ST.XAVIER’S COLLEGE

**Maitighar, Kathmandu**



**DBMS LAB ASSIGNMENT**

**Submitted By:**

**Utsav Luitel**

013BSCCSIT046

**Submitted To:**

**Er. Sanjay Kumar Yadav** ………..…………….

**Lecturer,**

**Department of Computer Science**

**Date of Submission:- *2015 October***

# Data Definition Language

A data definition language (DDL) is a computer language used to create and modify the structure of database objects in a database. These database objects include views, schemas, tables, indexes, etc.   
  
This term is also known as data description language in some contexts, as it describes the fields and records in a database table.

Commonly used DDL in SQL querying are:

* 1. **Domain Type in SQL**

The SQL-92 standard supports a variety of built-in domain types:

char(n) (or character(n)): fixed-length character string, with user-specified length.

varchar(n) (or character varying): variable-length character string, with user-specified maximum length.

int or integer: an integer (length is machine-dependent).

smallint: a small integer (length is machine-dependent).

numeric(p, d): a fixed-point number with user-specified precision, consists of p digits (plus a sign) and d of p digits are to the right of the decimal point. E.g., numeric(3, 1) allows 44.5 to be stored exactly but not 444.5.

real or double precision: floating-point or double-precision floating-point numbers, with machine-dependent precision.

float(n): floating-point, with user-specified precision of at least n digits.

date: a calendar date, containing four digit year, month, and day of the month.

time: the time of the day in hours, minutes, and seconds.

SQL-92 allows arithmetic and comparison operations on various numeric domains, including, interval and cast (type coercion) such as transforming between smallint and int. It considers strings with different length are compatible types as well.

SQL-92 allows create domain statement, e.g.,

create domain person-name char(20)

Data Manipulation language

The Data Manipulation Language (DML) is used to retrieve, insert and modify database information.

**Select operation :**

The SELECT command is the most commonly used command in SQL. It allows database users to retrieve the specific information they desire from an operational database

Example : **SELECT** name **FROM**  personal\_info **WHERE** salary > $50000

**The select Clause**

1. An example: Find the names of all branches in the *account* relation.

**select** *bname*

**from** *account*

1. **distinct** vs. **all**: elimination or not elimination of duplicates.

Find the names of all branches in the *account* relation.

**select distinct** *bname*

**from** *account*

By default, duplicates are not removed. We can state it explicitly using **all**.

**select all** *bname*

**from** *account*

1. select \* means select all the attributes. Arithmetic operations can also be in the selection list.

**The Where Clause**

The WHERE clause specifies join and filter conditions that determine the rows that the query returns. Join operations in the WHERE clause function the same as JOIN operations in the FROM clause.

SYNTAX

[WHERE JoinCondition | FilterCondition [AND | OR JoinCondition | FilterCondition] ...]

**The From Clause:**

The FROM clause is required in every SELECT statement in which data is being retrieved from tables or views. Use the FROM clause to:

List the tables and views containing the columns referenced in the select list and in the WHERE clause. The table or view names can be aliased using the AS clause.

Join types. These are qualified by join conditions specified in the ON clause.

The FROM clause is a comma-separated list of table names, view names, and JOIN clauses.

The **FROM** clause specifies one or more tables containing the data that the query retrieves from.

**Syntax:**FROM [FORCE] Table\_List\_Item [, ...]

[[JoinType] JOIN DatabaseName!]Table [[AS] Local\_Alias]

[ON JoinCondition [AND | OR [JoinCondition | FilterCondition] ...]

**The Rename Operation:**

Rename statement renames relations and attributes from one or more tables. The rename operation is done atomically, which means that no other session can access any of the tables while the rename is running.

**Syntax:**RENAME TABLE ***tbl\_name*** TO ***new\_tbl\_name***

[, ***tbl\_name2*** TO ***new\_tbl\_name2***] ...

**Tuple Variables:**

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.

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*