"DBMS"

"UNIT 1"

↳ collection of inter-related data.

↳ **Database** → A database is a collection of related data. It has some implicit properties —

(i) It represent some aspect of the real world.

(ii) A database is a logically co-herent collection of data with some inherent meaning. A random collection of data cannot be considered as database.

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

* Defining → Constructing → Manipulating
  - → Retrieve
  - → Insert
  - → Modify
  - → Delete

↳ **DBMS** → is a collection of programs that enable user to create and maintain a database. It is a general purpose software, that facilitates the process of defining, constructing & manipulating database for various applications. Defining a database involves specifying the data types, structures & constraints for the data to be stored in database.

Constructing the database involves, the process of storing the data on some storage medium.

Manipulating the database involves, the functions such as — to retrieve, querying the database, updating the data, and generating reports from the data.

★ **Characteristics of Database:**

① **Self describing nature of database.** ⇒

~~In traditional file system each user defines and implement the file needed for a specific function~~

The fundamental characteristic of database, is that it not only contains the data, but a complete description of database structure & constraints. All these definitions are stored in catalog or data dictionary which is updated automatically, and contains the information of each file, the data type, and constraints of each data-items.

② **Insulation of data & program/ Data Abstraction** ⇒

In traditional file processing, the structure of data files is embedded in the access program, so any change made to the structure / file require changing all programs. By contrast, the DBMS applications do not require such changes since the structure of database files are stored separately, and we call this property as program data independence.

③ **Support Multiple views of data** ⇒

A database has many users, each of whom may require a different view of the database. So, depending upon what type of user is using the database, his rights (types of operations) change.

④ **Sharing of data/ Multi-user environment** ⇒
(Multi-transaction Processing System)

A multi-user DBMS must allow multiple users to access the database

at the same time. The DBMS must include concurrency control software to ensure, that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct.

⇒ **Problems of File System:**

1) **Data redundancy & inconsistency**
   (Repetition) (Multiple names of a single data member)
   ↳ In traditional file system, the various files are created for diff. applications. So, the same information may be duplicated in several files. This redundancy leads to the wastage of storage space, and also it results in data inconsistency, ie., having multiple copies of values for the same data-item.

2) **Difficulty in accessing data**
   ↳ Conventional file processing systems do not allows data to be retrieved in a convenient and efficient manner. Everytime, when the retriever condⁿ. change, it increases the program complexity.

3) **Data isolation**
   ↳ Since in file system, the data is scattered in various files, and files maybe in different formats. So, making a small change in the structure of data or writing new application using that data is quite difficult.

4) **Integrity Problem**
   ↳ The data values stored in the database must satisfy certain types of consistency constraints. In case of file system, the programmers enforce these constraints by adding appropriate code in the application program. Enforcing these constraints becomes more difficult when constraints involve data items from multiple files.

5) **Atomicity Problem**
   ↳ A computer system like any other system is subject to failure. In many applications, it is crucial that if a failure occurs, the data should be re-stored to the consistent state that existed prior to failure. But in case of file system, we write the data simultaneously to the file, so it becomes difficult to ensure atomicity.

6) **Concurrent Access Anomality**
   ↳ For the sake of well-performance of the system and faster response, many system allows multiple users to update the data simultaneously. To avoid inconsistency, the system must maintain some type of super-vision. But, implementing such type of supervision is very difficult, because data may be accessed by many different application programs that have not been co-ordinated previously

7) Security Problem

↳ File system provides more than one method to access the data. Therefore, enforcing constraints, like - to control unauthorized access is difficult in file system.

* Advantages Of Database :

① Control Redundancy
② Restricting unauthorized access.
③ Enforcing integrity constraints
④ Providing persistent storage for data & programs.
⑤ Providing multiple-user interface
⑥ Providing backup & recovery mechanisms.

Users Of Database

→ DBA :

In a database environment, the primary resource is the database itself, & the secondary resource is DBMS. Administrating these resources is the responsibility of DBA. The DBA is responsible for —

(i) Authorizing access to the database
(ii) For co-ordinating & monitoring its use.
(iii) For acquiring software & h/w resources.
(iv) For the security of database & is accountable for any security breach.
(v) For periodical backup & recovery.

→ Database Designer :

are responsible for identifying the data to be stored in the database, and for choosing appropriate structures to represent and store the data. It is the responsibility of database designer to communicate with the users in order to understand their requirement, and to come-up with a design that meets these requirement.

