PHD Level SQL For Terrible Applications

TTP Conference 2022

Tyler Webb

Escambia County School District

2022-07-15

Who Am I?

- Escambia County School District
- Daily PostgreSQL development 5 years
- Also a “little bit” of PHP, JS, Python, C#

“Terrible” Application

- DB as an afterthought
- Every feature gets a table
- Inconsistency
  - Table Names
  - Column Names
    - ★ No foreign keys
  - Data Types
- No documentation

Workarounds

- Views
- CTEs
- Materialized Views
- Stored Procedures / Functions
- Replication

“Exotic” JOINs

- Everything is a table
- LATERAL JOINs
  - Join a record on a part of itself
- RIGHT JOINs are bad

Treating SQL like C

- PL/pgSQL (SQL Procedural Language)

Example DB

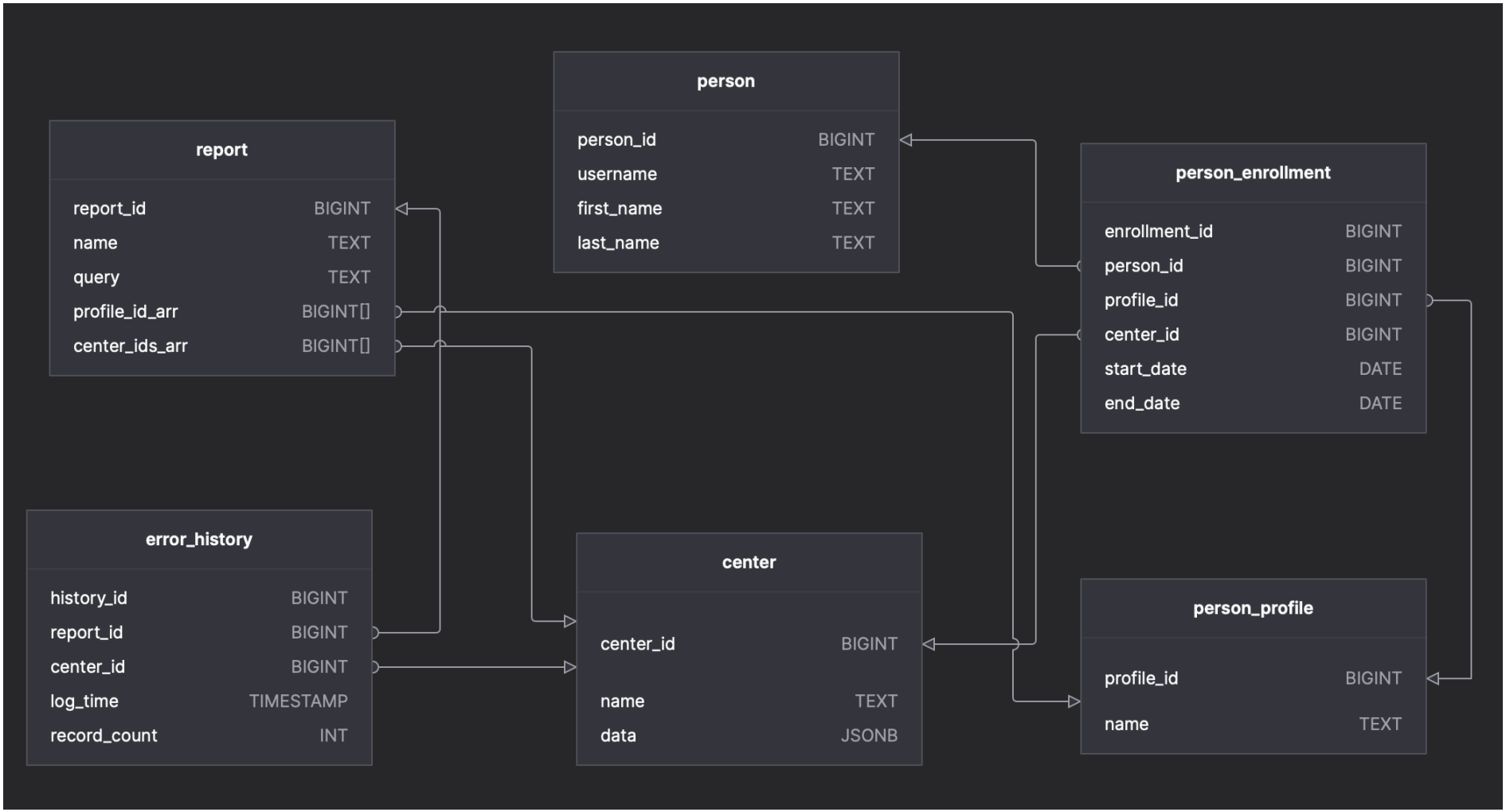

Figure 1: Example DB Diagram

Show all errors for a specific user

```
WITH cte_reports AS (
    SELECT r.report_id, pe.center_id
    FROM
        report r
            JOIN person_enrollment pe ON pe.person_id = {PERSON_ID}
                AND (pe.profile_id = ANY(r.profile_ids) OR pe.profile_id IN (1, 2))
                AND (pe.center_id = ANY(r.center_ids) OR pe.center_id IS NULL)
                AND CURRENT_DATE < COALESCE(pe.end_date, '9999-12-31')
    WHERE r.is_error_report
)
,   cte_history AS (
    SELECT eh.report_id, eh.center_id, eh.record_count
        ,   ROW_NUMBER() OVER (PARTITION BY eh.report_id, eh.center_id ORDER BY eh.log_time DESC) AS _rn
    FROM
        cte_reports cr
            JOIN error_history eh ON cr.report_id = eh.report_id
                AND COALESCE(cr.center_id, eh.center_id) = eh.center_id
)
SELECT
        c.name AS "Center"
    ,   r.name AS "Report"
    ,   ch.record_count AS "Number of Errors"
FROM
    cte_history ch
