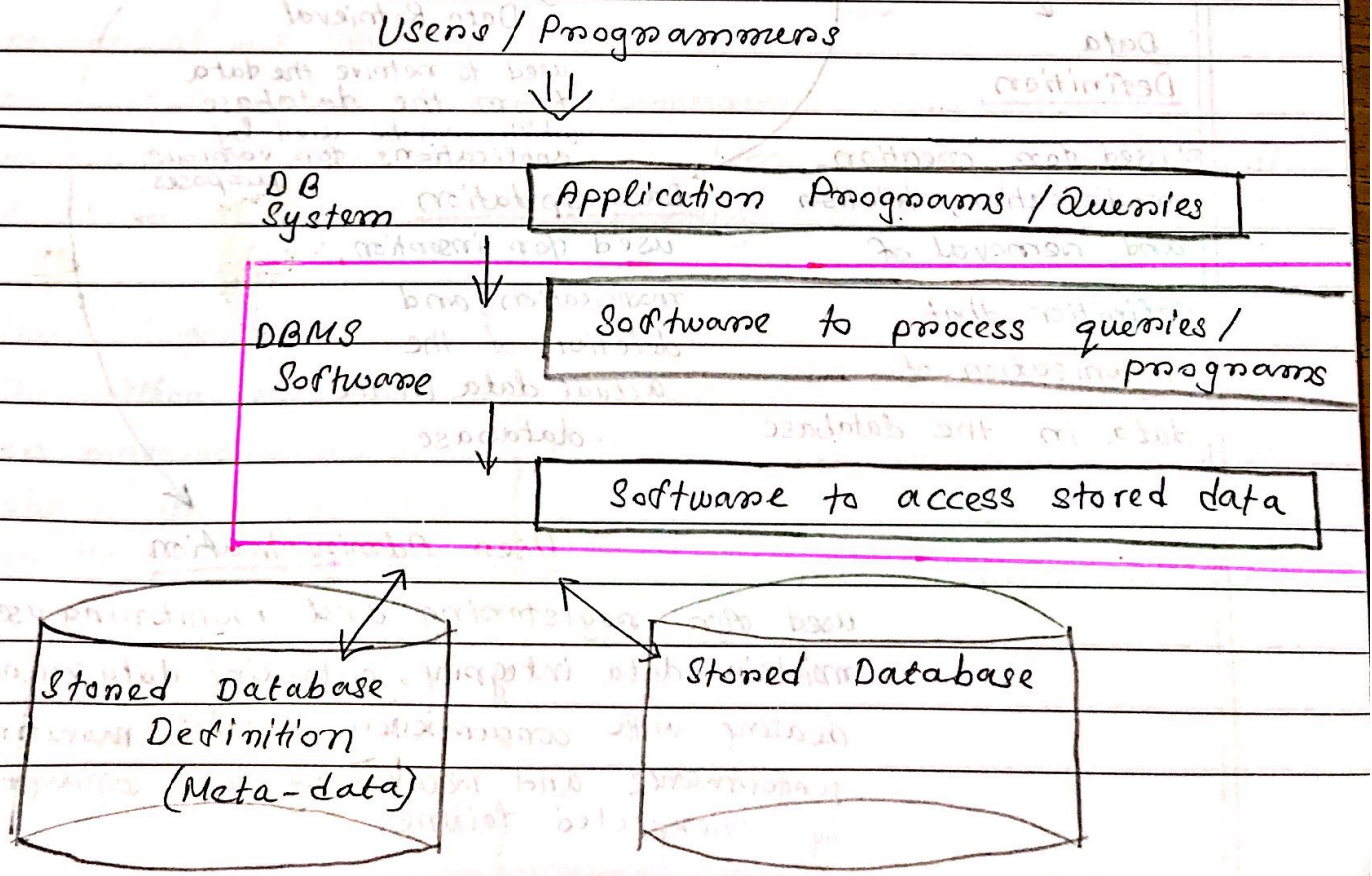
- Database Management System (DBMS):
A software package / system to facilitate the creation and maintenance of a computerized database. A DBMS is a software that is used to manage the database.

- Database System: The DBMS software together with the data itself.
Sometimes, the applications are also included.



File-based approach	Database approach
An approach that utilizes a collection of application programs which performs services to end-users (reports) Each program defines and manages its own data.	An approach that is collected and manipulated using specific software called DBMS, and many programs share this data.

Fig: DB System Environment



Database Management System

- DBMS is a software that is used to manage the database. For e.g. MySQL, Oracle.
- DBMS provides an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database and a lot more.
- It provides protection and security to the database.
  In the case of multiple users, it also maintains data.

DBMS allows users the following tasks:

Data Definition

⇒ used for creation, modification, ~~deletion~~ and removal of definition that organization of data in the database.

Data Updation

used for insertion, modification, and deletion of the actual data in the database

Data Retrieval

used to retrive the data from the database which can be used by applications for various purposes.

User Administration

used for registering and monitoring users, maintain data integrity, enforcing data security, dealing with concurrency control, monitoring performance and recovering info corrupted by unexpected failure.

Characteristics of DBMS

DBMS can reduce the complex relationship betⁿ data.
It contains automatic backup and recovery procedures
It is used to provide security of data.
It can view database from diff. viewpoints according to the requirements of user.
It can provide a clean and logical view of the process that manipulates data.
It contains ACID properties that maintain data in a healthy state in case of failure.

Advantages

1. Controls database redundancy
(It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.)

2. Data Sharing
(The authorized users of an organization can share the data among multiple users.)

3. Easily Maintenance
(due to centralized nature of the DB System)

Disadvantages

1. Size
(It occupies a large space of disks and large memory to run them efficiently)

2. Complexity
(DB System creates additional complexity and requirements)

3. Cost of Hardware and Software
(It requires a high speed of data processor and large memory size to run DBMS software)

4. Reduce time
(It reduces development time and maintenance need)

5. Backup
(It provides backup and recovery subsystems which create automatic backup of data from hardware and software failures and restores the data if required)

6. Multiple User Interface
(It provides diff types of user interfaces like graphical u.i., application program interfaces)

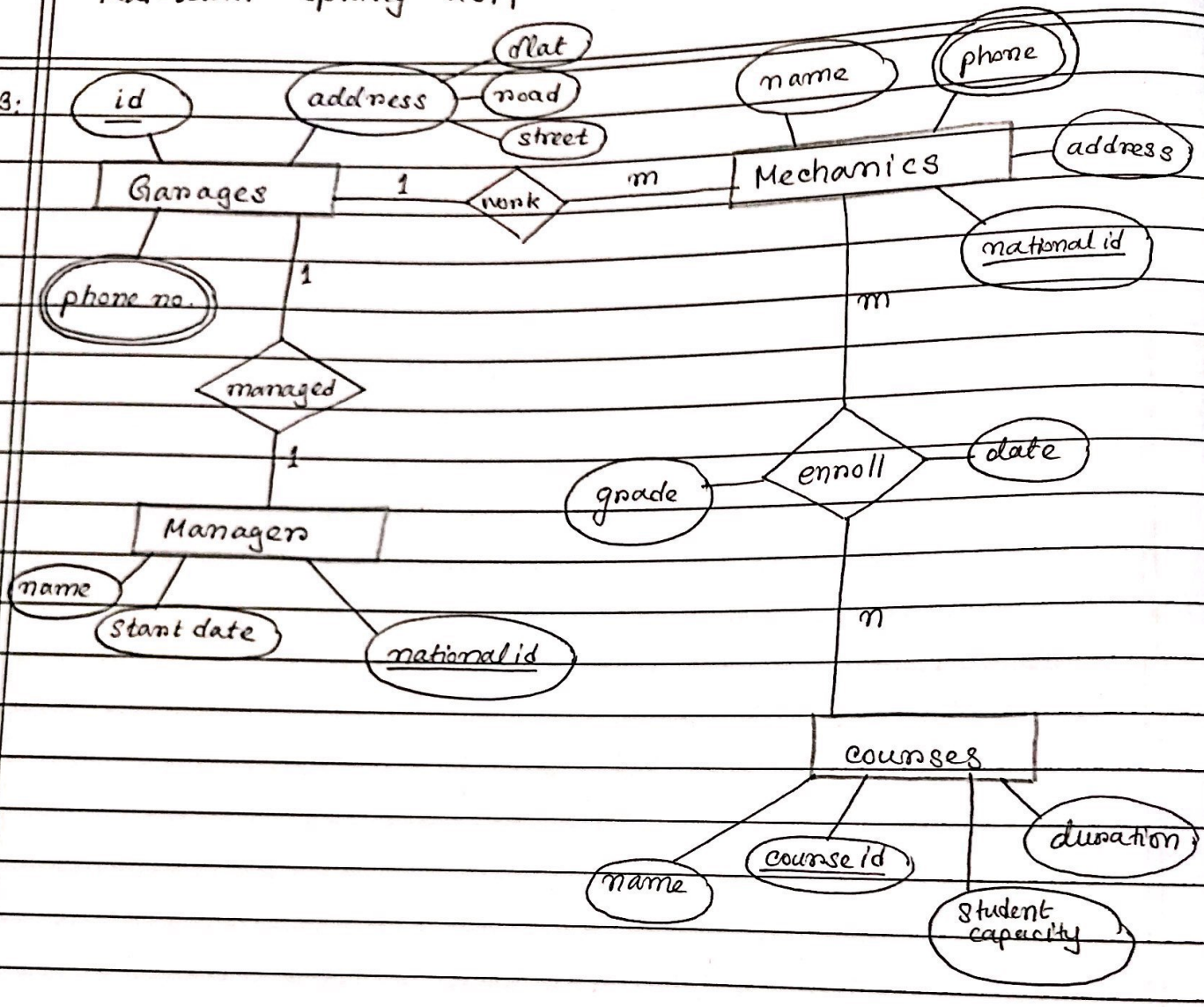
4. Higher impact of failure
(Failure is highly impacted the database because in most of the organization, all the data stored in a single database and if the database is damaged due to electric failure or database corruption then the data may be lost forever)

Differences betn DBMS and File System.

