Enhancing exception handling with stacked diagnostics

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Jason Myers
Principal Engineer

datacamp

Capturing more error information

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

Using stacked diagnostics

```
DO $$
DECLARE
   exc_message text;
   exc_detail text;
BEGIN
   UPDATE inventory SET cost = 35.0 WHERE name = 'Macaron';
   UPDATE inventory SET cost = 3.50 WHERE name = 'Panellets';
EXCEPTION
   WHEN others THEN
          GET STACKED DIAGNOSTICS exc_message = MESSAGE_TEXT,
                                  exc_detail = PG_EXCEPTION_DETAIL;
       INSERT INTO errors (msg, detail) VALUES (exc_message, exc_detail);
       RAISE INFO 'Exception Messaage: % | Exception Details: %', exc_message, exc_detail;
END$$;
```

Example diagnostic output

```
INFO:  Exception Messaage: new row for relation "inventory" violates check constraint
"cost_check" | Exception Details: Failing row contains (7, 35, Macaron).
DO

postgres=# \x on
Expanded display is on.
postgres=# select msg, detail from errors;
-[ RECORD 1 ]----------------------------------------------------------
msg    | new row for relation "inventory" violates check constraint "cost_check"
detail | Failing row contains (7, 35, Macaron).
```

So what all can you get?

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
MESSAGE_TEXT	the text of the exception's primary message
PG_EXCEPTION_DETAIL	the text of the exception's detail message, if any

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

More diagnostic datapoints

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

Let's practice!

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Hints to help handle nested exceptions

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Jason Myers
Principal Engineer

datacamp

Emulating savepoints with nested blocks

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

Emulating savepoint Block 2

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

Nested blocks with stacked diagnostics

```
DO $$
DECLARE
   exc_message text;
   exc_detail text;
   exc_context text;
BEGIN
    -- Block 1
    BEGIN
        UPDATE inventory SET cost = 2.33 WHERE name = 'Linga';
        UPDATE inventory SET cost = 2.33 WHERE name = 'Petit-Beurre';
    EXCEPTION
    WHEN others THEN
        GET STACKED DIAGNOSTICS exc_message = MESSAGE_TEXT,
                                exc_detail = PG_EXCEPTION_DETAIL,
                                exc_context = PG_EXCEPTION_CONTEXT;
       INSERT INTO errors (msg,detail, context)
           VALUES (exc_message, exc_detail, exc_context);
    END;
    -- Block 2
    BEGIN
        UPDATE inventory SET cost = 35.0 WHERE name = 'Macaron';
    EXCEPTION
    WHEN others THEN
       GET STACKED DIAGNOSTICS exc_message = MESSAGE_TEXT,
                               exc_detail = PG_EXCEPTION_DETAIL,
                               exc_context = PG_EXCEPTION_CONTEXT;
       INSERT INTO errors (msg,detail, context)
           VALUES (exc_message, exc_detail, exc_context);
    END;
END$$;
```

Nested blocks with stacked diagnostics

```
DO $$
DECLARE
    exc_message text;
    exc_detail text;
    exc_context text;
BEGIN
```

Nested blocks with stacked diagnostics: block 2

```
-- Block 1
BEGIN
    UPDATE inventory SET cost = 2.33 WHERE name = 'Linga';
    UPDATE inventory SET cost = 2.33 WHERE name = 'Petit-Beurre';
EXCEPTION
WHEN others THEN
    GET STACKED DIAGNOSTICS exc_message = MESSAGE_TEXT,
                            exc_detail = PG_EXCEPTION_DETAIL,
                            exc_context = PG_EXCEPTION_CONTEXT;
   INSERT INTO errors (msg,detail, context)
       VALUES (exc_message, exc_detail, exc_context);
END;
```

Nested blocks with stacked diagnostics: block 2

```
-- Block 2
    BEGIN
        UPDATE inventory SET cost = 35.0 WHERE name = 'Macaron';
    EXCEPTION
    WHEN others THEN
       GET STACKED DIAGNOSTICS exc_message = MESSAGE_TEXT,
                               exc_detail = PG_EXCEPTION_DETAIL,
                               exc_context = PG_EXCEPTION_CONTEXT;
       INSERT INTO errors (msg,detail, context)
           VALUES (exc_message, exc_detail, exc_context);
    END;
END$$;
```

