# 1.1 Introduction to DBMS

collate Notes

A database management system is a collection of interrelated data a set of programs to access these data. The collection of data referred to as a database contains information relevant to the enterprise.

The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

A DBMS provides an interface between the state and the software application.

It accepts the request for data from an application and instruments the operating system to provide the specific data i.e, helps in storing and retrieving data.

## ↳ Characteristi of a DM



1. Provides security and removes redundancy – DBMS provides security to the data stored in it because all uses have different rights to access the database. Some users can access the whole database while either can access a small part of it.
For Example, a Returner can access files related to his/her subjects but HOD of the department can

access files of all subject related to their department. It can reduce data redundancy and inconsistency by minimising isolated files in which the same data are repeated by using the method of Normalisation.

2. Self-describing Nature – A DBMS should be self describing in nature as it not only contains the database but also the metadata. A metadata (data about data) defines and describes not only the extent, type, structure and format of all data but also relationship between the data.

3. Support ACID properties – Any DBMS is able to support ACID (Atomicity Consistency, Isolation and Durability) properties. It is made sure in every DBMS that the real purpose of data should not be lost while performing transactions like delete, and update.
Example, if an employee age is updated then it should ensure that there is no duplicate data & no mismatch of employee information.

4. Concurrent use of database – A DBMS should support the concurrent use of the database i.e. when many users are accessing the data at the same time without any problem with the help of concurrency, economy of the system can be increased.

Example, Employees of railway reservation system can book and access tickets for passengers concurrently.

5. Insulation between data & program – In DBMS, the structure of data files is stored in the system catalogue. It helps in internal improvement of data efficiency or any changes in the data do not have effect on the application software.

6. Data Persistence – Persistence means if the data is not moved explicity then all the data will be maintained in DBMS. Any data stored in DBMS can never be lost. If system failure happens during or any transaction then it will be solved back or fully completed, but there is no risk to the data.

7. Backup and Recovery – It is possible that a whole database fails and backup of database must be taken so that it can be restored whenever needed. A database must have this characteristcis to enable more effectiveness.

8. Multiple Views – Users can have multiple views of database depending on their department and interest. DBMS supports multiple views of database to the users.

9. Represents complex relationship between data – data stored in a database is connected with each other and a relationship is made in between data. DBMS should be able to represent complex relationships between data to make efficient and accurate use of data.

10. Stores any Kind of data – A DBMS should be able to store any kind of data. Any kind of data that exists in the real world can be stored in a DBMS.

## ↳ Advantages of DBMS

1. It offers a variety of Techniques to store and retrieve data.
2. It serves as an efficient handles to balance the needs multiple application using the same data.
3. Application programmers nevers exposed to details of data representation and storage.
4. offers data integrity and Security.
5. It implies integrity constraints to get a high level of protection against prohibited access to data.
6. It schedules concurrent access to the data in such a manner that only one user can access

the same data at a time.

7. Reduced application development time.
8. Uniform administration procedures for data.

## ↳ Disadvantages of DBMS

1. Cost of hardware & software is quite high.
2. Most DBMS are complex system, so training for users to use the DBMS is required.
3. If all data is integrated into a single database then it is possible to risk losing the data in case of electric failure or corruption of database.
4. It cannot perform sophisticated calculations.
5. Use of the same program at a time by many users sometimes leads to the loss of some data.
6. Frequent upgrade of DBMS to add new functionality may make it difficult for users to work with new commands and rules.
7. As DBMS becomes big software due to its functionalities it requires lots of space & memory to run its application efficiency.

# 1.1.2 Application of DBMS

CollateNotes

Database are widely used there are some representative applications.

(1) Banking for customer information, accounts, loans and Banking transactions.

(2) Airlines For reservations and schedule information. Airlines were among the first to use database in a geographically distributed manner.

(3) Universities For student information, course registration and grades.

(4) Credit and transaction For purchase on credit cards and generation of montly statements.

(5) Telecommunication For keeping records of calls made, generating montly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.

(6) Finance For storing information about holdings, sales and purchases of financial instruments such as stocks and bonds; also for storing real time market data to enable online trading by customers and automated trading by the firm.

(7) Sales For customers, product and purchase information.

(8) On-line retailers for sales data noted above plus online order tracking, generation of recommendation

lists, and maintanance of online produce evaluation.

(9) Manufacturing for management of the supply chain and for tracking production of items in factories, inventories of items in warehouses and stores and orders for items.

(10) Human Resource Management for information about employees, salaries, Payroll taxes, benefits and for generation of paychecks.

## 1.2 Data Independence



Data Independence is defined as property of DBMS that helps you to change the database scheme at one level of database system without requiring to change the schema at the next higher level. Data independence helps to keep the data separated from all programs that make use of it.

Types of Data Independence

(1) Physical data independence
(2) logical data independence.

↳ Physical Data Independence

Physical data independence helps you to separate conceptual levels from the internal / physical levels.

It allows you to provide a logical description of the database without the need to specify Physical structure.

It is easy to achieve physical data independence as compared to logical independence.

you can easily change the physical storage structure or drives with an effect on the conceptual scheme.

Any change done would be absorbed by the mapping between the conceptual and internal levels.

It is achieved by the presence of internal level of database and then transformation from the conceptual level if the database to the interval level.

Example : Due to physical independence, any of the below change will not affect the conceptual layer.

1. Modifying indexes.
2. Changing the access method
3. Switching to different data structures.
4. Using a new storage device like hard drive or magnetic tapes.
5. Modifying the file organisation technique in the database.
6. Change of location of database from say C to D drive.

## ↳ logical Data Independence

Logical data independence is the ability to change the conceptual schema without changing the external view or the external API / programs Any change made will be absorbed by the mapping between external and conceptual levels. It is difficult to acheive logical data independence as compared to physical data independence.

Examples - Due to logical independence, any of the below change will not affect the external layer

(1) Add / modify / delete a new attribute, entity or relationship is possible without a rewrite of existing appercation programs.

(2) Merging two seconds into one.

(3) Breaking an existing record into two or more seconds.

## ↳ Importance of data Independence

(1) It helps in improving the quality of data.

(2) It makes database system maintenance affordable.

(3) Database incongruity is reduced significantly.

(4) No need to alter data structure is application programs.

(5) It allows to improve state which is undamaged or undivided.

(6) Easily make modifications in the physical level is needed to improve the performance of the system.

