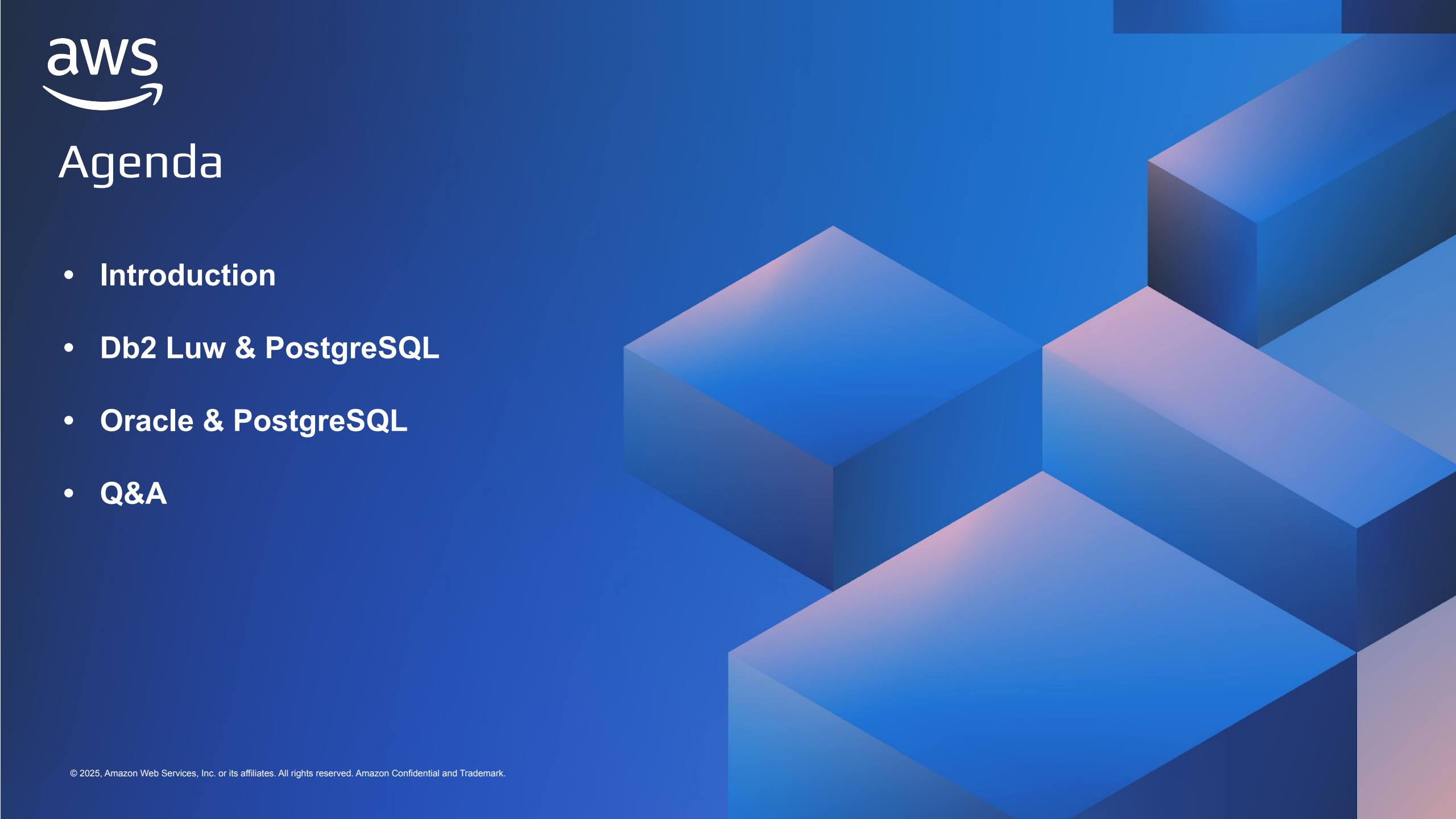
Being Mindful Of Query Optimizer Differences Between PostgreSQL, Oracle and Db2 luw

Puja Audhya & Rakesh Raghav

AWS



Agenda

- **Introduction**
- **Db2 Luw & PostgreSQL**
- **Oracle & PostgreSQL**
- **Q&A**

Introduction

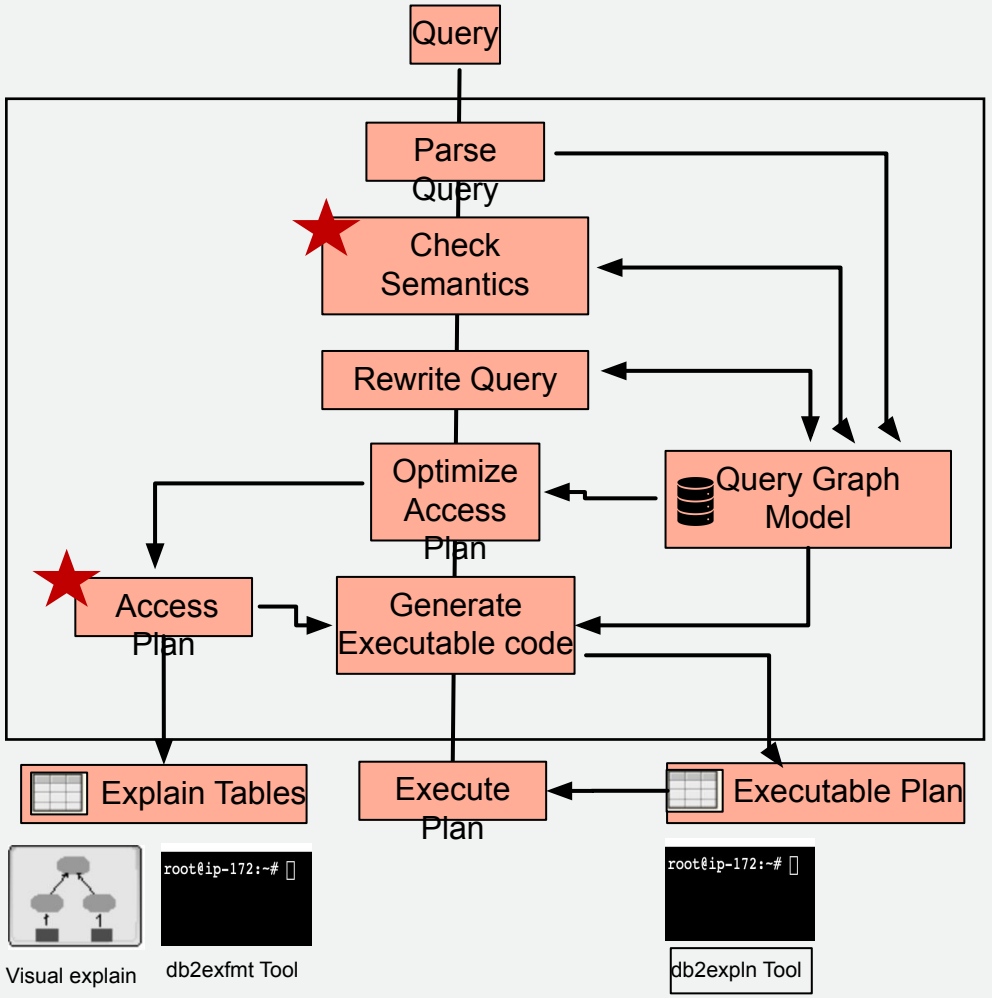
Database Migration
- What
  - From Legacy To Modern Technologies

- Why
  - Application Migration
  - Database Upgrades
  - Data Centre Exit

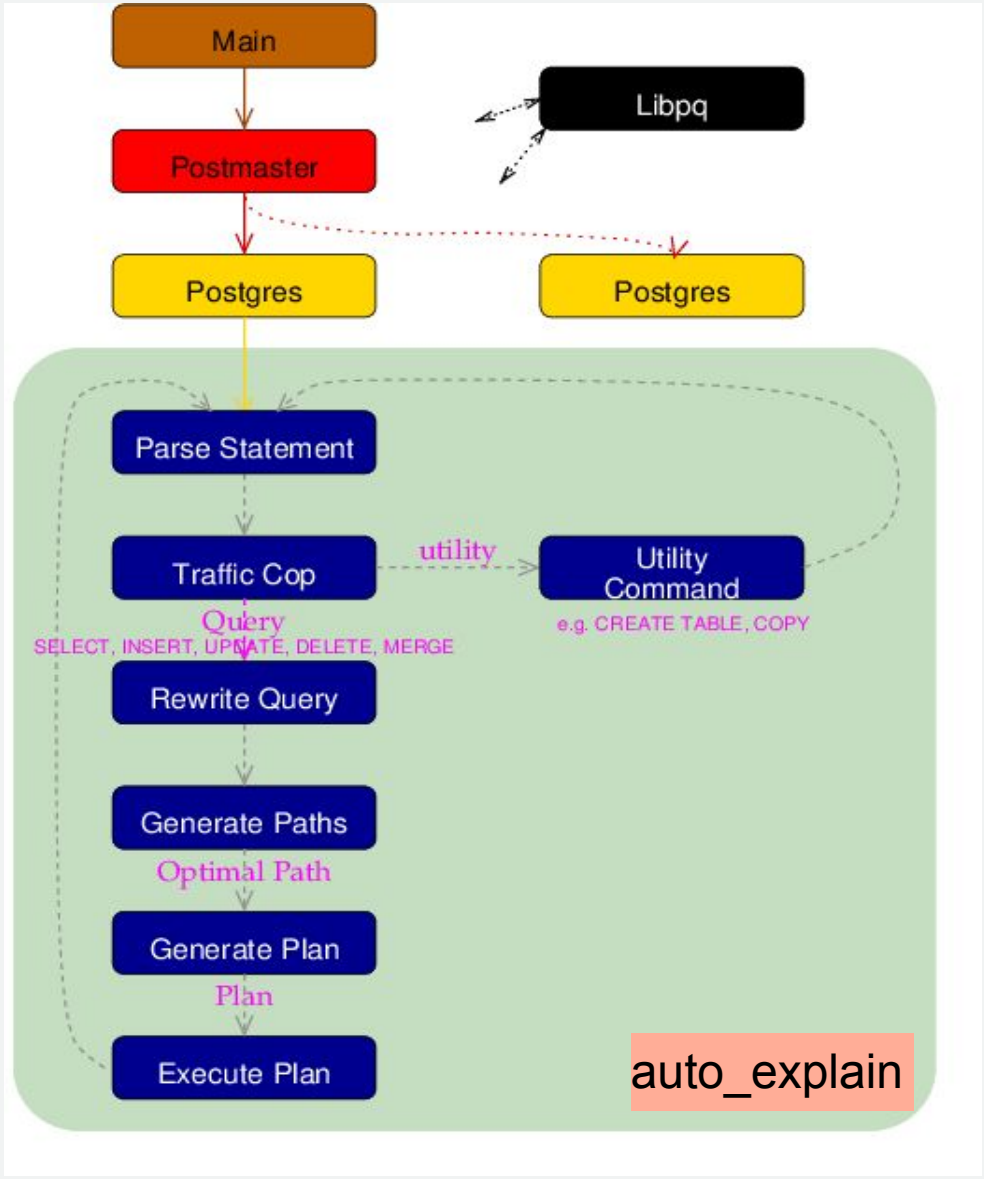
Performance – Key Indicator of Success
- Query Performance (one of the most important factors, yet often left for last)

Optimizer/compiler differences between Db2 LUW & PostgreSQL

Optimizer Comparison



Db2 LUW

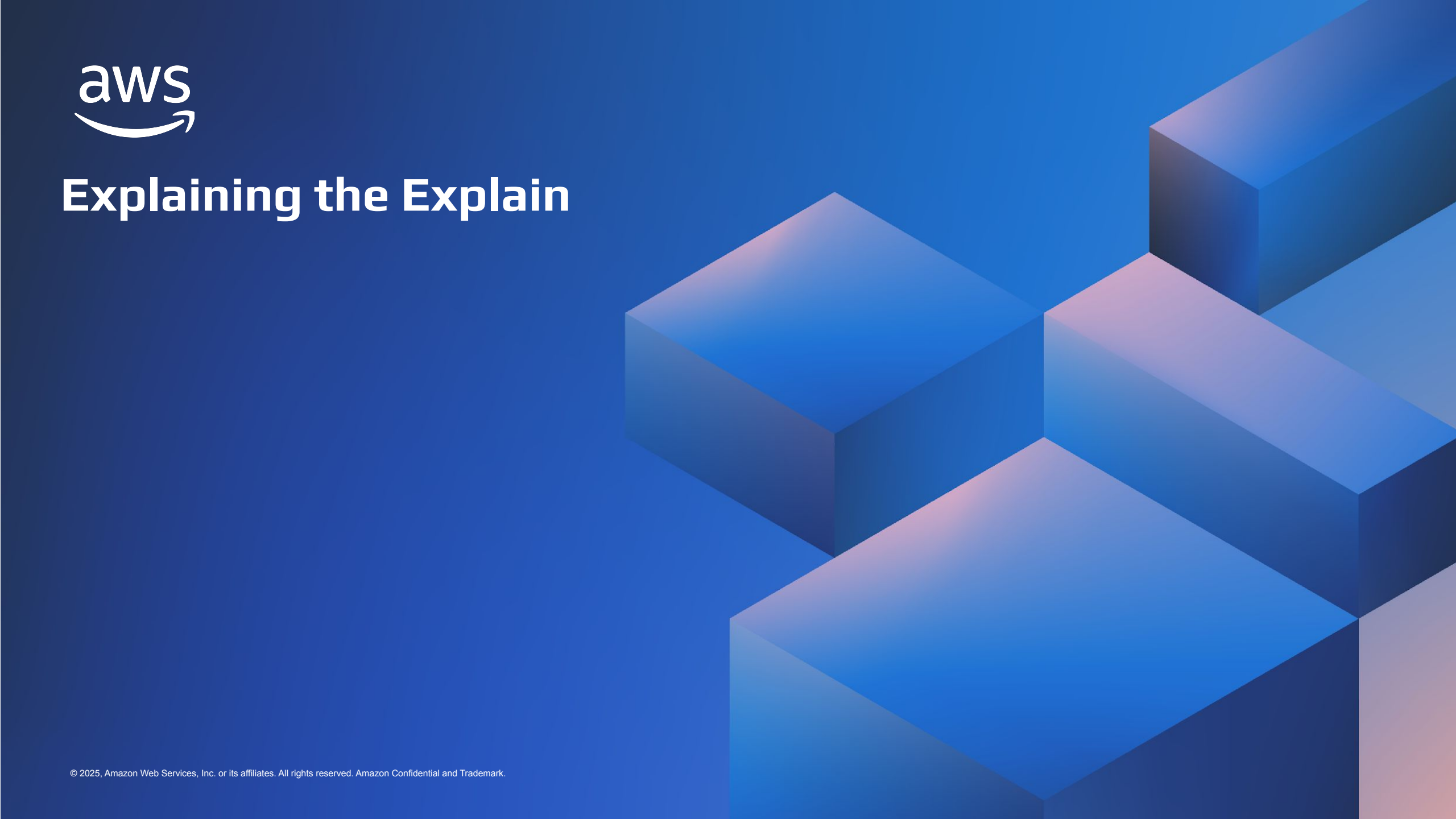


PostgreSQL

Image by BRUCE MOMJIAN, licensed under Creative Commons Attribution (CC BY) [https://momjian.us/main/writings/pgsql/internalpics.pdf#page=11]