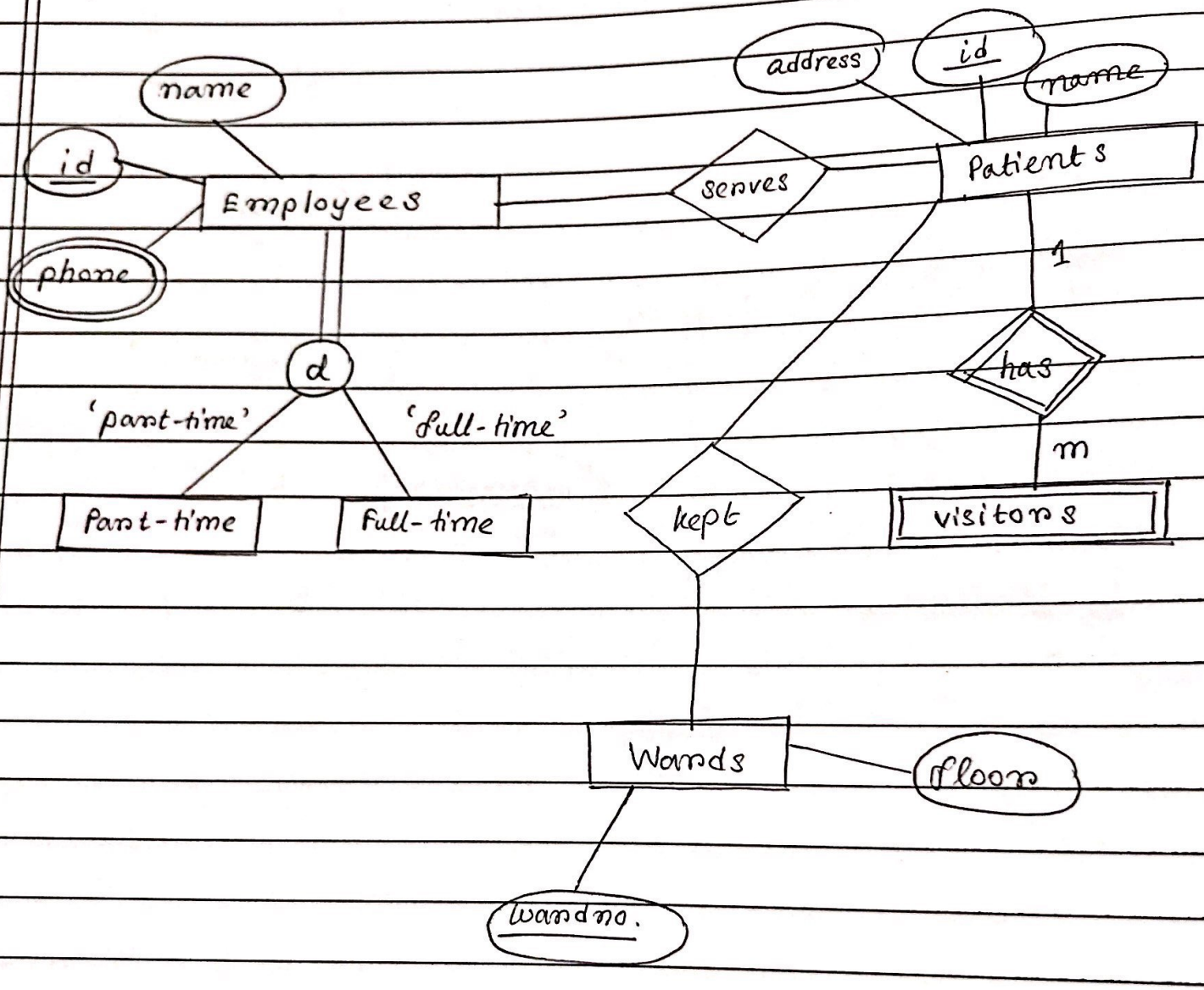
	DBMS	File System
1.	DBMS is a collection of data. In DBMS, the user is not reqd to write the procedures.	1. FS is a collection of data. The user has access to write the procedures for managing the database.
2.	DBMS gives an abstract view of data that hides the details.	2. FS provides the detail of the data representation and storage of data.
3.	DBMS provides a crash recovery mechanism i.e. DBMS protects the user from the system failure.	3. FS doesn't have a crash mechanism, i.e. if the system crashes while entering some data, then the content of the file will be lost.
4.	DBMS provides a good protection mechanism.	4. Difficult to protect file under the file system.
5.	DBMS contains a wide variety of sophisticated techniques to store and retrieve the data.	5. FS can't efficiently store and retrieve the data.
6.	DBMS takes care of Concurrent access of data using some form of locking.	6. In FS, concurrent access has many problems like redirecting the file while other deleting some info or updating some info.

Mid-term Spring 2019

Ans 3:



Q. Design an EER diagram for a Hospital Management System that keeps track of employees, patients and wards.



Lecture - 03

Data Modeling using the ER (Entity - Relationship) Model. This model is based on the real world.

Q. What is an entity type?

⇒ An entity type is specified as a collection of entities, having the same attributes.

* Entities with the same basic attributes are grouped on typed into an entity type *

For e.g. the entity type EMPLOYEE and PROJECT

Entity type typically corresponds to one or several related tables in the database. A characteristic or trait which defines or uniquely identifies the entity is called entity type.

Q. What is an entity set?

⇒ Each entity set type will have a collection of entities stored in the database called the entity set. An entity set is known as the set of all entities which share the same properties.

Q. What is an Extension of entity type?

⇒ An extension of an entity type is specified as a collection of entities of a particular entity type that are grouped into an entity set.

Q. What is an attribute?

⇒ An attribute refers to a database component. It is used to describe the property of an entity. An attribute can be defined as the characteristics of the entity.

Entities can be uniquely identified using the attributes.

It represent the instances in the row of the DB.

Cardinality

The number of times an entity of an entity set participates in a relationship set.

1. One to one
2. ~~No~~ Many to one
3. Many to many

Participation Constraint

1. Total participation (double line)
2. Partial " (single line)

ER Model

ER Model is used to model the logical view of the system from data perspective.

entity: an entity may be an object with a physical existence (car, person, employee, house) or an object with a conceptual existence (company, job, course)

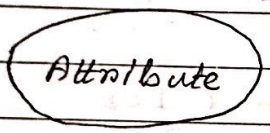
rectangle

Set of attributes in a database.

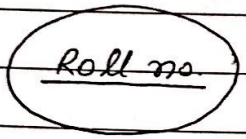
attributes: the properties which define the entity type. for e.g. (DOB, Name, address, roll no.)
It is represented by oval.

ellipses

(characteristics of the entity)

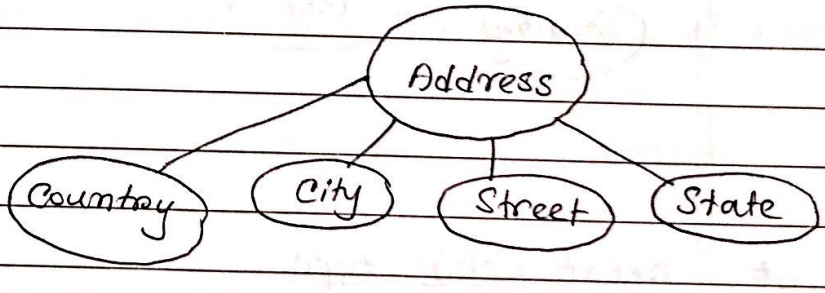


(i) **key attribute**
uniquely identifies each attribute

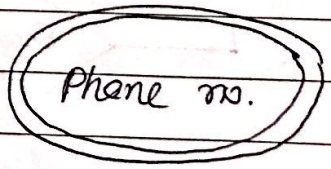


- oval with underlying lines

(ii) **Composite Attribute**
attribute composed of many other attributes.