## ↳ Difference between Physical & Logical data Independence

| | Logical Data Independence | Physical Data Independence |
|---|---|---|
| (1) | It is mainly concerned with the structure or changing the data definition. | Mainly concerned with the storage of the data |
| (2) | Difficult to acheive as retrieval of data is dependent on the logical data independence. | Easier to retrieve as compared to logical data independence. |
| (3) | changes to be made in application program if new fields are added / deleted from the database. | changes in the physical level usually do not need change at the application program level. |
| (4) | Concerned with conceptual schema. | Concerned with internal schema. |

| | |
|---|---|
| (5) Modification at logical level is significant whenever the logical structure of the database are changed. | Modification made at the internal level may or may not be needed to improve the performance of the structure. |
| (6) Example - Add/ modify / delete a new attribute. | Example- change in storage devices, hashing algorithms, compression techniques etc. |

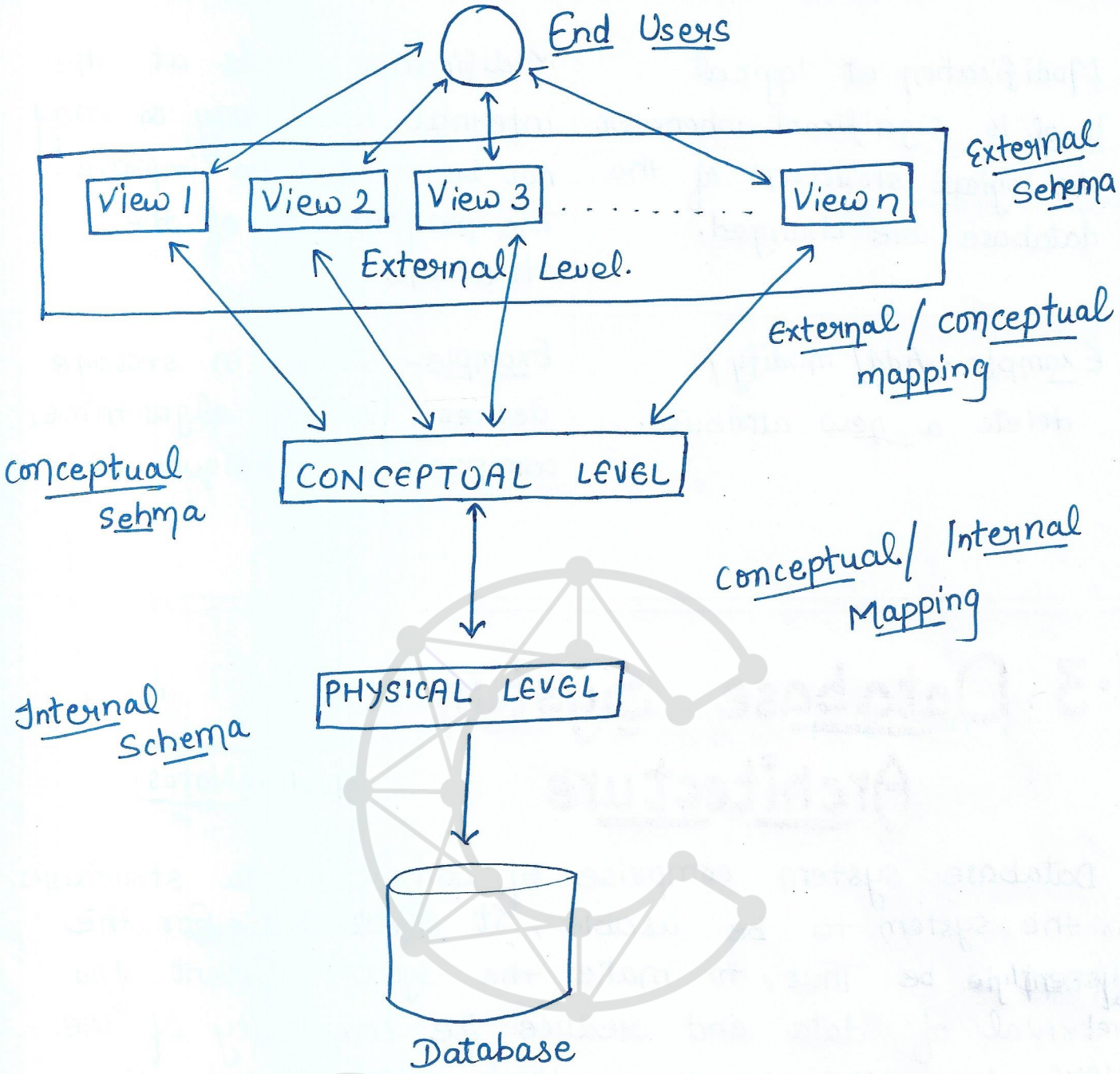
# 1.3 Database System Architecture

CollateNotes

Database system comprise of complex data structure. For the system to be usable, it must retrieve data efficiently... Thus, to make the system efficient for retrival of data and reduce the complexity of the users, developers use the method of data abstraction.

Following are the three levels of database architecture:-

(1) Internal / Physical level.
(2) Conceptual / logical level.
(3) External / View level.




1. Internal level / Schema

The internal schema is the lowest level representation of the entire database. It defines the physical storage structure of the database.

It contains multiple occurance of multiple type of interval secerd and is also called "stored second."

It helps to keep information about the actual representation of the entire database.
The internal view tells what data is stored in the database and how.
It never deals with the physical devices, instead views a physical device as a collection of physical pages.

2. Conceptual Schema / level.

Conceptual schema defines the structure of the whole database for the users.
This schema hides information about the physical storage structure and focuses on describing data types, entities, relationships etc.
It is defines all database entries attributes and their relationships.
It also defines security and integrity information.
In this level, the data available to a user must be contained in or derivable from the physical level.
It is also called the 'data model'.

3. External Schema / level

An external schema defines the part of the database which is viewed by individual users.
It hides the unrelated details of the database from the users.
This level is the closest to the users. Each external view is defined using an external Schema, which

consist of definitions of various types of external record of that specific view. An external view is just the content of the database as it is seen by specific particular user.

The external schema describes the segment of the database which is needed for a certain user group and hides the remaining details from the database from the specific user group.

## 1.3.1 Database System Architecture Layer Mapping

CollateNotes

The process of transforming request and results between these levels is called Mapping.

There are two types of mapping :-

- Conceptual / Internal Mapping
- External / Conceptual Mapping.



1. Conceptual / Internal Mapping

It defines the correspondence between the conceptual view and the stored database. It specifies how conceptual record and fields are represented at the internal level. It relates conceptual schema with the internal schema.

If structure of the stored database is changed ie. changes are made to the storage structure definition then the conceptual / internal mapping must be changed accordingly, so that the conceptual schema can remain invariant.

2. External / Conceptual Mapping

It defenis the correspondance between a particulare external view and conceptual view.

It relates each external schema with conceptual schema.

The differences that can exist between these two levels are analogous to those that can exist between the conceptual view and the stored database.

Any number of external views can exist at the same time, any number of users can share a given external view. where different external views can overlap.

There could be several mapping between external and conceptual levels.

Example- Fields can have different data types, fields and second name can be changed several conceptual fields can be combined into a single external field.

# 1.4 Database Users And DBA

CollateNotes

↳ Database Users

There are 4 different types of database system uses differentiated by the way they expect to interact with the system.

The different types of database users are given as:-

(1) Waive Users

They are unsophisticated users who interact with the system by involving one of the application programs that have been written previously.

The typical user interface for waive users is a forms interface, where the user. can fill in appropriate fields of the forms.

Naive users may also simply read reports generated from the database.

Example - Consider a user who wishes to find her amount balance over the world wide web. Such a user may access a form where she enters her amount number.

(2) Application programmers

They are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces.

RAD (Rapid application development) tools are tools that enable an application programmer to construct forms and reports with minimal programming effort.

(3) Sophisticated users

These users interact with the system without writing programs.

Instead, they from their requests in a database query language.

They submit each such query to a query processor, whose function is to break down DML (data manipulation language) statements into instructions that the storage manager understands.

Analysts who submit queries to explore data in the database fall in the category.

(4) Specialized Users

These users are sophisticated users who write specialized database applications that do not fit into the traditional data processing framework. Among these applications are computer aided design system, knowledge based and expert system, systems that store data with complex data types and environment modeling system.

## ↳ DBA (Database Administration)

A person who has central control of both the data and the programs that assets those data is called the database administrater. A DBA provides

necessary technical support for implementing a database.

A DBA works on aspects such as design development, testing and operational phases of the system.

