RDBMS (ACCS-16405)

Subject Notes

Definition

A DBMS is a software system that allows access to the data contained in a database. The objective of a DBMS is to provide a convenient and effective method of defining, storing and retrieving the information stored in the database. The DBMS interfaces with application programs so that data contained in database can be used by multiple application and users.

Some terms:

- Database: A collection of related data.
- Data: Known facts that can be recorded and have an implicit meaning.
- Mini-world: Some part of the real world about which data is stored in a database. For example, student grades, maintenance of a computerized database, employees in an organization, items data etc.
- Database Management System (DBMS): A software package/ system to facilitate the creation, manipulation and storage of data.

Typical DBMS Functionality

- Define a database in terms of data types, structures and constraints
- Construct or Load the Database on a secondary storage medium
- Manipulating the database : querying, generating reports, insertions, deletions & modifications
- Concurrent Processing and Sharing by a set of users and programs – yet, keeping all data valid and consistent

Other features:

- Protection or Security measures to prevent unauthorized access
- "Active" processing to take internal actions on data
- Presentation and Visualization of data

Example of a Database

(with a Conceptual Data Model)

- Mini-world for the example: Part of a UNIVERSITY environments.
- Some mini-world *entities*:
  - STUDENTs
  - COURSEs
  - SECTIONs (of COURSEs)
  - (academic) DEPARTMENTs
  - INSTRUCTORs

Note: The above could be expressed in the ENTITY-RELATIONSHIP data model.

Components of a DBMS

A database is composed (made up) of the following four basic components:-

1. Data
2. Hardware
3. Software
4. Users

1. Data: - A database is a storehouse for data. Data contained in a database should possess the following properties

(a) Consistent & valid (b) Shared

2. Hardware: - Physical parts of a computer system, which help in efficient storage, processing and retrieval of information, are termed as the Hardware.

3. Software: - It is group of programs which help to do all the operations related to a database. Examples are:-

(a) Dbase (I, II, III, III plus, IV) (b) FoxBASE
(c) FoxPro (For DOS and Windows) (d) Oracle
(e) Sybase (UNIX) (f) Ingress
(g) DB2 from IBM (h) SQL Server from Microsoft

4. Users: - A large variety of users is utilizing a database either concurrently or in a batch environment. These can be categorized as follows:-

(a) Naïve (Inexperienced) users: - These users are not aware of the presence of a Database system. They use previously well-defined functions in the form of "canned transactions" against the database. Examples are bank-tellers or reservation clerks who do this activity for an entire shift of operations.

(b) Online users: - These users are aware of the presence of a database system. They have gained a certain level of expertise as far as a DBMS is concerned.

(c) Application Programmers: - These are professional programmers who are responsible for developing application programs or user interfaces, which are utilized by naïve and online users.

(d) Data Entry Operators: - These users are proficient typists and are responsible for adding large volumes of data in the database of organizations.

(e) Database Administrator (DBA):- DBA is a person or a group of persons, who are entrusted with the task of defining, constructing and maintaining a database. They are the users who are most familiar with the database and are responsible for its upkeep and security.

(f) Database Designers: responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs.

(g) Sophisticated : These include business analysts, scientists, engineers, others thoroughly familiar with the system capabilities. Many use tools in the form of software packages that work closely with the stored database.

ARCHITECTURE FOR A DBMS

The generalized structure of a database system is called the ANSI / SPARC model. The ANSI-SPARC Architecture (American National Standards Institute, Standards Planning And Requirements Committee), is an abstract design standard for a database management system (DBMS), first proposed in 1975. The ANSI-SPARC model however never became a formal standard.A large number of commercial systems and research database models fit into this framework. The architecture is divided into three levels

a. External Level
b. Conceptual Level
c. Internal Level

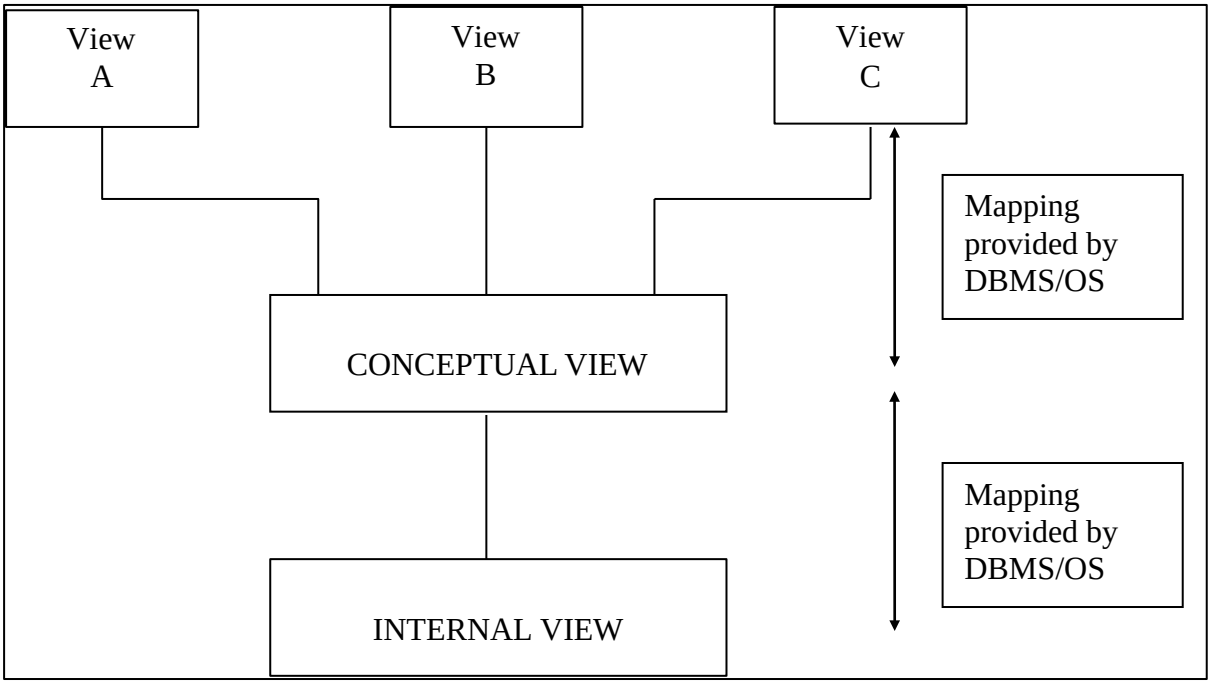


Fig I: - THREE LEVEL ARCHITECTURE OF A DBMS

External level: - The external user view is at the highest level of database abstraction where only those portions of the database of concern to a user are included. Any number of user views may exist for a given global view.

Each external view is described by means of a scheme called an external schema [Plural for scheme]. It consists of the description of logical records and the relationships in the external view. It also contains the method of deriving the objects in the external view from the objects in the conceptual view.

Conceptual or Global View: - At this level of database abstraction, all the database entities [anything of interest to an organization] and the relationships among them are included. One conceptual view represents the entire database. There is only one conceptual schema per database. It also contains the method of deriving the objects in the conceptual view from the objects in the internal view.

Internal View: - This view is at the lowest level of abstraction. It is closest to the physical storage method used. It indicates how the data will be stored and describes the data structures & access

methods to be used by the database. It is expressed by the internal schema, which contains the definition of the stored record, the method of representing data fields, and the access aids used.

Mapping between views: -

Two mappings are required in a database system with three different views:

- ✓ A mapping between the external and conceptual views gives the correspondence among the records and the relationships of the external & conceptual views. The external view is an abstraction of the conceptual view, which in turn is an abstraction of the internal view.

- ✓ A Mapping between the conceptual & the internal levels specifies the method of deriving the conceptual record from the physical database.

DATA INDEPENDENCE

Definition of data independence is "**It is the immunity of applications to change in storage structure and access strategy**". It means you can use programs to modify the structure of the database files and you can select any storage device to store the data.

Data Independence is achieved by use of the three levels of abstraction (**External, Conceptual and Internal**).

Data independence is divided further into two different types:

A). Logical Data Independence (LDI)

B). Physical Data Independence (PDI)

When a schema at a lower level is changed, only the mappings between this schema and higher-level schemas need to be changed in a DBMS that fully supports data independence. The higher-level schemas themselves are *unchanged*. Hence, the application programs need not be changed since they refer to the external schemas.

Three levels of abstraction, along with mappings from external to conceptual and from conceptual to internal, provide two distinct levels of data independence:-

a. Logical Data Independence
b. Physical Data Independence

- ✓ **Logical Data Inde pendence** : The capacity to change the conceptual schema without having to change the external schemas and their application programs.
- ✓ **Physical Data Independence** : The capacity to change the internal schema without having to change the conceptual schema.

Logical Data Independence: - It indicates that the conceptual schema can be changed without affecting the existing external schemas. The change would be absorbed by the mapping between the external and conceptual levels. It also insulates application programs from operations such as combining two records into one or splitting an existing record into two or more records.

Physical Data Independence: - It indicates that the physical storage structures or devices used for storing the data could be changed without necessitating a change in the conceptual view or any of the

external views. The change would be absorbed by the mapping between conceptual and internal levels.

RESPONSIBILITIES OF A DBA

1. DBA is the custodian of the data and controls the database structure.
2. He is responsible for definition and implementation of the internal level.
3. Different mappings are also specified by the DBA.
4. Different permissions regarding the use of database are granted by the DBA.
5. DBA is responsible for defining recovery procedures in case of a failure.
6. DBA is responsible for maintaining the integrity of a database.
7. He continuously interacts with the users to see if there is any problem.
8. DBA monitors the database and responds to any change in requirements.

ADVANTAGES OF USING A DBMS

1. One of the main advantages of using a database is that the organization can exert centralized management and control over the data.
2. Reduction of Redundancies :- Centralized control of data by DBA avoids unnecessary duplication of data and effectively reduces the amount of data storage required.
3. Shared Data :- A database allows the sharing of data under its control by any Number of application programs or users.
4. Integrity of data :- Centralized control ensures that adequate checks are there in the DBMS to provide data integrity. Data integrity means that data contained in a database is both accurate and consistent.
5. Security :- Data is a valuable resource for any organization and can be of confidential nature. Data in a database can be safeguarded against unauthorized usage and access. Proper security checks can be used to ensure that only right people get right access to right data. Usage of magnetic badger, username- password schemes, retina scanners, fingerprint scanners can be employed for proper security of a database.
6. Conflict Resolution :- Since a database is under the direct control of a DBA, the conflicting requirements of various users and applications can be resolved amicably.
7. Data Independence: - One of the features of a DBMS is Data Independence. Both logical & physical data independence can be achieved in a DBMS.
8. Providing backup and recovery services.
9. Providing multiple interfaces to different classes of users.
10. Representing complex relationships among data.
11. Enforcing integrity constraints on the database.
12. Drawing Inferences and Actions using rules

DISADVANTAGES OF A DBMS

1. A significant disadvantage of a DBMS is cost.
2. Hardware has to be upgraded which increases the overheads.
3. Purchasing and developing the software also requires a large amount of money.
4. Efficiency gets reduced as a lot of time is spent in implementing security, integrity and sharing of data.
5. Lack of duplication requires that data should be backed up frequently. This is a time consuming & expensive operation.

