



A11.10 Project: the run database

Databases and big data · A level · OCR H446 1.3.2, AQA 7517 4.10.1,
Eduqas A500QS 2.5 · about 75 min

Name _____

Class _____

Date _____

What this lesson is about

Design, normalise, build and fill a database of robot runs, log live runs in transactions, report with one joined query and export the report as JSON.

- The brief
- Design
- Plan
- Step 1: tables that refer to each other
- Step 2: measure a run
- Step 3: an error in SQL

Questions

5 marks in all

1. Why does the project's Run table not store each run's error? [1 mark]

- ☐ A It depends on the run's distance and the challenge's target, so storing it would be redundant and could become inconsistent
- ☐ B SQL cannot store negative numbers
- ☐ C The error is a primary key
- ☐ D JSON cannot hold it

2. A report query joins runs, robots, teams and challenges. How many JOIN clauses does it need? [1 mark]

3. What does this print? [1 mark]

```
import sqlite3
db = sqlite3.connect(":memory:")
db.execute("CREATE TABLE runs (team TEXT, cm INTEGER, target INTEGER)")
db.executemany("INSERT INTO runs VALUES (?, ?, ?)", [("Hawks", 27, 30), ("Hawks", 31, 30),
("Owls", 16, 15)])
for row in db.execute("SELECT team, COUNT(*), AVG(ABS(cm - target)) FROM runs GROUP BY team
ORDER BY team"):
    print(*row)
```

4. Put the stages of building the run database in order. [1 mark]

Number the lines 1 to 5 to put them in the right order.

- ___ Query the tables to produce the report
- ___ Insert the data and log new runs in transactions
- ___ Normalise the design to third normal form
- ___ Create the tables, parents before children
- ___ Identify the entities, keys and relationships from the brief

5. Ada's run is logged inside a transaction (with db:) and the program crashes halfway through the insert. [1 mark]
What is in the database?
- ☐ A No trace of the run, because the transaction was rolled back
 - ☐ B Half of the run's fields
 - ☐ C The run, marked as incomplete
 - ☐ D Nothing at all: the whole database is emptied

The task: the run database

The starter gives the data as lists of tuples: `teams` holds `(team_id, team_name)`, `robots` holds `(robot_id, name, team_id)`, `challenges` holds `(code, name, target_cm)` and `earlier_runs` holds `(robot_id, code, cm)`. Ids and distances are whole numbers; names and codes are text. 1. Create the four tables of the design, with a primary key on each and the three foreign keys declared with `REFERENCES`, and switch on foreign key checking. 2. Insert the data lists. Let the database number the runs. 3. For each challenge, in the order of `challenges`, the robot (Ada, robot 1) makes one run at speed 50: `S` drives forward the target distance, `R` drives backward the target distance. Measure the straight-line distance really covered, rounded to a whole number, and insert the run inside its own transaction (`with db:`). 4. With **one** query that joins `runs`, `robots`, `teams` and `challenges`, groups by team, and uses `COUNT` and `AVG`, print one line per team in name order: `<team_name>: <n> runs, mean error <error> cm`, where the error is the mean of the absolute differences between each run's `cm` and its challenge's `target_cm`, rounded to 1 decimal place. For example `Owls: 2 runs, mean error 2.5 cm`. 5. Save the report to `report.json` with `json.dump`, as a list with one object per team, each with the keys `team` (text), `runs` (a whole number) and `mean_error` (a number). Then print `saved report.json`. Three lines in all.

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

import sqlite3
import json

teams = [(1, "Hawks"), (2, "Owls")]
robots = [(1, "Ada", 1), (2, "Bolt", 1), (3, "Cog", 2)]
challenges = [("S", "sprint", 30), ("R", "reverse", 15)]
earlier_runs = [(2, "S", 27), (2, "R", 16), (3, "S", 33), (3, "R", 13)]
```

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



Do it on the robot

www.bugbotlab.com/learn/a11-10-project-the-run-database/

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

Challenges

1. Add each robot's best run at each challenge to the report, with a nested `SELECT` or a second `GROUP BY`.
2. A second robot logs runs to the same database at the same time. Which records could suffer a lost update, and which concurrency method would you choose for this system?
3. The organisers' system only accepts XML. Write the report as XML instead, and explain one advantage and one disadvantage of the change.