→ End Users :

- Casual End Users
- Naive Users → canned transaction
- Sophisticated End Users
- Stand alone Users

End users are the people whose job require access to the database for querying, updating & generating reports. There are several category of end-users —

(i) Casual End Users
↳ occassionally access the database, but they may need different information each time. All the middle-level managers and ocassional browsers come under this category.

(ii) Naive Users / Parametric Users
↳ make up a sizable portion of the database end-users. There main function revolves around constantly querying and updating the database, using standard type of queries, also called as "canned transaction".
All the front-desk workers, data-entry operators, clerks and clients come under this category.

ii) Sophisticated End-Users

↳ include engineers, scientist, data-analyst and all others who thoroughly familiarize with the facilities of DBMS, and know how to wright complex queries.

iv) Stand-alone Users

↳ They maintain personal database by using ready-made program packages that provide easy-to-use menu-based or graphic-based interface.

● System Analyst & Application Programmer

↳ DBA, DD comes under this.

System Analyst determine the requirement of end users and provides a co-ordination bw. the end-user, application programmer & Database designer.

Application Programmer is a person who develops the front-end programs, test them, de-bug them, and provide functionalities to use the database.

* Schema → structure of database
* Instance → snapshot of database at a particular time.

DATA MODELS :- ?

03/03/23

⇒ 3-schema Architecture Of DBMS :

The goal of 3-schema architecture is to separate the user applications, and the physical database. In this architecture, schemas can be defined at the following 3-levels:

1- Internal / Physical Level :

↳ It has the internal schema which describes the physical storage structure of the database, i.e., it describes how the data are actually stored. It describes complex low-level datastructures in detail.

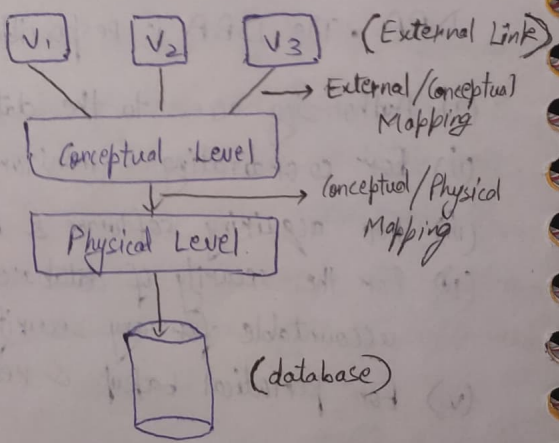
2- Conceptual / Logical Level :

↳ It has the conceptual schema, which describes the structure of whole database, ie, it explains what data are stored in the database, and what relationship exist among those data. It describes all the entities, data-types, constraints & operations.

3- External Level :

↳ It includes a no. of external schemas. Each schema describes the part of database that a particular user is interested, and hides the rest of the database from the user.

*



* Mapping — The 3 schemas are only the description of data. The only data that actually exist is at physical

level. Hence, the DBMS must transform a request specified on external schema into a request against conceptual schema, and then into a request on the internal schema for processing the stored database. So, the process of transforming request and results bw. different levels are called mapping.

Data Independence:

The 3-schema architecture can be used to explain the concept of data independence which can be defined as the capacity to change the schema at one level of database, without having change the schema at next higher level. We can define 2 types of data independence —

(i) Logical Data Independence – It is the capacity to change the conceptual schema without having to change the external schema or application programs. We may change the conceptual schema to expand the database, to reduce the database, to add or remove a constraint or to enable or disable a constraint

(ii) Physical Data Independence – It is the capacity to change the internal schema without having to change the conceptual schema. Changes to the internal schema may be needed to re-organize the files or to create additional access structures like index.

* Data Independence is achieved because – when the schema is changed at some level, only the mapping bw. the two levels is changed.

14/03/23

⇒ Database Interface:

(i) Menu-based interface for browsing
↳ these interfaces present the user with the list of options called menus. Menus allow not to memorize the specific commands and syntaxes. They are often called as browsing interface, which allow user to look through the contents of a database in an exploratory manner.

(ii) Form-based interface
↳ It displays a form to the user where the user can insert the data, search the relevant data, edit the data, or delete the required data.

(iii) Graphical User interface
↳ GUI displays a schema to the user in diagramatic form. Most GUI use a pointing device to pick certain parts of that displayed schema diagram.

