CSCI 403: Databases
5 - Insert, Update, and Delete

Modifying the Database

So far, we have discussed SELECT queries, which merely retrieve data from the database. Now we turn to queries which can modify the database (these are called queries, also, even though they are more in the nature of commands).

Unlike SELECT queries, a correctly formed modification query has the possibility of *failing* due to a constraint violation. For instance, you might try to insert a NULL value into a NOT NULL constrained column, or insert into a foreign key column a value which doesn't exist in the referenced table. Update can likewise violate column or key constraints, and deletes can cause foreign key constraint violations. In all of these instances, the query will fail entirely rather than doing a partial job - e.g., an update will not update some rows unless it can update all the rows indicated by the query.

Insert

The INSERT query is the primary means by which data is added to a table in a SQL database. At its simplest, INSERT queries let you add a single tuple to a single table, providing values in the order of the attributes in the table:

```
INSERT INTO tablename
VALUES (v1, v2, ...);
```

More generally, you can insert multiple tuples (into a single table), specifying only fields for which you have values:

```
INSERT INTO tablename (A1, A2, ...)
VALUES (v1, v2, ...), (w1, w2, ...);
```

The above will insert two rows into the table specified; the first row will get attribute A1 = value v1, A2 = v2, etc., and the second row will get A1 = w1, A2 = w2, etc. Any attributes which are not in the attribute list for the INSERT will get NULL unless a default value has been provided in the table definition.

You can also insert multiple tuples resulting from a SELECT query into a table with compatible attribute types:

```
INSERT INTO tablename (A1, A2, ...)
SELECT B1, B2, ...  ;
```

Example:

```
INSERT INTO mines_eecs_faculty (name,
office, email)
VALUES
('Painter-Wakefield, Christopher',
'BB 280I','cpainter@mines.edu');
```

Delete

A DELETE statement simply eliminates tuples matching the WHERE clause of the DELETE:

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

If no condition is provided, then the statement deletes all rows from the table.

Example:

```
DELETE FROM mines_courses
WHERE instructor = 'Painter-Wakefield,
Christopher';
```

Update

Update is used to modify existing data by changing the value of one or more attributes for one or more tuples in a table. The UPDATE query looks like

```
UPDATE tablename
SET A1 = e1, A2 = e2, ...
WHERE condition;
```

The UPDATE query can be the trickiest to understand in that it allows the updating of multiple rows (depending on the WHERE condition), and each row can be updated in an independent

fashion (depending on the expressions used in the SET clause).

When the UPDATE query is executed, the database will iterate over all tuples in the specified table that match the WHERE condition; the SET clause is then applied to each matching tuple individually. For each assignment in the SET clause of the form `A = e`, A is the name of the attribute receiving the update, and e is some expression that evaluates to a value that is type-compatible with A. Because the expression can include constants, functions, other attribute values, or even sub-queries, UPDATE is quite powerful.

Here's a simple example, affecting only one row of a table:

```
UPDATE mines_eecs_faculty
SET office = 'BB 280I',
email = 'cpainter@mines.edu'
WHERE name = 'Painter-Wakefield,
Christopher';
```

Here's a more complex example; suppose we have a person table composed of first and last names. When we want the full name as a single string, we can compute it on the fly using the concatenation (||) operator, but that might be tedious; instead, we might choose to maintain an additional column for full name. The update query might, perhaps, run on a regular schedule to update any full names of persons entered in the database since the last update. The full name column may be overwritten (some people might prefer a different form for their full name), so our update query should ignore rows that already have a full name:

```
UPDATE person
SET fullname =
firstname || ' ' || lastname
WHERE fullname IS NULL;
```

Note that the RHS of the assignment in the above query is an expression composed of concatenated attributes in the table; the attribute values come from the row that is being updated.

Bulk Loading of Data

While it is possible that your database may start empty and be filled one tuple at a time as data comes in, the more common case has you starting with some body of data from a previous system in a different format. In this case, while you could write code to generate INSERT statements for all of the data, running such code (and executing the resulting queries) is tedious and inefficient. Most DBMSes therefore provide some method for bulk-loading of data.

In the PostgreSQL database, the provided mechanism is the COPY command. The COPY command looks like

```
COPY tablename (A1, A2, ...)
FROM filename
WITH (option1, ...);
```

This specifies a filename from which data will be read, and a tablename into which the rows of data will be inserted (optionally specifying the attribute names corresponding to the values in the file). The options specify information about the formatting and encoding of the file.

The COPY command is limited in that it expects the filename to be the full path of some file on the database server's filesystem, which isn't always accessible to developers. Fortunately, the psql utility provides a command for utilizing the COPY command from a remote machine. In psql, you can issue the \COPY command; other than the beginning backslash, this command has the same syntax as the PostgreSQL COPY command. From psql, the filename needs to include the full path or the path relative to the working directory (the directory from which psql was launched).

The command below might be used to fill a table named 'foo' with values from a comma-separated values file named 'bar.csv':

```
\COPY foo FROM 'bar.csv'
WITH (FORMAT csv, ENCODING 'utf-8');
```