Advantage PostgreSQL

Let it help you!

PGDay Nordic 2024

Chris Ellis - @intrbiz@bergamot.social

Hello!

- I’m Chris
  - IT jack of all trades
- Using PostgreSQL ~18 years, across a range of projects:
  - A website search engine
  - UK postal address search, mapping
  - Service Directory
  - Monitoring
  - Smart Energy Analytics and IOT
  - TV, VoD catalogues
  - Booking / subscriptions

<3 PostgreSQL

SELECT * FROM audience WHERE ...

Right Tool For The Job?

Architecture

VS

Queues

Queues - A Simple Queue

```
CREATE TABLE queue.event (
  hook_id     UUID          NOT NULL,
  created     TIMESTAMP     NOT NULL,
  updated     TIMESTAMP             ,
  status      INTEGER       NOT NULL,
  payload     TEXT
);
```

Queues - Fetch A Batch

```
SELECT ctid, * FROM queue.event
WHERE status < 5 AND (status = 0 OR
 updated < (now() - '1 hour'::INTERVAL))
ORDER BY created DESC
LIMIT 10
FOR UPDATE SKIP LOCKED;
```

Queues - Index Time

```sql
CREATE INDEX queue_event_idx
ON queue.event (created)
WHERE status < 5;
```

Queues - Fetch A Batch

```
 Limit
 (cost=0.29..0.86 rows=10 width=54)
 (actual time=0.060..0.114 rows=10 loops=1)
  ->  LockRows
      (cost=0.29..4920.33 rows=86401 width=54)
      (actual time=0.057..0.109 rows=10 loops=1)
        ->  Index Scan Backward using queue_event_idx on event
            (cost=0.29..4056.32 rows=86401 width=54)
            (actual time=0.037..0.060 rows=10 loops=1)
              Filter: ((status < 5) AND ((status = 0) OR
                       (updated < (now() - '1 hour'::interval))))
Planning Time: 0.260 ms
Execution Time: 0.179 ms
```

Queues - Retry An Event

```
UPDATE queue.event
SET updated = now(),
    status = status + 1
WHERE ctid = '(719,117)';
```

Queues - Processed An Event

```sql
UPDATE queue.event
SET updated = now(),
    status = 2147483647
WHERE ctid = '(720,2)';
```

Queues - Partitioning

```
CREATE TABLE queue.event (
  hook_id     UUID        NOT NULL,
  created     TIMESTAMP   NOT NULL,
  updated     TIMESTAMP           ,
  status      INTEGER     NOT NULL,
  payload     TEXT
) PARTITION BY RANGE (created);
```

Queues - Partitioning

```
CREATE TABLE queue.event_2024_01
  PARTITION OF queue.event
  FOR VALUES FROM ('2024-01-01') TO ('2024-02-01');

...

CREATE TABLE queue.event_2024_12
  PARTITION OF queue.event_2024_12
  FOR VALUES FROM ('2024-12-01') TO ('2025-01-01');
```

Queues - Partition Retention

```
ALTER TABLE queue.event_2024_01
  DETACH PARTITION queue.event;

-- Archive old partition?
COPY queue.event_2024_01
  TO ‘archive/events_2024_01;

DROP TABLE queue.event_2024_01;
```

Text Search

Text Search - Simple

```
CREATE TABLE search.content (
   entity_id     UUID          NOT NULL,
   entity_type   TEXT          NOT NULL,
   content       TEXT          NOT NULL,
   vector        TSVECTOR      NOT NULL
);
```

Text Search - Simple

```
INSERT INTO search.content
VALUES (...,
  to_tsvector('english', 'Some pages
about Bridgnorth Library.  Where you can
borrow books, while the politicians still
allow.')
);
```

Text Search - Simple

```
SELECT
  ts_rank_cd(vector, to_tsquery(…)), *
FROM search.content
WHERE vector @@ to_tsquery('english',
  'bridgnorth & library')
ORDER BY 1;
```

Text Search - Simple

```sql
CREATE INDEX content_ftx
ON search.content
USING GIN (vector);
```

Text Search - Tack On

```
CREATE TABLE bergamot.host (
  id           UUID      NOT NULL,
  hostname     TEXT      NOT NULL,
  summary      TEXT              ,
  description  TEXT
);
```