(iv) Natural Language interface
↳ These interfaces accept request written in english or some other language. A natural language interface usually has its own syntax and a set of reserved words. After writing a query, it is interpretated, and the interface generates a high level query and submit it to the DBMS for further processing.

(v) Interface for Naive Users

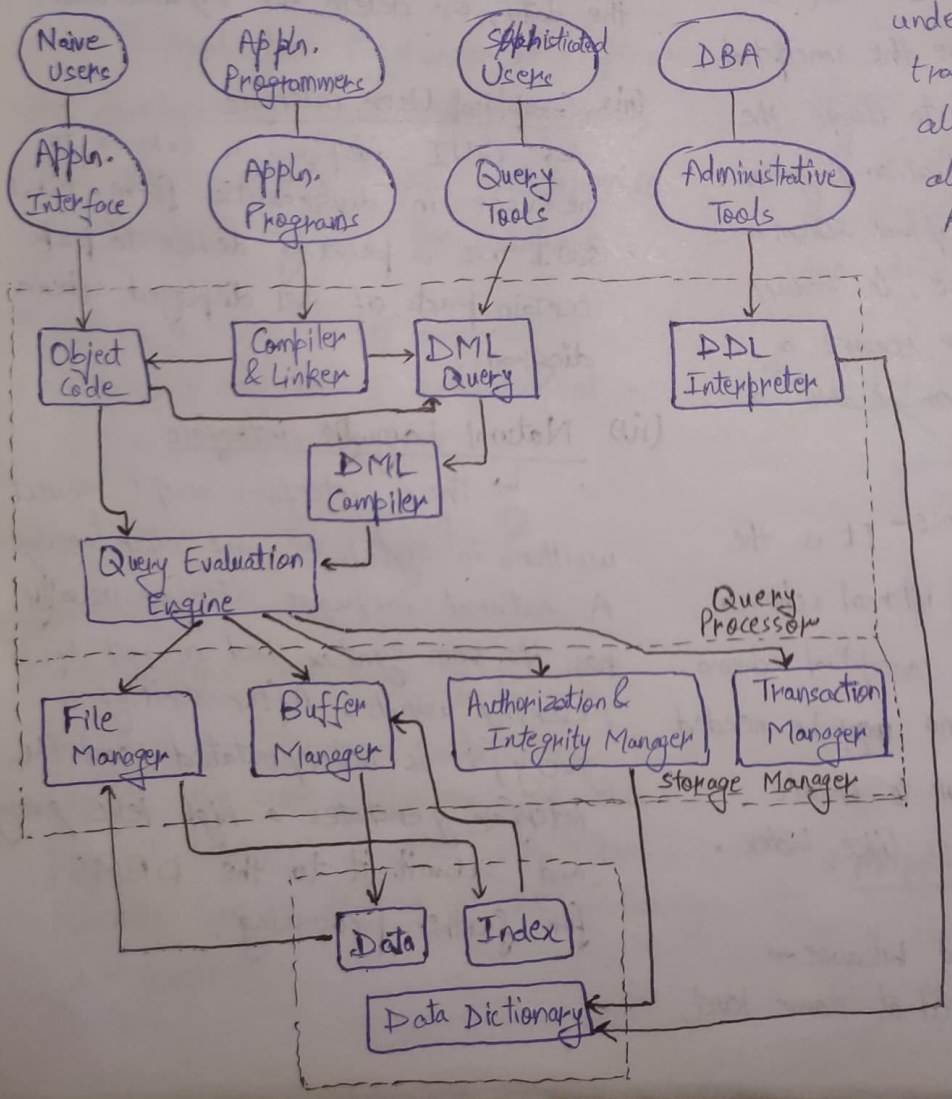
↳ Naive users such as the front-desk workers often have a small set of operations, that they must perform repeatedly. So the programmers design a special interface with the goal of minimising the no. of key strokes required for each request.

(vi) Interface for DBA

↳ These interfaces include the commands for creating accounts, setting system parameters, granting account authorization, and re-organizing the storage structure of database.

17/03/23

⇒ DBMS Structure — Query Processor, Storage Manager



A DBMS is partitioned into modules that deals with each of the responsibilities of the overall system. The functional components of a database system is broadly divided in two parts —

① Query Processor
② Storage Manager

① Query Processor → is important bcoz. it helps the database system to simplify & facilitate access to data. It includes the following components —

(i) DDL Interpreter - which interprets the DDL statement and record the definition in data dictionary.

