How to Stop Worrying by using Transactions and Exceptions

When all them pending transactions hit at once.

Source: ladiesandlemo...

Postgres Transactions

- PostgreSQL ***REALLY*** focuses on providing safety and sanity

Two main places

- Statement Blocks
- Concurrent Scenarios

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

```sql
BEGIN;

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

COMMIT;
```

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

Statement Block breakdown

```
UPDATE cookies SET quantity = 2
WHERE name = 'ANZAC';

INSERT INTO to_bake_list (name, quantity)
VALUES ('ANZAC ', 12);
```

Statement Block breakdown

```
UPDATE cookies SET quantity = 2
WHERE name = 'ANZAC';
```

💥

```
INSERT INTO to_bake_list (name, quantity)
VALUES ('ANZAC ', 12);
```

wHy DoN't I jUsT dO tHiS

iN pYtHoN wItH tRy/CaTcH?

Statement Block Transaction

```
BEGIN;

UPDATE cookies SET quantity = 2
WHERE name = 'ANZAC';

INSERT INTO to_bake_list (name, quantity)
VALUES ('ANZAC ', 12);

COMMIT;
```

Nested Transactions

Savepoints

```
BEGIN;
    UPDATE cookies SET quantity = 2
    WHERE name = 'ANZAC';

    INSERT INTO to_bake_list (name, quantity)
    VALUES ('ANZAC ', 12);

    SAVEPOINT inventory_removal;

    INSERT INTO ingredients (name) VALUES ('flour');
    RELEASE SAVEPOINT inventory_removal;
COMMIT;
```

```sql
BEGIN;
    UPDATE cookies SET quantity = 2
    WHERE name = 'ANZAC';

    INSERT INTO to_bake_list (name, quantity)
    VALUES ('ANZAC ', 12);

    SAVEPOINT inventory_removal;

    INSERT INTO ingredients (name) VALUES ('flour');
    ROLLBACK TO SAVEPOINT inventory_removal;
COMMIT;
```

MULTITHREADING

THEORY AND PRACTICE

dirty read

- A transaction reads data written by a concurrent uncommitted transaction.

nonrepeatable read

- A transaction re-reads data it has previously read and finds that data has been modified by another transaction (that committed since the initial read).

phantom read

- A transaction re-executes a query returning a set of rows that satisfy a search condition and finds that the set of rows satisfying the condition has changed due to another recently-committed transaction.

serialization anomaly

- The result of successfully committing a group of transactions is inconsistent with all possible orderings of running those transactions one at a time.

When all them pending transactions hit at once.

Source: ladiesandlemo...

Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read	Serialization Anomaly
Uncommitted	🔴	✅	✅	✅
Committed	⛔	✅	✅	✅
Repeatable	⛔	⛔	🔴	✅
Serializable	⛔	⛔	⛔	⛔

```
START TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     5
(1 row)

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

SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     6
(1 row)
COMMIT;
```

```
START TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     5
(1 row)

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

SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     6
(1 row)
COMMIT;
```

```
START TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     5
(1 row)

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

SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     6
(1 row)
COMMIT;
```

```
START TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     5
(1 row)

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

SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     6
(1 row)
COMMIT;
```

```
START TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     5
(1 row)

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

SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     6
(1 row)
COMMIT;
```

```
START TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     5
(1 row)

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

SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     6
(1 row)
COMMIT;
```

SERIALIZABLE

- Emulates serial transaction execution for all committed transactions.

REPEATABLE READ

- Sees data committed before the transaction began
- Sees results of previous statements in the transaction
- Does not sees any changes committed by concurrent transactions.

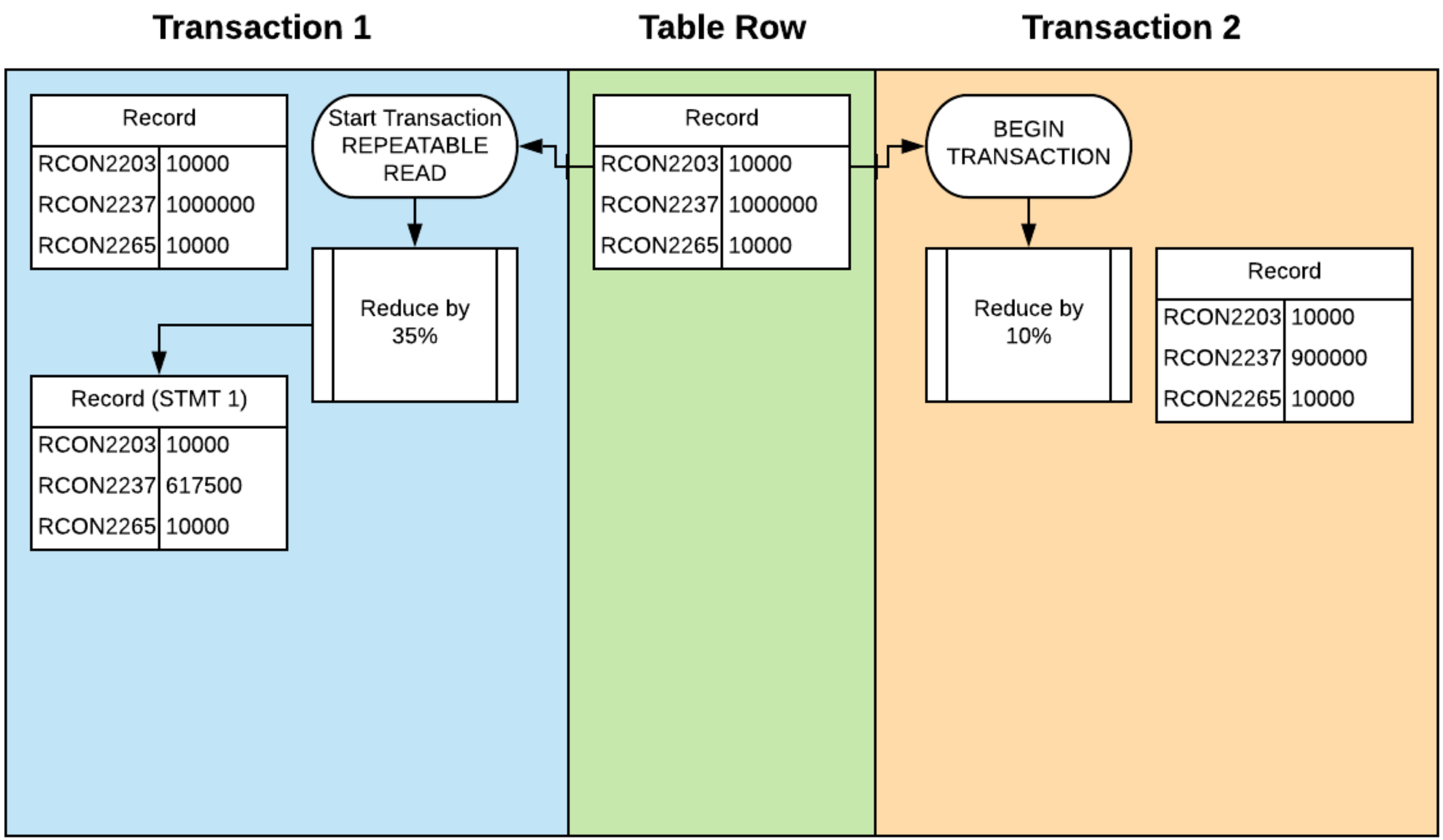
```sql
START TRANSACTION ISOLATION LEVEL REPEATABLE READ;

UPDATE ffiec_reci
SET RCON2237 = CAST(RCON2237 AS FLOAT) * .65
WHERE CAST(RCON2237 AS FLOAT) > 1000000;
SAVEPOINT first;

UPDATE ffiec_reci
SET RCON2237 = CAST(RCON2237 AS FLOAT) * .95
SAVEPOINT second;

COMMIT;
```

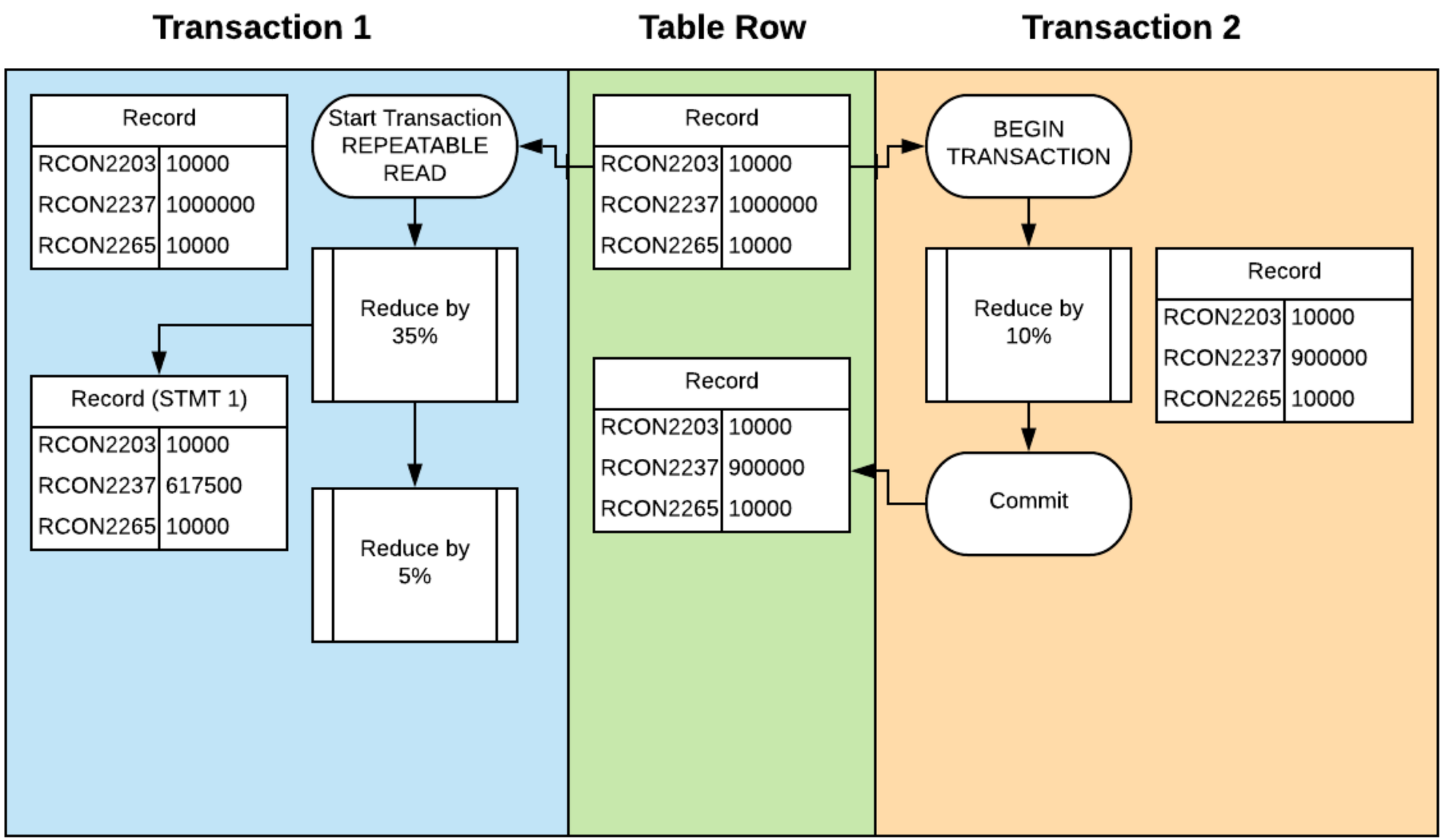
```sql
START TRANSACTION ISOLATION LEVEL REPEATABLE READ;

UPDATE ffiec_reci
SET RCON2237 = CAST(RCON2237 AS FLOAT) * .65
WHERE CAST(RCON2237 AS FLOAT) > 1000000;
SAVEPOINT first;

UPDATE ffiec_reci
SET RCON2237 = CAST(RCON2237 AS FLOAT) * .95
SAVEPOINT second;

COMMIT;
```

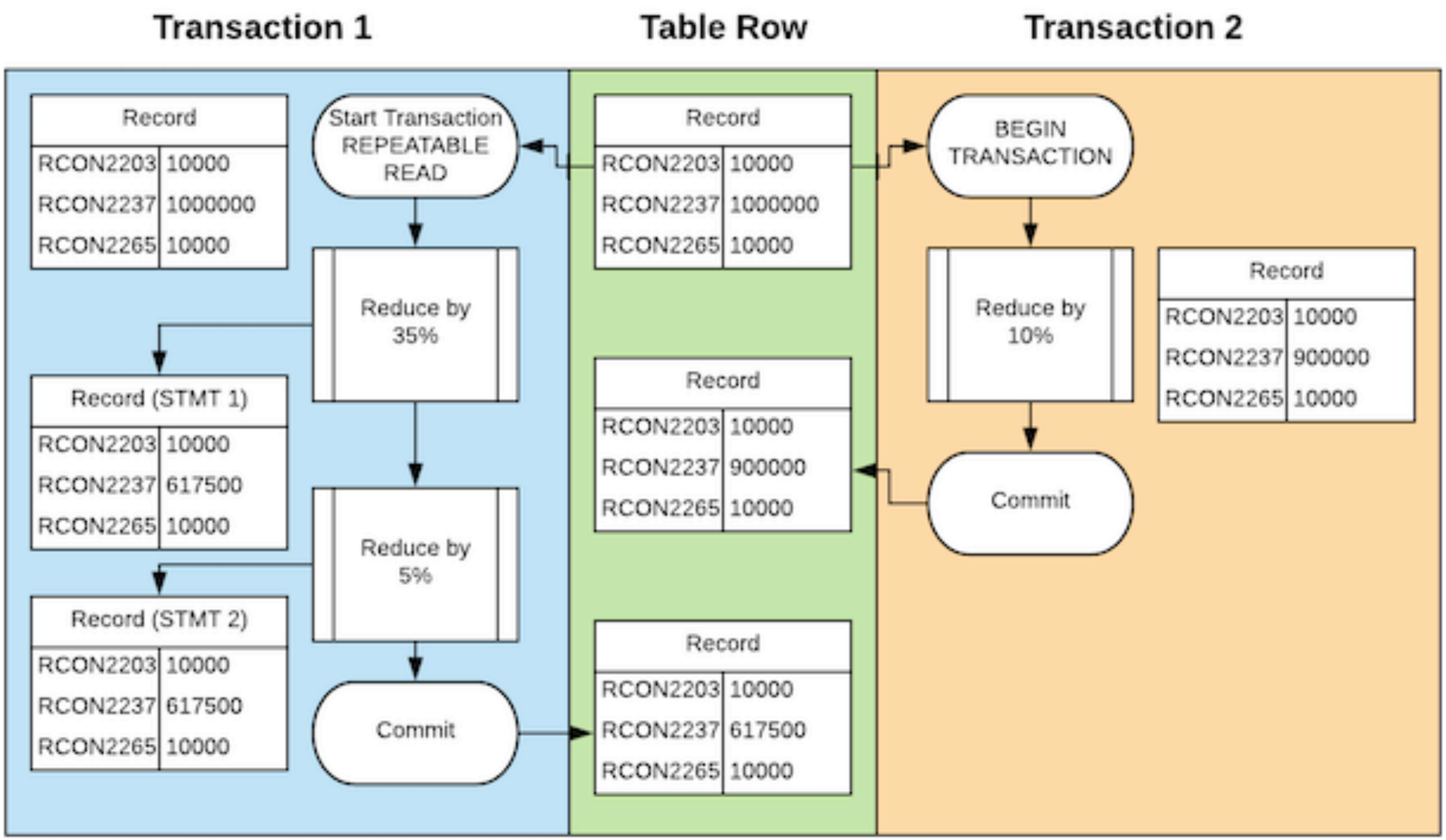
```sql
START TRANSACTION ISOLATION LEVEL REPEATABLE READ;

UPDATE ffiec_reci
SET RCON2237 = CAST(RCON2237 AS FLOAT) * .65
WHERE CAST(RCON2237 AS FLOAT) > 1000000;
SAVEPOINT first;

UPDATE ffiec_reci
SET RCON2237 = CAST(RCON2237 AS FLOAT) * .95
SAVEPOINT second;

COMMIT;
```

```sql
START TRANSACTION ISOLATION LEVEL REPEATABLE READ;

UPDATE ffiec_reci
SET RCON2237 = CAST(RCON2237 AS FLOAT) * .65
WHERE CAST(RCON2237 AS FLOAT) > 1000000;
SAVEPOINT first;

UPDATE ffiec_reci
SET RCON2237 = CAST(RCON2237 AS FLOAT) * .95
SAVEPOINT second;

COMMIT;
```

```sql
START TRANSACTION ISOLATION LEVEL REPEATABLE READ;

UPDATE ffiec_reci
SET RCON2237 = CAST(RCON2237 AS FLOAT) * .65
WHERE CAST(RCON2237 AS FLOAT) > 1000000;
SAVEPOINT first;

UPDATE ffiec_reci
SET RCON2237 = CAST(RCON2237 AS FLOAT) * .95
SAVEPOINT second;

COMMIT;
```


Transaction 1
Table Row
Transaction 2
Record
RCON2203 10000
RCON2237 1000000
RCON2265 10000
Start Transaction REPEATABLE READ
Reduce by 35%
Record (STMT 1)
RCON2203 10000
RCON2237 617500
RCON2265 10000
Record
RCON2203 10000
RCON2237 1000000
RCON2265 10000
BEGIN TRANSACTION
Reduce by 10%
Record
RCON2203 10000
RCON2237 900000
RCON2265 10000


Transaction 1
Table Row
Transaction 2
Record
RCON2203 10000
RCON2237 1000000
RCON2265 10000
Start Transaction REPEATABLE READ
Reduce by 35%
Record (STMT 1)
RCON2203 10000
RCON2237 617500
RCON2265 10000
Reduce by 5%
Record
RCON2203 10000
RCON2237 1000000
RCON2265 10000
Record
RCON2203 10000
RCON2237 900000
RCON2265 10000
BEGIN TRANSACTION
Reduce by 10%
Commit
Record
RCON2203 10000
RCON2237 900000
RCON2265 10000


Transaction 1
Table Row
Transaction 2
Record
RCON2203 10000
RCON2237 1000000
RCON2265 10000
Start Transaction REPEATABLE READ
Reduce by 35%
Record (STMT 1)
RCON2203 10000
RCON2237 617500
RCON2265 10000
Reduce by 5%
Record (STMT 2)
RCON2203 10000
RCON2237 617500
RCON2265 10000
Commit
Record
RCON2203 10000
RCON2237 1000000
RCON2265 10000
Record
RCON2203 10000
RCON2237 900000
RCON2265 10000
Record
RCON2203 10000
RCON2237 617500
RCON2265 10000
BEGIN TRANSACTION
Reduce by 10%
Commit
Record
RCON2203 10000
RCON2237 900000
RCON2265 10000

```
START TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     5
(1 row)

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

SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     5
(1 row)
COMMIT;
```

```
START TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     5
(1 row)

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

SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     5
(1 row)
COMMIT;
```

```
START TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     5
(1 row)

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

SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     5
(1 row)
COMMIT;
```

```
START TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     5
(1 row)

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

SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     5
(1 row)
COMMIT;
```

```
START TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     5
(1 row)

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

SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     5
(1 row)
COMMIT;
```

```
START TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     5
(1 row)

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

SELECT COUNT(*) FROM cookies WHERE name = 'lemon drop';
 count
-------
     5
(1 row)
COMMIT;
```

Detour

Anonymous DO Funtions

```
DO $$
DECLARE
    --- Variables
BEGIN
    --- Statements
END; $$ language 'plpgsql';
```

```
DO $$
DECLARE
    --- Variables
BEGIN
    --- Statements
END; $$ language 'plpgsql';
```

```
DO $$
DECLARE
    --- Variables
BEGIN
    --- Statements
END; $$ language 'plpgsql';
```

```
DO $$
DECLARE
    --- Variables
BEGIN
    --- Statements
END; $$ language 'plpgsql';
```

```
DO $$
DECLARE
    --- Variables
BEGIN
    --- Statements
END; $$ language 'plpgsql';
```

Exception Handling

Exception Example

```
DO $$f
DECLARE
    v_msg TEXT;
BEGIN
     INSERT INTO patients (a1c, glucose, fasting) values (20, 800, False);
EXCEPTION WHEN check_violation THEN
    v_msg = 'This A1C is not valid, should be between 0-13';
    INSERT INTO errors (msg) VALUES (v_msg);
END; $$ language 'plpgsql';
```

Exception Example

```
DO $$f
DECLARE
    v_msg TEXT;
BEGIN
     INSERT INTO patients (a1c, glucose, fasting) values (20, 800, False);
EXCEPTION WHEN check_violation THEN
    v_msg = 'This A1C is not valid, should be between 0-13';
    INSERT INTO errors (msg) VALUES (v_msg);
END; $$ language 'plpgsql';
```

Exception Example

```
DO $$f
DECLARE
    v_msg TEXT;
BEGIN
     INSERT INTO patients (a1c, glucose, fasting) values (20, 800, False);
EXCEPTION WHEN check_violation THEN
    v_msg = 'This A1C is not valid, should be between 0-13';
    INSERT INTO errors (msg) VALUES (v_msg);
END; $$ language 'plpgsql';
```

Exception Example

```
DO $$f
DECLARE
    v_msg TEXT;
BEGIN
     INSERT INTO patients (a1c, glucose, fasting) values (20, 800, False);
EXCEPTION WHEN check_violation THEN
    v_msg = 'This A1C is not valid, should be between 0-13';
    INSERT INTO errors (msg) VALUES (v_msg);
END; $$ language 'plpgsql';
```

Exception Example

```
DO $$f
DECLARE
    v_msg TEXT;
BEGIN
     INSERT INTO patients (a1c, glucose, fasting) values (20, 800, False);
EXCEPTION WHEN check_violation THEN
    v_msg = 'This A1C is not valid, should be between 0-13';
    INSERT INTO errors (msg) VALUES (v_msg);
END; $$ language 'plpgsql';
```

Exception Example

```
DO $$f
DECLARE
    v_msg TEXT;
BEGIN
     INSERT INTO patients (a1c, glucose, fasting) values (20, 800, False);
EXCEPTION WHEN check_violation THEN
    v_msg = 'This A1C is not valid, should be between 0-13';
    INSERT INTO errors (msg) VALUES (v_msg);
END; $$ language 'plpgsql';
```

Graceful Fallback

Graceful Fallback

- This is delightfully evil

Graceful Fallback

- This is delightfully evil
- Think long and hard before you do this...

Graceful Fallback

- This is delightfully evil
- Think long and hard before you do this...
- I'm so not responsible for what you do with this...

Graceful Fallback

- This is delightfully evil
- Think long and hard before you do this...
- I'm so not responsible for what you do with this...
- seriously...

Graceful Fallback

- This is delightfully evil
- Think long and hard before you do this...
- I'm so not responsible for what you do with this...
- seriously...
- You can perform other data actions in the exception handling section!

W... T... F...

Gracefal Fallback Example

```
DO $$
DECLARE
    v_msg TEXT;
BEGIN
     INSERT INTO patients (a1c, glucose, fasting) values (20, 800, False);
EXCEPTION WHEN check_violation THEN
    v_msg = 'This A1C is not valid, should be between 0-13';
    INSERT INTO errors (msg) values (v_msg);
    INSERT INTO patients (a1c, glucose, fasting) values (13, 800, False);
     v_msg = 'Set A1C to the maximum of 13';
    INSERT INTO errors (msg) values (v_msg);
END; $$ language 'plpgsql';
```

error_id	state	msg	detail	context
2	null	This A1C is not valid, should be between 0-13	null	null

Stacked Diagnostics

Name	Description
`RETURNED_SQLSTATE`	the SQLSTATE error code of the exception
`COLUMN_NAME`	the name of the column related to exception
`CONSTRAINT_NAME`	the name of the constraint related to exception
`PG_DATATYPE_NAME`	the name of the data type related to exception
`MESSAGE_TEXT`	the text of the exception's primary message

Name	Description
`TABLE_NAME`	the name of the table related to exception
`SCHEMA_NAME`	the name of the schema related to exception
`PG_EXCEPTION_DETAIL`	the text of the exception's detail message, if any
`PG_EXCEPTION_HINT`	the text of the exception's hint message, if any
`PG_EXCEPTION_CONTEXT`	line(s) of text describing the call stack at the time of the exception

Debug as a function

```
CREATE OR REPLACE FUNCTION debug_statement(
    sql_stmt TEXT
)
RETURNS BOOLEAN AS
$BODY$

    DECLARE
        v_state   TEXT;
        v_msg     TEXT;
        v_detail  TEXT;
        v_context TEXT;
    BEGIN
        BEGIN
            EXECUTE sql_stmt;
        EXCEPTION WHEN others THEN
            GET STACKED DIAGNOSTICS
                v_state   = RETURNED_SQLSTATE,
                v_msg     = MESSAGE_TEXT,
                v_detail  = PG_EXCEPTION_DETAIL,
                v_context = PG_EXCEPTION_CONTEXT;
            INSERT into errors (msg, state, detail, context) values (v_msg, v_state, v_detail, v_context);
            RETURN False;
        END;
        RETURN True;
    END;

$BODY$
LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION debug_statement(
    sql_stmt TEXT
)
RETURNS BOOLEAN AS
$BODY$

    DECLARE
        v_state   TEXT;
        v_msg     TEXT;
        v_detail  TEXT;
        v_context TEXT;
    BEGIN
```

```
        v_context TEXT;
    BEGIN
        BEGIN
            EXECUTE sql_stmt;
        EXCEPTION WHEN others THEN
            GET STACKED DIAGNOSTICS
                v_state   = RETURNED_SQLSTATE,
                v_msg     = MESSAGE_TEXT,
                v_detail  = PG_EXCEPTION_DETAIL,
                v_context = PG_EXCEPTION_CONTEXT;
            INSERT into errors (msg, state, detail, context) values (v_msg, v_state, v_detail, v_context);
            RETURN False;
        END;
        RETURN True;
    END;
$BODY$
```

```
        RETURN True;
    END;
$BODY$
LANGUAGE plpgsql;
```

Debug example

```
DO $$
DECLARE
    stmt VARCHAR(100) := 'INSERT INTO patients (a1c, glucose, fasting)
                          VALUES (20, 800, False)';
BEGIN
     EXECUTE stmt;
EXCEPTION WHEN OTHERS THEN
    PERFORM debug_statement(stmt);
END; $$ language 'plpgsql';
```

error_id	state	msg
2	23514	new row for relation "patients" violates check constraint "patients_a1c_check"

detail

Failing row contains (5, 20, 800, f, 2020-03-06 19:09:36.335213).

context

SQL statement "INSERT INTO patients (a1c, glucose, fasting) VALUES (20, 800, False)"
PL/pgSQL function debug_statement(text) line 10 at EXECUTE
SQL statement "SELECT debug_statement(stmt)"
PL/pgSQL function inline_code_block line 10 at PERFORM

Thank you