Historical Development of Database Technology

- ✓ Early Database Applications: The Hierarchical and Network Models were introduced in mid 1960‟s and dominated during the seventies. A bulk of the worldwide database processing still occurs using these models.
- ✓ Relational Model based Systems: The model that was originally introduced in 1970 was heavily researched and experimented with in IBM and the universities. Relational DBMS Products emerged in the 1980‟s.
- ✓ Object-oriented applications: OODBMSs were introduced in late 1980‟s and early 1990‟s to cater to the need of complex data processing in CAD and other applications. Their use has not taken off much.
- ✓ Data on the Web and E-commerce Applications: Web contains data in HTML (Hypertext markup language) with links among pages. This has given rise to a new set of applications and E-commerce is using new standards like XML (eXtended Markup Language).

Extending Database Capabilities

New functionality is being added to DBMSs in the following areas:

- ✓ Scientific Applications
- ✓ Image Storage and Management
- ✓ Audio and Video data management
- ✓ Data Mining
- ✓ Spatial data management
- ✓ Time Series and Historical Data Management

The above gives rise to new research and development in incorporating new data types, complex data structures, new operations and storage and indexing schemes in database systems.

When not to use a DBMS

Main inhibitors (costs) of using a DBMS:

- ✓ High initial investment and possible need for additional hardware.
- ✓ Overhead for providing generality, security, concurrency control, recovery, and integrity functions.

When a DBMS may be unnecessary:

- ✓ If the database and applications are simple, well defined, and not expected to change.
- ✓ If there are stringent real-time requirements that may not be met because of DBMS overhead.
- ✓ If access to data by multiple users is not required.

When no DBMS may suffice:

- ✓ If the database system is not able to handle the complexity of data because of modeling limitations
- ✓ If the database users need special operations not supported by the DBMS.

DBMS FACILITIES

Two main types of facilities are provided by a DBMS:-

i. The data definition facility or **DDL**

ii. The data manipulation facility or **DML**

iii. The data control facility or **DCL**

Data Definition Language (DDL):- This facility can be used to define the conceptual scheme and also give some details about ho to implement this scheme in the physical devices used to store the data. This definition includes all the entity sets and their associated attributes as well as relationships among them. It also includes any constraints that have to be maintained. The compiled form of these definitions is known as Data Dictionary, directory or System Catalog. The data dictionary contains information on the data stored in the database and is often consulted by the DBMS.

Data Manipulation Language (DML)

The language used to manipulate data in the database is called Data Manipulation Language [DML]. It involves retrieval of data from the database, insertion of new data into the database, and deletion or modification of existing data. A query is used to retrieve the data. It is a statement in the DML that requests the retrieval of data from the database.

The DML provides commands to select and retrieve data from the database. Commands are also provided to insert, update and delete records.

The DML can be procedural which means that user indicates not only what to retrieve but how to go about retrieving it. If the DML is non procedural, the user has to indicate only what is to be retrieved.

Data Control facility(DCL)

It has two commands associated with it

(a) Grant

(b) Revoke

ENTITY - RELATIONSHIP MODEL

The Entity - Relationship Model (ER Model) is a high-level conceptual data model developed by P.P.Chen in 1976 to facilitate database design. Conceptual Modeling is an important phase in designing a successful database for an enterprise.

The Entity-Relationship (E-R) Model is based on a view of a real world that consists of a set of objects called entities and relationships among entity sets which are basically group similar objects.

The E-R model is shown diagrammatically using Entity-Relationship (E-R) diagrams which represent the elements of the conceptual model that show the meanings and the relationships between those elements independent of any particular DBMS and implementation details.

The following, are the features of the E-R Model :

1. The E-R diagram used for representing E-R Model can he easily converted into relations (tables) in a relational model.
2. The E-R Model is used for the purpose of good database design.
3. It provides useful concepts that help the designers to understand the need of end users and implement them while designing DBMS.
4. It is helpful as a problem decomposition tool.
5. It is very simple, nontechnical, free of ambiguities and easy to understand.
6. It provides a standard and logical way of visualizing the data.
7. It gives precise understanding of the nature of data and how it is used by an enterprise.
8. It is a top-down approach to database design.
9. It is inherently an iterative process.

ER MODEL TERMINOLOGY

The following is the list of E-R Model terminology and their corresponding meaning :

NAMES	MEANING
Entity	Objects about which information can be stored. E.g. Specific room, specific part, specific student etc.
Attribute	Properties or characteristics of entities. E.g. name, age, class of an entity student.
Relationships	Associations between two or more entities.
Degree of relationships	Number of entities associated with a relationship.
Cardinality of relationship	How many occurrences of one entity are related to one occurrences of another entity.
Connectivity of relationship	Relationships classification i.e. 1:1, 1:N, N:1 and M:N
Direction of relationship	Line(s) connecting two entities showing the types of relationships.

1. ENTITY: The basic object that the E R Model represents is an entity. It is a "thing" in the real world about which we need to store information and which has an independent existence.
2. ENTITY TYPE AND ENTITY INSTANCE: An entity type is a collection of entities thathave the same attributes. In other words, the entities which are uniquely identifiable but share same properties or attributes form the entity type.
3. ENTITY SET: It is a collection of entities of the same type that share the same properties or attributes. The entity set is usually referred to, using the same name as the entity type.
4. ATTRIBUTES: An entity is atomic and cannot be broken down into smaller pieces but can be further described by some additional characteristics or properties known as attributes.
   So, the attributes are the significant properties or characteristics of an entity that help uniquely identify it. For example: The student entity can be further described by its roll number (Rollno), Name (name) and his class (class) which are the attributes for the entity STUDENT.

Properties of Attributes

a. The attributes of an entity should be unique.
b. The attributes should uniquely identify the entity.
c. The set of permitted values for each attribute is known as its domain from where its values are chosen.
d. An attribute of an entity is represented in ER diagram as an ellipse attached to a relevant entity by a line and labeled with entity name.

TYPES OF ATTRIBUTES

There are different types of attributes

- Simple attribute
- Composite Attribute
- Single-Valued Attribute
- Multi-Valued Attribute
- Derived Attribute

a) SIMPLE ATTRIBUTE: Simple Attributes are also known as atomic attributes which cannot be further subdivided into smaller components. It is composed of a single component and has independent existence. For example - Price, Reg_no, Max_speed of a CAR are simple attributes as they have independent existence. Other examples are Age, Sex of a STUDENT.

b) COMPOSITE ATTRIBUTE: Composite attributes are those attributes that can be divided into sub parts. These subparts represent attributes that have independent existence. The value of composite attribute is obtained by the concatenation of values of its sub parts. Composite attributes helps us to group together related attributes making modelling cleaner. For example - The name attribute can be composed of components like first_name, middle_name and last_name.

c) SINGLE-VALUED ATTRIBUTE: A single value attribute is the one that has only a Single value for a particular entity. For example-Each STUDENT has a single value. Example is AGE attribute.

d) MULTI-VALUED ATTRIBUTE: A multivalued attribute is an attribute that holds multiple values for a particular single entitiy. A multivalued attribute may have lower and upper limits.

e) DERIVED ATTRIBUTE: A derived attribute is the attribute that can be derived from the value o the related attribute. For Example AGE attribute can be derived from DOB attribute.

f) NULL VALUE: A null value is used when an attribute value is missing or not known. Null value also means not applicable. For example: Consider an attribute PAN-NO (Personal account number) for a particular EMPLOYEE is null. It means that it is missing, an employee may not be presently having it but he may have applied for it which may be filled later on.

RELATIONSHIPS

An association between two or more entity type is called a relationship.

A relationship can involve one or more entity types and belongs to a particular relationship type. As relationships are identified, they should be classified in terms of Degree, Cardinality, Connectivity and Direction.

DEGREE OF A RELATIONSHIP

The degree of a relationship is the number of entity types connected to a relationship. In general, if a relationship is of "n-ary" type then its degree is "n'.

Some common types of Relationships are :

1. UNARY RELATIONSHIP: When the association exist within a single entity type it is referred as unary relationship. The degree of unary relationship is I (Unit). Unary relationship are sometimes called the Recursive Relationship. A Recursive relationship occurs when an entity type is related to itself in different roles.
2. BINARY RELATIONSHIP : When the associations exists between two entity types, it is referred as Binary Relationship. The degree of binary relationship is 2.
3. TERNARY RELATIONSHIP - When associations exists among three entity types , it is referred as ternary relationship. It is rarely used in real world.
4. QUARTERNARY RELATIONSHIP - When associations exist among four entity types thenit is referred as quaternary relationship. It is rarely used and is decomposed into one or more angry relationship.

CARDINALTY OF RELATIONSHIP

Cardinality of a relationship quantifies the relationship between entities by measuring how many instances of one entity type are related to a single instance of another.

CONNECTIVITY OF RELATIONSHIP

The term connectivity is used to describe the classification or relationships which can be classified by one to one (1: 1), one to many (I:N) and many to many (M:N). If one to many is reversed it forms many

to one (N: 1) relationship.

- A one to one (1:1) relationship occurs when an instance of entity type (say A) is associated with exactly one instance of another entity type (say B) and vice versa. For example: A project carried out by a student.
- A one to many (1:N) relationship occurs when any one instance of entity type (say A) is associated with any (N) number of instance of another entity type (say B). But an instance of entity in B can be associated with at the most one instance of entity type A. The value of N may vary from (0, 1, N).

  For example : Suppose in a college, each department has 5 Lecturers each. On further requirement, the various departments employees more lecturers. Some departments may employee 1 lecturer, some may employee 6 lecturers, some may not employee lecturer at all. This relationship (employee) which exists between DEPARTMENT and LECTURER is one to many (Fig. 3.11). To each lecturer, there corresponds only one department. Consider another example of MONITOR and WINDOW popped on it. If only one MONITOR is connected to CPU, then multiple windows can be popped on that monitor so it is one to many relationship.
- A many to one (N:1) relationship occur when any (N) number of instances of entity type (say A) is associates with a most one instance of another entity type (say B).
- A many to many (M:N) relationships occurs when an instance of an entity type (say A) is associated with any number (say N) of instances of entity type (say B) and instance of entity type (B) is associated with any number (say M) of instances entity type A.

