15.5: STRUCTURED QUERY LANGUAGE SUMMARY

Contributed by Chuck Severance
Clinical Associate Professor (School of Information) at University of Michigan

So far, we have been using the Structured Query Language in our Python examples and have covered many of the basics of the SQL commands. In this section, we look at the SQL language in particular and give an overview of SQL syntax.

Since there are so many different database vendors, the Structured Query Language (SQL) was standardized so we could communicate in a portable manner to database systems from multiple vendors.

A relational database is made up of tables, rows, and columns. The columns generally have a type such as text, numeric, or date data. When we create a table, we indicate the names and types of the columns:

```
CREATE TABLE Tracks (title TEXT, plays INTEGER)
```

To insert a row into a table, we use the SQL `INSERT` command:

```
INSERT INTO Tracks (title, plays) VALUES ('My Way', 15)
```

The `INSERT` statement specifies the table name, then a list of the fields/columns that you would like to set in the new row, and then the keyword `VALUES` and a list of corresponding values for each of the fields.

The SQL `SELECT` command is used to retrieve rows and columns from a database. The `SELECT` statement lets you specify which columns you would like to retrieve as well as a `WHERE` clause to select which rows you would like to see. It also allows an optional `ORDER BY` clause to control the sorting of the returned rows.

```
SELECT * FROM Tracks WHERE title = 'My Way'
```

Using `*` indicates that you want the database to return all of the columns for each row that matches the `WHERE` clause.

Note, unlike in Python, in a SQL `WHERE` clause we use a single equal sign to indicate a test for equality rather than a double equal sign. Other logical operations allowed in a `WHERE` clause include `<`, `>`, `<=`, `>=`, `!=`, as well as `AND` and `OR` and parentheses to build your logical expressions.

You can request that the returned rows be sorted by one of the fields as follows:

```
SELECT title,plays FROM Tracks ORDER BY title
```

To remove a row, you need a `WHERE` clause on an SQL `DELETE` statement. The `WHERE` clause determines which rows are to be deleted:

```
DELETE FROM Tracks WHERE title = 'My Way'
```

It is possible to `UPDATE` a column or columns within one or more rows in a table using the SQL `UPDATE` statement as follows:

```
UPDATE Tracks SET plays = 16 WHERE title = 'My Way'
```

The `UPDATE` statement specifies a table and then a list of fields and values to change after the `SET` keyword and then an optional `WHERE` clause to select the rows that are to be updated. A single `UPDATE` statement will change all of the rows that match the `WHERE` clause. If a `WHERE` clause is not specified, it performs the `UPDATE` on all of the rows in the table.

These four basic SQL commands (INSERT, SELECT, UPDATE, and DELETE) allow the four basic operations needed to create and maintain data.