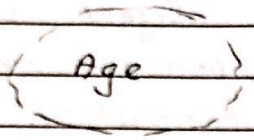
(iii) **Multivalued Attribute**
consists of more than one value for a given entity.



represented by double ov

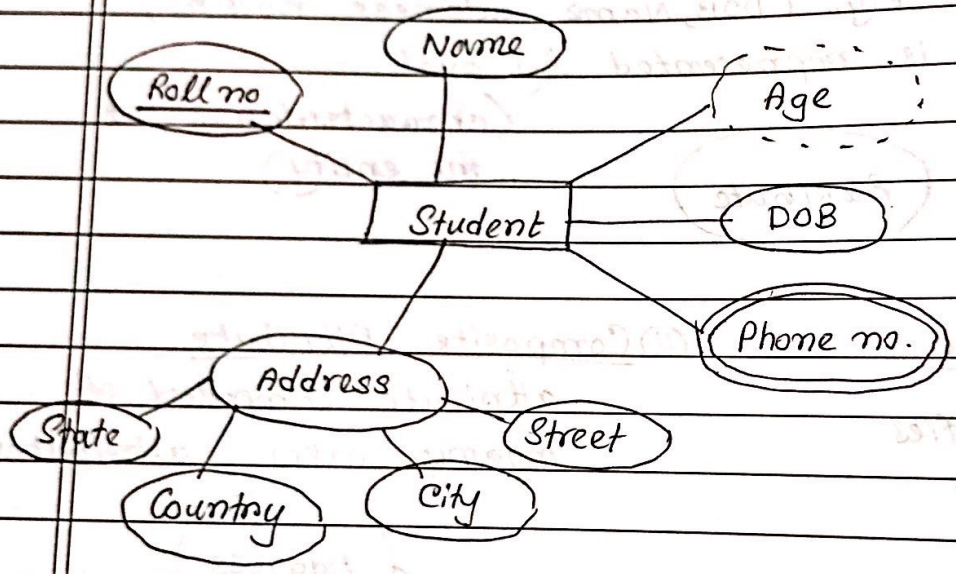
(iv) derived attribute

can be derived from other attributes.
Age (can be derived from DOB).



dashed oval

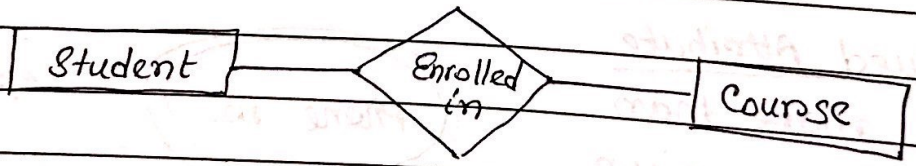
Example Entity type Student with its attributes



data flow is represented by straight lines.

relationship type

represents association betn entity types.

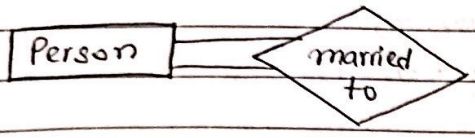


diamond

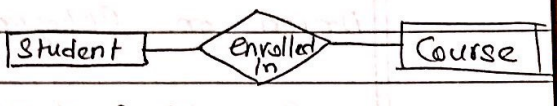
* Recursive relationship → a relationship type with the same participating entity in distinct roles.

Degree of relationship set

relationship represented by diamonds



1. Unary

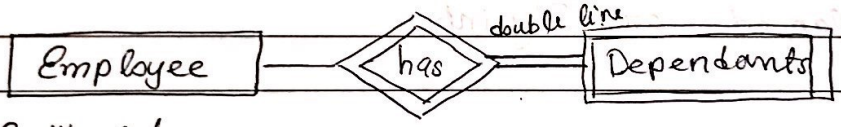


2. Binary

3. n-ary

When there are n entities participating in a relation, the relationship is called as n-ary relationship.

Weak entity type is represented by a double rectangle.
↳ for which key attribute can't be defined.



Weak Entity set

It is an entity set that doesn't have sufficient attributes to form a primary key.
The member of a weak e. set is known as a subordinate entity.

* The participation of weak entity type is always total *

An Identifying relationship is one betn a strong and a weak entity type, where the key of the strong entity type is reqd to uniquely identify instances of the weak entity type.

Enhanced ER Model

Diagramming Technique for displaying the Sub class and Super class. Specialization and Generalization; Union or Category; Aggregation, etc.

Specialized class (Sub class) — Generalized class (Super class)

IS-A Analysis

Sub-class entity inherits all attributes of super class.

There are 2 types of constraints on Sub-class relationship.

1. Total or Partial
2. Overlapped or Disjoint

An ER model is typically implemented as a database.
In a simple relational database implementation, each row of a table represents one instance of an entity type, and each field in a table represents an attribute type.

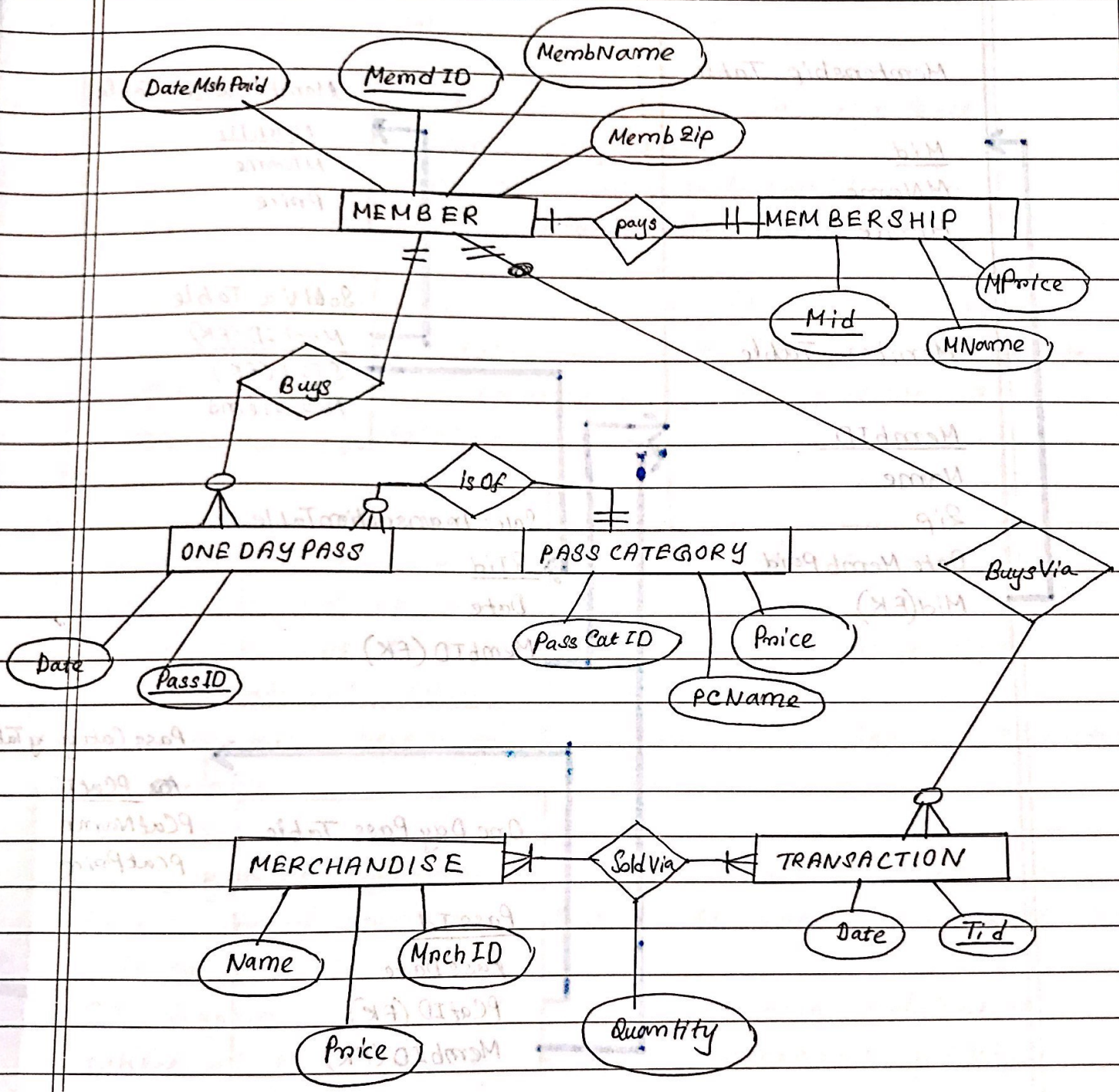
In a relational database, a relationship betn entities is implemented by storing the primary key of one entity as a pointer or "foreign key" in the table of another entity.

An entity-type is a category.

