



A11.5 SQL: changing data and referential integrity

Databases and big data · A level · OCR H446 1.3.2, AQA 7517 4.10.2,
Eduqas A500QS 2.5 · about 55 min

Name _____

Class _____

Date _____

What this lesson is about

INSERT, UPDATE and DELETE at A level, and how a database refuses orphan records. The robot logs its own moves.

- INSERT
- UPDATE
- DELETE
- Referential integrity
- The robot logs its own runs

Questions

5 marks in all

- What does referential integrity mean? [1 mark]
 - ☐ A Every foreign key value matches a primary key value in the table it refers to
 - ☐ B Every table has a primary key
 - ☐ C No two records are the same
 - ☐ D Every field has a value
- Runs refer to robots with ON DELETE CASCADE. What happens when robot 2 is deleted? [1 mark]
 - ☐ A Robot 2's runs are deleted too
 - ☐ B The delete is refused
 - ☐ C Robot 2's runs keep robot_id 2
 - ☐ D Robot 2's runs get robot_id null
- What does this print? [1 mark]

```
import sqlite3
db = sqlite3.connect(":memory:")
db.execute("PRAGMA foreign_keys = ON")
db.execute("CREATE TABLE robots (robot_id INTEGER PRIMARY KEY)")
db.execute("CREATE TABLE runs (run_id INTEGER PRIMARY KEY, robot_id INTEGER REFERENCES robots(robot_id))")
db.execute("INSERT INTO robots VALUES (1)")
db.execute("INSERT INTO runs (robot_id) VALUES (1)")
try:
    db.execute("INSERT INTO runs (robot_id) VALUES (5)")
    print("added")
except sqlite3.IntegrityError:
    print("refused")
print(db.execute("SELECT COUNT(*) FROM runs").fetchone()[0])
```

4. What does this print?

[1 mark]

```
import sqlite3
db = sqlite3.connect(":memory:")
db.execute("CREATE TABLE runs (run_id INTEGER PRIMARY KEY, robot_id INTEGER, cm REAL)")
db.executemany("INSERT INTO runs VALUES (?, ?, ?)", [(1, 1, 40.0), (2, 2, 38.0), (3, 1, 50.0)])
db.execute("UPDATE runs SET cm = cm + 5 WHERE robot_id = 1")
db.execute("DELETE FROM runs WHERE cm < 45")
print(db.execute("SELECT run_id, cm FROM runs").fetchall())
```

5. Why should a program insert values with ? placeholders rather than joining them into the SQL string? [1 mark]

- ☐ A The values are passed separately, so typed text cannot change the SQL (SQL injection)
- ☐ B Placeholders make the query run without a database
- ☐ C Placeholders are required for numbers
- ☐ D Joining strings is not allowed in Python

The task: log the legs, keep the links

The starter builds two tables, `robots` and `runs`. `runs.robot_id` refers to `robots(robot_id)` with `ON DELETE CASCADE`. Robot 1 is Ada (the robot on the mat) and robot 2 is Bolt, who already has run 1. `legs` is a list of `(move, cm)`, where move is "forward", "right" or "backward" and cm is a whole number of centimetres. 1. Switch foreign key checking on. 2. For each leg in order, drive it at speed 40, measure how far the robot really moved from `position()` before and after (the straight-line distance, rounded to a whole number), and `INSERT` it as a run for robot 1 with a field list and `?` placeholders, letting the database number it. 3. Try to insert a run for robot 9. When the database refuses with `sqlite3.IntegrityError`, print `refused: robot 9 does not exist`. 4. `UPDATE` the runs whose move is `right` so their move is `strafe`. 5. `DELETE` Bolt from `robots`. 6. With a query that joins `runs` to `robots`, print every run in run order as `run <run_id> <name> <move> <cm>`, then print `runs: <n>`, the number of records left in `runs`. Five lines in all, starting with the refusal.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

import sqlite3

db = sqlite3.connect(":memory:")
db.execute("CREATE TABLE robots (robot_id INTEGER PRIMARY KEY, name TEXT NOT NULL)")
db.execute("""CREATE TABLE runs (run_id INTEGER PRIMARY KEY,
                                robot_id INTEGER NOT NULL REFERENCES robots(robot_id) ON DELETE CASCADE,
                                move TEXT, cm INTEGER)""")
db.executemany("INSERT INTO robots VALUES (?, ?)", [(1, "Ada"), (2, "Bolt")])
db.execute("INSERT INTO runs VALUES (1, 2, 'forward', 30)")
db.commit()

legs = [("forward", 20), ("right", 15), ("backward", 10)]
```

Plan your program here, then type it in and press Run.



Do it on the robot

www.bugbotlab.com/learn/a11-5-changing-data-and-referential-integrity/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Change the `runs` table so deleting a robot sets its runs' `robot_id` to null instead. What must you remove from the table definition first?
2. Using `club.py`, give every robot in Hawks 1 cm extra on its wall runs with one `UPDATE` and a nested `SELECT`.

3. Why do most designers choose restrict rather than cascade for a table of exam results?

BugBotLab · free Python lessons that drive a robot · www.bugbotlab.com/learn/a11-5-changing-data-and-referential-integrity/ · You may copy this for your class.