



A11.6 Transactions and ACID

Databases and big data · A level · OCR H446 1.3.2, AQA 7517 4.10.5,
Eduqas A500QS 2.5 · about 55 min

Name _____

Class _____

Date _____

What this lesson is about

A transaction is all or nothing: atomicity, consistency, isolation and durability, commit and rollback, and redundancy.

- What a transaction is
- Half a change
- ACID
- Consistency: rules the database enforces
- A transaction around a physical action
- Record locking and redundancy

Questions

5 marks in all

- Which ACID property means a transaction happens completely or not at all? [1 mark]
 - ☐ A Atomicity
 - ☐ B Consistency
 - ☐ C Isolation
 - ☐ D Durability
- After COMMIT, a power cut hits the server. Which property guarantees the change is still there? [1 mark]
 - ☐ A Durability
 - ☐ B Atomicity
 - ☐ C Isolation
 - ☐ D Consistency
- Two transactions run at the same time, and neither sees the other's uncommitted changes. Which property is this? [1 mark]
 - ☐ A Isolation
 - ☐ B Durability
 - ☐ C Atomicity
 - ☐ D Redundancy

4. What does this print?

[1 mark]

```
import sqlite3
db = sqlite3.connect(":memory:")
db.execute("CREATE TABLE zones (name TEXT PRIMARY KEY, balls INTEGER CHECK (balls >= 0))")
db.executemany("INSERT INTO zones VALUES (?, ?)", [("red", 1), ("blue", 4)])
db.commit()
try:
    with db:
        db.execute("UPDATE zones SET balls = balls + 3 WHERE name = 'blue'")
        db.execute("UPDATE zones SET balls = balls - 3 WHERE name = 'red'")
except sqlite3.IntegrityError:
    print("rolled back")
print(db.execute("SELECT * FROM zones ORDER BY name").fetchall())
```

5. In OCR's transaction processing topic, what does redundancy refer to?

[1 mark]

- ☐ A Keeping extra copies, such as mirrored disks or replica servers, so a failure loses no data or service
- ☐ B Storing the same fact many times in one table
- ☐ C Deleting data that is no longer needed
- ☐ D Records that have no primary key

The task: all or nothing

The table `zones` holds each zone's `name` (text, the primary key) and its number of `balls` (a whole number), with the rule `CHECK (balls >= 0)`. It starts as red 3, blue 2, green 0. `moves` is a list of `(src, dst, n)`: two zone names and a whole number of balls. 1. Write `move(src, dst, n)` that, as **one transaction**, first adds `n` to `dst` with `UPDATE zones SET balls = balls + ?` and then takes `n` from `src`. If the database refuses the change with `sqlite3.IntegrityError`, the whole move must be undone: neither zone changes. Print `moved <n> <src> -> <dst>` if it worked, or `refused <n> <src> -> <dst>` if not. Do not test the numbers with an `if` of your own: let the `CHECK` rule refuse. 2. Call `move` for each item of `moves`, in order. 3. Print every zone in name order as `<name> <balls>`, then `total <sum>` using `SUM`. Eight lines in all. The robot does not move.

```
# 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 zones (name TEXT PRIMARY KEY, balls INTEGER NOT NULL CHECK (balls
>= 0))")
db.executemany("INSERT INTO zones VALUES (?, ?)", [("red", 3), ("blue", 2), ("green", 0)])
db.commit()

moves = [("red", "blue", 2), ("blue", "green", 5), ("blue", "green", 3), ("red", "green",
2)]
```

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



Do it on the robot

www.bugbotlab.com/learn/a11-6-transactions-and-acid/

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

Challenges

1. Take the `with db:` out of your `move`. Which line of output changes, and which ACID property has been lost?
2. A bank moves £50 from one account to another. Describe what each ACID property guarantees.
3. Why must a DBMS write to its log **before** it tells the user a transaction is committed?