  The values of M and N may vary 0, 1, ... M, N.

  For example: Consider the relationship AUTHOR writes BOOKS. This is many to many (M:N) relationship. This is because one author can write many books and one book may be written by many authors.

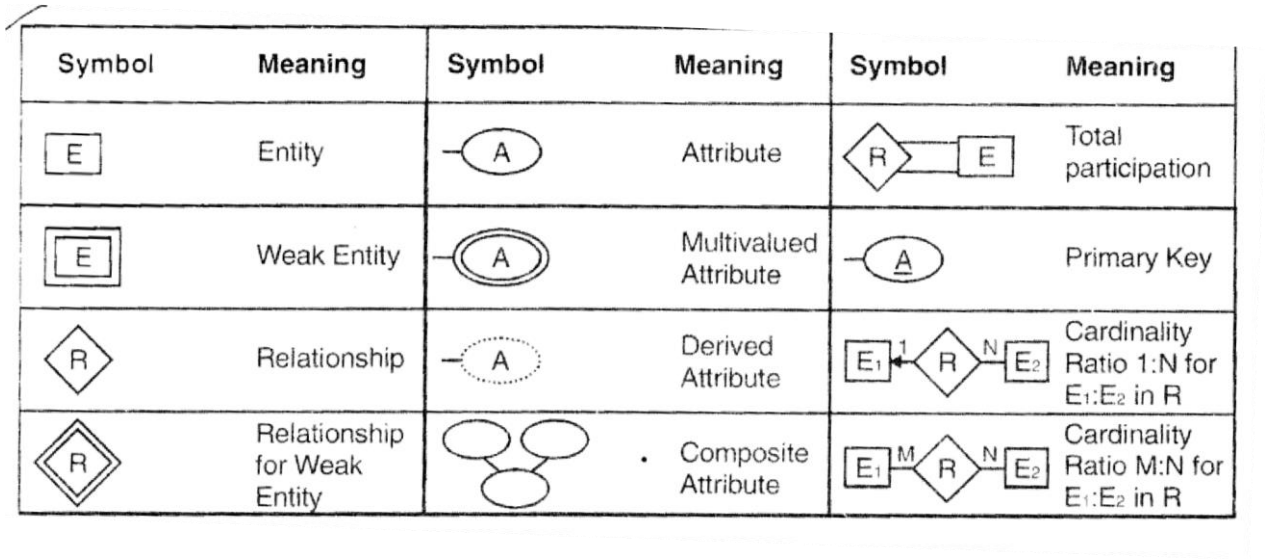
E-R DIAGRAM

The structure of the database employing the E-R Model is shown pictorially using entity relationship (E-R) diagrams. The various entities and the relationships between them are shown with the help of following conventions.

- Entity is an instance of an entity type so it is represented by a rectangle enclosing its name.
- Diamonds represents relationship. The name of the relationships is enclosed in Diamond.
- Lines link attributes to entity and entity to the relationship.
- Ellipses represents attributes
- Double ellipses represent multi valued attributes.
- Dotted ellipses denote derived attribute.
- Ellipses emanating from ellipses show composite attribute.

- Underline name inside Ellipses indicate primary key attributes.
- Weak entity is represented by double rectangle with its name enclosed in it.
- Double diamonds represents relationship for weak Entity.

ER Diagram Symbols



A complete E-R diagram should include:

1. One or more entities (entity sets). Each entity only appears once per diagram.
2. One or more relationships. Each relationship only appears once per diagram.
3. Connectivities for all relationships. But there should be no connections between relationships.
4. Indications for all existing optionally conditions.
5. Cardinalities for all relationships.
6. All composite attributes should be expanded.
7. Use colours to highlight important portion of your diagrams.

WEAK AND STRONG ENTITY SETS

The entity set that doesn't have sufficient attributes to form a primary key and relies on another for its identification is called a weak entity set. An entity that has a primary key is called a strong entity set.

So far we assumed that the attributes associated with an entity contained a key (to uniquely identify the entities) but this is not always the case.

For example: EMPLOYEE's can purchase policies to cover their DEPENDENT's so we need to record set about policies i.e. who is covered and who owns the policy.

EMPLOYEE entity set contains employees with following attributes EMP_Id, Name, B_Date, Address and Salary.

DEPENDENT entity set has the attributes Name (first name of dependent), Sex, Birth date, Relationship.

EMPLOYEE Table

EMP_ID	Name	B_date	Address	Salary

DEPENDENT Table

Name	Sex	Birth_date	Relationship

202	Abhi	08-Aug-78	28-Rani Ka Bagh	40000
303	Atul	15-Jan-82	24-Lawrence Rd.	20000
404	Anil	23-Mar-81	335- Rose Av.	60000
505	Atul	11-Jan-75	41-Model Town	80000

Rama	M	9-Aug-01	Son
Sanju	F	7-Sep-86	Brother
Anshu	M	6-Jan-58	Father
Sanju	F	7-Sep-85	Son

A weak entity set normally has a partial key which is a attribute(s) that can uniquely identify weak entities that are related to same owner entity. In the above example, Name of DEPENDENT is the partial key, if we assume that no two dependents of the same employee ever have the same name, but sex, birth-date, relationship of a dependent can be same for multiple employees so we cannot take them as a partial key. A partial key is represented by a dotted underline under the name of the partial key. The partial key is also known as discriminator of a weak entity as it provides means of distinguishing the entries of the weak entity set which depend upon strong entity set.

The primary key of a weak entity set is created by a combination of discriminator of weak entity set and the primary key of the strong entity set.

The relationship that associates the weak entity set and the owner (strong entity set) is known as an identifying relationship. A weak entity set has a total participation constraint w.r.t its identifying relationship because the weak entity cannot be identified without a strong entity.

Differences between a Strong and Weak Entity Set

WEAK ENTITY SET	STRONG ENTITY SET
1. It does not have sufficient attributes to form a primary key.	1. It always has an attribute(s) to form a primary key.
2. The member of weak entity set is called subordinate entity.	2. The member of strong entity set is called dominant entity set.
3. It contains a Partial Key or discriminator which is represented by a dashed underline.	3. It contains a Primary Key. It is represented by solid underline.
4. It is represented by a double rectangle.	4. It is represented by single rectangle.
5. The Primary Key of weak entity set is a combination of partial key and primary key of the strong entity set.	5. The Primary Key is chosen from one of its attributes which uniquely identifies its member.
6. The line connecting it and the identifying-relationship is double line.	6. The line connecting it and the identifying relationship is single arrow line.
7. Total participation constraint of it with respect to its identifying relationship always exists.	7. Total participation constraint may or may not always exist.

Extended ER (EER) Features

SPECIALIZATION

The specialization is the process of defining one or more subclass entity sets from the superclass entity set based on some unique attribute specific to the subclass entity set. For example : Consider the EMPLOYEE Superclass entity set which is divided into SECRETARY, ENGINFER and MANAGER subclass entity sets. These subclass entity set may include some common attributes which are the part of EMPLOYEE entity set like Name, DOB, Address, EMP_Id etc. entity set. But some attributes may be unique to each subclass.

In terms of EER diagram, the specialization is represented by a triangular component labeled "IS A" and the label „IS A" stands for „is a"

The 'is a' relationship may also be referred to as a superclass/subclass relationship.

[Read Aggregation in supplementary notes]

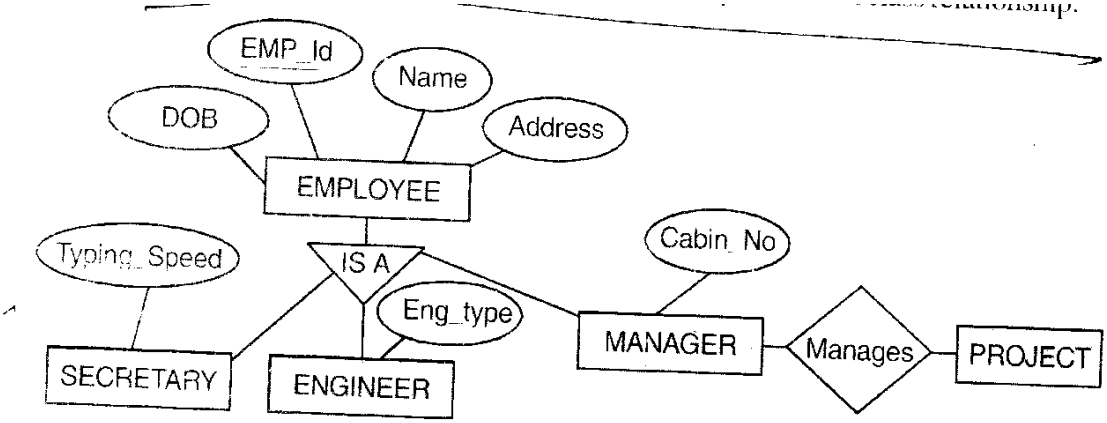


Fig. 3.17 - EER diagram notation for representing specialization and sub classes.

GENERALIZATION

Generalization is the reverse of specialization. There may exist a number of lower level entity sets that share the same attributes and participate in the same relationship sets. So we generalize these low level entity sets into a single higher level entity set based on these common attributes and relationships.