(ii) DML Compiler - translates the DML statements into an evaluation plan consisting of low-level instructions that query evaluation engine understand. A query can usually be translated into more than one alternative evaluation plan that all gives the same results. All these evaluation plans are passed to query evaluation engine.

(iii) Query Evaluation Engine - is responsible for query optimization, i.e., it pick up the evaluation plan with lowest cost, and execute the query.

(2) Storage Manager → is a program that provides the interface b/w. the low-level data stored in database and the queries submitted to the system. It has the following components —

(i) File Manager – which manages the allocation of space on disk storage and data structures used to represent information on disk.

(ii) Buffer Manager – is responsible for fetching data from the disk storage into main memory and deciding which data to cache.

(iii) Authorization & Integrity Manager – it test for the satisfaction of integrity constraints and check the authority of user to access data.

(iv) Transaction Manager – it is responsible to ensure that the database system remains in consistent state inspite of any failure.

* Relationship → association b/w. two entities.

"UNIT-2"

"RELATIONAL DATA MODEL"

- Relations (Table)
- Tuples (Rows)
- Attributes (Column Header)
- Field Value
- Domain

→ RDM Concepts:

The relational model was first introduced by EF Codd in 1970. The model uses the concept of mathematical relation which looks somewhat like a table of values as its basic building block.

The relational model represents the database as a collection of relation. Each relation resembles a table of values and is assigned a unique name.

When a relation is thought of table, each row in the table (tuple) represents a collection of related values. In the formal terminology, a column header in the table is called "attributes".

• Domain – A Domain is a set of atomic values. By atomic, we mean that every value is indivisible. A common method of specifying a domain is to specify a datatype from which the data values can be taken. When we specify the datatype, we specify both the value and the format. A Domain is also given a name.

Note NULL is the value for every domain.

21/03/23

- A relation schema (R) denoted by -

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

is made up of a relation name R, and a list of attributes -

$A_1, A_2, \ldots A_n$

where each attribute $A_i \in dom(A_i)$ $1 \le i \le n$.

A relation state (r) is a set of 'n' tuples, i.e.,

$r = \{ t_1, t_2, \ldots t_n \}$

, where each tuple t is an ordered list of n values, i.e.,

$t = \langle v_1, v_2, \ldots v_n \rangle$

where each value,

$v_i \in dom(A_i)$ $1 \le i \le n$

Properties Of Relational DataBase -:

① Ordering Of Tuples in a Relation:

Since relation is set of tuples, mathematically elements of set do not have any order. Therefore, order of tuples in a relation doesn't matter.

② Cardinality of a Relation:

↳ No. of tuples in a relation.

③ Degree of a Relation:

↳ No. of attributes in a relation.

④ Values & Null in the tuple:

Each value in a tuple is an atomic value, within the framework of relational model. Hence, no composite or multi-valued attributes are allowed. However "NULL" value can appear in every column.

Integrity Constraints — : (IC)

checking correctness of data — to implement integrity

Integrity constraints are a way of implementing the rules in the database. It restricts the data that can be stored in relation. They help us to ensure the integrity (correctness of data).

* 4 types →
  - Domain
  - Entity integrity
  - Key constraints
  - Referential Intigrity constraints.

(i) Domain IC:

↳ specifies that the value of each attribute A must be an atomic value from the dom(A) (OR) D(A)., and should be of same data-type and format. The datatype includes string, integer, character, date, etc.

(ii) Entity IC

↳ states that no primary key value can be null. This is because the primary key is used to identify a tuple in the relation. Having null values for the primary key implies that we can't identify some tuples.

(iii) Key IC

(a) Keys are the entity sets that is used to identify an entity within a relation.

(b) A relation can have multiple keys, but out of all the keys, one key will be the primary key which will have the unique values and no null values.

24/03/23

(iv) Referential Integrity Constraints

↳ specified bw. two relations and is used to maintain the consistency among tuples of two relations. This rule is concerned with Foreign Key, ie, the attribute of a relation having domain that are the primary key of another relation. The rule can be specified as follows —

(a) Given two relations R and S. Suppose R refers to the relation S through a set of attributes that forms the primary key of S. Then, this set of attribute forms a foreign key in R. The values of the foreign key in R must be equal to the primary key of a tuple in S or a NULL value.

