ST. XAVIER’S COLLEGE

**(Affiliated to Tribhuvan University)**

**Maitighar, Kathmandu**

****

**Database Management System**

**Theory Assignment #8**

**SUBMITTED BY:**

**Apil Neupane**

**013BSCCSIT008**

**SUBMITTED TO:**

|  |  |
| --- | --- |
| **Er. Sanjay Kr. Yadav**  **Lecturer** |  |
| **Department of Computer Science** | |

Submission Date: 27th September 2015

**Data-Definition Language:**

A data definition language or data description language (**DDL**) is a syntax similar to a computer programming language for defining data structures, especially database schemas.

DDL statements are used to build and modify the structure of your tables and other objects in the database.

When you execute a DDL statement, it takes effect immediately.

 Use these statements to create, alter, or drop data structures in an instance of SQL Server.

1. The create table statement does exactly that:

CREATE TABLE <table name> (

<attribute name 1> <data type 1>,

...

<attribute name n> <data type n>);

1. The alter table statement may be used as you have seen to specify primary and foreign key constraints, as well as to make other modifications to the table structure. Key constraints may also be specified in the CREATE TABLE statement.

ALTER TABLE <table name>

ADD CONSTRAINT <constraint name> PRIMARY KEY (<attribute list>);

ALTER TABLE <table name>

ADD CONSTRAINT <constraint name> FOREIGN KEY (<attribute list>)

REFERENCES <parent table name> (<attribute list>);

1. If you totally mess things up and want to start over, you can always get rid of any object you’ve created with a drop statement. The syntax is different for tables and constraints.

**DROP TABLE** <table name>;

ALTER TABLE <table name>

**DROP CONSTRAINT** <constraint name>;

**Data Manipulation Language:**

A data manipulation language (**DML**) is a family of syntax elements similar to a computer programming language used for selecting, inserting, deleting and updating data in a database. Performing read-only queries of data is sometimes also considered a component of **DML**.

DML statements are used to work with the data in tables.

1. The insert statement is used, obviously, to add new rows to a table.

INSERT INTO <table name>

VALUES (<value 1>, ... <value n>);

1. The update statement is used to change values that are already in a table.

UPDATE <table name>

SET <attribute> = <expression>

WHERE <condition>;

1. The **delete** statement does just that, for rows in a table.

DELETE FROM <table name>

WHERE <condition>;

**Domain Types in SQL:**

1. **Character Domain Type:**
2. CHAR(n):

Fixed length character string, with user-specified length n.

1. VARCHAR(n):

Variable length character strings, with user-specified maximum length n.

1. Null values are allowed in all the domain types. Declaring an attribute to be NOT NULL prohibits null values for that attribute.
2. **Number Domain Type:**
3. INT (also: INTEGER):

Integer (a finite subset of the integers that is machine-dependent).

1. SMALLINT:

Small integer (a machine-dependent subset of the integer domain type).

1. DECIMAL(p,d):

Fixed point number, with user-specified precision of p digits, with n digits to the right of decimal point.

1. FLOAT(n):

Floating point number, with user-specified precision of at least n digits.

1. REAL (also: DOUBLE PRECISION):

Floating point and double-precision floating point numbers, with machine-dependent precision.

1. **Date Domain Type:**
   1. **DATE:** dates, containing a (4 digit) year, month and date

E.g. DATE ‘2001-7-27’ (ANSI Syntax)

E.g. TO\_DATE(‘01-JUL-27’,’YY-MON-DD’) (Oracle Syntax)

* 1. **TIMESTAMP:** date plus time of day

E.g. TIMESTAMP ‘2001-7-27 09:00:30.75’

**Schema definition in SQL**

A database schema is a way to logically group objects such as tables, views, stored procedures etc. Think of a schema as a container of objects.

We can assign a user login permissions to a single schema so that the user can only access the objects they are authorized to access.

Schemas can be created and altered in a database, and users can be granted access to a schema. A schema can be owned by any user, and schema ownership is transferable.

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*)**

Example.

**create table** *branch* (

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

bcity **char**(30)

assets **integer**

**primary key** (*bname*)

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

The values of primary key must be *not null* and *unique*. SQL-92 considers **not null** in primary key specification is redundant but SQL-89 requires to define it explicitly.

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'')))

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

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

A newly loaded table is empty. The **insert** command can be used to load it, or use special bulk loader utilities.

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.

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.

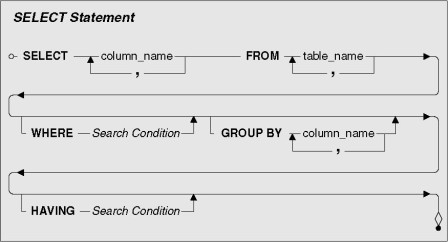
**The select Clause:**

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.

Each *select\_expr* indicates a column that you want to retrieve. There must be at least one *select\_expr*.

The SQL **SELECT** clause specifies the fields, constants, and expressions to display in the query results.

**Syntax:** SELECT [ALL | DISTINCT] [TOP nExpr [PERCENT]] Select\_List\_Item [AS Column\_Name] [, ...]



**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.

The WHERE clause is not only used in SELECT statement, but it is also used in UPDATE, DELETE statement, etc., which we would examine in subsequent chapters.

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*