        JOIN center c ON ch.center_id = c.center_id
        JOIN report r ON ch.report_id = r.report_id
WHERE ch._rn = 1 AND ch.record_count > 0;
```

Run Errors

```
DO $run_errors$
DECLARE
    _REPORT_ID BIGINT;
    _CENTER_IDS BIGINT[];
    ...  -- Variables for exception catching
BEGIN
    FOR _REPORT_ID, _CENTER_IDS
    IN
        SELECT r.report_id, r.center_ids
        FROM report r
        WHERE r.is_error_report
    LOOP
        BEGIN
            EXECUTE "f:report:run_and_log"(_REPORT_ID, _CENTER_IDS);
        EXCEPTION
            WHEN OTHERS THEN
                -- Put all error information in variables
                GET STACKED DIAGNOSTICS ...;
                -- Insert error variables into a local logging table
                INSERT INTO ecsd_report_run_errors (...) VALUES (...);
        END;
    END LOOP;
END
$run_errors$
LANGUAGE plpgsql;
```

“f:report:run_and_log”()

```
CREATE OR REPLACE FUNCTION "f:report:run_and_log" (
            _REPORT_ID BIGINT
        ,   _CENTER_IDS BIGINT[]
    )
RETURNS VOID AS $report_run_and_log$
DECLARE
    _QUERY_STRING TEXT := (SELECT query FROM report WHERE report_id = _REPORT_ID LIMIT 1);
    _CENTER_ID BIGINT;
    _KEY TEXT;
    _RECORD_COUNT BIGINT;
    _LOG_TIME TIMESTAMP;
BEGIN

    FOREACH _CENTER_ID IN ARRAY _CENTER_IDS
    LOOP

        _LOG_TIME := clock_timestamp();
        -- Replace every instance of {CENTER_ID} in query with _CENTER_ID
        EXECUTE REGEXP_REPLACE(_QUERY_STRING, '\{CENTER_ID\}', _CENTER_ID::TEXT, 'gi');
        GET DIAGNOSTICS _RECORD_COUNT := row_count;

        INSERT INTO error_history(report_id, center_id, log_time, record_count)
        VALUES (_REPORT_ID, _CENTER_ID, _LOG_TIME, _RECORD_COUNT);

    END LOOP;

END
$report_run_and_log$
LANGUAGE plpgsql;
```

Conclusion

- Contact me
  - twebb2@ecsdfl.us

```
DO $drop_tables$
BEGIN
    RAISE NOTICE '[%] Dropping tables...', clock_timestamp();
    DROP TABLE IF EXISTS center;
    DROP TABLE IF EXISTS person_profile;
    DROP TABLE IF EXISTS person;
    DROP TABLE IF EXISTS person_enrollment;
    DROP TABLE IF EXISTS report;
    DROP TABLE IF EXISTS error_history;
END
$drop_tables$
LANGUAGE plpgsql;

DO $center_create_and_populate$
DECLARE

    _NUM_BUILDINGS INT := 80;

BEGIN

    RAISE NOTICE '[%] Creating center...', clock_timestamp();

    CREATE TABLE
        center (
                center_id BIGINT GENERATED ALWAYS AS IDENTITY
            ,   name TEXT
            ,   data JSONB
        )
    ;

    CREATE INDEX "i:center:name" ON center (name);

    INSERT INTO center (name, data)
    SELECT
            'Center' || LPAD(building_num::TEXT, 3, '0')
        ,   jsonb_build_object(
                'area', (ARRAY['NORTH', 'EAST', 'SOUTH',
'WEST'])[FLOOR(RANDOM() * 4 + 1)]
            )
    FROM GENERATE_SERIES(1, _NUM_BUILDINGS) building_num
    ;

END
$center_create_and_populate$
LANGUAGE plpgsql;

DO $person_profile_create_and_populate$
DECLARE
BEGIN

    RAISE NOTICE '[%] Creating person_profile...',
clock_timestamp();

    CREATE TABLE
        person_profile (
                profile_id BIGINT GENERATED ALWAYS AS IDENTITY
            ,   name TEXT
```

```
        )
    ;

    CREATE INDEX "i:person_profile:name" ON person_profile
(name);

    INSERT INTO person_profile (name)
    VALUES
            ('System Administrator')
        ,   ('Programmer')
        ,   ('Administrator')
        ,   ('Person')
    ;

END
$person_profile_create_and_populate$
LANGUAGE plpgsql;

DO $person_create_and_populate$
DECLARE
    _M1 TEXT[] :=
'{"Michael","Christopher","James","Robert","John","David","Wil
liam","Joshua","Joseph","Matthew"}'::TEXT[]; -- Male first
names
    _M2 TEXT[] :=
'{"Smith","Johnson","Williams","Brown","Jones","Davis","Jackso
n","Thomas","Wilson","Lewis"}'::TEXT[]; -- Male last names

    _F1 TEXT[] :=
'{"Jennifer","Jessica","Ashley","Amanda","Sarah","Mary","Eliza
beth","Brittany","Heather","Stephanie"}'::TEXT[]; -- Female
first names
    _F2 TEXT[] :=
'{"Williams","Johnson","Smith","Jones","Brown","Davis","Jackso
n","White","Wilson","Thomas"}'::TEXT[]; -- Female last names

BEGIN

    RAISE NOTICE '[%] Creating person...', clock_timestamp();

-- ------------------------------------------------------------
------------------
-- Create person table.
-- ------------------------------------------------------------
------------------
    CREATE TABLE
        person (
                person_id BIGINT GENERATED ALWAYS AS IDENTITY
            ,   username TEXT
            ,   first_name TEXT
            ,   last_name TEXT
        )
    ;

-- ------------------------------------------------------------
------------------
-- Create indexes.
```

```
-- ------------------------------------------------------------
-----------------
    CREATE INDEX "i:person:username" ON person (username);

-- ------------------------------------------------------------
-----------------
-- Insert into the person table.
-- ------------------------------------------------------------
-----------------
    INSERT INTO
        person (
                username
            ,   first_name
            ,   last_name
        )
    SELECT *
    FROM
        (
            SELECT
                    LOWER(LEFT(fname, 1) || lname) AS username
                ,   fname                          AS
first_name
                ,   lname                          AS
last_name
            FROM
                UNNEST(_M1) fname
                    CROSS JOIN UNNEST(_M2) lname
            UNION
            SELECT
                    LOWER(LEFT(fname, 1) || lname) AS username
                ,   fname                          AS
first_name
                ,   lname                          AS
last_name
            FROM
                UNNEST(_F1) fname
                    CROSS JOIN UNNEST(_F2) lname
        ) insert_values
    ORDER BY RANDOM()
    ;

-- ------------------------------------------------------------
-----------------
-- Make usernames unique.
-- ------------------------------------------------------------
-----------------
    UPDATE person p
    SET username = p.username || x.rn
    FROM
        (
            SELECT person_id, ROW_NUMBER() OVER (PARTITION BY
