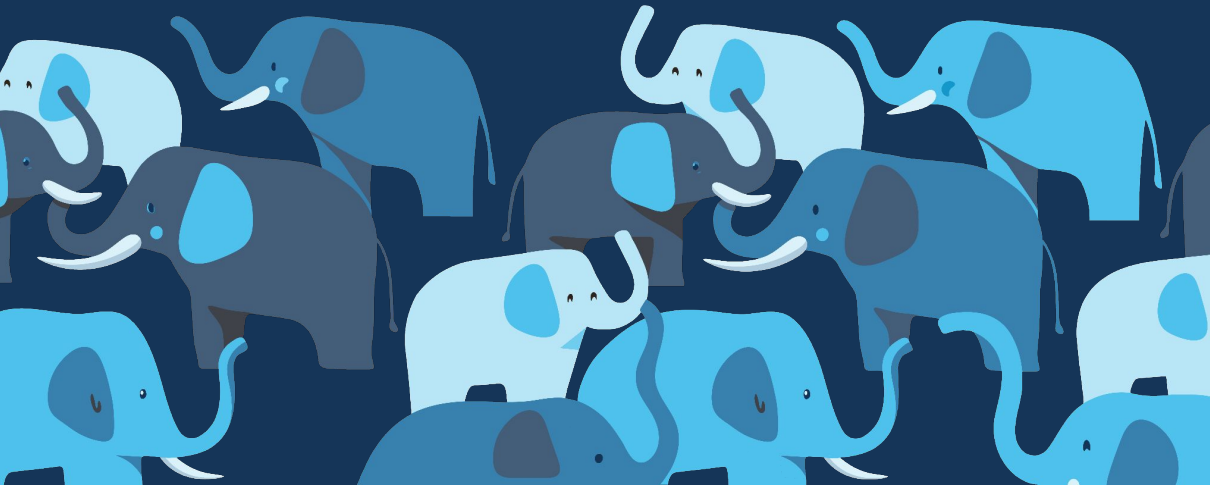
pgcov: Real Test Coverage for PostgreSQL

Pavlo Golub

PgConf.DE 2026

CYBERTEC

PostgreSQL is at the heart of everything we do, and we are proud to actively contribute to its community and project.



The Problem

The Blind Spot

We measure coverage everywhere:

- Go: `go test -cover`
- Rust: `cargo tarpaulin`
- Python: `pytest --cov`

But for SQL and PL/pgSQL?
No visibility at all.

SQL Is Real Code

PostgreSQL functions contain real branching logic:

```
IF v_stock = 0 THEN
    RETURN 'out_of_stock';
ELSIF v_stock <= 10 THEN
    RETURN 'low_stock';
ELSE
    RETURN 'in_stock';
END IF;
```

Tests pass. CI is green.

Which branches actually ran?

pgcov

How pgcov Works

Instrument → Run → Report

1. Rewrites SQL and PL/pgSQL bodies, injects `pg_notify` signals
2. Creates an isolated DB per test file
3. Collects fired signals → maps to source positions
4. Emits JSON / LCOV / **HTML** report

No extensions. No engine patches. Plain SQL.

Run pgcov — 80% Coverage

```
$ pgcov run -c "postgresql://..." ./examples/demo/

Tests:    1 passed, 0 failed, 1 total
Coverage: 80.00%
Time:     592ms
```


examples\demo\02_functions.sql (77.8%%) not tracked not covered covered

```
--
--   Branch signals:
--     'out_of_stock' signal  → only fires when stock = 0
--     'low_stock'    signal  → only fires when 0 < stock ≤ 10
--   (both are missed by the basic test that uses stock = 50)
-- ---------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION get_stock_status(
    p_product_id INT
) RETURNS TEXT AS $$
DECLARE
    v_stock  INT;
    v_result TEXT;
BEGIN
    SELECT stock_qty INTO v_stock
    FROM   products
    WHERE  id = p_product_id;

    IF NOT FOUND THEN
        RETURN 'unknown';
    END IF;

    IF v_stock = 0 THEN
        v_result := 'out_of_stock';
    ELSIF v_stock <= 10 THEN
        v_result := 'low_stock';
    ELSE
        v_result := 'in_stock';
    END IF;

    RETURN v_result;
END;
$$ LANGUAGE plpgsql;
```

What pgcov Found

Four branches not exercised:

- `in_stock`, `low_stock` — stock edge cases never called
- `premium` and `standard` pricing tiers — never tested

Add the Missing Tests

```
v_id := add_product('In Stock', 1.00, 1);
ASSERT get_stock_status(v_id) = 'in_stock';

v_id := add_product('Rare Find', 1.00, 5);
ASSERT get_stock_status(v_id) = 'low_stock';

v_price := apply_pricing_policy(v_id, 'premium');
ASSERT v_price = 85.00;

v_price := apply_pricing_policy(v_id, 'standard');
ASSERT v_price = 95.00;
```

One targeted assertion per red branch.

Run pgcov Again — 100% Coverage

```
$ pgcov run -c "postgresql://..." ./examples/demo/

Tests:    1 passed, 0 failed, 1 total
Coverage: 100.00%
Time:     594ms
```


examples\demo\02_functions.sql (100.0%%) not tracked not covered covered

```
CREATE OR REPLACE FUNCTION get_stock_status(
    p_product_id INT
) RETURNS TEXT AS $$
DECLARE
    v_stock  INT;
    v_result TEXT;
BEGIN
    SELECT stock_qty INTO v_stock
    FROM   products
    WHERE  id = p_product_id;

    IF NOT FOUND THEN
        RETURN 'unknown';
    END IF;

    IF v_stock = 0 THEN
        v_result := 'out_of_stock';
    ELSIF v_stock <= 10 THEN
        v_result := 'low_stock';
    ELSE
        v_result := 'in_stock';
    END IF;

    RETURN v_result;
END;
$$ LANGUAGE plpgsql;
```

Takeaway

Thank You

Think `go test -cover`*, but for SQL*

- Standalone CLI, no extensions required
- JSON / LCOV / HTML reports
- CI/CD friendly
- Not a replacement for pgTAP or unit tests

github.com/cybertec-postgresql/pgcov