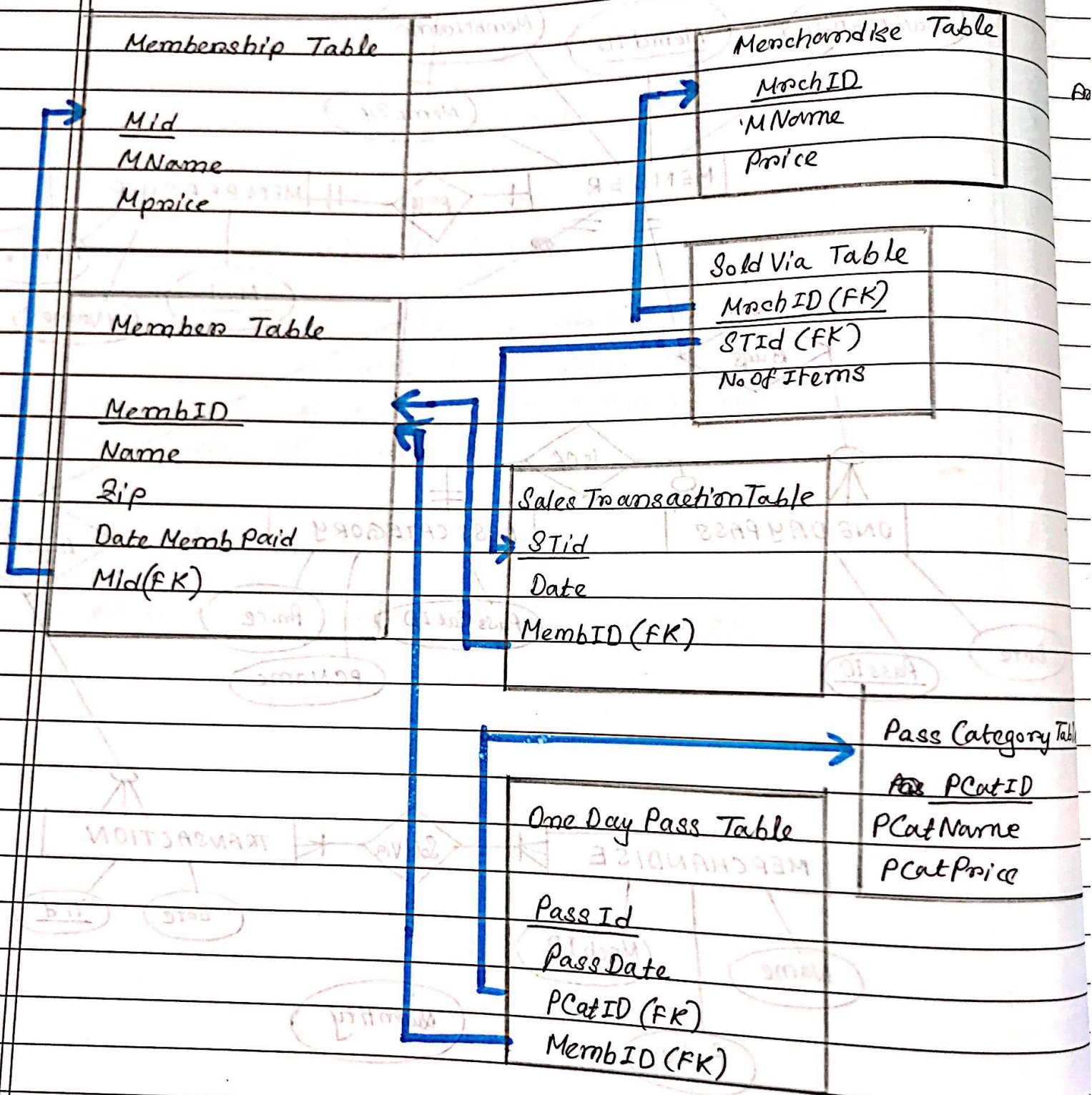
An entity is an instance of an entity-type.

ER Diagram Soln

Gym Fitness Database



Relational Schema for Gym Fitness Database.



June 27, 2019

Q. What is ER model in the DBMS?

Ans: The (entity relational) ER model is a high-level conceptual data model diagram.
It is based on the notion of real-world entities and the relationship between them.

It is best practice to complete ER modeling before implementing our database, as it helps us to analyze data requirements systematically to produce a well-designed database.

ER diagrams are a visual tool which is helpful to represent the ER model.
ER model was created as a uniform convention for relational database and network.
ER diagram helps us to explain the logical structure of databases.

ER diagrams provide a preview of how all your tables should connect, what fields are going to be on each table.
ER diagrams are translatable into relational tables which allows us to build databases quickly.

Q. What is the use of ER diagram?

⇒ For use in database design.

☑ An entity represents a real-world object or concept.

Q. What is an entity?

Ans: ⇒ An entity is a place, person, object, event, or a concept, which stores data in the database.
Every entity is made up of some 'attributes' which represent that entity.
Entity has a unique key.

examples
person: employee, student
place: building, store
object: machine, car, product
event: renewal, sale, registration
concept: account, course

Q. What is a weak entity? (an entity that depends on another entity)

Ans: ⇒ A weak entity does not have a key attribute.
It can be identified uniquely by considering the primary key of another entity.
For that, weak entity sets need to have participation.

Q. What is an attribute? property that describes an entity.

Ans: ⇒ It is a single-valued property of an entity-type or a relationship type.
It is represented by an ellipse.

Q. What is Relationship?

Ans: Association among two or more entities.
We identify it by verbs or verb phrases.

primary key ⇒ underlined
↳ identifiers

Types of Attributes

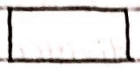
Simple — cannot be divided any further. It is called an atomic value.

Composite — possible to further breakdown
e.g. address, name → last
↓ ↓
first middle

Derived — does not include in the physical database. For e.g. age is not stored directly, it is derived from DOB. (dashed ellipse)

Multivalued — more than one value
e.g. mobile no., email address.
(double ellipses)

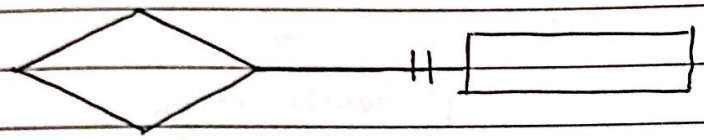
Comparison betⁿ Strong and Weak Entity

	Strong Entity Set 	Weak Entity Set 
1.	always has a primary key	does not have enough attributes to build a primary key
2.	represented by a rectangle symbol	double rectangle
3.	member of this set is called a dominant entity set	subordinate entity set
4.	relationship betⁿ two strong entity sets shown using a diamond symbol	relationship betⁿ one strong and a weak entity set shown using double diamond
5.	single connecting line	double connecting line

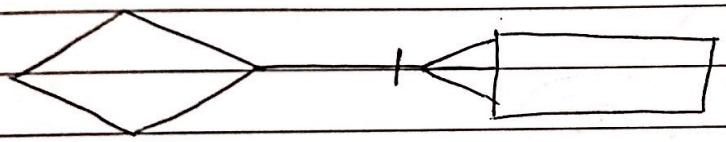
Entities and relationships can both have attributes.

Cardinality (the number of tuples in a relation) defines the numerical attributes of the relationship betⁿ two entities.

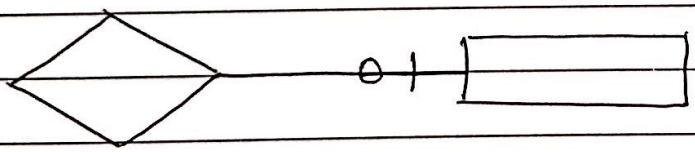
- one-to-one — 1:1
- many-to-one — m:1
- one-to-many — 1:m
- many-to-many — m:n



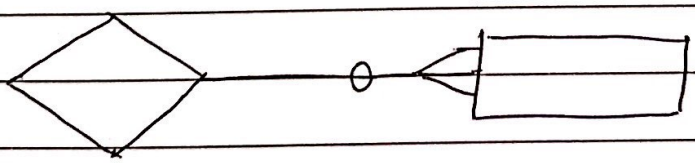
mandatory



mandatory many



optional



optional many

(EER)

Enhanced Entity-Relationship Modeling

Chapter 04:

EER → enhanced/extended ER

It was created to design more accurate database schemas.

It is a high-level / conceptual data model incorporating extensions to the original ER model; used in the design of databases.

It was developed to reflect more precisely the data properties and constraints that are found in more complex databases.

The EER model includes all of the ^ concepts modeling introduced by the ER model.

Additionally it includes:

These are fundamental to conceptual modeling:

- subclasses, superclasses (is-a)
- specialization, generalization
- attribute and relationship inheritance
- categories (union types) / category or union type
  {used to represent a collection of objects that is the union of objects of different entity types.}

The additional EER concepts are used to model applications more completely and more accurately.

• EER diagrams represent these additional subgroupings called Subclasses or Sup Subtypes.

Subclass and Superclass

Entity type Y is a subtype (subclass) of an entity type X if and only if every Y is necessarily an X

* A subclass entity inherits all attributes and relationships of its superclass entity. This property is called the attribute and relationship inheritance.

A subclass entity may have its own specific attributes and relationships (together with all the attributes and relationships from the superclass.)

Example: Vehicle superclass
Subclasses Car and Truck

The common attributes betn a car and a truck would be part of the superclass, while the attributes specific to a car or a truck (truck type, max payload etc.) would make up two subclasses.

* The subclass entity member is the same entity in the superclass but in a distinct specific role.

• It is not necessary that every entity in a superclass be a member of some subclass.

Subclass → subgroupings of entities that are meaningful.

An entity type has numerous subgroupings of its entities that are meaningful and need to be represented explicitly because of their significance to the database application.

We call each of these subgroupings a subclass of the entity type; and the entity type is called the superclass for each of these subclasses.

The type of an entity is defined by the attributes it possesses and the relationship types in which it participates.

A superclass or subclass represents a collection of entities of the same type and also describes an entity type.

Superclass → An entity type that represents a general concept at a high level.

Subclass → An entity type that represents a specific concept at lower levels.

Specialization

It is the process of defining a set of subclasses of a superclass.

The set of subclasses is based upon some distinguishing characteristics of the entities in the superclass.

- successive specialization

- Attributes of a subclass are called specific or local attributes.

- Subclass can participate in specific relationship types.

Therefore, subclasses can define:

- Specific attributes
- Specific relationship types

* difference

In a 1:1 relationship, two distinct entities are related, whereas in a superclass / subclass relationship the entity in the subclass is the same real-world entity as the entity in the superclass but is playing a specialized role.

Specialization: (top-down) A means of identifying sub-groups within an entity set which have attributes that are not shared by all the entities.

Generalization: (bottom-up) Multiple entity sets are synthesized into a higher-level entity set based on common features.

Generalization

It is the reverse of the specialization process

Several classes with common features are generalized into a ~~subclass~~ superclass.

- The process of making a superclass from a group of subclasses, that have common properties, is called Generalization.
- The process of making subclasses from a general concept is called Specialization.

Generalization	Specialization
is a bottom-up conceptual synthesis process.	is a top down conceptual refinement process.

