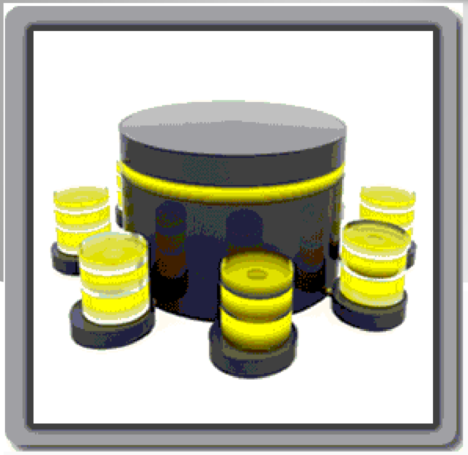
Database Concepts & SQL/Mysql



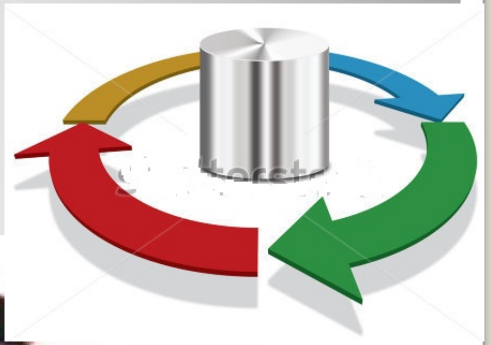
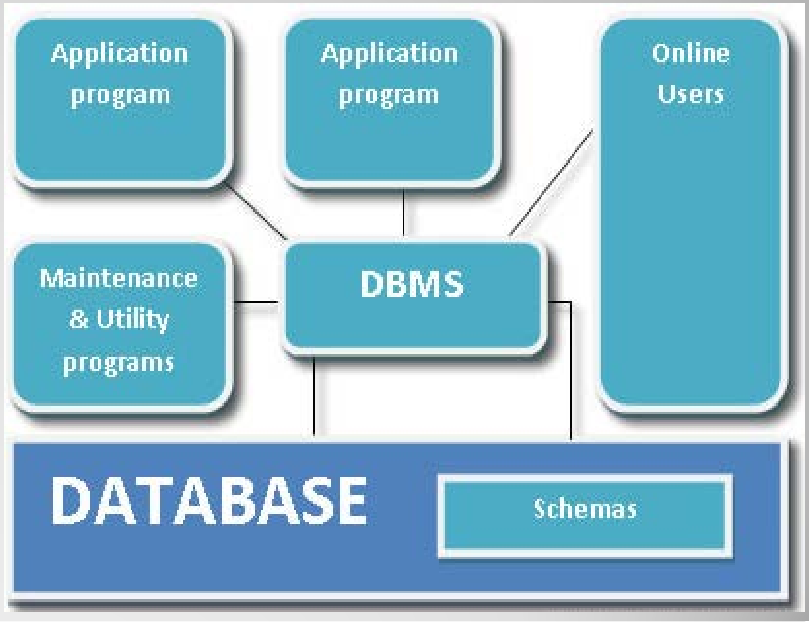


Table of Contents

Topic Name	Slide No.
Introduction to Database Concepts	3
Purpose of Databases	4
Databases Abstraction	5
Different Data Models	6
The Relational Model	7 - 14
Referential Integrity	15
Comparison of Data Models	16
Introduction to MySQL and SQL	17
MySQL Database System	18
Features and Advantages of MySQL	19
Starting MySQL	20
Processing Capabilities of SQL	21
Classification of SQL Statements	22 - 24
Difference between DDL and DML commands	22
Some MySQL SQL Elements	25 - 27
Difference between CHAR and VARCHAR datatypes	26
SQL Command Syntax	28
Integrity Constraints	29 - 35
Sample Database	36
Making Simple Queries	37 - 62

Introduction to Database Concepts :

- ✓ A collection of data is referred to as database and a database (management) system is basically a computer based record keeping system.
- ✓ It maintains any information that may be necessary to the decision-making processes involved in the management of the organization.
- ✓ The intention of a database is that the same collection of data should serve as many applications as possible.
- ✓ Database would permit not only the retrieval of data but also continuous modification of data needed for control of operations.
- ✓ It may be possible to search the database to obtain answers to queries or information for planning purpose.



Purpose of Databases :

A typical file processing system suffers from some major limitations like data redundancy, data inconsistency, unsharable data, unstandardized data, insecure data etc. On the other hand, a database system overcomes all these limitations and ensure continues efficiency.

The advantages provided by a database system are:

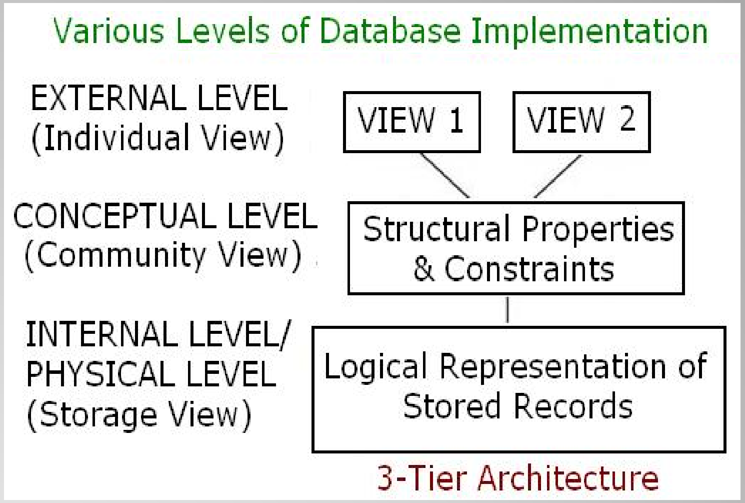
1. **Reduced data redundancy :**
   - ✓ Duplication of data is data redundancy. It leads to the problems like wastage of space and data inconsistency.
2. **Controlled data inconsistency:**
   - ✓ When the redundancy is not controlled, there may be occasions on which the two entries about the same data do not agree. At such times, database is said to be inconsistence.
   - ✓ If there is some redundancy retained in the database due to some technical reasons, the database management system ensures that any change made to either of the two entries is automatically made to the other.
3. **Shared data:**
   - ✓ The database allows sharing of data by several users. This means each user may have access to the same database/table/record at a same time.
4. **Standardized data:**
   - ✓ The database management system can ensure that all the data follow the applicable standards.
   - ✓ There may be some industry standards, organizational standards, and national or international standards.
   - ✓ Standardizing stored data formats is particularly desirable as an aid to data interchange or migration between systems.
5. **Secured data:**
   - ✓ Data is vital to any organization and some of it may be confidential. Confidential data must not be accessed by unauthorized persons.
   - ✓ Authentication schemes can be laid down, giving different levels of users, different permissions to access data.
6. **Integrated data:**
   - ✓ This means that data is accurate and consistent. Checks can be built in to ensure correct values are entered.
   - ✓ For example, while placing an order, the quantity must be a number above zero. Also, if an order is placed with a supplier, supplier must exist.

Databases Abstraction :

Data abstraction provides users with an abstract view of the system. It hides certain details of how the data is stored, created and maintained. A database management system allows users to access and modify data stores in the files. Each user may have different requirements and the data must be retrieved selectively and efficiently. The complex designs of the data structures are hidden from the users, thorough several levels of abstraction order to simplify user interaction with a system.

1. Various Levels of Database Implementation:

- **<u>Internal (or Physical) level</u>** – It describes how data are actually stored on the storage medium. At this level, complex low level structures are described in detail.
- **<u>Conceptual (or Logical) level</u>** – It describes what data are stored in the database. It also describes the relationships among the data. It is used by database administrators who decide what data is to be kept in the database.
- **<u>External (or View) level</u>** – Most users access only a part of the database and the system provides views according to the user's requirement.


Various Levels of Database Implementation
EXTERNAL LEVEL (Individual View)
VIEW 1
VIEW 2
CONCEPTUAL LEVEL (Community View)
Structural Properties & Constraints
INTERNAL LEVEL/ PHYSICAL LEVEL (Storage View)
Logical Representation of Stored Records
3-Tier Architecture

2. Concept of Data Independence:

Data independence is the ability to modify a scheme definition in one level without affecting a scheme definition in a higher level. Two types of Data Independence are:

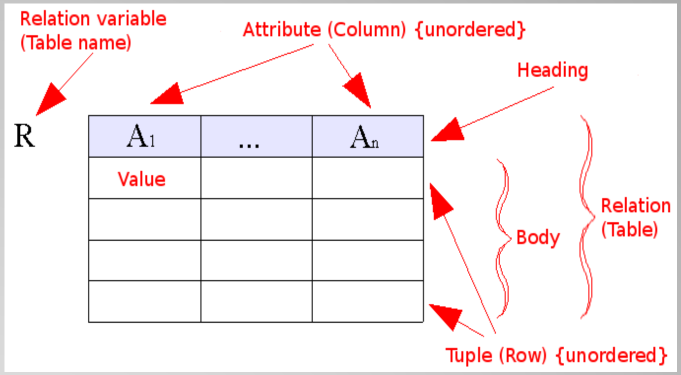
- **<u>Physical data independence</u>**
  - ✓ Modifies the scheme followed at the physical level without affecting the scheme followed at the conceptual level.
  - ✓ Modifications at the physical level are occasionally necessary in order to improve performance of the system.
- **<u>Logical data independence</u>**
  - ✓ Modifies the conceptual scheme without causing any changes in the schemes followed at view levels.
  - ✓ Modifications at the conceptual level are necessary whenever logical structure of the database get altered because of some unavoidable reasons.
  - ✓ More difficult to achieve because the application programs are heavily dependent on the logical structure of the database.

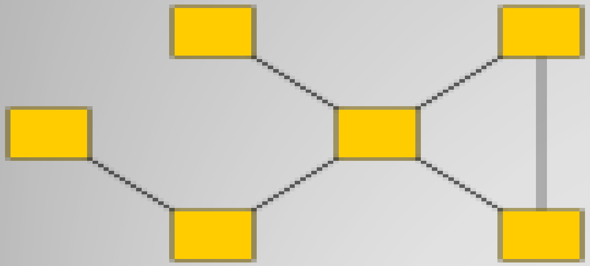
Different Data Models :

Data model is a collection of conceptual tools for describing data, data relationships, data semantics and consistency constraints. It helps in describing the structure of data at the logical level. It is a link between user's view of the world and bits stored in computer.

The three most widely accepted data models are the Relational, Network and Hierarchical models.

