**4.2 Data Definition language**

**4.2.1. Domain Type in SQL**

* Character (String) -- fixed or variable length character strings. The character set is implementation defined but often defaults to ASCII.
* Numeric -- values representing numeric quantities. Numeric values are divided into these two broad categories:
  + Exact (also known as *fixed-point*) -- Exact numeric values have a fixed number of digits to the left of the decimal point and a fixed number of digits to the right (the scale). The total number of digits on both sides of the decimal are the precision. A special subset of exact numeric types with a scale of 0 is called *integer*.
  + Approximate (also known as *floating-point*) -- Approximate numeric values that have a fixed precision (number of digits) but a *floating* decimal point.

All numeric types are signed.

* Datetime -- Datetime values include calendar and clock values (Date, Time, Timestamp) and intervals. The datetime types are:
  + Date -- calendar date with year, month and day
  + Time -- clock time with hour, minute, second and fraction of second, plus a timezone component (adjustment in hours, minutes)
  + Timestamp -- combination calendar date and clock time with year, month, day, hour, minute, second and fraction of second, plus a timezone component (adjustment in hours, minutes)
  + Interval -- intervals represent time and date intervals. They are signed. An interval value can contain a subset of the interval fields, for example - hour to minute, year, day to second. Interval types are subdivided into:
    - year-month intervals -- may contain years, months or combination years/months value.
    - day-time intervals -- days, hours, minutes, seconds, fractions of second.

**4.2.2. Scheme Definition in SQL.**

A schema is a collection of database objects (as far as this hour is concerned—tables) associated with one particular database username. This username is called the schema owner, or the owner of the related group of objects. You may have one or multiple schemas in a database. Basically, any user who creates an object has just created his or her own schema. So, based on a user's privileges within the database, the user has control over objects that are created, manipulated, and deleted. A schema can consist of a single table and has no limits to the number of objects that it may contain, unless restricted by a specific database implementation.

**4.3. Data Manipulation Language**

**4.3.1 The Select Clause**

SQL **SELECT** statement is used to fetch the data from a database table which returns data in the form of result table. These result tables are called result-sets.

Syntax:

SELECT column1, column2, columnN FROM table\_name;

Here only the records of column1, column2,… are displayed. If we want to fetch all the fields of the table, the we can use following syntax:

SELECT \*FROM table\_name;

Example:

SELECT Id, Name, Salary FROM Customers;

This statement displays the records of Id, Name, Salary columns of the Customers table.

**4.3.2 The Where Clause**

The WHERE clause is used to specify the conditions while fetching data from table. If the given condition is satisfied then only it returns specific value from the table.

Example:

SELECT Id, Name FROM Customers

WHERE address=’Bhaktapur’;

This statement will display the records of Id and Name columns of the Customers table whose address is Bhaktapur.

**4.3.3 The From Clause**

The SQL FROM clause is used to list the tables and any joins required for the SQL statement.

Syntax:

FROM table\_name;

Example:

SELECT Id, Name FROM Customers

This statement displays the Id and Name columns of the Customers table.

**4.3.4 The Rename Operation**

RENAME statement is used to rename a table.

Syntax:

RENAME TABLE {tbl\_name} TO {new\_tbl\_name};

**Where {tbl\_name} table that exists in the current database, and {new\_tbl\_name} is new table name.**

**4.3.5 Tuple Variable**

1. Tuple variables can be used in SQL, and are defined in the **from** clause:

**select distinct** *cname, T.loan#*

**from** *borrower* ***as*** *S, loan* ***as*** *T*

**where** *S.loan# = T.loan#*

Note: The keyword **as** is optional here.

1. These variables can then be used throughout the expression. Think of it as being something like the rename operator.

Finds the names of all branches that have assets greater than at least one branch located in Burnaby.

**select distinct** *T.bname*

**from** *branch S, branch T*

**where** *S.bcity=``Burnaby''* **and** *T.assets > S.assets*

**4.3.6 String Operations**

**4.3.7 Ordering The Display Of Tuples.**

The SQL **ORDER BY** clause is used to sort the data in ascending or descending order, based on one or more columns. Some database sorts query results in ascending order by default.

Syntax:

SELECT column-list

FROM table\_name

[WHERE condition]

[ORDER BY column1, column2, .. columnN] [ASC | DESC];

Example:

SELECT \* FROM CUSTOMERS

ORDER BY NAME, SALARY;

**4.3.8 Duplicate Tuples**