Ex!

dept (dept-no, dname) ≡ (S)
(dept-no ← Primary Key; S → referenced/parent)

Emp (Eno, ename, dept-no) ≡ (R)
(Eno ← Primary Key; dept-no ← Foreign Key; R → referencing/child)

CODD's RULE —:

1) Information Rule →

All information in Relational Database including tablename, column-names are represented by the values in tables.

2) Guaranteed Access Rule →

Every piece of data in a relational database can be accessed by using the combination of a table name and a column name.

3) Systematic Treatment Of Null Value →

RDBMS handels records that have unknown or in-applicable values (NULL values) in a pre-defined manner.

4) Active Online Catalog → (based on Relational Model)

The description of a database and its table are maintained in data-dictionaries which are updated automatically and can be queried using DML language.

5) Comprehensive Data Sub Language Rule →

A RDBMS may support multiple several languages, but atleast one of them should allow the user to do all the following operations —
define table and views,
Query and update the data,
set IC, set authorization,
& define transactions.

6) View Updating Rule →

Any view that is theoratically updatable can be updated using the RDBMS.

7) High Level Insert, update and delete →

The RDBMS supports insertion, updation & deletion at the table level, and the performance is improved since the commands effects multiple rows rather than a single record at run-time.

8) Physical Independence →

9) Logical Independence →

10) Integrity Independence →

Like table or view definition, integrity constraints are stored in the online catalog (data dictionary) and can

therefore be changed without necessiating any change in the application program.

11) Distribution Independence →
(Physical-Apart, Logically Same)

Application programs and the queries are not affected by the change in the distribution of physical data.

12) No Subversion Rule →

If the RDBMS has a language that access the information stored in database, then this language can't be used to bypass the IC.

25/03/23

"Entity Relationship Model"
(E.R. Model)

The ER Model percieves the real world consisting of basic objects called entities, attributes & relationship among those objects. The ER Model is useful in mapping the meanings and interaction of real-world situations onto a conceptual schema. It uses three basic notations- Entity sets; Attributes, & Relationship sets.

* Entity → is an object in the real world that is distinguishable from all other objects. Each entity has attributes, ie, the properties that describe it.

* Entity set → is a set of entities of the same type that share the same properties.

* Attributes → are the properties that are used to describe an entity. For each attribute, there is a set of permitted values called domain of the attribute. It can be of the following types—

(a) Composite Vs. Simple (Atomic)
↳ Composite Attributes can be divided into subparts which represent more basic attribute with independent meaning, for ex: Address.

Attributes that are not divisible are called atomic attributes.

Composite attributes can form a hierarchy. The value of a composite attribute is the concatenation of the values of its consituent simple attributes.

Composite attributes are useful to model a situation in which user sometimes refer to the composite attribute as a whole, but at other time refers specifically to its component. Ex: (Name)

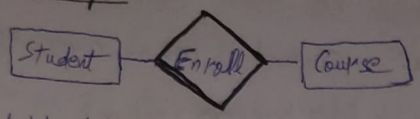
(b) Multivalued Vs. Single Valued
↳ Most attributes have a single value for a particular entity. Such attributes are called single valued attributes. For ex: Rollno, name. But, for some attributes, there is a set of values for a specific entity. Such attributes are called multi-valued attributes. A multi-valued attribute may have a lower and upper bound to constrain the no. of values allowed for a specific entity. Ex: (Phone)

(c) Derived
↳ The value of this type of attribute can be derived from the value of other attribute. The value of derived attribute is not stored, rather it is computed when required. Ex: (age)

— An attribute taken a NULL value when an entity does not have a value for it, ie, NULL value indicates not-applicable, unknown or something which is missing.

28/03/23

* Relationship →

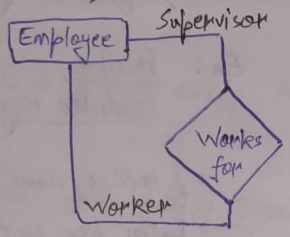
Ex: 

A relationship is an association among several entities. A relationship set is a set of relationship of the same type.

- **Degree Of A Relationship**
  ↳ is the no. of entities participating in the relationship. For ex, a relationship with two entities, is called binary relationship, with three it is called tertiary relationship, and for more, it is called "n-ary" relationship.

- **Descriptive Attributes**
  ↳A relationship may also have certain attributes. Those attributes are called as discriptive attributes.