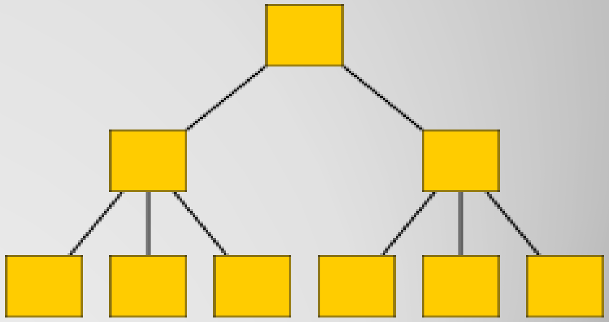
▪**Relational Data Model** – In this model, data is organized into tables with rows and columns. Each column has a unique name and is called an attribute. A row of the table represents a relationship among a set of values. As the table is a collection of such rows (or relationships), it has a close relationship with the mathematical concept of relation, from where this model takes its name.





▪**Network Data Model** – In this data model, data is represented by a collection of records and the relationships are represented by links. Each record is a collection of fields (attributes) each of which contains only one data value. A link is an association between two records.

▪**Hierarchical Data Model** – In this data model, data is represented by a collection of records and the relationships are represented by links. Each record is a collection of fields (attributes) each of which contains only one data value. Here, however records are organized as tree instead of arbitrary graphs.



The Relational Model :

Relational Model was proposed in 1970 by E.F. Codd of the IBM. It is a dominant model for commercial data processing applications. Nearly, all databases are based on this model. Let us explore this model in details.

1. Terminology

2. Views

3. Structure of Relational Databases

(A) Keys

4. The Relational Algebra

(A) The Select Operation

(B) The Project Operation

(C) The Cartesian Product Operation

(D) The Union Operation

(E) The Set Difference Operation

(F) The Set Intersection Operation

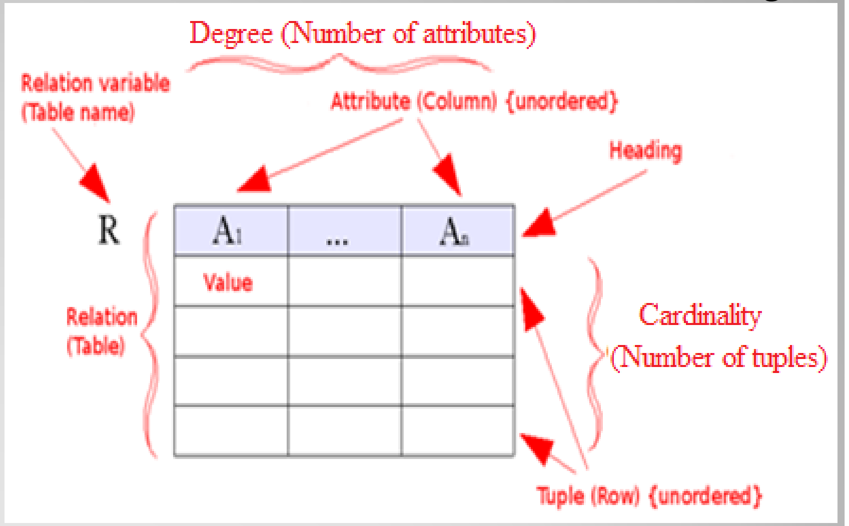
The Relational Model Continued :

Relational Model was proposed in 1970 by E.F. Codd of the IBM. It is a dominant model for commercial data processing applications. Nearly, all databases are based on this model. Let us explore this model in details.

1. Terminology:

Different terms used in relational model are being discussed here.

- **Relation:** A relation may be thought of as a set of rows with several columns. A relation has the following properties:
  - ✓ Row is a real world entity or relationship.
  - ✓ All values in particular column are of same kind.
  - ✓ Order of columns is immaterial.
  - ✓ Each row is distinct.
  - ✓ Order of rows is immaterial.
  - ✓ For a row, each column must have an atomic . value (indivisible).
  - ✓ For a row, a column cannot have more than one value.


Degree (Number of attributes)
Relation variable (Table name)
Attribute (Column) {unordered}
Heading
R
A1
...
An
Value
Relation (Table)
Cardinality (Number of tuples)
Tuple (Row) {unordered}

- **Domain:** A domain is a pool of values from which the actual value present in a given column are taken.
- **Tuple:** This is the horizontal part of the relation. One row represents one record of the relation. The rows of a relation are also called tuples.
- **Attributes** – The columns of a table are also called attributes. The column is the vertical part of the relation.
- **Degree:** The number of attributes(columns) in a relation determine the degree of a relation.
- **Cardinality** – It is the number of rows (or tuples) in a table.

BACK

The Relational Model Continued :

2. Views:

- ✓ A view is a pseudo-table or virtual table. It displays the data. The data is derived from one or more base tables.
- ✓ The view is a kind of table whose contents are taken upon other tables depending upon a given query condition.
- ✓ No stored file is created to store the contents of a view rather its definition is stored only.
- ✓ The usefulness of views lies in the fact that they provide an excellent way to give people access to some but not all of the information in a table
- ✓ Syntax:

```
CREATE VIEW <view name> AS SELECT <attribute list> FROM <table(s)> WHERE <condition(s)>;
```

3. Structure of Relational Database:

(A). **Keys:**

Keys come here for express difference among rows in terms of their attributes.

- **Primary Key –** It is a column (or columns) in a table that uniquely identifies each row. A primary key value is unique and cannot be null. There is only one primary key for a table.
- **Candidate key –** It is a column (or columns) that uniquely identify rows in a table. Any of the identified candidate keys can be used as the table's primary key.
- **Alternate key –** Any of the candidate keys that are not part of the primary key is called an alternate key.
- **Foreign key –** It is a column (or a set of columns) that refers to the primary key in another table i.e. it is used as a link to a matching column in another table.

To understand the concept of keys let's take an example, suppose in a school class there are four students who are eligible for being monitor. All these students are called *candidate key*. The student who selected for monitor will be treated as *primary key*.

Now suppose a student who become a monitor is not available in class, so the class teacher choose another student from rest three student who are eligible, for taking the responsibly of monitor. This student is called *alternate key*.

BACK

The Relational Model Continued :

4. The Relational Algebra:

The relational algebra is a collection of operations on relation \. The various operations of relational algebra are as following:

(A) The Select Operation:

- ✓ The select operation selects tuples from a relation that satisfy a given condition.
- ✓ The selection is denoted by lowercase Greek letter σ (sigma).
- ✓ Suppose we have table named Items as shown in Fig. (a) and to select those tuples from Items relation where the price is more than 19.00, we shall write

$$\sigma_{\text{price}} > 9.00 \text{ (Items)}$$

- ✓ That means from table Item, select the tuples satisfying the condition price > 9.00. The relation that results from above query is shown in Fig. (b).

Item#	Item-Name	Price
11	Milk	15.00
12	Cake	5.00
13	Bread	9.00
14	Ice Cream	14.00
15	Cold Drink	8.00

(a)

Item#	Item-Name	Price
11	Milk	15.00
14	Ice Cream	14.00

(b)

- ✓ In a selection condition, all the relational operators(=, ≠, <, ≤, >, ≥) may be used.
- ✓ More than one condition may be combined using connectives and(denoted by∧) and or (denoted by ∨).
- ✓ To find those tuples pertaining to prices between 5.00 and 9.00 from relation Items, we shall write

$$\sigma_{\text{price}} > 4.00 \wedge \text{price} < 9.00\text{(Items)}$$

- ✓ The result of this query is as shown in this table:

Item#	Item-Name	Price
12	Cake	5.00
15	Cold Drink	8.00

NOTE: Here, in select operation 'σ' is used only to see how technically the operation is done. In reality the SELE COMMAND is used to perform select operation instead of 'σ' which we will see in later slides.

BACK

The Relational Model Continued :

(B) The Project Operation:

- ✓ The Project Operation yields a "vertical" subset of a given relation in contrast to the "horizontal" subset returned by select operation.
- ✓ The projection lets you select specified attributes in a specified order and duplicating tuples are automatically removed.
- ✓ Projection is denoted by Greek letter pi(π).
- ✓ Suppose we have table Suppliers as in Fig.-(a) and to project Supplier names and their cities from the relation Supplier, we shall write $\pi_{\text{Supp-Name, City}}$ (Suppliers)
- ✓ The relation resulting from this query is as shown in Fig.-(b).

Supp#	Supp-Name	Status	City
S1	Britannia	10	Delhi
S2	New Bakers	30	Mumbai
S3	Mother Dairy	10	Delhi
S4	Cookz	50	Bangalore
S5	Haldiram	40	Jaipur

(a)

Supp-Name	City
Britannia	Delhi
New Bakers	Mumbai
Mother Dairy	Delhi
Cookz	Bangalore
Haldiram	Jaipur

(b)

- ✓ Duplicating tuples are automatically removed in the resulting relation. For instance, if you write

  π_{City} (Suppliers)
- ✓ The resulting relation will be as following:

City
Delhi
Mumbai
Bangalore
Jaipur

- ✓ Project operation can also be applied on a resulting relation of a query.
- ✓ Consider the table Items given in previous slide, if we want only the names of those items that are costlier than Rs. 9.00, we may write it as

  $\pi_{\text{Item-Name}}$ ($\sigma_{\text{Price > 9.00}}$ (Items))
- ✓ First the inner query is evaluated and then outer query is evaluated.
- ✓ The result of the above query is as following:

Item-Name
Cake
Cold Drink

NOTE: Here, in project operation 'π' is used only to see how technically the operation is done. In reality the SEI COMMAND is used to perform projext operation instead of 'π' which we will see in later slides.

BACK

The Relational Model Continued :

(C) The Cartesian Product Operation:

- ✓ The cartesian product is a binary operation and is denoted by a cross (x).
- ✓ The cartesian product of two relations a and B is written as A x B.
- ✓ The cartesian product of two relation yields a relation with all possible combination of the tuples of the two relations operated upon.
- ✓ All tuples of first relation are concatenated with all the tuples of second relation to form the tuples of the new relation.
- ✓ Suppose we have two relations Student and Instructor as following:

Student

Stud#	Stud-Name	Hosteler
S001	Meenakshi	Y
S002	Radhika	N
S003	Abhinav	N

Instructor

Inst#	Inst-Name	Subject
101	K. Lal	English
102	R.L. Arora	Maths

- ✓ The cartesian product of these two relations, Students x Instructor, will yield a relation that will have a degree of 6 (3 + 3: sum of degrees of student and Instructor) and a cardinality 8 (4 x 2: product of cardinalities of two relations).
- ✓ The resulting relation (Students x Instructor) is as following:

Stud#	Stud-Name	Hosteler	Inst#	Inst-Name	Subject
S001	Meenakshi	Y	101	K. Lal	English
S001	Meenakshi	Y	102	R.L. Arora	Maths
S002	Radhika	N	101	K. Lal	English
S002	Radhika	N	102	R.L. Arora	Maths
S003	Abhinav	N	101	K. Lal	English
S003	Abhinav	N	102	R.L. Arora	Maths

- ✓ See the resulting relation contains all possible combinations of tuples of the two relations.