The functions of a DBA includes-

(1) Schema difinition

The DBA creates the orginal database schema by executing a set of data definition statement in the DDL.

Defines the logical Schema of the database ie, the overall logical structure of the database.

(2) Storage structure and access method definition

The DBA decides how the data is to be represented in the stored database.

(3) Schema and Physcial-organisation modification

The DBA carries out changes to the schema and Physical organisation to reflect the changing needs to the organisation, or to after the physical organisation to improve performance.

(4) Granting of authorisation for data access

By granting different types of authorization, the database administration can regulate which parts of the database various users can access. The authorisation information is Kept in a special system structure that the database system consults whenever someone attempts to access the data in the system.

(5) Routine Maintenance

(i) Periodically backing up the database, either onto tapes or onto remote servers, to prevent loss of data in case of disasters.

(ii) Ensuring that enough free desk space is available for normal operations, and upgrading desk space as required.

(iii) Monitoring jobs running on the database and ensuring that performance is not degraded by very expensive tasks submitted by the access.

(6) Approving data access

The DBA determines which users need access to which part of the database.
According to this, various types of authorizations are required / granted to different users.

(7) Assisting application programmers

The DBA provides assistance to application programmers to develop application programs.

# 1.5 Entity-Relationship Model

CollateNotes

The entity relationship model is a data model based on a perception of a real world that consists of a collection of basic objects.

The ER is a high level conceptual data model diagram.

It based on the nation of real-world entities and the relationship between them.

ER modeling helps to analyze data requirement systematically to produce a well-designed database.

It is considered a good practice to complete ER modeling before implementing your database.

## ↳ ER Diagram

ER diagram displays the relationships of entity set stored in a database. It helps in explaining the logical structure of database. ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications. They are translatable into relational tables which allows for quick building of database.

It provides a preview of how all your tables should convert and what fields are going to be on each table and describe the entities, attributes

and relationships. Components of ER diagrams.
The ER model is based on three basic concepts. namely.

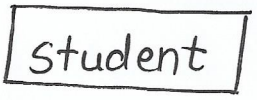
(1) Entity
(2) Attributes
(3) Relationships

(1) Entity An Entity is a real-world thing or object that is easily recognizable.
It may be a physical things or an event that happens in the world or an object in the database.
An entity can be a place, person, object, event or a concept, which stores data on the database.
Every entity is made up of some 'attributes' which represent that entity.

Example Person - employee, student, patient
Place - store, building
object - Product, car, Machine
Event - Sale, Registration
Concept - Course, Account.

Entity Set
An Entity set is a group of similar kind of entities with attributes sharing similar values. Entities are represented by their properties called attributes. All attributes have their separate values.

Example – A student entity may have a name, age, class as attributes.




Rectangle is used to represent an entity set.

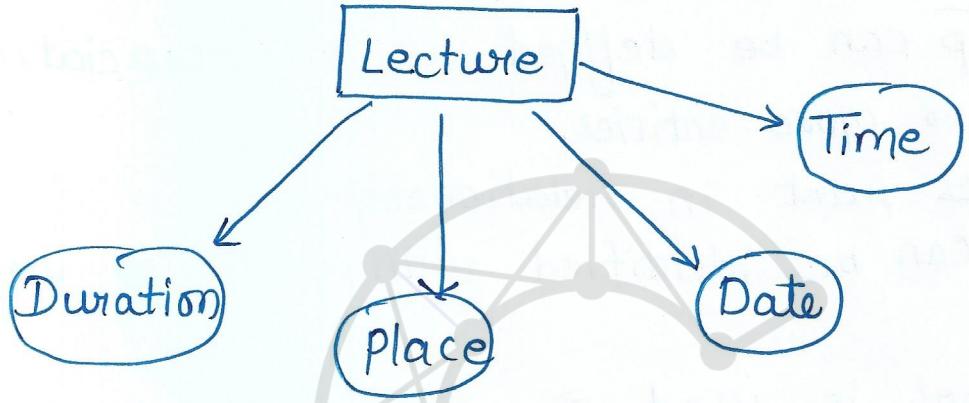
## Difference between strong Entity Set & weak Entity set

| | Strong Entity Set | Weak Entity Set |
|---|---|---|
| (1) | Always has a primary key. | It doesn't have enough attributes to build a primary key. |
| (2) | Represented by rectangle symbol. | Represented by a double rectangle symbol. |
| (3) | Contains primary key represented by the under line symbol. | Contains a partial key represented by a dashed underline symbol. |
| (4) | The relationship between two strong entity sets in shown by using a diamond symbol. | The relationship between are strong & a weak entity set is shown by double diamond symbol. |
| (5) | The connecting line of strong entity set with the relation ship is single line. | The line connecting the weak entity set for identifying relation-ship is a double line. |

(2) Attributes

An attributes is a single valued property of either an entity type or a relationship - type. An attributes is represented by an ellipse.

Example - A lecture can have attributes : time, date, place, duration etc.




## ↳ Types of Attributes :-

(1) Simple attributes - A simple attributes cannot be divided any further i.e. it is an atomic value.
Example - Student's contact number.

(2) Composite attributes - A composite attributes can be broken down into more attributes.
Example - Name attributes can be divided into first, second and last name.

(3) Derived attributes - The values are derived from other attributes present from the date of birth.
Example - Age derived from the date of birth.

(4) Multivalued attribute - Multivalued attributes can have more than one values.
Example - A person / employee can have more than one mobile number, email address etc.

(3) Relationship

Relationship can be defined as an association among two or more entities.
Entities takes part in relationships.
Relationships can be identified with verbs or verb phrases.
Diamond Symbol is used to represent set of relationships among a member from each of several entity sets.

Cardinality

It is used to define the numerical attributes of the relationship between two entities or entity sets.
Different types of cardinal relationships are :-

1. One-to-One

One entity from entity set X can be associated with at most one entity set Y and vice-versa.
Example - One student can registar for numerous courses but all those conses have a single line back to that one student.


2. One-to-many

One entity from entity set X can be associated with multiple entities of entity set Y but an entity from entity set Y can be associated with at least one entity.

Example - One class consisting of multiple students.


3. Many-to-one

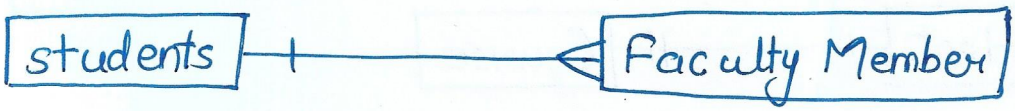
More than one entity from set X can be associated with at most one entity of entity set Y. However, an entity from entity set Y may or may not be associated with more than one entity from entity set X.

Example - Many students belong to the same class.

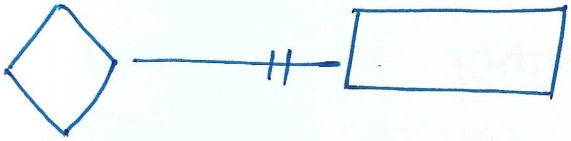

4. Many-to-Many

One entity from X can be associated with more than one entity from Y and vice-versa.

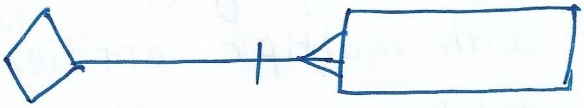
Example - student as a group associated with multiple faculty member, and faculty member can be associated with multiple students.



