Welcome to Transactions and Error Handling in PostgreSQL

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Jason Myers
Principal Engineer

datacamp

Learning objectives and datasets

- Maintain data integrity and credibility with transaction
- Concurrency's effects on transactions
- Error Handling with transactions

Motivation for using transactions

`patient_intake`

name	priority
Oscar Parker	1
Prisha Ahmed	2
Rhea Taylor	3

```
UPDATE patient_intake SET priority=1
WHERE name='Prisha Ahmed';

UPDATE patient_intake SET priority=2
WHERE name='Oscar Parker';
```

Errored table results

`patient_intake`

name	priority
Oscar Parker	1
Prisha Ahmed	1
Rhea Taylor	3

Using a transaction

```
BEGIN;

UPDATE patient_intake SET priority=1
WHERE name='Prisha Ahmed';

UPDATE patient_intake SET priority=2
WHERE name='Oscar Parker';

COMMIT;
```

Another reason to use transactions

- Multiple statements that should succeed or fail as a group
- Handling how statements are affected by concurrent operations

Transaction blocks

```
BEGIN TRANSACTION;

UPDATE patient_intake SET priority=1
WHERE name='Prisha Ahmed';

UPDATE patient_intake SET priority=2
WHERE name='Oscar Parker';

COMMIT;
```

Any number of statements

```
BEGIN;

UPDATE patient_intake SET priority=2
WHERE name='Oscar Parker';

UPDATE patient_intake SET priority=1
WHERE name='Rhea Taylor';

UPDATE patient_intake SET priority=3
WHERE name='Prisha Ahmed';

COMMIT;
```

Let's practice!

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Transaction sizes and PostgreSQL protections

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Jason Myers
Principal Engineer

datacamp

Keep transactions small

- Easier to reason about
- Database Performance
- Error domains

Dividing operations

```
BEGIN TRANSACTION;

INSERT sales SET quantity=6
WHERE name='chocolate chip';

INSERT baking_list SET quantity=12
WHERE name='chocolate chip';

UPDATE inventory SET quantity = quantity - 6
WHERE name='oatmeal dark chocolate';

COMMIT;
```

Is this a transaction?

```
UPDATE cookies SET deliciousness = 11 WHERE name = 'Ginger Molasses';
```

What about selects?

```
SELECT deliciousness FROM cookies where name = 'ANZAC';
```

Let's practice!