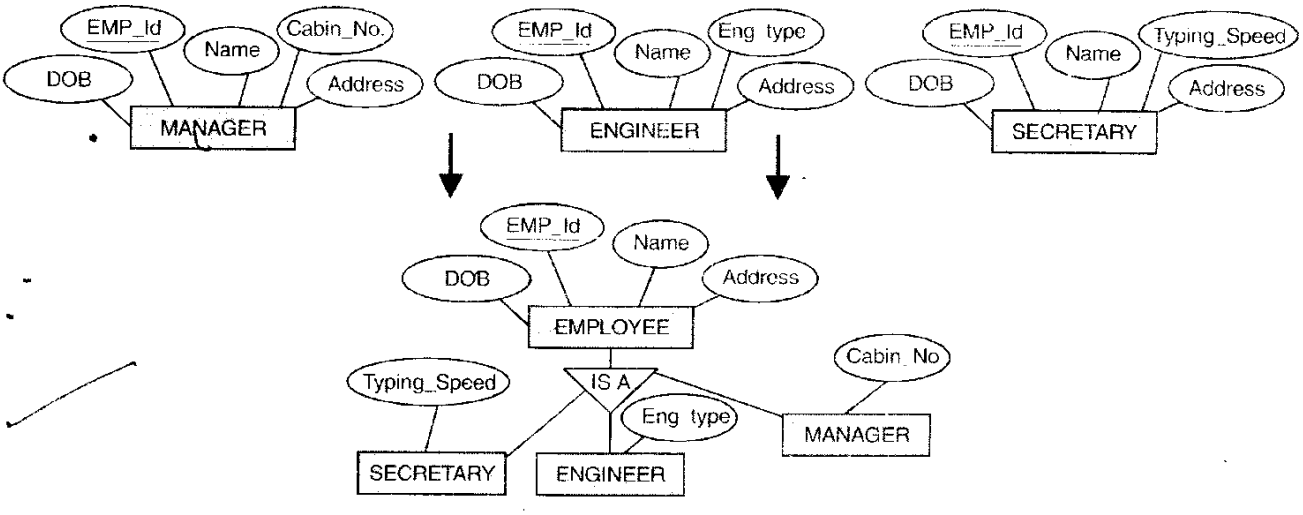


Fig. 3.18 - EER diagram notation for Representing Generalization

DATA MODELS

A model is a way to represent data. There are different models for representation of data. Primary among them are:-

1. Relational Model.
2. Hierarchical Model.
3. Network Model.

Relational Model

- ✓ Relational Data Model has an advantage that it is simple to implement and easy to understand as it uses table format.
- ✓ In this approach, a relation is only constructed by setting the association among the attributes of an entity as well the relationship among different entities.
- ✓ One of the main reasons for introducing this model was to increase the productivity of the application programmers by eliminating the need to change application program, when a change is made to the database.
- ✓ Data structure used in the data model is represented by both entities and relationship between them.

Hierarchical Model

- ✓ It is a tree structure model.
- ✓ It has one root and many branches. We call it **parent-child** relationship. In this a single file has relation with many files and similarly we can say that it is the arrangement of individual data with group data. The representation of this model is expressed by linking different tables.

- ✓ Such type of representation is better for a linkage having many relationships with one. Sometimes it will create ambiguity in designing and defining the association and relationship between parent and children.

Network Model

- ✓ A complex approach of DBMS. In this, we link all the records by using a chain or pointer. It has many to many relationship. Network approach is created when there are more than one relations in the database system.
- ✓ Network approach starts from one point and after connecting similar type of data it returns back to the same record.

To understand these models we consider data about certain **suppliers**, about the **parts** they supply and about the **quantity** of parts supplied by them.

Information about suppliers:-

(i) Name of the supplier. (ii) A supplier number.

(iii) A status code. (iv) City in which supplier lives.

Information about parts:-

(i) Part number (ii) Part Name

(ii) Color of part (iv) Weight of part

(v) City on which part is located

Information about shipments:-

i. Part number which is supplied.

ii. Supplier number who supplied the part.

iii. Quantity in which part is supplied.

This information can be represented using the Relational, Hierarchical and Network Models.

Relational Model

Information is represented in the relational model in shape of tables. There are columns in a table which represent the attributes of an entity about which the table is constructed. The rows of a table are referred to as tuples.

So to represent the information about suppliers in relational model, we need to create three tables. We can call them S-Table, P-Table & SP-Table.

Tables in Relational Model

S-Table

S#	SNAME	STATUS	CITY
S1	Smith	10	London
S2	Clark	20	Rome
S3	Blake	30	Paris
S4	Adam	40	Amsterdam

P-Table

P#	PNAME	WT	COLOR	CITY
P1	Nut	100	Red	London
P2	Bolt	200	Black	London
P3	Screw	300	Black	Paris

SP-Table

S#	P#	QTY
S1	P1	100
S1	P2	250
S2	P1	250
S2	P2	300
S3	P2	200
S1	P3	250

To analyze the relational model, we consider the anomalies of Insert, delete and update.

i. Insert :- Suppose a new supplier comes from area W.
   Solution :- We can insert a new tuple into table S.

ii. Delete :- Suppose supplier S3 stops supplying part P2.
   Solution :- Remove the tuple concerning S3 & P2 from SP table.

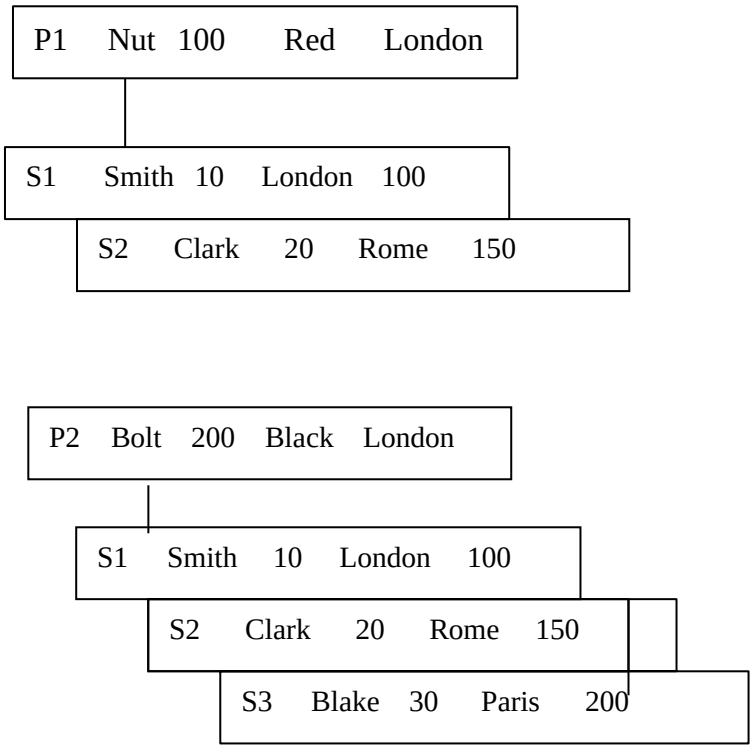
iii. Update :- Suppose supplier S1 moves from London to Paris.
   Solution :- Necessary update can be performed in table S only at one place, so the consistency is maintained.

Thus we observe that the relational model is free from anomalies of insert, delete and update and thus is very good for representation of data.

HIERARCHICAL MODEL

Information is represented in the hierarchical model in parent-child form. In this, some information is considered as superior while some information is considered as subordinate.

We consider the parts as superior and the supplier information as subordinate. So representation can be done as follows:-



Let us consider the anomalies of insert, delete and update for the hierarchical model.

i. Insert :- Suppose a new supplier comes from area W .

Solution :- Since we don‟t know the part supplied by this supplier, so information cannot be inserted. This is because part is a superior and supplier is a subordinate.

ii. Delete :- Suppose supplier S3 stops supplying part P2**.**

Solution **:-** We can remove the supplier S3 information from part P2. But after that we lose any further information about supplier S3. This deletion results in loss of information.

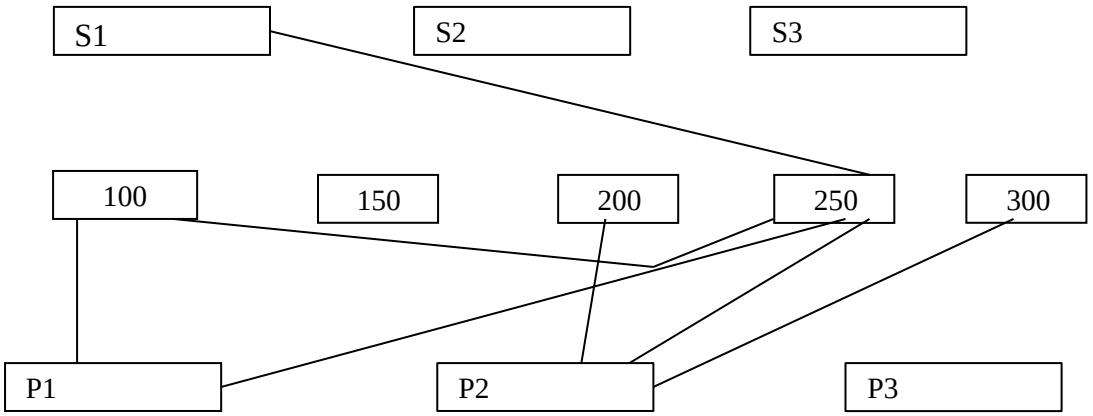
iii. Updation :- Suppose supplier S1 moves from London to Paris.

Solution:- This update can be performed very easily but at multiple places. This may result in inconsistency of database.

So we observe that the hierarchical model suffers from anomalies of delete, insert & update and thus it is not suitable for data representation.

NETWORK DATA MODEL

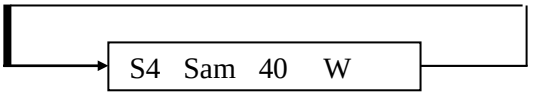
In network data model information is represented by means of pointer or chains. We may construct three records i.e. supplier record, parts record & quantity record. The arrangement is as shown below



Let us consider the three anomalies of insert, delete and update for network model.

(iii) Insert :- Suppose a new supplier comes from area W.

Solution :- A new supplier record can be constructed and when it is known that what part is supplied by him, the appropriate chains can be constructed. Initially its chain can be constructed to itself.



(iv) Update:- Suppose supplier S1 moves from London to Paris.

Solution:- This change is done in the supplier record S1. Therefore there is no chance of inconsistency as the update is performed only at one place.

(v) Deletion :- Suppose supplier S3 stops supplying part P2.

Solution :- The chains connecting records S3, P2 & qty can be removed.

Thus we see that the network data model is free from the anomalies of insert, update & delete.

But it has two big drawbacks i.e.

1. It is expensive to implement the storage of data using records concept & pointers.
2. Complexity is increased because of large number of records.

Thus we observe that out of these models, relational model is the best for representation of data.

SQL (STRUCTURED QUERY LANGUAGE)

ORACLE DATA TABLES

Oracle tables, which consists of rows and columns, are used for storing data. The columns refer to the attributes. Each column in a table has a column name and a data type. A value's data type associates a fixed set of properties with a value. The data types available in Oracle fall under the following categories.

Category	Available data types
Character	CHAR, VARCHAR, VARCHAR2, NCHAR, NVARCHAR2, LONG, RAW, LONGRAW
Number	Number
Date/Time	Date
LOB's (Large Objects)	BFILE, BLOB CLOB, NCLOB

Fig. Data types available in Oracle 9i

CHARACTER DATA TYPES

Character Data Types are used to declare columns containing character (alphanumeric) data, in the database. The following are the different character data types.

