DBMS

Module - 01

List and explain the characteristics of database approach and how it differs from traditional file system. (July/August 2021, Jan/Feb 2021, Aug/Sep 2021, July/August 2021)

⇒ The main characteristics of the database approach versus the file-processing approch are the following.

* Self-describing nature of a database system
* Insulation between programs and data, and data abstraction
* Support of multiple views of the data.
* Sharing of data and multiuser transaction processing.

Self-describing nature of a database system

DB approach:

* The database system contains not only the database itself but also a "complete definition or description of the database and constraints".
* This definition is stored in the database catalog, which contains information such as the structure of each file, the type and storage format of each data item and various constraints on the data.
* The information stored in the catalog is called meta-data, and it describes the structure of the primary database.

Traditional file processing:

* data definition is typically part of the application programs themselves.
* Hence, these programs are constrained to work with only specific database, whose structure is declared in the

application progeams.

* In typical file-processing application, the file structure and, in the extreme case, the exact location of name within a STUDENT record are already coded within each program that access this data item.

Insulation between n programs and data, and data abstraction

DB approach:

* By contrast, DBA access programs do not require such changes in most cases.
* The structure of data files is stored in the DBMS catalog seperately from the access programs. We call this property Program-data Independence.

Traditional file processing:

* The structure of data files is embedded in the application programs, so any changes to the structure of file may require changing all programs that access that file.

Support of multiple Views of the Data.

* A database typically has many types of users, each of whom may require a different perspective or view of the database.
* A view may be a subset of the database or it may contain virtual data is derived from the database files but is not explicitly stored.

* A multiuser DBMS whose users have a variety of distinct applications must provide facilities for defining multiple views.

sharing of data and multiuser transaction processing :-

* A multiuser DBMS, as its name implies, must allow multiple users to access the database at the same time.
* This essential if data for multiple application is to be integrated and maintained in a single database.
* A fundamental role of multiuser DBMS software is to ensure that concurrent transactions operate correctly and efficiently.
* A transaction is an executing program or process that includes one or more database access, such as reading or updating of database records.
* The DBMS must enforce several transaction properties.
* The isolation property ensures that each transaction appears to execute in isolation from other transactions even though hundred's of transactions may be executing concurrently.
The atomicity property ensures that either all the database operations in a transaction are executed or none are.

2) What are the advantages of DBMS? Discuss the five advantages by comparing with file.
(July/August 2021, Jan/Feb 2021-18 CBS 3, July/August 2021, Dec 2019/Jan 2020, Aug/Sept 2020-17 CBS 3, Jan/Feb 2021, Jan/Feb 2021-15 CBS 3)

→ The advantages of DBMS are:
1) Controlling Redundancy
2) Restricting Unauthorized Access
3) Providing Persistent Storage for Program Objects
4) Providing Storage Structure and Search Techniques for efficient Query Processing.
5) Providing Backup and Recovery.
6) Providing Multiple User Interfaces
7) Representing Complex Relationships among Data
8) Enforcing Integrity Constraints.
9) Permitting Inferencing and Actions using Rules and Triggers.

1) Controlling Redundancy:
→ Data redundancy such as tends to occur in the 'file processing' approach) leads to wasted storage space, duplication of effort, and a higher likelihood of the introduction of inconsistency.
→ On the other hand, redundancy can be used to improve performance of queries. Indexes for example, are entirely redundant but help the DBMS in processing queries more quickly.
→ A DBMS should provide the capability to automatic-

* A multiuser DBMS whose users have a variety of distinct applications must provide facilities for defining multiple views.

sharing of data and multiuser transaction processing :-

* A multiuser DBMS, as its name implies, must allow multiple users to access the database at the same time.
* This essential if data for multiple application is to be integrated and maintained in a single database.
* A fundamental role of multiuser DBMS software is to ensure that concurrent transactions operate correctly and efficiently.
* A transaction is an executing program or process that includes one or more database access, such as reading or updating of database records.
* The DBMS must enforce several transaction properties.
* The isolation property ensures that each transaction appears to execute in isolation from other transactions even though hundred's of transactions may be executing concurrently. The atomicity property ensures that either all the database operations in a transaction are executed or none are.

-ally enforce the rule that no inconsistencies are introduced when data is updated.

2) Restricting unauthorized access:

→ When multiple users share a large database, it is likely that most users will not be authorized to access all information in the database. For example, financial data such as salaries and bonuses is often considered confidential, and only authorized persons are allowed to access such data.

→ A DBMS should provide a security and author-ization subsystem, which the DBA uses to create accounts and to specify account restrictions. Then, the DBMS should enforce these restrictions automatically.

3) Providing Persistent Storage for Program Objects:

→ Object-oriented database systems make it easier for complex runtime objects (eg, lists, trees) to be saved in secondary storage so as to survive beyond program termination and to be retrievable at a later time.

→ Object-oriented database systems typically offer data structure compatibility with one or more object-oriented programming languages.

4) Providing Storage Structures and Search Techniques for Efficient Query Processing:

→ Database systems must provide capabilities for

efficiently executing queries and updates. Because the database is typically stored on disk, the DBMS must provide specialized data structures and search techniques to speed up disk search for the desired records. Auxiliary files called indexes are often used for this purpose.

→ The query processing and optimization module of the DBMS is responsible for choosing an efficient query execution plan for each query based on the existing storage structures.

5) Providing Backup and Recovery:

→ A DBMS must provide facilities for recovering from hardware or software failures. The backup and recovery subsystem of the DBMS is responsible for recovery.

→ For example, if the computer system fails in the middle of complex update transaction, the recovery subsystem is responsible for making sure that the database is restored to the state it was in before the transaction started executing.

3) Define a data model. Discuss the main categories of data model with examples. (July/August 2021 - 18CS53)

⇒ A data model - a collection of concepts that can be used describe the structure of a database - Provides the necessary means to achieve this abstraction. Structure of database means the data types, relationship, and constrai -nts that apply to the data

Categories of Data models :-

1. High-level or Conceptual data models
2. Low - level or Physical data models
3. Representational (implementation) data models

1. Conceptual data models :-

- It Provide concepts that are close to the way many users Perceive data
- use the concepts such as "entities, attributes, and relationship"
- It creats various business rules

2. Physical data models :-

* Provide concepts that describe the details of how data is stored on the computer storage media, typically magnetic disks
* Concepts provided by low-level data models are generally meant for computer specialists not for end users

3. Representation data models :-

* Provide concepts that ~~describe the details of how~~ may be easily understood by end users but that are not too far removed from the way data is organized in computer storage
* In between high level and low level
* hide many details of data storage on disk but can be implemented on a computer system directly

4. Explain the different types of end users. Discuss the main activities of each with example.
(July | August 2021, Jan | Feb 2021 - 18CS53, Jan | Feb 2021 - 17CS53) marks: [6m, 5m, 4m]

Ans:- End users are the people whose jobs require access to the database for querying, updating, and generating report.

* 1) Casual end user: occasionally access the database, but they may need different information each time. They use a sophisticated database query language to specify their requests and are typically middle- or high-level managers or other occasional browsers.

Example:- High level managers who access the data weekly basis.

* 2) Naive or parametric end users: make up a sizable portion of database end users. The main job function revolves around constantly querying and updating the database, using standard types of queries and updates - called canned transactions - that have been carefully programmed and tested

Example: * Bank tellers check account balances and post withdrawals and deposits.

