Data Manipulation Language

Wednesday, November 29, 2017 12:22 PM

Data Manipulation Language (DML) is a subset of SQL commands that are responsible for managing data within tables.

DML Commands follow the CRUD (Create, Read, Update, Delete) acronym.

1. Create (INSERT command)
2. Read/Retrieve (SELECT command)
3. Update (UPDATE command)
4. Delete (DELETE command)

SELECT

Use the SELECT statement to retrieve records from a table.

Syntax:

```
SELECT column1, column2, … FROM tablename;
```

Shorthand Syntax for retrieving all columns:

```
SELECT * FROM tablename;
```

ORDER BY

Use the ORDER BY clause to sort results of a SELECT statement

Syntax:

```
SELECT column1, column2, … FROM tablename
  ORDER BY column_name;
```

Aggregate Functions

Many databases ship with pre-defined functions referred to as **aggregate** functions. These functions typically process multiple records or values and return a single value.

Function	Description	Usage
AVG()	Returns the average value	SELECT AVG(column_name) FROM tablename;
SUM()	Returns the sum of the columns	SELECT SUM(column_name) FROM tablename;
MIN()	Returns the minimum value	SELECT MIN(column_name) FROM tablename;
MAX()	Returns the maximum value	SELECT MAX(column_name) FROM tablename;
COUNT()	Returns the number of records in the result	SELECT COUNT(column_name) FROM tablename;

GROUP BY

Use a GROUP BY clause to organize some column values by specified criteria *before* performing some aggregate function.

Syntax:

```
SELECT aggregate_function() FROM tablename
  GROUP BY column_name;
```

WHERE

Use a WHERE clause to filter results from a SELECT query.

Syntax:

```
SELECT column1, column2, … FROM tablename
  WHERE condition;
```

Subqueries

A **subquery** is a SQL query that is nested inside another. The inner query's result is used in the outer query.

Syntax:

```
SELECT column1, column2, … FROM tablename
  WHERE condition = (SELECT column1, column2, … FROM tablename);
```

Notice the use of parenthesis "()" to contain the inner query.

IN

Use the IN operator to evaluate a range of values.

Example:

```
SELECT * FROM Customers WHERE CustomerID IN (1,2,3,4)
```

LIKE

Use the LIKE operator to evaluate alphanumeric values.

Syntax:

```
SELECT column1, column2, … FROM tablename WHERE column1 LIKE 'value';
```

NOTE: The LIKE operator supports the wildcard character, %, which will represent one or more characters in its specified position.

Example:

```
SELECT * FROM Customers WHERE first_name LIKE 'R%';
```

This query will return all rows that have values in the first_name column that start with 'R'.

INSERT

Use the INSERT command to add records to a table.

Syntax:

```
INSERT INTO tablename (column1, column2,…) VALUES (value1, value2, …);
```

Alternative syntax:

```
INSERT INTO tablename VALUES (value1, value2, …);
```

NOTE: Use the above version if you're specifying values for all columns instead of specific ones.

UPDATE

Use the UPDATE command to change data within a table.

Syntax:

```
UPDATE tablename SET column_name = value
```

Syntax:

```
UPDATE tablename
 SET column1 = value1,
     column2 = value2,
     …;
```

Syntax:

```
UPDATE tablename
 SET column = value
 WHERE condition;
```

DELETE

Use the DELETE command to remove a row from a table

Syntax:

```
DELETE FROM tablename;
```

Syntax for filtered records:

```
DELETE FROM tablename WHERE condition;
```