Good schedules/transactions

Overview of this video

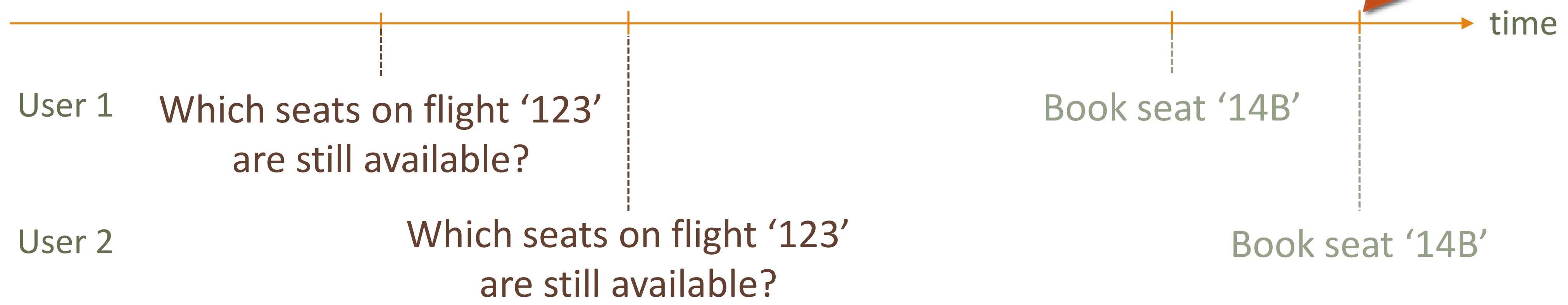
We will try to characterize when a schedule/transaction is bad

Problem 1: Concurrency

(from the introduction to transactions video)

Flights(flightNo, date, seatNo, seatStatus)

Might lead to an inconsistent database


time
User 1
Which seats on flight '123' are still available?
Book seat '14B'
User 2
Which seats on flight '123' are still available?
Book seat '14B'

```
SELECT seatNo
FROM   Flights
WHERE  flightNo = 123
   AND date = '2020-10-30'
   AND seatStatus = 'available';
```

```
UPDATE Flights
SET    seatStatus = 'occupied'
WHERE  flightNo = 123
   AND date = '2020-10-30'
   AND seatNo = '14B';
```

What are the problems?

There is an issue with how the two transactions interact

- Specifically, the two transactions are not **isolated** from each other
- Alternately, the outcome is not **consistent** with the real world

What could we do to solve it?

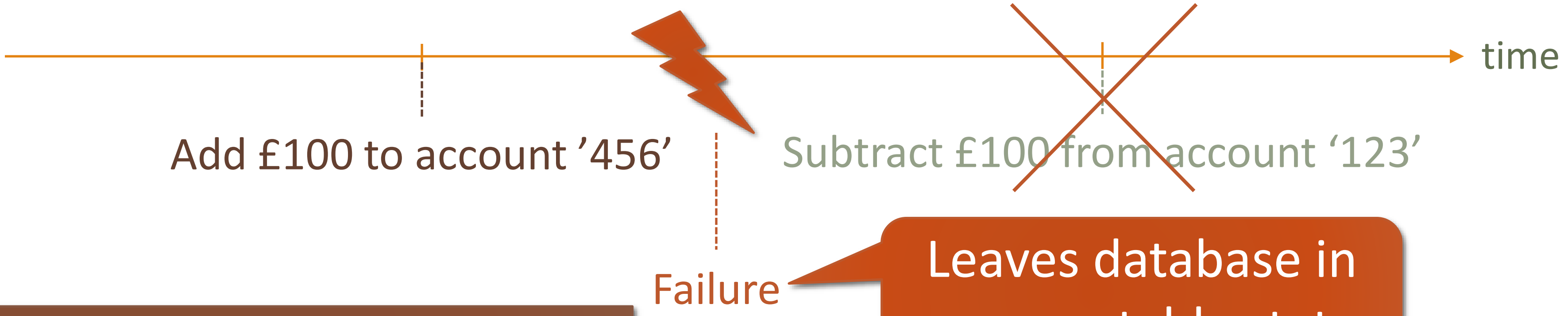
- We could just undo the second transaction when it tries to do the second operation

Problem 2: Partial Execution

(from the introduction to transactions video)

Accounts(accountNo, accountHolder, balance)

Goal: Transfer £100 from account ‘123’ to account ‘456’


time
Add £100 to account ’456’
~~Subtract £100 from account ‘123’~~
Failure

Leaves database in unacceptable state

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

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

What is the problem?