* Shipping clerks (e.g., at UPS) who use buttons, bar code scanners, etc., to update status of in-transit packages.

* 3) Sophisticated end users: include engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS in order to implement their complex requirements.

Example: users such as a business analyst, Scientist etc. interact with the Database without writing any application Programs.

4) Stand-alone users: use "personal" databases, possibly employing a special purpose (e.g., financial) software package. Mostly maintain personal using ready-to-use packaged applications.

Example: An example is a tax program user that creates its own internal database

5. What are the responsibilities of DBA and Database Designers (July/August 2021, Dec 2019 | Jan 2020 - 17CS53)
marks : [4m]

→ Database Administrators:

* In any organisation where many people use the same resources, there is a need for a chief administrator to oversee and manage these resources.
* In a database environment, the primary resources in the database itself, and the secondary resource is the DBMS and related software
* Administrating these resources is the responsibility of the database administrator (DBA).
* The DBA is responsible for authorizing access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed
* The DBA is accountable for problems such as security breaches and poor system response time.

In large organizations, the DBA is assisted by a staff that carries

Database Designers:

* Database Designers are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data.

* In a database environment, these tasks are mostly undertaken before the database is actually implemented and populated with data.

* It is the responsibility of database designers to communicate with all prospective database users in order to understand their requirements and to create a design that meets these requirements.

* Database designers typically interact interact with each potential groups of users and develop views of the database that meet the data and processing requirements of these groups. Each view user groups. The final database design must be capable of supporting the requirements of all user groups

6) With a neat block diagram, explain the architecture of a typical DBMS. (Dec 2019 / Jan 2020 - 17CS53)

→ The below figure shows the architecture of DBMS.

The DBMS accepts SQL commands generated from a variety of interfaces, produces query evaluation plans, executes these plans against the database, and returns the answers.

When a user issues a query, the parsed query is presented to a query optimizer, which uses information about how the data is stored to produces an efficient execution plans evaluating the query.

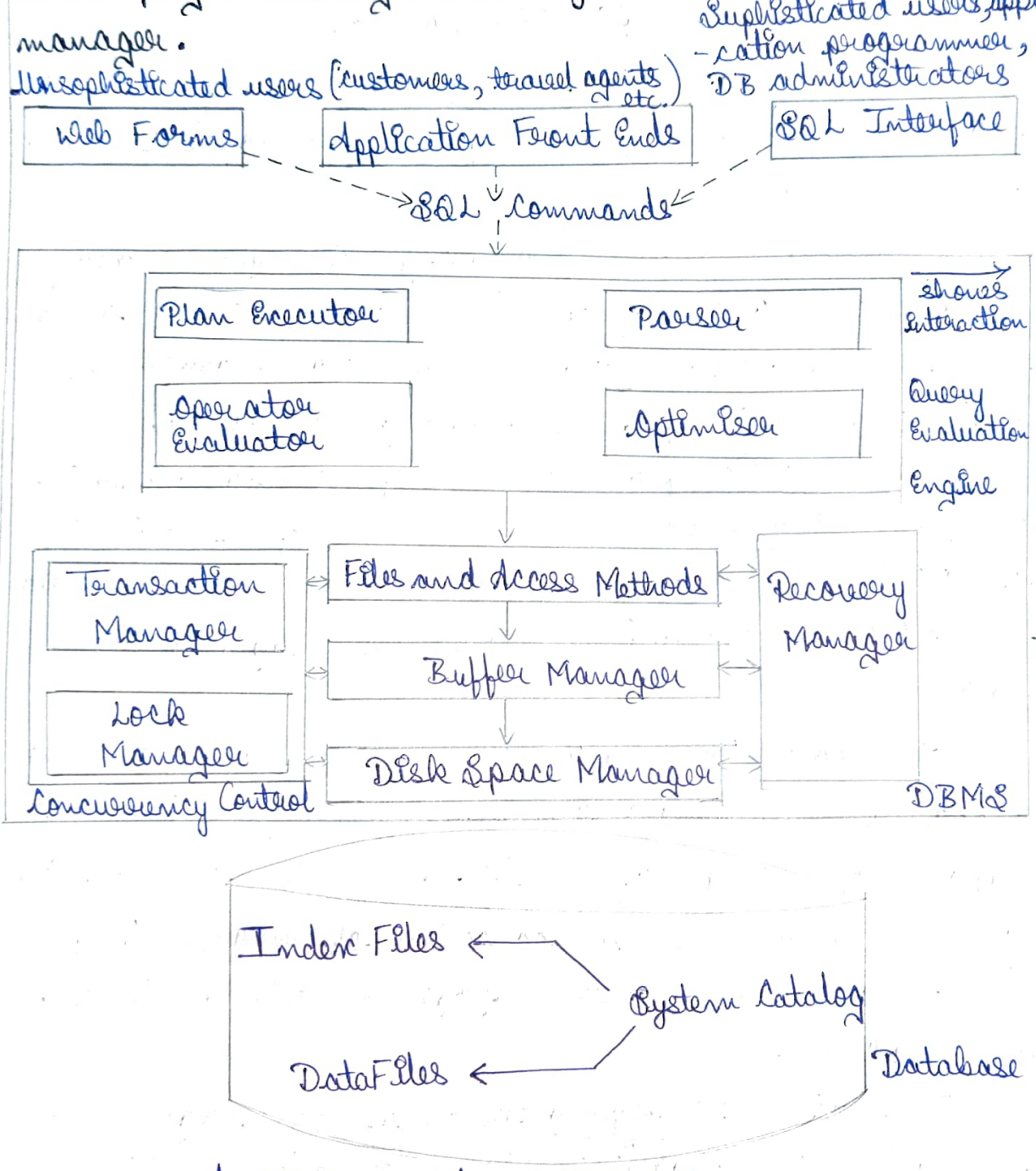
An execution plan is a blueprint for evaluating query, usually represented as a tree of relational operators. These relational operators serves as the building blocks for evaluating queries posed against the data.

The code that implements relational operators sits on top of the file and access methods layer. This layer supports the concept of a file, which in a DBMS, is a collection of pages or a collection of records. Heap files, or files of unordered pages as well as indexes are supported.

In addition to keeping track of the pages in a file, this layer organizes the information within a page. The files and access methods layer code

sits on top of the buffer manager, which brings pages in from disk to main memory as needed in response to read requests

The lowest layer of the DBMS software deals with management of space on disk, where the data is stored. Higher layers allocate, deallocate, read and write pages through this layer called the disks space manager.



Architecture of a DBMS

The DBMS supports concurrency and crash recovery by carefully scheduling user requests and maintaining a log of all changes to the database DBMS components associate with concurrency control.

And recovery include the transaction manager which ensures that transaction request and release locks according to suitable locking protocol and schedules the execution transactions.

7) With an aid of a neat diagram, describe a Three-Schema Architecture.

(Or)

Describe the three-schema architecture. Why do we need mapping between schema levels.

(July/August 2021, Jan/Feb 2021-18CS53, Jan/Feb 2021, July/August 2021-17CS53, July/August 2021, Feb/Feb 2021-15CS53)

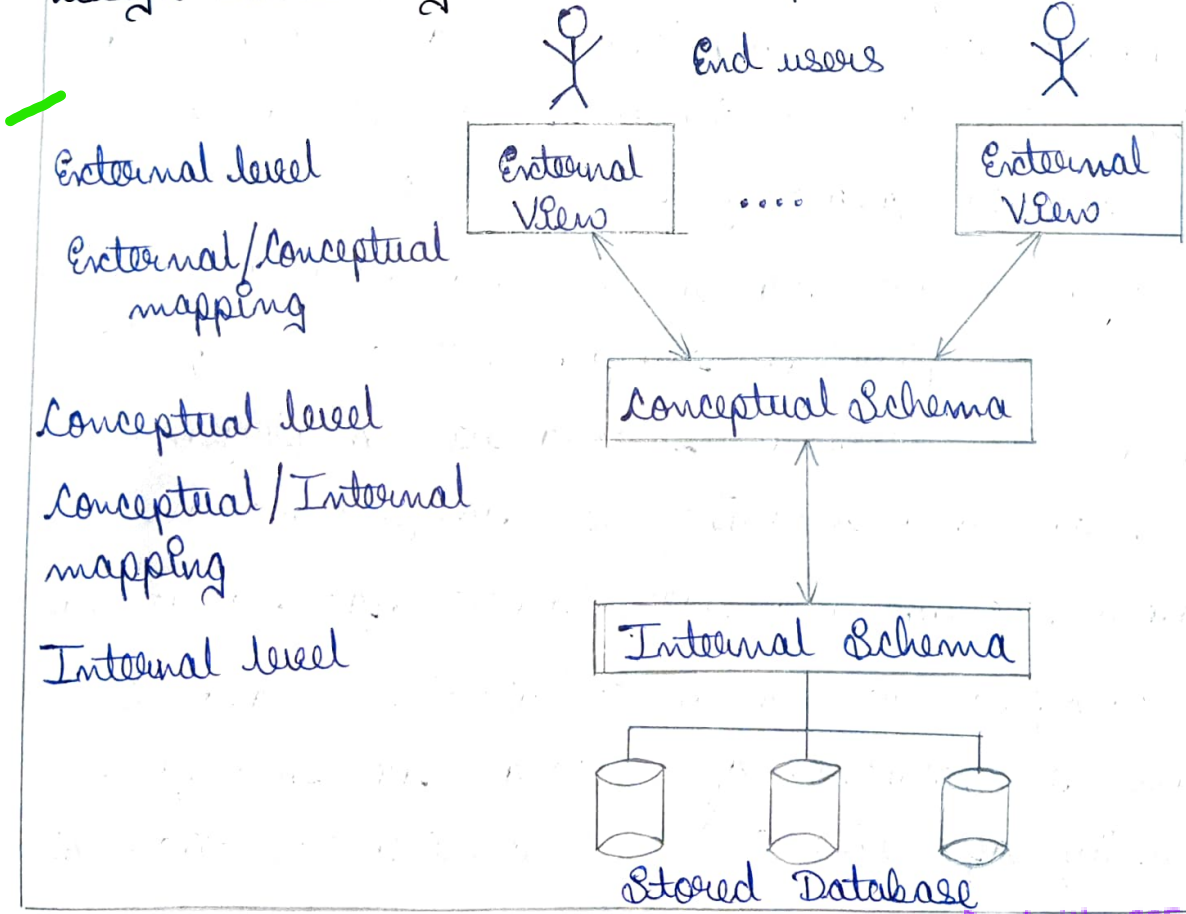
The goal of the three-schema architecture is to separate the user applications from the physical database. In this architecture, schemas can be defined at the following three levels:

1. The internal level has an internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

2. The conceptual level has a conceptual schema, which describes the structure of the whole database for a community of users. This schema hides the details of

physical storage structures and concentrates on describing entities, data types, relationships, user operations and constraints. Usually, a representational data model is used to describe the conceptual schema when a database system is implemented. This implementation conceptual schema is often based on a conceptual schema design in a high-level data model.

3. The external or view level includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. This schema is also implemented using a representational data model, possibly based on an external schema design in a high-level conceptual data model.



The DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database. So for the process of transforming requests and results between levels we need mapping.

s) With a neat diagram, explain the component modules of DBMS and their interactions. (Aug/Sept 2020 - 17CS53, July/August 2021, Jan/Feb 2021 - 15CS53)

→ The figure illustrates a simplified form of the typical DBMS components. The figure is divided into two parts.

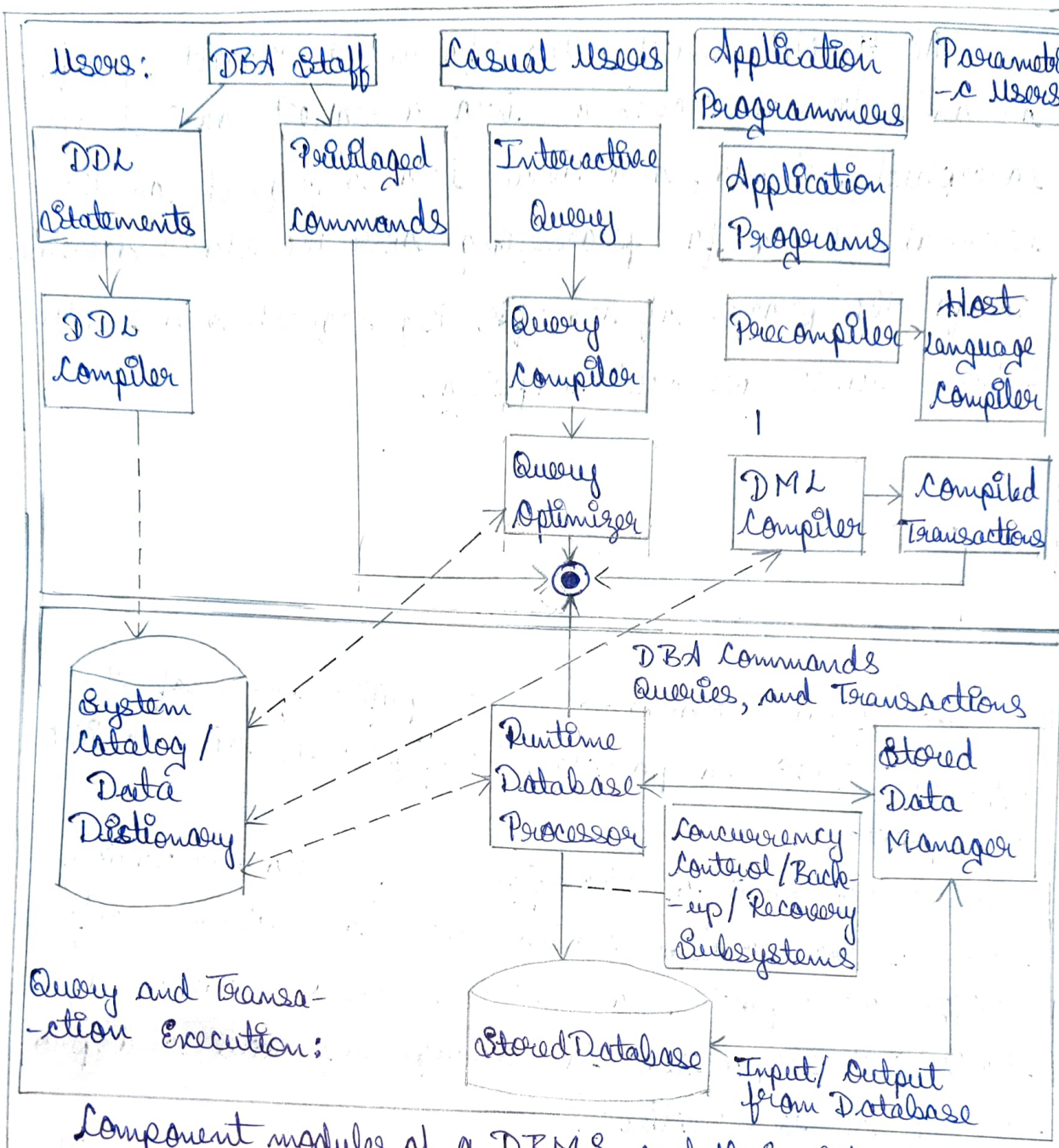
1) The top part of the figure refers to the various users of the database environment and their interfaces.