Factors Affecting Optimizer Behaviour

Optimization level	Db2 goes from 0 – 9 ; 0 with minimal optimization and 9 being highest PostgreSQL does a near exhaustive optimization/planning until hitting *geqo_threshold* Value { Number of tables joining in a SQL, Default is 12 In practice, *geqo_threshold* is never hit. The defaults of from_collapse_limit and join_collapse_limit being 8 means that geqo_threshold will never be reached .
Statistics	Db2 Rely on *Runstats* command to gather stats and you can create correlated statistics, change default sample size, distribution statistics on all or limited columns PostgreSQL rely on **analyze** command to gather table or column level statistics. *default_statistics_target* control controls sample size for analyze command to gather statistics, Default is 100. CREATE STATISTICS command can be used to create correlation stats between columns



Explaining the Explain

Explain in Db2 LUW and PostgreSQL

```
                                                   Rows
                                                  RETURN
                                                  (   1)
                                                   Cost
                                                    I/O
                                                    |
                                                4.67154e+06
                                                  ^NLJOIN
                                                  (   2)
                                                  47538.6
                                                  29129.3
                          /-----------------------+-----------------------\
                    4.67154e+06                                              1
                      ^HSJOIN                                              FETCH
                      (   3)                                               (  10)
                      28145.1                                              22.1587
                       29125                                               3.25484
        /------------+------------\                                  /-----+-----\
  4.67154e+06                         1142                           1       7.05528e+06
    TBSCAN                          ^HSJOIN                        IXSCAN   TABLE: DMS_SAMPLE_DW
    (   4)                          (   5)                         (  11)        DIM_PERSON
    27570.5                         170.35                         13.6177          Q1
     29100                            25                              2
      |                       /------------+-----------\              |
  4.67154e+06             1142                          62       7.05528e+06
TABLE: DMS_SAMPLE_DW     ^HSJOIN                      TBSCAN    INDEX: SYSIBM
SPORTING_EVENT_TICKET    (   6)                       (   9)   SQL220915081549640
      Q2                 149.886                      20.4325         Q1
                           22                            3
              /---------+---------\                      |
           1142                    62                    62
          TBSCAN                 TBSCAN     TABLE: DMS_SAMPLE_DW
          (   7)                 (   8)        DIM_SPORT_TEAM
          129.423                20.4325            Q3
            19                      3
            |                       |
           1142                     62
   DP-TABLE: DMS_SAMPLE_DW   TABLE: DMS_SAMPLE_DW
     DIM_SPORTING_EVENT         DIM_SPORT_TEAM
            Q5                      Q4
```

PostgreSQL Explain Output

`explain Analyze`

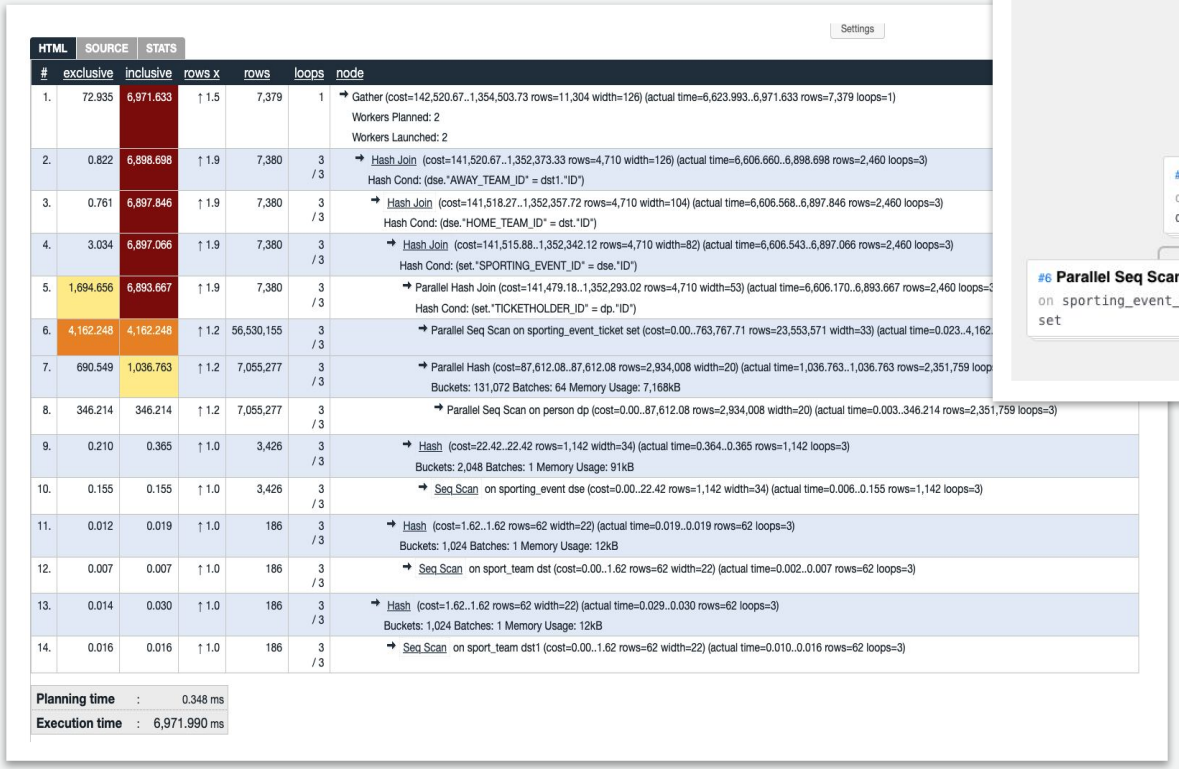
```
                                                                        QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------
Gather  (cost=142520.67..1354503.73 rows=11304 width=126) (actual time=6623.993..6971.633 rows=7379 loops=1)
  Workers Planned: 2
  Workers Launched: 2
  ->  Hash Join  (cost=141520.67..1352373.33 rows=4710 width=126) (actual time=6606.660..6898.698 rows=2460 loops=
        Hash Cond: (dse."AWAY_TEAM_ID" = dst1."ID")
        ->  Hash Join  (cost=141518.27..1352357.72 rows=4710 width=104) (actual time=6606.568..6897.846 rows=2460
              Hash Cond: (dse."HOME_TEAM_ID" = dst."ID")
              ->  Hash Join  (cost=141515.88..1352342.12 rows=4710 width=82) (actual time=6606.543..6897.066 rows=
                    Hash Cond: (set."SPORTING_EVENT_ID" = dse."ID")
                    ->  Parallel Hash Join  (cost=141479.18..1352293.02 rows=4710 width=53) (actual time=6606.170.
                          Hash Cond: (set."TICKETHOLDER_ID" = dp."ID")
                          ->  Parallel Seq Scan on sporting_event_ticket set  (cost=0.00..763767.71 rows=23553571
                          ->  Parallel Hash  (cost=87612.08..87612.08 rows=2934008 width=20) (actual time=1036.76
                                Buckets: 131072  Batches: 64  Memory Usage: 7168kB
                                ->  Parallel Seq Scan on person dp  (cost=0.00..87612.08 rows=2934008 width=20) (
                    ->  Hash  (cost=22.42..22.42 rows=1142 width=34) (actual time=0.364..0.365 rows=1142 loops=3)
                          Buckets: 2048  Batches: 1  Memory Usage: 91kB
                          ->  Seq Scan on sporting_event dse  (cost=0.00..22.42 rows=1142 width=34) (actual time=
              ->  Hash  (cost=1.62..1.62 rows=62 width=22) (actual time=0.019..0.019 rows=62 loops=3)
                    Buckets: 1024  Batches: 1  Memory Usage: 12kB
                    ->  Seq Scan on sport_team dst  (cost=0.00..1.62 rows=62 width=22) (actual time=0.002..0.007 r
        ->  Hash  (cost=1.62..1.62 rows=62 width=22) (actual time=0.029..0.030 rows=62 loops=3)
              Buckets: 1024  Batches: 1  Memory Usage: 12kB
              ->  Seq Scan on sport_team dst1  (cost=0.00..1.62 rows=62 width=22) (actual time=0.010..0.016 rows=
Planning Time: 0.348 ms
Execution Time: 6971.990 ms
(26 rows)

Time: 6973.730 ms (00:06.974)
```