BACK

The Relational Model Continued :

(D) The Union Operation:

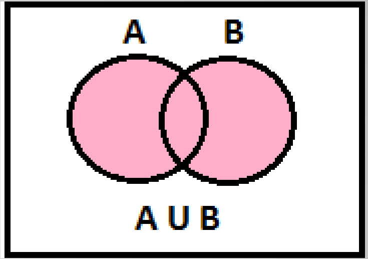
- ✓ The union operation requires two relation and produces a third relation that contains tuple from both the operand relation.
- ✓ The union operation is denoted by U. Thus, to denote the union of two relation X and Y, we will write as X U Y.
- ✓ For a union operation A U B to be valid, the following two conditions must be satisfied by the two operands A and B:
  1. The relations A and B must be of the same degree. That is, they must have the same number of attributes.
  2. The domains of the *i*th attributes of A and the *i*th attributes of b must be the same.
- ✓ Suppose we have two Drama and Song as following:

Drama

Rollno	Name	Age
13	Kush	15
17	Swati	14

Song

Rollno	Name	Age
2	Manya	15
10	Rishabh	15
13	Kush	15



- ✓ Result of Song U Drama will be as following:

Rollno	Name	Age
2	Manya	15
10	Rishabh	15
13	Kush	15
17	Swati	14

- ✓ Notice that one duplicating tuple (13, Kush, 15) has been automatically removed.

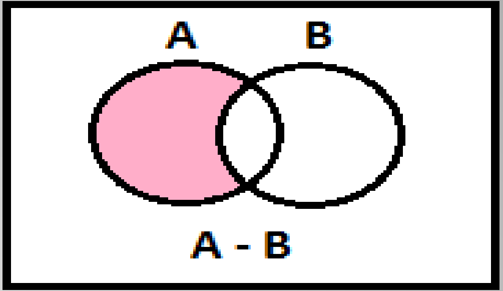
BACK

The Relational Model Continued :

(E) The Set Difference Operation:

- ✓ The set difference operation denoted by – (minus) allows us to find tuples that are in one relation but not in another.
- ✓ The expression A – B results in a relation containing those tuples in A but not in B.
- ✓ Suppose we have two Drama and Song as given in previous slide.
- ✓ Result of Song – Drama will be as following:

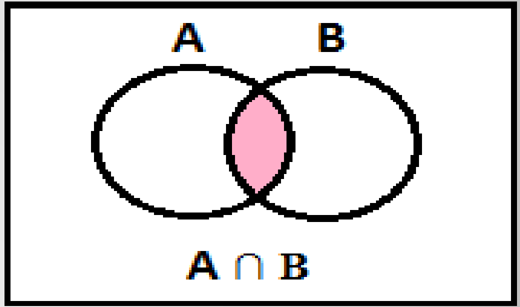
Rollno	Name	Age
2	Manya	15
10	Rishabh	15



(F) The Set Intersection Operation:

- ✓ The set intersection operation finds tuples that are common to the two operands relations.
- ✓ The set intersection operation is denoted by ∩. That means A ∩ B will yield a relation having tuples common to A and B.
- ✓ Suppose we have two Drama and Song as given in previous slide.
- ✓ Result of Song ∩ Drama will be as following:

Rollno	Name	Age
13	Kush	15



NOTE: **Any relational algebra expression using set intersection can be rewritten by replacing the intersection op with a pair of set difference operations as: A ∩ B = A – (A – B)**

BACK

Referential Integrity :

Referential Integrity is a system of rules that a DBMS uses to ensure that relationships between records in related table are valid, and that users don't accidentally delete or change related data.

Conditions to set Referential Integrity:

- ✓ The matching field from the primary table is a primary key or has a unique index
- ✓ The related fields have the same data type
- ✓ Both tables belong to the same database

When referential integrity is enforced, given rules should be followed:

- ✓ One can't enter a value in the foreign key field of the related table that doesn't exist in the primary key of the primary table.
- ✓ One can't delete a record from a primary table, if matching records exist in related table.
- ✓ One can't change a primary key value in the primary table, if that record has related records.

For example, suppose Table B has a *foreign key* that points to a field in Table A. Referential integrity would prevent you from adding a record to Table B that cannot be linked to Table A. In addition, the referential integrity rules might also specify that whenever you delete a record from Table A, any records in Table B that are linked to the deleted record will also be deleted. This is called *cascading delete.*

Finally, the referential integrity rules could specify that whenever you modify the value of a linked field in Table A, all records in Table B that are linked to it will also be modified accordingly. This is called *cascading update.*

Comparison of Data Models:

Characteristic	Hierarchical model	Network model	Relational model
Data structure	✓One to many or one to one relationships ✓Based on parent child relationship	✓Allowed the network model to support many to many relationships ✓A record can have many parents as well as many children.	✓One to One, One to many, Many to many relationships ✓Based on relational data structures
Data manipulation	✓Does not provide an independent stand alone query interface ✓Retrieve algorithms are complex and asymmetric	✓Uses CODASYL (Conference on Data Systems Languages) ✓Retrieve algorithms are complex and symmetric	✓Relational databases are what brings many sources into a common query (such as SQL) ✓Retrieve algorithms are simple and symmetric
Data integrity	✓Cannot insert the information of a child who does not have any parent. ✓Multiple occurrences of child records which lead to problems of inconsistency during the update operation ✓Deletion of parent results in deletion of child records	✓Does not suffer form any insertion anomaly. ✓Free from update anomalies. ✓Free from delete anomalies	✓Does not suffer from any insert anomaly. ✓Free form update anomalies ✓Free from delete anomalies

Introduction to MySQL and SQL:

MySQL:

- ✓ MySQL is a freely available open source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL).
- ✓ MySQL can be downloaded from site www.mysql.org. MySQL is created and supported by MySQL AB, a company based in Sweden.
- ✓ In MySQL database, information is stored in Tables. A single MySQL database can contain many tables at once and store thousands of individuals records.
- ✓ MySQL provides you with a rich set of features that support a secure environment for storing, maintaining, and accessing data.
- ✓ MySQL is a fast, reliable, scalable alternative to many of the commercial RDBMs available today.

Structured Query Language (SQL)

- ✓ In order to access data within the MySQL database, all programmers and users must use, Structured Query Language (SQL).
- ✓ SQL is the set of commands that is recognized by all RDBMS.
- ✓ The Structured Query Language (SQL) is a language that enables you to create and operate on relational database, which are sets of related information stored in tables.
- ✓ The SQL (Structured Query Language) has proved to be a standard language as it allows users to learn one set of command and use it to create, retrieve, alter, and transfer information regardless of whether they are working on a PC, a workstation, a mini, or a mainframe.

MySQL Database System:

MySQL Database System is a combination of a MySQL server instance and a MySQL database. MySQL database system operates using client/server architecture, in which the server runs on the machine containing the databases and clients connect to the server over a network.

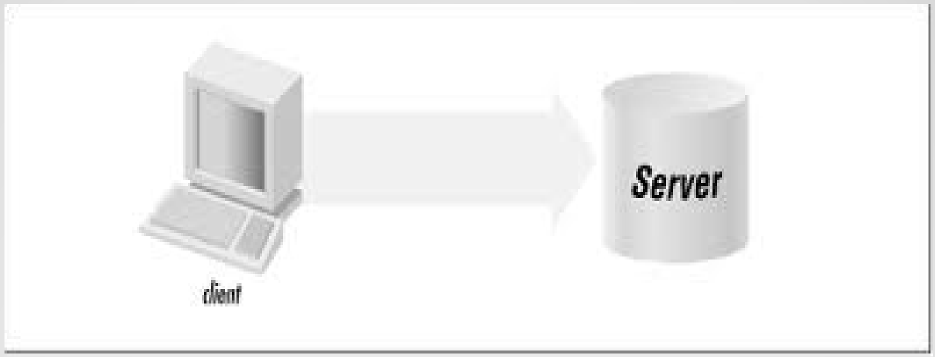
SQL Server and Clients:

MySQL Server :

- ✓ Listens for client request coming in over the network
- ✓ Accesses database contents according to those requests
- ✓ Provides contents to the clients

MySQL Clients:

- ✓ MySQL clients are programs that connect to the MySQL server and issue queries in a pre-specified format. MySQL is compatible with the standards based SQL. The client program may contact the server programmatically or manually.



Features and Advantages of MySQL:

Features of MySQL:

1. Speed: If the server hardware is optimal, MySQL runs very fast.

2. Ease of use: MySQL is a high-performance, relatively simple database system.

3. Cost: Available free of cost.

4. Query Language Support: Understands standard based SQL.

5. Portability: Provides portability as it has been tested with a broad range of different compiler and can work on many different platforms.

6. Data Types: Provide many data types to support different types of data.

7. Security: Offers a privilege and password system that is very flexible and secure.

8. Localization: The server can provide error messages to clients in many languages.

9. Connectivity: Clients can connect to MySQL Server using several protocols.

10.Client and Tools: Provides command-line programs such as mysqldump and mysqladmin, and graphical programs such as MySQL Administrator and MySQL Query Browser.

Advantages of MySQL:

1. **Reliability and performance**: MySQL is very reliable and high performance relational database management system.

2. **Availability of source**: MySQL source code is available that is why now we can recompile the source code.

3. **Cross-Platform support**: MySQL supports more than twenty different platforms including the major Linux distribution. Mac OS X, Unix and Microsoft Windows.

4. **Powerful uncomplicated software**: The MySQL has most capabilities to handle most corporate database applications and is very easy and fast.

5. **Integrity**: MySQL provides forms of integrity checking.

6. **Authorization**: MySQL DDL includes commands for specifying access rights to relations and views.

Starting MySQL:

- ✓ To start MySQL make sure that MySQL Server is installed on your machines.
- ✓ Once it is installed, you need to click at

```
Start
  | _ _  All Programs
            |_ _MySQL
                  |_ _ MySQL Server
                            |_ _ MySQL Command Line Client
```

- ✓ It will start the MySQL client where you have to specify the password before start working.
- ✓ We can quit from MySQL by typing Quit at the mysql> prompt.