2) The lower part shows the internal modules of the DBMS responsible for storage of data and processing of transactions.

The database and the DBMS catalog are usually stored on the disk. Access to the disk is controlled primarily by the operating system (OS), which schedules disk the read/write.

Many DBMSs have their own buffer management module to schedule disk read/write.

Stored data manager controls access to DBMS information that is stored on disk, whether it is part of database or the catalog.



Component modules of a DBMS, and their interactions

Top half figure:

→ It shows interfaces for the DBA staff, casual users who work with interactive interfaces to formulate queries.

→ The DBA staff works on defining the database and tuning it by making changes to its definition using DDL and other privileged commands.

* The DDL compiler process schema definitions, specified

In the DDL, and stores descriptions of the schemas (meta-data) in the DBMS catalog.

* The catalog includes information such as the names, size of files, names and data types of data items, storage details of each file, mapping information and constraints.

→ The casual users interact using some form of interface called as interactive query interface.

* Queries are parsed and validated for correctness of the query syntax, the names of files and data elements, and so on by a query compiler that compiles into an internal form.

* This internal query is subjected to query optimization which is concerned with the rearrangement and possible reordering of operations, elimination of redundancies and use of correct algorithms and indexes during execution.

→ Application programmers write program in host languages such as Java / C / C++ that are submitted to a precompiler which extracts DML commands from an application program, then commands are sent to the DML compiler for compilation and rest of the program is sent to the host language compiler.

* The object codes for the DML commands and the rest of the program are linked, forming a canned transaction.

In the lower part of the figure,

→ The runtime database processor executes

1) the privileged commands
2) the executable query plans and
3) the canned transactions with runtime parameters

→ It works with the system catalog and may update it with statistics.

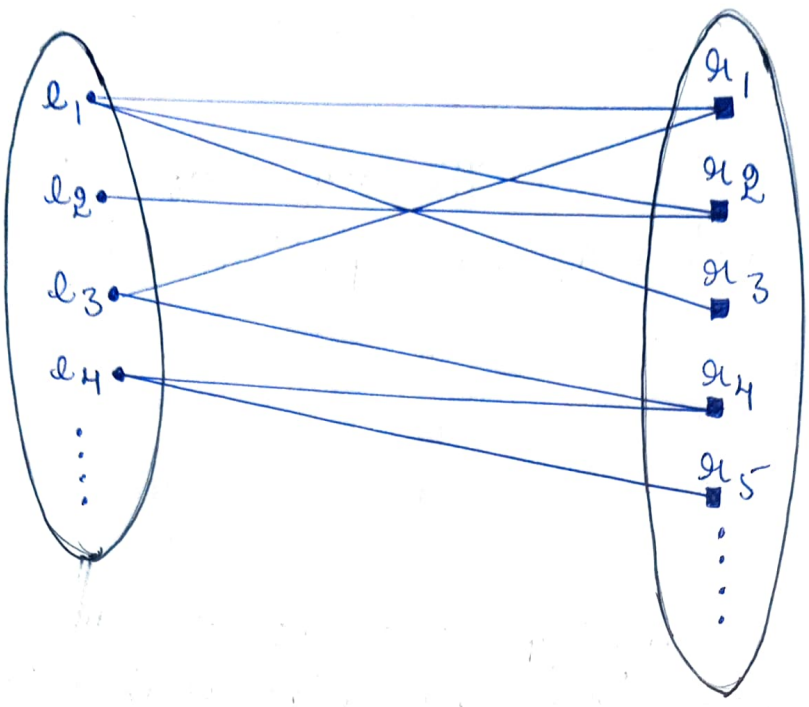
→ It also works with the stored data manager, which in turn uses basic operating system services for carrying out low-level input/output operations between the disk and main memory.

9) What is meant by Recursive relationship type? Give some examples of recursive relationship type.
(Jan/Feb 2021 - 17CS5-3)

→ Same entity type participates more than once in a relationship type in different roles. Such relationship types are called recursive relationship or self-referencing relationships.

Examples:

1) *The SUPERVISION relationship type relates an employee to a supervisor where both employee and supervisor entities are members of the same EMPLOYEE entity set. Hence, the EMPLOYEE entity type participate twice in SUPERVISION: once in the role of supervisor and once in the role of supervisee. Each relationship instance r_i in SUPERVISION associates two different employee entities e_j and e_k, one of which plays the role of supervisor and the other the role of supervisee.



2) Student can be a class monitor and handle other students but a person who is working as a class leader is itself a student of the class and hence a class monitor has a recursive relationship of entity student.

10) Define the following terms:
Jan/Feb 2021-18CS53, Dec 2019/Jan 2020, Aug/Sept 2020 -17CS53

(a) Database: A database is a collection of related data.

(b) DBMS catalog: A database catalog of a database instance contains the information such as the structure of each file, the type and storage format of each data item, and various constraints on the data.

(c) Entity: An entity is a thing or object in the real world with an independent existence.

(d) Snapshot: The data in the database at a particular moment is called a database state or snapshot.

(e) Degree of relationship: The degree of relationship type is the number of participating entity types.

(f) Cardinality: Cardinality represents the number of times an entity of an entity set participates in a relationship set or we can say that cardinality of a relationship is the number of tuples (rows) in a relationship.

(g) Weak entity: Entity types that do not have key attributes of their own are called weak entity types.

(h) Program Data Independence: The ability to modify the schema without affecting the programs and the application to rewritten. The structure of data files is stored in the DBMS catalog separately from the access programs. This property is called program data independence.

(i) Total participation: The participation of an entity set E in a relationship set R is said to be total if every entity in E participates in at least one relationship in R. That means each entity in the entity set must participate in the relationship.

(j) Value sets: Each simple attribute of an entity type is associated with a value set, which specifies the set values that may be assigned to that attribute for

each individual entity.

(k) Data model: A data model is a collection of concepts that can be used to describe the structure of a database which provides the necessary means to achieve this abstraction.

(l) Metadata: The DBMS stores the descriptions of the schema constructs and constraints also called the metadata in the DBMS catalog so that DBMS software can refer to the schema whenever it needs to be.

(m) Schema: The description of a database is called the database schema.

(n) Instance: The data in the database at a particular moment in time is called a database state or snapshot or Instances in the database.

(o) Canned Transaction: Canned transaction is the process of constantly querying and updating the database, using standard types of queries and updates.

