ST. XAVIER’S COLLEGE

(Affiliated to Tribhuvan University)

Maitighar, Kathmandu



Database Management System

Lab Assignment #8

Submitted by:

Bishal Pandey  
013BSCCSIT016

Submitted to:

|  |  |
| --- | --- |
| Er. Sanjay Kumar Yadav  Lecturer  St. Xavier’s College |  |

Date of Submission:

**Data Definition Language**

Stands for "Data Definition Language." A DDL is a language used to define data structures and modify [data](http://techterms.com/definition/data). For example, DDL commands can be used to add, remove, or modify [tables](http://techterms.com/definition/table) within in a [database](http://techterms.com/definition/database). DDLs used in database applications are considered a subset of [SQL](http://techterms.com/definition/sql), the Structured Query Language. However, a DDL may also define other types of data, such as [XML](http://techterms.com/definition/xml).

A Data Definition Language has a pre-defined [syntax](http://techterms.com/definition/syntax) for describing data. For example, to build a new table using SQL syntax, the CREATE command is used, followed by parameters for the table name and [column](http://techterms.com/definition/column) definitions. The DDL can also define the name of each column and the associated [data type](http://techterms.com/definition/datatype). Once a table is created, it can be modified using the ALTER command. If the table is no longer needed, the DROP command can be used to delete the table.

Since DDL is a subset of SQL, it does not include all the possible SQL commands. For example, commands such as SELECT and INSERT are considered part of the Data Manipulation Language (DML), while access commands such as CONNECT and EXECUTE are part of the Data Control Language (DCL). The DDL, DML, and DCL languages include most of the commands supported by SQL.

The SQL DDL (Data Definition Language) allows specification of not only a set of relations, but also the following information for each relation:

* The schema for each relation.
* The domain of values associated with each attribute.
* Integrity constraints.
* The set of indices for each relation.
* Security and authorization information.
* Physical storage structure on disk.

**Domain Types in SQL**

1. 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.
2. 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.
3. SQL-92 allows **create domain** statement, e.g.,

**create domain** *person-name* **char**(20)

**Schema definition in SQL**

1. An SQL relation is defined by:

**create table** *r* ( tex2html_wrap_inline1854

*integrity- tex2html_wrap_inline1856*  ,

..., *integrity- tex2html_wrap_inline1856*  )

where *r* is the relation name, tex2html_wrap_inline1730 is the name of an attribute, and tex2html_wrap_inline1864 is the domain of that attribute. The allowed integrity-constraints include

**primary key** tex2html_wrap_inline1866

and

**check(*P*)**

1. Example.

**create table** *branch* (

bname **char**(15) **not null**

bcity **char**(30)

assets **integer**

**primary key** (*bname*)

**check** (*assets >= 0*))

1. The values of primary key must be *not null* and *unique*. SQL-92 consider **not null** in primary key specification is redundant but SQL-89 requires to define it explicitly.
2. Check creates type checking functionality which could be quite useful. E.g.,

**create table** *student* (

*name* **char**(15) **not null**

*student-id* **char**(10) **not null**

*degree-level* **char**(15) **not null**

**check** (*degree-level* **in**

(``Bachelors'', ``Masters'', ``Doctorate'')))

1. Some checking (such as *foreign-key* constraints) could be costly, e.g.,

**check** (*bname* **in** (**select** *bname* **from** *branch*))

1. A newly loaded table is empty. The **insert** command can be used to load it, or use special bulk loader untilities.
2. To remove a relation from the database, we can use the **drop table** command:

**drop table** *r*

This is not the same as

**delete** *r*

which retains the relation, but deletes all tuples in it.

1. The **alter table** command can be used to add or drop attributes to an existing relation *r*:

**alter table** *r* **add** *A* *D*

where *A* is the attribute and *D* is the domain to be added.

**alter table** *r* **drop** *A*

where *A* is the attribute to be dropped.

**The Data Manipulation Language (DML)**

The Data Manipulation Language (DML) is used to retrieve, insert and modify database information. These commands will be used by all database users during the routine operation of the database. Let's take a brief look at the basic DML commands:  
  
The [Data Manipulation Language](http://databases.about.com/od/administration/g/dml.htm) (DML) is used to retrieve, insert and modify database information. These commands will be used by all database users during the routine operation of the database. Let's take a brief look at the basic DML commands:

## 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:**

The basic syntax of SELECT statement is as follows:

**SELECT column1, column2, column FROM table\_name;**

**Example:**

Consider the **Info** table have following records:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Address | Phone |
| 1 | Ram | Kathmandu | 553344 |
| 2 | Shyam | Pokhara | 667788 |
| 3 | Hari | Bhaktapur | 445566 |
| 4 | Bir | Chitwan | 223344 |

SQL Select query:  **SELECT id, name FROM Info**

After the query is processed following result is seen:

|  |  |
| --- | --- |
| Id | Name |
| 1 | Ram |
| 2 | Shyam |
| 3 | Hari |
| 4 | Bir |

## The Where Clause

The SQL WHERE clause is used to specify a condition while fetching the data from single table or joining with multiple tables. If the given condition is satisfied then only it returns specific value from the table. You would use WHERE clause to filter the records and fetching only necessary records. The WHERE clause is not only used in SELECT statement, but it is also used in UPDATE, DELETE statement, etc.

**Syntax**

The basic syntax of SELECT statement with WHERE clause is as follows:

**SELECT column1, column2, column N FROM table\_name**

**WHERE [condition]**

**Example**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Address | Phone | Salary |
| 1 | Ram | Kathmandu | 553344 | 5000 |
| 2 | Shyam | Pokhara | 667788 | 3000 |
| 3 | Hari | Bhaktapur | 445566 | 7000 |
| 4 | Bir | Chitwan | 223344 | 2000 |

SQL Select query:  **SELECT id, name FROM Info WHERE salary > 4000**

The query above is processed and following result appears:

|  |  |
| --- | --- |
| Id | Name |
| 1 | Ram |
| 3 | Hari |

## The FROM Clause

The sql FROM clause is used to select table while retrieving data from a table . The statement FROM is not only used for retrieving data but also used for any join operation.

**Syntax**

The basic syntax of SQL FROM is:

**SELECT column1,column2,…..columnN FROM table\_number**

**Example:**

SQL query: SELECT \* FROM Info

## The Rename Operation

With Rename operation, we can rename a table and give a new name.

**Syntax:**

**RENAME TABLE table\_name TO new\_table\_name**

**Example**

SQL query: RENAME TABLE info To detail\_info