Db2 Formatted explain output

`db2exfmt -d <DB> -1 -o sql.exfmt`

Visual/Formatted Explain in PostgreSQL

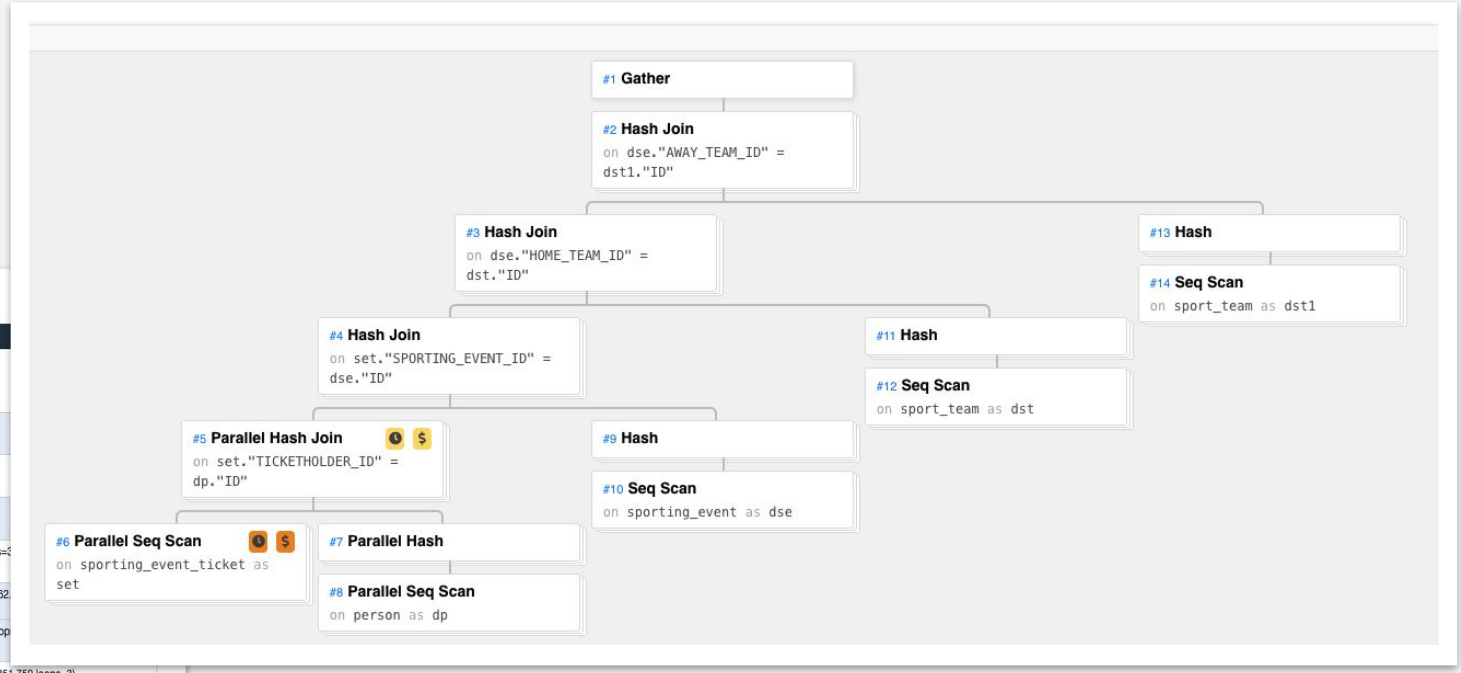
https://explain.depesz.com



HTML | SOURCE | STATS — Settings

#	exclusive	inclusive	rows x	rows	loops	node
1.	72.935	6,971.633	↑ 1.5	7,379	1	→ Gather (cost=142,520.67..1,354,503.73 rows=11,304 width=126) (actual time=6,623.993..6,971.633 rows=7,379 loops=1) Workers Planned: 2 Workers Launched: 2
2.	0.822	6,898.698	↑ 1.9	7,380	3 / 3	→ Hash Join (cost=141,520.67..1,352,373.33 rows=4,710 width=126) (actual time=6,606.660..6,898.698 rows=2,460 loops=3) Hash Cond: (dse."AWAY_TEAM_ID" = dst1."ID")
3.	0.761	6,897.846	↑ 1.9	7,380	3 / 3	→ Hash Join (cost=141,518.27..1,352,357.72 rows=4,710 width=104) (actual time=6,606.568..6,897.846 rows=2,460 loops=3) Hash Cond: (dse."HOME_TEAM_ID" = dst."ID")
4.	3.034	6,897.066	↑ 1.9	7,380	3 / 3	→ Hash Join (cost=141,515.88..1,352,342.12 rows=4,710 width=82) (actual time=6,606.543..6,897.066 rows=2,460 loops=3) Hash Cond: (set."SPORTING_EVENT_ID" = dse."ID")
5.	1,694.656	6,893.667	↑ 1.9	7,380	3 / 3	→ Parallel Hash Join (cost=141,479.18..1,352,293.02 rows=4,710 width=53) (actual time=6,606.170..6,893.667 rows=2,460 loops=3) Hash Cond: (set."TICKETHOLDER_ID" = dp."ID")
6.	4,162.248	4,162.248	↑ 1.2	56,530,155	3 / 3	→ Parallel Seq Scan on sporting_event_ticket set (cost=0.00..763,767.71 rows=23,553,571 width=33) (actual time=0.023..4,162
7.	690.549	1,036.763	↑ 1.2	7,055,277	3 / 3	→ Parallel Hash (cost=87,612.08..87,612.08 rows=2,934,008 width=20) (actual time=1,036.763..1,036.763 rows=2,351,759 loop Buckets: 131,072 Batches: 64 Memory Usage: 7,168kB
8.	346.214	346.214	↑ 1.2	7,055,277	3 / 3	→ Parallel Seq Scan on person dp (cost=0.00..87,612.08 rows=2,934,008 width=20) (actual time=0.003..346.214 rows=2,351,759 loops=3)
9.	0.210	0.365	↑ 1.0	3,426	3 / 3	→ Hash (cost=22.42..22.42 rows=1,142 width=34) (actual time=0.364..0.365 rows=1,142 loops=3) Buckets: 2,048 Batches: 1 Memory Usage: 91kB
10.	0.155	0.155	↑ 1.0	3,426	3 / 3	→ Seq Scan on sporting_event dse (cost=0.00..22.42 rows=1,142 width=34) (actual time=0.006..0.155 rows=1,142 loops=3)
11.	0.012	0.019	↑ 1.0	186	3 / 3	→ Hash (cost=1.62..1.62 rows=62 width=22) (actual time=0.019..0.019 rows=62 loops=3) Buckets: 1,024 Batches: 1 Memory Usage: 12kB
12.	0.007	0.007	↑ 1.0	186	3 / 3	→ Seq Scan on sport_team dst (cost=0.00..1.62 rows=62 width=22) (actual time=0.002..0.007 rows=62 loops=3)
13.	0.014	0.030	↑ 1.0	186	3 / 3	→ Hash (cost=1.62..1.62 rows=62 width=22) (actual time=0.029..0.030 rows=62 loops=3) Buckets: 1,024 Batches: 1 Memory Usage: 12kB
14.	0.016	0.016	↑ 1.0	186	3 / 3	→ Seq Scan on sport_team dst1 (cost=0.00..1.62 rows=62 width=22) (actual time=0.010..0.016 rows=62 loops=3)

Planning time : 0.348 ms
Execution time : 6,971.990 ms



https://explain.dalibo.com/
https://www.pgexplain.dev/

Predicate {Filter Condition} handling

PostgreSQL

```
 Index Scan using sporting_event_ticket_pkey on sporting_event
   Index Cond: ("ID" = '67315111'::numeric)
 Planning Time: 0.076 ms
 Execution Time: 0.026 ms
(4 rows)
```

```
                                                                                QUERY PLAN
 Bitmap Heap Scan on sporting_event_ticket  (cost=699618.01..2048473.94 rows=27127543
   Recheck Cond: (("SEAT_LEVEL" = 1) AND ("TICKET_PRICE" > '20'::numeric))
   Rows Removed by Index Recheck: 3666203
   Heap Blocks: exact=32761 lossy=265137
   ->  Bitmap Index Scan on sporting_event_ticket_idx  (cost=0.00..692836.12 rows=2712
         Index Cond: (("SEAT_LEVEL" = 1) AND ("TICKET_PRICE" > '20'::numeric))
 Planning Time: 0.097 ms
 Execution Time: 30186.913 ms
(8 rows)
```

```
 Seq Scan on sporting_event_ticket  (cost=0.00..1234839
   Filter: ("TICKET_PRICE" > '30'::numeric)
   Rows Removed by Filter: 2411898
 Planning Time: 0.090 ms
 Execution Time: 11797.909 ms
(5 rows)
```

Access Predicate

- The access predicates express the start and stop conditions and is show by Index Cond

Index Filter Predicate

- These do not contribute to the start or stop conditions and do not narrow the scanned range.

Table Level Filter predicate

- Predicates on columns that are not part of the index and is shown as filter

Db2 LUW

Range Delimited

- Provide start and stop key values for the index search

Index Sargable

- cannot limit the scope of a search, but can be evaluated from the index

Data Sargable

- predicates usually require access to individual rows in a table

```
Predicates:
----------
2) Start Key Predicate,
        Comparison Operator:            Equal (=)
        Subquery Input Required:        No
        Filter Factor:                  2.14062e-07

        Predicate Text:
        --------------
        (Q1.ID = 67315111)


2) Stop Key Predicate,
        Comparison Operator:            Equal (=)
        Subquery Input Required:        No
        Filter Factor:                  2.14062e-07

        Predicate Text:
        --------------
        (Q1.ID = 67315111)
```

```
Predicates:
----------
2) Sargable Predicate,
        Comparison Operator:            Less Than (<)
        Subquery Input Required:        No
        Filter Factor:                  0.763559

        Predicate Text:
        --------------
        (20 < Q1.TICKET_PRICE)
```

```
Predicates:
----------
2) Sargable Predicate,
        Comparison Operator:            Not Applicable
        Subquery Input Required:        No
        Filter Factor:                  0.90575

        Predicate Text:
        --------------
        NOT(Q1.SEAT_ROW IN ('A', 'B'))
```

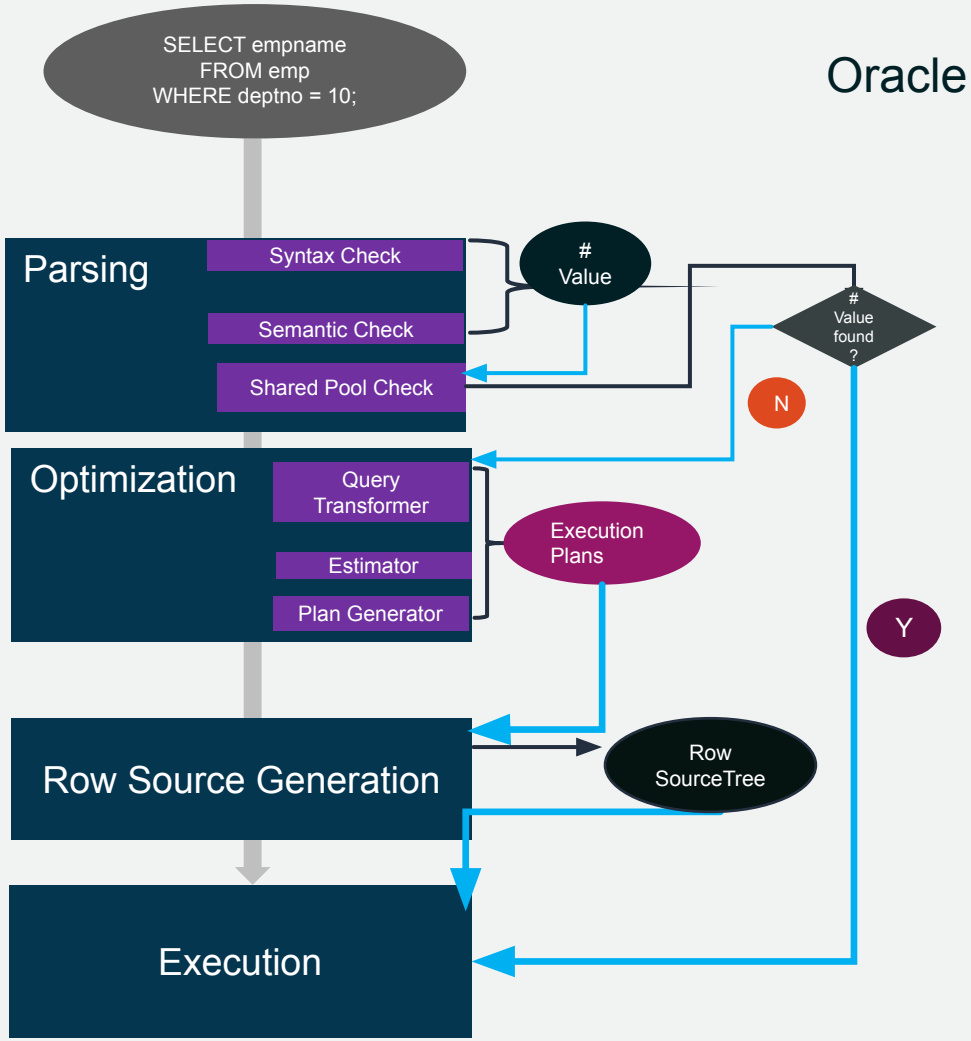
Explain Operator/Nodes

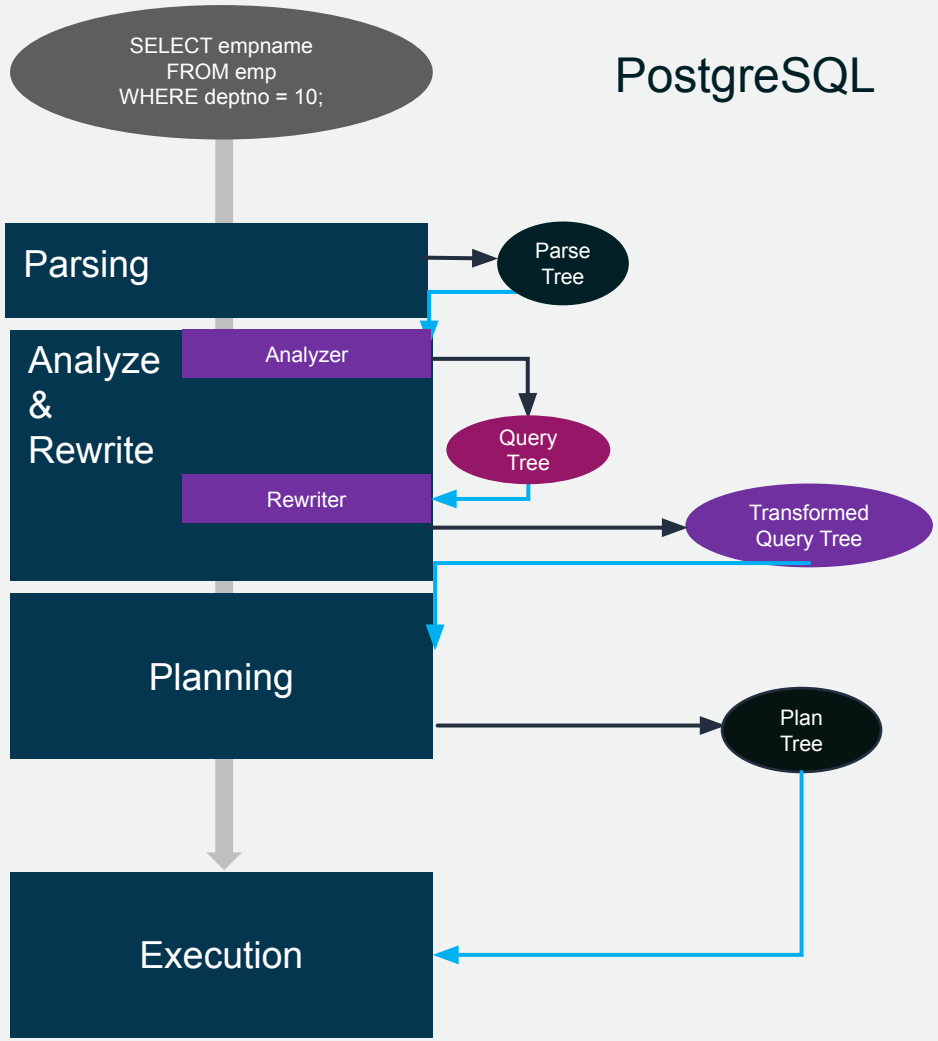
	Db2 LUW	PostgreSQL	Remarks
Table and Index	TBSCAN	Seq Scan	Scan table to get required data
	IXSCAN	Index Scan	Scanning index to get required data
	IXAND	BitmapAnd	Combining multiple index to satisfy filter condition
	XANDOR	BitmapOr	Combining multiple index to satisfy filter condition
Joins	MSJOIN	Merge Join {Merge Cond:}	Merge Join
	NLJOIN	Nested Loop {Join Filter:}	Nested Loop Join
	HSJOIN	Hash Join {Hash Cond:}	Hash Join
Miscellaneous	GRPBY	Group Key:	
	SORT	Sort {Sort Key: unique1} Incremental Sort (Limit SQL}	Order by or group by resulting in sort
	FILTER	Filter	where condition applied

How Optimizer Works

In Oracle
Vs
In PostgreSQL

SQL Processing





Cost of PostgreSQL Query

total_cost = startup cost + run cost

Startup cost = cost incurred before the first row is fetched

Run cost = cost of running all the rows

Unit: Arbitrary
Represent: Estimated Cost
Relevant for comparison within the database

Cost of Oracle Query

total_cost ∝ I/O cost, CPU cost, Communication

Unit: Arbitrary
Represent: Estimated Cost
Relevant for comparison within the database