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Isolation levels

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

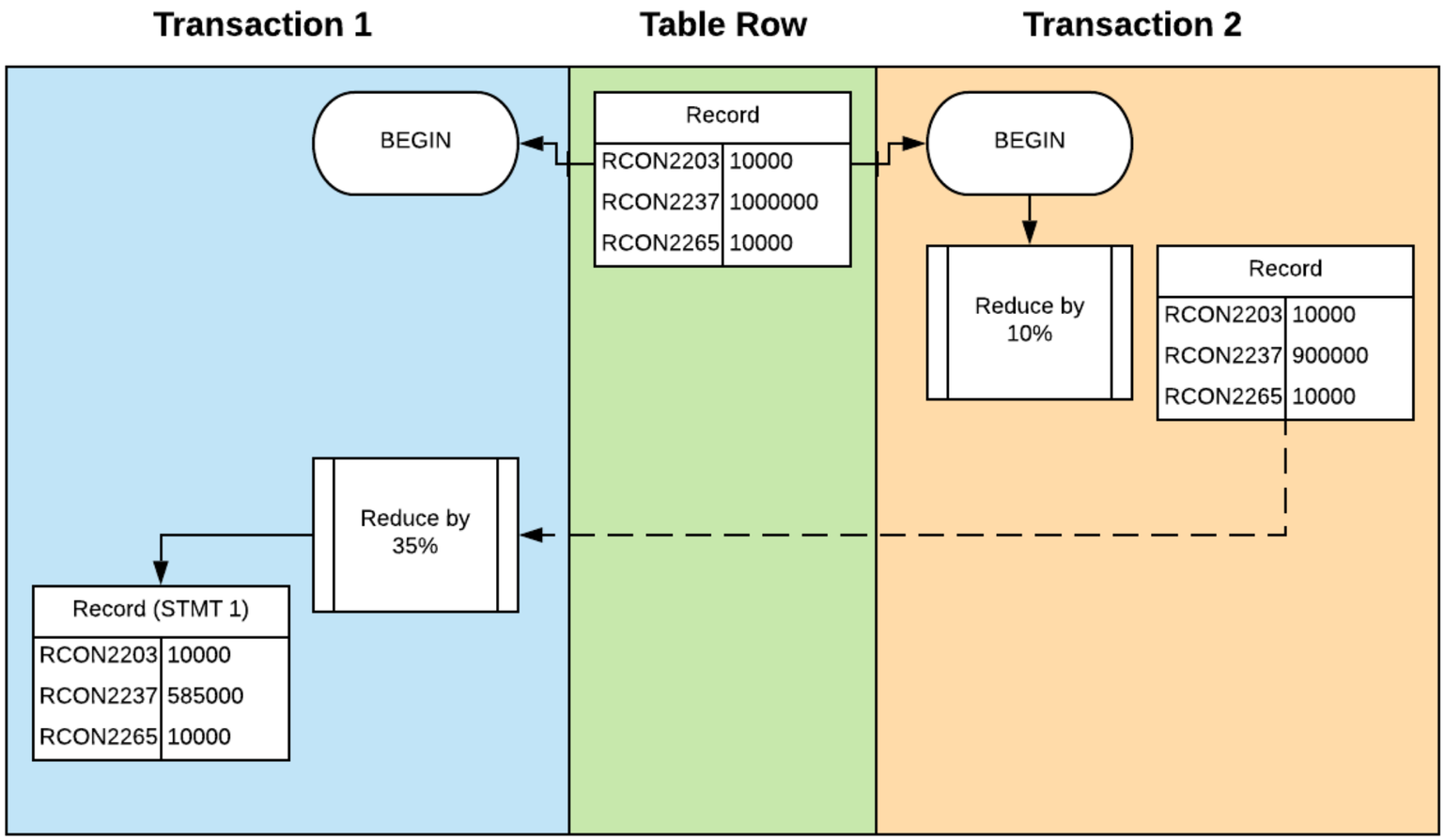
Jason Myers
Instructor

datacamp

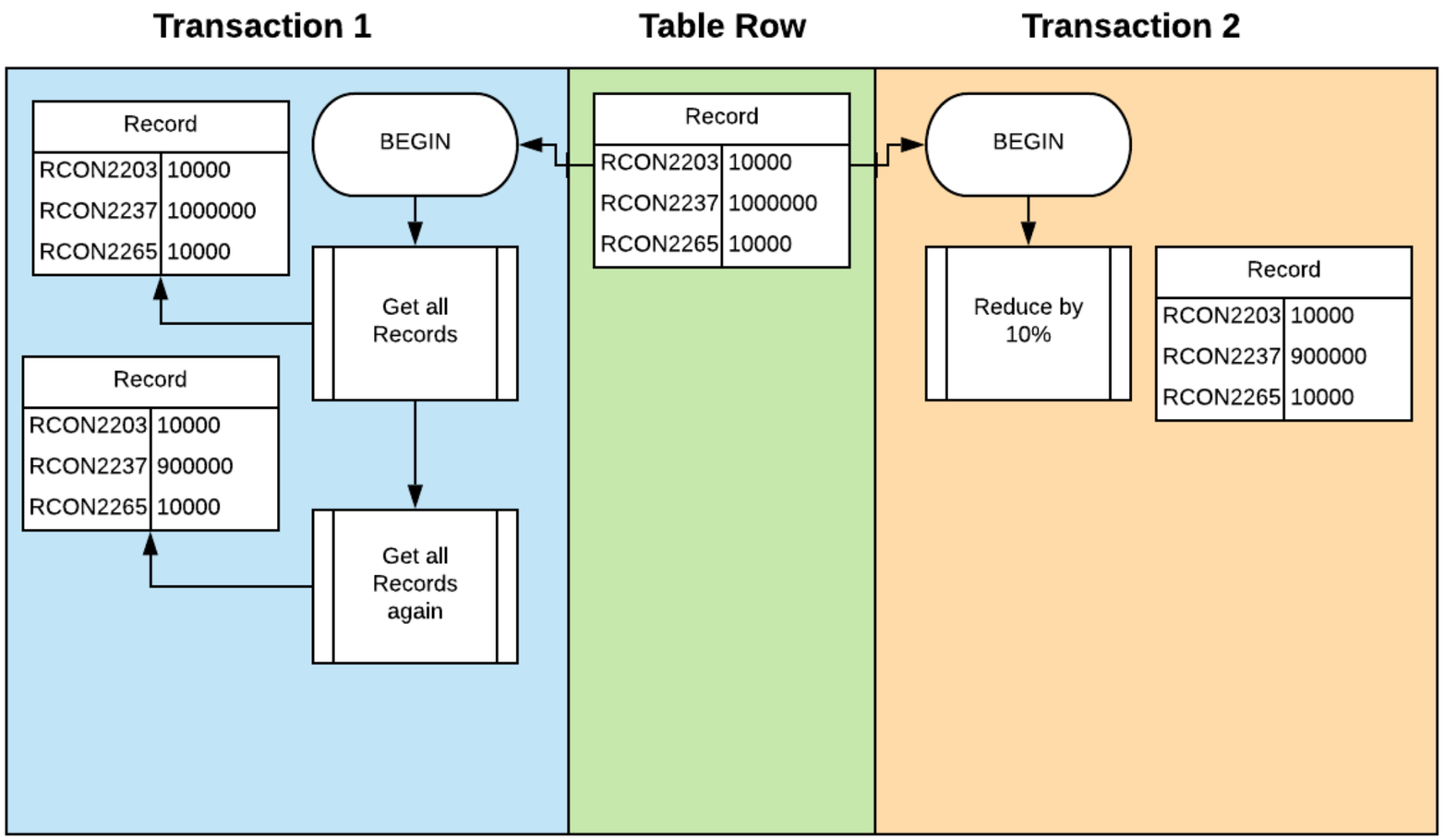
Concurrency

- Handling the coordination of multiple operations at the same time.
- Has a unique set of issues in a database.
- Uses rules known as isolation levels to make it easier to reason about outcomes.

