Lecture 6: On ACID

COMP207

Minor updates to the assignment

- Fixed some minor errors:
  - eparture.sql -> epature.sql
    - It was actually meant to be eparture but to make the least amount of problems for students that have done some of the work I decided to consistently misspell it everywhere instead
  - 7 tables -> 6 tables
  - Forgot to sort in question 5
  - CodeGrade was missing a foreign key
  - Removed an additional copy of some rows of the output for question 7
  - Changed the birthday of Louise in question 3

Single queries -> transactions

- In the next 3 weeks, we will look at transactions
- They sound easy but they are somewhat complex
- Instead of doing one query at a time, we can do a sequence of queries one after the other
  - Query = insert/select/update/delete – can also do create/drop here but that's rare
- Transaction starts with START TRANSACTION; and ends when you write COMMIT; or ROLLBACK;

Transactions in MySQL

- Example of a transaction in MySQL…

ACID

- While it is conceptually simple, we want transactions to satisfy some properties:
  - A is for atomicity: either do the full transaction or nothing
  - C is for consistency: definitions range from must satisfy constraints to must match a real-world event
  - I is for isolation: the transactions should operate as if no other transactions are running at the same time
  - D is for durability: after commit has been done, things that happen later should not undo that

Atomicity in MySQL

- In essence, you satisfy Atomicity by unrolling transactions that did not finish
- (MySQL does it in a weird way)
- Example of atomicity in MySQL

Consistency in MySQL

- The easy version (can't break constraints): We have seen it before but we can try to break constraints easily enough – we just get errors
- The hard version (must reflect a real-world even): ???

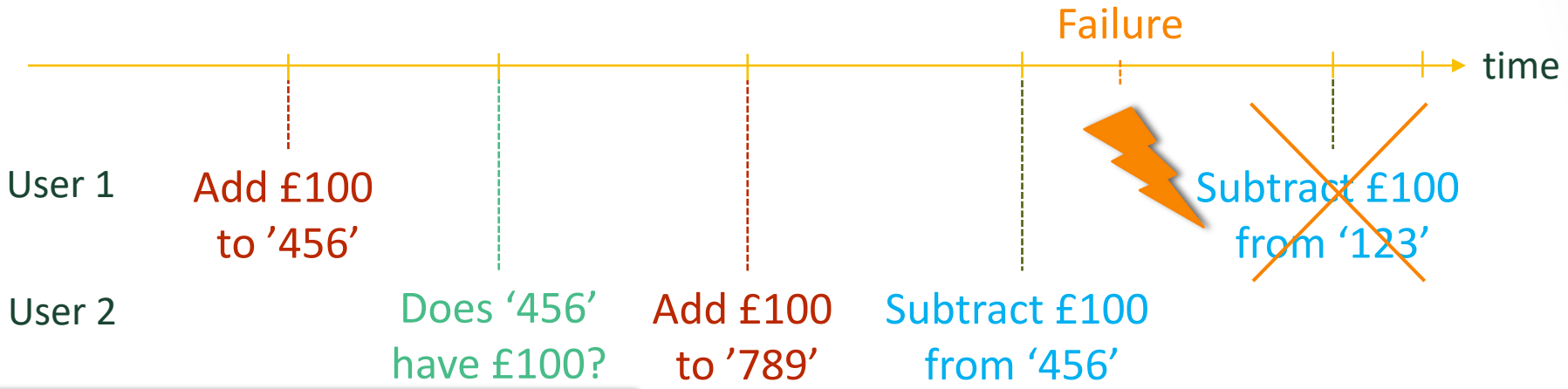
Isolation in MySQL

- There are 4 levels of how strongly you want isolation
- In this course, unless explicitly mentioned, the default is SERIALIZABLE, meaning the transactions should operate as if we were first running one transaction to completion then another and then another and so on
  - That said, formally, that requires the earlier levels, but for us, SERIALIZABLE does not require them
- There are 3 other levels: READ UNCOMMITTED, READ COMMITTED, REPEATABLE READS
  - They are quite easy to understand (I feel)
  - READ UNCOMMITTED means that you can do whatever (in this course – MySQL does have some constraints on it)
  - READ COMMITTED that you can only read (think: SELECT) things that have been committed
  - REPEATABLE READS means that if you read one thing once and try to read the same thing again later, it will still be there
- SET GLOBAL TRANSACTION ISOLATION LEVEL <level>;

Durability in MySQL

- I do not really have a way of showing this directly as an example
- I will look at what happens if we try to run the example from the videos

Problem 3: Concurrency & Partial Execution




```
UPDATE Accounts
SET    balance = balance + 100
WHERE  accountNo = 456;
```

```
UPDATE Accounts
SET    balance = balance - 100
WHERE  accountNo = 456;
```

```
SELECT balance
FROM   Accounts
WHERE  accountNo = 456;
```

```
UPDATE Accounts
SET    balance = balance + 100
WHERE  accountNo = 789;
```

```
UPDATE Accounts
SET    balance = balance - 100
WHERE  accountNo = 123;
```

Short-hand notation

- Show examples of short-hand notation