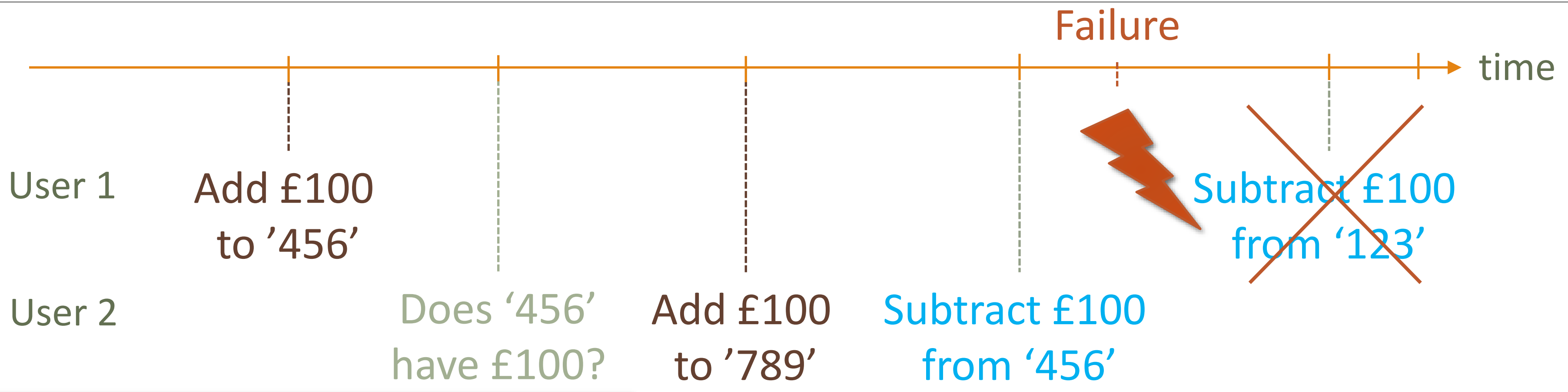
There is an issue with how only parts of the transaction gets executed

- A transaction should be **atomic** (=indivisible)
- Maybe not **consistent** with the real world, depending on how you define that…

What could we do to solve it?

- We could just undo the transaction when the computer comes back online

Problem 3: Currency & Partial Execution



```
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 = 456;
```

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

What are the problems?

There is an issue with how only parts of the first transaction gets executed

- A transaction should be **atomic** (=indivisible)

There is an issue with how the two transactions interact

- There could be some problems with whether it is **consistent** (i.e. the check user 2 did should perhaps not have succeed, because user 1 had not committed yet)
- Also, the two transactions are not **isolated** from each other

What could we do to solve it?

- The short version is that there are no nice way to solve this (next slide has the long…)

No good solutions to the problem

This is the long version!

Let us look at some options:

- We could do nothing, but then the bank lost money
  - This would be the **atomicity** issue
- We could undo the first transaction, but not the second, but then the second transaction might not be valid anymore, because there could be too little money on the account to transfer 100£
  - This would also give that **consistency** issue
  - This would be the **isolation** issue
- We could undo both transactions, but the second one have finished and the person doing it might have already gone away (because everything looked good when he finished)
  - This is a new issue! We would expect that any change you made was **durable** (i.e. did not latter disappear after having finished/disappear based on something someone else did) in the database

Summary - The ACID properties

These mentioned issues are together called the ACID properties in databases, i.e.

A: Atomicity

C: Consistency

I: Isolation

D: Durability (or permanency)