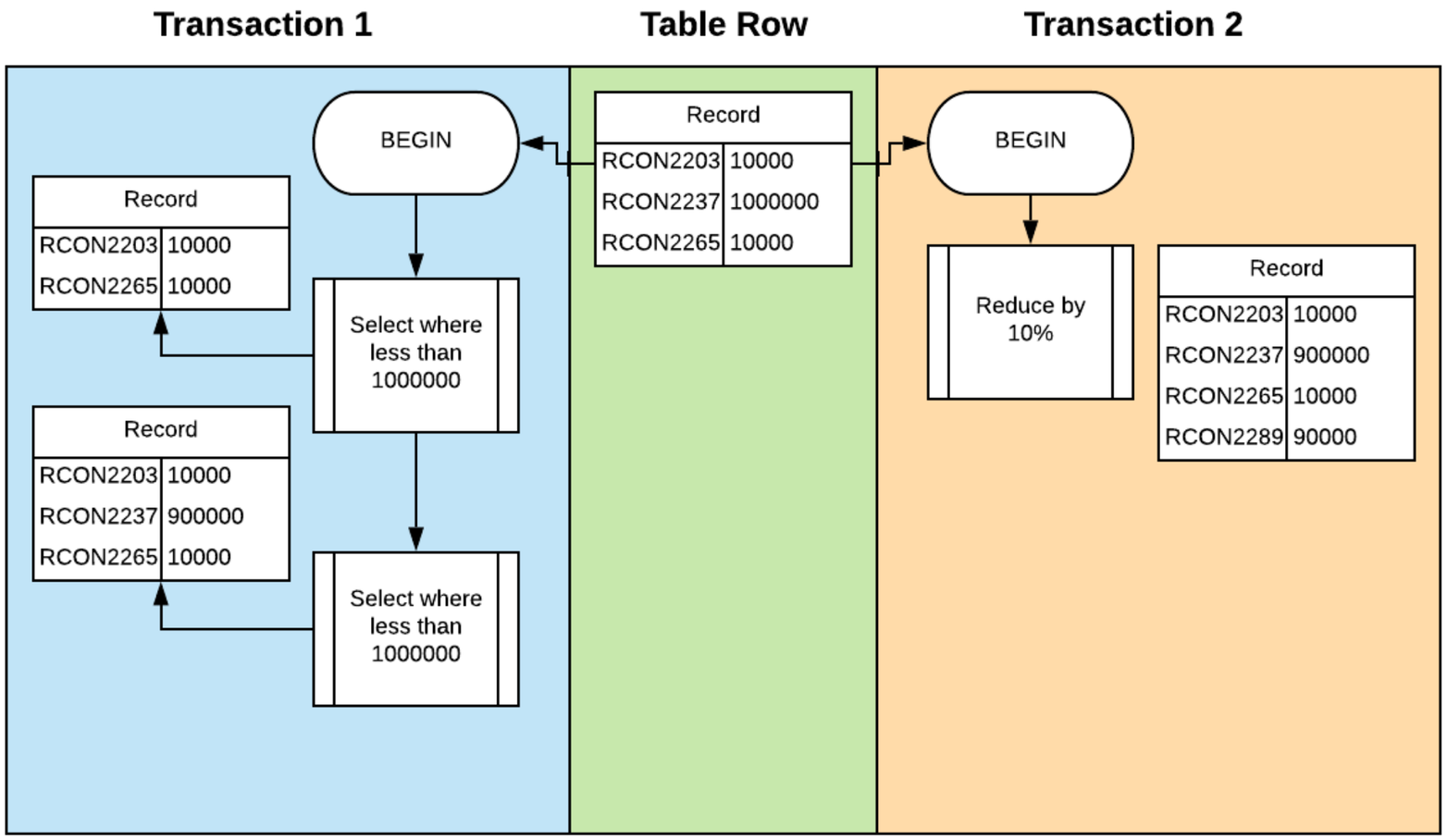
Dirty Reads


Transaction 1
Table Row
Transaction 2
BEGIN
Record
RCON2203 10000
RCON2237 1000000
RCON2265 10000
BEGIN
Reduce by 10%
Record
RCON2203 10000
RCON2237 900000
RCON2265 10000
Reduce by 35%
Record (STMT 1)
RCON2203 10000
RCON2237 585000
RCON2265 10000