Constraints on S and G

There are (3) constraints that apply to a specialization / generalization.

1. Membership constraints
2. Disjointness constraints
3. Completeness constraints

1. Membership constraints

a) condition defined → predicate-defined (or condition defined) subclass

Condition is a constraint that determines subclass members.

attribute-defined specialization
e.g. JobType

b) user-defined

If no ~~member~~ condition determines membership, the subclass is called user-defined.

Membership in such a subclass is determined by the ~~users~~ database users when they apply the operation to add an entity to the subclass; hence, membership is specified individually for each entity by the user, not by any cond that may be evaluated automatically.

2. Disjointness Constraint → specifies that the subclasses of the specialization must be disjoint.

(d) disjoint

An entity can be a member of at most one of the subclasses of the specialization

e.g. postgrad, undergrad
you cannot be both

{OR}

(o) overlapping

Same entity may be a member of more than one subclass of the specialization.
e.g. staff, student
some people are both

{AND}

3. Completeness Constraint

Total	Partial
Total specifies that every entity in the superclass must be a member of some / atleast one subclass in s/g.	Partial an entity not to belong to any one of the subclasses.
double line	single line

* disjointness and completeness constraints are independent.

Hence, we've four types of S/G:

- disjoint, total
- disjoint, partial
- overlapping, total
- overlapping, partial

Generalization usually is total because the superclass is derived from the subclasses.

A subclass itself may have further subclasses specified on it, forming a hierarchy or a lattice of specializations.

A specialization hierarchy has the constraint that every subclass participates as a subclass in only one superclass/subclass relationship, i.e each subclass has only one parent, which results in a tree structure.

---

* Hierarchy has a constraint that every subclass has only one superclass (called Single Inheritance)
⇒ results in a tree structure.

Specialization lattice → a subclass can be a subclass in more than one superclass/subclass relationship.

* In a lattice, a subclass can be a subclass of than one superclass. (called Multiple

* In a lattice or hierarchy, a subclass inherits attributes not only of its direct superclass, but also of all its predecessor superclasses, all the way to the root.

* A subclass with more than one superclass is called a shared subclass (multiple inheritance)

Leaf node is a class that has no subclasses of its own.

Notice:- the existence of at least one shared subclass leads to a lattice (and hence to multiple inheritance); if no shared subclasses existed, we would have a hierarchy rather than a lattice.

* For modeling a single class/subclass relationship with more than one superclass, where the superclasses represent diff. entity types.

In this case, the subclass will represent a collection of objects that is a subset of the UNION of distinct entity types; we call such a subclass a union type or a category.

Subclass which represent objects collection of distinct entity types is classified as both union type and intersection type.

Quiz Question for EER Diagram **Chapter 04** **CSE370 Database System**

I. An entity may exist in the database only by being a member of the superclass or a subclass
a) True b) False c) Partially True

II. Specialization is the process of defining a set of subclasses of a superclass.

III. The set of subclasses is based upon some distinguishing characteristics of the entities in the superclass

IV. Attributes of a subclass are called specific / local attributes.

V. The subclass can not participate in specific relationship types but superclass can.
a) True b) False c) Partially True (The subclass also can)

vi. Generalizated superclass is opposite of specialized subclass
a) True b) False c) Partially True

vii. We can view {CSE370, CSE320} as a specialization of Teacher a) True b) False

viii. we can view CSE370 as a generalization of Teacher and Students a) True b) False

ix. Subclass by a specific condition within a superclass called specialized subclasses

x. Subclass by a specific condition displayed within diagram by special

xi. Subclasses in a specialization membership inherits same attribute of the superclass called _____ specialization

xii. If no condition determines membership, the subclass is called user-defined.

xiii. Two basic constraints can apply to a specialization or generalization are disjointness constraint and completeness constraint.

xiv. An entity can be a member of at most one of the subclasses of the specialization called disjointness constraint (disjoint)

xv. _____ Total specifies that every entity in the superclass must be a member of some subclass in the specialization/generalization. Shown in EER diagrams by double line

xvi. A subclass can be subclass of more than one superclass called shared subclasses

xvii. In a lattice or hierarchy, a subclass inherits attributes not only of its direct superclass, but also of all its predecessor superclasses

xviii. A subclass with more than one superclass is called a shared subclass ✓

xix. In specialization, start with an entity type and then define subclasses of the entity type by successive specialization called a top down conceptual refinement process

xx. In generalization, start with many entity types and generalize those that have common properties Called a bottom up conceptual synthesis process

xxi. Superclasses can represent different entity types Such a subclass is called category or union type

Chapter 2: Database System Concepts and Architecture

Data Model → set of concepts to describe structure of a DB

A set of concepts / A collection of concepts used to describe the structure of a database, and certain constraints that the database should obey.

Data model helps to achieve data abstraction.

Data Abstraction

The suppression of the details of data organization and storage and the highlighting of essential features for better understanding.

Categories of Data Models (C P R)

- Conceptual Data models (high-level, semantic data models)
- Physical Data models (low-level, internal d. models)
- Representational Data model (implementation)

(meant for comp. specialists)

- **conceptual data models**: provide concepts that are close to the way many users perceive data.
  (also called entity-based or object-based data models)
  e.g. entities, attributes
- **physical data models**: provide concepts that describe details of how data is stored in the computer.
  e.g. record formats, access paths
- **representational data models**:
  example: relational data model

[skeleton of a database]

[specified during DB design]

Database Schema (intension): The structural description of a database.

It includes descriptions of the DB structure, data types, and the constraints on the DB.

- **Schema Diagram**: An illustrative display of a DB Schema.
  (displays only some aspects of schema)
- **Schema Construct**: A component of schema or an object within the schema, e.g. Student, Course

Database State (extension): contains snapshot of the database

The actual data stored in a database at a particular moment in time.

This includes the collection of all the data in the database.

It is also called the current set of occurences or instances or snapshot in the database.

Every time we insert, delete or change the value of ~~an or~~ a data item in a record, we change one state of the DB into another state.

- Change occurence: Schema changes very infrequently.
  Instance/State changes every time DB is updated
- Initial state: empty → schema
  always have some data → state

Important characteristics of the database approach.

(1) Insulation of programs and data ⇒ program-data independence

(2) support of multiple user views

(3) use of a catalog to store DB schema.

The three-schema architecture was proposed to help achieve and visualise DB characteristics.

goal? ⇒ To separate the user applications and the physical DB.

It defines DBMS schema at (3) levels. ICE Schema

Internal Schema	Conceptual Schema	External Schema
Physical Schema/ Internal Schema at internal level to describe physical storage structure and access paths (e.g. indexes)	at conceptual level	at external level

The 3 schemas are only descriptions of data; the only data that actually exists is at the physical level.

Process of transforming requests and results bet^n levels are called mappings.

If a DB System is not multi-layered, then it becomes difficult to make any changes in the database system.

Data Independence

The ability to change schema at one level of the DB System without having to change the schema at the next higher level.

Two types of DI: → logical DI
→ physical DI

- Logical Data Independence (→ CS) ⇒ The capacity to change the conceptual schema without having to change the external schemas and their associated application programs.

- Physical Data Independence (→ IS) ⇒ The capacity to change internal schema without having to change the conceptual schema. Hence, the external schema need not be changed as well.

DBMS Languages → DDL
→ DML

DDL

Data Definition Language

a language that is used to define database schemas.

- DDL is not classified further.

Commands: DROP
COMMENT
RENAME

DML

Data Manipulation Language

a language that is used to manipulate data. (retrieve, insert, delete, modify)

- high-level DML e.g. SQL
- low-level DML

SELECT
INSERT
DELETE

Classification of DBMS

Criterias used to classify DBMSs :-

- data model
- number of users
- number of sites
- cost
- purpose

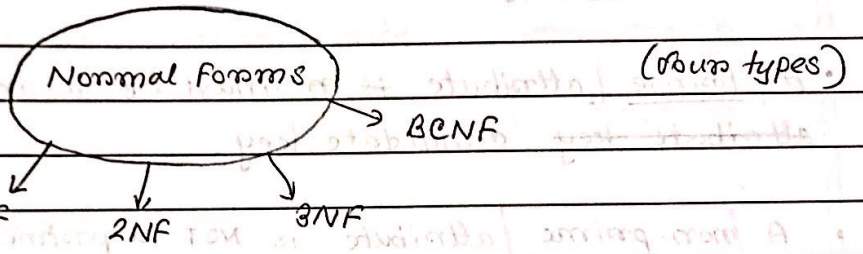
Normalization

MIFTA SINTAHA

* Normalization is the process of efficiently organising the data in the database.
→ used to minimize the redundancy from a relation or set of relations.
→ used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.

Normalization divides the larger table into the smaller table and links them using relationship.

The normal form is used to reduce redundancy from the database table.



Normal Form	
1NF	A relation is 1NF if it contains an atomic value
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
4NF	A relation will be in 4NF if it is Boyce Codd normal form and has no multi-valued dependency.
5NF	A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.

There are two goals of the normalization process

- The first is to eliminate redundant data.
  → Redundant data is defined as storing the same data in more than one table.
- The second is to ensure that data dependencies make sense by having only related data stored in the same table.

Both of these are imp. as they reduce the amount of space a database consumes and ensures that data is logically stored.

- A **Prime** attribute ~~is a~~ must be a member of some ~~attribute key~~ candidate key.
- A **non-prime** attribute is NOT a prime attribute → that is, it is not a member of any candidate key.