Processing Capabilities of SQL:

The various processing capabilities of SQL are:

1. **Data Definition Language (DDL):** The SQL DDL provides commands for defining relation schemas, dleting relations, creating indexes, and modifying relation schemas.

2. **Interactive Data Manipulation Language (DML):** The SQL DML includes a query language based on both the relational algebra and the tuple relational calculus. It also includes commands to insert, delete, and modify tuples in the database.

3. **Embedded Data Manipulation Language:** The embedded form of SQL is designed for use within general-purpose programming language such as PL/1, Cobol, Fortran, Pascal, and C.

4. **View Definition:** The SQL DDL also includes commands for defining views.

5. **Authorization:** The SQL DDL includes commands for specifying access rights to relations and views.

6. **Integrity:** The SQL provides forms of integrity checking. Further products and standards of SQL are likely to include enhanced features for integrity checking.

7. **Transaction control:** SQL includes commands for specifying the beginning and ending of transaction along with commands to have a control over transaction processing.

Classification of SQL Statements:

SQL provides many different types of commands used for different purposes. These commands can be divided into following categories:

1) Data Definition Language (DDL) commands
2) Data Manipulation Language (DML) commands
3) Transaction Control Language (TCL) commands
4) Session Control commands
5) System Control commands.

Difference between DDL and DML commands

DDL Commands	DML Commands
✓DDL stands for Data Definition Language.	✓DML stands for Data Manipulation Language.
✓Allow us to perform task related to data definition.	✓Used to manipulate data.
✓CREATE, ALTER, DROP, GRANT, REVOKE etc are DDL commands.	✓INSERT INTO, UPDATE, DELETE, SELECT etc are DML commands.
✓DDL have no types.	✓DML are of two types: 1. Procedural DMLs 2. Non-procedural DMLs

Classification of SQL Statements Contiued:

1. Data Definition Language (DDL) commands:

- ✓ A database scheme is specified by a set of definitions which are expressed by a special language called a data definition language(DDL).
- ✓ The result of DDL statements is a set of tables which are stored in a special file called data dictionary.
- ✓ A Data Dictionary is a file that contains “metadata” i.e., “data about data”.
- ✓ The DDL provides a set of definitions to specify the storage structure and access methods used by the database system.

Listed are few ideal DDL functions:

1. It should identify the types of data division such as data item, segment, record, and data-base file.
2. It should give a unique name to each data-item-type, record-type, file-type, database, and other data subdivision
3. It should specify the proper data types.
4. It should specify how the record types are related to each other.
5. It may define the type of encoding the program uses in the data items, binary, character, bit, string etc.

Data Definition Language (DDL) commands allow us to perform tasks related to data definition. One can perform the following tasks:

i. **Create, alter and drop schema objects** - DDL commands are used to create, define, change or delete objects such as a table, a view or an index etc.

ii. **Grant and revoke privileges and roles -** DDL commands are used to grant or revoke permissions or privileges to work upon schema objects.

iii. **Maintenance commands -** These commands are used to analyze information on a table with an aim of maintaining it.

BACK

Classification of SQL Statements Contiued:

2. Data Manipulation Language (DML) commands

- ✓ Data manipulation means:
  - ➢ Retrieval of information stored in database
  - ➢ Insertion of new information into database
  - ➢ Deletion of information from database
  - ➢ Modification of data stored in database
- ✓ A Data Manipulation Language (DML) is a language that enables user to access or manipulate data as organized by the appropriate data model.
- ✓ DML commands query and manipulate data in existing schema objects. Some DML commands are as following:
  - ➢ **INSERT INTO:** Used to insert a tuple in a table.
  - ➢ **UPDATE:** Used to modify a tuple in a table.
  - ➢ **DELETE:** Used to delete a tuple in a table.
  - ➢ Other examples of DML commands are: SELECT, LOCK TABLE etc.
- ✓ Types of DMLs:
  1. **Procedural DMLs -** These require a user to specify what data is needed and how to get it.
  2. **Non-Procedural DMLs** - These require a user to specify what data is needed without specifying how to get it.

3. Transaction Control Language (TCL) commands

- ✓ To manage and control the transactions, the transaction control commands are used. These commands manage changes made by DML commands.
- ✓ Some TCL commands are as following:
  - ➢ **COMMIT -** Makes all the changes made by statements issued, permanent.
  - ➢ **ROLLBACK -** Undoes all changes since the beginning of a transaction or since a savepoint.
  - ➢ **SAVEPOINT -** Marks a point up to which all earlier statements have been successfully completed and if required – in case of failure – one may undo the changes, i.e., rollback up to this very point.
  - ➢ **SET TRANSACTION -** Establishes properties for the current transactions.

BACK

Some MySQL SQL Elements:

Some basic elements that play an important role in defining/querying a database are:

1. Literals
2. Datatypes
3. Nulls
4. Comments.

1.Literals

- ✓ Literals are fixed data values.
- ✓ A fixed data value may be of character type or numeric literal.
- ✓ All character literals are enclosed in single quotation marks or double quotation marks e.g., ‘Synthiya’, ‘Ronak Raj Singh’, ‘8’.
- ✓ Numbers that are not enclosed in quotation marks are numeric literals e.g., 22, 18, 1997 all numeric literals.
- ✓ Numeric literals can either be integer literals or be real literals e.g., 17 is an integer literal but 17.0 and 17.5 are real literals.

2. Datatypes:

MySQL uses many different data types, divided into three categories:

(A) Numeric (B) Date and time, and (C) String types

Most commonly used data types in MySQL are as following:

Data Type	Syntax	Explanation
INTEGER	INTEGER	It represent a number without a decimal point. The range of integer is from -2,147,483,648 to 2,147,483,647. Width of integer is up to 11 digits.
SMALLINT	SMALLINT	A 16-bit signed integer value. The range of SMALLINT is from -32768 to 32767. Width of SMALLINT is up to 5 digits.
NUMERIC	NUMERIC(P,S)	For example, numeric(6,2) is a number that has 4 digits before the decimal and 2 digits after the decimal.
DECIMAL	DECIMAL(P,S)	For example, decimal(6,2) is a number that has 4 digits before the decimal and 2 digits after the decimal.

Continued

2. Datatypes Continued:

Data Type	Syntax	Explanation
CHARACTER	CHAR(X)	Where ***x*** is the number of characters to store. This data type is space padded to fill the number of characters specified.
CHARACTER VARYING	VARCHAR2(X)	Where ***x*** is the number of characters to store. This data type does NOT space pad.
DATE	DATE	Stores year, month, and day values.
TIME	TIME	Stores the hour, minute, and second values.
TIMESTAMP	TIMESTAMP	Stores year, month, day, hour, minute, and second values.
BLOB	BLOB	A field with a maximum length of 65535 characters. BLOBs are "Binary Large Object" and used to store large amount of data, such as images or other types of files.

Difference between CHAR and VARCHAR datatypes

CHAR	VARCHAR
✓The CHAR datatype specifies a fixed length character string.	✓VARCHAR specifies a variable length string.
✓The length should be a number from 0 to 255	✓The maximum length can be a number up to 65,535.
✓CHAR(*n*) will take n characters of storage even if you enter less than n characters to that column.	✓VARCHAR(*n*) will take only the required storage for the actual number of characters entered to that column.
✓If a value is shorter than this length *n* then blanks are added.	✓No blanks are added if the length is shorter than maximum length *n*.
✓With CHAR data type column, search seems working faster than VARCHAR data type column	✓With VARCHAR data type column, search seems working slower than CHAR data type column.

BACK

Some MySQL SQL Elements Continued:

3. Nulls:

- ✓ Columns having no value is said to have NULL value.
- ✓ Can appear in columns of any data type provided that are not restricted by NOT NULL or PRIMARY KEY integrity constraints.
- ✓ We should not use null to represent a value of zero, because they are not equivalent.
- ✓ Any arithmetic expression containing a null always evaluates to null.

4. Comments:

- ✓ A comment is a text that is not executed; it is only for documentation purpose.
- ✓ Does not effect the statement execution.
- ✓ Describe the statement's purpose within an application.
- ✓ Can appear between any keywords, parameters or punctuation mark in a statement.

We can include a comment in a statement in a statement using either of these means:

- ➢ **Begin the comment with /* :** Proceed with the text of the comment. This text can span multiple lines. End the comment with */
- ➢ **Begin the comment with -- :** Proceed with the text of the comment. This text cannot extend to a new line. End the comment with a line break.
- ➢ **Begin the comment with #:** Proceed with the text of the comment. This text cannot extend to a new line. End the comment with a line break.

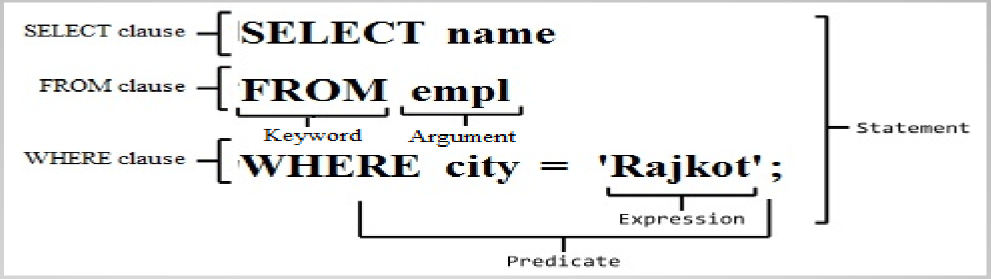
Example:

```
SELECT ename, sal, job, loc        /* Select all employees whose compensation is grater that 300. */
FROM empl, dept
WHERE  empl.deptno = dept.deptno               # joiing two tables
AND sal > 300;            -- this condition is testing whether salary is more than 300 or not.
```

SQL Command Syntax:

Conventions and Terminology used in SQL Query:

- ✓ **Keywords:** Keywords are words that have a special meaning and printed in capital letters. They are understood to be instruction.
- ✓ **Commands or Statements:** Commands are instruction given by us to a SQL database. Commands consist of one or more logically distinct parts called clauses.
- ✓ **Clauses:** Clauses begin with a keyword for which they are generally named, and consist of keywords and arguments.
- ✓ **Arguments:** Argument complete or modify the meaning of a clause.



Symbols used in Syntax Statements:

Symbol	Meaning
|	Symbolic way of saying "or" i.e., whatever precedes this symbol may optionally be replaced by whatever follows it.
{}	Everything enclosed in it, is treated as a unit for the purpose of evaluating |, . , . . Or other symbols.
[]	Everything enclosed in it is optional.
…	This means whatever precedes it may be repeated any number of times.
.,..	Whatever precedes this, may be repeated any number of times with the individual occurrence separated by commas.
<>	SQL and other special terms are in angle brackets.

- ✓ A statement in MYSQL SQL is completed with a semicolon(;).
- ✓ Commands in SQL are not case-sensitive but text literals or character literals that we specify in quotation marks are case-sensitive.

Integrity Constraints:

(A) Introduction to Constraint

(B) Different Constraints

(C) Applying Table Constraints

(D) Assigning Names to Constraints

(A) Introduction to Constraint:

- ✓ A Constraints is a condition or check applicable on a field or set of fields.
- ✓ Constraints are also known as integrity constraints because they applied to maintain data integrity.
- ✓ Two types of constraints:
  1. **Column constraints:** Apply only to individual columns.
  2. **Table constraints:** Apply to group of one or more columns.

Syntax:

```
CREATE TABLE <table name>
      (    <column name> <data type> [ ( <size> ) ] <column constraint>,
           <column name> <data type> [ ( <size> ) ] <column constraint>…
           <table constraint> (<column name>, [, <column name> … ] ) …);
```

The fields in parenthesis after the table constraints are the fields to which they apply. The column constraints apply to the columns whose definition they follow.

Example:

```
CREATE TABLE emp
      (  Eno       integer      NOT NULL,
         Ename     char(20)     NOT NULL,
         Job       char(20)     NOT NULL,
         Salary    decimal    );
```

The above command creates emp table, in which column Eno, Ename, and Job can never have NULL values. Any attempt to put NULL values in these columns will be rejected.

Integrity Constraints Continued:

(B) Different Constraints:

Following are the different constraints in MySQL:

1. NOT NULL
2. UNIQUE
3. PRIMARY KEY
4. DEFAULT
5. CHECK
6. FOREIGN KEY

1. NOT NULL Constraint

- ✓ The NOT NULL constraint enforces a column to NOT accept NULL values.
- ✓ The NOT NULL constraint enforces a field to always contain a value. This means that you cannot insert a new record, or update a record without adding a value to this field.

Example: CREATE TABLE emp

```
(   Eno        integer      NOT NULL,
    Ename      char(20)     NOT NULL,
    Job        char(20)     NOT NULL,
    Salary     decimal    );
```

The above table enforces the Eno column and the Ename column to not accept NULL values.

NOTE: **MySQL also supports some other constraint such as ENUM and SET constraint that are used to define columns that can contain only a given set of values.**

BACK

Integrity Constraints Continued :

(B) Different Constraints Continued:

2. UNIQUE Constraint:

- ✓ This constraint ensure that no two rows have the same value in the specified columns.
- ✓ When applied to the columns ensure that there cannot exist more than one NULL value in the column.

Example:

```
CREATE TABLE emp
  (  Eno      integer     NOT NULL UNIQUE,
     Ename    char(20)    NOT NULL,
     Job      char(20)    NOT NULL,
     Salary   decimal   );
```

Here, multiple constraints NOT NULL and UNIQUE have been applied on one column by putting space in between. The comma(,) comes in the end of column definition.

3. Primary Key Constraint:

- ✓ This constraint declares a column as primary key of the table.
- ✓ Only one column or one combination can be applied in this constraint.
- ✓ The primary key cannot allow NULL values, thus this constraints must be applied to columns declared as NOT NULL.

Example:

```
CREATE TABLE emp
  (        Eno      integer     NOT NULL PRIMARY KEY ,
           Ename    char(20)    NOT NULL,
           Job      char(20)    NOT NULL,
           Salary   decimal   );
```

BACK

Integrity Constraints Continued :

(B) Different Constraints Continued:

4. Default Constraint:

- ✓ A default value can be specified for a column using the DEFAULT clause.
- ✓ When a user does not enter a value for the column, automatically the defined default value is inserted in field.
- ✓ A column can have only one default value.

Example:

```
CREATE TABLE emp
      (         Eno       integer      NOT NULL PRIMARY KEY ,
                Ename     char(20)     NOT NULL,
                Job       char(20)     DEFAULT = ‘CLERK’,
                Salary    decimal  );
```

Here, if no value is provided for job, the default value of ’CLERK’, will be entered.

5. Check Constraint:

- ✓ This constraint limits values that can inserted into a column of table. **Check constraint will not work because The CHECK clause is parsed but ignored by all storage engines.**

Example:

```
CREATE TABLE emp
      (         Eno       integer      NOT NULL PRIMARY KEY ,
                Ename     char(20)     NOT NULL,
                Job       char(20)     NOT NULL,
                Salary    decimal      CHECK > 2000   );
```

- ✓ This statement ensure that the value inserted for salary must be greater than 2000.
- ✓ When a check constraint involves more than one column from the same table, it is specified after all the column have been defined.

NOTE: **CHECK constraint involves more than one column from the same table, it is specified after all the columns have been defined, which we will see in letter slides.**

BACK

Integrity Constraints Continued :

(B) Different Constraints Continued:

6. Foreign Key Constraint:

✓ Tables references one another through common fields and to ensure validity of references, referential integrity is enforced.

✓ Referential integrity is ensured through FOREIGN KEY constraint.

Syntax: columnname dataype (size) REFERENCES tablename
[(columnname)] [ON DELETE CASCADE] [ON UPDATE CASCADE]

✓ Here the first *columnname* is the name of the related column in *child table*.

✓ The *columnname* appearing after REFERENCES clause is the name of related column in the *parent table*.

✓ The *tablename* after REFERENCES clause is the name of the *parent table* to which the column in current table is related.

✓ If ON DELETE CASCADE is used, then in case a row is deleted in parent table, all its rows in the child table will automatically be deleted.

✓ If ON UPDATE CASCADE is used, then in case a row is deleted in parent table, all its rows in the child table will automatically be deleted.

Example: Suppose we have two tables as following:

Persons (P_Id, Name, Address, City)
Orders (O_Id, Ono, P_Id)

Here, P_ID is a primary key in a Persons table and foreign key in a Orders table.

```
CREATE TABLE Persons          ⇐ Parent table
(    P_Id integer(10) NOT NULL PRIMARY KEY
     Name char(10), Address char(20),  City char(20)
);
CREATE TABLE Orders           ⇐ Child table
(    O_Id integer(10), Ono integer(10),          → Primary key of parent table
     P_Id integer(10) REFERENCES Persons(P_Id),
);
        ↘ Foreign key          ↘ Parent table's name
```

NOTE: **The foreign key constraint can also be applied through FOREIGN FEY table constraint, which we will see in letter slides.**

(C) Applying Table Constraints:

✓ When a constraint is to be applied on a group of columns of a table, it is called *table constraint*.

✓ The table constraint appear in the end of the definition.

Example: CREATE TABLE items

```
(       icode       char(5)     NOT NULL ,
        descp       char(20)    NOT NULL,
        ROL         integer,
        QOH         integer,
        CHECK       (ROL < QOH),
        PRIMARY KEY  (icode, descp)  );
```

These are a table constraints.

The above statement define a primary key that have two columns namely icode and descp. It also ensures that the ROL must be less than QOH in each row.

✓ To define foreign key constraint for a group of columns, FOREIGN KEY constraints should be used.

Example: CREATE TABLE Orders

```
(       Orderno       Number(6, 0)     NOT NULL ,
        O_Id          integer(15)      NOT NULL,
        City          char(20),
        Itemno        char(5),
        FOREIGN KEY  (Orderno, Itemno)  REFERENCED  OrdMaster  (Orderno, Itemno)
);
```

These is a table constraints.

In above statement both columns Orderno and Itemno are declared as FOREIGN KEY.

BACK

Integrity Constraints Continued :

(D) Assigning Names to Constraints:

- ✓ By default, MySQL assigns a unique name to each constraint defined by us.
- ✓ MySQL names constraints as:

  SYS_Cn

- ✓ Where n is an integer that makes the constraint name unique.
- ✓ For example, SYS_C003215, SYS_C008596 etc. are constraint names generated by MySQL.

We can assign a name to the constraint by following syntax:

Syntax: CONSTRAINT < name-of-constraint> <definition-of-constraint>

Example:

```
CREATE TABLE items
    (       icode       char(5)       CONSTRAINT p_Itemkey PRIMARY KEY,
            descp       char(20)      NOT NULL,
            ROL         integer,
    );
CREATE TABLE Orders
    (       Orderno       Number(6, 0)       NOT NULL ,
            O_Id          integer(15)        NOT NULL,
            Itemno        char(5),
            CONSTRAINT Group_fkey_Orders FOREIGN KEY (Orderno, Itemno)
            REFERENCED  OrdMaster  (Order#, Item#)   );
    );
```

BACK

Sample Database:

In this presentation we are assume that we have database named SampleDB in which there are two tables namely EMP and DEPT.

Database: SampleDB

Tables_in_ Sample_DB
EMP
DEPT

Table: EMP

Eno	Ename	Job	Mgr	Hiredate	Sal	Comm	Dno
115	RAJ	MANAGER	156	17-NOV-81	2000		10
121	SONALI	CLERK	115	09-JUN-81	950		30
196	ANKIT	SALSEMAN	147	07-APR-81	1250	300	10
134	SIYA	CLERK	178	21-SEP-81	1100		20
182	RADHIKA	ANALYST	178	20-FEB-81	3200		30
147	PAYAL	MANAGER	156	03-DEC-81	2450		30
156	MANOJ	PRESIDENT		22-FEB-81	5000		10
123	KRUPA	SALSEMAN	147	17-DEC-80	1600	1200	20
195	SAGAR	CLERK	115	12-JAN-83	800		NULL
178	NEHA	MANAGER	156	23-JAN-82	2975		20

Table: DEPT

Dno	Dname	Loc
10	ACCOUNTING	RAJKOT
20	RESEARCH	AHEMDABAD
30	SALSE	BARODA
40	OPERATIONS	JAMNAGAR

Making Simple Queries:

1. Creating Databases
2. Opening Databases
3. Removing Databases
4. CREATE TABLE Command
5. Creating Table from Existing Table
6. Viewing a Table Structure
7. Inserting Data Into Table
8. Modifying Data with UPDATE Command
9. Deleting Data with DELETE Command
10. ALTER TABLE Command
11. The DROP TABLE Command
12. The DROP VIEW Command
13. The SELECT Command
14. Reordering Columns in Query Results
15. Selecting All Columns
16. Eliminating Redundant Data - DISTINCT keyword
17. Selecting From All the Rows – ALL keyword
18. How to Perform Simple Calculations ?
19. Scalar Expression with Selected Fields
20. Using Column Aliases
21. Handling Nulls
22. Putting Text in the Query Output
23. Selecting Specific Rows – WHERE clause
24. Relational Operator
25. Logical Operators
26. Condition Based on a Range
27. Condition Based on a List
28. Condition Based on Pattern Matches
29. Searching for NULL
30. Operator Precedence
31. Sorting Results – ORDER BY clause
32. Sorting by Column Alias

Making Simple Queries Continued:

1. Creating Databases:

The CREATE DATABASE command is used to create a database.