Text Search - Tack On

```sql
CREATE INDEX host_text_idx
ON bergamot.host
USING GIN (to_tsvector('english',
             description));
```

Text Search - Simple

```
SELECT *
FROM bergamot.host
WHERE
  to_tsvector('english', description) @@
  to_tsquery('english', 'webserver');
```

GIS

Location Search

```
CREATE TABLE club.venue (
   Id            UUID        NOT NULL,
   name          TEXT        NOT NULL,
   description   TEXT        NOT NULL,
   address       TEXT        NOT NULL,
   location      POINT
);
```

Location Search

```sql
SELECT *
FROM club.venue
WHERE st_dwithin(location, my_location, 2000);
```

Data Modelling

Arrays

Tags

```
CREATE TABLE bergamot.host (
  id            UUID       NOT NULL,
  group_id      UUID       NOT NULL,
  hostname      TEXT       NOT NULL,
  ...
  tags          TEXT[]             ,
);
```

Tags

```
SELECT *
FROM bergamot.host
WHERE tags @> ARRAY['web'];

SELECT *
FROM bergamot.host
WHERE tags @> ARRAY['web', 'app1'];
```

Tags

```
CREATE INDEX tags_idx
ON bergamot.host
   USING GIN (tags);
```

Roll Ups

```
CREATE TABLE iot.daily_reading (
  meter_id       UUID       NOT NULL,
  read_range     DATERANGE  NOT NULL,
  energy         BIGINT,
  energy_profile BIGINT[],
  PRIMARY KEY (device_id, read_range)
);
```

Roll Ups

t_xmin	t_xmax	t_cid	t_xvac	t_ctid	t_infomask 2	t_infomask	t_hoff
4	4	4	4	6	2	2	1

24 bytes

device_id	read_at	temperature	light
16	8	4	4

32 bytes

Going Over The Top

- Like with everything, there can be too much of a good thing
- In one DB design, is used arrays to capture all the many-to-many relationships
- It worked pretty well in some ways
- But the lack of foreign keys was a bit of a nightmare
- It probably ended up being more hassle than just implementing all the mapping tables.

Unknown Unknowns

Unknown Unknowns

```
CREATE TABLE insurance.quote (
  id           UUID         NOT NULL,
  customer_id  UUID         NOT NULL,
  status       STATUS       NOT NULL,
  price        NUMERIC      NOT NULL,
  answers      JSONB
);
```

Unknown Unknowns

```
SELECT count(*),
       count(*) FILTER (WHERE (answers ->> 'locks')
                                IS NULL),
       count(*) FILTER (WHERE (answers ->> 'locks')
                                IS NOT NULL),
       count(*) FILTER (WHERE (answers ->> 'locks')
                                = '3-level'),
       count(*) FILTER (WHERE (answers ->> 'locks')
                                = 'unknown')
FROM insurance.quotes;
```

Stopping Things Going Wrong

Subscriptions

```
CREATE TABLE club.subscription (
  id          UUID      NOT NULL,
  member_id   UUID      NOT NULL,
  plan_id     UUID      NOT NULL,
  status      STATUS    NOT NULL,
  ...
);
```

Subscriptions

```
CREATE UNIQUE INDEX active_subs
ON club.subscription
   (member_id)
WHERE status = 'active';
```

Problem Solving With SQL

Pulling Things Together

```sql
SELECT *
FROM search.content
WHERE vector @@ to_tsquery('library')
AND st_dwithin(location, my_location, 2000)
AND tags @> ARRAY['service_catalogue'];
```

Recursion

```
WITH RECURSIVE groups(id) AS (
      SELECT g.id FROM bergamot.group g
      WHERE g.id = <id>
    UNION
      SELECT g.id FROM bergamot.group g, groups gg
      WHERE g.parent_id = gg.id
)
SELECT id, bool_and(s.ok OR s.suppressed) AS ok
FROM groups
JOIN status s ON (s.id = groups.id);
```

Lateral Joins

```
SELECT h.*, q.*
FROM bergamot.hosts h
LEFT JOIN LATERAL (
    SELECT sampled, load_avg_5
    FROM metrics.cpu c
    WHERE c.host_id = h.id
    ORDER BY sampled DESC
    LIMIT 1
) q ON (true);
```