username ORDER BY person_id) AS rn
            FROM person
        ) x
    WHERE p.person_id = x.person_id
    ;
```

```
END
$person_create_and_populate$
LANGUAGE plpgsql;

DO $person_enrollment_create_and_populate$
DECLARE
BEGIN

    RAISE NOTICE '[%] Creating person_enrollment...',
clock_timestamp();

    CREATE TABLE
        person_enrollment (
                enrollment_id BIGINT GENERATED ALWAYS AS
IDENTITY
            ,   person_id BIGINT
            ,   profile_id BIGINT
            ,   center_id BIGINT
            ,   start_date DATE
            ,   end_date DATE
        )
    ;

    CREATE INDEX "i:person_enrollment:profile_id,center_id" ON
person_enrollment (profile_id, center_id);
    CREATE INDEX "i:person_enrollment:center_id" ON
person_enrollment (center_id);
    CREATE INDEX "i:person_enrollment:start_date" ON
person_enrollment (start_date);
    CREATE INDEX "i:person_enrollment:end_date" ON
person_enrollment (end_date);

-- ------------------------------------------------------------
------------------
-- Insert values with some assumptions.
-- A "person_enrollment" record with no center_id has that
"profile_id" for ALL centers.
-- There's only one System Administrator. A few Programmers.
Each center gets one Administrator. The rest are persons.
-- Some persons can go to multiple centers.
-- persons are active while "end_date IS NULL"
-- ------------------------------------------------------------
------------------
    -- ------------------------------------------------------
------------------
    -- System Administrator
    -- ------------------------------------------------------
------------------
    INSERT INTO
        person_enrollment (
                person_id
            ,   profile_id
            ,   center_id
            ,   start_date
        )
    SELECT
            person_id
```

```
        ,   sys_admin_profile.profile_id
        ,   NULL
        ,   CURRENT_DATE
    FROM
        person
            CROSS JOIN (SELECT profile_id FROM person_profile
WHERE name = 'System Administrator') sys_admin_profile
    ORDER BY RANDOM()
    LIMIT 1
    ;

    -- ------------------------------------------------------
------------------
    -- Center Administrators
    -- ------------------------------------------------------
------------------
    INSERT INTO
        person_enrollment (
                person_id
            ,   profile_id
            ,   center_id
            ,   start_date
        )
    SELECT
            u_sorted.person_id
        ,   admin_profile.profile_id
        ,   c.center_id
        ,   CURRENT_DATE
    FROM
        (
            SELECT person_id, ROW_NUMBER() OVER (ORDER BY
RANDOM()) rn
            FROM person p
            WHERE NOT EXISTS (SELECT FROM person_enrollment pe
WHERE p.person_id = pe.person_id)
        ) u_sorted
            JOIN center c ON u_sorted.rn = c.center_id
            CROSS JOIN (SELECT profile_id FROM person_profile
WHERE name = 'Administrator') admin_profile
    ORDER BY RANDOM()
    ;

    -- ------------------------------------------------------
------------------
    -- Programmers
    -- ------------------------------------------------------
------------------
    INSERT INTO
        person_enrollment (
                person_id
            ,   profile_id
            ,   center_id
            ,   start_date
        )
    SELECT
            u_sorted.person_id
        ,   programmer_profile.profile_id
```

```
        ,   NULL
        ,   CURRENT_DATE
    FROM
        (
            SELECT person_id, ROW_NUMBER() OVER (ORDER BY
RANDOM()) rn
            FROM person p
            WHERE NOT EXISTS (SELECT FROM person_enrollment pe
WHERE p.person_id = pe.person_id)
        ) u_sorted
            CROSS JOIN (SELECT profile_id FROM person_profile
WHERE name = 'Programmer') programmer_profile
    WHERE rn <= 10
    ;

    -- ------------------------------------------------------
------------------
    -- Persons
    -- ------------------------------------------------------
------------------
    INSERT INTO
        person_enrollment (
                person_id
            ,   profile_id
            ,   center_id
            ,   start_date
        )
    SELECT
            u_sorted.person_id
        ,   person_profile.profile_id
        ,   u_sorted.center_id
        ,   CURRENT_DATE
    FROM
        (
            SELECT person_id, FLOOR(RANDOM() * (SELECT