Optimizer Statistics Collection

Optimizer Statistics Collection

Oracle

Automatic statistics collection
Default table and index statistics collection
Collection during predefined maintenance windows
Database scheduler and automated maintenance tasks
Uses Oracle's data modification monitoring feature
Manages real-time statistics on tables

Manual statistics collection

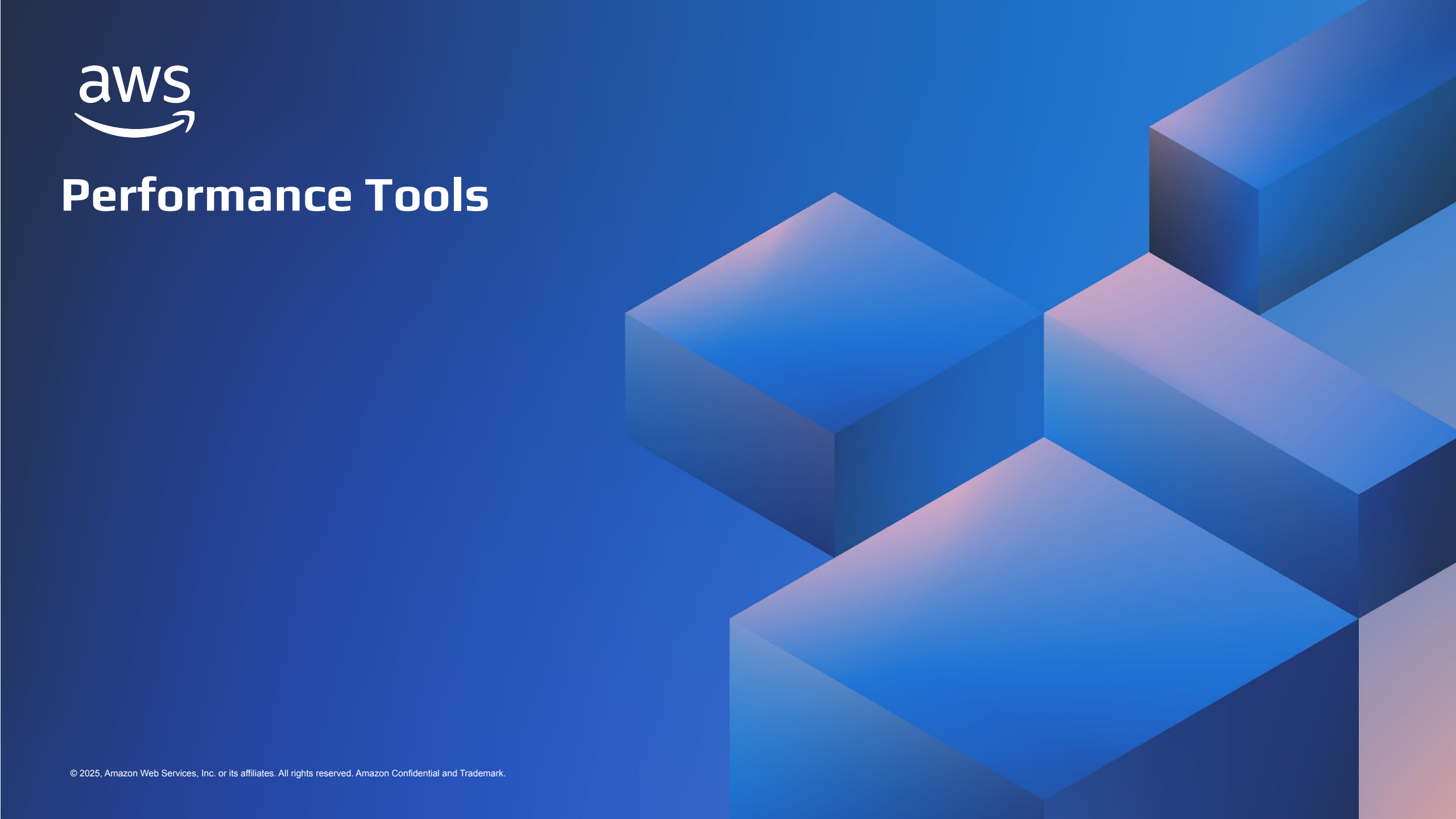
Statistics level
GATHER_INDEX_STATS
GATHER_TABLE_STATS
GATHER_SCHEMA_STATS
GATHER_DICTIONARY_STATS
GATHER_DATABASE_STATS

PostgreSQL

Automatic statistics collection
Default - autovacuum daemon
Parameters to control – storage parameters of tables
(can be set or changed using CREATE TABLE or ALTER TABLE statements)

Manual statistics collection
Performed using - ANALYZE, VACUUM ANALYZE

Statistics level
Database, Table, Column



Performance Tools

Performance Tools

Oracle

SQLT – Reactive in nature
diagnose specific SQL performance issues
Detailed execution plans
Object statistics
Comparative retrospective analysis over time

SPM – Proactive feature
Manage execution plans
Stable performance
Preserve desired execution plan
Protect against performance degradation due to database upgrades, schema changes etc
Captures plan, and pins it
Can evolve over time

PostgreSQL

EXPLAIN/EXPLAIN ANALYZE – check execution plan and runtime statistics for SQL queries

Pg_stat_statements extension – monitor and analyze SQL execution performance across database (for all SQL statements)

pgBadger – Log analyzer to get detailed performance reports and graphs

Postgres Explain Visualizer – Web-based visualizer for EXPLAIN plan.

Auto_Explain – automatically logs execution plans for slow queries

Access Paths

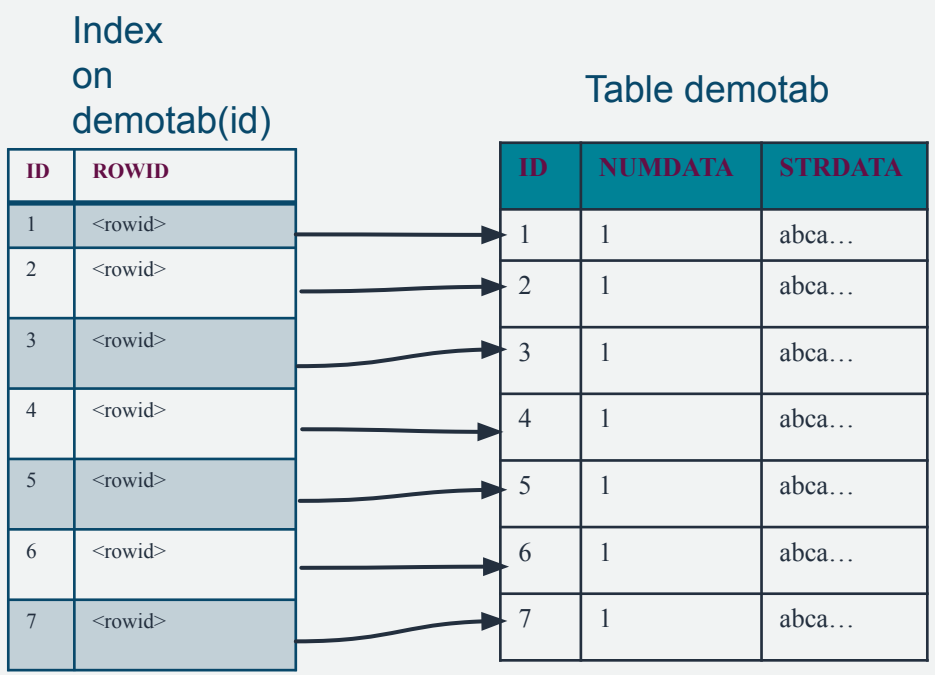
DEMOTAB

Oracle

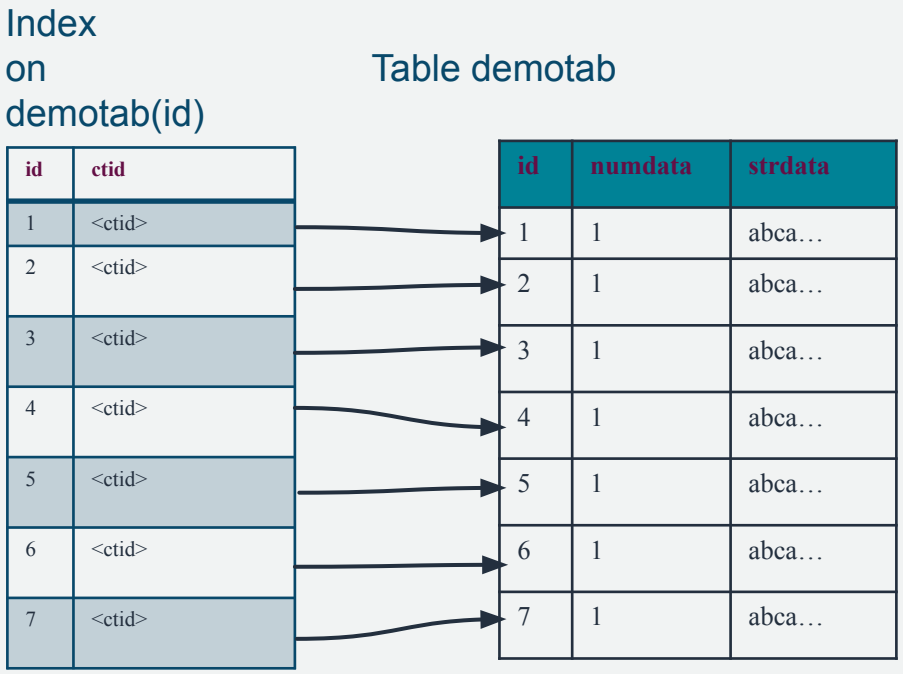
```
COLUMN_NAM DATA_TYPE
---------- ----------
ID         NUMBER
NUMDATA    NUMBER
STRDATA    VARCHAR2

3 rows selected.
```

PostgreSQL

```
 column_name | data_type
-------------+-----------
 id          | integer
 numdata     | integer
 strdata     | text
(3 rows)
```



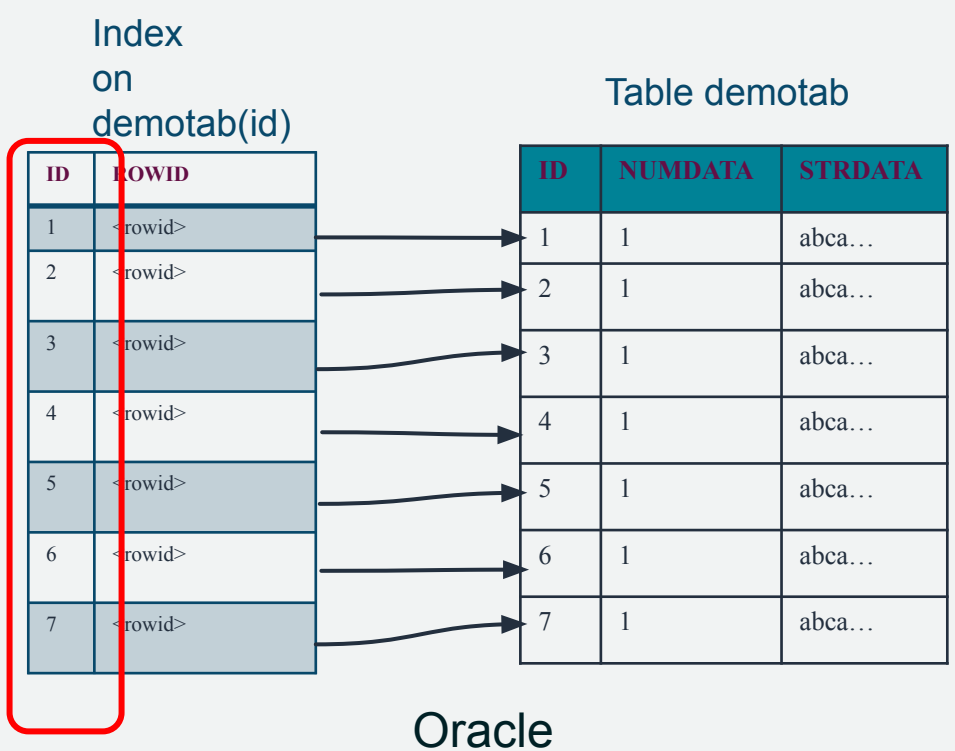
Oracle

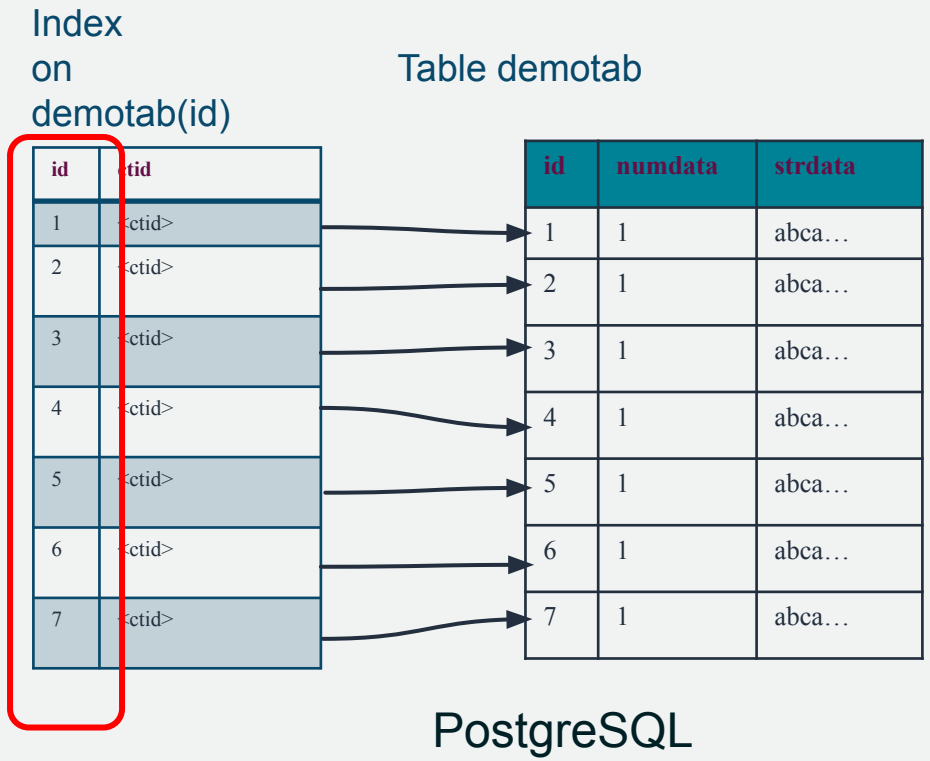


PostgreSQL

Query With Aggregate On Indexed Column

SELECT SUM(ID) from demotab;





Query With Aggregate On Indexed Column

```
SQL_ID  2mp0691ycgcgc, child number 0
-------------------------------------
SELECT SUM(ID) from demotab

Plan hash value: 946847014

-------------------------------------------------------------------------------------
| Id  | Operation             | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |             |       |       |     7 (100)|          |
|   1 |  SORT AGGREGATE       |             |     1 |     4 |            |          |
|   2 |   INDEX FAST FULL SCAN| DEMOIDX_ID  | 10000 | 40000 |     7   (0)| 00:00:01 |
-------------------------------------------------------------------------------------
```

Oracle

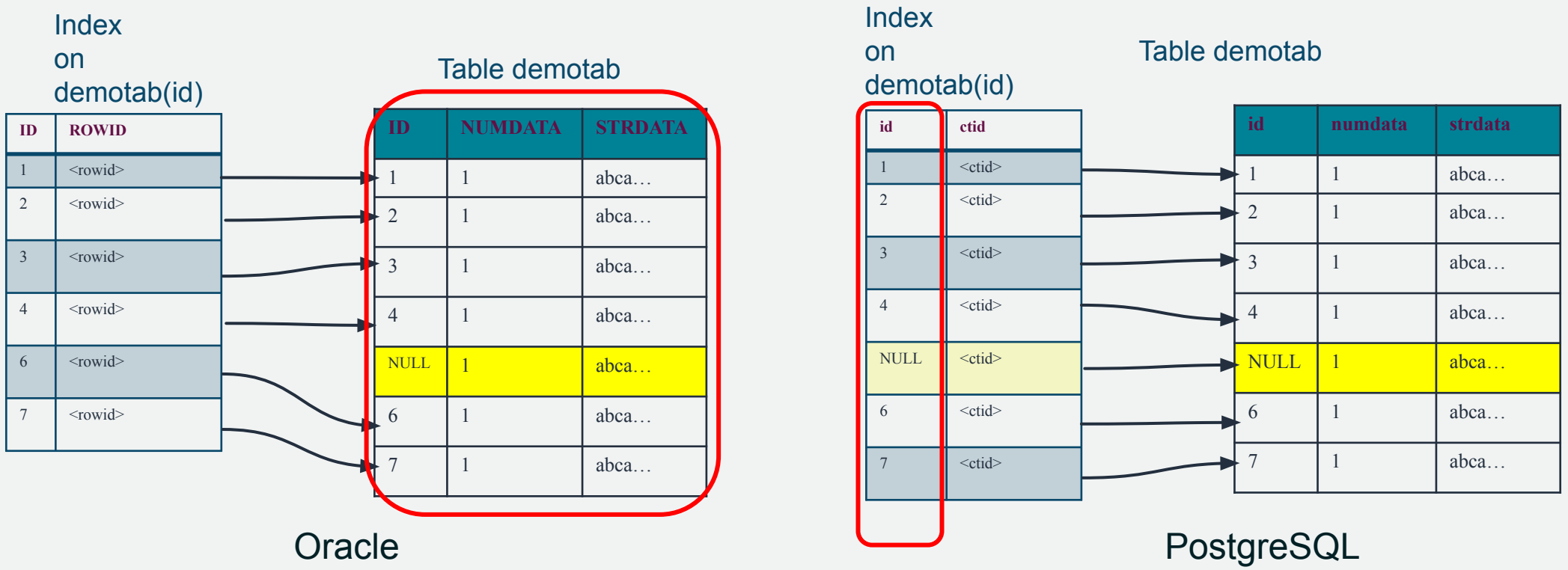
```
pgconpg=> EXPLAIN (analyze,verbose,costs,buffers) SELECT SUM(id) from demotab;
                                                              QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=342.29..342.30 rows=1 width=8) (actual time=1.434..1.435 rows=1 loops=1)
   Output: sum(id)
   Buffers: shared hit=30
   ->  Index Only Scan using demoidx_id on public.demotab  (cost=0.29..317.29 rows=10000 width=4) (actual time=0.041..0.845 rows=10000 loops=1)
         Output: id
         Heap Fetches: 0
         Buffers: shared hit=30
 Query Identifier: -6626188229673023644
 Planning:
   Buffers: shared hit=6
 Planning Time: 0.067 ms
 Execution Time: 1.459 ms
(12 rows)
```

PostgreSQL

aws

Query selecting all rows from indexed column

SELECT id from demotab;



Query selecting all rows from indexed column

SELECT id from demotab;

```
SQL_ID  77zh6jvr2253j, child number 0
-------------------------------------
SELECT id from demotab

Plan hash value: 4082944399

-------------------------------------------------------------------------------
| Id  | Operation          | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |         |       |       |   389 (100)|          |
|   1 |  TABLE ACCESS FULL| DEMOTAB | 10000 | 40000 |   389   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

Oracle

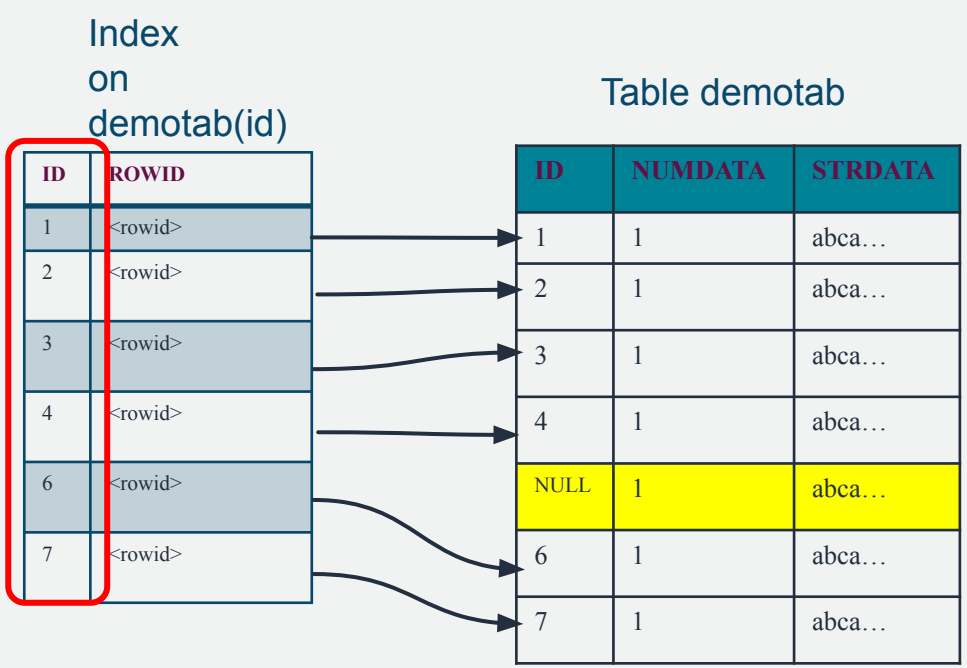
```
pgconpg=> EXPLAIN (analyze,verbose,costs,buffers) SELECT id from demotab;
                                                              QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------
 Index Only Scan using demoidx_id on public.demotab  (cost=0.29..317.29 rows=10000 width=4) (actual time=0.019..0.824 rows=10000 loops=1)
   Output: id
   Heap Fetches: 0
   Buffers: shared hit=30
 Query Identifier: -6445792544559811907
 Planning Time: 0.049 ms
 Execution Time: 1.268 ms
(7 rows)
```

PostgreSQL

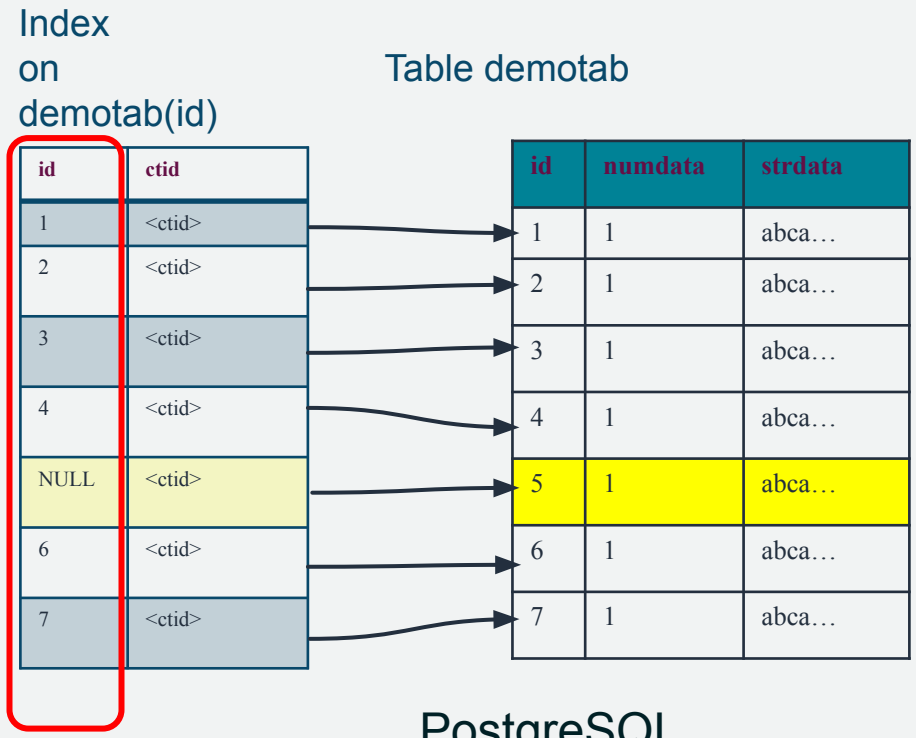
aws

Query selecting all (NOT NULL) rows from indexed column

SELECT id from demotab where id is NOT NULL;



Oracle



PostgreSQL

Query selecting all (NOT NULL) rows from indexed column

SELECT id from demotab where id is NOT NULL;

```
SQL_ID  c4xctawphfsyc, child number 0
-------------------------------------
SELECT id from demotab where id is not null

Plan hash value: 2265200488

-----------------------------------------------------------------------------------------
| Id  | Operation             | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |            |       |       |     7 (100)|          |
|*  1 |  INDEX FAST FULL SCAN| DEMOIDX_ID | 10000 | 40000 |     7   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------
```

Oracle

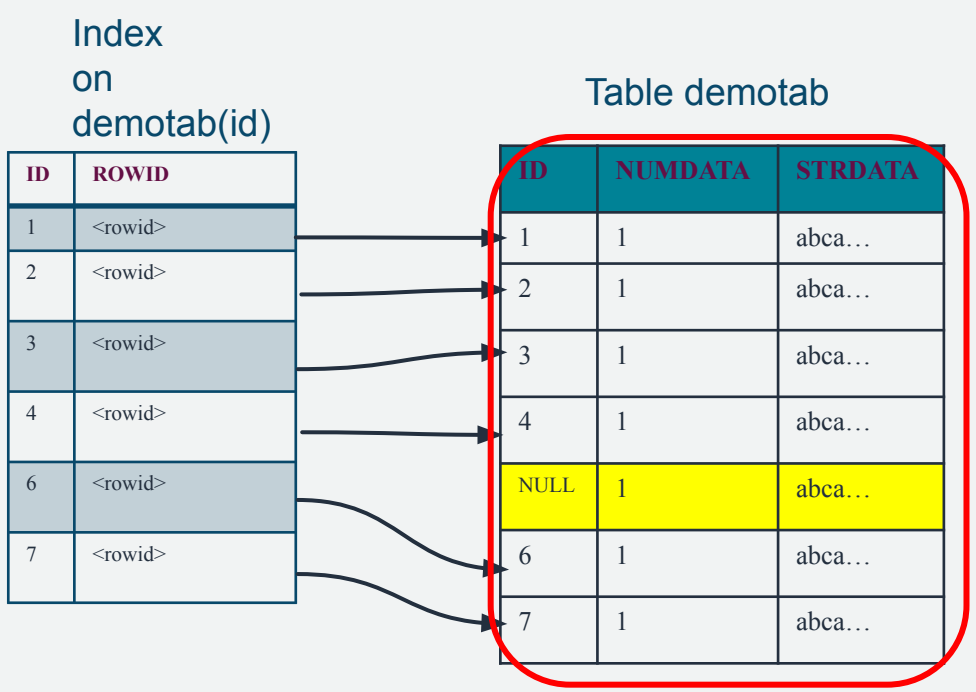
```
pgconpg=> EXPLAIN (analyze,verbose,costs,buffers) SELECT id from demotab where id is not null;
                                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------
 Index Only Scan using demoidx_id on public.demotab  (cost=0.29..342.29 rows=10000 width=4) (actual time=0.011..0.849 rows=10000 loops=1)
   Output: id
   Index Cond: (demotab.id IS NOT NULL)
   Heap Fetches: 0
   Buffers: shared hit=30
 Query Identifier: 8060139790616063777
 Planning Time: 0.055 ms
 Execution Time: 1.319 ms
(8 rows)
```

PostgreSQL

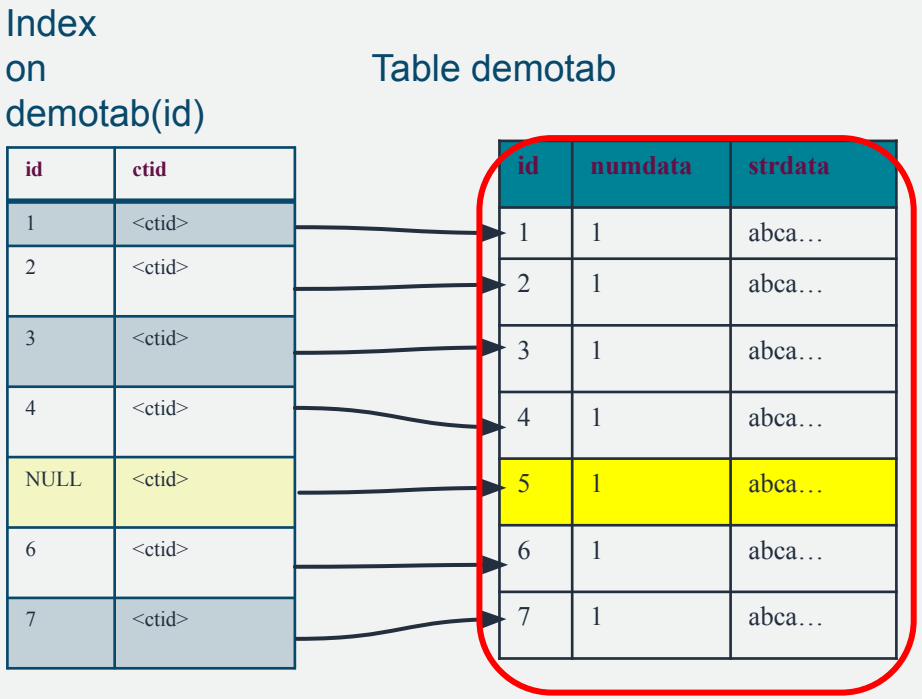
aws

Query selecting all rows from unindexed column(s)

SELECT numdata, strdata from demotab;



Oracle



PostgreSQL

Query selecting all rows from unindexed column(s)

SELECT numdata, strdata from demotab;

```
SQL> SQL_ID     9z0sjqjp0fsxr, child number 0
-------------------------------------
SELECT numdata, strdata from demotab

Plan hash value: 4082944399

-------------------------------------------------------------------------------
| Id  | Operation          | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |          |       |       |   389 (100)|          |
|   1 |  TABLE ACCESS FULL| DEMOTAB  | 10000 |  9804K|   389   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

Oracle

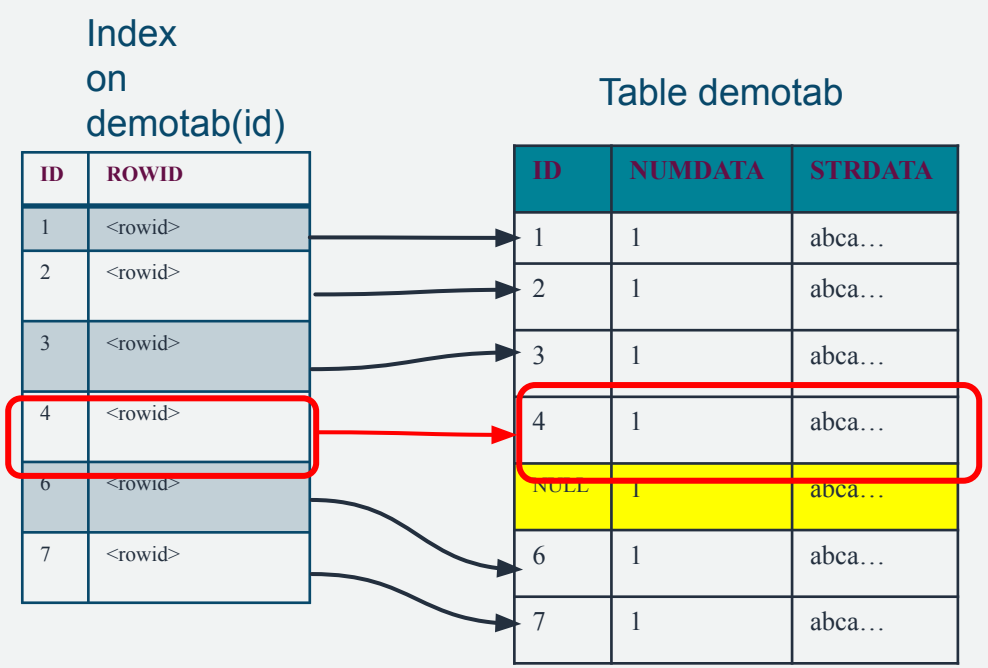
```
pgconpg=> EXPLAIN (analyze,verbose,costs,buffers) SELECT numdata, strdata from demotab;
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Seq Scan on public.demotab  (cost=0.00..1572.00 rows=10000 width=1008) (actual time=0.006..2.000 rows=10000 loops=1)
   Output: numdata, strdata
   Buffers: shared hit=1472
 Query Identifier: 2184092447303702546
 Planning Time: 0.040 ms
 Execution Time: 2.456 ms
(6 rows)
```

PostgreSQL

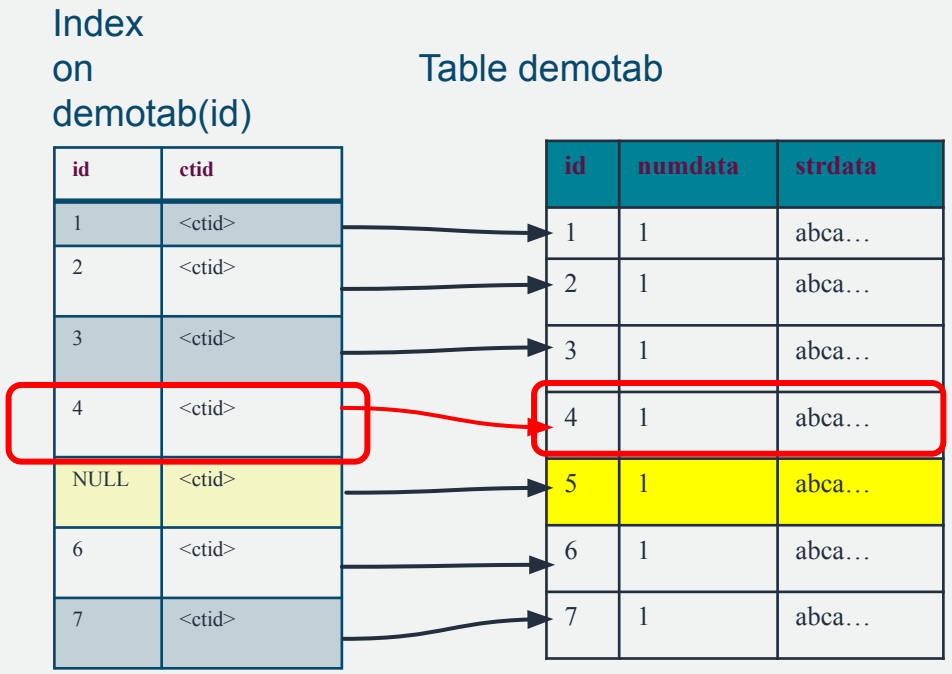
aws

Query selecting rows (Unindexed column selected) from indexed column with conditions

SELECT numdata, strdata from demotab where id=4;



Oracle



PostgreSQL

Query selecting rows (Unindexed column selected) from indexed column with conditions

SELECT numdata, strdata from demotab where id=4;

```
SQL_ID  20yrq196kb76t, child number 0
-------------------------------------
SELECT numdata, strdata from demotab where id=4

Plan hash value: 2494484603

-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |            |       |       |     2 (100)|          |
|   1 |  TABLE ACCESS BY INDEX ROWID| DEMOTAB    |     1 |  1008 |     2   (0)| 00:00:01 |
|*  2 |   INDEX UNIQUE SCAN          | DEMOIDX_ID |     1 |       |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

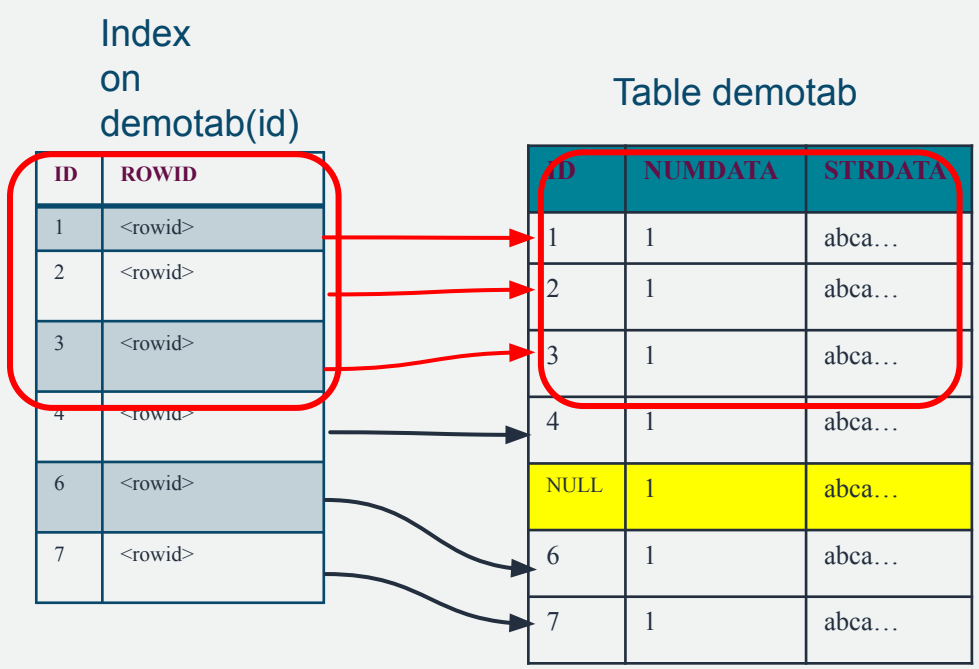
Oracle

```
pgconpg=> EXPLAIN (analyze,verbose,costs,buffers) SELECT numdata, strdata from demotab where id=4;
                                                              QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------
 Index Scan using demoidx_id on public.demotab  (cost=0.29..8.30 rows=1 width=1008) (actual time=0.010..0.010 rows=1 loops=1)
   Output: numdata, strdata
   Index Cond: (demotab.id = 4)
   Buffers: shared hit=3
 Query Identifier: 2476251053117721190
 Planning Time: 0.060 ms
 Execution Time: 0.026 ms
(7 rows)
```

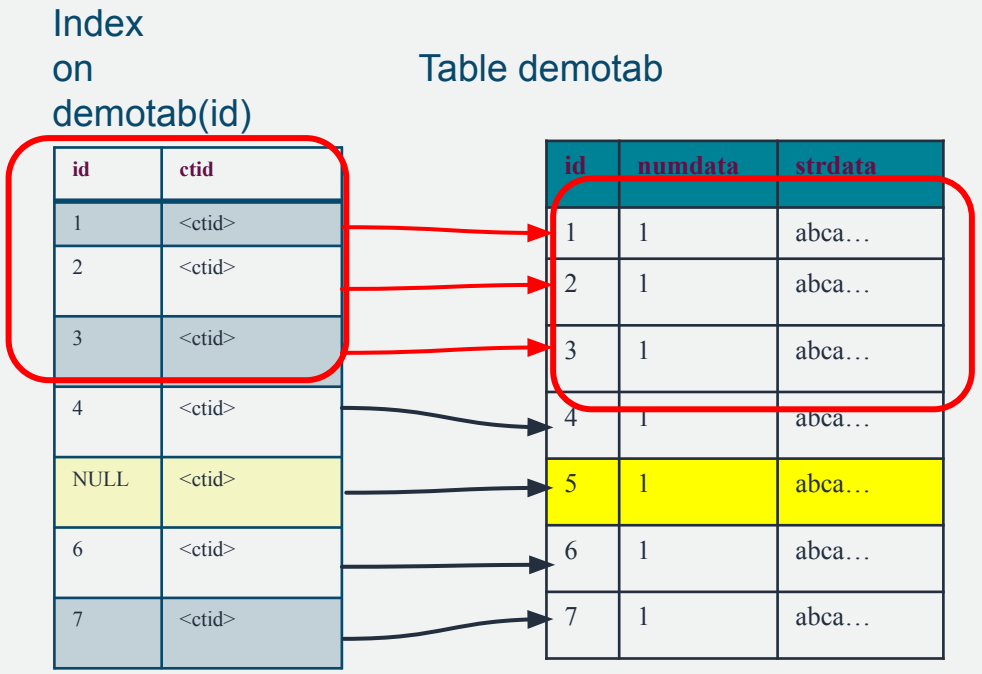
PostgreSQL

Query selecting rows (Unindexed column selected) from indexed column with conditions

SELECT numdata, strdata from demotab where id < 4;



Oracle



PostgreSQL

Query selecting rows (Unindexed column selected) from indexed column with conditions

SELECT numdata, strdata from demotab where id < 4;

```
SQL_ID  6gc7qvd5zjc1c, child number 0
-------------------------------------
SELECT numdata, strdata from demotab where id < 4

Plan hash value: 3993455251

-----------------------------------------------------------------------------------------------
| Id  | Operation                            | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                     |            |       |       |     3 (100)|          |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| DEMOTAB    |     3 |  3024 |     3   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN                   | DEMOIDX_ID |     3 |       |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Oracle

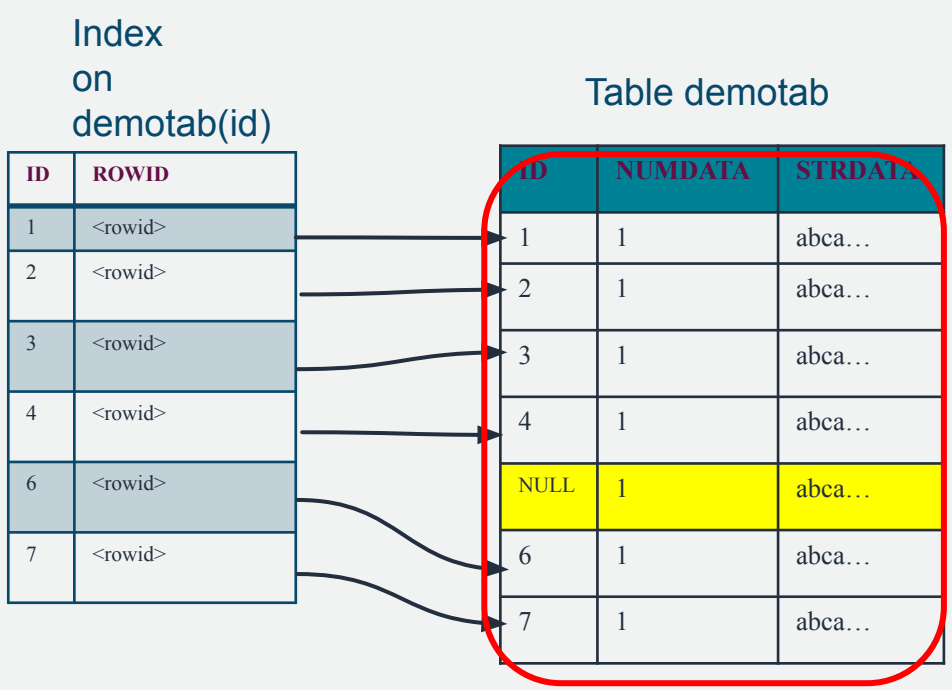
```
pgconpg=> EXPLAIN (analyze,verbose,costs,buffers) SELECT numdata, strdata from demotab where id < 4;
                                                      QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Index Scan using demoidx_id on public.demotab  (cost=0.29..8.34 rows=3 width=1008) (actual time=0.007..0.008 rows=3 loops=1)
   Output: numdata, strdata
   Index Cond: (demotab.id < 4)
   Buffers: shared hit=3
 Query Identifier: 4293266419214812759
 Planning:
   Buffers: shared hit=3
 Planning Time: 0.076 ms
 Execution Time: 0.023 ms
(9 rows)
```

PostgreSQL

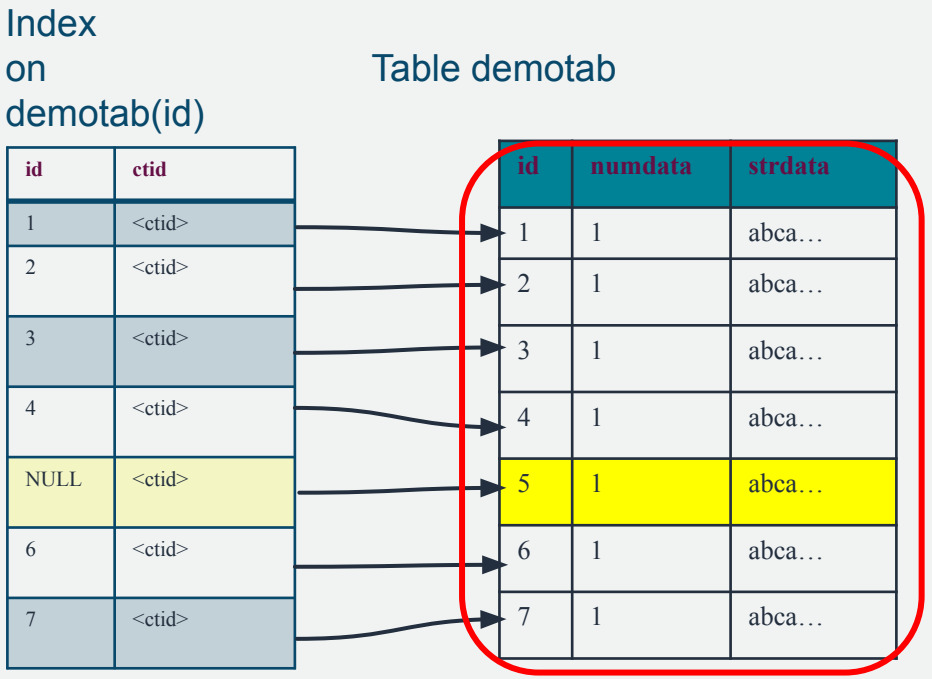
aws

Query selecting rows (Unindexed column selected) from indexed column with conditions

SELECT numdata, strdata from demotab where id < 10000;



Oracle



PostgreSQL

Query selecting rows (Indexed column selected) from indexed column with conditions

SELECT numdata, strdata from demotab where id < 10000;

```
SQL_ID  98xxaun705yts, child number 0
-------------------------------------
SELECT numdata, strdata from demotab where id < 10000

Plan hash value: 4082944399

-------------------------------------------------------------------------------
| Id  | Operation          | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |         |       |       |   389 (100)|          |
|*  1 |  TABLE ACCESS FULL| DEMOTAB |  9999 |  9842K|   389   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

Oracle

```
pgconpg=> EXPLAIN (analyze,verbose,costs,buffers) SELECT numdata, strdata from demotab where id < 10000;
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Seq Scan on public.demotab  (cost=0.00..1597.00 rows=9999 width=1008) (actual time=0.007..2.092 rows=9999 loops=1)
   Output: numdata, strdata
   Filter: (demotab.id < 10000)
   Rows Removed by Filter: 1
   Buffers: shared hit=1472
 Query Identifier: 4293266419214812759
 Planning:
   Buffers: shared hit=3
 Planning Time: 0.071 ms
 Execution Time: 2.542 ms
(10 rows)
```

PostgreSQL

Query Processing: Example

Oracle VS PostgreSQL

Query in Oracle

Tables

PERSON p
SPORTING_EVENT_TICKET t
SEAT s
SPORTING_EVENT e
STADIUM_NOTES sn

Predicate Information

```
t.SPORT_LOCATION_ID = s.SPORT_LOCATION_ID
      AND t.SEAT_LEVEL = s.SEAT_LEVEL
      AND t.SEAT_SECTION = s.SEAT_SECTION
      AND t.SEAT_ROW = s.SEAT_ROW
      AND t.SEAT = s.SEAT
t.SPORTING_EVENT_ID = e.ID
t.TICKETHOLDER_ID IS NOT NULL
p.ID = t.TICKETHOLDER_ID
t.SPORT_LOCATION_ID = sn.ID
```

```
Query Block Name / Object Alias (identified by operation id):
-------------------------------------------------------------

   1 - SEL$34470967
   4 - SEL$34470967 / E@SEL$3
   7 - SEL$34470967 / T@SEL$1
   8 - SEL$34470967 / T@SEL$1
   9 - SEL$34470967 / P@SEL$1
  10 - SEL$34470967 / P@SEL$1
  11 - SEL$34470967 / S@SEL$2
  12 - SEL$34470967 / SN@SEL$4
  13 - SEL$34470967 / SN@SEL$4
```

```
SELECT
        p.ID AS person_id,
        p.FULL_NAME,
        p.FIRST_NAME,
        p.LAST_NAME,
-- Ticket and Seat Details
        t.SEAT_LEVEL,
        t.SEAT_SECTION,
        t.SEAT_ROW,
        t.SEAT,
        t.TICKET_PRICE,
        s.SEAT_TYPE,
-- Event Details
        e.SPORT_TYPE_NAME,
        e.START_DATE_TIME,
-- Stadium Details
        sn.NAME as stadium_name,
        sn.OWNER as stadium_owner,
        sn.TYPE as stadium_type,
        sn.TEXT as stadium_notes
FROM
        PERSON p INNER JOIN
        SPORTING_EVENT_TICKET t ON p.ID = t.TICKETHOLDER_ID INNER
JOIN
        SEAT s ON (t.SPORT_LOCATION_ID = s.SPORT_LOCATION_ID
        AND t.SEAT_LEVEL = s.SEAT_LEVEL
        AND t.SEAT_SECTION = s.SEAT_SECTION
        AND t.SEAT_ROW = s.SEAT_ROW
        AND t.SEAT = s.SEAT) INNER JOIN
        SPORTING_EVENT e ON t.SPORTING_EVENT_ID = e.ID LEFT JOIN
        STADIUM_NOTES sn ON t.SPORT_LOCATION_ID = sn.ID
WHERE
        t.TICKETHOLDER_ID IS NOT NULL;
```

Query In PostgreSQL

Tables

PERSON p
SPORTING_EVENT_TICKET t
SEAT s
SPORTING_EVENT e
STADIUM_NOTES sn

Predicate Information

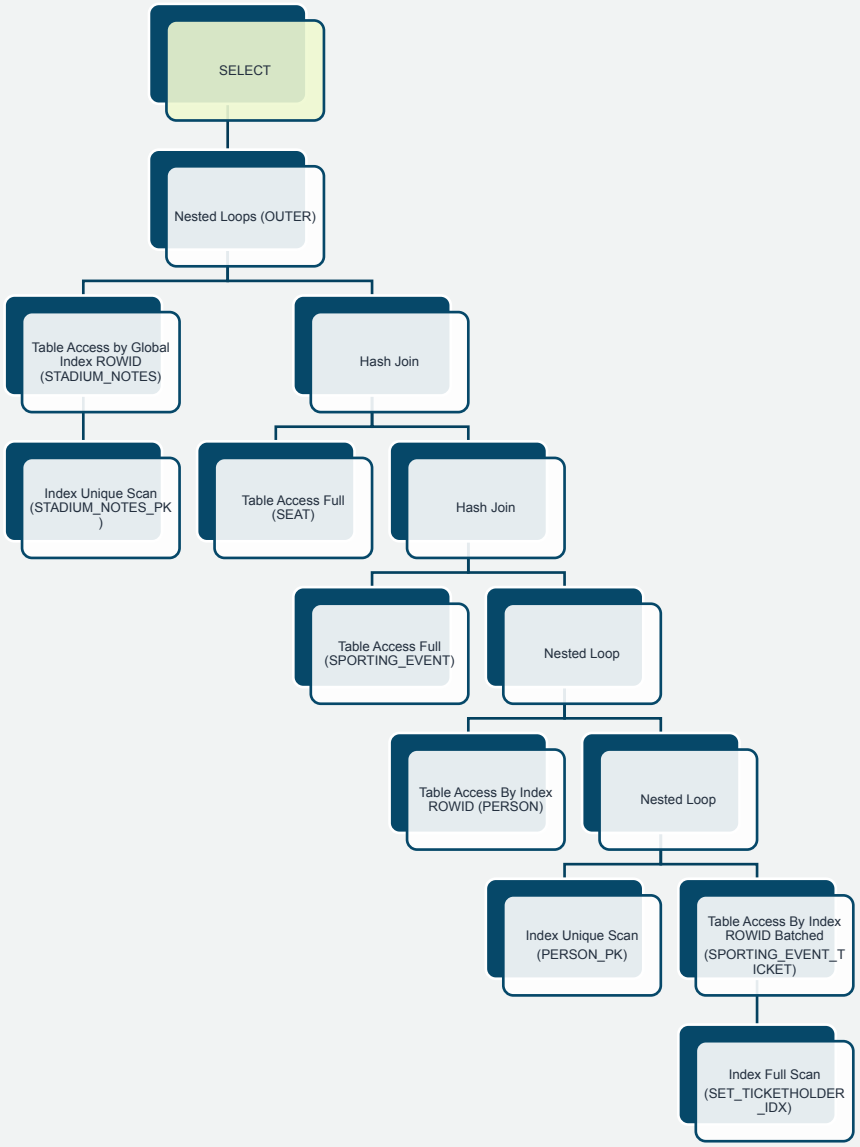
```
t.SPORT_LOCATION_ID =
s.SPORT_LOCATION_ID
        AND t.SEAT_LEVEL =
s.SEAT_LEVEL
        AND t.SEAT_SECTION =
s.SEAT_SECTION
        AND t.SEAT_ROW =
s.SEAT_ROW
        AND t.SEAT = s.SEAT
t.SPORTING_EVENT_ID = e.ID
t.TICKETHOLDER_ID IS NOT NULL
p.ID = t.TICKETHOLDER_ID
t.SPORT_LOCATION_ID = sn.ID
```

aws

```
SELECT
person."ID" AS ID,
person."FULL_NAME" AS FULL_NAME,
person."FIRST_NAME" AS FIRST_NAME,
person."LAST_NAME" AS LAST_NAME,
sporting_event_ticket."SEAT_LEVEL" AS SEAT_LEVEL,
sporting_event_ticket."SEAT_SECTION" AS SEAT_SECTION,
sporting_event_ticket."SEAT_ROW" AS SEAT_ROW,
sporting_event_ticket."SEAT" AS SEAT,
sporting_event_ticket."TICKET_PRICE" AS TICKET_PRICE,
seat."SEAT_TYPE" AS SEAT_TYPE,
sporting_event."SPORT_TYPE_NAME" AS SPORT_TYPE_NAME,
sporting_event."START_DATE_TIME" AS START_DATE_TIME,
stadium_notes."NAME" AS STADIUM_NAME,
stadium_notes."OWNER" AS STADIUM_OWNER,
stadium_notes."TYPE" AS STADIUM_TYPE,
stadium_notes."TEXT" AS STADIUM_NOTES
FROM
newyoda.person
INNER JOIN newyoda.sporting_event_ticket ON person."ID" =
sporting_event_ticket."TICKETHOLDER_ID"
INNER JOIN newyoda.seat ON (
sporting_event_ticket."SPORT_LOCATION_ID" = seat."SPORT_LOCATION_ID"
AND sporting_event_ticket."SEAT_LEVEL" = seat."SEAT_LEVEL"
AND sporting_event_ticket."SEAT_SECTION" = seat."SEAT_SECTION"
AND sporting_event_ticket."SEAT_ROW" = seat."SEAT_ROW"
AND sporting_event_ticket."SEAT" = seat."SEAT"
)
INNER JOIN newyoda.sporting_event ON
sporting_event_ticket."SPORTING_EVENT_ID" = sporting_event."ID"
LEFT JOIN newyoda.stadium_notes ON
sporting_event_ticket."SPORT_LOCATION_ID" = stadium_notes."ID"
WHERE
sporting_event_ticket."TICKETHOLDER_ID" IS NOT NULL;
```

Explain Plan in Oracle

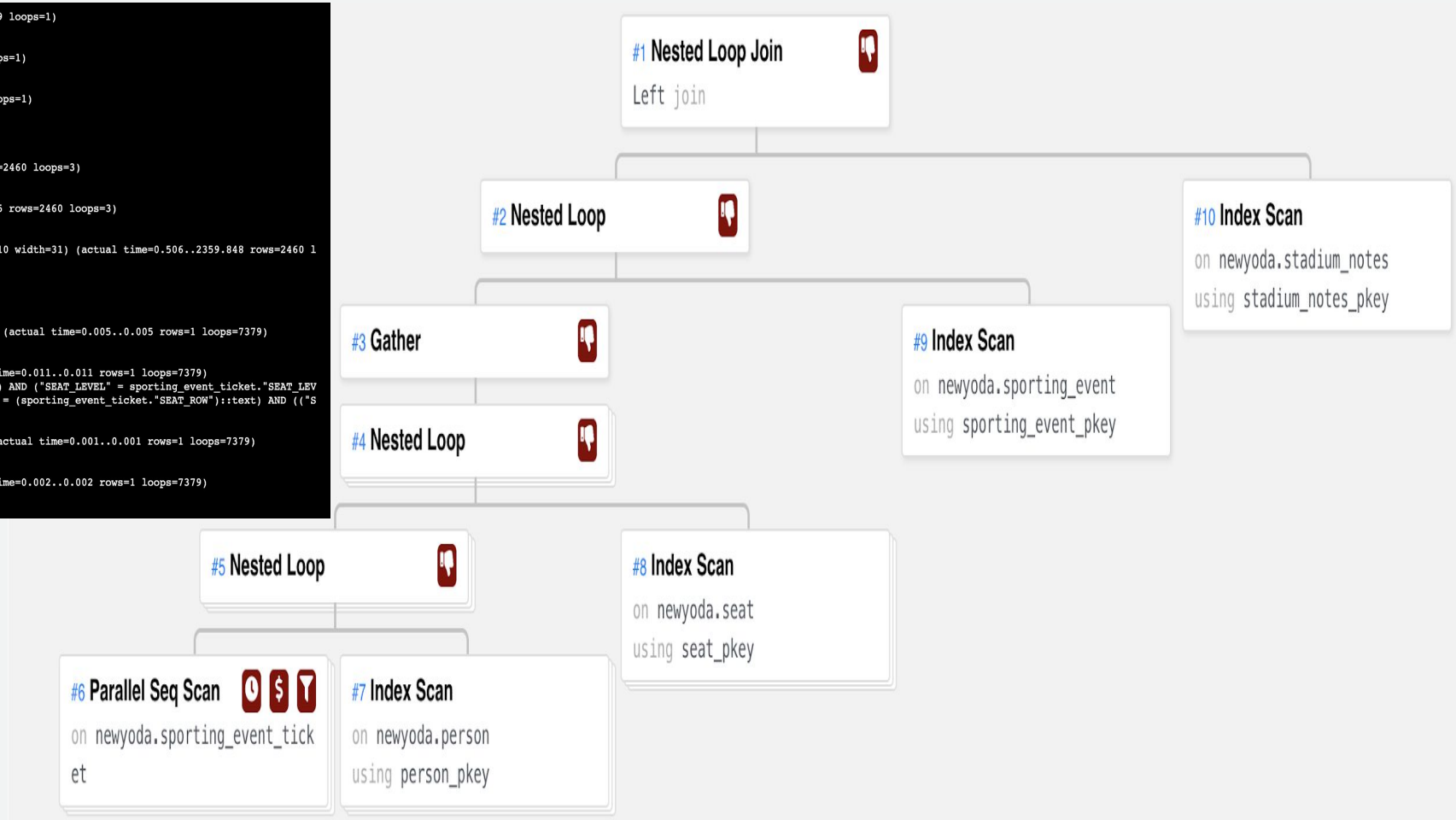
```
Plan hash value: 838293507

-----------------------------------------------------------------------------------------
| Id  | Operation                                | Name                  |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                       |
|   1 |  NESTED LOOPS OUTER                      |                       |
|*  2 |   HASH JOIN                              |                       |
|*  3 |    HASH JOIN                             |                       |
|   4 |     TABLE ACCESS FULL                    | SPORTING_EVENT        |
|   5 |     NESTED LOOPS                         |                       |
|   6 |      NESTED LOOPS                        |                       |
|   7 |       TABLE ACCESS BY INDEX ROWID BATCHED| SPORTING_EVENT_TICKET |
|*  8 |        INDEX FULL SCAN                   | SET_TICKETHOLDER_IDX  |
|*  9 |       INDEX UNIQUE SCAN                  | PERSON_PK             |
|  10 |      TABLE ACCESS BY INDEX ROWID         | PERSON                |
|  11 |    TABLE ACCESS FULL                     | SEAT                  |
|  12 |   TABLE ACCESS BY GLOBAL INDEX ROWID     | STADIUM_NOTES         |
|* 13 |    INDEX UNIQUE SCAN                     | STADIUM_NOTES_PK      |
-----------------------------------------------------------------------------------------
```



aws

Explain Plan In PostgreSQL

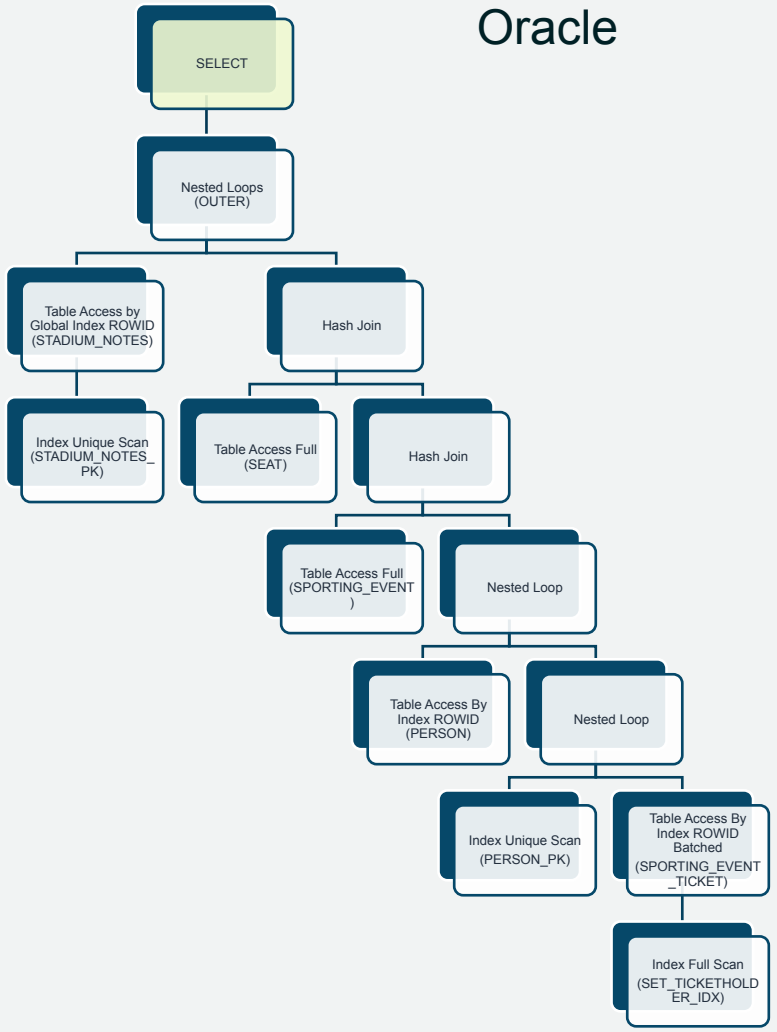
```
Nested Loop Left Join  (cost=1001.71..800857.09 rows=1 width=144) (actual time=0.680..2422.293 rows=7379 loops=1)
  Buffers: shared hit=202509 read=443791
  I/O Timings: shared read=1086.083
  ->  Nested Loop  (cost=1001.14..800848.53 rows=1 width=78) (actual time=0.670..2403.590 rows=7379 loops=1)
        Buffers: shared hit=165614 read=443791
        I/O Timings: shared read=1086.083
        ->  Gather  (cost=1000.86..800841.52 rows=1 width=66) (actual time=0.661..2391.120 rows=7379 loops=1)
              Workers Planned: 2
              Workers Launched: 2
              Buffers: shared hit=143477 read=443791
              I/O Timings: shared read=1086.083
              ->  Nested Loop  (cost=0.86..799841.42 rows=1 width=66) (actual time=0.556..2405.208 rows=2460 loops=3)
                    Buffers: shared hit=143477 read=443791
                    I/O Timings: shared read=1086.083
                    ->  Nested Loop  (cost=0.43..799833.88 rows=1 width=57) (actual time=0.530..2373.145 rows=2460 loops=3)
                          Buffers: shared hit=113959 read=443791
                          I/O Timings: shared read=1086.083
                          ->  Parallel Seq Scan on sporting_event_ticket  (cost=0.00..763767.71 rows=4710 width=31) (actual time=0.506..2359.848 rows=2460 l
oops=3)
                                Filter: ("TICKETHOLDER_ID" IS NOT NULL)
                                Rows Removed by Filter: 18840925
                                Buffers: shared hit=84441 read=443791
                                I/O Timings: shared read=1086.083
                          ->  Index Scan using person_pkey on person  (cost=0.43..7.66 rows=1 width=33) (actual time=0.005..0.005 rows=1 loops=7379)
                                Index Cond: ("ID" = sporting_event_ticket."TICKETHOLDER_ID")
                                Buffers: shared hit=29518
                    ->  Index Scan using seat_pkey on seat  (cost=0.43..7.55 rows=1 width=22) (actual time=0.011..0.011 rows=1 loops=7379)
                          Index Cond: (("SPORT_LOCATION_ID" = sporting_event_ticket."SPORT_LOCATION_ID") AND ("SEAT_LEVEL" = sporting_event_ticket."SEAT_LEV
EL") AND (("SEAT_SECTION")::text = (sporting_event_ticket."SEAT_SECTION")::text) AND (("SEAT_ROW")::text = (sporting_event_ticket."SEAT_ROW")::text) AND (("S
EAT")::text = (sporting_event_ticket."SEAT")::text))
                          Buffers: shared hit=29518
        ->  Index Scan using sporting_event_pkey on sporting_event  (cost=0.28..7.00 rows=1 width=22) (actual time=0.001..0.001 rows=1 loops=7379)
              Index Cond: ("ID" = sporting_event_ticket."SPORTING_EVENT_ID")
              Buffers: shared hit=22137
  ->  Index Scan using stadium_notes_pkey on stadium_notes  (cost=0.57..8.56 rows=1 width=79) (actual time=0.002..0.002 rows=1 loops=7379)
        Index Cond: ("ID" = sporting_event_ticket."SPORT_LOCATION_ID")
        Buffers: shared hit=36895
```



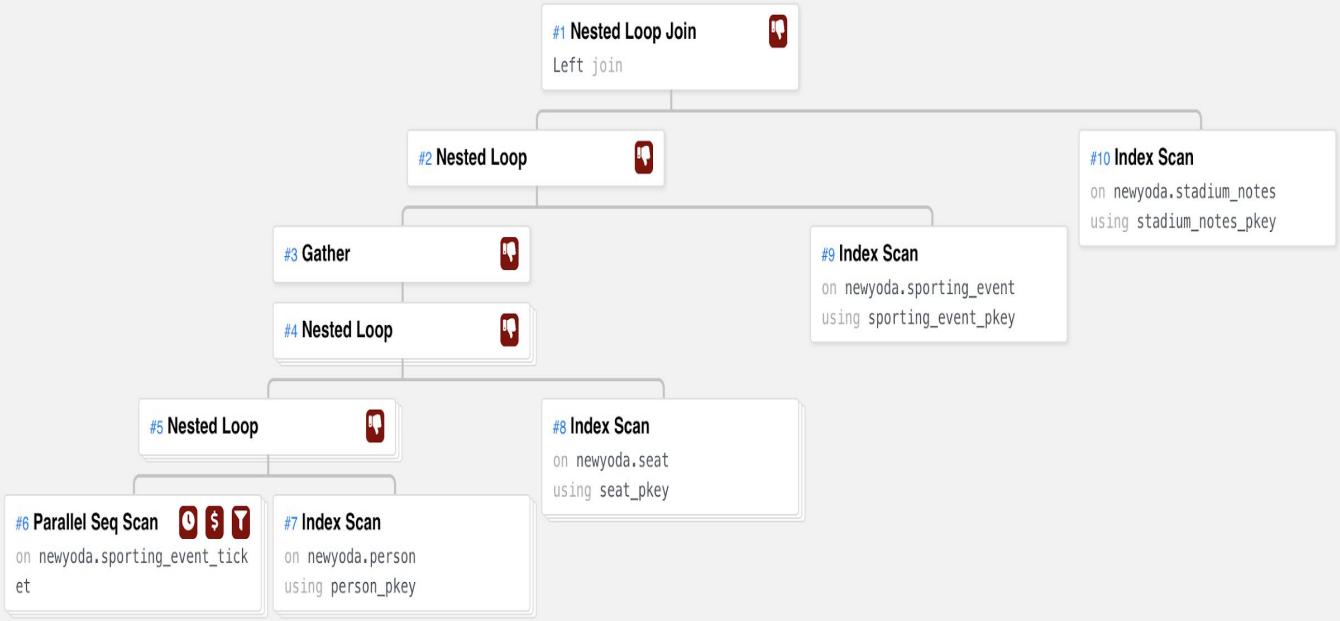
aws

Query Processing

Oracle



PostgreSQL



aws

Q & A

Thank You!

Puja Audhya
AWS

Rakesh Raghav
AWS