CSCI 403
Database Management

7

Insert, update, delete

Adding new data

INSERT

Simple INSERT

We saw this last time:

```
INSERT INTO name
VALUES (val1, val2, …);
```

e.g.

```
CREATE TABLE junk (x INTEGER, y DATE);
INSERT INTO junk VALUES (4, '2010-07-06');
INSERT INTO junk VALUES (33, '2015-11-29');
SELECT * FROM junk;
```


INSERT

More generally:

```
INSERT INTO table (column1, column2, …)
VALUES (value1, value2, …),
       (row2value1, row2value2, …), …;
```

The number and types of values must match the number of and types of the specified columns.

Any column not specified gets NULL (unless there is a default).

If you omit the columns list, then SQL will assume you are providing values for all columns in order.

Examples

```
CREATE TABLE junk (x INTEGER, y DATE);

INSERT INTO junk (x) VALUES (42);

INSERT INTO JUNK (x, y) VALUES (42, NULL);

INSERT INTO junk (x, y) VALUES (17, '02-MAY-99');

INSERT INTO junk
VALUES (1, '2018-08-20'),
       (2, '2018-08-22'),
       (3, '2018-08-24');

INSERT INTO junk VALUES (123+456, current_date);
```

Same effect

INSERT INTO…SELECT

Shorthand way to get data from one table to another:

```
INSERT INTO table (column1, column2, …)
SELECT expr1, expr2, … FROM …
```

E.g.,

```
CREATE TABLE mines_cs_courses (course_id text,
section text, instructor text);

INSERT INTO mines_cs_courses
SELECT course_id, section, instructor
FROM mines_courses
WHERE course_id LIKE 'CSCI%';
```

Getting rid of data

DELETE

DELETE

DELETE deletes rows matching the (optional) WHERE clause:

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

With no WHERE clause, just DELETES all rows.

Pro-tip

DELETE is irrevocable.

Think

```
rm -rf /*
```

for you linux folks.

Easy trick to make sure you are deleting what you *intend* to delete:

First do your query, replacing DELETE with SELECT *.

This will show you exactly what you will DELETE!

Modifying rows

UPDATE

UPDATE

UPDATE table

SET column1 = expr1, column2 = expr2, …

WHERE condition;

Example:

```
UPDATE mines_cs_faculty
SET office = 'BB 280N'
WHERE name LIKE 'Painter%';
```


Understanding Update

Update:

- Modifies only rows matching (optional) WHERE condition
- Modifies each row independently
  - Each assignment of the form “columnx = expressionx”
  - The expression in the assignment is evaluated on a per-row basis

Example 1

Suppose we have this table:

foo

x	y	z
1	9	<null>
2	16	0
3	4	-1

```
UPDATE foo SET y = sqrt(y) WHERE z IS NOT NULL;
```

foo

x	y	z
1	9	<null>
2	4	0
3	2	-1

Example 2

foo

x	y	z
1	9	<null>
2	16	0
3	4	-1

```
UPDATE foo SET z = y + z, x = x + 1;
```

foo

x	y	z
2	9	<null>
3	16	16
4	4	3

Example 3

- Table **person** with last, first, preferred names:
  - New people added, but only last & first
  - Need to set something for preferred
    - (Could have done this in INSERT, but someone forgot)
  - Don't want to mess up existing preferred names

```
UPDATE person
SET preferred = first || ' ' || last
WHERE preferred IS NULL;
```


Up Next

- Next lecture:
  Constraints, default values, sequences