COUNT(*) FROM center) + 1) AS center_id
            FROM person p
            WHERE NOT EXISTS (SELECT FROM person_enrollment pe
WHERE p.person_id = pe.person_id)
        ) u_sorted
            CROSS JOIN (SELECT profile_id FROM person_profile
WHERE name = 'person') person_profile
    ;

    -- ------------------------------------------------------
------------------
    -- Multiple center persons
    -- ------------------------------------------------------
------------------
    INSERT INTO
        person_enrollment (
                person_id
            ,   profile_id
            ,   center_id
            ,   start_date
        )
    SELECT
```

```
            u_sorted.person_id
        ,   person_profile.profile_id
        ,   u_sorted.center_id
        ,   CURRENT_DATE
    FROM
        (
            SELECT person_id, FLOOR(RANDOM() * (SELECT
COUNT(*) FROM center) + 1) AS center_id
            FROM person p
            WHERE NOT EXISTS (SELECT FROM person_enrollment pe
WHERE p.person_id = pe.person_id AND pe.center_id IS NULL)
        ) u_sorted
            CROSS JOIN (SELECT profile_id FROM person_profile
WHERE name = 'person') person_profile
    WHERE 1=1
        AND NOT EXISTS (SELECT FROM person_enrollment pe WHERE
u_sorted.person_id = pe.person_id AND u_sorted.center_id =
pe.center_id) -- Make sure they don't already have an active
enrollment at the center
        AND RANDOM() < 0.30 -- 30% or less persons go to
multiple centers
    ;

END
$person_enrollment_create_and_populate$
LANGUAGE plpgsql;

DO $report_create_and_populate$
DECLARE

    _NUM_REPORTS INT := 2000;

BEGIN

    RAISE NOTICE '[%] Creating report...', clock_timestamp();

-- ------------------------------------------------------------
------------------
-- Create report table.
-- ------------------------------------------------------------
------------------
    CREATE TABLE
        report (
                report_id BIGINT GENERATED ALWAYS AS IDENTITY
            ,   name TEXT
            ,   query TEXT
            ,   profile_ids BIGINT[]
            ,   center_ids BIGINT[]
            ,   is_error_report BOOLEAN
        )
    ;

-- ------------------------------------------------------------
------------------
-- Create indexes.
-- These were added to optimize queries
-- No need to get real fancy, just whatever the DB thinks is
```

```
best will work fine
-- ------------------------------------------------------------
-----------------
    CREATE INDEX "i:report:profile_ids" ON report
(profile_ids);
    CREATE INDEX "i:report:center_ids" ON report (center_ids);

    -- ------------------------------------------------------------
-----------------
    -- Create a temporary function that will create us random
arrays.
    -- ------------------------------------------------------------
-----------------
    CREATE OR REPLACE FUNCTION
        pg_temp.random_int_array(
                _low BIGINT
            ,   _high BIGINT
            ,   _length INT
        )
    RETURNS BIGINT[] AS
    $func$SELECT ARRAY(SELECT DISTINCT FLOOR(RANDOM() *
(_high - _low + 1) + _low)::BIGINT FROM generate_series(1,
_length) ORDER BY 1)$func$
    LANGUAGE sql;

    INSERT INTO
        report (
                name
            ,   query
            ,   profile_ids
            ,   center_ids
            ,   is_error_report
        )
    SELECT
            'Report' || LPAD(report_num::TEXT, 4, '0') AS name
        ,   $sample_query$SELECT num FROM GENERATE_SERIES(1,
FLOOR(RANDOM() * 100 + 1)::INT) num$sample_query$ AS query --
Example queries just return a random number of records back
        ,   pg_temp.random_int_array(3, 4, FLOOR(RANDOM() * 2
+ 1)::INT) AS profile_ids -- We know Admin and Person profiles
are id 3 and 4
        ,   pg_temp.random_int_array(1, 80, FLOOR(RANDOM() *
80 + 60)::INT) AS center_ids -- Each report can be attached up
to 80 centers
        ,   RANDOM() < 0.40 -- 40% of the reports are error
reports
    FROM
        GENERATE_SERIES(1, _NUM_REPORTS) report_num
    ;

END
$report_create_and_populate$
LANGUAGE plpgsql;

DO $error_history_create_and_populate$
DECLARE
-- ------------------------------------------------------------
```

```
------------------
-- My naming convention for local variables is start_with an
underscore and all caps*
-- ------------------------------------------------------------