Syntax: CREATE DATABASE [IF NOT EXISTS] <database name>;

The IF NOT EXISTS clause, if used, will first test whether a database by mentioned name already exists or not. If exists, then create database command is simply ignored, otherwise a database with the mentioned name is created.

Example: CREATE DATABASE SampleDB; ⟸ *Creates database having name as SampleDB*

CREATE DATABASE IF NOT EXISTS SampleDB; ⟸ *Creates database having name as SampleDB, If there is no database by the name SampleDb already existing.*

2. Opening Databases:

To perform operation on database we first need to open database .

Syntax: USE <database name>;

Example: USE SampleDB;

Before opening a database we need to ensure that it must already exist.

To check the names of existing database, following command is used:

SHOW DATABASES;

3. Removing Databases:

When we drop a database, all its tables also get removed along with the database.

Syntax: DROP DATABASE <database name>;

Example: DROP DATABASE SampleDB;

BACK

Making Simple Queries Continued :

4. CREATE TABLE Command:

- ✓ The CREATE TABLE statement is used to create a table in a database.
- ✓ Each table must have at least one column.

Syntax:

```
CREATE TABLE <table-name>
( <column name>  <data type>  [ (size)],
  <column name> <data type>  [ (size) …]  );
```

Example:

```
CREATE TABLE emp
  (  Eno        integer,
     Ename      char(20),
     Job        char(20),
     Mgr        integer(10),
     Hiredate   date,
     Sal        decimal,
     Comm       decimal,
     Dno        integer   );
```

5. Creating Table from Existing Table:

- ✓ By using SELECT statement with CREATE TABLE we can define table and put data into it without going through the usual data definition process..
- ✓ The new table stores the result produced by the SELECT statement.
- ✓ The name of the new table must be unique..

Example:

```
CREATE TABLE emp_info AS
      (    SELECT eno, ename
           FROM emp
           WHERE eno > 122  );
```

Above statement create a new table called *emp_info* that stores two columns: *eno* and *ename* for the employees that have *eno* grater than 122 in relation *emp*.

The newly created table *emp_info* will look as shown below:

eno	ename
115	RAJ
121	SONALI

2 rows in set (0.01 sec)

NOTE: **When create a table, we can place constraint on the values that can be entered into its fields, which we have seen in previous slides.**

BACK

Making Simple Queries Continued :

6. Viewing a Table Structure:

✓ To view the structure of an already created/existing table, we may use DESCRIBE or DESC command is used.

Syntax: DESCRIBE | DESC <tablename> ;

Example: DESC emp;

or DESCRIBE emp;

Above statement will show you the complete structure of table named EMP as shown below:

Field	Type	Null	Key	Default	Extra
Eno	integer	YES		NULL	
Ename	char(20)	YES		NULL	
Job	char(20)	YES		NULL	
Mgr	integer	YES		NULL	
Hiredate	Date	YES		NULL	
Sal	decimal	YES		NULL	
Comm	decimal	YES		NULL	
Dno	integer	YES		NULL	

10 rows in set (0.09 sec)

✓ To view the tables in a database, SHOW TABLES command is used after opening the database as following:

Example: mysql> USE Sample_DB;

Database changed

mysql> SHOW TABLES;

Above statement give output as

Tables_in_Sample_DB
EMP
DEPT

2 rows in set (0.01 sec)

BACK

Making Simple Queries Continued :

7. Inserting Data Into Table:

A. INSERT INTO Command:

✓ INSERT INTO command is used to add rows to relations.

Syntax: INSERT INTO <tablename> [<column list>]
VALUES (<value>, <value> ….);

Example: INSERT INTO emp
VALUES (142, 'SAMIR', 'CLERK', 125, 21-MAY-81, 5000, 500, 20);

Above statement insert a row in to emp table. The same can be done with the alternative command as shown below:

Example: INSERT INTO emp (eno, ename, job, mgr, hiredate, sal, comm, dno)
VALUES (142, 'SAMIR', 'CLERK', 125, 21-MAY-81, 5000, 500, 20);

Note that the data values are in the same order as the column names in the table. Data can be added only to some columns in a row by specifying the columns and their data as shown below:

Example: INSERT INTO emp (eno, ename, job)
VALUES (199, 'TEENA', 'MANAGER',);

✓ In an INSERT statement, only those columns can be skipped that columns have either default value defined or they allow NULL values.

Continued

7. Inserting Data Into Table Continued:

B. Inserting NULL values:

✓ NULL value can be inserted in to a column by typing NULL without quotes.

Example:

```
INSERT INTO emp (eno, ename, job, mgr, hiredate, sal, comm, dno)
 VALUES (142, ‘SAMIR’,  NULL, 125,  21-MAY-81, 5000,  NULL,  20);
```

Here, for Mgr and Comm columns, NULL values have been inserted.

C. Inserting Data from Another Table:

✓ INSERT INTO can also be used to take values from one table and place them in another table.

Example:

```
INSERT INTO emp (eno, ename, sal)
 SELECT emp_no, ename, sal
 FROM Temp
 WHERE Months > 24 ;
```

It will extract all those rows from Temp that have months more than 24 and insert this produced results into the table emp.

To insert using a query, the following conditions must be true:

1. Both the tables must be already created.
2. The columns of the tables being inserted into, must match the columns output by the subquery.

BACK

8. Modifying Data with UPDATE Command:

- ✓ UPDATE command is used for change some or all of the values in exicting row.
- ✓ The UPDATE command specifies the rows to be changed using the WHERE clause, and the new data using the SET keyword.

Example:

```
UPDATE emp
 SET sal = 1500
 WHERE sal = 1100 ;
```

Above statement change *sal* to 1500 only for that employees that have *sal* as 100.

A. Updating Multiple Columns:

- ✓ To update multiple columns, multiple column assignment can be specified with SET clause, separated by commas.

Example:

```
UPDATE emp
 SET sal = 1500,  comm = 200
 WHERE eno < 122 ;
```

Above statement change *sal* to 1500 and *comm* to 200 for employees having *eno* less than 122.

B. Updating to NULL Values:

- ✓ The NULL values can also be entered just as other values by using UPDATE command.

Example:

```
UPDATE emp
 SET comm = NULL
 WHERE eno = 147 ;
```

Above statement insert a NULL values to column *comm* for employees that have *eno* as 147.

Continued

Making Simple Queries Continued :

8. Modifying Data with UPDATE Command Continued:

C. Using Expression in Update:

✓ Scalar expression can be used in the SET clause of the UPDATE command.

Example:
```
UPDATE emp
SET sal = sal * 2
WHERE ( eno = 115  OR  eno = 126 ) ;
```
Above statement double the salary of employees having *eno* 115 or 126.

9. Deleting Data with DELETE Command:

✓ The DELETE command removes rows from a table.

✓ Removes the entire rows, not individual field values.

✓ No field argument is needed or accepted.

Syntax:
```
DELETE FROM <tablename>
[ WHERE <predicate> ] ;
```
Example : DELETE FROM emp;

Above statement removes all the contents of *emp* table.

✓ Some specific rows can also be deleted by using condition as shown below:

Example :
```
DELETE FROM emp
 WHERE sal < 1500 ;
```
Above statement removes all the rows from *emp* table that have *sal* less than 1500.

BACK

Making Simple Queries Continued :

10. ALTER TABLE Command:

- ✓ The ALTER TABLE command is used to change definition of existing tables.
- ✓ ALTER TABLE is used:
    - To add a column
    - To add an integrity constraint
    - To redefine a column (datatype, size, default value).

A. Adding Columns:

- ✓ The new column will be added with NULL values for all rows currently in the table.
- ✓ Several new columns can be added, separated by commas, in a single command.
- ✓ It may be possible to drop or alter columns.

Syntax: ALTER TABLE <table name> ADD <column name> <data type><size> [<constraint name>] ;

Example:

```
ALTER TABLE emp
ADD ( Ph_no  integer) ;
```

Above statement add a new column *Ph_no* of type integer in table *emp*.

B. Changing a Column Name:

- ✓ For changing the name of the column CHANGE clause of ALTER TABLE command is used.

Syntax:

```
ALTER TABLE
CHANGE [ COLUMN ] old_col_name  new_col_name  column_defination ;
```

Example:

```
ALTER TABLE emp
CHANGE  Sal   Salary DECIMAL ;
```

Here, complete column description is given along with the new name.

Above statement change the existing column namely *Sal* of table *emp* to *Salary.*

Continued

10. ALTER TABLE Command Continued:

C. Modifying Column Definition:

By using the MODIFY clause we can change any of the following parts of a column definition:

- **Datatype:** We can change any column's datatype if all rows for the column contains nulls.
- **Default value:** We can decrease any column's size if all rows for the column contains nulls whereas we can always increase the size or the precision of a column.
- **Integrity Constraint:** By using the MODIFY clause we can add only NOT NULL constraint to an existing column.
- **Order of column:** With MODIFY clause we can reorder the columns within a table by using FIRST or AFTER<colname> clause.

Syntax:
```
ALTER TABLE <table name>
MODIFY (column name  newdatatype (newsize) ) [ FIRST | AFTER column] ;
```

Example:
```
ALTER TABLE emp
MODIFY ( Job char(30) ) ;
```

Above statement modify column *Job* of table *emp* to have new width of 30 characters.

```
ALTER TABLE dept
MODIFY Dname Char FIRST ;
```

Above statement reorder an existing column say *Dname* to be the first column in the table *Dept*.

D. Modifying Column Definition:

✓ To remove a component of a table, DROP clause of ALTER TABLE is used.

✓ Keywords mostly used with DROP clause of ALTER TABLE command are as following:

- **PRIMARY KEY –** Drops the table's primary key constraint.
- **COLUMN <columnname> –** Remove mentioned column from the table.
- **FOREIGN KEY<constraintname> –** Removes the mentioned foreign key constraint from the table.