- **CHAR(n):** This data type is used for storing fixed length character strings. Once a 'Char' column is defined, all values stored in that column will always have the length as specified in the length of the declaration. If you assign a value that is less than the size of the column then SQL pad the remaining length with spaces. The maximum size of CHAR data type is 2000 characters. If you do not specify a length the default size is 1 character.

  For Example : If data type of name field is mentioned as CHAR (20) and the name information of the particular record completes in 10 characters then the remaining 10 characters space is padded with blank characters. However, if you try to insert a value that is larger than the size of the column (i.e. 20) then it returns an error.
- **VARCHAR(N) :** This data type is used to store variable length character strings, but Oracle recommends to use VARCHAR2(n) data type instead of VARCHAR(n) data type.
- **VARCHAR2(n) :** This data type is used to store variable length character string. When declaring a column of data type VARCHAR2, you must specify its size. The size declaration is not optional as in the case of 'char' data type. The VARCHAR2(n) columns have a maximum size of 4000 bytes. I differs from char data type in that when columns are declared of VARCHAR2(n) data type they will only be as long as the value that is assigned. If the length of the inserting value is less than the specified in declaration (i.e. n) it will not cause value to be padded with spaces. The VARCHAR2(n) data type is used in most of the character data type declaration. The VARCHAR data type is synonym with VARCHAR2, but this may change in future versions of Oracle so you should avoid using VARCHAR data type.

  For Example : If data type of name field is mentioned as VARCHAR2(20) and the name information of a particular record

completes in 10 characters then the remaining 10 characters are not padded with blank characters. The free space of 10 characters will be used for some other purpose. However, if you try to insert a value that is larger than the size of the column (i.e. 20) then it returns an error message.

- **NCHAR(n) :** This data type is used to store fixed length national character set. This character set enable developers and administrators to extend the standard database character set, in order to store data in languages and character sets other than English. When you create a table with NCHAR Column, you define the column length either in characters or in bytes. If the database national character set was a fixed width character set then declare the NCHAR Column size as the number of characters and if it is of variable width then declare the column size in bytes. The maximum column size allowed is 2000 bytes.
- **NVARCHAR2(n) :** This data type is used to store variable-length string in the database's national character set. When you create a table with NVARCHAR2 Column, you define the column length either in characters or in bytes. The key difference between VARCHAR2 and NVARCHAR2 data type is the way in which NVARCHAR2 data is stored in the database. The maximum column size allowed is 4000 bytes.
- **LONG(n) :** This data type is used to store large amount of variable length character strings. It is mostly used to store long text strings. Long columns can be up to 2GB in size. A table cannot contain more than one long column, nor can long columns appear in integrity constraints. Also, long columns cannot appear in where clauses, group by clauses, order by clauses, Of the SQL statements. Due to these restrictions, it is less commonly used.
- **RAW AND LONGRAW data types :** RAW and LONGRAW data types are used to store binary data such as sound, graphics, documents etc. The data stored by these data types is not interpreted by the Oracle. The maximum length of RAW and LONGRAW columns is 2000 bytes and LOB bytes respectively. It strongly recommends that 2GB data types should be used instead of RAW and LONGRAW data types.

NUMBER DATA TYPES

The NUMBER data types are used for storing numeric data such as integers, floating point numbers and real numbers. The NUMBER data type offers the greater flexibility for storing numeric data.

- **NUMBER (p,s) :** This data type is used to store zero, positive and negative fixed and floating point numbers. The NUMBER data type has a precision (p) and a scale(s). The precision (p) is the total number of digits in the number and can range from 1 to 38 digits. The scale(s) describes the number of digits to the right of the decimal point in any given number. The scale can range from -84 to +127. If a value exceeds the precision, Oracle returns an error and if the value exceeds the scale then Oracle rounds it.

DATE/TIME DATATYPE

It stores date and time information.

- **DATE :** This data type is used to store date and time information. It is stored in a specified internal Oracle format that includes not only the month, day and year but also the hours, minutes and seconds. The default date format is 'DD-MON-YY'. For example, '12-JAN-05'.

  If you specify a date value without a time component, the default time is 12:00:00 AM (midnight).If you specify time value without a date, the default date is the first date of the current month. The default date format is specified by

initialization parameter NLS_DATE_FORMAT.

Note : Oracle date data type used 7 bytes for data storage and 1 byte is used for the length data.

Byte7	Byte6	Byte5	Byte4	Byte3	Byte2	Byte1
Seconds	Minutes	Hours	Day	Month	Year	Century

LOB'S (LARGE OBJECTS)

The large objects (LOB's) provide a more flexible storage mechanism for large amounts of binary and character based data such as text, image, video, sound etc. These include BLOB, CLOB, BFILE.

- **BLOB :** The Binary Large Objects (BLOB) data type is used to store binary data such as images, audio files, video etc. It can store up to 4GB of binary data.
- **CLOB :** The Character Large Objects (CLOB) data character type is used for storing large amount of character data. It is used in place of VARCHAR2 column to store text in applications when the size of data is greater than 4000 characters. It can store up to 4GB of data.
- **BFILE :** The binary files (BFILE) data type when declared for a column contains pointers to large binary files (images, videos etc.) stored on the file system of the database server. The length of binary data is limited by the operating system.

WHAT IS A TABLE ?

A table is a database object which is used to store data in relational databases. Each table consists of rows and columns. A column in the database table represents the table's attributes and a row represents a single set of column values in a database table. Each column of the table has a column name and a data type associated with it. The data types which can be used include VARCHAR2, NUMBER, DATE etc. A row of a table is also known as record. Relationship can exist between tables such as relationship between EMP and DEPT, PRODUCTS and ORDERS, EMPLOYEES and their JOB_SKILLS. Within a table foreign keys are used to represent relationships.

Row1
Row2
Row3
Row4

Column1	Column2	Column3
Rollno	**Sname**	**Class**
515	Anurag	BCA-I
526	Kapil	BSc-I
617	Pardeep	BA-I
518	Sandeep	BCA-I

STUDENT Table

The following are the guidelines when designing your tables.

- Use descriptive names for tables, columns and indexes. For Example, a table consisting of information of students of a college should have field names like Rollno, Sname, Class etc. and the table should be named as STUDENT or STU_DATA etc .
- Table should be normalized before creation at least up to 3NF.
- A table must have at least one column.

- Each column of the table should have the proper data type.
- Columns that define nulls are defined last, to conserve storage space.
- Make the proper documentation of the meaning of each table and its columns.
- It is recommended that all the integrity constraints should be determined before creating a table. The constraints are the rules that must be satisfied for the value in the column to be valid such as a primary key.
- Do not use cryptive codes or numbers in your table name.
- Use the same name to describe the same attributes across tables. For Example, the department number fields in employee and department table should have same name.

RULES FOR NAMING A TABLE

The following rules should be followed while defining the name of the table.

- A table name/column name must begin with an alphabetic character.
- A table name/column name cannot exceed 30 characters in length.
- A table name cannot be a SQL reserved words.
- A table name/column name can contain letters A-Z, a-z, 0-9 and the characters underscore (_), hash (#), dollar sign ($).
- Each table owned by an Oracle account must have a unique name.
- Within a single table, a column name should be unique. However, you may use the same column names in different tables.
- Oracle Is not case sensitive when defining table or columns names, unless it is enclosed within double quotes. Double quotes lets you use any characters for a name which may even include reserved words. Although this is not recommended.
- Oracle table may contain up to 1000 columns (In Oracle 8i).
- A table name/column name cannot contain quotation marks.

Valid Table names – Emp, Student, Student_of_college, Dept#1, Stu, Empno (In Scott/Tiger), "emp"
Invalid Table names – 12mp, -emp, stu…..ge (32 characters), Dept&1, 9tu, stu (In Scott/tiger), E'mp, E"mp, select, delete, table

CREATING A TABLE

In order to store and manage data it is necessary to create tables. In Oracle, tables are created using CREATE TABLE command which is one of the important DDL statement. The CREATE TABLE command specifies the name of the table, name of columns in the table as well as the data types and if required constraints associated with each column of the table.

To create a table, in your schema you must have the CREATE TABLE system privilege. To create a table in another user's schema, you must have the CREATE ANY TABLE system privilege.

The syntax for creating table is

CREATE TABLE [SCHEMA].<table-name>

(

<column_name1><data_type> [DEFAULT <expression>] [<column constraints>] [,<column_name2><data_type> [DEFAULT <expression>] [<column constraints>]] [,……]

[<column_namen><data_type> [DEFAULT <expression>] [<column constraints>]]

[,<table constraints>]

);

INTEGRITY CONSTRAINTS IN CREATE TABLE

Oracle uses integrity constraints to prevent invalid data entry into the base tables of the database. One can define integrity constraints to enforce the business rules that one wants to associate with the information in the database. If the integrity constraints are violated due to the execution of any of DML statements (Insert, Update, Select) then Oracle rolls back the statement and returns an error. The different kinds of constraints are:

- Primary Key Constraint
- Foreign Key Constraint
- Check Integrity Constraint
- Unique Key Integrity Constraint
- NOT NULL Integrity Constraint

A constraint can constrain a single column or a group of columns in a table. The more constraints you add to a table definition the less work you have to do in applications to maintain the data integrity. On the other hand, having large number of constraints to a table takes a longer time to make updations to a table. There are two ways to specify constraints

a) Column constraints b) Table constraints

The Column constraints are the constraints that are defined as the part of column definition i.e. they impose rules only on column on which they are defined. These constraints are usually applied when we create or modify the structure of the table. A column constraint cannot be applied if a constraint spans more than one column in a table. So, to make a constraint span more than one column in a table we specify table constraint.

The Table constraints are defined usually at the end of the 'create table' statement which can impose rules on any columns in the table instead only on the column on which it is defined. A table constraints are usually applied to group of one or more columns. At the table level constraint you can apply any constraint except NOT NULL constraints. You can also apply the table level constraints using ALTER TABLE statement.

NOTE : Whenever you define a constraint you have the option of naming the constraint. i.e. it is not essential but recommended to name the constraint. If you use an effective naming scheme for your constraint name, you will be better able to identify the table it acts upon and the type of constraints it represents. For example primary key of instructor table should be named PK_INSTRUCTOR.

The name of constraint is specified during its creation. If you do not specify it then Oracle will assign a system generated name to the constraint of the form SYS_Cn, where n is a six digit number, Eg. SYS_C000545. Since system generated constraints do not provide great deal of information about the constraint so you should always name them.

NOTE : Constraint name cannot be duplicated under a single user.