Writable CTEs

```
WITH invoice_commission AS (
    UPDATE billing.commission_record
    SET invoice_id = 123
    WHERE invoice_id IS NULL
    RETURNING *
) INSERT INTO billing.invoice
SELECT 123, current_date, sum(value) AS total
FROM invoice_commission;
```

Generate Series - Presenting Data

```
SELECT r.device_id, t.time, array_agg(r.read_at),
       avg(r.temperature), avg(r.light)
FROM generate_series(
   '2022-10-06 00:00:00'::TIMESTAMP,
   '2022-10-07 00:00:00'::TIMESTAMP, '10 minutes') t(time)
JOIN iot.alhex_reading r
   ON (r.device_id = '26170b53-ae8f-464e-8ca6-2faeff8a4d01'::UUID
       AND r.read_at >= t.time
       AND r.read_at < (t.time + '10 minutes'))
GROUP BY 1, 2
ORDER BY t.time;
```

Window Functions

Window Functions - Roll Up

```
SELECT
  commission AS daily_total,
  sum(commission) OVER
  (PARTITION BY date_trunc('week', day))
   AS weekly_total
FROM billing.daily;
```

Window Functions - Moving On Up

```
SELECT load_user, avg(load_user) OVER
 (ORDER BY day
  ROWS BETWEEN 2 PRECEDING
  AND CURRENT ROW)
   AS moving_average
FROM metrics.application_cpu;
```

Window Functions - Counters

```sql
SELECT
 day,
 energy,
 energy - coalesce(lag(energy)
    OVER (ORDER BY day), 0) AS consumed
FROM iot.meter_reading
ORDER BY day;
```

Mind The Gap

Custom Aggregates - Mind The Gap

```sql
WITH days AS (
  SELECT t.day::DATE
  FROM generate_series('2017-01-01'::DATE,
'2017-01-15'::DATE, '1 day') t(day)
), data AS (
   SELECT *
   FROM iot.meter_reading
   WHERE day >= '2017-01-01'::DATE
   AND   day <= '2017-01-15'::DATE
)
```

Custom Aggregates - Mind The Gap

```
SELECT day,
       coalesce(energy,
         (((next_read - last_read)
            / (next_read_time - last_read_time))
            * (day - last_read_time))
            + last_read) AS energy_interpolated
FROM (
    … from next slide …
) q
ORDER BY day
```

Custom Aggregates - Mind The Gap

```
SELECT t.day, d.energy,
  last(d.day)    OVER lookback    AS last_read_time,
  last(d.day)    OVER lookforward AS next_read_time,
  last(d.energy) OVER lookback    AS last_read,
  last(d.energy) OVER lookforward AS next_read
FROM days t
LEFT JOIN data d ON (t.day = d.day)
WINDOW
  lookback AS (ORDER BY t.day),
  lookforward AS (ORDER BY t.day DESC)
```

Custom Aggregates - Mind The Gap

```
CREATE FUNCTION last_agg(anyelement, anyelement)
RETURNS anyelement LANGUAGE SQL IMMUTABLE STRICT AS $$
        SELECT $2;
$$;

CREATE AGGREGATE last (
        sfunc = last_agg,
        basetype = anyelement,
        stype = anyelement
);
```

Any Questions?

Appendix - Mind The Gap

```
WITH days AS (
  SELECT t.day::DATE
  FROM generate_series('2017-01-01'::DATE, '2017-01-15'::DATE, '1 day') t(day)
), data AS (
       SELECT *
       FROM iot.meter_reading
       WHERE day >= '2017-01-01'::DATE AND day <= '2017-01-15'::DATE
)
SELECT day, coalesce(energy_import_wh, (((next_read - last_read) / (next_read_time - last_read_time)) * (day -
last_read_time)) + last_read) AS energy_import_wh_interpolated
FROM (
  SELECT t.day, d.energy_import_wh,
       last(d.day) OVER lookback AS last_read_time,
       last(d.day) OVER lookforward AS next_read_time,
       last(d.energy_import_wh) OVER lookback AS last_read,
       last(d.energy_import_wh) OVER lookforward AS next_read
  FROM days t
  LEFT JOIN data d ON (t.day = d.day)
  WINDOW
       lookback AS (ORDER BY t.day),
       lookforward AS (ORDER BY t.day DESC)
) q  ORDER BY q.day
```