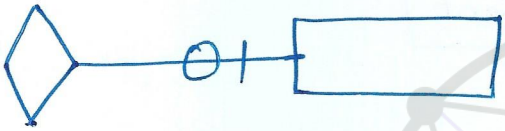

Relationship cardinality -



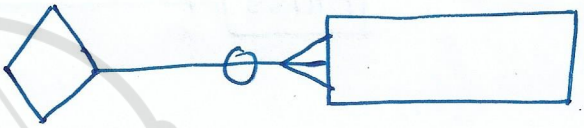
Manadatary one



Mandatory many



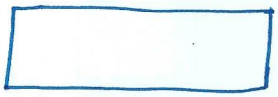
optional One



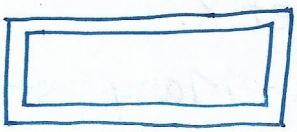
optional Many

ER Diagram Natations

(1) Rectangles - represents entity types.



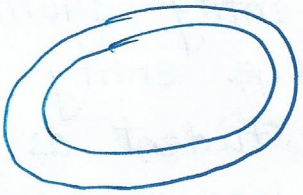
Entity / strong Entity



weak entity

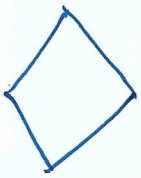
(2) Ellipses - Represents attributes

Attributes



Multivalued attributes

(3) Diamonds - represents relationship types.



Relationship

Weak relationship

(4) Lines - It links attributes to entity types and entity types with other relationship types.

(5) Primary Key - attributes are underlined.

# 1.6 Constraints

CollateNotes

An ER enterprise schema may define certain constraints to which contents of the database must confirm.

Types of constraints namely :-

(1) Mapping cardinalities.

(2) Key constraints

(3) Participation constraints

(4) Domain Constraints

(5) Integrity Constraints.

## (1) Mapping Condinalities

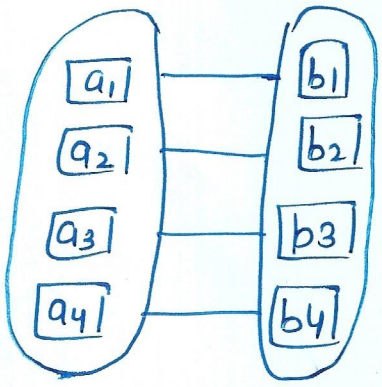
Mapping condinalities or condinality ratios, express the number of entities to which another entity can be associated via a relationship set. These cardinalities are useful in describing binary relationship set and also for relationship sets that involve more than two entity sets. For a binary relationship set R between entity sets A and B, the mapping condinalities must be one of the following :-

(1) One-to-One - An entity in A is associated with at most one entity in B and vice-versa.

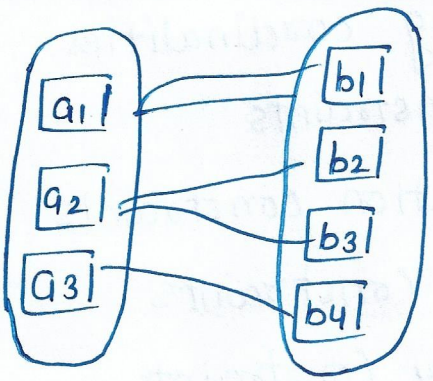
(2) One-to-many - An entity in A is associated with any number of entities in B. An entity in B however, can be associated with atmost one entity in A.

(3) Many-to-one - An entity in A is associated with atmost one entity in B. An entity in B, However can be associated with any number of entities in A.

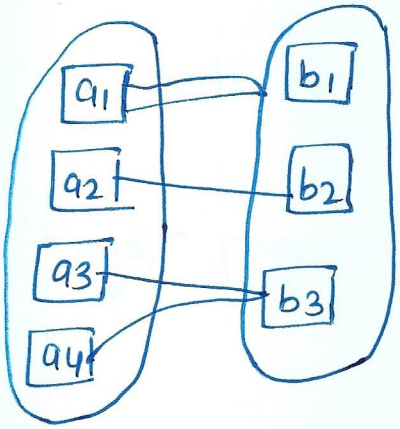
(4) Many-to-Many - An entity in A is associated with any number of entities in B & vice-versa.



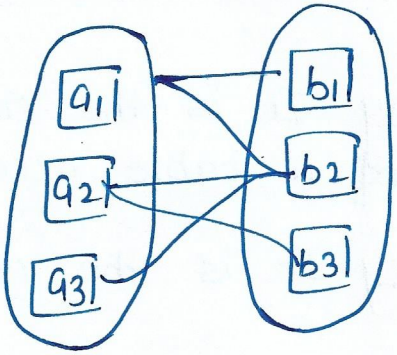

one-to-one




One-to-many




Many to one




Many to Many

2. Key Constraints

Keys are the entity set that is used to identify an entity within its entity set uniquely.
An entity set can have multiple keys, but out of which one key will be the primary key.
A primary key can contains a unique and non null value in the relational table.

There are 6 Types of Key constraints given here :-

(1) Super Key collection of all the key present in all table.

(2) Candidate Key It is a super key with no redundancy. Here minimum no. of cenbinations of keys are used.

(3) Primary Key It is a set of values which Uniquely identify the table for data retrieval.

(4) Foreign Key: If we are referring a primary key of one table to another then we call it as foreign key.

(5) Alternate Key It is the combination of all the attributes of a table except primary key.

(6) Composite Key It is the combination of primary keys.

(3) Participation Constraints

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

Example-

Every loan entity expected to be related to atleast one customer through a borrower relationship. Hence, the participation of loan in the relationship set borrower is total. In contrast, an individual can be a bank customer whether or not they have a loan with the bank. Hence, the participation of Customer in the borrower relationship set is partial.

(4) Domain Constraints

A domain is a unique set of values permitted for an attribute in a table. For Example, a domain of month-of-year can accept January, February..... December as possible values.

Domain constraints are user defined data type and we can define them like this.

Domain Constraint = data type + Constraints.

(5) For Integrity Constraints

Integrity Constraints are a set of rules. It is used to maintain the quality of information. Integrity constraints ensure that the data insertion, updating and other processes have to be performed in such a way that data integrity is not affected. Thus, integrity constraint is used to guard against accidental damage to the database.

# 1.7 Keys

CollateNotes

A DBMS Key is an attributes or set of an attributes which helps you to identify a raw (Tuple) in a relation (table).

They allow us to find the relation between two tables.

Keys also helps in uniquely identifying a raw in a table by a combination of one or more columns in that table.

Reasons for using Keys in DBMs:

(1) Allows to establish a relationship b/w and identify the relation b/w tables.

(2) Helps to enforce identify and integrity in the relationship.

(3) A table can contain thousands of records with duplicate values. Here, Keys ensure that you can uniquely identify a table second.

## Various types of Keys in DBMs :

### (1) Super Key

A SuperKey is a group of single or multiple Keys which identify rows in a table. A super Key which may have additional attributes that are not needed for unique identfication.

Examples

| EmpNun. | Emp ID | Emp Name |
|---|---|---|
| 9814527 | A5 | shawn |
| 1246718 | A6 | James |
| 1999345 | A7 | Emily |

Here, EmpNun & EmpID are super Keys.

### (2) Primary Key

A column or group of column in a table which help to identify every low in that table is called a primary Key.

(1) Two rows can't have same primary Key value.

(2) The primary Key field can't be null.

(3) The value in a primary Key column cannot be modified / updated if any foreign Key refers to that primary Key.