NOT NULL INTEGRITY CONSTRAINT

A NOT NULL constraint requires a column of a table contain no null values. This constraint can only apply at the column level and not at table level. By default all columns allow null values. Null means that the value is either missing, unknown or in applicable. A null column means that there is no value assigned to a column. You can apply NOT NULL constraint on multiple

columns. The Syntax for NOT NULL constraints is :

<column_name> <data type> [constraint <constraint name>] NOT NULL;

PRIMARY KEY CONSTRAINT

A table's primary key is a column or a set of columns that uniquely identifies each row in the table. The uniqueness property of the Primary Key ensures that there are no duplicate rows in a single table .Although you can create a table without a primary key but it is regarded as a poor practice and should be avoided.

The following properties must hold on for the primary key.

- Each table in a database can have atmost one primary key constraint.
- A column(s) that is a primary key or a part of primary key cannot be NULL, i.e. a value must exist for primary key columns for each row.
- A primary key is a combination of both the uique key constraint as well as NOT NULL constraint.
- If a primary consists of only one column then it is recommended to define primary key with column constraint, otherwise if it consists of more than one column i.e. composite key then it is defined with table constraint.
- The maximum number of columns in the Primary Key is 32 (in Oracle 9i).
- Column that is defined as LONG or LONG RAW cannot be part of a primary key.

The Syntax is

At column level

<column_name> <data type> [CONSTRAINT <constraint name>] PRIMARY KEY;

At table level

[CONSTRAINT <constraint name>] PRIMARY KEY(<column_name1> [,column_name>],…);

FOREIGN KEY CONSTRAINT

Table can be related in Oracle through the use of Foreign key constraints. A foreign key is a single or combination of columns with values based on the primary key values from another table. A foreign key constraint is also known as Referential integrity constraint, specifies that the values of the foreign key corresponds to actual values of primary key in the other table. However you can insert null values in the foreign key columns. The table that includes the foreign is known as the child table or the detail table and the table that is referenced by the child table's foreign key is known as parent table or master table.

If a row is inserted in the child's table with a NOT NULL values in the foreign key column(s), there must be a row in the parent table that has the same value in the referenced column (s). If there is no corresponding row in the parent table an error will occur so, this establishes a parent child relationship between the tables and is the basis for referential integrity in the database. The syntax is :

At column level :

<child_column> <data type>[CONSTRAINT <constraint name>] REFERENCES <parent_table> [(<parent_column>)] [ON DELETE CASCADE];

At table level :

[CONSTRAINT <constraint name>] FOREIGN KEY(child <colname>) references <parent table>(<parent colname>) on delete cascade;

The following points should always be remembered when using foreign key constraint.

- A foreign key can reference to the primary key in the same table.
- A composite foreign key must reference a composite Primary or unique key with the same number of columns and same data types.
- The number of columns composing the foreign key must match the number of primary key columns.
- Each foreign key attribute's data value must match the primary key's data value in the other table that it references or foreign key attributes data value must be wholly null.
- The data type for the foreign key column need not be specified. Whenever you define it at the column level. It guarantees that foreign key column will have the same data type definition as its primary key.
- The maximum number of columns in the foreign key is 32.

NOTE : Sometimes we may want to delete the dependent rows i.e. rows in the child table, when we want to delete rows they depend on i.e. rows in the parent table. For this purpose we use ON DELETE CASCADE clause.

UNIQUE KEY CONSTRAINT

A unique key integrity constraint ensures that each row in a table will have unique values for a given column or set of columns provided that values are not null. i.e. no two rows of a table have a duplicate values in a specified column or set of columns. Unique key Integrity constraint allows you to enter null values for columns without NOT NULL constraints.

The syntax is

At column level:

<column_name> <data type> [CONSTRAINT <constraint name>] UNIQUE;

At Table level

[CONSTRAINT<constraint name>] UNIQUE (<column_name>);

The following points should always be remembered when using UNIQUE key constraint.

- Whereas primary key and foreign key constraints enforce data and referential integrity, the unique key integrity constraint enforces a unique set of values on a column.
- Column with unique key constraint can contain null values. Since null values don't equal another null value, so rows with null values in the unique key constraint columns are said to be unique.
- Unique constraint can take up to 32 columns.
- Unique key constraint is not a substitute for a primary key constraint.

Primary Key Constraint v/s Unique Key Constraint

- A table can have only one primary key, but it can have many unique constraints.
- When a primary key is defined, columns that compose the primary key automatically become mandatory but when a unique key constraint is defined, the columns that compose unique key constraint are not automatically mandatory. To make it mandatory you must also specify that the column is NOT NULL.

CHECK CONSTRAINT

The Check constraint allows you to define a condition that a value entered into a table has to satisfy before it can be accepted. It is also known as business rule constraint. With the check constraint, you can specify an expression that must be true for every row in

a table which is being inserted or updated. The check constraint is a boolean condition that is either true or false. If the condition evaluates to true, the column value is accepted by the Oracle otherwise oracle will return an error.

The Syntax is :-

At column level :

<column_name> <data type> [CONSTRAINT<constraint NAME>] CHECK (<CONDITION>);

At table level :

[CONSTRAINT <constraint name>] CHECK(<condition>);

The following points should be kept in mind when using CHECK Integrity constraints. One of he limitations of the column check constraint is that it cannot reference other columns in the same table. However, you can use a table level check constraint to reference any column in a table.

- A single column can have more than one check constraint.
- You can define any number of check constraints on a column.
- A check constraint can reference pseudo columns such as rownum, NEXTVAL etc. and SQL functions such as sysdate, UID, USER etc.
- It cannot contain subqueries and sequences.

MODIFYING THE TABLE STRUCTURE USING ALTER TABLE

After the tables are created and begin to be used, requirements are likely to occur which requires modifications in the structure of the table such as adding new columns to the table, modifying an existing column's definition etc. So these modifications can be performed using the ALTER table statement. This statement changes the structure of the table and not its contents. There are many types of modifications that can be made to the structure of the tables using ALTER TABLE statement such as

- Add or drop one or more columns to / from existing table.
- Increase or decrease the width of the existing column.
- Change an existing column from mandatory to optional or vice versa.
- To enable or disable integrity constraints.
- Add/ Modify / Delete integrity constraints associated to / from the table.
- Specify a default value for existing column.

NOTE : To ALTER a table, the table must be contained in your schema or you must have either the ALTER object privilege for the table or ALTER any table system privilege.

ADDING NEW COLUMNS / CONSTRAINTS IN EXISTING TABLE

The syntax is

ALTER TABLE <tablename> ADD (column_specification | constraint_specification)

The following points should be kept in mind when adding new columns / constraints into the existing tables.

- You can add one or more than one columns at the same time.
- There is no need of parentheses; if you are adding only a single column or constraint.

- Before adding a primary key make sure that no duplicate values exist for a column (s) on which primary key is applied.
- Before adding a foreign key, make sure that primary key constraint must exist in the referring table.
- You may add a column at any time. If NOTNULL is not specified. However, you can add a new column with NOT NULL if the table is empty.
- If a table contains records and you add a new column (s) o this table, then the initial value of each row for the new column is NULL.

MODIFYING COLUMNS/ CONSTRAINTS IN EXISTING TABLE

The ALTER TABLE statement also offers the ability to modify columns / constraints in the existing table without impacting any other data in the table. Using ALTER TABLE statement, we can increase or decrease the column widths changing a column from mandatory to optional (NOT NULL to NULL) and vice versa.

The syntax is

ALTER TABLE

<tablename> **MODIFY**

(column_specification|constraint_specification)

Oracle doesn't allow you to decrease the columns width if the column has values , even if all the values are of valid length . So to reduce the width of the column the values must be set to NULL.

DROPPING CONSTRAINTS / COLUMNS :

You cannot only modify columns that exit in your tables but you can also drop them entirely. You can also remove the constraints from table using the ALTER TABLE statement. When a constraint is dropped, any associated index with that index is also dropped.

The syntax for dropping a column is:

ALTER TABLE <tablename> DROP COLUMN <column name>;

This syntax is valid if you want to drop one column at a time. The following points should be kept in mind when dropping the column / constraints:

- You cannot drop all the columns in a table.
- You cannot drop a columns from tables whose owner is 'SYS'.
- You cannot drop a parent Key column (primary key) unless you drop the foreign Keys that references it. To do this CASCADE keyword is used.

You Can enable or disable the key constraints in situations like when loading large amount of data into a table , performing batch operations that make massive change to a table , migrating the organization's legacy data.

- Instead of dropping a column, you can make the column unused and drop it later on.

DROP TABLE

Sometimes it is necessary to remove table from database completely or you may be dropping table that was created for a particular task and are no longer needed .

The syntax for dropping a table is

DROP TABLE <TABLENAME> [CASCADE CONSTRAINTS];

The cascade constraint is an optional parameter which is used tables which have foreign keys that reference the table being dropped . if cascade constraint is not specified and user attempts to drop a table that has records in a child table , then an error will occur. So by using cascade constraints , all child table foreign keys are dropped. Now let us delete the INSTRUCTOR table using the DROP TABLE statement.

SQL > DROP TABLE INSTRUCTOR ;

RENAMING TABLES

Oracle provides the facility to change the name of the table by using a ALTER TABLE statement. When you rename the table Oracle automatically updates foreign keys in the data dictionary but it doesn't update a stored code modules, stored queries or reports client applications. So care must be taken when renaming the tables. The Syntax is

ALTER TABLE <Old_tablename>

RENAME TO <new_tablename>;

OR

RENAME <old_tablename> to <new_tablename>;

DATA DICTIONARY

Data Dictionary is a repository of description of data in the database. It contains information about.

- Data – Names of the tables, names of attributes of each table, length of attributes etc.
- Relationship between database transactions and data items referenced by them which is useful in determining which transactions are effected when certain data definitions are changed.
- Constraints on the data i.e. range of values permitted.
- Detailed information on physical database design such as storage structure, access path etc.
- Description of database users, their responsibilities and access rights.

Every Oracle's database has a data dictionary, it is a catalog of your entire Oracle database objects. As you create users, tables, constraints and other database objects. Oracle automatically maintains a catalog of items stored in the database.

Some of the examples are

- **USER_CONSTRAINTS :** This view used to see the constraints associated with a table.
- **USER_TABLES :** This view is used to show information about the tables owned by the user.
- **ALL_TABLES :** This view is used to show information that you own and also the tables that you have granted privileges on.
- **ALL_USERS :** This view is used to show all the usernames, User_ID and their date of creation.

DML STATEMENTS

The DML statements are categorized into four basic statements.