Example:
```
ALTER TABLE dept
DROP  PRIMARY  KEY ,  DROP  FOREIGN KEY fk_1, DROP COLUMN dno;
```

Above statement removes the primary key, the foreign key constraint namely fk_1, and the column namely *dno* from the *Dept* table.

BACK

Making Simple Queries Continued :

11. The DROP TABLE Command:

✓ The DROP TABLE command is used for drop a table from the database.

Syntax: DROP TABLE [IF EXISTS] <tablename> ;

Example: DROP TABLE dept;

Above statement drop the table namely *Dept* from the database.

✓ The IF EXISTS clause first checks whether the given table exists in the database or not.

Example: DROP TABLE IF EXISTS emp;

Above statement first check for existence of *emp* table in current database and then drop the table namely *emp* from the database.

12. The DROP VIEW Command:

✓ Used for delete a view from database.

✓ When view is dropped it does not cause any change in its base table.

Example: DROP VIEW taxpayee ;

Above statement drops the view *taxpayee* from the database.

BACK

Making Simple Queries Continued :

13. The SELECT Command:

The SELECT statement can be used to retrieve a subset of rows or columns from one or more tables.

Syntax: SELECT <column name> [, <column name>, …]
FROM <table name> ;

Example : SELECT ename, job, sal
FROM emp;

Above statement display the information of columns *ename*, *job* and *sal* as shown below:

Ename	Job	Sal
RAJ	MANAGER	2000
SONALI	CLERK	950
ANKIT	SALSEMAN	1250
SIYA	CLERK	1100
RADHIKA	ANALYST	3200
PAYAL	MANAGER	2450
MANOJ	PRESIDENT	5000
KRUPA	SALSEMAN	1600
SAGAR	CLERK	800
NEHA	MANAGER	2975

10 rows in a set (0.00 sec)

14. Reordering Columns in Query Results:

While giving a querying, the result can be obtained in any order.

Example : SELECT job, ename, sal, hiredate
FROM emp;

Above statement display *job* as first column, *ename* as second column, *sal* as third column and *hiredate* as forth column as shown below:

Job	Ename	Sal	Hiredate
MANAGER	RAJ	2000	17-NOV-81
CLERK	SONALI	950	09-JUN-81
SALSEMAN	ANKIT	1250	07-APR-81
CLERK	SIYA	1100	21-SEP-81
ANALYST	RADHIKA	3200	20-FEB-81
MANAGER	PAYAL	2450	03-DEC-81
PRESIDENT	MANOJ	5000	22-FEB-81
SALSEMAN	KRUPA	1600	17-DEC-80
CLERK	SAGAR	800	12-JAN-83
MANAGER	NEHA	156	23-JAN-82

10 rows in a set (0.02 sec)

BACK

15. Selecting All Columns:

To select all the columns from the table the asterisk (*) can be used.

Example : SELECT * FROM emp;

Above statement display all the rows present in the *emp* table as shown below:

Eno	Ename	Job	Mgr	Hiredate	Sal	Comm	Dno
115	RAJ	MANAGER	156	17-NOV-81	2000		10
121	SONALI	CLERK	115	09-JUN-81	950		30
196	ANKIT	SALSEMAN	147	07-APR-81	1250	300	10
134	SIYA	CLERK	178	21-SEP-81	1100		20
182	RADHIKA	ANALYST	178	20-FEB-81	3200		30
147	PAYAL	MANAGER	156	03-DEC-81	2450		30
156	MANOJ	PRESIDENT		22-FEB-81	5000		10
123	KRUPA	SALSEMAN	147	17-DEC-80	1600	1200	20
195	SAGAR	CLERK	115	12-JAN-83	800		NULL
178	NEHA	MANAGER	156	23-JAN-82	2975		20

10 rows in a set (0.02 sec)

BACK

Making Simple Queries Continued :

16. Eliminating Redundant Data - DISTINCT keyword :

- ✓ The DISTINCT keyword eliminates duplicate rows from the results of a SELECT statement.
- ✓ Only one NULL value is returned in the results with DISTINCT keyword.
- ✓ DISTINCT applies to entire output row, not a specific field.
- ✓ The DISTINCT keyword can be specified only once in a given SELECT clause.

Example : SELECT DISTINCT job
FROM emp;

Above statement display *job* column as shown below:

Job
MANAGER
CLERK
SALSEMAN
ANALYST
PRESIDENT

Here, all the duplicate entry has not reappeared. This is the property of DISTINCT.

5 rows in a set (0.03 sec)

17. Selecting From All the Rows – ALL keyword:

- ✓ With ALL keyword the result retains the duplicate output rows.

Example : SELECT ALL job
FROM emp;

Above statement display values of *job* column from every row of the table without considering the duplicate entries as shown below:

Job
MANAGER
CLERK
SALSEMAN
CLERK
ANALYST
MANAGER
PRESIDENT
SALSEMAN
CLERK
MANAGER

Here, all the duplicate entry has reappeared. This is the property of ALL.

10 rows in a set (0.00 sec)

BACK

18. How to Perform Simple Calculations ?

✓ The SELECT statement is used to perform simple calculations and to retrieve rows computed without reference to any table.

Example : SELECT 1 + 6 ;

Above statement computes the expression 1 + 6 as shown below:

1 + 6
7

The result of expression given with SELECT.

1 rows in a set (0.00 sec)

✓ To perform simple calculations we can also use table DUAL, which is one row, one column dummy table.

Example : SELECT 4 * 3 FROM dual ;

Above statement computes the expression 4 * 3 as shown below:

4 * 3
12

The result of expression given with DUAL.

1 rows in a set (0.00 sec)

✓ To obtain a current date curdate( ) function is used.

Example : SELECT curdate( ) ;

Above statement returns the current system date as shown below:

curdate ( )
2013-04-05

Current date of the system.

1 rows in a set (0.00 sec)

BACK

Making Simple Queries Continued :

19. Scalar Expression with Selected Fields:

- ✓ In SQL we can place scalar expression and constant among the selected fields.
- ✓ If value in the expression is NULL then result of expression will be NULL only.
- ✓ The operators used in arithmetic expressions are plus (+), minus (-), divide (/), multiply(*), and modulo (%).

Example :

```
SELECT  ename, sal+ 20
FROM  emp ;
```

Above statement add 20 to every row of column *sal* as shown below:

Ename	Sal+ 20
RAJ	2020
SONALI	970
ANKIT	1270
SIYA	1120
RADHIKA	3220
PAYAL	2470
MANOJ	5020
KRUPA	1620
SAGAR	820
NEHA	2995

10 rows in a set (0.02 sec)

BACK

20. Using Column Aliases:

- ✓ Using column alias name we can give a different name to a columns for output purposes.
- ✓ If alias name contains more than one word, than we have to enclose them in quotes.

Syntax:

```
SELECT <columnname> AS  [columnalias]  [, <columnname>  AS  [columnalias] ] …….
FROM  <tablename> ;
```

Example :

```
SELECT  ename, job AS  “Emp Post”
FROM  emp ;
```

Above statement give column *Job* a alias name Designation as shown below:

Ename	Emp Post
RAJ	MANAGER
SONALI	CLERK
ANKIT	SALSEMAN
SIYA	CLERK
RADHIKA	ANALYST
PAYAL	MANAGER
MANOJ	PRESIDENT
KRUPA	SALSEMAN
SAGAR	CLERK
NEHA	MANAGER

10 rows in a set (0.02 sec)

- ✓ We can provide alias to an expression as shown below:

Example : SELECT 22/7 as PI;

Output:

PI
3.1429

1 rows in a set (0.02 sec)

BACK

Making Simple Queries Continued :

21. Handling Nulls:

- ✓ When we list a column having null values, only non-null values are displayed.
- ✓ If we want to replace null with a value in the output, we can use IFNULL( ) function.
- ✓ The value being replaced for Null values through IFNULL( ), should be of the same type as that of the column.

Syntax: IFNULL (<columnname>, value-to-be-substituted)

Example :

```
SELECT  eno, ename, IFNULL (comm, "Zero")
FROM  emp ;
```

Above statement substitute value zero for null in *comm* as shown below:

Eno	Ename	IFNULL (comm, "Zero")
115	RAJ	Zero
121	SONALI	Zero
196	ANKIT	300
134	SIYA	Zero
182	RADHIKA	Zero
147	PAYAL	Zero
156	MANOJ	Zero
123	KRUPA	1200
195	SAGAR	Zero
178	NEHA	Zero

10 rows in a set (0.02 sec)

- ✓ We can also use column alias to replace output heading.

Example :

```
SELECT  eno, ename, IFNULL (comm, "Zero")
AS  "Commission Earned"
FROM  emp ;
```

Above statement give column *Comm* a alias name "Commission Earned" as shown below:

Eno	Ename	Commission Earned
115	RAJ	Zero
121	SONALI	Zero
196	ANKIT	300
134	SIYA	Zero
182	RADHIKA	Zero
147	PAYAL	Zero
156	MANOJ	Zero
123	KRUPA	1200
195	SAGAR	Zero
178	NEHA	Zero

10 rows in a set (0.02 sec)

BACK

Making Simple Queries Continued :

22. Putting Text in the Query Output:

✓ SQL enables us to putting some items as symbols and comment in the output.

Example :

SELECT ename, sal+ 20, '/-'
FROM emp ;

Above statement print the same symbol in every row as shown below:

Ename	Sal+ 20	/-
RAJ	2020	/-
SONALI	970	/-
ANKIT	1270	/-
SIYA	1120	/-
RADHIKA	3220	/-
PAYAL	2470	/-
MANOJ	5020	/-
KRUPA	1620	/-
SAGAR	820	/-
NEHA	2995	/-

10 rows in a set (0.04 sec)

✓ We can insert text in our query also, to making it more presentable.

Example :

SELECT ename, 'gets the salary', sal+ 20, '/-'
FROM emp ;

Above statement give column *Comm* a alias name "Commission Earned" as shown below:

Ename	gets the salary	Sal+ 20	/-
RAJ	gets the salary	2020	/-
SONALI	gets the salary	970	/-
ANKIT	gets the salary	1270	/-
SIYA	gets the salary	1120	/-
RADHIKA	gets the salary	3220	/-
PAYAL	gets the salary	2470	/-
MANOJ	gets the salary	5020	/-
KRUPA	gets the salary	1620	/-
SAGAR	gets the salary	820	/-
NEHA	gets the salary	2995	/-

10 rows in a set (0.02 sec)

BACK

Making Simple Queries Continued :

23. Selecting Specific Rows – WHERE clause:

- ✓ The WHERE clause in SELECT statement specifies the criteria for selection of rows to be returned.
- ✓ With the use of WHERE clause, only those rows that satisfied the given condition are displayed in the output.