Example

| Stu ID | Roll No. | Name |
|---|---|---|
| 1 | 27 | Tom |
| 2 | 29 | Nick |
| 3 | 41 | Dana |

Here, STU ID is a primary key.

(3) Alternate Key

All the keys which are not primary key are called Alternate Key.

It is a candidate key which is currently not the primary key.

Example

| Stu. ID | First Name | Email |
|---|---|---|
| 1. | Tom | ala@gmail.com |
| 2. | Nick | xyz@gmail.com |
| 3. | Dana | mmk@gmail.com |

Stu ID and Email can be a primary key. Stu ID is the primary key, then Email becomes the alternate key.

(4) Candidate Key

A super key with no repeated attributes is called a Candidate key.

The primary key is selected from the candidate key.

Properties of Candidates key -

(i) Must contain unique values.
(ii) Many have multiple attributes.
(iii) Must not contain null values.
(iv) Contain minimum fields to ensure uniqueness.
(v) Uniquely identify each record in table.

Example -

| Stu ID | Name | Email |
|---|---|---|
| 1. | Tom | abc @ gmail.com |
| 2. | Nick | xyz @ gmail.com |
| 3. | Dana | mmk @ gmail.com |

Here, stuID, email are candidate keys which help to uniquely identify the student record.

(5) Foreign Key

A foreign key is a column added to create a relationship with another table.

Foreign keys helps to maintain data integrity and allows navigation between two different instance of an entity.

Every relationship in the model nuds to be supported by a foreign key.

Example -

| Dept Code | Debt Name |
|---|---|
| 01 | Science |
| 02 | English |
| 03 | Maths |

| Teacher ID | Debt Code | F Name |
|---|---|---|
| B002 | 02 | David |
| B017 | 03 | Sara |
| B018 | 02 | Mike |

Here, Debt code acts as a foreign key to the table teacher which helps in providing a relationship b/w teacher & department table.

(6) Compound Key

Compound key has many fields which allows in uniquely recognizing a specific record. It is possible that each column may not be unique by itself within the database. However, when combined with the other column or columns, the combination of composite keys become unique.

Example

| Order No. | Prod ID | PName | Qty |
|---|---|---|---|
| A51 | BQ68 | Mouse | 5 |
| A55 | DK14 | USB | 3 |
| A55 | EF52 | LCD | 10 |
| A46 | DK32 | Laser Pointer | 3 |

Here, Order No. & Prod ID cannot be a primary key. However, a compound key of Order ID and Prod ID could be used as it can uniquely identify each record.

(7) Composite Key

A key which has multiple attributes to uniquely identify rows in a table is called a composite key.

The difference b/w compound and composite key is that any part of the compound key can be a foreign key, but the composite key may or may not be a part of the foreign key.

Difference between primary key & foreign key

| | Primary Key | foreign Key |
|---|---|---|
| 1. | Helps to uniquely identify a record in the table | It is a field in the table that is the primary key of another table. |
| 2. | Never accepts null value. | May accept multiple null values. |
| 3. | Primary key is a clustered index & data in the DBMS table are physically organised in sequence of the clustered index. | A foreign key cannot automatically create an index, clustered or non clustered. you can manually create an Index on the foreign key. |

(4) Only single primary key can exist in a table. Multiple foreign keys may exist in a table.

# 1.8 Design Issues

CollateNotes

The of an entity set and a relationship set are not precise and it is possible to define a set of entities and the relationships among them in different ways.

The basic design issues of an ER database schema are discussed below:

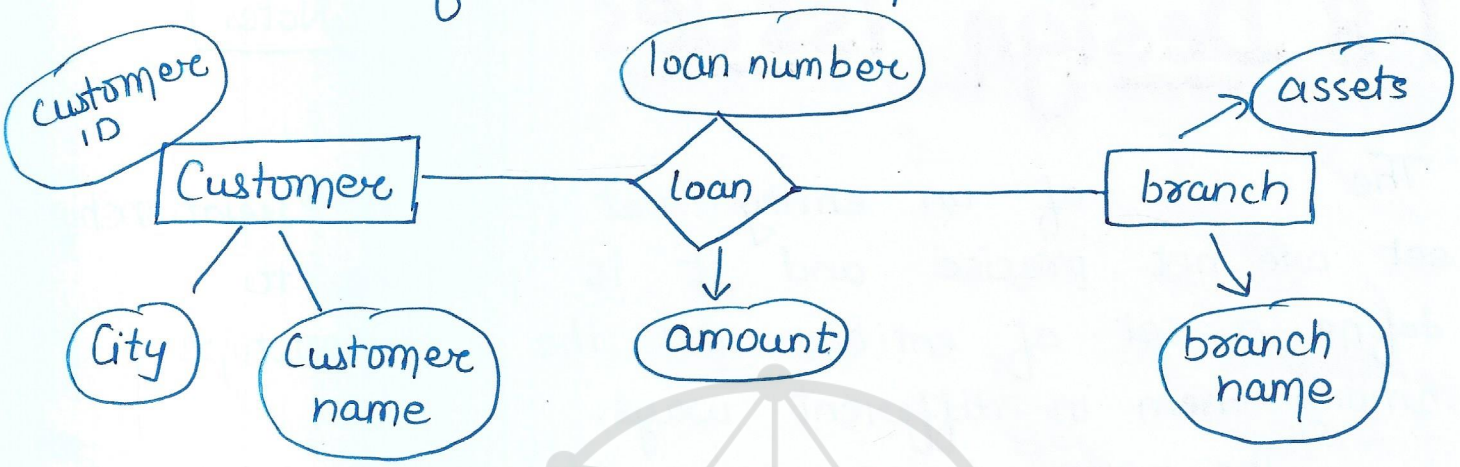
(1) Use of entity sets versus attributes

It is important to determine what constitutes an attribute and an entity set. The distinctions mainly depend on the structure of the real world interprise being modeled and on the semantics associated with the attributes. A common mistakes is to use the primary key of an entity set as an attributes of another entity set, instead of using a relationship.

Example - it is incorrect to model customer-ID as an attributes of loan even if each loan had only one customere instead borrower is the correct way to show connection b/w loans & customers.

Another related mistakes is to designate the

primary key attributes of the related entity set as attributes of the relationship set. Example- loan number and Customer-Id should not appear as attribute of the relationship borrower.




(2) Use of entity sets verses relationship sets

It is not always clear whether an abject is best expressed by an entity set or a relationship set.

In a situation where several customers hold a loan jointly, we must replicate the values for the descriptive attributes loan-number and amount in each such relationship.

Two problems arise due to this replication which are- (i) The data is stored multiple times, wasting storage space.

(ii) Updates potentially leaving the data in an inconsistent state, where the values differs in two relationship for attributes that are supposed

to have the same value. One possible guideline in determining whether to use an entity set or a relationship set is to designate a relationship set to describe an action that occurs between entities.

(3) Binary versus n-ary relationship sets.

Relationships in database are often binary. Some relationships that appear to be nonbinary could actually be better represented by several binary relationships.

An identifying attributes may have to be created for the entity set created to represent the relationship set. This attributes along with the extra relationship sets required, increases the complexity of the designed and overall storage requirements.

A n-ary relationship set shows more clearly that several entity participate in a single relationship There may not be a way to translate constraints on the ternary relationship into constraints on the binary relationships. Example - consider a constraint that says that R is many-to-one from A, B to C that is each pair of entities from A and B is associated with at most one C entity. This constraint cannot be expressed by using cordinabity constraints on the relationship

set $R_A$, $R_B$ and $R_C$.