1NF ⇒ There are no duplicated rows in the table.
⇒ Each cell is single-valued or **atomic**
(one field cannot have more than one value)

2NF ⇒ A table is in 2NF if it is in 1NF and if all (non-prime) non-key attributes are fully functionally dependent on all of the key. (key determinant of non-prime attributes) fully func dependent.

3NF ⇒ A table is in 3NF if it is in 2NF and if it has no transitive dependencies. ($x \xrightarrow{\text{determines}} y$, $y \to z$, $x \to z$) ✗ not allowed)

In $x \to y$, and $y \to z$, with x as the PK, we consider this a problem only if y is not a candidate key. When y is a candidate key, there's no problem with transitive dependency.

rows → records
columns → attributes

Relational Model

Relational Model was proposed by E.F. Codd to model data in the form of relations or tables.

After designing conceptual model in of DB using ER diagram, we need to convert the conceptual model in the relational model using ~~RDBM~~ RDMBS languages like Oracle SQL, MySQL.

Q. What is Relational Model?

Ans: Relational Model represents (how) data is stored in Relational Databases.
A relational database stores data in the form of relations (tables).

STUDENT — relation

Roll_No	Name	Address	Age	Phone	← attributes
1	Roe	DELHI	18	...	
2	Sue	DHAKA	19	...	
3	Mill	TOKYO	20	...	
4	Bay	PARIS	19	...	

Attributes are the properties that define a relation
e.g. Roll_No, Name.

Tuple → (a single row of a table, which contains a single record for that relation)
→ each row in the relation is called a tuple.
The above relation contains (4) tuples, one of which is :→ 1 Roe DELHI 18 ...

(4 rows)

A relational database schema is the tables, columns and relationships that link together the components into a database. It helps us to organize and understand the structure of database.

- **Relational Schema**
  If a schema has more than 1 relation, it is called Relation Schema.
  A relational schema represents name of the relational relation with its attributes e.g.
  STUDENT (Roll_No, Name, Address, Phone, and Age)
  is relational schema for STUDENT.

(describes the relation name, attributes, and their names)

- **Relation instance** → (finite set of tuples)
  The set of tuples of a relation at a particular instance of time is called as relation instance.
  ⇒ It can change whenever there is insertion, deletion or updation in the DB.
  ⇒ RI do not have duplicate tuples.

- **Degree**: the number of attributes in the relation is known as degree of the relation.
  The STUDENT relation defined above has degree 5.

numbers

- **Cardinality**: the number of tuples in a relation is known as cardinality.
  Here, cardinality is 4.

- **Column**: It represents the set of values for a particular attribute.

- NULL values → The value which is not known or unavailable is called NULL value.
  It is represented by blank space.

Constraints in Relational Model

While designing Relational Model, we define some conditions which must hold for data present in database are called Constraints.
These constraints are checked before performing any operation (insertion, deletion and updation) in DB. If there's a violation in any of the constraints, operation will fail.

Domain Constraints : These are attribute level constraints. An attribute can only take values which lie inside the domain range.
e.g. if a constraint AGE > 0 is applied on STUDENT relation, inserting negative value of AGE will result in failure.

Key Integrity

Every relation in the database should have atleast one set of attributes which defines a tuple uniquely. Those set of attributes is called KEY e.g. Roll_No in STUDENT is a key.
No two students can have same roll number.
So a key has two properties:

① It should be unique for all tuples
② It can't have NULL values

Key constraints are also called Entity Constraints

Referential Integrity (work on the concept of Foreign Keys)

When one attribute of a relation can only take values from other attribute of same relation or any other relation, it is called R.I.

Suppose, we've 2 relations

STUDENT

Roll_No	Name	Age	Phone	Branch_Code
				CS
				CS
				ECE
				IT

BRANCH

Branch_Code	Branch_Name
CS	
IT	
ECE	
CV	

Branch-Code of Student can only take values which are present in Branch-Code of BRANCH which is called the RI constraint.

The relation which is referencing to other relation is called REFERENCING RELATION (here → STUDENT)

RI constraint states that if a relation refers to a key attribute of a different or same relation, then that key must exist.

and the relation to which others relations refer is called REFERENCED RELATION (here → BRANCH)

* Every relation has some conditions that must hold for it to be a valid relation.

Relational Integrity Constraints

- Domain constraints
- Key constraints
- Referential Integrity Constraints

Anomalies

An anomaly is an irregularity, or something which deviates from the expected or normal state.

When designing databases, we identify 3 types of anomalies: Insert, Update, and Delete.

- **Insertion Anomaly in Referencing Relation**

We can't insert a row in REFF REFERENCING RELATION if referencing's attribute value is not present in referenced attribute value.

- Deletion / Updation Anomaly in Referenced Relation

We can't delete or update a row from referenced relation ~~relation~~ if the value of referenced ~~relation~~ is used in the value of referencing attribute. attribute

If value not used by referencing relation, we can delete the row from the referenced relation.

It can be handled by:

- ON DELETE CASCADE
- ON UPDATE CASCADE

SUPER KEYS

Any set of attributes that allow us to identify unique rows (tuples) in a given relation are known as superkeys.

Out of these super keys we can always choose a proper subset among these which can be used as a primary key. Such keys are known as Candidate Keys.
If there is a combination of two or more attributes which is being used as the primary key then we call it as a Composite key.

Primary Keys are underlined.
Tables include a unique identifier in each row known as primary key.

Tables with same primary key can be joined to pull related information

primary key (row value)
attribute name (column value)

Weak entity's participation is always total.
Weak entity does not have a key of its own.
So its key will be combination of key of its identifying entity and its partial key.

⇒ Attribute domain

every attribute has some pre-defined value scope known as attribute domain.

"Flatten" composite attributes

Production Units
SerialNo ExactWeight ProductType ProductDesc Qtest LotNo (Foreign Key)

one to many

Lot
LotNo CreateDate CostofMaterials

RawMaterials Usage (many to many)
LotNo MaterialID Units

Raw Materials
MaterialID Type UnitCost

Q. Normalization is a process within logical design. What is the general goal of normalization? Why is it important to identify and remove partial and transitive functional dependencies during normalization.

Ans: The general goal is to remove redundancy in the data.

Schema Diagram can only be derived from ER/EER diagram.

PK uniquely identifies each row. Always underlined.

PK of one included in another table as FK.
(referencing a PK from another entity)

Step 1 For regular entities, create a relation and indicate it's primary key. Always underline PK.

Step 2 For weak entities, create a relation with it's name and include it's own attributes and the PK of the entity it's dependent on as a FK.
The PK of the weak entity would be a mixture of FK and it's own partial key.

PK is the combo of PK of owner and partial key of weak entity

partial key + (derived FK) → together PK

Step 3 ~~For weak entities, create a relation~~

For binary 1:1 relationships,

→ include the PK of the entity that optionally participates in the relationship as the FK of the total participating entity's relation table.

→ if both entities have total participation, then a relation with the relationship name and include all of the entities's attributes. Choose any one key as the PK.

An jodi total 1:1 total ↓↓ table

optional 1:1 total
PK → include PK as FK

include PK of "one" as FK in "many"

Q # (one-to-many → doesn't matter, total or partial participation)

Step4: * For 1:N relationships, include PK of the entity that's on ~~the~~ side with 1, as the FK of the entity on the N side

Step5: For M:N relationships, make another relation with the relationship's name and include it's own attributes (if any) and PK of both the related entities as the FK.
The combo of both FKs will make the PK.

Ans: Relationship Relation and two FKs

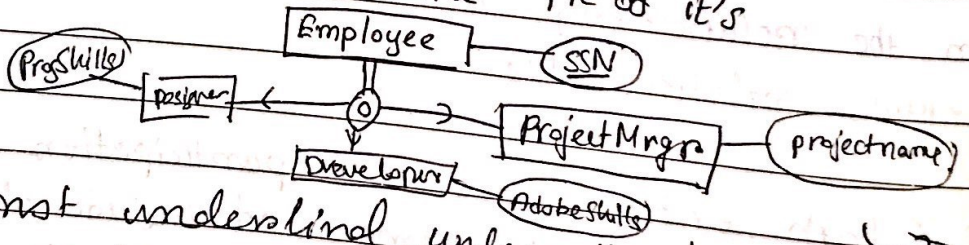
Step6: For recursive relations, include a new FK that is actually the PK of the entity itself but with a different name. [ssn] [Supervissn]

Step7: For multivalued attributes, make a separate relation with an attribute of the same name and a PK of the entity that it's the attribute of as the FK of this relation. PK is the FK itself.

* dept-locations: dno | DLoc

Relation and FK

Step8: For EER, each subclasses should have it's own relation which contains it's specific attributes and a FK that is the PK of it's superclass



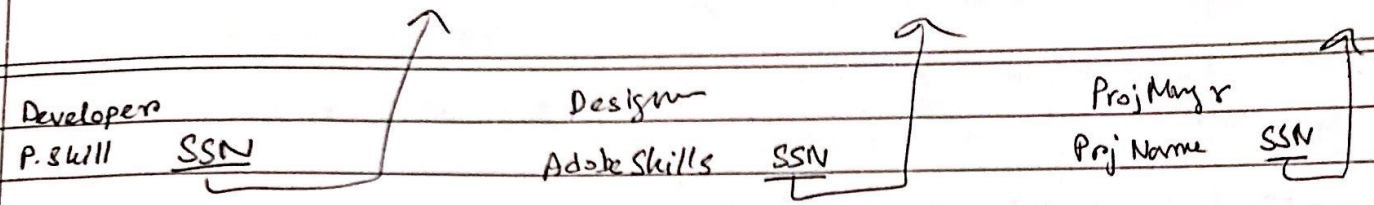
FKs are not underlined unless they're part of combo to form PK.

next page

PK → key attribute

Value set → domain

point to SSN of Employee's Table's SSN



Here, FKs are underlined.

