MYSQL REVISION TOUR

Lesson Plan -1

Step 1: Learning Objectives:

To enable students to

- Understand the basic terminology of Database system
- Create their own database and table.
- Apply various SQL commands for solving queries

Step 2: Introduction:

- MYSQL - A freely available open source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL).
- SQL (Structured Query Language) - A language that enables you to create and operate on relational databases, which are sets of related information stored in tables.
- DIFFERENT DATA MODELS
  - A **data model refers to a set of concepts to describe the structure of a database**, and certain constraints (restrictions) that the database should obey. The four data models that are used for database management are :
    1. **Hierarchical data model**
    2. **Relational data model**
    3. **Network data model**
    4. **Object Oriented data model**

RELATIONAL MODEL TERMINOLOGY

1. **Relation :** A table storing logically related data is called a Relation.
2. **Tuple :** A **row of a relation** is generally referred to as a tuple.
3. **Attribute :** A **column** of a relation is generally referred to as an attribute/Field.
4. **Degree :** This refers to the **number of attributes** in a relation.
5. **Cardinality :** This refers to the **number of tuples** in a relation.
6. **Primary Key :** This refers to a set of one or more attributes that can uniquely identify tuples within the relation.
7. **Candidate Key :** All attribute combinations inside a relation that can serve as primary key are candidate keys(as these are candidates for primary key position).
8. **Alternate Key :** A candidate key that is not primary key, is called an alternate key.
9. **Foreign Key :** A non-key attribute, whose values are derived from the primary key of some other table, is known as foreign key in its current table ie. Primary Key of one table when used in another table for reference.

REFERENTIAL INTEGRITY

- A referential integrity is a system of rules that a DBMS uses to ensure that relationships between records in related tables are valid, and that users don't accidentally delete or change related data. This integrity is ensured by foreign key.

CLASSIFICATION OF SQL STATEMENTS

SQL commands can be mainly divided into following categories:

1. **Data Definition Language(DDL) Commands**
   Commands that allow you to perform task, related to data definition e.g;
- Creating, altering and dropping.
- Granting and revoking privileges and roles.
- Maintenance commands.

2. **Data Manipulation Language(DML) Commands**
   Commands that allow you to perform data manipulation e.g., retrieval, insertion, deletion and modification of data stored in a database.

3. **Transaction Control Language(TCL) Commands (Grant and Revoke)**
   Commands that allow you to manage and control the transactions e.g.,
- Making changes to database, permanent
- Undoing changes to database, permanent
- Creating savepoints
- Setting properties for current transactions.

DATA TYPES

Numeric Data Type

1. int – used for number without decimal.
2. Decimal(m,d) – used for floating/real numbers. m denotes the total length of number and d is number of decimal digits.

Date and Time Data Type

1. date – used to store date in YYYY-MM-DD format.
2. time – used to store time in HH:MM:SS format.

String Data Types

1. char(m) – used to store a fixed length string. **m** denotes max. number of characters.
2. varchar(m) – used to store a variable length string. **m** denotes max. no. of characters.

DIFFERENCE BETWEEN CHAR AND VARCHAR DATA TYPE

S.NO	Char Datatype	Varchar Datatype
1.	It specifies a **fixed length** character String.	It specifies a **variable length** character string.
2.	When a column is given datatype as CHAR(**n**), then MySQL ensures that all values stored in that column have this length i.e. **n** bytes. If a value is shorter than this length **n** then blanks are added, but the size of value remains **n** bytes.	When a column is given datatype as VARCHAR(**n**), then the maximum size a value in this column can have is **n** bytes. Each value that is stored in this column store exactly as you specify it i.e. no blanks are added if the length is shorter than maximum length **n**.

NULL VALUE

If a column in a row has no value, then column is said to be **null** , or to contain a null. **You should use a null value** when the actual value is not known or when a value would not be meaningful.

DATABASE COMMANDS

1. VIEW EXISTING DATABASE

To view existing database names, the command is: **SHOW DATABASES ;**

2. CREATING DATABASE IN MYSQL

For creating the database in MySQL, we write the following command : **CREATE DATABASE** <databasename> ;

e.g. In order to create a database Student, command is : **CREATE DATABASE** Student ;

3. ACCESSING DATABASE

For accessing already existing database , we write :

USE <databasename> ;

e.g. to access a database named Student, we write command as : **USE** Student ;

4. DELETING DATABASE

For deleting any existing database , the command is :

DROP DATABASE <databasename> ;

e.g. to delete a database , say student, we write command as : **DROP DATABASE** Student ;

5. VIEWING TABLE IN DATABASE

In order to view tables present in currently accessed database , command is : **SHOW TABLES ;**

CREATING TABLES IN MYSQL

- Tables are created with the CREATE TABLE command. When a table is created, its columns are named, data types and sizes are supplied for each column.

Syntax of CREATE TABLE command is :

CREATE TABLE <table-name>(<column name> <data type> ,<column name> <data type> ,……) ;

E.g. in order to create table EMPLOYEE given below :

ECODE	ENAME	GENDER	GRADE	GROSS
Integer Primary Key	Varchar of length 20 Not NULL	Char of length 1	Char of length 2	Integer

We write the following command:

CREATE TABLE employee (ECODE integer Primary Key , ENAME varchar(20) Not Null , GENDER char(1) , GRADE char(2) , GROSS integer) ;

INSERTING DATA INTO TABLE

The rows are added to relations(table) using INSERT command of SQL.

Syntax of INSERT is:

INSERT INTO <tablename> [<column list>] VALUES (<value1> , <value2> , …..) ;

e.g. to enter a row into EMPLOYEE table (created above), we write command as :

INSERT INTO employee VALUES(1001 , 'Ravi' , 'M' , 'E4' , 50000);

OR

INSERT INTO employee (ECODE , ENAME , GENDER , GRADE , GROSS) VALUES(1001 , 'Ravi' , 'M' , 'E4' , 50000);

Output: The Data gets added in the table.

ECODE	ENAME	GENDER	GRADE	GROSS
1001	Ravi	M	E4	50000

In order to insert another row in EMPLOYEE table, we write again INSERT command :

INSERT INTO employee VALUES(1002 , 'Akash' , 'M' , 'A1' , 35000);

ECODE	ENAME	GENDER	GRADE	GROSS
1001	Ravi	M	E4	50000

1002	Akash	M	A1	35000

INSERTING NULL VALUES

To insert value NULL in a specific column, we can type NULL without quotes and NULL will be inserted in that column.

E.g. in order to insert NULL value in GRADE column of above table, we write INSERT command as :

INSERT INTO EMPLOYEE VALUES (1004 , 'Aman' , 'M' , NULL , 38965) ;

ECODE	ENAME	GENDER	GRADE	GROSS
1001	Ravi	M	E4	50000
1002	Akash	M	A1	35000
1004	Aman	M	NULL	38965

SIMPLE QUERIES USING SELECT COMMAND

The SELECT command is used to extract and display information from a table.

Syntax of SELECT command is :

SELECT <column name>,<column name> FROM <tablename> WHERE <condition name> ;

SELECTING ALL DATA

In order to retrieve everything (all columns) from a table, SELECT command is used as :

SELECT * FROM <tablename> ;

e.g. In order to retrieve everything from **Employee** table, we write SELECT command as :

SELECT * FROM Employee ;

EMPLOYEE

ECODE	ENAME	GENDER	GRADE	GROSS
1001	Ravi	M	E4	50000
1002	Akash	M	A1	35000
1004	Aman	M	NULL	38965

SELECTING PARTICULAR COLUMNS

A particular column from a table can be selected by specifying column-names with SELECT command.

E.g. From the table given below, if we want to select ECODE and ENAME column, then command is :

EMPLOYEE

ECODE	ENAME	GENDER	GRADE	GROSS
1001	Ravi	M	E4	50000
1002	Akash	M	A1	35000
1004	Neela	F	B2	38965
1005	Sunny	M	A2	30000
1006	Ruby	F	A1	45000
1009	Neema	F	A2	52000

SELECT ECODE , ENAME **FROM** EMPLOYEE ;

E.g.2 To display only ENAME, GRADE and GROSS column, the command is :

SELECT ENAME , GRADE , GROSS **FROM** EMPLOYEE ;

SELECTING PARTICULAR ROWS

We can select particular rows from a table by specifying a condition through **WHERE clause** along with SELECT statement.

E.g. To display the records of the female employees, then command is :

SELECT * FROM **EMPLOYEE** WHERE **GENDER = 'F' ;**

E.g.2. To display the records from Employee where salary is greater than 48000, then command is :

SELECT * FROM **EMPLOYEE** WHERE **GROSS > 48000 ;**

ELIMINATING REDUNDANT DATA

The **DISTINCT** keyword eliminates duplicate rows from the results of a SELECT statement.

For example, Statement 1 given below will display all the data in the Gender column where as Statement 2 will display the unique values in the Gender column.