Results

```
INFO:  Message: new row for relation "inventory" violates check constraint
"cost_check" | Details: Failing row contains (7, 35, Macaron). | Context SQL
statement "UPDATE inventory SET cost = 35.0 WHERE name = 'Macaron'"
PL/pgSQL function inline_code_block line 23 at SQL statement
DO
postgres=# \x on
Expanded display is on.
postgres=# select * from errors;
-[ RECORD 1 ]-----------------------------------------------------------------
error_id | 15
state    |
msg      | new row for relation "inventory" violates check constraint "cost_check"
detail   | Failing row contains (7, 35, Macaron).
context  | SQL statement "UPDATE inventory SET cost = 35.0 WHERE name = 'Macaron'"+
         | PL/pgSQL function inline_code_block line 23 at SQL statement
```

Custom exception handling vs stacked diagnostics

Custom

- Clear error context
- Expected error condition
- Standard error message too generic

Stacked Diagnostics

- Need to be able to get more context for the error
- Many possible error conditions
- Debugging
- Generalizing exception handling

Let's practice!

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Mixing it all together with debugging functions

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Jason Myers
Principal Engineer

datacamp

Named function overview

```
CREATE OR REPLACE FUNCTION function_name(
    parameter1 TEXT,
    parameter2 INTEGER
)
RETURNS BOOLEAN AS $$
    DECLARE
    BEGIN
        STATEMENTS
    END;
$$ LANGUAGE plpgsql;
```

A function for debugging

```
CREATE OR REPLACE FUNCTION debug_statement(
    sql_stmt TEXT
)
RETURNS BOOLEAN AS $$
    DECLARE
        v_state   TEXT;
        v_msg     TEXT;
        v_detail  TEXT;
        v_context TEXT;
    BEGIN
        BEGIN
            EXECUTE sql_stmt;
```

The rest of a function for debugging

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

Using the function as a statement

```
SELECT debug_statement(
  'UPDATE inventory SET cost = 35.0 WHERE name = ''Macaron'' '
);
-[ RECORD 1 ]---+--
debug_statement | t
```

Reviewing the functions recording of the exception

```
SELECT * FROM errors;
-[ RECORD 1 ]----------------------------------------------------------------
error_id | 20
state    | 23514
msg      | new row for relation "inventory" violates check constraint "cost_check"
detail   | Failing row contains (7, 35, Macaron).
context  | SQL statement "UPDATE inventory SET cost = 35.0 WHERE name = 'Macaron' "
         | PL/pgSQL function debug_statement(text) line 9 at EXECUTE
```

Using the function with in a function

```
DO $$
DECLARE
    stmt VARCHAR(100) := 'UPDATE inventory SET cost = 35.0 WHERE name = ''Macaron'' ';
BEGIN
     EXECUTE stmt;
EXCEPTION WHEN OTHERS THEN
    PERFORM debug_statement(stmt);
END; $$ language 'plpgsql';
```

Error recording from the DO function

```
SELECT * FROM errors;
-[ RECORD 1 ]-----------------------------------------------------------
error_id | 21
state    | 23514
msg      | new row for relation "inventory" violates check constraint "cost_check"
detail   | Failing row contains (7, 35, Macaron).
context  | SQL statement "UPDATE inventory SET cost = 35.0 WHERE name = 'Macaron' "+
         | PL/pgSQL function debug_statement(text) line 9 at EXECUTE                +
         | SQL statement "SELECT debug_statement(stmt)"                             +
         | PL/pgSQL function inline_code_block line 7 at PERFORM
```

Let's practice!

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Wrapping it up

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL

Jason Myers
Principal Engineer

Look at all your learned

Transactions

- BEGIN
- COMMIT
- ROLLBACK / SAVEPOINT

Isolation and Concurrency

- READ [UN]COMMITTED
- REPEATABLE READY
- START TRANSACTION

Exception Handling

- EXCEPTION WHEN
- Nested Blocks
- Stacked Diagnostics

Functions

- DO / Unnamed
- CREATE OR REPLACE ... RETURNS
- PERFORM / EXECUTE

Other courses I suggest to explore further

- Creating PostgreSQL Databases
- Functions for Manipulating Data in PostgreSQL
- PostgreSQL Summary Stats and Window Functions

Excellent Work!

TRANSACTIONS AND ERROR HANDLING IN POSTGRESQL