FKs are itself PK in case of EER (subclass / superclass)

Subclasses dont have PK
↳ identified with PK of superclass.

Chapter 5

The Relational Data Model and Relational Database Constraints

• Relational Model Concepts

The relational model represents the database as a collection of relations.

The relational model of data is based on the concept of Relation.

Q. What is a Relation?

⇒ A Relation is a mathematical concept based on the ideas of sets.

The model was first proposed by Dr. E.F. Codd

- A relation looks like a table of values.
- A relation typically contains a set of rows.

* rows are called tuples *

The data elements in each row represent certain facts that correspond to a real-world entity or relationship.

- Each column has a column header that gives an indication of the meaning of the data items in that column. All values in a column are of the same data type.

* The column header is called an attribute or attribute name *

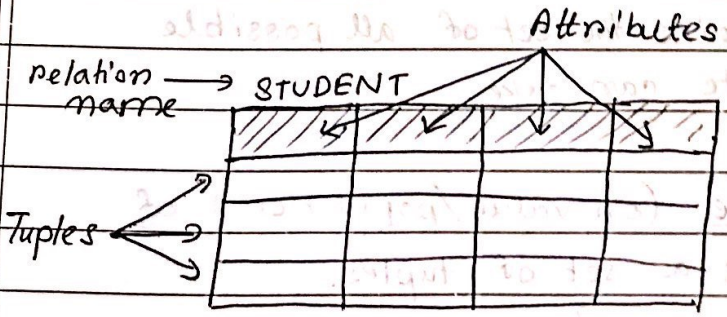


Fig: Example of a Relation

A relation schema is used to describe a relation.

Q.

Ans:

* The Schema (or description) of a Relation is denoted by:

$R(A1, A2, \ldots An)$

- R is the name of the relation (relation name)
- The attributes of the relation are A1, A2, ... An

* Each attribute has a domain (or a set of valid values)

- A tuple is ~~a set~~ an ordered set of values (enclosed in angled brackets '< ... >')
  Each value is derived from an appropriate domain.

- A relation is a set of such tuples (rows).

State: The relation state is a subset of the Cartesian product of the domains of its attributes.

- each domain contains the set of all possible values the attribute can take.

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

r(R): a specific state (or value/population) of relation of R → this is a set of tuples.

$r(R) = \{t1, t2, \ldots tn\}$

$t_1 = \langle v_1, v2 \ldots, vn \rangle$

Informal Terms	Formal Terms
Table	Relation
Column Header	Attribute
All possible column values	Domain
Row	Tuple
Table Definition	Schema of a Relation
Populated Table	State of the Relation

(• Hence, composite and multivalued attributes are not allowed.)

(it is not divisible into components within the framework of the basic relation model)

Q • A domain D is a set of atomic values
(each value in the domain is indivisible)

• A data type or format is specified for each domain.

Ans A domain is thus given a name, data type, and format.

Null values represent attributes whose values are unknown or do not exist for some individual tuple.

* tuples are not considered to be ordered
* attributes in $R(A1, A2 \ldots An)$ and values in $t = <v_1, v_2, \ldots v_n>$ are to be ordered.

Values in a tuple

- All values are considered atomic (indivisible)
- Each value in a tuple must be from the domain of the attribute for that column.

(each v_i must be a value from $dom(A_i)$.

* A special null value is used to represent values that are unknown or inapplicable to certain tuples *.

- component values of a tuple t is denoted by

  t.Ai or t[Ai]

  ⇒ This is the value v_i of attribute A_i for tuple t.

---

* Constraints are conditions that must hold on all valid relation states.

There are (3) main types of constraints in the relational model.

Schema-based constraints:
- Key constraints
- Entity integrity constraints
- Referential integrity constraints

(Key, En, Ref)

- Another implicit constraint is the domain constraint. Every value in a tuple must be from the domain of its attribute (or it could be null, if allowed for that attribute).

Key Constraints

⊙ Superkey of R is a set of attributes SK of R with the following condition:

- No two tuples in any valid relation state r(R) will have the same value for SK
- That is, for any distinct tuples t1 and t2 in r(R), $t1[SK] \neq t2[SK]$

$$t1[SK] \neq t2[SK]$$

A superkey SK specifies a uniqueness constraint that no two distinct tuples in any state r of R can have the same value for SK.

⊙ Key of R: A minimal superkey

That is, a key is a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey (does not possess the superkey uniqueness property)

(A minimal superkey is a superkey from which we cannot remove any attributes and still have the uniqueness constraint in condition 1 hold)

* • Any key is a SK (but NOT vice versa)
- Any set of attributes that includes a key is a superkey
- A minimal superkey is also a key.

A relation schema can have more than one key.
In this case, each of the keys is called a candidate key.
Designate one of the candidate keys as the primary key of the relation.
(underlined)
PK is the candidate key whose values are used to identify tuples in the relation.
The primary key value is used to uniquely identify each tuple in a relation.

Key constraint statement: Not two tuples in a relation must have the same value for the key attributes.

A relational database schema is a set S of relation schemas that belong to the same database

S is the name of the whole database schema.

$S = \{R_1, R_2, \ldots, R_n\}$

are the names of the individual relation schemas within the database S.

Entity integrity constraint

states that no primary key value can be null.

This is because PK value is used to identify individual tuples in a relation.

~~Having null~~

Referential Integrity → displayed as a directed arc

A constraint used to specify a relationship among tuples in two relations:

⇒ The referencing relation (childing) and the referenced relation (parent).

Tuples in referencing relation have attributes FK (foreign key attributes) that reference the PK (primary key attributes) of the referenced relation.

The value in the FK column of FK of the referencing relation can be either:

* a value of an existing PK value of a corresponding PK in the referenced relation.
* a null (if it is not part of its own PK)

ER diagram to Table

- Entity type becomes a table.
- All single-valued attribute becomes a column for the table.
- A key attribute (e.g. id) of the entity is represented by the primary key.
- The multivalued attribute (e.g. ~~address~~ hobby, email, phone) is represented by a separate table.

For e.g. hobby is a multivalued attribute. It is not possible to represent multiple values in a single column of STUDENT table.
So we create a table Student_Hobby with column name Stud_id and Hobby.
Using both the column, we create a composite key.

- Composite attribute represented by components.
  ↳ (e.g. address)
  City, State, PIN, Street → can merge as an individual column.
- Derived attributes are not considered in the table.
  ↳ (e.g. Age)

Relation model can be represented as a table with columns and rows.

- Rows are called tuples.
- Column header → attribute

Relation instances do not have duplicate tuples.
It can change whenever there's insertion, deletion or updation in the database.

Properties of Relations

1. Name of the relation is distinct from all other relations
2. Each relation cell contains exactly one atomic (single) value.
3. Each attribute contains a distinct name.
4. Attribute domain has no significance
5. tuple has no duplicate value
6. Order of tuple can have a different sequence

CUSTOMER (Cust_id, Cust_name, Address, Phone#)

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

- R: relation name
- (A1, A2, ..., An): schema of relation
- A1, A2, ..., An: attributes

Each attribute has a domain or a set of valid values.

Informal Terms	Formal Terms
Table	Relation
Column Header	Attribute
All possible Column Values	Domain
Row	Tuple
Table Definition	Schema of a Relation
Populated Table	State of the Relation.

Relational Integrity Constraints

- Constraints are conditions that must hold on all valid relation states.

There are three main types of constraints in the relational model:

- Key constraints
- Entity integrity constraints
- Referential integrity constraints.

(implicit constraint) ⇒ domain constraint

* Any key is a superkey (but not vice versa)

(unique) ↖ The primary (PK) key attributes are underlined.

(can't have NULL values) ↙ It is used to uniquely identify each tuple in a relation. It cannot have null values.

$S = \{R_1, R_2, \ldots, R_n\}$

→ DB Schema

→ Individual relation schemas within the database S

- SQL constraints are used to specify rules for data in a table.

Constraints limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table.

If there's any violation betn the constraint and the data action, the action is aborted.

Constraints can be column level or table level.

- NOT NULL - ensures that a column cannot have a NULL value
- UNIQUE - " " all values in a column are different
- PRIMARY KEY - A combination of NOT NULL and UNIQUE. Uniquely identifies each row in a table.
- FOREIGN KEY - Uniquely identifies a row/record in another table.
- CHECK - ensures that all values in a column satisfies a specific condition.
- DEFAULT - Sets a default value for a column when no value is specified.
- INDEX - Used to create and retrieve data from the database very quickly.

NOT NULL Constraint

⇒ By default, a column can hold NULL values.
The NOT NULL constraint enforces a column to NOT accept NULL values.
This enforces a field to always contain a value, which means we cannot insert a new record, or update a record without adding a value to this field.

An # A PK constraint automatically has a UNIQUE constraint.
You can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

The PK ~~uni~~ constraint uniquely identifies each record in a table.
It must contain UNIQUE values, and cannot contain NULL values.

A table can have only ONE primary key;
this pk can consist of single or multiple columns (fields)

A FOREIGN KEY is a key used to link two tables together.
A FK is a field (or collection of fields) in one table that refers to the PK in another table.