Syntax: SELECT <column name> [, <column name>, …..]
FROM <table name>
WHERE <condition> ;

Example : SELECT eno, ename, sal
FROM emp
WHERE sal < 1200 ;

Above statement display the *eno*, *ename* and *sal* for employees having salary less than 1200 and outputs the result as shown below:

Eno	Ename	Sal
121	SONALI	950
134	SIYA	1100
195	SAGAR	800

3 rows in a set (0.02 sec)

24. Relational Operator:

- ✓ Relational operator is used to compare two values.
- ✓ Result of comparison is true or false.
- ✓ The SQL recognizes following relational operators:

  =, >, <, >=, <=, <> (not equal to)
- ✓ In CHARACTER data type comparison, < means earlier in the alphabet and > means later in the alphabet.
- ✓ Apostrophes are necessary around all CHAR, DATE and TIME data.

Example : SELECT eno, ename, job
FROM emp
WHERE job <> 'CLERK';

Above statement display the *eno*, *ename* and *sal* for employees having *job* as clerk and outputs the result as shown below:

Eno	Ename	Job
121	SONALI	CLERK
134	SIYA	CLERK
195	SAGAR	CLERK

3 rows in a set (0.02 sec)

BACK

Making Simple Queries Continued :

25. Logical Operators:

- ✓ The logical operators OR (||), AND (&&), and NOT (!) are used to connect search conditions in the WHERE clause.
- ✓ When all the logical operators are used together, the order of precedence is NOT (!), AND (&&), and OR(||).

Example :

1. To list the employee's details having *Job* as 'CLERK' or 'MANAGER' from table *emp*, logical operator OR will be used as:

```
SELECT eno, ename, job, sal
FROM emp
WHERE ( job = 'CLERK'   OR   job = 'MANAGER' );
```

Output:

Eno	Ename	Job	Sal
121	SONALI	CLERK	950
134	SIYA	CLERK	1100
195	SAGAR	CLERK	800

3 rows in a set (0.01 sec)

2. To list the employee's details having *Job* as 'CLERK' but salary < 1000 from table *emp*, logical operator AND will be used as:

```
SELECT eno, ename, job, sal
FROM emp
WHERE ( job = 'CLERK'   AND   sal < 1000);
```

Output:

Eno	Ename	Job	Sal
121	SONALI	CLERK	950
195	SAGAR	CLERK	800

2 rows in a set (0.01 sec)

3. To list the employee's details having *Job* is other than 'CLERK' from table *emp*, logical operator NOT will be used as:

```
SELECT eno, ename, job, sal
FROM emp
WHERE ( NOT job = 'CLERK' );
```

Output:

Eno	Ename	Job	Sal
115	RAJ	MANAGER	2000
196	ANKIT	SALSEMAN	1250
182	RADHIKA	ANALYST	3200
147	PAYAL	MANAGER	2450
156	MANOJ	PRESIDENT	5000
123	KRUPA	SALSEMAN	1600
178	NEHA	MANAGER	2975

7 rows in a set (0.01 sec)

BACK

Making Simple Queries Continued :

26. Condition Based on a Range:

- ✓ The Between operator defines a range of values that the column must fall in to make the condition true.
- ✓ The range includes both lower and the upper value.

Example :
```
SELECT  eno, ename, sal
FROM  emp
WHERE sal BETWEEN 3000 AND 5000  ;
```

Above statement display the employees having salary between 3000 to 5000 as shown below:

Eno	Ename	Sal
182	RADHIKA	3200
156	MANOJ	5000

2 rows in a set (0.02 sec)

- ✓ The operator NOT BETWEEN retrieves the rows, which are not satisfying the BETWEEN condition.

Example :
```
SELECT  eno, ename, sal
FROM  emp
WHERE sal NOT BETWEEN 900 AND 5000  ;
```

Above statement display the employees having salary below 900 or above 5000 as shown below:

Eno	Ename	Sal
156	MANOJ	5000
195	SAGAR	800

2 rows in a set (0.00 sec)

27. Condition Based on a List:

- ✓ To specify a list of values, IN operator is used.
- ✓ The IN operator selects values that match any value in a given list of values.

Example :
```
SELECT  eno, ename, job  FROM  emp
WHERE job IN('CLERK', 'MANAGER',  'ANALYST');
```

Above statement display a list of employees from 'PRESIDENT', 'MANAGER' or 'ANALYST' as shown below:

Eno	Ename	Job
115	RAJ	MANAGER
182	RADHIKA	ANALYST
147	PAYAL	MANAGER
156	MANOJ	PRESIDENT
178	NEHA	MANAGER

5 rows in a set (0.03 sec)

- ✓ The NOT IN operator finds rows that do not match in the list.

Example :
```
SELECT  eno, ename, job  FROM  emp
WHERE job IN('CLERK','MANAGER', 'CLERK', 'SALESMAN');
```

Above statement list employees which have not *job* as mentioned in the list.

Eno	Ename	Job
182	RADHIKA	ANALYST
156	MANOJ	PRESIDENT

BACK

28. Condition Based on Pattern Matches:

- ✓ In SQL string-matching operator LIKE is used for comparison on character strings using patterns.
- ✓ Patterns are described using two special wildcard characters as following:
  1. Percent (%) – The % character matches any substring.
  2. Underscore (_) – The _ character matches any substring.
- ✓ Patterns are case-sensitive. For example,
  - ‘Ran%’ matches any string beginning with ‘Ran’.
  - ‘%idge%’ matches any string containing ‘idge’ as a substring.
  - ‘- - - -’ matches any string of exactly 4 characters.
  - ‘- - - %’ matches any string of at least 3 characters.

Example :

```
SELECT  eno, ename, mgr
FROM  emp
WHERE mgr LIKE  ’15%’;
```

Above statement list employees which have mgr starting with 15 as shown below:

Eno	Ename	Mgr
115	RAJ	156
147	PAYAL	156
178	NEHA	156

3 rows in a set (0.01 sec)

- ✓ The keyword NOT LIKE is used to select rows that do not match the specified pattern of characters.
- ✓ To include the special pattern character (i.e., %, _) in a pattern an escape character is used.
- ✓ The escape character is used immediately before a special pattern character.
- ✓ For a LIKE comparison the escape character is define using the ESCAPE keyword. For example,
  - LIKE ‘wx\%yz%’ ESCAPE’\’ matches all strings beginning with ‘wx%yz’.
  - LIKE ‘wx\\yz%’ ESCAPE’/’ matches all string beginning with ‘wx\yz’.
- ✓ In above example backslash (‘\’) is used as the escape character.

BACK

Making Simple Queries Continued :

29. Searching for NULL:

- ✓ The NULL value in a column can be searched for in a table using IS NULL in a WHERE clause.
- ✓ Non-NULL values can be listed using IS NOT NULL.

Example :

```
SELECT  eno, ename, job
FROM  emp
WHERE dno IS NULL ;
```

Above statement list employees which have dno as NULL as shown below:

Eno	Ename	Job
195	SAGAR	CLERK

```
1  rows in a set  (0.00  sec)
```

30. Operator Precedence:

- ✓ When expression has multiple operators, then the evaluation of expression takes place in the order of operator precedence.
- ✓ The operator with higher precedence are evaluated first.
- ✓ Operator precedence of MySQL are shown in the following list, from highest precedence to the lowest:

!
- (unary minus) ~ (unary bit inversion)
^
*, /, DIV, %, MOD
-, +
< >
&
|
==, >=, >, <=, <, <>, !=, IS, LIKE, REGEXP, IN
BETWEEN, CASE, WHEN, THEN, ELSE
NOT
&&, AND
XOR
||, OR
:=

BACK

Making Simple Queries Continued :

31. Sorting Results – ORDER BY clause:

- ✓ The ORDER BY clause allows sorting of query results by one or more columns.
- ✓ The sorting can be done either in ascending or descending order, the default order is ascending.

Syntax:
```
SELECT <column name> [, <column name> , … ]
FROM  <table name>
[WHERE  <predicate> ]
[ ORDER BY <column name> ];
```

Example :
```
SELECT  eno, ename, job
FROM  emp
ORDER BY ename;
```

Above statement list employee's detail in alphabetical order of their names as shown below:

Eno	Ename	Job
196	ANKIT	SALSEMAN
123	KRUPA	SALSEMAN
156	MANOJ	PRESIDENT
178	NEHA	MANAGER
147	PAYAL	MANAGER
182	RADHIKA	ANALYST
115	RAJ	MANAGER
195	SAGAR	CLERK
134	SIYA	CLERK
121	SONALI	CLERK

10 rows in a set (0.04 sec)

- ✓ Descending order is specified by DESC.
- ✓ Ascending order is specified by ASC.
- ✓ Ordering can be performed on multiple attributes, separated by commas.

Example:
```
SELECT  eno, ename, job
FROM  emp
ORDER BY job DESC, ename ASC;
```

Above statement list employee's detail in descending order of their of job and ascending order of name as shown below:

Eno	Ename	Job
196	ANKIT	SALSEMAN
123	KRUPA	SALSEMAN
156	MANOJ	PRESIDENT
178	NEHA	MANAGER
147	PAYAL	MANAGER
115	RAJ	MANAGER
195	SAGAR	CLERK
134	SIYA	CLERK
121	SONALI	CLERK
182	RADHIKA	ANALYST

10 rows in a set (0.04 sec)

BACK

32. Sorting by Column Alias:

✓ Column alias can be used for sorting records because once a column alias is declared, it can be used elsewhere in the statement.

Example :

```
SELECT  eno, ename, job “Emp Post”
FROM  emp
ORDER BY “Emp Post”;
```

Above statement list employee’s detail in alphabetical order of their jobs by using column alias as shown below:

Ename	Emp Post
RADHIKA	ANALYST
SONALI	CLERK
SIYA	CLERK
SAGAR	CLERK
RAJ	MANAGER
PAYAL	MANAGER
NEHA	MANAGER
MANOJ	PRESIDENT
ANKIT	SALSEMAN
KRUPA	SALSEMAN

10 rows in a set (0.03 sec)

BACK

More Presentation are available on
www.cbsecsnip.in for XI and XII
computer science and informatics
practices.
Thank you