11) List and explain the different types of attributes along with their symbols in ER model with examples.

→ The different types of attributes are:

1) Simple
2) Composite
3) Single-valued

4) Multivalued

5) Stored

6) Derived

1) Simple attribute:

→ Attributes that are not divisible are called simple or atomic attributes.

Eg: The roll number of a student, the Id number of an employee

→ A simple attribute is represented by an oval.

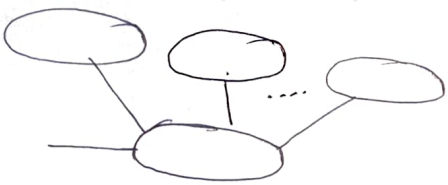
Eg: (roll_no)

2) Composite attribute:

→ These attribute can be divided into smaller subparts, which represent more basic attributes with independent meanings.

For example: person's address which is composed of atomic attributes such as ~~city~~, street_address, City, State.

→ These attributes are represented as key,



3) Single valued attribute:

→ Most attributes have a single value for a particular entity, such attributes are called as single valued attribute.

→ For example, Age is a single-valued attribute.

→ This attribute is represented by simple oval.

Eg:

4) Multivalued attribute:

→ This attribute of an entity is an attribute that can have more than one value associated with the the key of the entity.

→ For example, A person can have more than one phone number hence "phone number" will become a multi-valued attribute.

~~This~~ attribute is represented using double oval with the name of the attribute inside the oval.

5) Stored Attribute:

→ Stored attributes are those attributes that are stored in the physical database.

→ Eg: ~~date of birth~~. Student_Id, name, course_Id.

→ This attribute is represented by an oval.

6) Derived Attribute:

→ If an attribute's value can be derived from the values of other attributes, then the attribute is derivable and is said to be derived attribute.

→ For example, if we have an attribute for birth date then age is derivable.

→ These attributes are represented with a dotted line.

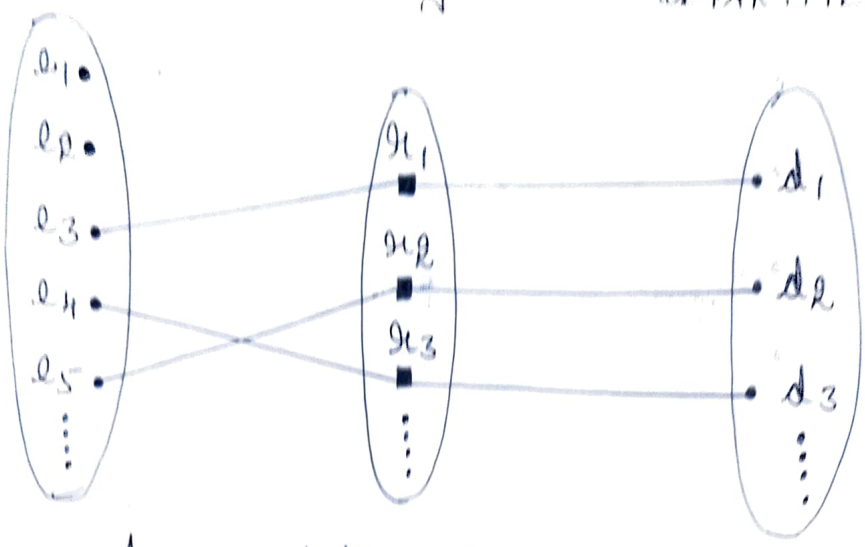
12) What are the structural constraints on a relation type? Explain with examples.
(July / August 2021, Aug / Sept 2020 - 17CS53)

→ Cardinality ratios and participation constraints taken together are called structural constraints.

→ The cardinality ratio for a binary relationship specifies the maximum number of relationship instances that an entity can participate in.
The possible cardinality ratios for binary relationship types are:

1) One to One (1:1) Relationship:

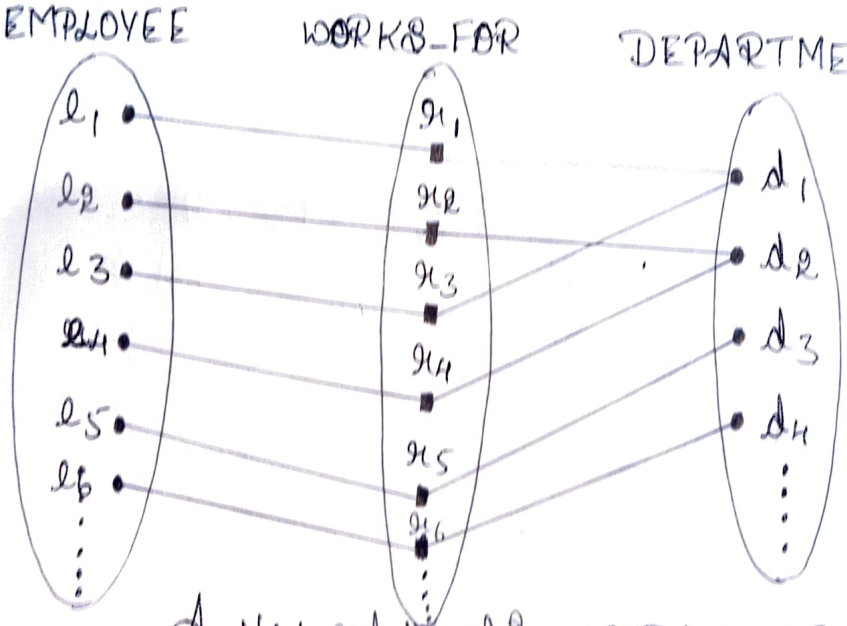
An entity set A is associated with at most one entity in B is associated with atmost one entity in A.
Eg: 1:1 binary relationship is MANAGES, which relates a department entity to the employee who manages that department. This represents the miniworld constraints that at any point in time, an employee can manage one department only and a department can have manager only.



A 1:1 relationship, MANAGES

2) Many to one (N:1) Relationship:

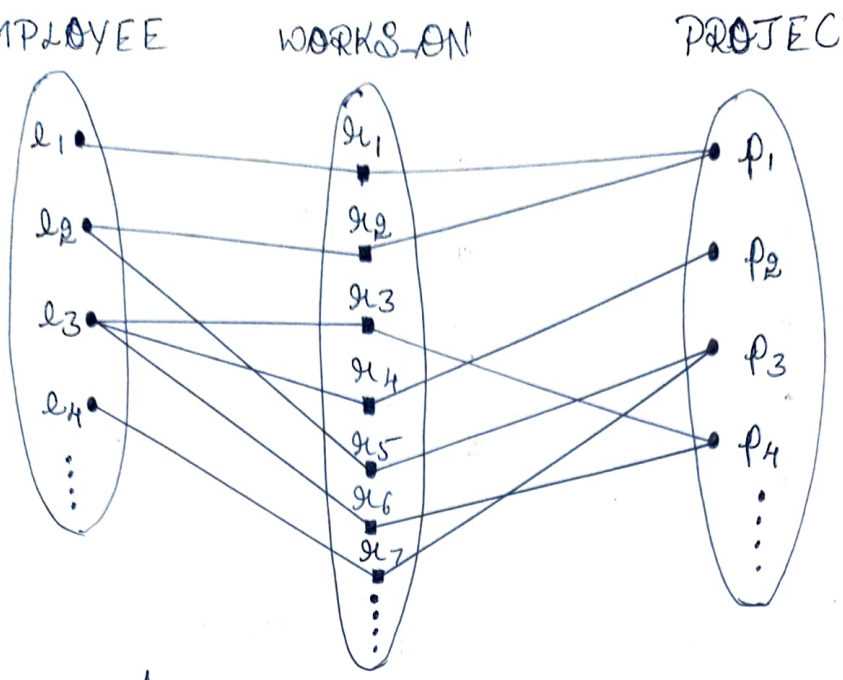
An entity set A is associated with atmost one entity in B and an entity in B is associated with any number of entities in A with a possibility of zero.



A N:1 relationship, WORKS_FOR

3) Many to Many (M:N) Relationship:

An entity set A is associated with any number of entities in B with a possibility of zero.



A M:N relationship, WORKS_ON

4) One to Many (1:N) Relationship:

An entity set A is associated with any number of entities in B with a possibility of zero and an entity B is associated with atmost one entity in A.

→ The participation constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type.

→ This constraint are of two types - Total participation and partial participation.

→ For example, in a company policy states that every employee must work for a department, then an employee entity can exist only if it participates in atleast one WORKS_FOR relationship instance. Thus, the participation of EMPLOYEE in WORKS_FOR is called total participation, meaning that every entity in the total set of employee entities must be related to a

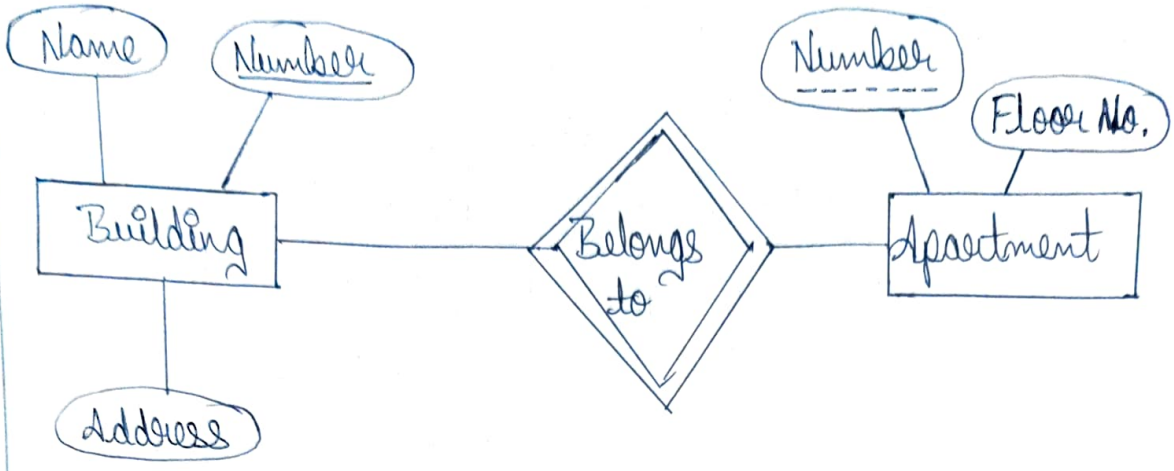
department entity via WORKS_FOR. We do not expect every employee to manage a department, so the participation of EMPLOYEE in the MANAGES relationship type is partial, meaning that some or part of the set of employee entities are related to some department entity via MANAGES, but not necessarily all.

13) What is a weak entity type? Explain the role of partial key in design of weak entity type. (July/August 2021 - 17CS53)

→ Entity types that do not have key attributes of their own are called weak entity types.

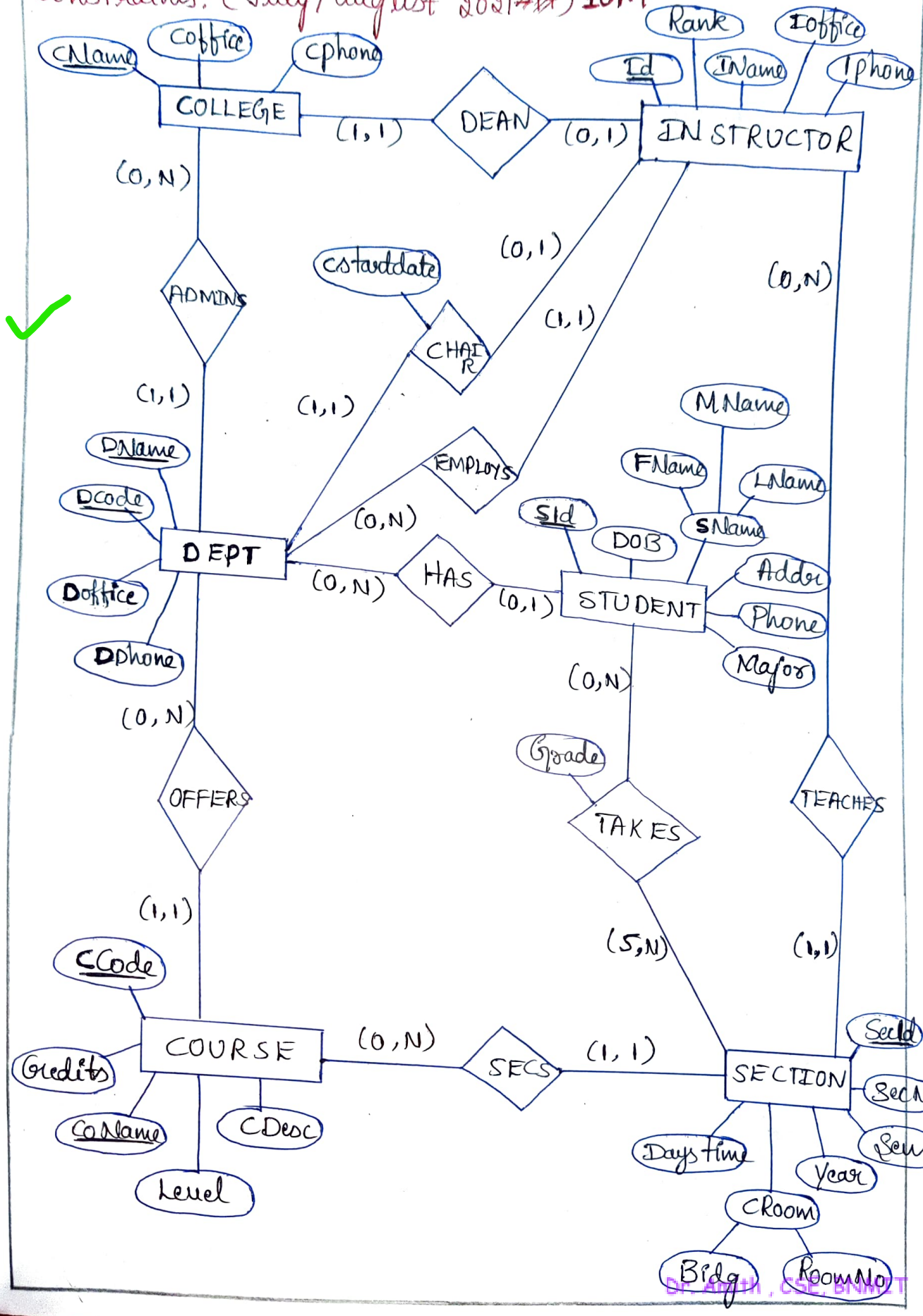
* The set of attributes that are used to uniquely identify a weak entity set is called the partial key.
* Only a bunch of the tuples can be identified using the partial keys.
* The partial key of the weak entity set is also known as a discriminator.
* It is just a part of the key as only a subset of the attributes can be identified using it.
* This key is partially unique and can be combined with other strong entity set to uniquely identify the tuples.

The below figure illustrates the partial key apartment number is shown with a dashed line.

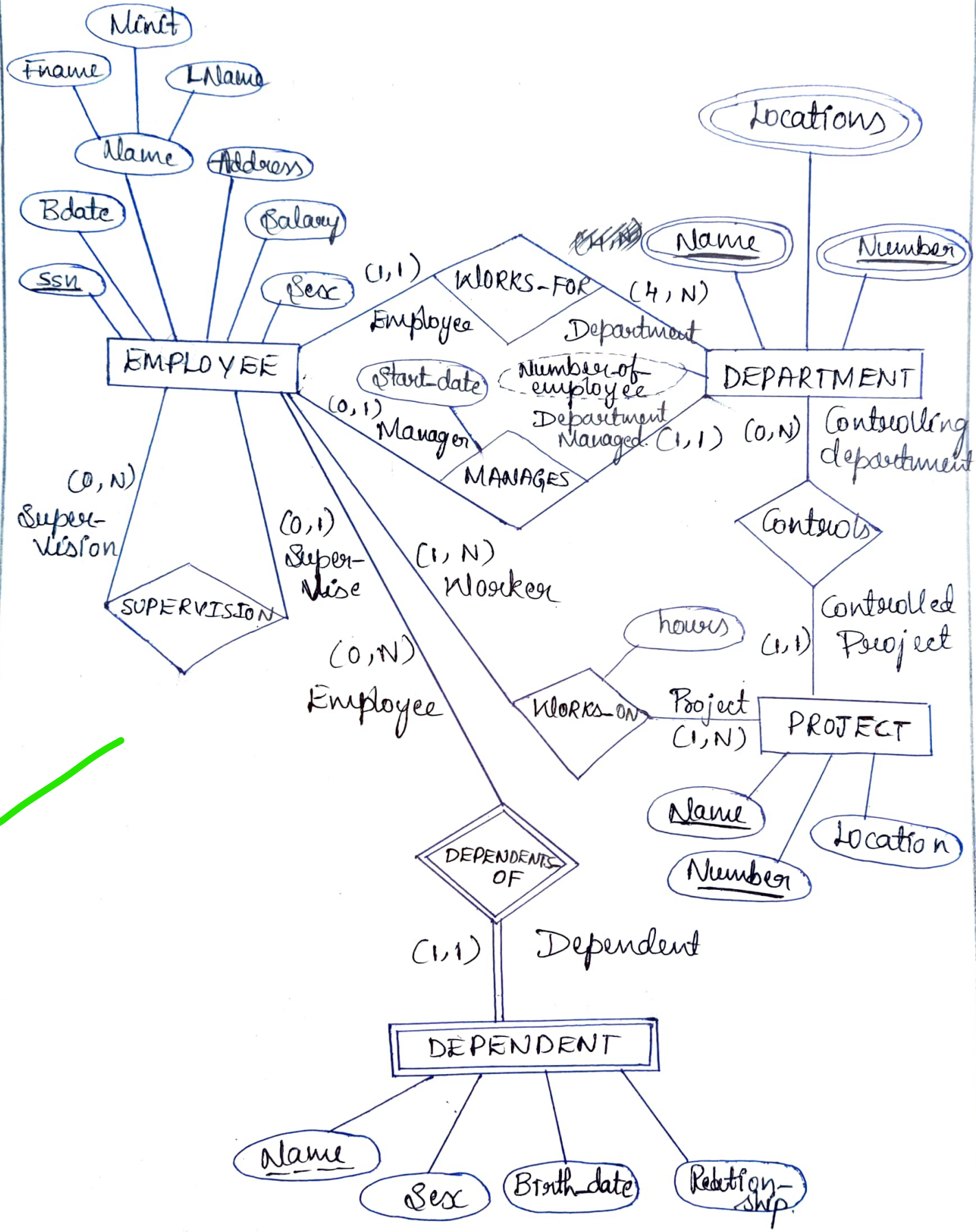


* Here we have an apartment as a weak entity and building as a strong entity type connected via 'belongs to' relationship set. Apartment number is not globally unique that is more than one apartment may have same number globally but it is unique for a particular building since a building may not have same apartment number which cannot be primary key of entity Apartment but it is a partial key shown with dashed line.

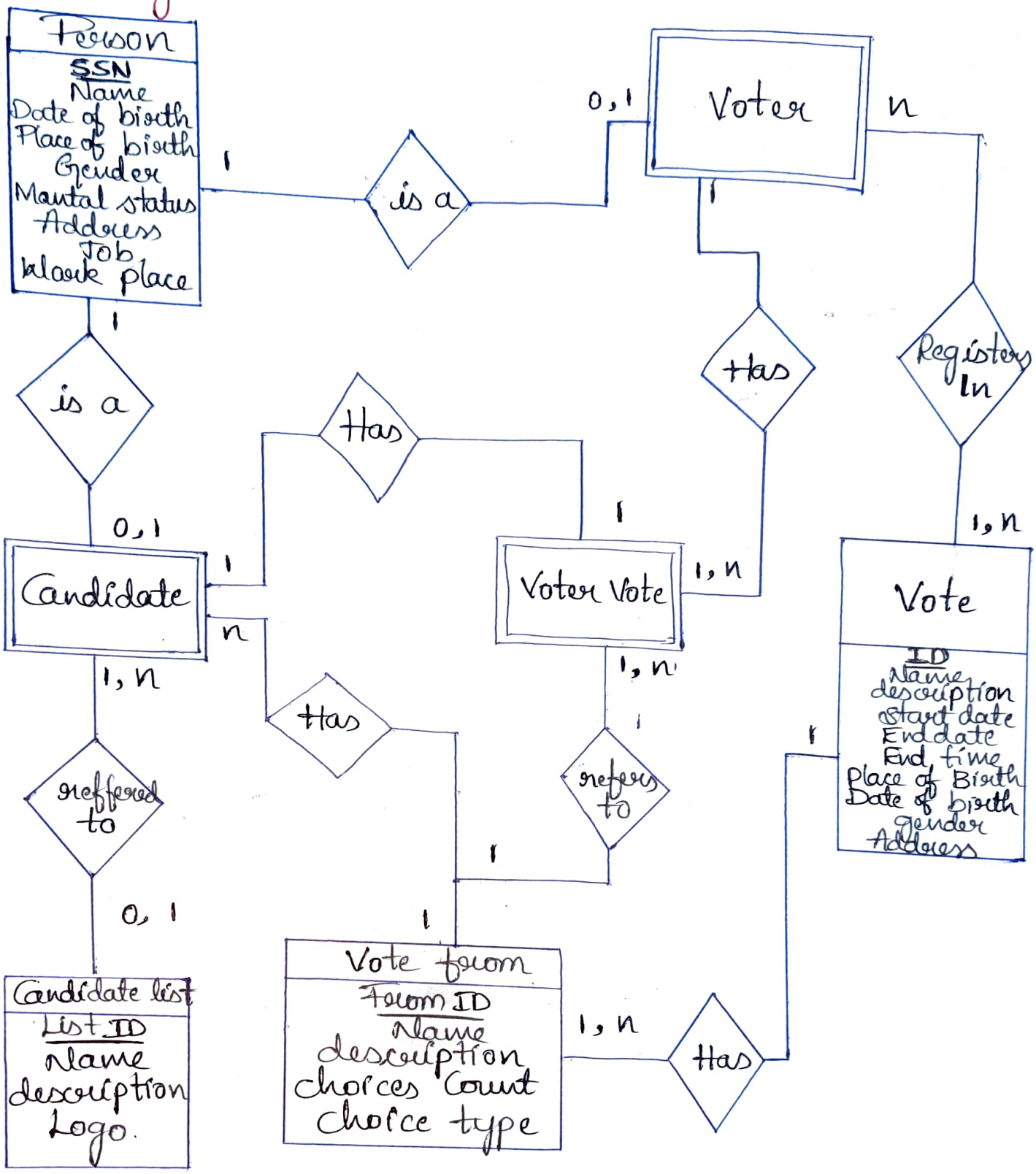
14) a) Design an ER-diagram for a University database schema and indicate all key and cardinality constraints. (July/August 2021) 10M