## Alternative E-R Notations

Some of the alternatives notations that are widely used in ER diagrams are summarised below!

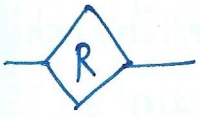

Entity set


weak entity set.

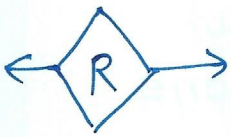

Relationship set


Relationship set for weak entity set


Primary Key




Many-to-Many relationship

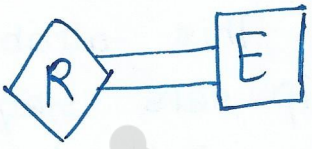



One-to-One relationship


Attribute

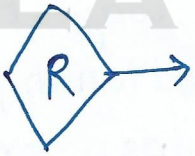

multivalued attribute


Derived Attribute

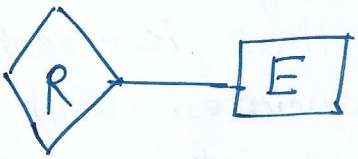



total participation of entity set in relationship.

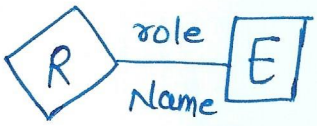

attribute of weak entity set.



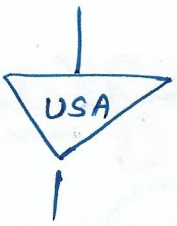

many-to-one relationship



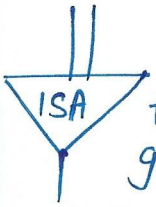

Cardinality limits



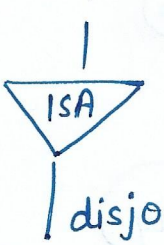

role indicator



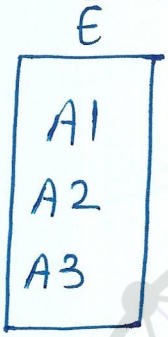

(ISA) specialisation or generalisation)




total generalisation



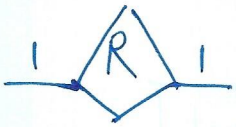

disjoint generalisation




entity set E with attributes A1, A2 and A3 where A1 is primary key.

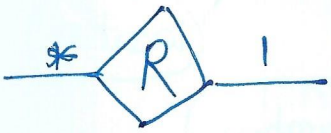


Many-to-Many relationship





One-to-One relationship

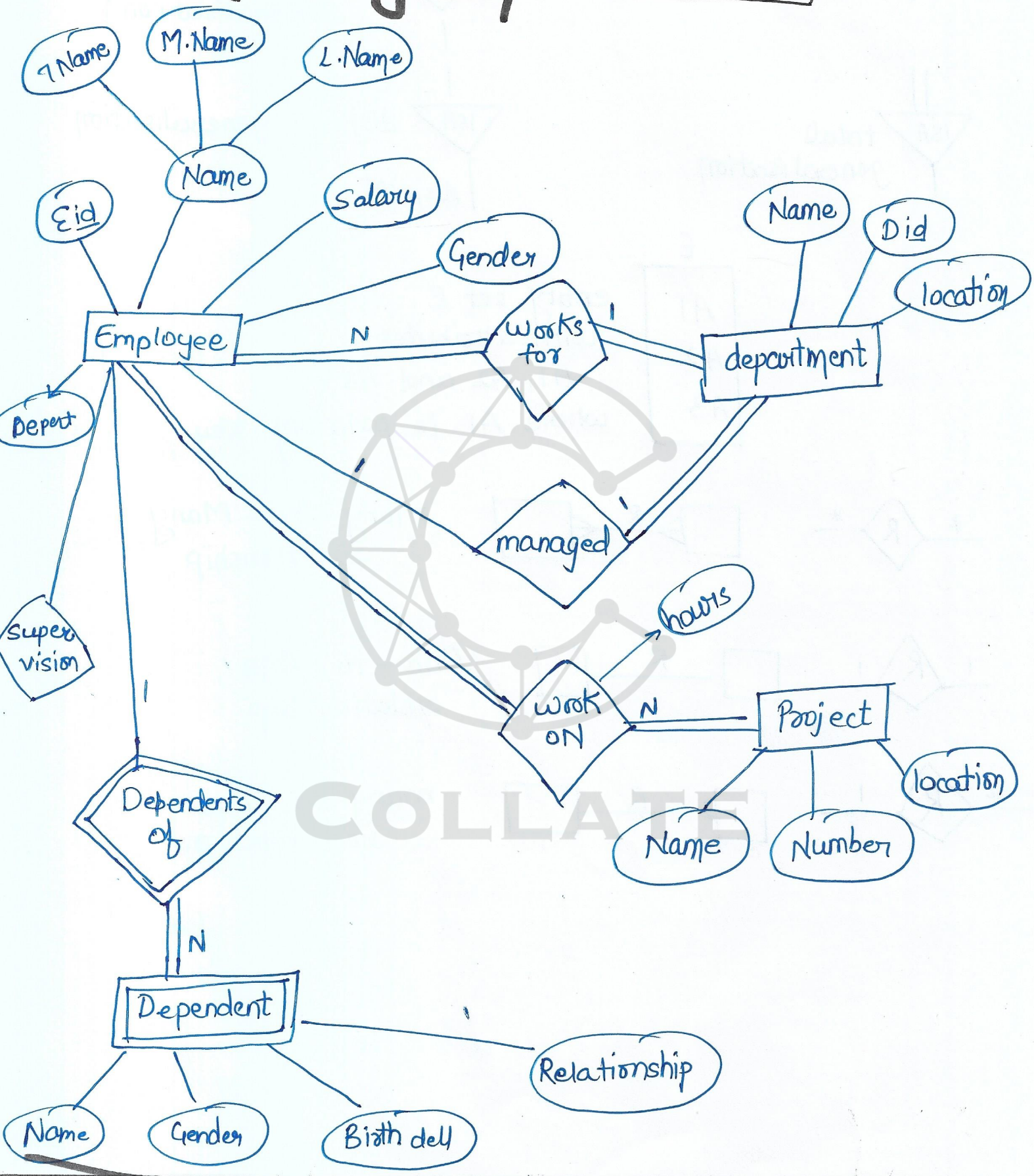




many-to-one relationship.

# 1.9 ER Diagram

CollateNotes



# 1.10 Extended E-R Features

CollateNotes

The basic ER concept can model most database features, some aspects of a database may be more apply expressed by certain extensions to the basic ER model.

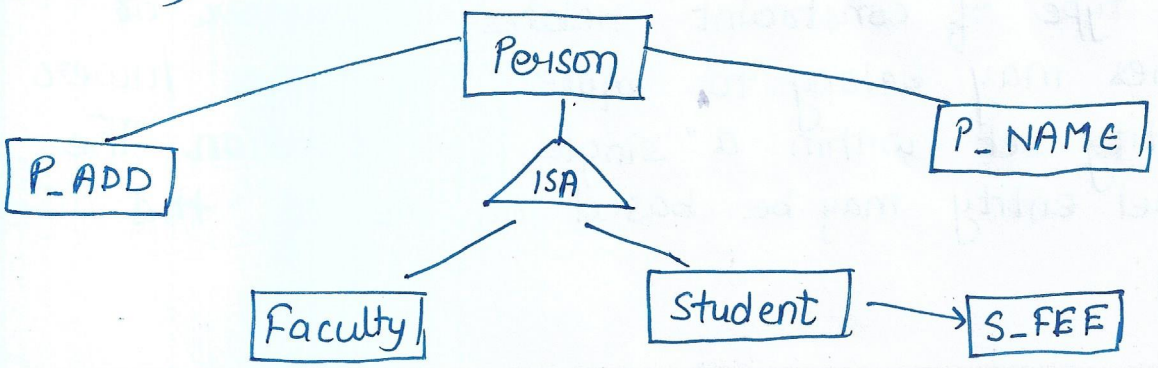
The extended ER features include :- (1) Generalisation (2) Specialisation (3) Aggregation.

