L4: SQL Data Manipulation Basics

CS1106/CS6503: Intro to Relational Databases

Dr Kieran T. Herley

Semester One, 2023-24

School of Computer Science & Information Technology
University College Cork

Data Manipulation

- SELECT statement extracts information fro our DB, but leaves the DB unchanged
- SQL also includes date manipulation statements to alter DB contents

INSERT	adds new rows to existing table
DELETE	removes some rows from table
UPDATE	changes some values within table

- Careful now! Potentially destructive and irreversible.

Inserting a Single Row

- Add a new row with the details of a new student

```
INSERT INTO students
VALUES ('987654321', 'Graine', 'Gogerty', '1992-12-13',
    'Skibbereen', 'ck401', 525 );
```

- Values supplied must match table columns in number, order and type.
- Note: id "number" treated as string.

Inserting a Multiple Rows

- Can insert multiple rows all at once

```
INSERT INTO students VALUES ( . . .);
INSERT INTO students VALUES ( . . .);
INSERT INTO students VALUES ( . . .);
```

or

```
INSERT INTO students
VALUES
    ( . . .),
    ( . . .),
    ( . . .);
```

i.e. multiple (...) separated by commas

Populating a Database From Scratch

- Can “populate” an empty DB with a barrage of INSERTs
- File `students_populate.sql` contains

```
INSERT INTO students VALUES ( . . 'Aoife', 'Ahern'. . . .);
INSERT INTO students VALUES ( . . 'Barry', 'Barry'. . . .);
      . . .
INSERT INTO students VALUES ( . . 'Fionn', 'Fitzgerald'. . . .);
```

- NB MySQL or SQLite can accept SQL instructions from a file.
- Could use to set up DBs, but we use sqlite files instead

Inserting Partial Rows

- Can also perform insertions with only some column values are supplied

```
INSERT INTO students (id_number, first_name, last_name)
VALUES ('987654321', 'Graine', 'Gogerty');
```

- "Missing" values (e.g. hometown) set to NULL (can specify default)

id_number	first_name	last_name	date_of_birth	hometown	course	points	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	← old rows
987654321	Graine	Gogerty	NULL	NULL	NULL	NULL	← new row

NULL

- NULL is a special marker that is compatible with any domain/type; is not a value *per se*
- Typically used to denote situations where a value
    - is not known
    - irrelevant
    - is not applicable
- Can be problematic; over-reliance on NULLs may suggest poor DB design
- Checking NULLness: `IS NULL` or `IS NOT NULL`

Bad Insertions

- “Bad” insertions may be rejected (i.e. not take effect)
    - Attempt to insert duplicate key
    - Values incompatible with column type etc.
- However some “bad” insertions may be technically legal but nonsense; such insertions will contaminate your table
    - Mixing up columns (e.g. confusing first name and last name, or first name and hometown)
    - “Fat fingers” errors, types etc. e.g. 5000 points

Deleting a Row

- Use DELETE to remove row(s)

```
DELETE
FROM students
WHERE id_number = '987654321';
```

  specify victim(s) using SELECT-style WHERE condition

- What wrong with the following?

```
DELETE
FROM students
WHERE first_name = 'Graine' AND last_name = 'Gogerty';
```

Deleting Multiple Rows

- Can also delete multiple rows

```
DELETE
FROM students
WHERE hometown = 'Tralee';
```

- Need to be very careful with this!

Updating Values Within a Table

- Use UPDATE to modify existing values within table

```
UPDATE students
SET points = 500
WHERE id_number = '112356489';
```

- Uses SELECT-style WHERE condition to specify target

Updating Multiple Values

- Can update multiple values all at once

```
UPDATE students
SET points = 1.2*points
WHERE hometown = 'Tralee';
```

- Increases all Tralee students' points by 20%
- Interpretation of

$$\text{points} = 1.2 * \text{points}$$

Left hand side indicates value to be updated; right hand side specifies number to be to be used (1.2 times existing points value of row)