Handling mistakes

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Jason Myers
Principal Engineer

datacamp

Mistakes happen

Mistakenly updated Biscuits

```
UPDATE cookies SET quantity = 13 WHERE name = 'Biscuits';
```

Corrected to update Biscotti

```
UPDATE cookies SET quantity = 1 WHERE name = 'Biscuits';

UPDATE cookies SET quantity = 13 WHERE name = 'Biscotti';
```

Rolling back mistakes

```
BEGIN TRANSACTION

UPDATE cookies SET quantity = 13 WHERE name = 'Biscuits';
```

```
ROLLBACK;
```

```
SELECT quantity FROM cookies where name = 'Biscuits';
```

```
13
```

Rollback multiple statements

```
BEGIN TRANSACTION;

UPDATE cookies SET deliciousness = 111 where name = 'Cats Tongue';

UPDATE cookies SET deliciousness = 8 where name = 'Gingerbread';

ROLLBACK;
```

```
SELECT name, deliciousness FROM cookies where name in ('Cats Tongue', 'Gingerbread');
```

```
'Cats Tongue'  10
'Gingerbread'   9
```

Let's practice!

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Rolling back to a savepoint

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Jason Myers
Principal Engineer

Without savepoints

```
BEGIN TRANSACTION;

UPDATE cost = 2.33 WHERE name = 'Linga';

UPDATE cost = 500 WHERE name = 'Macaron';

ROLLBACK;
```

Using Savepoints and targeted rollbacks

```
BEGIN TRANSACTION;

UPDATE cost = 2.33 WHERE name = 'Linga';

SAVEPOINT oops;

UPDATE cost = 500 WHERE name = 'Macaron';

ROLLBACK TO oops;
```

Releasing a savepoint when done with it

```
BEGIN TRANSACTION;

UPDATE cost = 2.33 WHERE name = 'Linga';
UPDATE cost = 2.33 WHERE name = 'Petit-Beurre';
UPDATE cost = 2.33 WHERE name = 'Rosette';

SAVEPOINT oops;

UPDATE cost = 5.00 WHERE name = 'Macaron';
UPDATE cost = 3.50 WHERE name = 'Panellets';
```

```
RELEASE SAVEPOINT oops;
```

Two critical things about rollbacks and savepoints

- `ROLLBACK` without a TO will rollback the whole transaction
- `ROLLBACK TO ___` where `___` is not a valid savepoint name will cause an error

Let's practice!

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Multiple savepoints and rollback

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Jason Myers
Principal Engineer

datacamp

Multiple savepoints

```
BEGIN TRANSACTION;

UPDATE inventory SET quantity = quantity - 1 WHERE name in ('flour', 'sugar');

SAVEPOINT inventory_step;

INSERT baking_list SET quantity=12 WHERE name='Torun';

SAVEPOINT queuing_step;

UPDATE cookies SET quantity = 12 WHERE name = 'Torun';
```

Duplicating savepoint names

- A new SAVEPOINT with the same name as a prior one in the same transaction shadows it instead of overwritting or releasing it!
- Generally avoid reusing names within a transaction.

Duplicate savepoint name example

```
BEGIN TRANSACTION;

UPDATE inventory SET quantity = quantity - 1 WHERE name in ('flour', 'sugar');

SAVEPOINT oops;

INSERT baking_list SET quantity=12 WHERE name='Torun';

SAVEPOINT oops;

UPDATE cookies SET quantity = 12 WHERE name = 'Torun';
```

Let's practice!

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Isolation levels, savepoints, and rollbacks

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Jason Myers
Instructor

datacamp

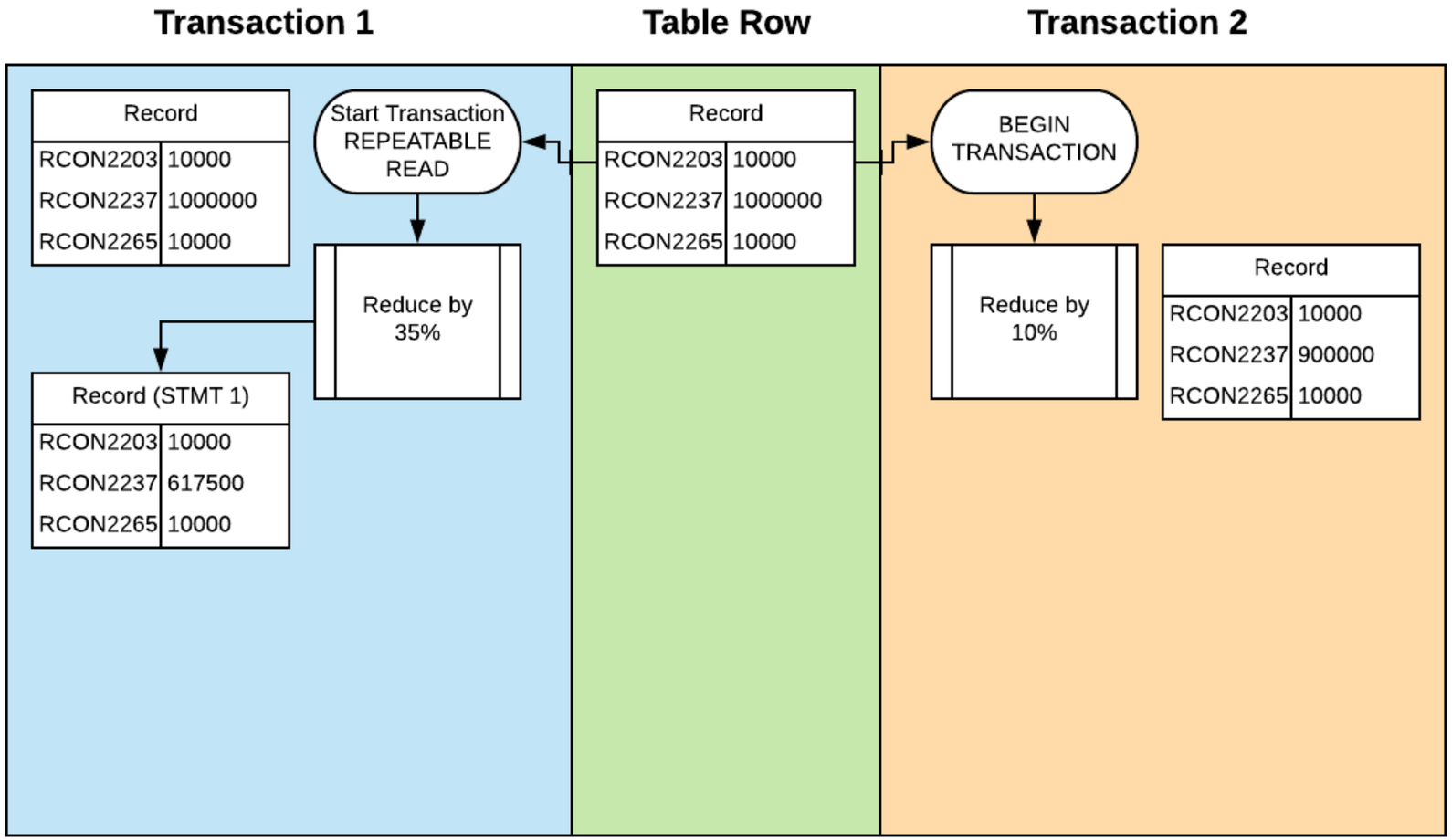
Isolation Levels

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

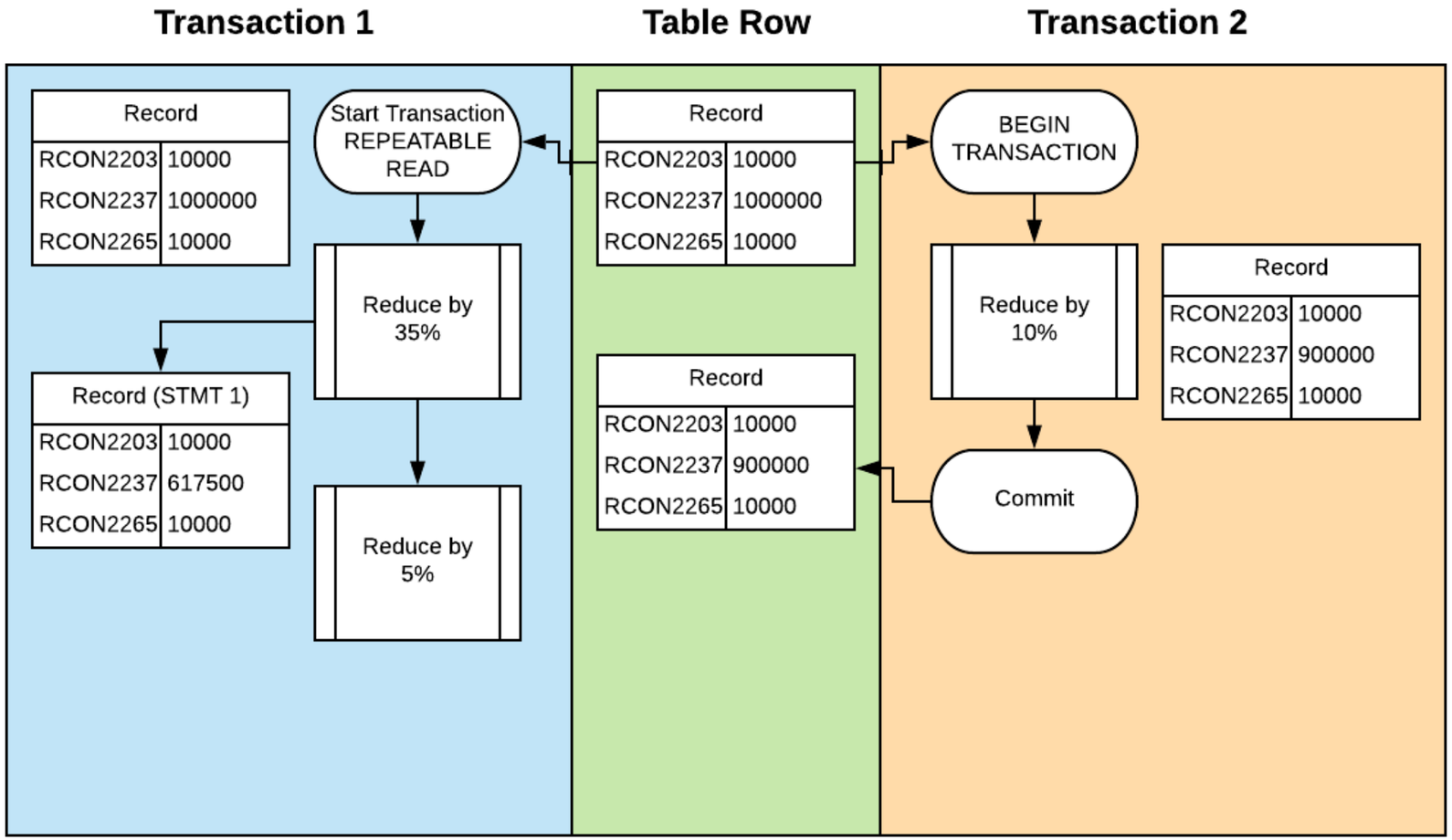
Repeatable Read

- **SERIALIZABLE**
  - Emulates serial transaction execution for all committed transactions.
- **REPEATABLE READ**
  - Sees data
    - committed before the transaction began
    - results of previous statements in the transaction
  - Does not sees any changes committed by concurrent transactions.

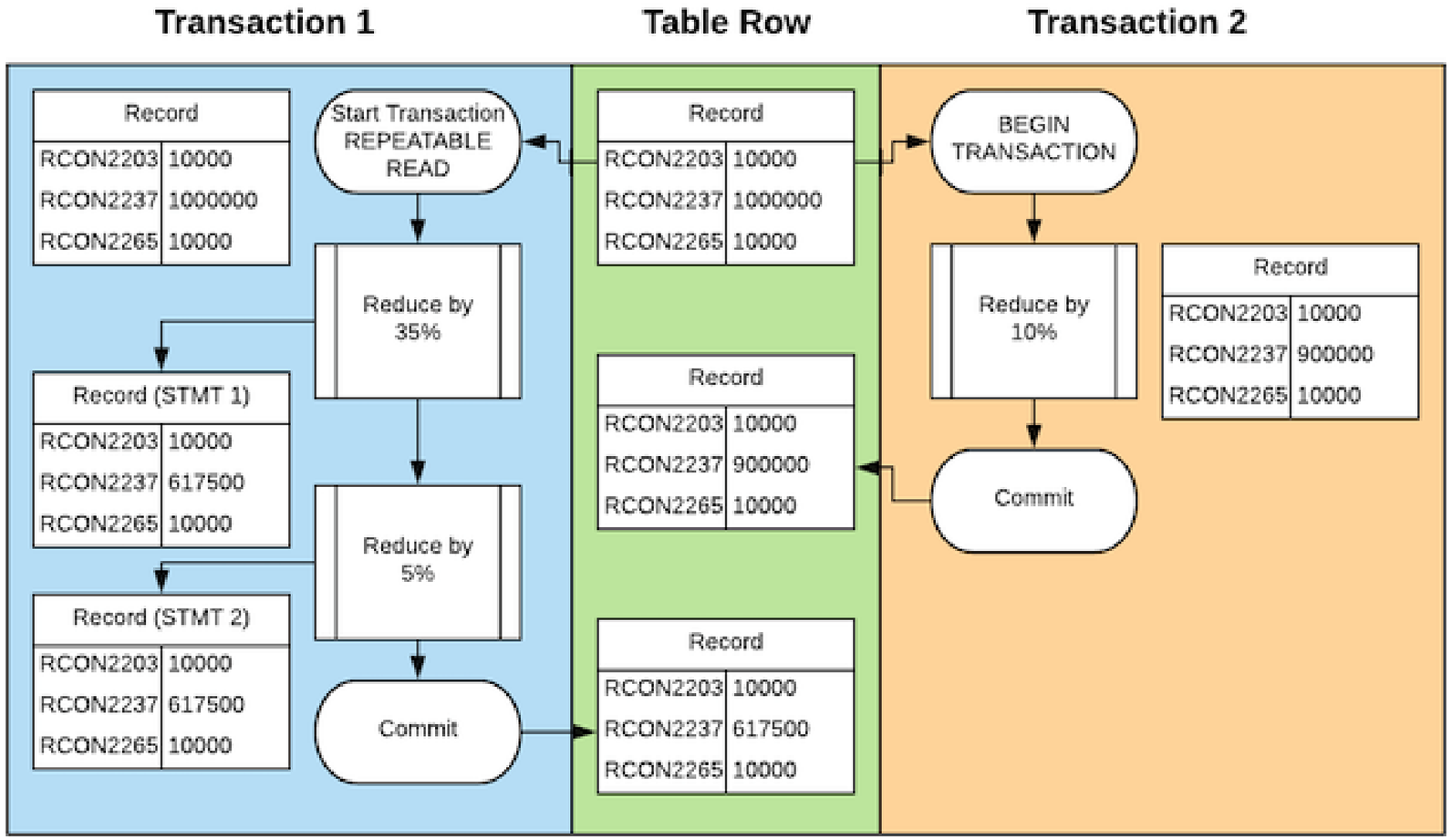
Visualizing REPEATABLE READ



Visualizing REPEATABLE READ



Visualizing REPEATABLE READ



REPEATABLE READ

```
START TRANSACTION ISOLATION LEVEL REPEATABLE READ;
UPDATE inventory
SET quantity = quantity - 4
WHERE name = 'macaron';
SAVEPOINT first;
UPDATE inventory
SET quantity = quantity - 12
SAVEPOINT second;

COMMIT;
```

Let's practice!

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL