Top 5 PostgreSQL Query Tuning Tips!

Senior Database Consultant

Janis Griffin

Quest

Where Next Meets Now.

Who Am I?

Current – 30+ Years in Oracle®, DB2®, ASE, SQL Server®, MySQL®, PostgreSQL

DBA and Developer

- Specialize in Performance Tuning
- Customers Common Question: How do I tune it?

Oracle ACE Pro

Janis.Griffin@quest.com

X - @DoBoutAnything

Quest®

Where Next Meets Now.

Agenda

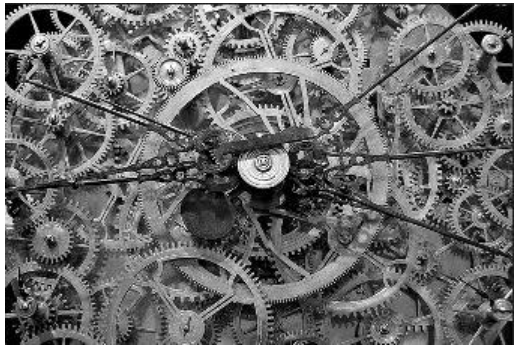
Quest

- **Challenges of Tuning**
- **My 5 Tips**
  - Monitor Wait Time
  - Review the Explain Plan
  - Gather Object Information
  - Find the Driving Table
  - Engineer out the Stupid
- **Several Case Studies**

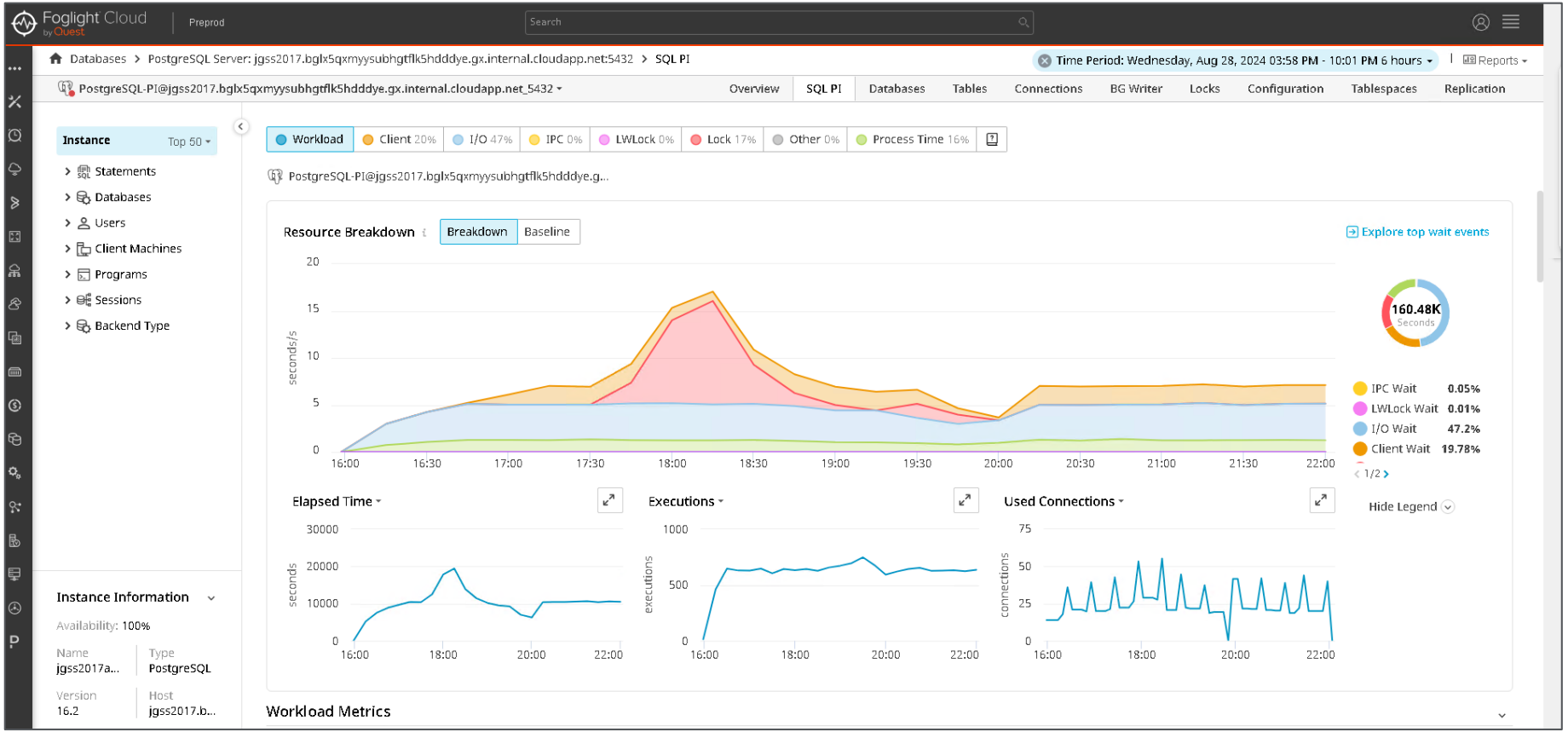
Challenges of Tuning

Quest

- **SQL Tuning is Hard**
  - Who should tune – DBA or Developer
  - Which SQL to tune
- **Requires Expertise in Many Areas**
  - Technical – Plan, Data Access, SQL Design
  - Business – What is the Purpose of SQL?
- **Tuning Takes Time**
  - Large Number of SQL Statements
  - Each Statement is Different
- **Low Priority in Some Companies**
  - Vendor Applications
  - Focus on Hardware or System Issues
- **Never Ending**

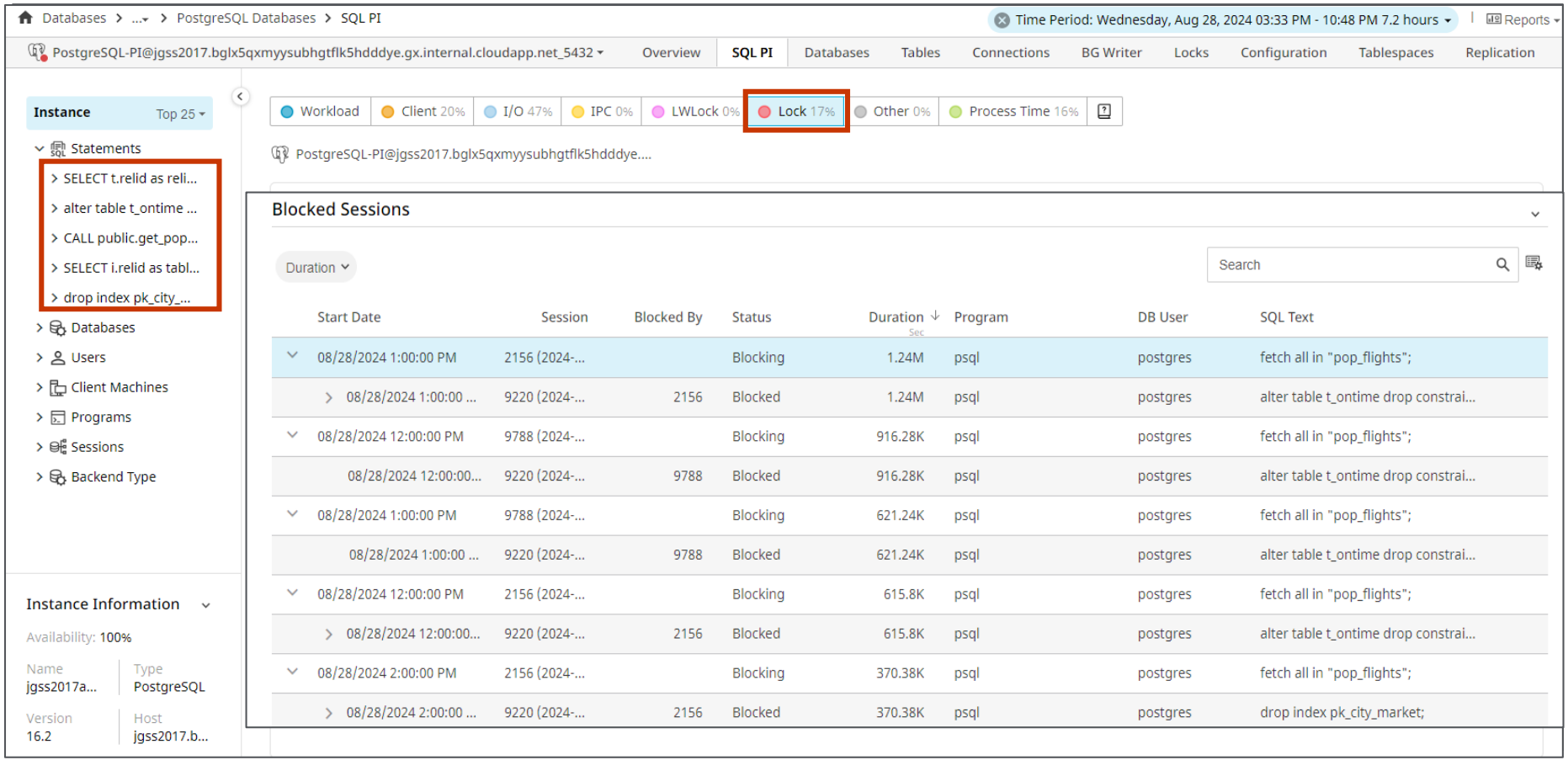


Monitor Wait Time – All levels



Monitor Wait Time – Statement Level

Quest



Review the Explain Plan

Quest

- **EXPLAIN Command - explain description & options**
  - Gives estimated costs (start_up / total cost) as it doesn't actually run it

```
dvdrental=# explain select * from film where title like '%Braveheart%';
                        QUERY PLAN
-----------------------------------------------------------
 Seq Scan on film  (cost=0.00..76.50 rows=1 width=384)
   Filter: ((title)::text ~~ '%Braveheart%'::text)
(2 rows)
```

- **EXPLAIN analyze - what is explain cost**
  - Executes the query so actual run time statistics are shown

```
dvdrental=# explain analyze select * from film where title like '%Braveheart%';
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 Seq Scan on film  (cost=0.00..76.50 rows=1 width=384) (actual time=0.064..0.421 rows=1 loops=1)
   Filter: ((title)::text ~~ '%Braveheart%'::text)
   Rows Removed by Filter: 999
 Planning Time: 0.157 ms
 Execution Time: 0.450 ms
(5 rows)
```

Examine the Explain Plan

Quest

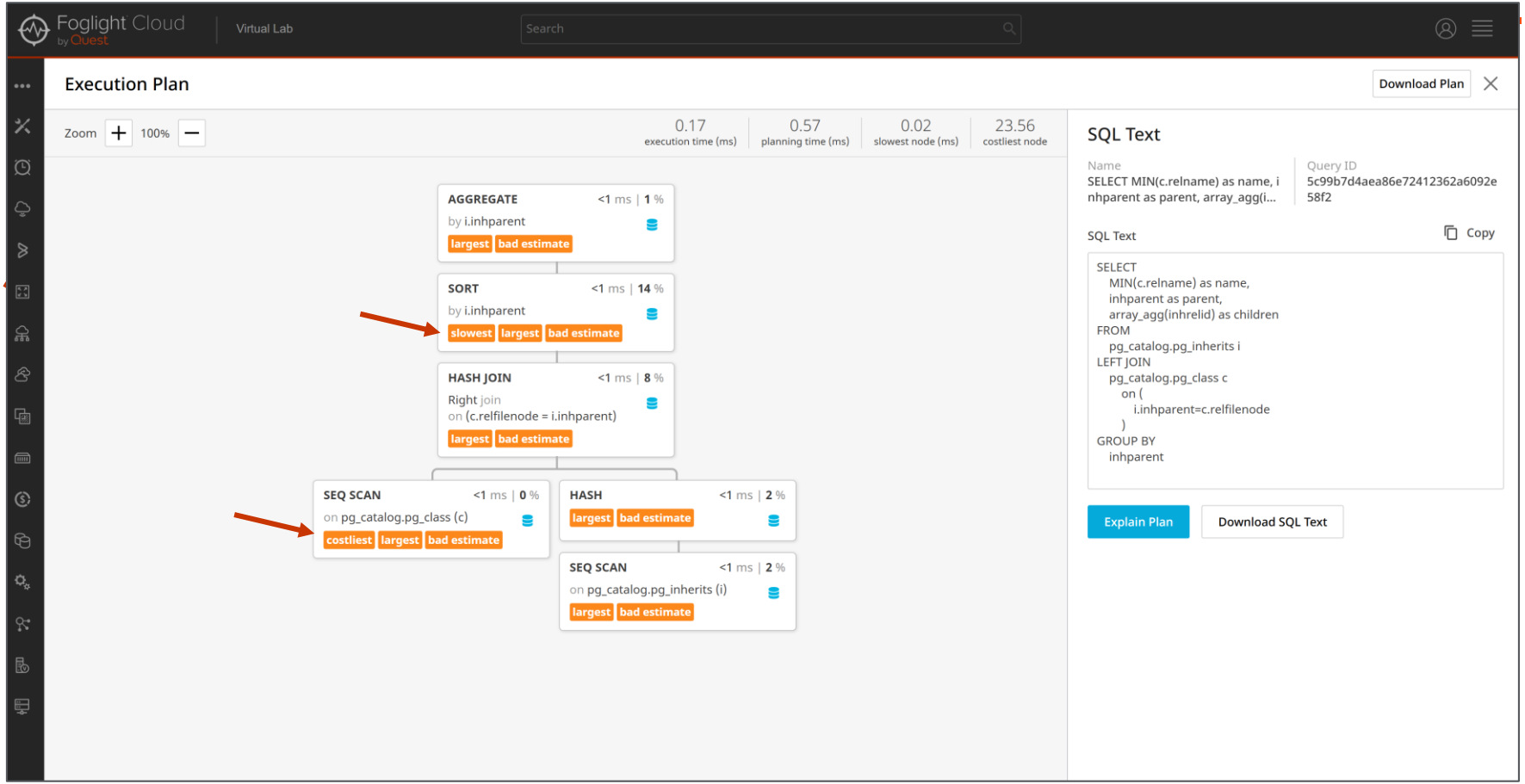
- **Find Expensive Operators**
  - Examine costs and row counts (shows the # of rows processed – not what it evaluated)
    - Need to look at **# rows removed**
    - Gives an estimate of resources (CPU and disk I/O)
  - Look for Seq Scan or Index Scan
- **Review the Filter Conditions**
  - Know which step filtering predicate is applied
- **Review Join Methods**
  - Nested Loops join: Usually efficient for smaller data sets
  - Hash Join: Useful on very large data sets (DW)
  - Merge Join: Efficient for larger data sets

Explain Plan - Look for Common Mistakes

- **Identify Common Mistakes**
  - Using functions on indexed columns
    - In WHERE, ON & HAVING clause
    - Create a Functional Index instead
      - Create index lower_title_idx on film(lower(title));
  - Nested views
    - One view calling or joining to other views
  - Use of cursors or row by row processing
- **Missing or Poor Indexing**
- **Problems Outside of the Plan**
  - Missing or stale statistics
  - Database misconfiguration
  - No database constraints

Graphical Explain Plan

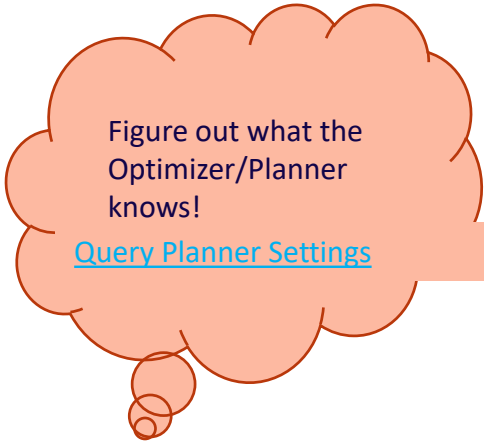
Quest



Gather Object Information

Quest

- **Understand objects in explain plans**
  - Table Definitions & Sizes
    - Is it a View?
      - > Get underlying definition
    - Number of Rows / Partitioning?
  - Examine Columns in Where Clause
    - Know the Cardinality of columns
    - Is there Data Skew
      - > Consider partial index
    - Are there indexes on the join / filtering columns
  - Index & Constraint Definitions
    - Entity Relationship Diagrams (ERDs) can help
- **Statistics Collection Configuration**
  - Analyze / Vacuum



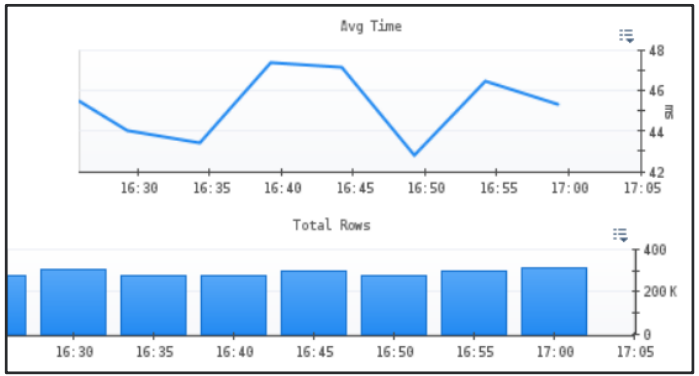
Case Study

Who registered yesterday for SQL Tuning Class?

Quest

Who registered yesterday for SQL Tuning

```
PREPARE billing (timestamp,timestamp) as
SELECT s.fname, s.lname, r.signup_date
FROM   test.student s
    INNER JOIN test.registration r ON s.student_id = r.student_id
    INNER JOIN test.class c ON r.class_id = c.class_id
WHERE  c.name  = 'SQL TUNING'
AND    r.signup_date BETWEEN $1 AND $2
AND    r.cancelled ='N';
```



Statement Summary

Server	jgss2017.internal.cloudapp.net:5432
Query ID	3075482329797333000
User	postgres
Database	test
Calls	5.4 K
Avg Time	45 ms
Total Time	24.5 min
Avg Rows	52.3
Query	PREPARE billing (timestamp,timestamp) as SELECT s.fname, s.lname, r.signup_date FROM test.student s INNER JOIN test.registration r ON s.student_id = r.student_id INNER JOIN test.class c ON r.class_id = c.class_id WHERE c.name = $3 AND r.signup_date BETWEEN $1 AND $2 AND r.cancelled = $4

Using Explain on Prepared Statements (version 16+)

Quest

- **Prepared Statements & PL/pgSQL functions use Cached Plans**
  - Postgres does not automatically cache plans for standard SQL statements

```
test=#
test=# explain (GENERIC_PLAN) SELECT s.fname, s.lname, r.signup_date
test-# FROM   test.student s
test-#     INNER JOIN test.registration r ON s.student_id = r.student_id
test-#     INNER JOIN test.class c ON r.class_id = c.class_id
test-# WHERE  c.name  = 'SQL TUNING'
test-# AND   r.signup_date BETWEEN $1 AND $2
test-# AND   r.cancelled ='N';
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.29..1950.45 rows=1 width=35)
   ->  Nested Loop  (cost=0.00..1948.11 rows=1 width=13)
         Join Filter: (r.class_id = c.class_id)
         ->  Seq Scan on registration r  (cost=0.00..1909.67 rows=398 width=18)
               Filter: ((signup_date >= $1) AND (signup_date <= $2) AND (cancelled = 'N'::bpchar))
         ->  Materialize  (cost=0.00..26.51 rows=2 width=5)
               ->  Seq Scan on class c  (cost=0.00..26.50 rows=2 width=5)
                     Filter: ((name)::text = 'SQL TUNING'::text)
   ->  Index Scan using pk_student on student s  (cost=0.29..2.33 rows=1 width=32)
         Index Cond: (student_id = r.student_id)
(10 rows)
```

Explain (Analyze, Buffers)

Quest

```
explain (analyze, buffers)
SELECT s.fname, s.lname, r.signup_date
FROM   test.student s
    INNER JOIN test.registration r ON s.student_id = r.student_id
    INNER JOIN test.class c ON r.class_id = c.class_id
WHERE  c.name  = 'SQL TUNING'
AND   r.signup_date BETWEEN '2014-03-21 00:00:00' and '2014-03-28 00:00:00'
AND   r.cancelled ='N';

                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=26.81..2045.81 rows=67 width=35) (actual time=1.181..32.482 rows=57 loops=1)
   Buffers: shared hit=695
   ->  Hash Join  (cost=26.52..2023.92 rows=67 width=13) (actual time=1.159..31.964 rows=57 loops=1)
         Hash Cond: (r.class_id = c.class_id)
         Buffers: shared hit=524
         ->  Seq Scan on registration r  (cost=0.00..1909.67 rows=33279 width=18) (actual time=0.030..24.081 rows=33259 loops=1)
               Filter: ((signup_date >= '2014-03-21 00:00:00'::timestamp without time zone)
                    AND (signup_date <= '2014-03-28 00:00:00'::timestamp without time zone)
                    AND (cancelled = 'N'::bpchar))
               Rows Removed by Filter: 46722
               Buffers: shared hit=510
         ->  Hash  (cost=26.50..26.50 rows=2 width=5) (actual time=0.256..0.257 rows=2 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 9kB
               Buffers: shared hit=14
               ->  Seq Scan on class c  (cost=0.00..26.50 rows=2 width=5) (actual time=0.043..0.251 rows=2 loops=1)
                     Filter: ((name)::text = 'SQL TUNING'::text)
                     Rows Removed by Filter: 998
                     Buffers: shared hit=14
   ->  Index Scan using pk_student on student s  (cost=0.29..0.33 rows=1 width=32) (actual time=0.007..0.007 rows=1 loops=57)
         Index Cond: (student_id = r.student_id)
         Buffers: shared hit=171
 Planning:
   Buffers: shared hit=20
 Planning Time: 0.693 ms
 Execution Time: 32.553 ms
```

PKs / FKs only

Review Table & Indexes

Quest

```
test=# \d test.registration
                        Table "test.registration"
   Column    |            Type             | Collation | Nullable | Default
-------------+-----------------------------+-----------+----------+---------
 student_id  | numeric(18,0)               |           |          |
 class_id    | numeric(18,0)               |           |          |
 cancelled   | character(1)                |           |          |
 signup_date | timestamp without time zone |           |          |
Indexes:
    "pk_registration" UNIQUE CONSTRAINT, btree (student_id, class_id, signup_date)
Foreign-key constraints:
    "registration_class_id_fkey" FOREIGN KEY (class_id) REFERENCES test.class(class_id)
    "registration_student_id_fkey" FOREIGN KEY (student_id) REFERENCES test.student(student_id)
```

# of Rows
class 1,000
student 10,000
registration 79,981

```
test=# SELECT tablename, indexname FROM pg_indexes WHERE tablename in ('class','student','registration');
  tablename   |    indexname
--------------+-----------------
 student      | pk_student
 class        | pk_class
 registration | pk_registration
(3 rows)
```

```
test=# \di+ test.*
                                                 List of relations
 Schema |      Name       | Type  |  Owner   |    Table     | Persistence | Access method |  Size
--------+-----------------+-------+----------+--------------+-------------+---------------+---------
 test   | pk_class        | index | postgres | class        | permanent   | btree         | 56 kB
 test   | pk_registration | index | postgres | registration | permanent   | btree         | 4248 kB
 test   | pk_student      | index | postgres | student      | permanent   | btree         | 304 kB
(3 rows)
```

Find the Driving table

Quest

- **Need to know the size of the actual data sets in each step**
  - In Joins (Right, Left, Outer)
  - What are the filtering predicates
  - When is each filtering predicate applied
    - Try to filter earlier rather than later
- **Compare size of final result set with # of rows at each step**
- **Find the driving table**
    - To reduce buffers (I/O)

```
SELECT s.fname, s.lname, r.signup_date
FROM   student s
   INNER JOIN registration r ON s.student_id = r.student_id
   INNER JOIN class c ON r.class_id = c.class_id
WHERE  c.name  = 'SQL TUNING'
AND    r.signup_date BETWEEN  $1 AND $2
AND    r.cancelled = 'N'
```

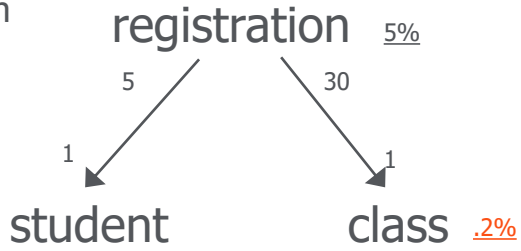
Joins

Filtering Predicates

SQL Diagramming

Quest

- **Great Book “SQL Tuning” by Dan Tow**
  - Oldie but a goodie that teaches SQL Diagramming
  - http://www.singingsql.com




```
select count(1) from registration where cancelled = 'N'
and signup_date between `2022-12-10 00:00' and `2022-12-11 00:00'

4344 / 79,981 * 100  = 5.43%

select count(1) from class where name = 'SQL TUNING'

 2 / 1000 * 100 = .2%
```

Drive the Query with Class

Quest

- CREATE INDEX cl_name ON test.class(name);

```
                                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
Nested Loop  (cost=10.36..2029.36 rows=67 width=35) (actual time=1.117..31.488 rows=57 loops=1)
  Buffers: shared hit=685
  ->  Hash Join  (cost=10.07..2007.46 rows=67 width=13) (actual time=1.096..31.019 rows=57 loops=1)
        Hash Cond: (r.class_id = c.class_id)
        Buffers: shared hit=514
        ->  Seq Scan on registration r  (cost=0.00..1909.67 rows=33279 width=18) (actual time=0.017..23.553 rows=33259 loops=1)
              Filter: ((signup_date >= '2014-03-21 00:00:00'::timestamp without time zone)
                   AND (signup_date <= '2014-03-28 00:00:00'::timestamp without time zone)
                   AND (cancelled = 'N'::bpchar))
              Rows Removed by Filter: 46722
              Buffers: shared hit=510
        ->  Hash  (cost=10.05..10.05 rows=2 width=5) (actual time=0.069..0.070 rows=2 loops=1)
              Buckets: 1024  Batches: 1  Memory Usage: 9kB
              Buffers: shared hit=4
              ->  Bitmap Heap Scan on class c  (cost=4.29..10.05 rows=2 width=5) (actual time=0.060..0.064 rows=2 loops=1)
                    Recheck Cond: ((name)::text = 'SQL TUNING'::text)
                    Heap Blocks: exact=2
                    Buffers: shared hit=4
                    ->  Bitmap Index Scan on cl_name  (cost=0.00..4.29 rows=2 width=0) (actual time=0.054..0.054 rows=2 loops=1)
                          Index Cond: ((name)::text = 'SQL TUNING'::text)
                          Buffers: shared hit=2
  ->  Index Scan using pk_student on student s  (cost=0.29..0.33 rows=1 width=32) (actual time=0.007..0.007 rows=1 loops=57)
        Index Cond: (student_id = r.student_id)
        Buffers: shared hit=171
Planning:
  Buffers: shared hit=50
Planning Time: 5.034 ms
Execution Time: 31.594 ms
```

```
Planning Time: 0.693 ms
Execution Time: 32.553 ms
```

Why Seq Scan on Registration?

Quest

- Can’t use Primary Key as class_id is not left leading column

```
test=# \d test.registration
                              Table "test.registration"
   Column    |            Type             | Collation | Nullable |      Default
-------------+-----------------------------+-----------+----------+-------------------
 student_id  | numeric(18,0)               |           |          |
 class_id    | numeric(18,0)               |           |          |
 cancelled   | character(1)                |           |          |
 signup_date | timestamp without time zone |           |          | CURRENT_TIMESTAMP
Indexes:
    "pk_registration" UNIQUE CONSTRAINT, btree (student_id, class_id, signup_date)
Foreign-key constraints:
    "registration_class_id_fkey" FOREIGN KEY (class_id) REFERENCES test.class(class_id)
    "registration_student_id_fkey" FOREIGN KEY (student_id) REFERENCES test.student(student_id)
```

- Not much difference in throughput – 6k vs 5.4k (685 vs 695 buffers)
  - Needs more information to drive by Class

Add Index on Registration (Class_id)

Quest

- create index REG_ALT on test.registration(class_id);

```
                                   QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------
Nested Loop  (cost=9.60..450.25 rows=67 width=35) (actual time=0.107..0.750 rows=57 loops=1)
  Buffers: shared hit=328
  ->  Nested Loop  (cost=9.32..428.36 rows=67 width=13) (actual time=0.093..0.389 rows=57 loops=1)
        Buffers: shared hit=157
        ->  Bitmap Heap Scan on class c  (cost=4.
              Recheck Cond: ((name)::text = 'SQL
              Heap Blocks: exact=2
              Buffers: shared hit=4
              ->  Bitmap Index Scan on cl_name  (
                    Index Cond: ((name)::text = '
                    Buffers: shared hit=2
        ->  Bitmap Heap Scan on registration r  (
              Recheck Cond: (class_id = c.class_i
              Filter: ((signup_date >= '2014-03-2
                    AND (signup_date <= '2014-03-28
                    AND (cancelled = 'N'::bpchar))
              Rows Removed by Filter: 47
              Heap Blocks: exact=146
              Buffers: shared hit=153
              ->  Bitmap Index Scan on reg_alt  (
                    Index Cond: (class_id = c.cla
                    Buffers: shared hit=7
  ->  Index Scan using pk_student on student s  (
        Index Cond: (student_id = r.student_id)
        Buffers: shared hit=171
Planning:
  Buffers: shared hit=46
Planning Time: 3.223 ms
Execution Time: 0.833 ms
```

```
Nested Loop  (cost=26.81..2045.81 rows=67 width=35) (actual time=1.181..32.482 rows=57 loops=1)
  Buffers: shared hit=695
  ->  Hash Join  (cost=26.52..2023.92 rows=67 width=13) (actual time=1.159..31.964 rows=57 loops=1)
        Hash Cond: (r.class_id = c.class_id)
        Buffers: shared hit=524
        ->  Seq Scan on registration r  (cost=0.00..1909.67 rows=33279 width=18) (actual time=0.030..24.081 rows=33
              Filter: ((signup_date >= '2014-03-21 00:00:00'::timestamp without time zone)
                    AND (signup_date <= '2014-03-28 00:00:00'::timestamp without time zone)
                    AND (cancelled = 'N'::bpchar))
              Rows Removed by Filter: 46722
              Buffers: shared hit=510
        ->  Hash  (cost=26.50..26.50 rows=2 width=5) (actual time=0.256..0.257 rows=2 loops=1)
              Buckets: 1024  Batches: 1  Memory Usage: 9kB
              Buffers: shared hit=14
              ->  Seq Scan on class c  (cost=0.00..26.50 rows=2 width=5) (actual time=0.043..0.251 rows=2 loops=1)
                    Filter: ((name)::text = 'SQL TUNING'::text)
                    Rows Removed by Filter: 998
                    Buffers: shared hit=14
  ->  Index Scan using pk_student on student s  (cost=0.29..0.33 rows=1 width=32) (actual time=0.007..0.007 rows=1
        Index Cond: (student_id = r.student_id)
        Buffers: shared hit=171
Planning:
  Buffers: shared hit=20
Planning Time: 0.693 ms
Execution Time: 32.553 ms
```

Add Covering Index on Registration

Quest

- create index REG_ALT on test.registration(class_id, student_id, signup_date) include (cancelled);

```
                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------
Nested Loop  (cost=4.99..44.26 rows=67 width=35) (actual time=0.099..0.455 rows=57 loops=1)
  Buffers: shared hit=183
  ->  Nested Loop  (cost=4.71..22.37 rows=67 width=13) (actual time=0.084..0.139 rows=57 loops=1)
        Buffers: shared hit=12
        ->  Bitmap Heap Scan on class c  (cost=4.29..10.05 rows=2 width=5) (actual time=0.035..0.037 rows=2 loops=1)
              Recheck Cond: ((name)::text = 'SQL TUNING'::text)
              Heap Blocks: exact=2
              Buffers: shared hit=4
              ->  Bitmap Index Scan on cl_name  (cost=0.00..4.29 rows=2 width=0) (actual time=0.028..0.028 rows=2 loops=1)
                    Index Cond: ((name)::text = 'SQL TUNING'::text)
                    Buffers: shared hit=2
        ->  Index Only Scan using reg_alt on registration r  (cost=0.42..5.83 rows=33 width=18) (actual time=0.030..0.045 rows=29 loops=2)
              Index Cond: ((class_id = c.class_id)
                        AND (signup_date >= '2014-03-21 00:00:00'::timestamp without time zone)
                        AND (signup_date <= '2014-03-28 00:00:00'::timestamp without time zone))
              Filter: (cancelled = 'N'::bpchar)
              Heap Fetches: 0
              Buffers: shared hit=8
  ->  Index Scan using pk_student on student s  (cost=0.29..0.33 rows=1 width=32) (actual time=0.005..0.005 rows=1 loops=57)
        Index Cond: (student_id = r.student_id)
        Buffers: shared hit=171
Planning:
  Buffers: shared hit=28
Planning Time: 0.862 ms
Execution Time: 0.533 ms
```

```
Planning Time: 3.223 ms
Execution Time: 0.833 ms
```

```
Planning Time: 0.693 ms
Execution Time: 32.553 ms
```

Shared Hits

IX 1: 695

IX 2: 328

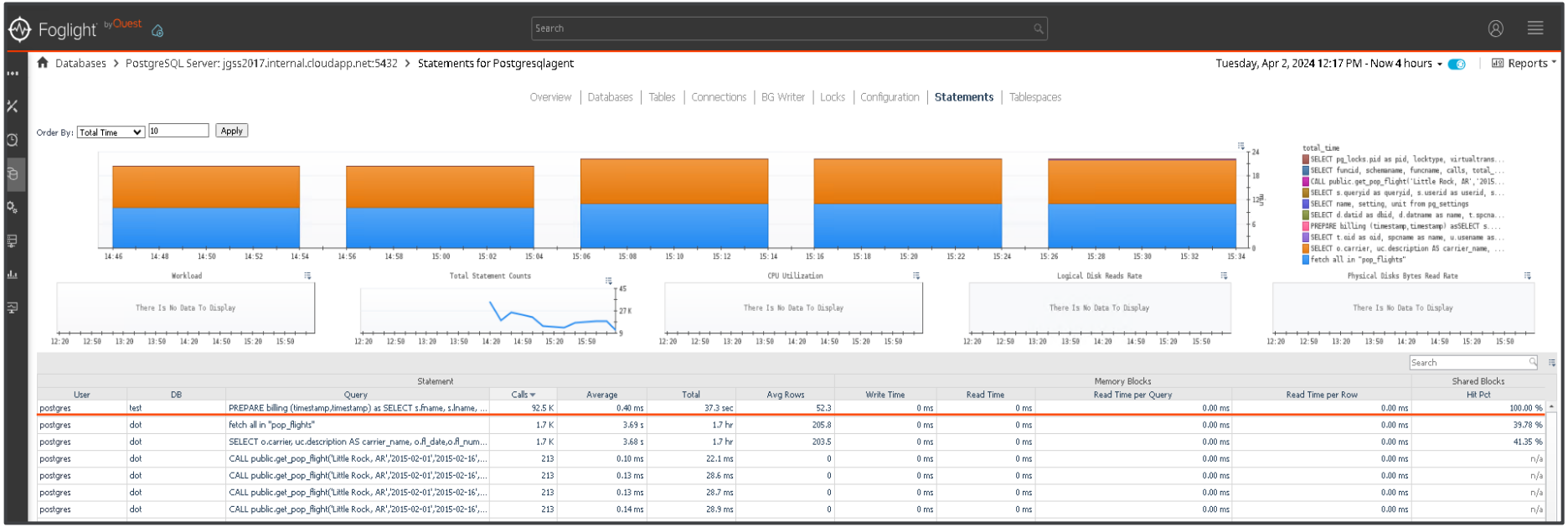
IX 3: 183

Case Study

Flights by City & Day of Week

Quest

SQL Taking the Most Time



Flights by City & Day of Week

Quest

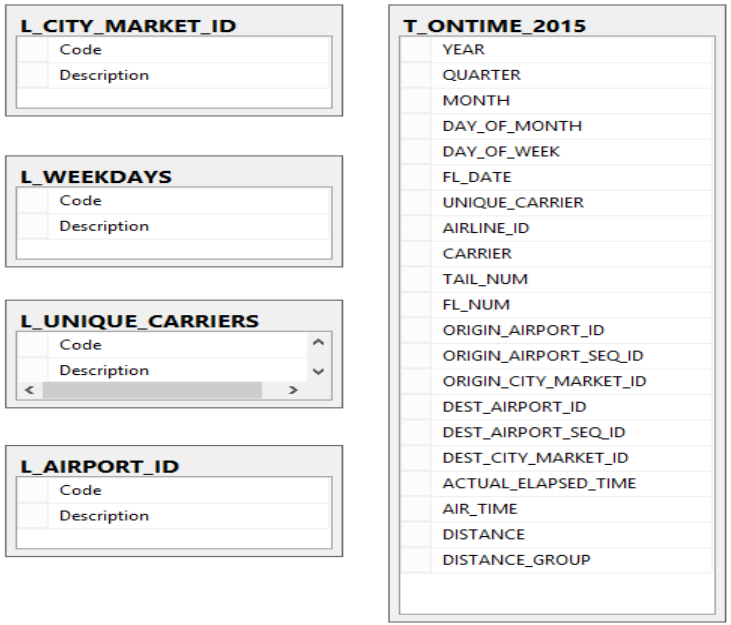
```
CREATE OR REPLACE PROCEDURE
    get_pop_flight(_city varchar,_beg_date date, _end_date date,_day_of_week varchar, INOUT pop_flights refcursor)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
OPEN pop_flights FOR SELECT o.carrier, uc.description AS carrier_name, o.fl_date,o.fl_num,o.tail_num
                    ,ao.description AS origin_airport,co.Description AS origin_city ,ad.description AS destination_airport
                    ,cd.Description AS destination_city ,w.Description Day_of_Wed
FROM public.t_ontime o
INNER JOIN L_UNIQUE_CARRIERS AS uc ON uc.Code = o.UNIQUE_CARRIER
      INNER JOIN L_AIRPORT_ID AS ao ON ao.Code = o.ORIGIN_AIRPORT_ID
      INNER JOIN L_AIRPORT_ID AS ad ON ad.Code = o.DEST_AIRPORT_ID
      INNER JOIN L_CITY_MARKET_ID AS co ON co.Code = o.ORIGIN_CITY_MARKET_ID
      INNER JOIN L_CITY_MARKET_ID AS cd ON cd.Code = o.DEST_CITY_MARKET_ID
      INNER JOIN L_WEEKDAYS AS w ON w.Code = o.DAY_OF_WEEK
where fl_date BETWEEN _beg_date AND _end_date
      AND co.Description = _city
      AND w.Description = _day_of_week;
END;
$BODY$;
```

```
BEGIN;
CALL public.get_pop_flight('Little Rock, AR','2015-02-01','2015-0 16','Sunday','pop_flights');
fetch all in "pop_flights";
COMMIT;
```

Star Schema

Quest

- **US DOT - On-time Performance**



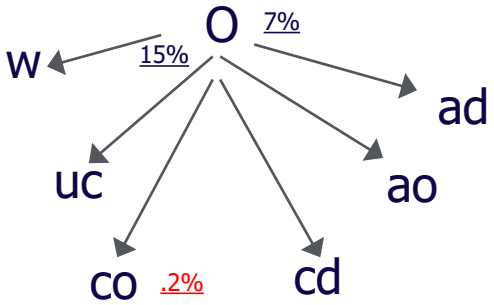
L_UNIQUE_CARRIERS:	1620
L_AIRPORT_ID:	6438
L_CITY_MARKET_ID:	5823
L_WEEKDAYS:	8
T_ONTIME:	6784044

Examine the Explain Plan

2.07seconds

```
                                                          QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------
Nested Loop  (cost=1016.79..159139.35 rows=1 width=263) (actual time=769.880..2069.920 rows=62 loops=1)
  Join Filter: (o.dest_city_market_id = cd.code)
  Rows Removed by Join Filter: 360964
  Buffers: shared hit=21617 read=103930
  ->  Nested Loop  (cost=1016.79..158970.33 rows=1 width=250) (actual time=769.617..1959.944 rows=62 loops=1)
        Join Filter: (o.dest_airport_id = ad.code)
        Rows Removed by Join Filter: 399094
        Buffers: shared hit=19261 read=103930
        ->  Nested Loop  (cost=1016.79..158767.48 rows=1 width=217) (actual time=769.282..1829.759 rows=62 loops=1)
              Join Filter: (o.origin_airport_id = ao.code)
              Rows Removed by Join Filter: 399094
              Buffers: shared hit=15665 read=103930
              ->  Nested Loop  (cost=1016.79..158564.62 rows=1 width=184) (actual time=768.353..1705.455 rows=62 loops=1)
                    Join Filter: ((o.unique_carrier)::bpchar = uc.code)
                    Rows Removed by Join Filter: 100378
                    Buffers: shared hit=12069 read=103930
                    ->  Nested Loop  (cost=1016.79..158515.17 rows=1 width=167) (actual time=768.195..1664.851 rows=62 loops=1)
                          Join Filter: (o.origin_city_market_id = co.code)
                          Rows Removed by Join Filter: 32132
                          Buffers: shared hit=11263 read=103930
                          ->  Seq Scan on l_city_market_id co  (cost=0.00..110.79 rows=1 width=21) (actual time=0.448..1.428 rows=1 loops=1)
                                Filter: ((description)::text = 'Little Rock, AR'::text)
                                Rows Removed by Filter: 5822
                                Buffers: shared hit=38
                          ->  Gather  (cost=1016.79..158360.67 rows=3497 width=154) (actual time=684.680..1615.868 rows=32194 loops=1)
                                Workers Planned: 2
                                Workers Launched: 2
                                Buffers: shared hit=11225 read=103930
                                ->  Hash Join  (cost=16.79..157010.97 rows=1457 width=154) (actual time=559.696..1481.533 rows=10731 loops=3)
                                      Hash Cond: (o.day_of_week = w.code)
                                      Buffers: shared hit=11225 read=103930
                                      ->  Parallel Seq Scan on t_ontime o
                                            (cost=0.00..156615.38 rows=97130 width=40) (actual time=558.148..1460.402 rows=76606 loops=3)
                                            Filter: ((fl_date >= '2015-02-01'::date) AND (fl_date <= '2015-02-16'::date))
                                            Rows Removed by Filter: 2145399
                                            Buffers: shared hit=11030 read=103930
                                      ->  Hash  (cost=16.75..16.75 rows=3 width=122) (actual time=0.966..0.968 rows=1 loops=3)
                                            Buckets: 1024  Batches: 1  Memory Usage: 9kB
                                            Buffers: shared hit=3
                                            ->  Seq Scan on l_weekdays w
                                                  (cost=0.00..16.75 rows=3 width=122) (actual time=0.934..0.936 rows=1 loops=3)
                                                  Filter: ((description)::text = 'Friday'::text)
                                                  Rows Removed by Filter: 7
                                                  Buffers: shared hit=3
                    ->  Seq Scan on l_unique_carriers uc  (cost=0.00..29.20 rows=1620 width=31) (actual time=0.004..0.237 rows=1620 loops=62)
                          Buffers: shared hit=806
              ->  Seq Scan on l_airport_id ao  (cost=0.00..122.38 rows=6438 width=41) (actual time=0.003..0.838 rows=6438 loops=62)
                    Buffers: shared hit=3596
        ->  Seq Scan on l_airport_id ad  (cost=0.00..122.38 rows=6438 width=41) (actual time=0.003..0.836 rows=6438 loops=62)
              Buffers: shared hit=3596
  ->  Seq Scan on l_city_market_id cd  (cost=0.00..96.23 rows=5823 width=21) (actual time=0.004..0.748 rows=5823 loops=62)
        Buffers: shared hit=2356
Planning Time: 2.060 ms
Execution Time: 2070.099 ms
```

Find the Driving Table



Filtering Selectivity

```
select count(1) from t_ontime where  origin_city_market_id = 32600;
select 12953.000/6666016.000 * 100;  = .194   or .2%

Select count(1) from t_ontime where fl_date
                between '2015-12-01 00:00:00.000' and'2015-12-31 00:00:00.000‘;
select 469717.00 / 6666016.00 * 100; = 7.04 or 7%

select count(1) from t_ontime where day_of_week = 1;
select 987647.00/6666016.00 * 100; = 14.81 or 15%
```

Tune the Query

- **Create index on T_ONTIME & L_CITY_MARKET_ID**
  - Create index CO_MARKET_DESC on public.L_CITY_MARKET_ID(description);
  - create index OCO_MARKET_DESC on public.T_ONTIME(origin_city_market_id);
  - create unique index PK_CITY_MARKET on public.L_CITY_MARKET_ID(code);

```
dot=# \d public.t_ontime
                    Table "public.t_ontime"
       Column        |         Type          | Collation |
---------------------+-----------------------+-----------+
 year                | integer               |           |
 quarter             | integer               |           |
 month               | integer               |           |
 day_of_month        | integer               |           |
 day_of_week         | integer               |           |
 fl_date             | date                  |           |
 unique_carrier      | character varying(10) |           |
 airline_id          | integer               |           |
 carrier             | character varying(10) |           |
 tail_num            | character varying(10) |           |
 fl_num              | integer               |           |
 origin_airport_id   | integer               |           |
 origin_airport_seq_id | integer             |           |
 origin_city_market_id | integer             |           |
 dest_airport_id     | integer               |           |
 dest_airport_seq_id | integer               |           |
 dest_city_market_id | integer               |           |
 actual_elapsed_time | numeric(6,2)          |           |
 air_time            | numeric(6,2)          |           |
 flights             | numeric(6,2)          |           |
 distance            | numeric(6,2)          |           |
 distance_group      | numeric               |           |
 total_add_gtime     | numeric               |           |
Indexes:
    "oco_market_desc" btree (origin_city_market_id)
```

```
dot=# \d public.l_city_market_id
                  Table "public.l_city_market_id"
   Column    |          Type          | Collation | Nullable |
-------------+------------------------+-----------+----------+
 code        | integer                |           |          |
 description | character varying(100) |           |          |
Indexes:
    "co_market_desc" btree (description)
    "pk_city_market" UNIQUE, btree (code)
```

```
                                QUERY PLAN
------------------------------------------------------------------------------------------------------------------
Nested Loop  (cost=254.79..58285.68 rows=1 width=263) (actual time=3.836..321.645 rows=62 loops=1)
  Buffers: shared hit=8754 read=2787
  ->  Nested Loop  (cost=254.51..58285.37 rows=1 width=250) (actual time=3.795..320.654 rows=62 loops=1)
        Join Filter: (o.dest_airport_id = ad.code)
        Rows Removed by Join Filter: 399094
        Buffers: shared hit=8575 read=2780
        ->  Nested Loop  (cost=254.51..58082.52 rows=1 width=217) (actual time=2.872..212.043 rows=62 loops=1)
              Join Filter: (o.origin_airport_id = ao.code)
              Rows Removed by Join Filter: 399094
              Buffers: shared hit=4979 read=2780
              ->  Nested Loop  (cost=254.51..57879.66 rows=1 width=184) (actual time=2.129..101.717 rows=62 loops=1)
                    Join Filter: ((o.unique_carrier)::bpchar = uc.code)
                    Rows Removed by Join Filter: 100378
                    Buffers: shared hit=1383 read=2780
                    ->  Nested Loop  (cost=254.51..57830.21 rows=1 width=167) (actual time=2.016..65.792 rows=62 loops=1)
                          Join Filter: (o.day_of_week = w.code)
                          Rows Removed by Join Filter: 358
                          Buffers: shared hit=577 read=2780
                          ->  Nested Loop  (cost=254.51..57811.66 rows=40 width=53) (actual time=1.964..65.327 rows=420 loops=1)
                                Buffers: shared hit=577 read=2779
                                ->  Index Scan using co_market_desc on l_city_market_id co  (cost=0.28..8.30 rows=1 width=21) (actual time=0.063..0.066 rows=1 loops=1)
                                      Index Cond: ((description)::text = 'Little Rock, AR'::text)
                                      Buffers: shared hit=1 read=2
                                ->  Bitmap Heap Scan on t_ontime o  (cost=254.23..57795.26 rows=810 width=40) (actual time=1.891..65.043 rows=420 loops=1)
                                      Recheck Cond: (origin_city_market_id = co.code)
                                      Filter: ((fl_date >= '2015-02-01'::date) AND (fl_date <= '2015-02-16'::date))
                                      Rows Removed by Filter: 12533
                                      Heap Blocks: exact=3339
                                      Buffers: shared hit=576 read=2777
                                      ->  Bitmap Index Scan on oco_market_desc  (cost=0.00..254.03 rows=23146 width=0) (actual time=1.243..1.244 rows=12953 loops=1)
                                            Index Cond: (origin_city_market_id = co.code)
                                            Buffers: shared hit=1 read=13
                          ->  Materialize  (cost=0.00..16.77 rows=3 width=122) (actual time=0.000..0.000 rows=1 loops=420)
                                Buffers: shared read=1
                                ->  Seq Scan on l_weekdays w  (cost=0.00..16.75 rows=3 width=122) (actual time=0.026..0.027 rows=1 loops=1)
                                      Filter: ((description)::text = 'Friday'::text)
                                      Rows Removed by Filter: 7
                                      Buffers: shared read=1
                    ->  Seq Scan on l_unique_carriers uc  (cost=0.00..29.20 rows=1620 width=31) (actual time=0.003..0.224 rows=1620 loops=62)
                          Buffers: shared hit=806
              ->  Seq Scan on l_airport_id ao  (cost=0.00..122.38 rows=6438 width=41) (actual time=0.002..0.768 rows=6438 loops=62)
                    Buffers: shared hit=3596
        ->  Seq Scan on l_airport_id ad  (cost=0.00..122.38 rows=6438 width=41) (actual time=0.003..0.790 rows=6438 loops=62)
              Buffers: shared hit=3596
  ->  Index Scan using pk_city_market on l_city_market_id cd  (cost=0.28..0.30 rows=1 width=21) (actual time=0.011..0.011 rows=1 loops=62)
        Index Cond: (code = o.dest_city_market_id)
        Buffers: shared hit=179 read=7
Planning:
  Buffers: shared hit=5 read=15
Planning Time: 4.337 ms
Execution Time: 321.892 ms
```

```
Planning Time: 2.060 ms
Execution Time: 2070.099 ms
```

Adjust the index

Quest

- Create index OCO_MARKET_DESC_FL_DATE on public.T_ONTIME(origin_city_market_id, fl_date);

```
dot=# \d public.t_ontime
                        Table "public.t_ontime"
       Column         |         Type          | Collation | Nullable |
----------------------+-----------------------+-----------+----------+
 year                 | integer               |           |          |
 quarter              | integer               |           |          |
 month                | integer               |           |          |
 day_of_month         | integer               |           |          |
 day_of_week          | integer               |           |          |
 fl_date              | date                  |           |          |
 unique_carrier       | character varying(10) |           |          |
 airline_id           | integer               |           |          |
 carrier              | character varying(10) |           |          |
 tail_num             | character varying(10) |           |          |
 fl_num               | integer               |           |          |
 origin_airport_id    | integer               |           |          |
 origin_airport_seq_id | integer              |           |          |
 origin_city_market_id | integer              |           |          |
 dest_airport_id      | integer               |           |          |
 dest_airport_seq_id  | integer               |           |          |
 dest_city_market_id  | integer               |           |          |
 actual_elapsed_time  | numeric(6,2)          |           |          |
 air_time             | numeric(6,2)          |           |          |
 flights              | numeric(6,2)          |           |          |
 distance             | numeric(6,2)          |           |          |
 distance_group       | numeric               |           |          |
 total_add_gtime      | numeric               |           |          |
Indexes:
    "oco_market_desc_fl_date" btree (origin_city_market_id, fl_date)
```

```
                                                                    QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------
Nested Loop  (cost=3239.51..3435.51 rows=1 width=263) (actual time=2.879..36.186 rows=62 loops=1)
  Buffers: shared hit=1236
  ->  Nested Loop  (cost=3239.23..3435.21 rows=1 width=250) (actual time=2.861..35.865 rows=62 loops=1)
        Join Filter: ((o.unique_carrier)::bpchar = uc.code)
        Rows Removed by Join Filter: 100378
        Buffers: shared hit=1050
        ->  Hash Join  (cost=3239.23..3385.76 rows=1 width=233) (actual time=2.518..4.186 rows=62 loops=1)
              Hash Cond: (ad.code = o.dest_airport_id)
              Buffers: shared hit=244
              ->  Seq Scan on l_airport_id ad  (cost=0.00..122.38 rows=6438 width=41) (actual time=0.014..0.667 rows=6438 loops=1)
                    Buffers: shared hit=58
              ->  Hash  (cost=3239.22..3239.22 rows=1 width=200) (actual time=2.470..2.473 rows=62 loops=1)
                    Buckets: 1024  Batches: 1  Memory Usage: 17kB
                    Buffers: shared hit=186
                    ->  Hash Join  (cost=3092.69..3239.22 rows=1 width=200) (actual time=1.587..2.443 rows=62 loops=1)
                          Hash Cond: (ao.code = o.origin_airport_id)
                          Buffers: shared hit=186
                          ->  Seq Scan on l_airport_id ao  (cost=0.00..122.38 rows=6438 width=41) (actual time=0.006..0.648 rows=6438 loops=1)
                                Buffers: shared hit=58
                          ->  Hash  (cost=3092.67..3092.67 rows=1 width=167) (actual time=0.799..0.802 rows=62 loops=1)
                                Buckets: 1024  Batches: 1  Memory Usage: 14kB
                                Buffers: shared hit=128
                                ->  Nested Loop  (cost=15.04..3092.67 rows=1 width=167) (actual time=0.133..0.765 rows=62 loops=1)
                                      Join Filter: (o.day_of_week = w.code)
                                      Rows Removed by Join Filter: 358
                                      Buffers: shared hit=128
                                      ->  Nested Loop  (cost=15.04..3074.12 rows=40 width=53) (actual time=0.107..0.531 rows=420 loops=1)
                                            Buffers: shared hit=127
                                            ->  Index Scan using co_market_desc on l_city_market_id co  (cost=0.28..8.30 rows=1 width=21) (actual time=0.039..0.040 rows=1 loops=1)
                                                  Index Cond: ((description)::text = 'Little Rock, AR'::text)
                                                  Buffers: shared hit=3
                                            ->  Bitmap Heap Scan on t_ontime o  (cost=14.76..3057.72 rows=810 width=40) (actual time=0.060..0.304 rows=420 loops=1)
                                                  Recheck Cond: ((origin_city_market_id = co.code) AND (fl_date >= '2015-02-01'::date) AND (fl_date <= '2015-02-16'::date))
                                                  Heap Blocks: exact=120
                                                  Buffers: shared hit=124
                                                  ->  Bitmap Index Scan on oco_market_desc_fl_date  (cost=0.00..14.56 rows=810 width=0) (actual time=0.038..0.038 rows=420 loops=1)
                                                        Index Cond: ((origin_city_market_id = co.code) AND (fl_date >= '2015-02-01'::date) AND (fl_date <= '2015-02-16'::date))
                                                        Buffers: shared hit=4
                                      ->  Materialize  (cost=0.00..16.77 rows=3 width=122) (actual time=0.000..0.000 rows=1 loops=420)
                                            Buffers: shared hit=1
                                            ->  Seq Scan on l_weekdays w  (cost=0.00..16.75 rows=3 width=122) (actual time=0.011..0.012 rows=1 loops=1)
                                                  Filter: ((description)::text = 'Friday'::text)
                                                  Rows Removed by Filter: 7
                                                  Buffers: shared hit=1
        ->  Seq Scan on l_unique_carriers uc  (cost=0.00..29.20 rows=1620 width=31) (actual time=0.002..0.195 rows=1620 loops=62)
              Buffers: shared hit=806
  ->  Index Scan using pk_city_market on l_city_market_id cd  (cost=0.28..0.30 rows=1 width=21) (actual time=0.004..0.004 rows=1 loops=62)
        Index Cond: (code = o.dest_city_market_id)
        Buffers: shared hit=186
Planning:
  Buffers: shared hit=28
Planning Time: 4.669 ms
Execution Time: 36.378 ms
```

41.047 ms

Previous>

326.229 ms

2.07 S

Engineer out the Stupid

Quest

- No Primary or Foreign Keys! (See appendix for more Stupid Things)

```
dot-# \di+ public.*
                                                    List of relations
 Schema |          Name           | Type  |  Owner   |       Table       | Persistence | Access method |  Size
--------+-------------------------+-------+----------+-------------------+-------------+---------------+--------
 public | oco_market_desc_fl_date | index | postgres | t_ontime          | permanent   | btree         | 47 MB
 public | pk_city_market          | index | postgres | l_city_market_id  | permanent   | btree         | 144 kB
 public | pk_l_airport            | index | postgres | l_airport_id      | permanent   | btree         | 160 kB
 public | pk_l_unique_carriers    | index | postgres | l_unique_carriers | permanent   | btree         | 72 kB
 public | pk_weekdays             | index | postgres | l_weekdays        | permanent   | btree         | 16 kB
(5 rows)
```

Add PKs & FKs

```
                    Table "public.t_ontime"
      Column        |         Type          | Collation | Nullable | Default
--------------------+-----------------------+-----------+----------+---------
 year               | integer               |           |          |
 quarter            | integer               |           |          |
 month              | integer               |           |          |
 day_of_month       | integer               |           |          |
...

Indexes:
    "oco_market_desc_fl_date" btree (origin_city_market_id, fl_date)
Foreign-key constraints:
    "fk_airline_id" FOREIGN KEY (origin_airport_id) REFERENCES l_airport_id(code)
    "fk_dest_airline_id" FOREIGN KEY (dest_airport_id) REFERENCES l_airport_id(code)
    "fk_dest_city" FOREIGN KEY (dest_city_market_id) REFERENCES l_city_market_id(code)
    "fk_origin_city" FOREIGN KEY (origin_city_market_id) REFERENCES l_city_market_id(code)
    "fk_unique_carrier" FOREIGN KEY (unique_carrier) REFERENCES l_unique_carriers(code)
```

```
                                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
Nested Loop  (cost=15.89..3184.20 rows=5 width=263) (actual time=0.359..1.710 rows=62 loops=1)
  Buffers: shared hit=907
  ->  Nested Loop  (cost=15.60..3182.69 rows=5 width=250) (actual time=0.352..1.589 rows=62 loops=1)
        Buffers: shared hit=721
        ->  Nested Loop  (cost=15.32..3181.19 rows=5 width=217) (actual time=0.344..1.466 rows=62 loops=1)
              Buffers: shared hit=535
              ->  Nested Loop  (cost=15.04..3179.68 rows=5 width=184) (actual time=0.333..1.350 rows=62 loops=1)
                    Buffers: shared hit=349
                    ->  Nested Loop  (cost=14.76..3178.20 rows=5 width=167) (actual time=0.312..0.989 rows=62 loops=1)
                          Join Filter: (w.code = o.day_of_week)
                          Rows Removed by Join Filter: 358
                          Buffers: shared hit=163
                          ->  Seq Scan on l_weekdays w  (cost=0.00..1.10 rows=1 width=122) (actual time=0.016..0.017 rows=1 loops=1)
                                Filter: ((description)::text = 'Friday'::text)
                                Rows Removed by Filter: 7
                                Buffers: shared hit=1
                          ->  Nested Loop  (cost=14.76..3176.60 rows=40 width=53) (actual time=0.291..0.937 rows=420 loops=1)
                                Buffers: shared hit=162
                                ->  Seq Scan on l_city_market_id co  (cost=0.00..110.79 rows=1 width=21) (actual time=0.237..0.617 rows=1 loops=1)
                                      Filter: ((description)::text = 'Little Rock, AR'::text)
                                      Rows Removed by Filter: 5822
                                      Buffers: shared hit=38
                                ->  Bitmap Heap Scan on t_ontime o  (cost=14.76..3057.72 rows=810 width=40) (actual time=0.049..0.210 rows=420 loops=1)
                                      Recheck Cond: ((origin_city_market_id = co.code)
                                            AND (fl_date >= '2015-02-01'::date) AND (fl_date <= '2015-02-16'::date))
                                      Heap Blocks: exact=120
                                      Buffers: shared hit=124
                                      ->  Bitmap Index Scan on oco_market_desc_fl_date  (cost=0.00..14.56 rows=810 width=0) (actual time=0.030..0.030 rows=420 loops=1)
                                            Index Cond: ((origin_city_market_id = co.code)
                                                  AND (fl_date >= '2015-02-01'::date) AND (fl_date <= '2015-02-16'::date))
                                            Buffers: shared hit=4
                    ->  Index Scan using pk_l_unique_carriers on l_unique_carriers uc  (cost=0.28..0.30 rows=1 width=31) (actual time=0.005..0.005 rows=1 loops=62)
                          Index Cond: (code = (o.unique_carrier)::bpchar)
                          Buffers: shared hit=186
              ->  Index Scan using pk_l_airport on l_airport_id ao  (cost=0.28..0.30 rows=1 width=41) (actual time=0.001..0.001 rows=1 loops=62)
                    Index Cond: (code = o.origin_airport_id)
                    Buffers: shared hit=186
        ->  Index Scan using pk_l_airport on l_airport_id ad  (cost=0.28..0.30 rows=1 width=41) (actual time=0.002..0.002 rows=1 loops=62)
              Index Cond: (code = o.dest_airport_id)
              Buffers: shared hit=186
  ->  Index Scan using pk_city_market on l_city_market_id cd  (cost=0.28..0.30 rows=1 width=21) (actual time=0.001..0.001 rows=1 loops=62)
        Index Cond: (code = o.dest_city_market_id)
        Buffers: shared hit=186
Planning:
  Buffers: shared hit=8
Planning Time: 2.462 ms
Execution Time: 1.818 ms
```

4.28 ms

Previous>

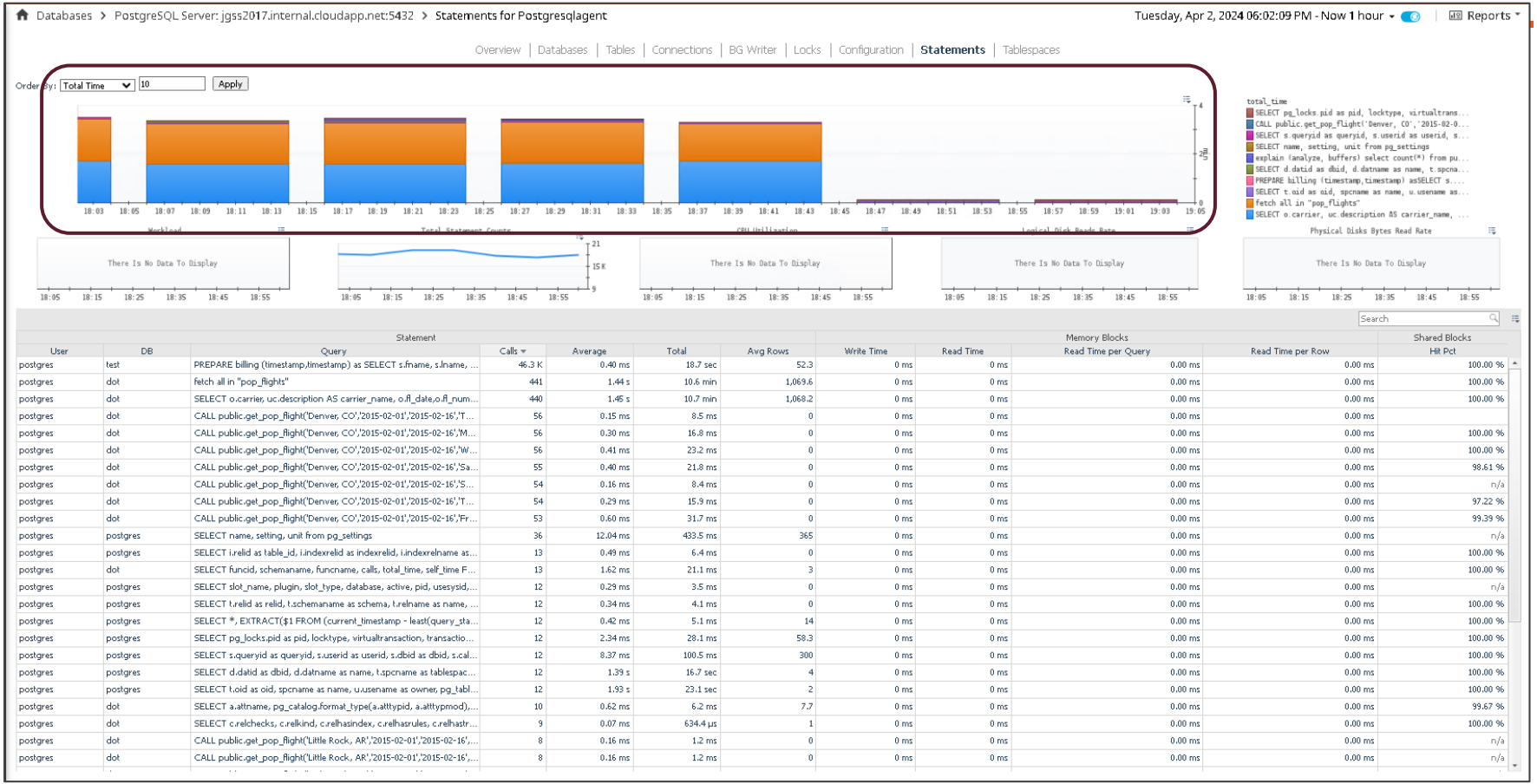
41.047 ms

326.229 ms

2.07 S

Best Average Time

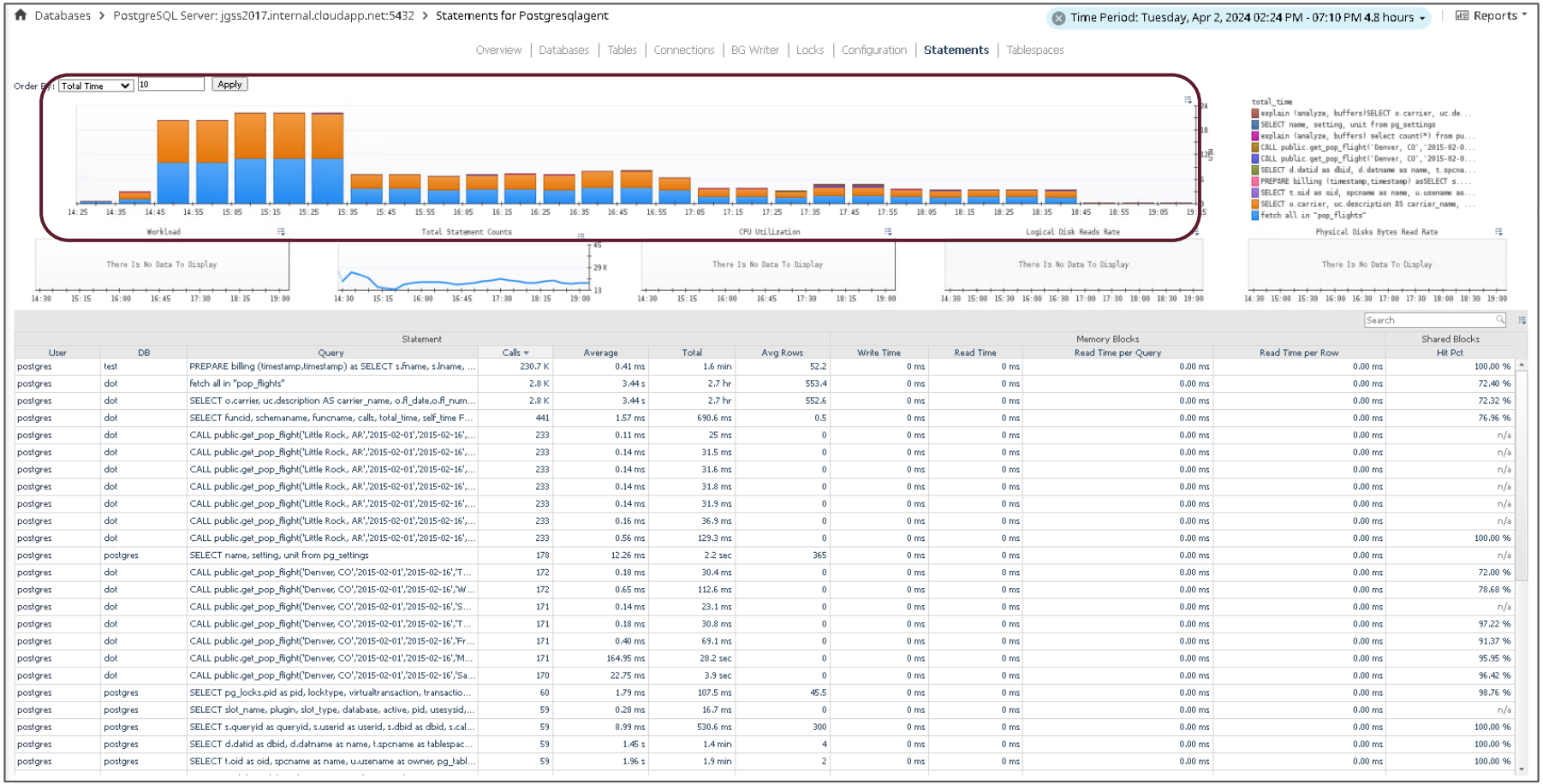
Quest



Statement							Memory Blocks				Shared Blocks
User	DB	Query	Calls	Average	Total	Avg Rows	Write Time	Read Time	Read Time per Query	Read Time per Row	Hit Pct
postgres	test	PREPARE billing (timestamp,timestamp) as SELECT s.fname, s.lname, ...	46.3 K	0.40 ms	18.7 sec	52.3	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
postgres	dot	fetch all in "pop_flights"	441	1.44 s	10.6 min	1,069.6	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
postgres	dot	SELECT o.carrier, uc.description AS carrier_name, o.fl_date,o.fl_num...	440	1.45 s	10.7 min	1,068.2	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
postgres	dot	CALL public.get_pop_flight('Denver, CO','2015-02-01','2015-02-16','T...	56	0.15 ms	8.5 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	
postgres	dot	CALL public.get_pop_flight('Denver, CO','2015-02-01','2015-02-16','M...	56	0.30 ms	16.8 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
postgres	dot	CALL public.get_pop_flight('Denver, CO','2015-02-01','2015-02-16','W...	56	0.41 ms	23.2 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
postgres	dot	CALL public.get_pop_flight('Denver, CO','2015-02-01','2015-02-16','Sa...	55	0.40 ms	21.8 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	98.61 %
postgres	dot	CALL public.get_pop_flight('Denver, CO','2015-02-01','2015-02-16','S...	54	0.16 ms	8.4 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	n/a
postgres	dot	CALL public.get_pop_flight('Denver, CO','2015-02-01','2015-02-16','T...	54	0.29 ms	15.9 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	97.22 %
postgres	dot	CALL public.get_pop_flight('Denver, CO','2015-02-01','2015-02-16','Fr...	53	0.60 ms	31.7 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	99.39 %
postgres	postgres	SELECT name, setting, unit from pg_settings	36	12.04 ms	433.5 ms	365	0 ms	0 ms	0.00 ms	0.00 ms	n/a
postgres	postgres	SELECT i.relid as table_id, i.indexrelid as indexrelid, i.indexrelname as...	13	0.49 ms	6.4 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
postgres	dot	SELECT funcid, schemaname, funcname, calls, total_time, self_time F...	13	1.62 ms	21.1 ms	3	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
postgres	postgres	SELECT slot_name, plugin, slot_type, database, active, pid, usesysid,...	12	0.29 ms	3.5 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	n/a
postgres	postgres	SELECT t.relid as relid, t.schemaname as schema, t.relname as name, ...	12	0.34 ms	4.1 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
postgres	postgres	SELECT *, EXTRACT($1 FROM (current_timestamp - least(query_sta...	12	0.42 ms	5.1 ms	14	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
postgres	postgres	SELECT pg_locks.pid as pid, locktype, virtualtransaction, transactio...	12	2.34 ms	28.1 ms	58.3	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
postgres	postgres	SELECT s.queryid as queryid, s.userid as userid, s.dbid as dbid, s.cal...	12	8.37 ms	100.5 ms	300	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
postgres	postgres	SELECT d.datid as dbid, d.datname as name, t.spcname as tablespac...	12	1.39 s	16.7 sec	4	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
postgres	postgres	SELECT t.oid as oid, spcname as name, u.usename as owner, pg_tabl...	12	1.93 s	23.1 sec	2	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
postgres	dot	SELECT a.attname, pg_catalog.format_type(a.atttypid, a.atttypmod),...	10	0.62 ms	6.2 ms	7.7	0 ms	0 ms	0.00 ms	0.00 ms	99.67 %
postgres	dot	SELECT c.relchecks, c.relkind, c.relhasindex, c.relhasrules, c.relhastr...	9	0.07 ms	634.4 µs	1	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
postgres	dot	CALL public.get_pop_flight('Little Rock, AR','2015-02-01','2015-02-16',...	8	0.16 ms	1.2 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	n/a
postgres	dot	CALL public.get_pop_flight('Little Rock, AR','2015-02-01','2015-02-16',...	8	0.16 ms	1.2 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	n/a

Entire Tuning Effects on Workload

Quest



Statement							Memory Blocks				Shared Blocks
User	DB	Query	Calls	Average	Total	Avg Rows	Write Time	Read Time	Read Time per Query	Read Time per Row	Hit Pct
postgres	test	PREPARE billing (timestamp,timestamp) as SELECT s.fname, s.lname, ...	230.7 K	0.41 ms	1.6 min	52.2	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
postgres	dot	fetch all in "pop_flights"	2.8 K	3.44 s	2.7 hr	553.4	0 ms	0 ms	0.00 ms	0.00 ms	72.40 %
postgres	dot	SELECT o.carrier, uc.description AS carrier_name, o.fl_date,o.fl_num...	2.8 K	3.44 s	2.7 hr	552.6	0 ms	0 ms	0.00 ms	0.00 ms	72.32 %
postgres	dot	SELECT funcid, schemaname, funcname, calls, total_time, self_time F...	441	1.57 ms	690.6 ms	0.5	0 ms	0 ms	0.00 ms	0.00 ms	76.96 %
postgres	dot	CALL public.get_pop_flight('Little Rock, AR','2015-02-01','2015-02-16',...	233	0.11 ms	25 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	n/a
postgres	dot	CALL public.get_pop_flight('Little Rock, AR','2015-02-01','2015-02-16',...	233	0.14 ms	31.5 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	n/a
postgres	dot	CALL public.get_pop_flight('Little Rock, AR','2015-02-01','2015-02-16',...	233	0.14 ms	31.6 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	n/a
postgres	dot	CALL public.get_pop_flight('Little Rock, AR','2015-02-01','2015-02-16',...	233	0.14 ms	31.8 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	n/a
postgres	dot	CALL public.get_pop_flight('Little Rock, AR','2015-02-01','2015-02-16',...	233	0.14 ms	31.9 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	n/a
postgres	dot	CALL public.get_pop_flight('Little Rock, AR','2015-02-01','2015-02-16',...	233	0.16 ms	36.9 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	n/a
postgres	dot	CALL public.get_pop_flight('Little Rock, AR','2015-02-01','2015-02-16',...	233	0.56 ms	129.3 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
postgres	postgres	SELECT name, setting, unit from pg_settings	178	12.26 ms	2.2 sec	365	0 ms	0 ms	0.00 ms	0.00 ms	n/a
postgres	dot	CALL public.get_pop_flight('Denver, CO','2015-02-01','2015-02-16','T...	172	0.18 ms	30.4 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	72.00 %
postgres	dot	CALL public.get_pop_flight('Denver, CO','2015-02-01','2015-02-16','W...	172	0.65 ms	112.6 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	78.68 %
postgres	dot	CALL public.get_pop_flight('Denver, CO','2015-02-01','2015-02-16','S...	171	0.14 ms	23.1 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	n/a
postgres	dot	CALL public.get_pop_flight('Denver, CO','2015-02-01','2015-02-16','T...	171	0.18 ms	30.8 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	97.22 %
postgres	dot	CALL public.get_pop_flight('Denver, CO','2015-02-01','2015-02-16','Fr...	171	0.40 ms	69.1 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	91.37 %
postgres	dot	CALL public.get_pop_flight('Denver, CO','2015-02-01','2015-02-16','M...	171	164.95 ms	28.2 sec	0	0 ms	0 ms	0.00 ms	0.00 ms	95.95 %
postgres	dot	CALL public.get_pop_flight('Denver, CO','2015-02-01','2015-02-16','Sa...	170	22.75 ms	3.9 sec	0	0 ms	0 ms	0.00 ms	0.00 ms	96.42 %
postgres	postgres	SELECT pg_locks.pid as pid, locktype, virtualtransaction, transactio...	60	1.79 ms	107.5 ms	45.5	0 ms	0 ms	0.00 ms	0.00 ms	98.76 %
postgres	postgres	SELECT slot_name, plugin, slot_type, database, active, pid, usesysid,...	59	0.28 ms	16.7 ms	0	0 ms	0 ms	0.00 ms	0.00 ms	n/a
postgres	postgres	SELECT s.queryid as queryid, s.userid as userid, s.dbid as dbid, s.cal...	59	8.99 ms	530.6 ms	300	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
postgres	postgres	SELECT d.datid as dbid, d.datname as name, t.spcname as tablespac...	59	1.45 s	1.4 min	4	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %
postgres	postgres	SELECT t.oid as oid, spcname as name, u.usename as owner, pg_tabl...	59	1.96 s	1.9 min	2	0 ms	0 ms	0.00 ms	0.00 ms	100.00 %

Summary

- **Monitor Wait time**
- **Review the Execution Plan**
  - Look for Costly Steps
- **Gather Object Info**
- **Find the Driving Table**
- **Engineer out the Stupid**
  - Common mistakes
- **Compare your Tuning Results**
  - Brag about Yourself … No one else will!

Quest

References & Additional Info

- PostgreSQL Query Optimization - The Ultimate Guide...
- PostgreSQL Cheatsheet
- Dynamic & Collected Statistics Collection Configuration
- Working With Postgres WAL Made Easy 101
- Autovacuum Configuration Information
- Routine Vacuuming & Analyze
- Appendix
  - More Stupid Mistakes
  - More Explain Common Mistakes
  - Examine the Explain Plan of OLAP Queries

Thank YOU!!!

Quest

More Stupid Mistakes

Quest

- **Not using appropriate indexes slows query & increases database workload**
  - Try using SQL Diagraming techniques to find the best index to drive the least amount of data required
  - Too many indexes can increase the time spent on DML operations
  - Using the wrong type of index - e.g. B-tree index for Full-text search queries
- **Data type mismatch**
  - Comparing columns with wrong data types can lead to errors or incorrect results (implicit conversions)
  - Make sure that the data types of the columns being compared or combined in the query are compatible
- **Not utilizing referential integrity**
  - Generic table design (i.e. PK - tag, value)
  - No primary or foreign keys, etc…
- **Common Postgres Beginner Mistakes & Best Practices**
- **Don't Do This! (PostgreSQL Mistakes & How to Avoid Them**

Explain Plan - Look for Common Mistakes

- **Identify Common Mistakes**
  - Using functions on indexed columns
    - In WHERE, ON & HAVING clause
    - Create a Functional Index instead
      - > Create index lower_title_idx on film(lower(title));
  - Nested views
    - One view calling or joining to other views
  - Use of cursors or row by row processing
- **Missing or Poor Indexing**
- **Problems Outside of the Plan**
  - Missing or stale statistics
  - Database misconfiguration
  - No database constraints

Examine the Explain Plan of OLAP Queries

- **Consider using set operations**
  - Instead of NOT EXISTS or NOT IN – **use EXCEPT**
  - Instead of EXISTS or IN – **use INTERSECT**
  - Instead of complex OR - **use UNION**
  - Review join order, try for most selective join first
- **Avoid multiple scans on same table**
  - Sometimes a design issue where generic table is designed
    - PK, tag, value
- **Consider Temp tables, CTEs & Materialized views on OLAP queries**
  - No indexes or statistics on temp tables – be careful of large temp tables
    - Can cause excessive I/O because of writes to disk
  - CTEs can prevent Optimizer from choosing join order
  - *Dombrovskaya, Henrietta; Bailliekova, Anna. PostgreSQL Query Optimization: The Ultimate Guide to Building Efficient Queries (p. 121). Apress. Kindle Edition.*

More Infor on Wait Events

Quest

- [RDS for PostgreSQL wait events]
- [Aurora PostgreSQL wait events]
- [https://www.postgresql.org/docs/current/monitoring-stats.html]
  - Blocking Locks Query

```
select pid,
       usename,
       pg_blocking_pids(pid) as blocked_by,
       query as blocked_query
from pg_stat_activity
where cardinality(pg_blocking_pids(pid)) > 0;
```

```
dot=# \i pg_lock.sql
 pid  | usename  | blocked_by |                          blocked_query
------+----------+------------+-----------------------------------------------------------
 6352 | postgres | {10228}    | update public.t_ontime set day_of_week = 9 where day_of_week =1;
(1 row)
```

  - Waiting to read data from the client (either too much data or client is slow)

```
select datname, pid, usename,  application_name, wait_event,
       wait_event_type, query_start, state_change, state, query
from pg_catalog.pg_stat_activity;
```

```
dot=# \i pg_wait.sql
 datname |  pid  | usename  | application_name |     wait_event      | wait_event_type |         query_start           |        state_change           |       state        |
         query
---------+-------+----------+------------------+---------------------+-----------------+-------------------------------+-------------------------------+--------------------+-------------------------------
---------------------------------------------------------------------
         |  3356 |          |                  | AutoVacuumMain      | Activity        |                               |                               |                    |
         |  2892 | postgres |                  | LogicalLauncherMain | Activity        |                               |                               |                    |
 dot     |  9716 | postgres | psql             | ClientRead          | Client          | 2024-04-15 21:04:34.452726+00 | 2024-04-15 21:04:34.472967+00 | idle in transaction | fetch all in "pop_flights";
```