* The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

FK → PK
child → parent

The FK Constraint is used to prevent actions that would destroy links betⁿ tables.
The FK Constraint also prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to.

⊛ Main difference betⁿ where and having clause in SQL is that, condition specified in WHERE clause is used while fetching data (rows) from table, and data which doesn't pass the cond will not be fetched into result set;
HAVING clause is used to filter summarized data or group data

→ WHERE clause is used to filter rows and it applies to each and every row
→ HAVING clause is used to filter groups in SQL.

Group By clause ⇒ WHERE is used before ; } syntax level difference
HAVING used after
GROUP BY clause

SQL → Structured Query Language

Select

SELECT

SELECT * FROM Customers;
⇒ (selects all the records in the "Customers" table)

* SQL keywords are NOT case sensitive.

* semicolon at the end of each SQL statement.

• The SELECT DISTINCT statement is used to return only distinct (diff) values.

SELECT → selects All (including duplicates)

* WHERE clause is used to filter records.
It is used to extract only those records that fulfill a specified condition.

aggregate func (count, max, min, sum, avg)

Having clause used, since WHERE keyword cannot be used with aggregate funcs.

Relational Schema

Step 1 — Regular Entity Table
- simplest attributes
- PK underlined

Step 2 — Weak Entity Table
- PK of dependent entity inserted as PK
- Partial key underlined
- attributes

combo PK → combo ⇒ PK

Step 3 — 1 : 1 relationships

optional 1:1 total — (PK) ▽ FK

total 1 : total → new table

Step 4 — 1 : N relationships

include PK of "1" as FK in "many"

Step 5 — M : N relationships

Relation with two FKs

combo PK

* make table (enter PKs of both as FK)

Step 6 — multivalued attributes

Relation and FK

combo PK

* make table

multivalued attribute + ~~PK~~ FK

Step 7 — For recursive relation, include a new FK that is PK of entity itself but with a diff name.

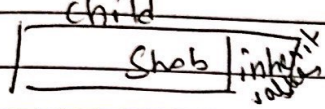
EER → Schema

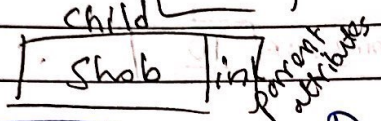
8A Works for any specialization (total or partial, disjoint or overlapping)

Works for all

8B Option only works for specialization whose subclasses are total

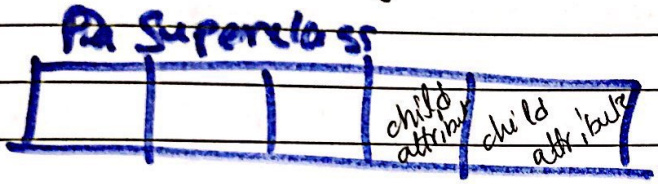
* First identify total





Doesn't work for partial

8C discriminating attribute



Single relation with ~~dis~~ one type attribute

Doesn't work for overlapping

8D Single relation with multiple type attributes

Flag

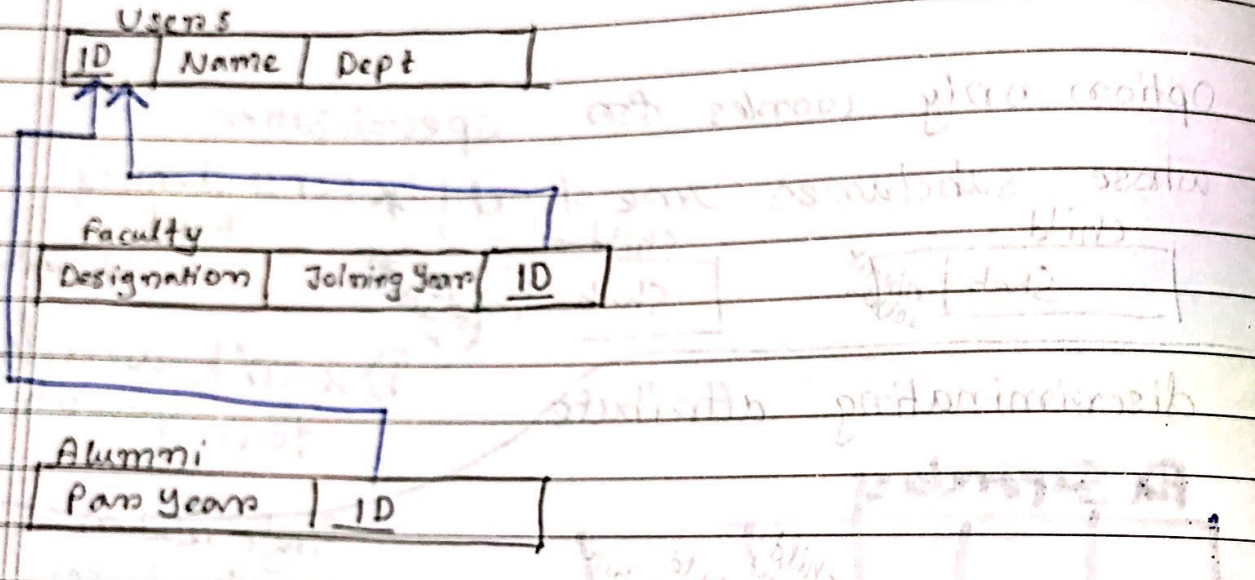
For overlapping will add flags for each subclasses

Works for all cases

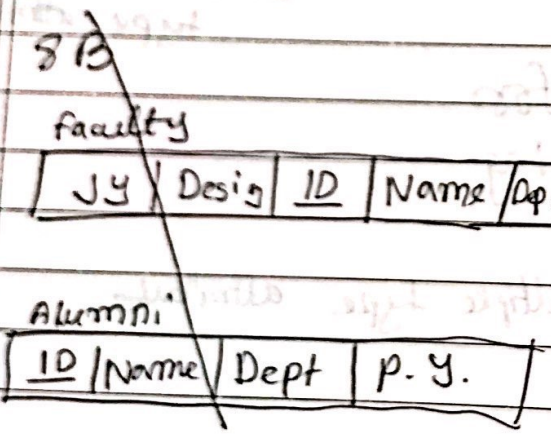
8A, 8D always hoy

Q8)

(i) 8A



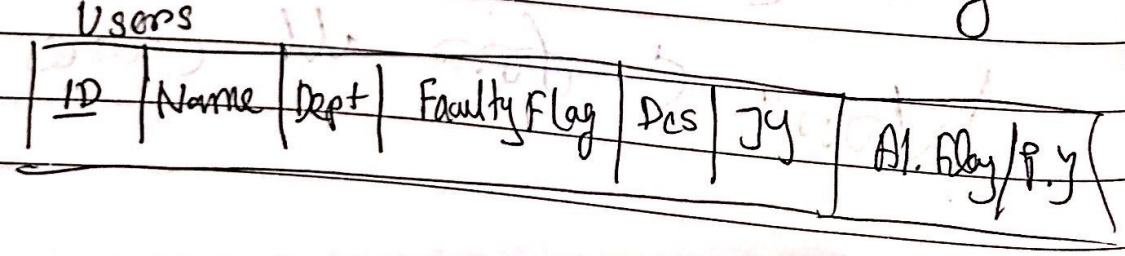
(ii) 8B



(ii) 8B Not possible, because partial

(iii) 8C Not possible, because overlapping

(iv) 8D



Chapter 10 Normalization

Two levels of schemas

- logical (user view) level / conceptual level
- storage (base relation) level)

↓ design concerned

1NF (First Normal form)

2NF (Second Normal Form)

3NF (Third Normal Form)

BCNF (Boyce-Codd Normal Form)

Redundant information causes:

- storage wastage
- problems with update anomalies

DIM

Deletion Insertion Modificatio

(data redundancy → same data but at diff. places)

Normalization → reduce data redundancy
→ improve data integrity
is the process of minimizing redundancy from a relation.

Redundancy in relation may cause insertion, deletion and updation anomalies.

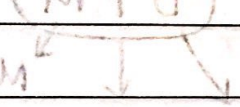
Normal forms
are used to eliminate or reduce redundancy in database tables.

1NF

* 1NF does (NOT) contain any composite or multivalued attribute ; if present, 1NF violated.

* Every attribute is single-valued attribute (atomic) - atomic value

- no multivalued attribute
- no composite attribute
- atomic value



2NF → No partial dependency

(i.e. no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table)

To be in 2NF, a relation must be 1NF and relation must not contain any partial dependency.

2NF tries to reduce the redundant data getting stored in memory.

[All ~~non-key~~ non-prime attributes are fully functional dependent on the primary key PK]

A func dependency on part of any candidate key is a violation of 2NF.

non-prime attribute — not part of any candidate key.

§ ২টা থাকবে না
just ekta থাকে

3NF → used to reduce data duplication, and achieve data integrity.

- must be 2NF
- No transitive dependency for non-prime attributes

(i.e. no non-prime attributes depends on other non-prime attributes)

All the non-prime attributes must depend only ~~th~~ on the keys.

~~depen~~ (key chara jodi SQL dependent hoy)

FD denoted by arrow

◇ Functional dependency is a constraint betn two sets of attributes

$A \rightarrow B$

A functionally determines B.

A	B
1	3
2	3
4	0
1	3
4	0

For all instances of a particular value of A, there is the same value of B.

FDs are constraints that are derived from the meaning and interrelationships of the data attributes.