------------------
    _DATE DATE := CURRENT_DATE - 5;

BEGIN

    RAISE NOTICE '[%] Creating error_history...',
clock_timestamp();

    CREATE TABLE
        error_history (
                history_id BIGINT GENERATED ALWAYS AS IDENTITY
            ,   report_id BIGINT
            ,   center_id BIGINT
            ,   log_time TIMESTAMP
            ,   record_count INT
        )
    ;

-- ------------------------------------------------------------
------------------
-- Here we create the indexes.
-- No need to create one for history_id since there already is
one made since we marked the column with "AS IDENTITY"
-- ------------------------------------------------------------
------------------

    -- ------------------------------------------------------
------------------
    -- Simple index with the following naming convention:
    --      i:table_name:column_name(s)
    -- Where "i" means "index"
    -- These are what exist by default from the vendor
    -- ------------------------------------------------------
------------------
    CREATE INDEX "i:error_history:log_time" ON error_history
(log_time);
    CREATE INDEX "i:error_history:report_id" ON error_history
(report_id);
    CREATE INDEX "i:error_history:center_id" ON error_history
(center_id);

-- ------------------------------------------------------------
------------------
-- Populate the test table with test data.
-- Note: Need to make sure the `person`, `report`, and
`center` tables are created and populated already.
-- ------------------------------------------------------------
------------------

    WHILE _DATE < CURRENT_DATE
    LOOP

        INSERT INTO
```

```
            error_history (
                    report_id
                ,   center_id
                ,   log_time
                ,   record_count
            )
        SELECT *
        FROM
            (
                SELECT
                        data.report_id
AS report_id    -- What report was ran
                    ,   data.center_id
AS center_id    -- What center was this run from
                    ,   _DATE + ((RANDOM() * 86399 + 1) || '
seconds')::INTERVAL                 AS log_time     -- Random
time throughout the day
                    ,   CASE WHEN RANDOM() < 0.8 THEN 0 ELSE
FLOOR(RANDOM() * 100 + 1)::INT END AS record_count -- The
amount of errors found in report (Most report return 0)
                FROM
                    ( -- For each center_id they run a
report_id "random_num" times in a day
                    SELECT
                            r.report_id
                        ,   c.center_id
                        ,   (RANDOM() * 40 + 5)::INT AS
random_num -- For the amount of times a center ran a report
                    FROM
                        report r
                            JOIN center c ON r.center_ids
@> ARRAY[c.center_id]
                    ) data
                    CROSS JOIN LATERAL GENERATE_SERIES(1,
data.random_num) run -- Makes a new record for the amount of
"random_num"
            ) insert_data
        ORDER BY insert_data.log_time
        ;

        _DATE := _DATE + 1;
    END LOOP;

END
$error_history_create_and_populate$
LANGUAGE plpgsql;
DO $ecsd_report_run_errors_create_and_populate$
DECLARE
BEGIN

    CREATE TABLE
        ecsd_report_run_errors (
                run_id BIGINT GENERATED ALWAYS AS IDENTITY
            ,   report_id BIGINT
            ,   log_time TIMESTAMP
            ,   returned_sqlstate TEXT
            ,   column_name TEXT
```

```
            ,   constraint_name TEXT
            ,   pg_datatype_name TEXT
            ,   message_text TEXT
            ,   table_name TEXT
            ,   schema_name TEXT
            ,   pg_exception_detail TEXT
            ,   pg_exception_hint TEXT
            ,   pg_exception_context TEXT
        )
    ;

    CREATE INDE "i:ecsd_report_run_errors:report_id" ON
ecsd_report_run_errors (report_id);
    CREATE INDE "i:ecsd_report_run_errors:log_time" ON
ecsd_report_run_errors (log_time);

    -- No need to populate this table since we won't be
querying it in our example(s)

END
$ecsd_report_run_errors_create_and_populate$
LANGUAGE plpgsql;

VACUUM FULL ANALYSE center;
VACUUM FULL ANALYSE person_profile;
VACUUM FULL ANALYSE person;
VACUUM FULL ANALYSE person_enrollment;
VACUUM FULL ANALYSE report;
VACUUM FULL ANALYSE error_history;
VACUUM FULL ANALYSE ecsd_report_run_errors;
```