b) Design ER-diagram for company database with atleast four entities. (July / August 2021) - 8M, 4M



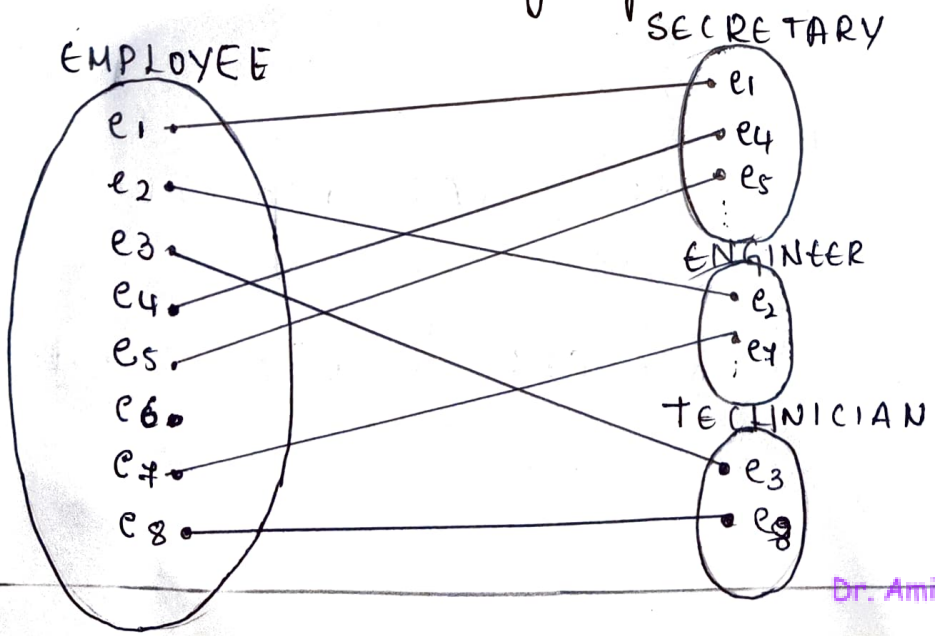
c) Design a ER diagram for election information system.



15. Write a short note on Specialization and Generalization, with an example each (Jan/Feb 2021,) (Aug/Sept 2020 - 17CS53)
Marks [6M]

Ans: Specialization

- Specialization is the process of defining a set of subclass of an entity type; this entity type is called the Superclass of the specialization.
- The set of subclass that forms a specialization is defined on the basics of some distinguishing characteristic of the entits in the Superclass.
- For example, the set of subclass {SECRETARY, ENGINEER, TECHNICIAN} is a specilization of the superclass EMPLOYEE that a distinguishes among employee entities based on the employee



Generalization:-

One can think of a reverse process of abstraction in which we suppress the differences among several entity types, identify their common features, and generalize them into a single superclass of which the original entity are special subclasses

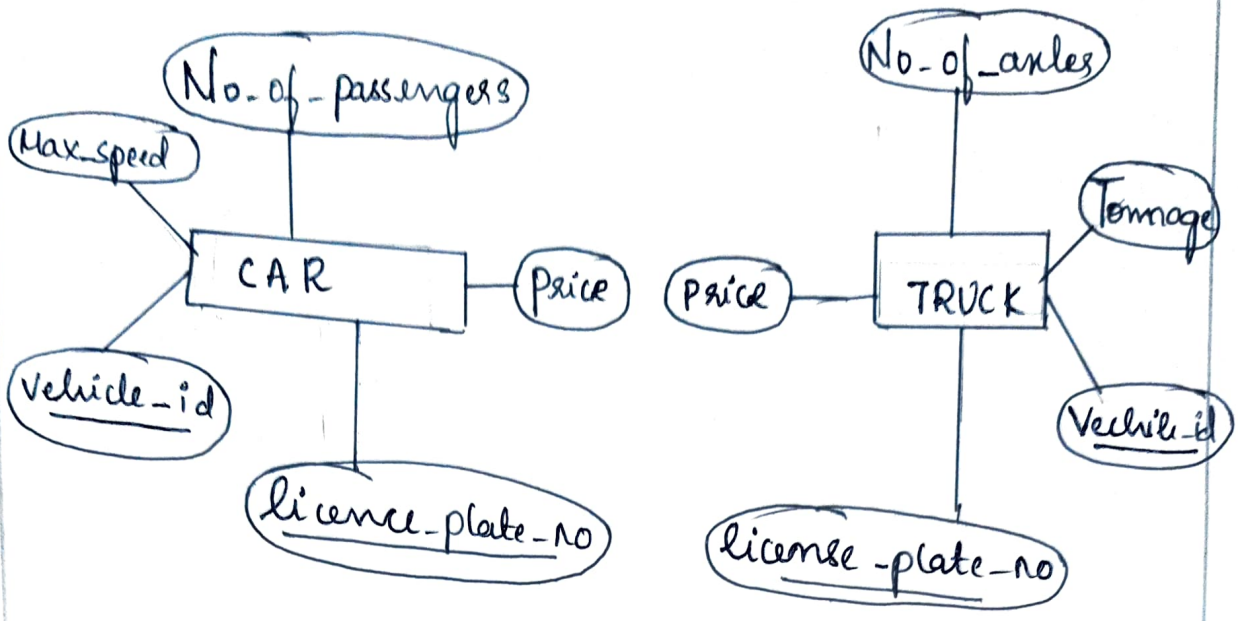
- Generalization follows the bottom-up approach. It generalizes or simplifies the entites.
- For example, Consider the entity type CAR and TRUCK shown in below figure. Because they have several common attributes they can be generalized into the entity type VEHICLE, as shown in figure.
- Both CAR and TRUCK are now subclasses of generalized superclass VEHICLE. We use the term generalization to refer to the process of defining a generalized entity of type from the given entity types

- Generalization

a) Two entity types, CAR and TRUCK

b) Generalizing CAR and TRUCK into the superclass VEHICLE.

(a)



(b)