## (1) Generalisation

Generalisation is the process of extracting common properties from a set of entities and create a generalized entity from it.

It is a bottom up approach in which two or more entities can be generalised to a higher level entity if they have some attributes in common.

For Example - student and Faculty can be generalized to a higher level centity called Person as shown below. In this case, common attributes like P_NAME, P_ADD become part of higher entity (Person) and specialised attributes like S_FEE become part of specialised entity (student).



It is a contained relationship that exists between a higher level entity set and one or more lowel level entity sets.

For all practical purpose, generalisation is a simple inversion of specialisation.

## ↳ Constraints on Generalisation

Certain constraints can be placed on a particular generalisation. One type of constraint involves determining which entities can be members of a given lower-level entity set. Such membership may be one of the following:

(1) Condition-defined

In condition defined lowel level entity sets, membership is evaluated on the basis of whether or predicate. Since all lowee-level entities are evaluated on the basis of the same attributes, this type of generalisation is said to be attribute defined.

(2) User-defined

User defined lowee-level entity sets are not constrained by a membership condition, rather, the database user assigns entities to a given entity set.

Second type of constraint relates to whether or not entities may belong to more than one lowee level entity set within a single generalisation. The lower level entity may be based on one of the following:

(i) Disjoint - A disjointness constraint requires that an entity belong to no more than one lower level entity set.

(ii) Develapping - In overlapping generalisations, The same entity may belong to more than one lower level entity set within a single generalisation.

Third type, the completeness constraint on a generalisation or specialisation specifies whether or not an entity in the higher level entity set must belong to at least one of the lower level entity sets within the generation / specailisation. This constraint may be no. of the following:

(i) Total generation / specialisation Each higher level entity must belong to a lowel level entity set.

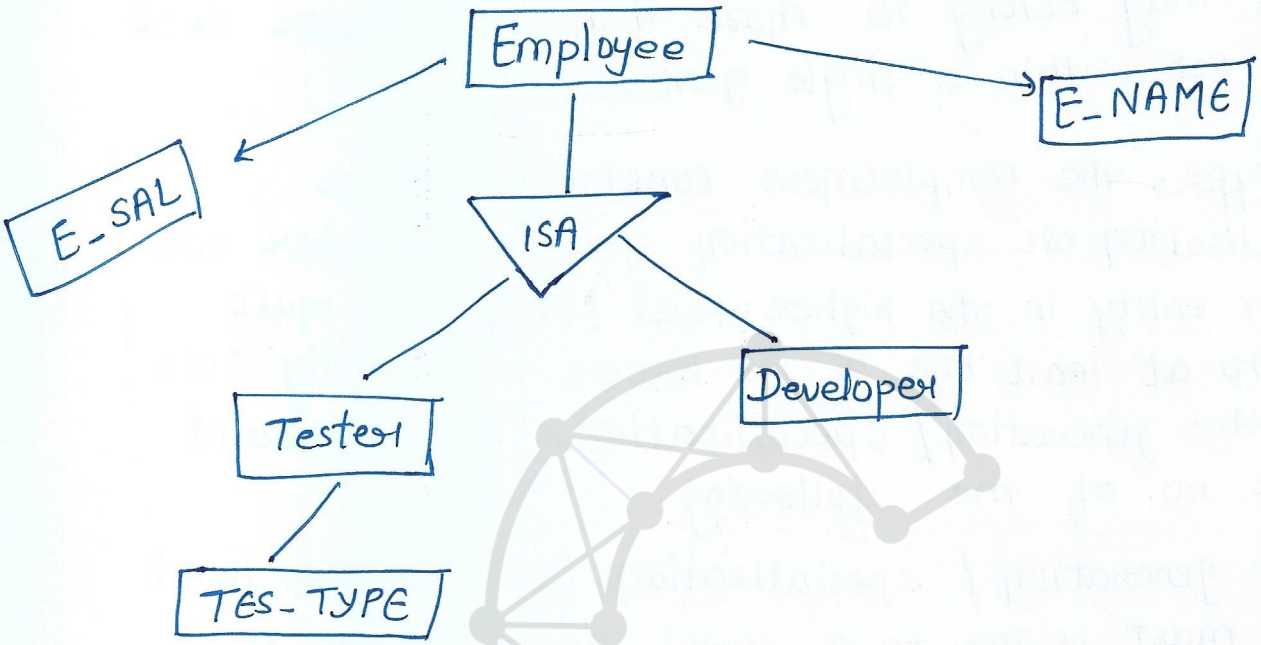
(ii) Partial generalisation / specialisation. Some higher level entities may not belong to any lowere level entity set.

(2) Specialisation

In specialisation an entity is divided into sub untities based on their characterisistics. It is a top down approach where higher level entity is specialized into two or more lowere level entities.

Example - Employee entity is an employee management system can be specialised into developer tester etc. as shown below.

In this case, common attributes like E_NAME, E-SAL etc become part of higher entity (employee) and specialized attributes like TES-TYPE become part of specialized entity (Tester).




In term of an ER diagram, specialisation is dipicted by a triangle component labeled ISA.

The label ISA stands for "is.a" and represents, for Example, that an employee is a tester.

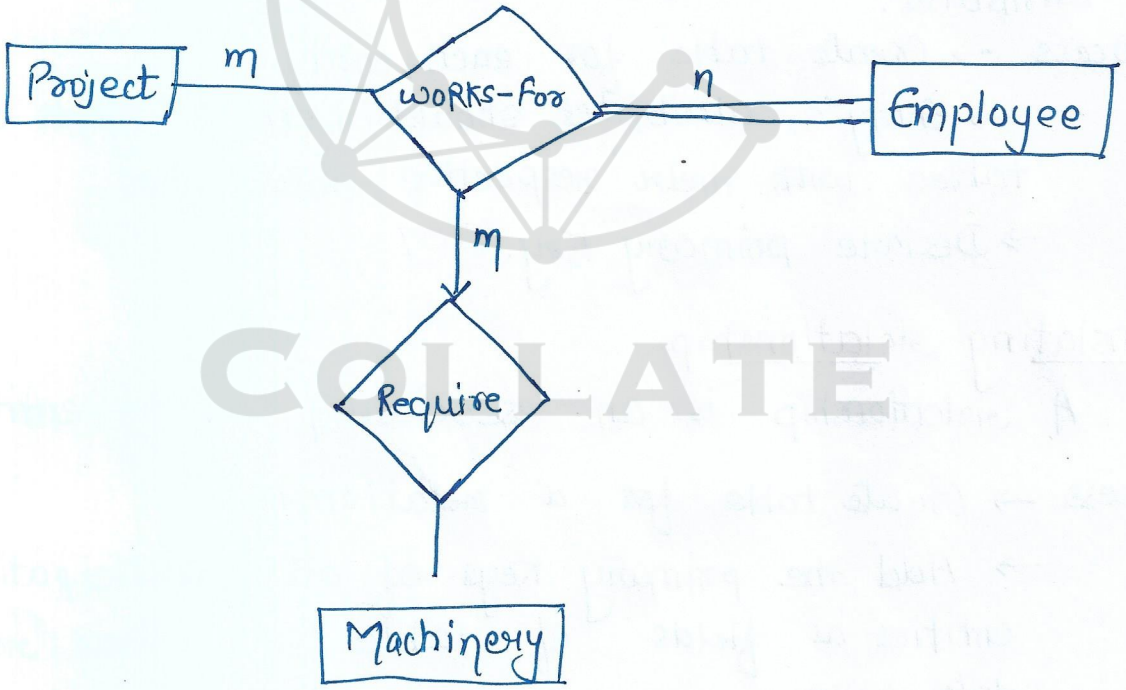
The ISA relationship may also be referred to as a superclass - subclass relationship.

