**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

**Maitighar, Kathmandu**

****

**DATABASE MANAGEMENT SYSTEM**

**THEORY ASSIGNMENT#8**

**Submitted by:**

**Saurav Bajracharya**

**013BSCCSIT035**

**Submitted to:**

|  |  |
| --- | --- |
| **Er. Sanjay Kumar Yadav** |  |

**Lecturer**

**Department of Computer Science**

**Date of submission: 27th September, 2015**

**Data definition language:**

Data Definition Language (DDL) is a standard for commands that define the different structures in a database. DDL statements create, modify, and remove database objects such as tables, indexes, and users. Common DDL statements are CREATE, ALTER, and DROP.

**Domain type in SQL:**

CHAR (n): Fixed length character string, with user-specified length n.

VARCHAR (n): Variable length character strings, with user-specified maximum length n.

Null values are allowed in all the domain types. Declaring an attribute to be NOT NULL prohibits null values for that attribute.

**Number Domain Types in SQL**

• INT (also: INTEGER) Integer (a finite subset of the integers that is machine-dependent).

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

• DECIMAL (p,d) Fixed point number, with user-specified precision of p digits, with n digits to the right of decimal point.

• FLOAT (n) Floating point number, with user-specified precision of at least n digits.

• REAL (also: DOUBLE PRECISION) Floating point and double-precision floating point numbers, with machine-dependent precision.

**Date Domain Types of SQL**

• DATE dates, containing a (4 digit) year, month and date

– E.g. DATE ‘2001-7-27’

• TIMESTAMP date plus time of day – E.g. TIMESTAMP ‘2001-7-27 09:00:30.75’

**Schema definition in SQL**

The structure of a database system described in a formal language supported by the database management system (DBMS). In a relational database, the schema defines the tables, the fields in each table, and the relationships between fields and tables. Schemas are generally stored in a data dictionary. Although a schema is defined in text database language, the term is often used to refer to a graphical depiction of the database structure.

**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.

**The select clause**

The SELECT statement is used to select data from a database.The result is stored in a result table, called the result-set.

SQL SELECT Syntax

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

And

SELECT \* FROM *table\_name*;

**The where clause**

The WHERE clause is used to extract only those records that fulfill a specified criterion.

SQL WHERE Syntax

SELECT *column\_name*,*column\_name*

FROM *table\_name*

WHERE *column\_name operator value*;

**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 rename clause**

You can rename a table or a column temporarily by giving another name known as alias.The use of table aliases means to rename a table in a particular SQL statement. The renaming is a temporary change and the actual table name does not change in the database.The column aliases are used to rename a table's columns for the purpose of a particular SQL query.

## Syntax:

SELECT column1, column2....

FROM table\_name AS alias\_name

WHERE [condition];

**Tuple variable**

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.

**String operations**

SQL string functions are used primarily for string manipulation. The following table details the important string functions:

|  |  |
| --- | --- |
| **Name** | **Description** |
| [**ASCII()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_ascii) | Returns numeric value of left-most character |
| [**BIN()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_bin) | Returns a string representation of the argument |
| [**BIT\_LENGTH()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_bit-length) | Returns length of argument in bits |
| [**CHAR\_LENGTH()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_char-length) | Returns number of characters in argument |
| [**CHAR()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_char) | Returns the character for each integer passed |
| [**CHARACTER\_LENGTH()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_character-length) | A synonym for CHAR\_LENGTH() |
| [**CONCAT\_WS()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_concat-ws) | Returns concatenate with separator |
| [**CONCAT()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_concat) | Returns concatenated string |
| [**CONV()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_conv) | Converts numbers between different number bases |
| [**ELT()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_elt) | Returns string at index number |
| [**EXPORT\_SET()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_export-set) | Returns a string such that for every bit set in the value bits, you get an on string and for every unset bit, you get an off string |
| [**FIELD()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_field) | Returns the index (position) of the first argument in the subsequent arguments |
| [**FIND\_IN\_SET()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_find-in-set) | Returns the index position of the first argument within the second argument |
| [**FORMAT()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_format) | Returns a number formatted to specified number of decimal places |
| [**HEX()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_hex) | Returns a string representation of a hex value |
| [**INSERT()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_insert) | Inserts a substring at the specified position up to the specified number of characters |
| [**INSTR()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_instr) | Returns the index of the first occurrence of substring |
| [**LCASE()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_lcase) | Synonym for LOWER() |
| [**LEFT()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_left) | Returns the leftmost number of characters as specified |
| [**LENGTH()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_length) | Returns the length of a string in bytes |
| [**LOAD\_FILE()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_load-file) | Loads the named file |
| [**LOCATE()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_locate) | Returns the position of the first occurrence of substring |
| [**LOWER()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_lower) | Returns the argument in lowercase |
| [**LPAD()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_lpad) | Returns the string argument, left-padded with the specified string |
| [**LTRIM()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_ltrim) | Removes leading spaces |
| [**MAKE\_SET()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_make-set) | Returns a set of comma-separated strings that have the corresponding bit in bits set |
| [**MID()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_mid) | Returns a substring starting from the specified position |
| [**OCT()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_oct) | Returns a string representation of the octal argument |
| [**OCTET\_LENGTH()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_octet-length) | A synonym for LENGTH() |
| [**ORD()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_ord) | If the leftmost character of the argument is a multi-byte character, returns the code for that character |
| [**POSITION()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_position) | A synonym for LOCATE() |
| [**QUOTE()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_quote) | Escapes the argument for use in an SQL statement |
| [**REGEXP**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#operator_regexp) | Pattern matching using regular expressions |
| [**REPEAT()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_repeat) | Repeats a string the specified number of times |
| [**REPLACE()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_replace) | Replaces occurrences of a specified string |
| [**REVERSE()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_reverse) | Reverses the characters in a string |
| [**RIGHT()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_right) | Returns the specified rightmost number of characters |
| [**RPAD()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_rpad) | Appends string the specified number of times |
| [**RTRIM()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_rtrim) | Removes trailing spaces |
| [**SOUNDEX()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_soundex) | Returns a soundex string |
| [**SOUNDS LIKE**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#operator_sounds-like) | Compares sounds |
| [**SPACE()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_space) | Returns a string of the specified number of spaces |
| [**STRCMP()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_strcmp) | Compares two strings |
| [**SUBSTRING\_INDEX()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_substring-index) | Returns a substring from a string before the specified number of occurrences of the delimiter |
| [**SUBSTRING(), SUBSTR()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_substring) | Returns the substring as specified |
| [**TRIM()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_trim) | Removes leading and trailing spaces |
| [**UCASE()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_ucase) | Synonym for UPPER() |
| [**UNHEX()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_unhex) | Converts each pair of hexadecimal digits to a character |
| [**UPPER()**](http://www.tutorialspoint.com/sql/sql-string-functions.htm#function_upper) | Converts to uppercase |

**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.

The basic syntax of ORDER BY clause is as follows:

SELECT column-list

FROM table\_name

[WHERE condition]

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

Example:

SELECT \* FROM CUSTOMERS

ORDER BY NAME, SALARY;

**Duplicate tuples**

The SQL **DISTINCT** keyword is used in conjunction with SELECT statement to eliminate all the duplicate records and fetching only unique records.

Syntax:

SELECT DISTINCT column1, column2,.....columnN

FROM table\_name

WHERE [condition]

Example:

SELECT DISTINCT SALARY FROM CUSTOMERS

ORDER BY SALARY;