Statement 1: SELECT GENDER **FROM** EMPLOYEE;

GENDER
M
M
F
M
F
F

Statement 2: SELECT DISTINCT(GENDER) **FROM** EMPLOYEE;

DISTINCT(GENDER)
M
F

VIEWING STRUCTURE OF A TABLE

If we want to know the structure of a table, we can use DESCRIBE or DESC command, as per following syntax :

DESCRIBE | **DESC** <tablename> ;
e.g. to view the structure of table **EMPLOYEE**, command is :

DESCRIBE EMPLOYEE ; **OR DESC** EMPLOYEE ;

USING COLUMN ALIASES

The columns that we select in a query can be given a different name, i.e. column alias name for output purpose with the help of the keyword AS..
Syntax :

SELECT <columnname> **AS** column alias , <columnname> **AS** column alias ….. FROM <tablename> ;

e.g. In output, suppose we want to display ECODE column as EMPLOYEE_CODE in output , then command is :

SELECT **ECODE AS "EMPLOYEE_CODE"** FROM **EMPLOYEE ;**

CONDITION BASED ON A RANGE

The **BETWEEN** operator defines a range of values that the column values must fall in to make the condition true. The range include both lower value and upper value.

e.g. To display ECODE, ENAME and GRADE of those employees whose salary is between 40000 and 50000, the command is:

SELECT ECODE , ENAME ,GRADE FROM **EMPLOYEE** WHERE GROSS **BETWEEN** 40000 AND 50000 ;

Output will be :

ECODE	ENAME	GRADE
1001	Ravi	E4
1006	Ruby	A1

CONDITION BASED ON A LIST

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

Eg. To display the details of the employees who have grade either as A1or A2.
Ans:
SELECT * FROM EMPLOYEE WHERE GRADE **IN** ('A1' , 'A2');

Output will be :

ECODE	ENAME	GENDER	GRADE	GROSS
1002	Akash	M	A1	35000
1006	Ruby	F	A1	45000
1005	Sunny	M	A2	30000
1009	Neema	F	A2	52000

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

E.g.
SELECT * FROM EMPLOYEE
WHERE GRADE **NOT IN** ('A1' , 'A2');

Output will be :

ECODE	ENAME	GENDER	GRADE	GROSS
1001	Ravi	M	E4	50000
1004	Neela	F	B2	38965

CONDITION BASED ON PATTERN MATCHES

LIKE operator is used for pattern matching in SQL. Patterns are described using two special wildcard characters:

1. percent(%) – The % character matches any substring.
2. underscore(_) – The _ character matches any character.

e.g. to display names of employee whose name starts with R in EMPLOYEE table, the command is :

SELECT ENAME FROM **EMPLOYEE** WHERE ENAME **LIKE 'R%' ;**

Output will be:

ENAME
Ravi
Ruby

e.g. to display details of employee whose second character in name is 'e'.

SELECT * FROM **EMPLOYEE** WHERE **ENAME LIKE '_e%' ;**

Output will be :

ECODE	ENAME	GENDER	GRADE	GROSS
1004	Neela	F	B2	38965
1009	Neema	F	A2	52000

e.g. to display details of employee whose name ends with 'y'.

SELECT * FROM **EMPLOYEE** WHERE **ENAME LIKE '%y' ;**

Output will be :

ECODE	ENAME	GENDER	GRADE	GROSS
1005	Sunny	M	A2	30000
1006	Ruby	F	A1	45000

SEARCHING FOR NULL

The NULL value in a column can be searched for in a table using IS NULL in the WHERE clause.

E.g. to list employee details whose salary contain NULL, we use the command :

SELECT ***** FROM **EMPLOYEE** WHERE GROSS **IS NULL ;**

Output:

Roll_No	Name	Marks
1	ARUN	NULL
2	RAVI	56
4	SANJAY	NULL

E.g. to display the names of those students whose marks is NULL, we use the command:

SELECT **Name** FROM **EMPLOYEE** WHERE Marks **IS NULL ;**

Output will be :

Name
ARUN
SANJAY

Step 3: SQL Assignment -1

Instruction: The Assignment should be done in the registers.

Q1. Define the terms:

(i) Database Abstraction
(ii) Data inconsistency
(iii) Conceptual level of database implementation/abstraction
(iv) Primary Key
(v) Candidate Key
(vi) Relational Algebra
(vii) Domain

Q2. Answer the following questions :

1. Differentiate between DDL and DML?
2. What is a constraint?
3. What are single row functions?
4. Compare CHAR and VARCHAR data types.
5. What are the differences between DELETE and DROP commands of SQL?
6. A table "Animals" in a database has 3 columns and 10 records. What is the degree and cardinality of this table?
7. What is the difference between commit and rollback command?
8. Which keyword is used to remove redundant data from a relation?
9. Observe the following table and answer the parts(i) and(ii) accordingly

Table:Product

Pno	Name	Qty	PurchaseDate
101	Pen	102	12-12-2011
102	Pencil	201	21-02-2013
103	Eraser	90	09-08-2010
109	Sharpener	90	31-08-2012
113	Clips	900	12-12-2011

(i) Write the names of most appropriate columns, which can be considered as candidate keys.

(ii) What is the degree and cardinality of the above table?

10. What is a Primary Key?
11. What is a Foreign Key? What is its use?
12. What are the various Integrity Constraints used in SQL?