- **Role name & recursive relationship** –:

Ex: 

Each entity type that participates in a relationship plays a particular role in the relationship. The role name signifies the role that a participating entity plays in each relationship. Since, entity set participating in a relationship are generally distinct, so the roles are implicit and are not usually specified. However, there is the case when the same entity participates in a relationship more than once in different roles. In such cases, the role name becomes essential for distinguishing the meaning of each participation. Such relationship are called recursive relation. For ex, when an employee works for another employee, where we have a relationship "Works for", where both the entities are employees. So, we need to specify the role of each employee.

- **Constraints on relationship types** –

Cardinality Ratio (OR) Mapping Constraints
Participation Constraints.

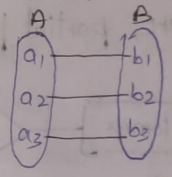
↳ Relationship type usually contain constraint that limit impossible combinati-ion of the entities that may participate in the corresponding relationship set.

There are 2 main types of relationship constraints :

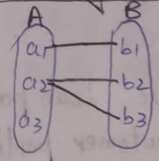
(i) Cardinality Ratio (OR) Mapping Constraint

It express the no. of entities to which another entity can be associated. Let us assume we have two entities A & B, and a relationship R.

- **One-to-One Relationship**
  An entity in set A is associated with atmost one entity in B, And, an entity in B is associated with atmost one entity in A.

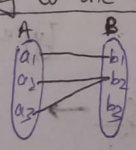


- **One-to-many Relationship**



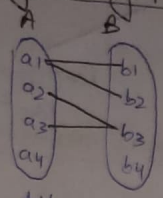
An entity in A is associated with any no. of (0 to more) of entities in B. An entity in B can be associated with atmost one entity in A.

- **Many-to-one Relationship**



An entity in A is associated with atmost one entity in B. An entity in B can be associated with any no. of entities in A.

- Many-to-many Relationship

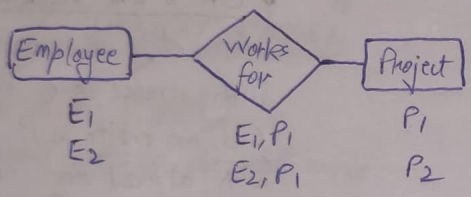


An entity in A is associated with any no. of entities in B. And, an entity in B is associated with any no. of entities in A.

(ii) Participation Constraint

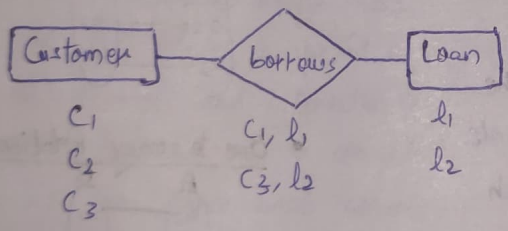
The participation of an entity set E in a relationship set R is said to be total if every entity in A participates in atleast one relationship in R. If only some entities in E participates in relationship R, the participation is said to be partial.

Ex:



* Here, partial participation is happening.

Ex:



* Here, Loan participation is total, while Customer participation is partial.

31/03/23

⇒ KEYS –:

A Key is an attribute or a set of attributes which help us in uniquely identifying the rows of a table. Types →

(i) Super Key → is a combination of all possible attributes which can uniquely identify the rows in the table. This means that a super key may have some extra attributes which is not necessary for uniquely identifying the rows in the table.

Ex: Student { roll_no, name, reg_no }

Possible Super Keys:
① { roll_no }
② { reg_no }
③ { roll_no, name }
④ { roll_no, reg_no }
⑤ { reg_no, name }
⑥ { roll_no, name, reg_no }.

(ii) Candidate Keys → is a minimal super key with no redendent attributes. It is called a minimal super key because we select a candidate key from a set of super key such that the selected candidate key is the minimum attribute require to uniquely identify a tuple in table.

Ex: Possible Candidate Keys:
① { roll_no }
② { reg_no }

* { roll_no, name }
↳ This key can't be considered as candidate key because when we take the subset of this key, we get two attributes, 1st is roll_no, 2nd is name. Rollno is a candidate key, so it is not a minimal super key. Hence, it can't be taken as a candidate key.

(iii) Primary Key → is the minimal set of attributes which uniquely identify any tuple in a table. It is selected from the list of candidate keys, i.e., any candidate key can become a primary key and it is choosen by DBA or data expert.

(iv) Alternate Key → All the candidate keys which are not primary keys are called alternate keys.

For ex. in the above case, {roll no} & the {reg. nos} are the candidate keys. It has been choosen {roll. nos} to be the primary key, then {reg. nos} is the alternate key.