Higher-and lower level entity sets are depicted as regular entity sets - that is, as rectangles containing containing the name of the entity set.

## (3) Aggregation

An ER diagram is not capable of representing relationship between an entity and a relationship which may be required in some scenarios. In those cases, a relationship with its corresponding entities is aggregated into a higher level entity is aggregated into a higher level entity. Example, employee working for a project may require some machinery. So, Require relationship is needed between relationship. WORKS-FOR and entity Machinery.

Using aggregation, works-for relationship with its entities Employee and Project is aggregated into single entity and relationship Require is created between aggregated entity and machinery.




Aggregation is an abstraction through which relationship are treated as higher level entities.

# 1.11 Translating ER Model Into Relational Model

CollateNotes

After designing the ER diagram, we can convert it to Relational model.

There are several processes available to convert ER diagrams into relational schema.

ER diagram mainly consists of –

(1) Entity and its attributes

(2) Relationship among entities.

## Translating an entity

An entity is a real world object with some attributes.

Process → Create table for each entity.

→ Entity's attributes should become fields of tables with their respective data types.

→ Declare primary Key.

## Translating relationship

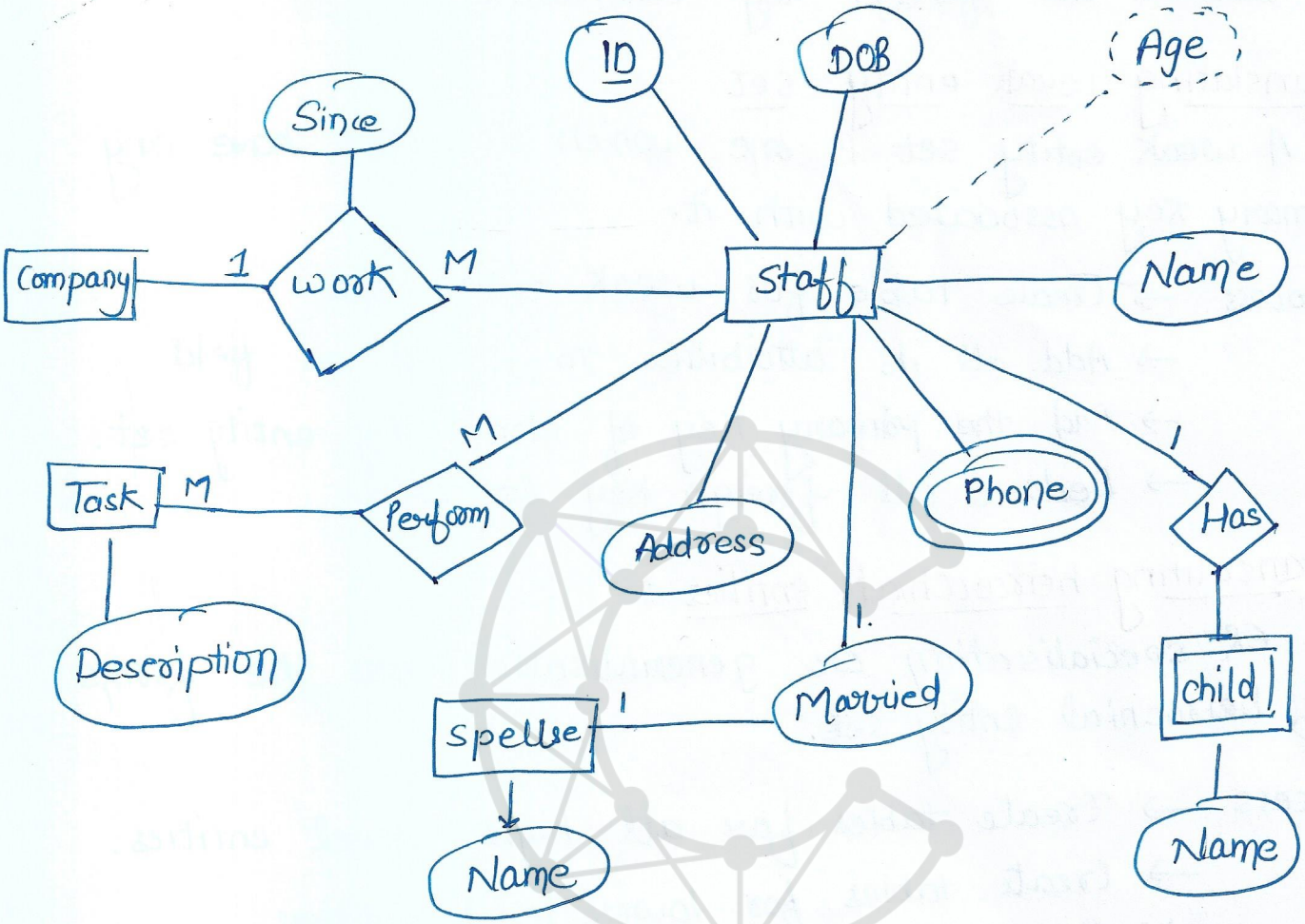
A relationship is an association among entities.

Process → Create table for a relationship.

→ Add the primary keys of all participating entities as fields of table with their respective data types.

→ If relationship has any attributes add each attribute as field of table.

Example – To produce relational schema for the following ER diagram.




The relational schema for the ER diagram is given as –

staff (staff ID, DOB, address, spouse ID) → foreign key

company (company ID, name, address)

child (Child ID, name, staff ID)

staff ID → foreign key

→ Declare a primary key composing all the primary keys of participating entities.
→ Declare all foreign key constraints.

Translating weak entity set

A weak entity set is one which does not have any primary key associated with it.

Process → Create table for weak entity set.
→ Add all its attributes to table as field
→ Add the primary key of identifying entity set.
→ Declare all foreign key constains.

Transaliting heirarchial entities :

ER specialisation or generalisation are the forms of heirachial entity sets.

Process → Create tables for all higher level entities.
→ Create tables for lower level entities
→ Add primary keys of higher level entities in the table of lower level entities.
→ In lower level tables, add all the other attributes of lower level entities
→ Declare primary key of higher level table and the primary key for lower level table.
→ Declare foreign key constains.

spouse (spouse ID, name)
phone (phone ID, phone Number, staff ID
Task (Task ID, description)
work (work ID, company ID, staff ID, since)
perform (perform ID, staff ID, Task ID)

* It is highly recommended that every table start with its primary key attribute conventionally named as Table Name ID.

* The key attribute of the entity is primary key of the table which is usually underlined. It can be composite if required but can never be null.