- INSERT : Used to add rows (records) to a table etc.
- UPDATE : Used to modify existing rows (records) in table/view's base table etc.
- DELETE : Used to remove rows from a table etc.
- SELECT : Used to retrieve rows (records) from a table etc. >

Now we shall study the following DML statements.

INSERT STATEMENT

The INSERT statement is used to add rows to a table. To insert rows into a table, the table must be in your own schema or you must have insert privilege on the table. You can insert literal values (26.5,'Anshu'), expressions containing operators and functions, null values etc.

The Syntax is

INSERT INTO <table_name> [(<columnnme1>[,columnname2>] …..[,<columnnameN>])] VALUES [(<columnnme1>[,columnname2>]…..[,<columnnameN>]);

You can insert rows into the table without specifying the column names because they are optional. Although it is always recommended that the column names must be specified when inserting rows into a table, because

- It is possible that the number of columns might increase or decrease which results in failure of INSERT statement.
- Wrong values for the column names may be entered even if INSERT statement is successful.

The following points should be considered while inserting the data into the tables.

- If you do not specify the column name when inserting a row, the Oracle uses all the columns by default. In addition, the column order that Oracle uses is the order in which the columns were specified when the table is created which you can see using SQL*PLUS Describe command.
- The number of columns in the list of column names must match the number of values that appear in parentheses after the keyword 'VALUES'.
- The data types for a column and its corresponding value must match. For example; If the data type of a column is Number then you cannot enter 'Abc' i.e. string values into it.
- The newly inserted row goes into a table at an arbitrary location i.e. the table has no implied ordering.
- The character data will be enclosed within single quotes.
- Whenever you insert values for a Date data type column, then it must be specified in single quotes('). Oracle automatically converts the character field to date data type field.

UPDATE STATEMENT

Quite often it is required to make changes or modifications in the records of the table, so in order to make these changes, the update statement is used. With this statement the user can modify the existing data stored in the table . It can update zero or more rows in table. To update rows in a table, it must be in your own schema or you must have update privilege on the table.

The syntax is:

UPDATE <tablename>

SET <columnname1> =<expression>[,<columnname2>=<expression>] ……[,<columnnameN>=<expression>]

[WHERE condition];

DELETE STATEMENT

The delete statement is used to remove the row (s) from table. An expression is used in the WHERE clause to filter the records that are actually being removed . You can remove zero or more rows from a table. If you don't provide such an expression then DELETE statement will remove all the rows from the table. You can use one or multiple conditions in WHERE clause. You don't need to know the physical ordering of the rows in a table to perform a DELETE. Oracle database engine determines the internal location of the rows. To delete rows from the table , if must be in your schema or you must have delete privilege on the table.

The Syntax is:

DELETE [FROM] < tablename >

[WHERE CONDITION];

TRUNCATE TABLE STATEMENT

The truncate table statement also deletes all the rows from the given table but it in much faster than the DELETE statement. It is not a DML statement. Once you execute a TRUNCATE TABLE statement you cannot perform the Rollback to recover the lost rows .

The syntax is:

TRUNCATE TABLE <tablename>

TCL STATEMENTS:

A transaction is a series of SQL statements executed by a user that are treated as one logical unit of work. In Oracle either an entire transaction succeeds or an entire transaction fails. The change made to the database by the user are not visible to other users until they are made permanent to the database. In oracle you cannot begin a transaction explicitly. i.e. there is no begin transaction statement. As transaction implicitly begin with the first statement that modifies data. The commit or Rollback statements marks the end of the transaction.

Since transactions are atomic i.e. either every statement succeeds or none of the statement succeeds there are number of transaction control statements available that allow us to control this behavior. Some of the most commonly used TCL statements used are:

1. COMMIT

In oracle changes you make to your data are not permanent until you tell Oracle to make them so. This is accomplished using a SQL statement known as commit. A commit ends your transaction and makes the changes you have made permanent. If you exit the SQL* PLUS environment . It will automatically commit your work.

The syntax is:

SQL>COMMIT;

The following points should be remembered while using Commit statement

- Whenever you execute a DDL statement (such as CREATE TABLE etc.) an implicit commit statement is generated before and after it is executed, even if the DDL fails. This is not in the case of DML statements.
- SQL*PLUS also provides facility to automatically commit you work without your explicitly telling it to do so using SET AUTOCOMMIT ON on the SQL prompt.
- The commit statement that one user issues has no effect on another users database changes.

2. ROLLBACK

The ROLLBACK statement is used to undo all the changes that the user has made to the database since the last COMMIT statement was issued or since the database session begins whichever is later. Oracle maintains all the data necessary to modify the database so it looks exactly like it did before you begin your transaction . Rollback also serve to end the current transaction and begin a new one.

If you have completed a series of Insert, Update or delete statements but have not yet explicitly or implicitly committed them and you uncommitted work.

The syntax is:

SQL> Rollback;

3. SAVEPOINT

A save point allows you to create a marked point within a transaction. You may rollback your transaction to that marked point without rolling back any of the work that preceded it. SAVEPOINT's are useful transaction feature since they allow you to divide a single large transaction into a series of smaller parts. While all the statements executed are still part of the larger transaction, you can group certain statements together and roll them back as if they were a single statement. Any other prior statements in the transaction are unaffected. You may have multiple SAVEPOINTS in a single transaction.

The Syntax is:

SQL> SAVEPOINT <name>;

SELECT STATEMENT

The SELECT DML statement is used to retrieve rows from the table. This statement is a tremendously powerful tool.

SELECT<select_list> FROM<table_list>

[WHERE<condition>] [GROUP BY <column1>,<column2>, ……. ,<column>]

[HAVING<condition>] [ORDER BY <expression>];

SQL BUILT-IN FUNCTIONS

Oracle provides an extensive set of built-in functions and operators for the conversion and manipulation of strings, dates and numbers. Functions are similar to operators in that they manipulate data items and return a result. But they differ from operators in the format in which they appear with their arguments. The number of arguments that we specify in the function may vary from zero, one, two or more arguments. The true power of these functions is realized by nesting these functions within each other. Oracle functions are divided into two categories :

- Single-row or Scalar functions
- Group or Aggregate functions

SINGLE-ROW FUNCTIONS

The single-row functions act on each row of the table and return a single value for each row selected. It is also known as scalar functions. The single row functions can appear in the <select_list> if the SELECT statement does not contain a GROUP BY clause, WHERE clause etc.

The single-row functions are categorized into following categories :

- Number functions
- Character functions
- Date functions
- Conversion functions
- Miscellaneous functions

NUMBER FUNCTIONS

Oracle has a variety of functions to process and manipulate number values in the database. These functions are also known as Arithmetic functions, as these functions accept numerical input and return numerical values.

The number functions are given below :

1. ABS (n) : The ABS function is used to get the absolute value of the parameter n passed to the function.

2. CEIL (n) : This function returns the next smallest integer value greater than or equal to the parameter n passed in.

3. EXP (n) : This function returns e raised to the nth power (e=2.7182…)

4. FLOOR (n) : This function returns the largest integer value less than or equal to the parameter n.

5. LOG (m,n) : This function returns the logarithm, base m, of n. The base 'm' can be any positive number other than 0 to 1 and n can be any positive number.

6. MOD (m,n): This function returns remainder of m divided by n. Returns m if n=0.

7. POWER (m,n) : This function returns 'm' raised to the 'nth' power. The base 'm' an the exponent 'n' can be any numbers, but if 'm' is negative, 'n' must be integer.

8. ROUND (m,n) : This function rounds a number 'm' up either to the left or right of the decimal point depending upon the sign of 'n'. Here, 'n' must be integer.

- If 'n' is negative then 'm' is rounded up to that many digits o the left of the decimal point.
- If 'n' is positive, then 'm' is rounded up to that many digits to the right of the decimal point.
- If 'n' is omitted, then 'm' is rounded to 0 places.

9. SIN (n) : This function returns the sine of n where n is expressed in radians.

10. SQRT(n): This function returns square root of n, where n is greater than equal to 0.

11. TRUNC (m,n) : This function drops all digits of number m either to the left or right of the decimal point depending on sign of n, where n must be an integer.

- If n is negative, then m is truncated to that many digits left of the decimal point.
- If n is positive, then m is truncated to that many digits right of the decimal point.
- If n is omitted, then m is truncated to 0 decimal places.

CHARACTER FUNCTIONS

Oracle provides a variety of functions for working with string functions in SQL. These functions accept literal string of characters ('abcd', 'ravi' etc) or name of the character column. The literal strings must be specified in single quotes and column names must appear without quotes.

The character functions are given below:

1. ASCII (String) : It returns the decimal representation in the database character set of the first character of the string.

2. CHR (N) : It returns the character having the binary equivalent to 'n' in either the database character set or national character set.

3. CONCAT (m,n) : It CONCATENATE 'm' with 'n' where 'm' or 'n' can beeither column names or literals. It is equivalent to concatenation operation (||)

4. INITCAP (N) : It changes the first letter of a word of series into uppercase, and rest of the letters into lowercase.

5.INSTR : It is used to determine the numeric position of a particular search string within a character string or iterating through a document looking for multiple instances of a character string. The Syntax is

INSTR (char1, char2 [, start [, occurrence]])

6. LENGTH (N) : It returns the number of characters in the string/Column name n.

7. LOWER (string) : This function converts every letter in a string to lower case.

8.LPAD : It makes a string of a certain length by adding certain set of characters to the left of the string. The Syntax is

LPAD (string, n [, pad_char])

9.LTRIM : It trims leading spaces or a designated character on the left side of the string. The Syntax is

LTRIM (string [,char_trim])

10. RPAD : It is just a reverse of LPAD i.e. it makes a string of certain length by adding certain set of characters to the right of the string.

11. RTRIM : It is just a reverse of LTRIM i.e. it trims leading spaces or a designated character on the right side of the string.

12. REPLACE : This function is used to replace character or group of characters in a string with zero or more characters. Its Syntax is

REPLACE (string, exist_str [, rep_str])

13. SUBSTR : It is used to extract a piece of character string. The Syntax is

SUBSTR (string, m [,n])

14. TRIM : This function removes leading and / or trailing characters from beginning and /or end of the character string .

16. UPPER(string): This function is used to convert all the characters in a string to upper case.

DATE FUNCTIONS

In order to process and manipulate dates, Oracle provides a number of functions that operate on various date related data types in oracle. The default date format in Oracle is DD-MON-YY HH:MI:SS

The date (and time) functions available in oracle are

1. SYSDATE : this function returns the current date and time in a date value based on the database time zone.

2. ADD_MONTHS: This function adds or subtract a number of months to/from a date value.

3. LAST_DAY(date): this function returns the date of the last day of the month containing the date parameters.

4.TRUNC: This function when used with dates return a date value truncated to the value optionally specified in the format parameter. The Syntax is:

TRUNC(date[,format])

This function truncate the date value according to the format specified. Without a format, date is truncated to 12 AM (midnight) in the morning.

Some of the formats are :

- MON , MM , MONTH – it truncates to the first date of the current month for any date and time up to 11:59:59PM
- DD , DDD – it truncates to 12 AM of current date up to 11:59:59PM.
- D- it truncates the date back to a Sunday.

5. ROUND: This functions rounds off a date/ time value formatted to the next highest part of date. The Syntax is:

ROUND(date [,FORMAT])

If the format specified in 'MON', then if date<=15, then first day / date given month is displayed otherwise if day 15 then rounds to first day of next month .

6. MONTHS_BETWEEN: This function determines the number of months between the two dates. The value returned is a real number indicating the whole months and a fraction of a month between two dates. If the first date is earlier in time than the second date , value returned is a negative number .

7. NEXT_DAY(date,day_week): it returns the date value of the next named day of the week after the given date .

CONVERSION FUNCTIONS

The conversion functions convert a value from one data type to another. These functions just return the converted value and does not make any changes in the database. The data type conversion can be done automatically or implicitly done by the oracle or explicitly by the user.

Implicit Data Type Conversion

The Oracle automatically converts one data type to another when used in the assignments.

- To have oracle automatically convert one data type to another the source data type must already look like the target date type it is being converted to . the following guidelines describe the automatic conversion of data from one date type to another based on the function that will use the data.
- Any number or date can be converted to a character string.
- A char or varchar2 will be converted to a number if it contains only numbers, a decimal point or minus sign on the left.
- A char or varchar2 will be converted to date only if it is in the format DD-MON-YY.
- A date will not be converted to a number and vice – versa.

Explicit Data Type Conversion Functions

The following are the list of conversion functions which are done explicitly done by the user.

1. TO_CHAR (date conversion): This date based function is used for transforming a DATE value into a VARCHAR2 value, optionally using a specified date format. . The Syntax is:
   TO_CHAR(date value[,date_format])
2. TO_CHAR(Number Conversion): This number based function is used for transforming numeric value into varchar2 value optionally using specified number format. The Syntax is:
   TO_CHAR(n[,num_format])

TO_DATE: The TO_DATE function is used for converting a character string to its date equivalent. A date format string can be provided to tell oracle how to interpret a character String . The Syntax is:

TO_DATE(char_value[,for _string])

3. TO_NUMBER: The TO_NUMBER is used to convert a character value to a number value

MISCELLANEOUS FUNCTIONS

1. DECODE : the DECODE is a comparison function used for comparing a base value against up to 255 evaluation values in a single function call. Optionally a default value can be passed at the end of the function to be returned in the case where the base value does not equal any of the evaluation values. The syntax is:
   DECODE(base_value , eva_value1 ,ret_value1[,eva_value2 ,ret_value2]);
2. LEAST(value1,value2,…….): The Least function is used for returning the smallest value in a list of values based on the database character set.The value in This function may be columns expression or values.
3. GREATEST (value1, value2,…..) : It is just opposite of the least function . This Function is used for returning the greatest value in a list of values based on the database character set .
4. NVL: The NVL function is used for evaluating an expression and returning a given value if the expression evaluates to NULL . the Syntax is:
   NVL (expression, sub_value);
5. NVL2(expr,N_ret_val, NTN_ret_val): NVL2 takes the NVL function one step further . If the first expression in NVL@ is NULL, the 'N_ret_val' is returned . If value is NOT NULL, then NTN_ret_val is returned.
6. UID: The UID function returns a number that uniquely identifies the connected user.
7. USER: This function returns the name of the currently connected user.

AGGREGATE FUNCTIONS:

The Aggregate functions act on group of rows to give a result per group or rows rather on single rows. This is the reason why they are called aggregate function on group functions. Even though these functions are Group functions they do not require the use of a group by clause If a query with an aggregate functions returns no rows or only rows with NULLs for the argument to the aggregate function , this function returns NULL.

The Aggregate functions available in Oracle are:

1. **COUNT**: The Count function returns the total number of NON-NULL values in a set of records . Null values are not counted unless COUNT(*) is used , which counts the total number of rows in the table.The Syntax is:
   COUNT([DISTINCT/ALL]Columnname)

2. **SUM**: this function returns the sum of all the values for a column . it can apply only to columns with number data types . The Syntax is:
   SUM[DISTINCT/ALL]Columnname)

3. **AVG**: this function returns the average of all the NON-NULL values for a column. It can apply only to columns with NUMBER data type. The Syntax is :
   AVG([DISTINCT/ALL]Columnname)

4. **MIN and MAX** : This function returns the minimum value for a specified column which may assume any of the data types. The MAX function returns the maximumvalue for a specified column which may be any data type. The Syntax is:
   MIN([DISTINCT/ALL]Columnname)
   And similar for MAX function .

5. **VARIANCE(n)**: This function gives the variance of all values for a group of rows except NULL values. Here n is a column name of NUMBER data type.

6. **STDDEV(n)**: This function returns the standard deviation of all values for a group of rows except NULL values. Oracle calculates the standard deviation as square root of variance .Here n is a clumnname of Number data type.

GROUP BY CLAUSE

The Group By Clause is another clause in the select statement .This optional clause is used for grouping set of records in the result set of query for the purpose of aggregating data or displaying a single row of summary information for the entire group . The Syntax is

SELECT <select_list>
FROM <table_list>
[WHERE <condition>]
[GROUP BY<column1>, <column2>,……..<columnN>]
[HAVING<condition>]
[ORDER BY <expression>]

JOINS

A join is a mechanism that allows tables to be related to one another. The rows retrieved after joining two tables are based on the condition that a column in the first table which is defined as a foreign key must match to the column in the second table which is defined as a primary key referenced by the foreign key.

The various types of joins are:

- Equii-join
- Outer join
- Self join
- Cartesian join

EQUII-JOIN

The EQUII JOIN or INNER JOIN is a join in which join condition contains an equality operator (=). It combines rows that have equivalent values for the specified columns.

If the join condition is omitted from the join query, then the result is a Cartesian product. In the Cartesian product each row of one table is joined to every row of another table.

OUTER JOIN

A outer join simply extends the results of an INNER JOIN. While using the EQUII JOIN we have seen that if there

exists certain records in one table which do not have corresponding values in the second , then those rows will not be selected . We can forcefully select such rows by using OUTER JOIN. the results of an OUTER JOIN will be all those rows that satisfy the join condition , along with the rows from one table that do not have corresponding rows in the other table to satisfy the join condition . The OUTER JOIN is accomplished with a plus (+) operator within the parenthesis.

If you want to return all the rows from table A when performing join operation with table B then append the outer join operator (+) within parentheses to all columns of B in the join condition . for all rows in table A that have no matching rows in table B, Oracle returns NULL for any (select-list) expressions containing columns of B.

Outer join queries are subject to the following rules and restrictions:

- The Outer join (+) cannot be on both the sides in join condition.
- It can be applied only to a column not to arbitrary expression.
- A condition containing the (+) operator cannot be combined with another condition using the OR logical operator.
- A condition cannot use the IN comparison operator to compare a column marked with the (+) operator with an expression.
- A condition cannot compare any column marked with the (+) operator with a subquery.

SELF JOIN

Another form of join is the SELF JOIN which is a join of a table to itself. For SELFJOIN we have the need to open two copies of the same table. Since the table names are same so to avoid confusion we use aliasing that qualify the column names in the join condition. This type of join is used when a table has a foreign key that references its own primary key.

SET OPERATORS

Sometimes it is necessary to combine query results from two or more SQL queries into a single result. This is done with the help of three set operators

UNION [ALL]

INTERSECT

MINUS

The Syntax for using set operator is

<SELECT statement1><set operator ><SELECT statement2>

[ORDER BY Clause];

A query that contains a set operator is known as compound query . ORACLE provides some guidelines to be followed when writing compound queries:

- In each of the individual queries that make up a compound query, the number of columns and data types of the columns of <Select _list> must match.
- You can use same or different set operator with two or more tables. Precedence must be considered.
- You cannot specify an order by clause in any of the individual queries included in the compound query. However, you can order the result of the entire compound query.
- You cannot use set operators of LOB data types such as LOB, BLOB.
- Oracle uses the column names from the first SELECT statement for giving the query results and therefore only column names from the first SELECT statement can be used in the ORDER BY clause.

UNION [ALL]:

UNION set operator returns a table consisting of all rows either appearing in the result of <SELECT statement1> or in the result of <SELECT statement2>. Duplicates are automatically eliminated unless the clause ALL is used.

INTERSECT:

The INTERSECT set operator returns all those distinct rows that exist in both the <select statement1> and <select statement2>. None of those rows that are exclusively returned either in the <SELECT statement1> or in the <SELECT statement2> are included in the result set.

MINUS

The MINUS set operator returns all those rows that appear in the result of <SELECT statement1> but not in the <SELECT statement2>.

NESTED QUERIES

So far we have only concentrated on simple comparison conditions in a WHERE clause of a SELECT statement i.e. we have compared a column with a constant or we have compared two columns. As we have already seen for the INSERT statement, queries can be used for assignments to columns. A query result can also be used in a condition of a WHERE clause. In such a case a query is called a subquery and complete SELECT statement is called a nested query. In addition to a WHERE clause the subquery can be placed within a HAVING clause, FROM clause. One cannot use the ORDER BY clause within the subquery, and if used, it should be used as the last clause with the main SELECT statement.

The subqueries are of following types:

- **Single – Row Subqueries :** These queries return only one value from the subquery i.e. inner SELECT statement.
- **Multiple – Row Subqueries:** These queries return more row than one row from the subquery i.e. inner SELECT statement.
- **Multiple – Column Subqueries:** These queries return more than one column from the subquery i.e. inner SELECT statement.

  The following operators are used in the Multiple Row Subqueries:

  - IN operator
  - ANY operator
  - ALL operator

The following rules should be kept in mind when using subqueries :

- The nested query must be enclosed in the parentheses and it must appear to the right hand side of the comparison operator.
- A subquery cannot contain ORDER BY Clause.
- BETWEEN cannot be used with the subquery.
- The result of the subquery are not themselves displayed but are passed to the parent SQL statement for its use.
- The subquery must either have only one column or compare its selected columns to multiple columns in parentheses in the main query.
- The subqueries that produce only one row can be used with either single or many valued operators.