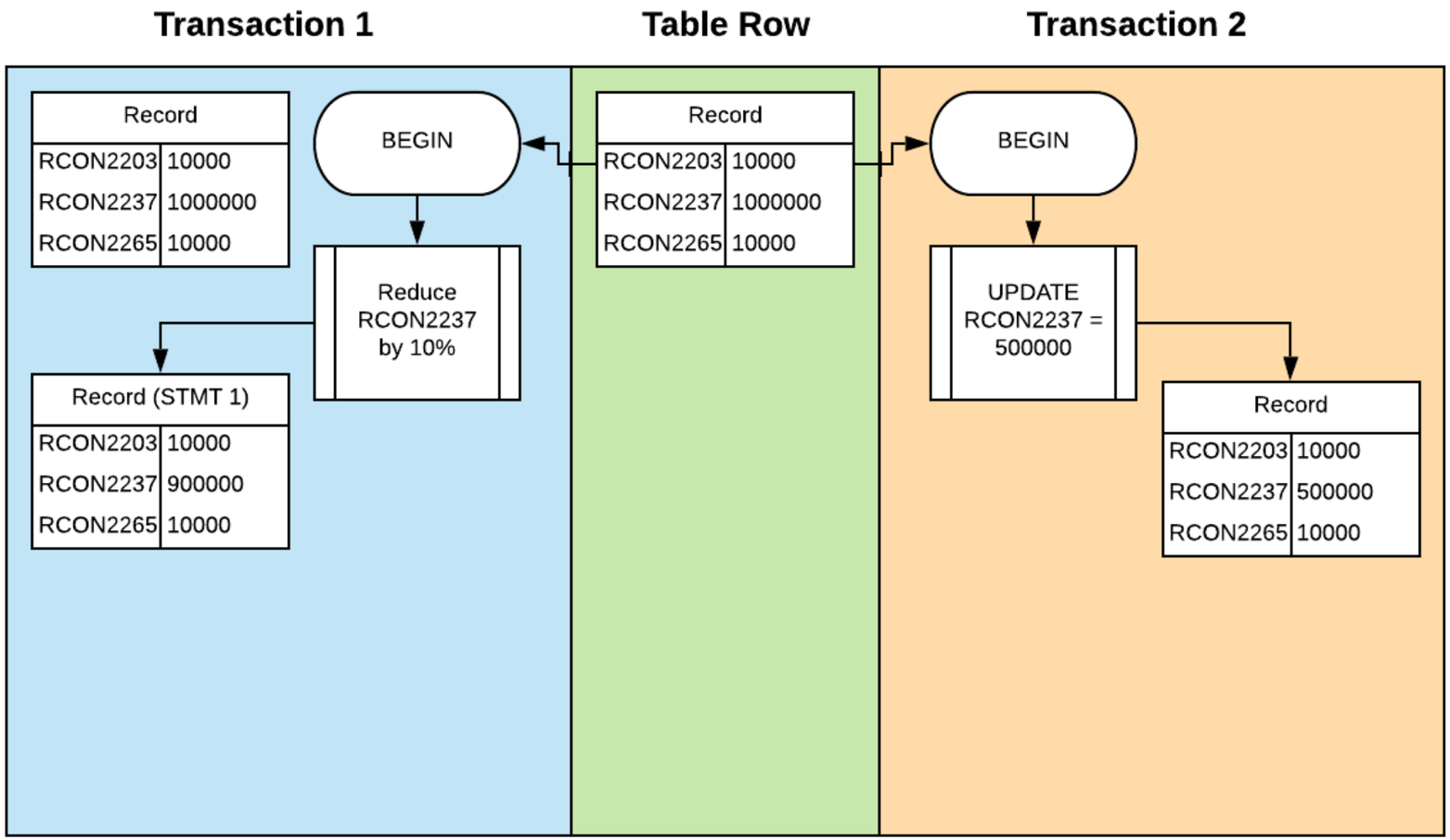
Nonrepeatable Read



Phantom Read



Serialization Anomaly



Isolation levels

Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read	Serialization Anomaly
Read Uncommitted	Protected (PostgreSQL)	vulnerable	vulnerable	vulnerable
Read Committed	Protected	vulnerable	vulnerable	vulnerable
Repeatable Read	Protected	Protected	Protected (PostgreSQL)	vulnerable
Serializable	Protected	Protected	Protected	Protected

Affects of isolation levels

```
START TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';

-- Cookie 6 has been added in an external transaction.

SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
COMMIT;
```

Statement results

- First SELECT result: 5
- Second SELECT result: 6

Affects of isolation levels

```
START TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';

-- Cookie 6 has been added in an external transaction.

SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
COMMIT;
```

Statement results

- First SELECT result: 5
- Second SELECT result: 5

Let's practice!

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL