Catching exceptions

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Jason Myers
Principal Engineer

Statements that error

```
INSERT INTO sales (name, quantity, cost)
VALUES
  ('chocolate chip', 6, null);
```

```
ERROR:  null value in column "cost" violates not-null constraint
DETAIL:  Failing row contains
(1, "chocolate chip", 6, null, 2020-04-28 19:58:55.715886).
```

Generic exception capture

R

```
tryCatch(
  sqrt("a"),
  error=function(e)
    print("Boom!")
)
```

Python

```
try:
   math.sqrt("a")
except Exception as e:
   print("Boom!")
```

PL/pgSQL

```
BEGIN
SELECT
  SQRT("a");
EXCEPTION WHEN others THEN RAISE INFO 'Boom!';
END;
```

Results

```
     R: Boom!
Python: Boom!
   SQL: INFO: Boom!
```

PL/pgSQL DO commands (anonymous functions)

```
DO $$
DECLARE
   some_variable text;
BEGIN
    SELECT text from a table;
END;
$$ language 'plpgsql';
```

Exception handling function

```
DO $$
BEGIN
    SELECT SQRT("a");
EXCEPTION
    WHEN others THEN
        INSERT INTO errors (msg) VALUES ('Can not take the square root of a string.');
        RAISE INFO 'Can not take the square root of a string.';
END;
$$ language 'plpgsql';
```

Using exception handling wisely

- Using an EXCEPTION clause adds significant overhead
- Python or R exception handling is more efficient
- Don't sacrifice getting the right context to solve the exception
- Don't optimize before you understand your exceptions.

[1] https://www.postgresql.org/docs/12/plpgsql-control-structures.html {{1}}

Changing data sets

patients

column	type
patient_id	integer
a1c	double (float)
glucose	integer
fasting	boolean
created_on	timestamp

errors

column	type
error_id	integer
state	string
msg	string
detail	string
context	string

Let's practice!

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Rollbacks, savepoints, and exceptions

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Jason Myers
Principal Engineer

datacamp

Automatically rolls back

```
DO $$
BEGIN
    UPDATE cookies SET deliciousness = 11 where name = 'Cats Tongue';
    UPDATE cookies SET deliciousness = 12 where name = 'Gingerbread';
EXCEPTION
WHEN others THEN
   INSERT INTO errors (msg) VALUES ('Deliciousness only goes to 11!');
   RAISE INFO 'Deliciousness only goes to 11!';
END;
$$ language 'plpgsql';
```

[1] https://www.postgresql.org/docs/current/plpgsql-transactions.html

```
DO $$
BEGIN
    -- Block 1
    BEGIN
        UPDATE inventory SET cost = 2.33 WHERE name = 'Linga';
        UPDATE inventory SET cost = 2.33 WHERE name = 'Petit-Beurre';
        UPDATE inventory SET cost = 2.33 WHERE name = 'Rosette';
    EXCEPTION
    WHEN others THEN
       INSERT INTO errors (msg) VALUES ('Max cost is 10!');
       RAISE INFO 'Max cost is 10!';
    END;
    -- Block 2
    BEGIN
        UPDATE inventory SET cost = 35.0 WHERE name = 'Macaron';
        UPDATE inventory SET cost = 3.50 WHERE name = 'Panellets';
    EXCEPTION
    WHEN others THEN
       INSERT INTO errors (msg) VALUES ('Max cost is 10!');
       RAISE INFO 'Max cost is 10!';
    END;
END;
$$ language 'plpgsql';
```

Emulating savepoints

```
DO $$
BEGIN
    -- Block 1
    BEGIN
        UPDATE inventory SET cost = 2.33 WHERE name = 'Linga';
        UPDATE inventory SET cost = 2.33 WHERE name = 'Petit-Beurre';
        UPDATE inventory SET cost = 2.33 WHERE name = 'Rosette';
    EXCEPTION
    WHEN others THEN
       INSERT INTO errors (msg) VALUES ('Max cost is 10!');
       RAISE INFO 'Max cost is 10!';
    END;
```

Emulating savepoint continued

```
    -- Block 2
    BEGIN
        UPDATE inventory SET cost = 35.0 WHERE name = 'Macaron';
        UPDATE inventory SET cost = 3.50 WHERE name = 'Panellets';
    EXCEPTION
    WHEN others THEN
       INSERT INTO errors (msg) VALUES ('Max cost is 10!');
       RAISE INFO 'Max cost is 10!';
    END;
END;
$$ language 'plpgsql';
```

A quick aside

- outside datasets
- variables
- incorrect use of field substitution

Let's practice!

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Specific exception handling and messages

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Jason Myers
Principal Engineer

datacamp

Catching a specific type of exception

```
DO $$
BEGIN
    UPDATE inventory SET quantity = quantity - 1 WHERE name in ('flour', 'sugar');
EXCEPTION
    WHEN check_violation THEN
            INSERT INTO errors (msg) VALUES ('Quantity can not be less than 0.');
            RAISE INFO 'Quantity can not be less than 0.';
END;
$$ language 'plpgsql';
```

Output of our exception handler

```
INFO:  Quantity can not be less than 0.
DO
postgres=# select * from errors;
 error_id | state |                msg                | detail | context
^---------+-------+-----------------------------------+--------+---------
        1 |       | Quantity can not be less than 0. |        |
(1 row)
```

Common types of exception conditions

Condition Name	Example
unique_violation	Insert two of the same value in a unique column
not_null_violation	Insert null into a field that doesn't allow nulls
check_violation	Failing a check constraint such as being higher than 11 in deliciousness
division_by_zero	Dividing by 0

So many more at the link in the citation below

[1] https://www.postgresql.org/docs/9.4/errcodes-appendix.html

Catching multiple exceptions

```
DO $$
BEGIN
    UPDATE inventory SET quantity = quantity - 6, cost = null
    WHERE name='oatmeal dark chocolate';
```

Catching multiple exception types individually

```
-- Add check_violation exception
EXCEPTION
  WHEN check_violation THEN
     INSERT INTO errors (msg) VALUES ('Quantity can not be less than 0.');
     RAISE INFO 'Quantity can not be less than 0.';

-- Add non-null exception
  WHEN not_null_violation THEN
     INSERT INTO errors (msg) VALUES ('Cost can not be null.');
     RAISE INFO 'Cost can not be null.';
END; $$ language 'plpgsql';
```

Catching multiple exceptions output

```
INFO:  Cost can not be null.
DO
postgres=# select * from errors;
 error_id | state |                msg                | detail | context
^---------+-------+-----------------------------------+--------+---------
        2 |       | Cost can not be null.             |        |
(1 row)
```

Time to apply it!

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Graceful exception handling

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Jason Myers
Principal Engineer

datacamp

Graceful degradation in action

```
DO $$
BEGIN
    UPDATE cookies SET quantity = quantity-6 WHERE name = 'Linga';
EXCEPTION
    WHEN check_violation THEN
        INSERT INTO errors (msg) values ('Quantity can not be less than 0');
        UPDATE cookies SET quantity = 0 WHERE name = 'Linga';
        INSERT INTO errors (msg) values ('Set quantity to the 0 for Linga.');
END$$;
```

When to use graceful degradation

- Loading data from an external system where you want to replace nulls with 0s
- Getting readings from an instrument that is only accurate up to a certain threshold
- Receiving dates that are out of bounds that you want to set to some sentinel value
- Writing all records that cause exceptions to another table for further processing

When to consider using graceful exception handling

- When the new value would be hidden behind a math operation such as a sum, avg, or other aggregate.
- When the new value affects data in a time series.

Let's practice!

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL