18

Database: SQL Fundamentals

18.4 SQL

- The next several subsections discuss the keywords listed in the following slide in the context of SQL queries and statements.

SQL keyword	Description
`SELECT`	Retrieves data from one or more tables.
`FROM`	Tables involved in the query. Required in every `SELECT`.
`WHERE`	Criteria for selection that determine the rows to be retrieved, deleted or updated. Optional in a SQL query or a SQL statement.
`GROUP BY`	Criteria for grouping rows. Optional in a `SELECT` query.
`ORDER BY`	Criteria for ordering rows. Optional in a `SELECT` query.
`INNER JOIN`	Merge rows from multiple tables.
`INSERT`	Insert rows into a specified table.
`UPDATE`	Update rows in a specified table.
`DELETE`	Delete rows from a specified table.

Fig. 18.10 | SQL query keywords.

18.4.1 Basic SELECT Query

- The basic form of a `SELECT` query is

  ```
  SELECT * FROM tableName
  ```

  in which the asterisk (*) *wildcard character* indicates that all columns from the *`tableName`* should be retrieved.
- To retrieve specific columns, replace the * with a comma-separated list of column names.

AuthorID	LastName
1	Deitel
2	Deitel
3	Deitel
4	Morgano
5	Kern

Fig. 18.11 | Sample `AuthorID` and `LastName` data from the `Authors` table.

Software Engineering Observation 18.1

In general, you process results by knowing in advance the order of the columns in the result—for example, selecting `AuthorID` and `LastName` from table `Authors` ensures that the columns will appear in the result with `AuthorID` as the first column and `LastName` as the second. Programs typically process result columns by specifying the column number in the result (starting from 1 for the first column). Selecting columns by name avoids returning unneeded columns and protects against changes to the order of the columns in the table(s) by returning the columns in the exact order specified.

Common Programming Error 18.4

If you assume that the columns are always returned in the same order from a query that uses the asterisk (*), the program may process the results incorrectly.

18.4.2 WHERE Clause

- SQL uses the optional `WHERE` clause in a query to specify the selection criteria for the query. The basic form of a query with selection criteria is

```
SELECT columnName1, columnName2, … FROM
      tableName WHERE criteria
```

- Strings in SQL are delimited by single (') rather than double (") quotes.

Title	EditionNumber	Copyright
Visual Basic 2010 How to Program	5	2011
Visual C# 2010 How to Program	4	2011
Java How to Program	9	2012
C++ How to Program	8	2012
Android for Programmers: An App-Driven Approach	1	2012

Fig. 18.12 | Sampling of titles with copyrights after 2005 from table `Titles`.

18.4.2 WHERE Clause (Cont.)

Pattern Matching: Zero or More Characters

- The WHERE clause can contain operators <, >, <=, >=, =, <> and LIKE. Operator LIKE is used for string pattern matching with wildcard characters percent (%) and underscore (_).
- A percent character (%) in a pattern indicates that a string matching the pattern can have zero or more characters at the percent character's location in the pattern.
- An underscore (_) in the pattern string indicates a single character at that position in the pattern.

AuthorID	FirstName	LastName
1	Paul	Deitel
2	Harvey	Deitel
3	Abbey	Deitel

Fig. 18.13 | Authors whose last name starts with D from the Authors table.

Portability Tip 18.1

See the documentation for your database system to determine whether SQL is case sensitive on your system and to determine the syntax for SQL keywords.

Portability Tip 18.2

Read your database system's documentation carefully to determine whether it supports the `LIKE` operator as discussed here.

18.4.2 WHERE Clause (Cont.)

Pattern Matching: Any Character

- An underscore (_) in the pattern string indicates a single wildcard character at that position in the pattern.
- The following query locates the rows of all the authors whose last names start with any character (specified by _), followed by the letter o, followed by any number of additional characters (specified by %):

```
SELECT AuthorID, FirstName, LastName
       FROM Authors
       WHERE LastName LIKE '_o%'
```

- The preceding query produces the row shown in the next slide, because only one author in our database has a last name that contains the letter o as its second letter.

AuthorID	FirstName	LastName
4	Michael	Morgano

Fig. 18.14 | The only author from the `Authors` table
whose last name contains o as the second letter.

18.4.3 ORDER BY Clause

- The result of a query can be sorted in ascending or descending order using the optional ORDER BY clause. The simplest form of an ORDER BY clause is

  ```
  SELECT columnName1, columnName2, … FROM
        tableName ORDER BY column ASC
  SELECT columnName1, columnName2, … FROM
        tableName ORDER BY column DESC
  ```

  where ASC specifies ascending order, DESC specifies descending order and column specifies the column on which the sort is based. The default sorting order is ascending, so ASC is optional.
- Multiple columns can be used for ordering purposes with an ORDER BY clause of the form

  ```
  ORDER BY column1 sortingOrder, column2
  sortingOrder, …
  ```
- The WHERE and ORDER BY clauses can be combined in one query. If used, ORDER BY must be the last clause in the query.

AuthorID	FirstName	LastName
1	Paul	Deitel
2	Harvey	Deitel
3	Abbey	Deitel
5	Eric	Kern
4	Michael	Morgano

Fig. 18.15 | Sample data from table Authors in ascending order by LastName.

AuthorID	FirstName	LastName
4	Michael	Morgano
5	Eric	Kern
1	Paul	Deitel
2	Harvey	Deitel
3	Abbey	Deitel

Fig. 18.16 | Sample data from table `Authors` in descending order by `LastName`.

AuthorID	FirstName	LastName
3	Abbey	Deitel
2	Harvey	Deitel
1	Paul	Deitel
5	Eric	Kern
4	Michael	Morgano

Fig. 18.17 | Sample data from `Authors` in ascending order by `LastName` and `FirstName`.

ISBN	Title	Editio n-Number	Copy-right
0132404168	C How to Program	6	2010
0132662361	C++ How to Program	8	2012
0132575663	Java How to Program	9	2012
0132152134	Visual Basic 2005 How to Program	5	2011
0132151421	Visual C# 2005 How to Program	4	2011

Fig. 18.18 | Sampling of books from table `Titles` whose titles end with `How to Program` in ascending order by `Title`.

18.4.4 Merging Data from Multiple Tables: `INNER JOIN`

- An `INNER JOIN` operator merges rows from two tables by matching values in columns that are common to the tables. The basic form for the `INNER JOIN` operator is:

```
SELECT columnName1, columnName2, …
FROM table1
INNER JOIN table2
      ON table1.columnName = table2.columnName
```

- The **ON clause** of the INNER JOIN specifies the columns from each table that are compared to determine which rows are merged.
- The following query produces a list of authors accompanied by the ISBNs for books written by each author:

```
SELECT FirstName, LastName, ISBN
    FROM Authors
    INNER JOIN AuthorISBN
       ON Authors.AuthorID = AuthorISBN.AuthorID
    ORDER BY LastName, FirstName
```

18.4.4 Merging Data from Multiple Tables: `INNER JOIN` (Cont.)

- Note the use of the syntax *`tableName.columnName`* in the `ON` clause. This syntax, called a qualified name, specifies the columns from each table that should be compared to join the tables.
- The "*tableName*." syntax is required if the columns have the same name in both tables. The same syntax can be used in any SQL statement to distinguish columns in different tables that have the same name.
- In some systems, table names qualified with the database name can be used to perform cross-database queries. As always, the query can contain an `ORDER BY` clause.

FirstName	LastName	ISBN
Abbey	Deitel	013705842X
Abbey	Deitel	0132121360
Harvey	Deitel	0132152134
Harvey	Deitel	0132151421
Harvey	Deitel	0132575663
Harvey	Deitel	0132662361
Harvey	Deitel	0132404168
Harvey	Deitel	013705842X
Harvey	Deitel	0132121360
Paul	Deitel	0132152134
Paul	Deitel	0132151421
Paul	Deitel	0132575663
Paul	Deitel	0132662361
Paul	Deitel	0132404168
Paul	Deitel	013705842X
Paul	Deitel	0132121360
Eric	Kern	013705842X
Michael	Morgano	013705842X
Michael	Morgano	0132121360

Fig. 18.19 | Sampling of authors and ISBNs for the books they have written in ascending order by `LastName` and `FirstName`.

Software Engineering Observation 18.2

If a SQL statement includes columns with the same name from multiple tables, the statement must precede those column names with their table names and a dot (e.g., `Authors.AuthorID`).

Common Programming Error 18.5

Failure to qualify names for columns that have the same name in two or more tables is an error.

18.4.5 INSERT Statement

- An `INSERT` statement inserts a new row into a table. The basic form of this statement is

  ```
  INSERT INTO tableName ( columnName1,
        columnName2, …, columnNameN )
        VALUES ( value1, value2, …, valueN )
  ```

  where *tableName* is the table in which to insert the row. The *tableName* is followed by a comma-separated list of column names in parentheses. The list of column names is followed by the SQL keyword VALUES and a comma-separated list of values in parentheses.
- Always explicitly list the columns when inserting rows. If the table's column order changes or a new column is added, using only VALUES may cause an error.

18.4.5 INSERT Statement (Cont.)

- The INSERT statement

```
INSERT INTO Authors ( FirstName, LastName )
      VALUES ( 'Sue', 'Red' )
```

inserts a row into the `Authors` table.

- The statement indicates that values are provided for the `FirstName` and `LastName` columns.
- The corresponding values are `'Sue'` and `‘Red'`.
- We do not specify an `AuthorID` in this example because `AuthorID` is an autoincremented column.
- For every row added to this table, the DBMS assigns a unique `AuthorID` value that is the next value in the autoincremented sequence (i.e., 1, 2, 3 and so on). In this case, Sue Red would be assigned `AuthorID` number 6.

AuthorID	FirstName	LastName
1	Paul	Deitel
2	Harvey	Deitel
3	Abbey	Deitel
4	Michael	Morgano
5	Eric	Kern
6	Sue	Red

Fig. 18.20 | Sample data from table Authors after an INSERT operation.

Common Programming Error 18.6

It's normally an error to specify a value for an autoincrement column.

Common Programming Error 18.7

SQL delimits strings with single quotes (`'`). A string containing a single quote (e.g., O'Malley) must have two single quotes in the position where the single quote appears (e.g., `'O''Malley'`). The first acts as an escape character for the second. Not escaping single-quote characters in a string that's part of a SQL statement is a SQL syntax error.

18.4.6 UPDATE Statement

- An `UPDATE` statement modifies data in a table. Its basic form is

```
UPDATE tableName
       SET columnName1 = value1,
           columnName2 = value2, …,
           columnNameN = valueN
       WHERE criteria
```

  where *tableName* is the table in which to update data. The *tableName* is followed by keyword `SET` and a comma-separated list of column name/value pairs in the format *columnName = value*. The optional `WHERE` clause *criteria* determines which rows to update.

AuthorID	FirstName	LastName
1	Paul	Deitel
2	Harvey	Deitel
3	Abbey	Deitel
4	Michael	Morgano
5	Eric	Kern
6	Sue	Black

Fig. 18.21 | Sample data from table Authors after an UPDATE operation.

18.4.7 DELETE Statement

- A SQL `DELETE` statement removes rows from a table. Its basic form is

  ```
  DELETE FROM tableName WHERE
          criteria
  ```

  where *`tableName`* is the table from which to delete a row (or rows). The optional `WHERE` *`criteria`* determines which rows to delete. If this clause is omitted, all the table's rows are deleted.

AuthorID	FirstName	LastName
1	Paul	Deitel
2	Harvey	Deitel
3	Abbey	Deitel
4	Michael	Morgano
5	Eric	Kern

Fig. 18.22 | Sample data from table `Authors` after a `DELETE` operation.