Strong & Weak Entity :

An entity having sufficient attributes which can be used as candidate key or primary key is a strong entity.

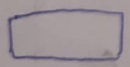
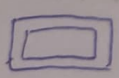
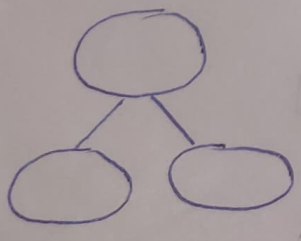
Entity types that do not have the key attributes of their own and depends on another entity for their existence are called weak entities.

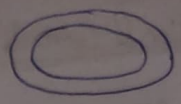
For ex, Student is a strong entity, while Fee Reciept is a weak entity.

Entities belonging to weak entity types are identified with another entity sets called "Identifying Entity Set". Every weak entity must be associated with an identifying entity, and such relationship is called "identifying relationship."

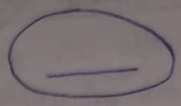
A weak entity always have a total participation.

⇒

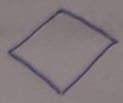
Symbols	Descriptions
	Strong Entity / Regular
	Weak Entity
	Attribute
	Composite Attribute



Multi-valued attribute

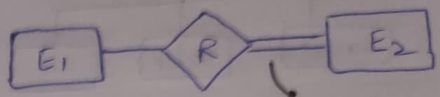


Key attribute



Relationship

Identifying Relationship



Total Participation of E_2 in R.

→ one-to-many

01/04/23

⇒ Extended ER Diagram :

→ Specialization → is the process of defining a set of sub-class of an entity type. This entity type is called "Super class". The set of sub-classes that forms a specialization is defined on the basis of some distinguishing characteristics of the entities in the super class.

It is the process from top to bottom.

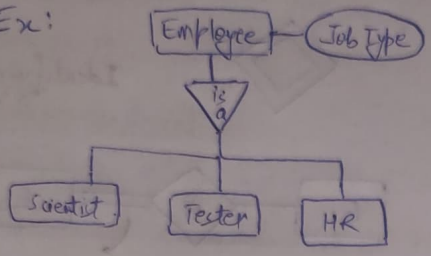
→ Generalization → For all practical purpose, generalization is simply the inversion of specialization. Generalization proceeds from the recognition that a no. of entities share common features. On the basis of their common features, generalization synthesize these entity sets into a single higher level entity sets.

Constraints on Specialization & Generalization:

1) Condition defined (OR) Attribute defined →

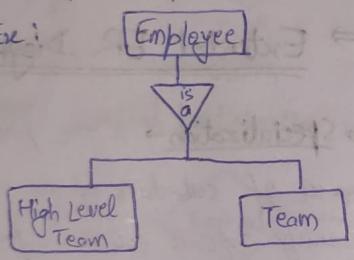
In this, lower level entity sets membership is evaluated on the basis of whether an entity satisfies an explicit condition or a value or attribute.

Ex:



2) User defined → User-defined lower level entity sets are not constraint by a membership condition. Rather, the database user assign entities to a given entity set. Such type of specialization is called user-defined specialization.

Ex:

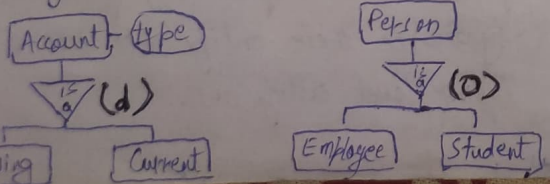


08/04/23

[redacted]: A second type of constraint relates to whether an entity belong to more than one lower level entity within a single generalization. The lower level entity maybe of one of the following —

(i) Disjoint → An entity may belong to not more than one lower level entity set.

(ii) Overlapping → The same entities may belong to more than one lower level entity set within a single specialization.



The constraint involve determining which entity can be the member of given lower level entity set. such membership may be one of the following :-

3) Completeness Constraint →

(A) Total Specialization:

This constraint specializes that every entity in the super class must be the member of at least one sub-class in specialization.

{ : Represented by double line }

(B) Partial Specialization:

It allows an entity in super class may or may not belong to any of it's sub-class.

AGGREGATION →:

One limitation of ER model is that it can't express relationship among relationship. One method of doing this is to have quaterly relationship, but it will result in much redundant information. So, the best way to model this situation is to use the concept of "aggregation". Aggregation is an abstraction through which relationship are treated as higher-level entity sets.