Writing Efficient Queries, Part 1

Advanced SQL and Cloud

NASHVILLE SOFTWARE SCHOOL

Objectives

1. Understand what indexes are and how they work
2. Identify when indexes are used in execution plans
3. Recognize common indexing gotchas
4. Understand join strategies and avoid common join pitfalls
5. Practice analyzing and optimizing queries

A Note on Modern SQL Engines

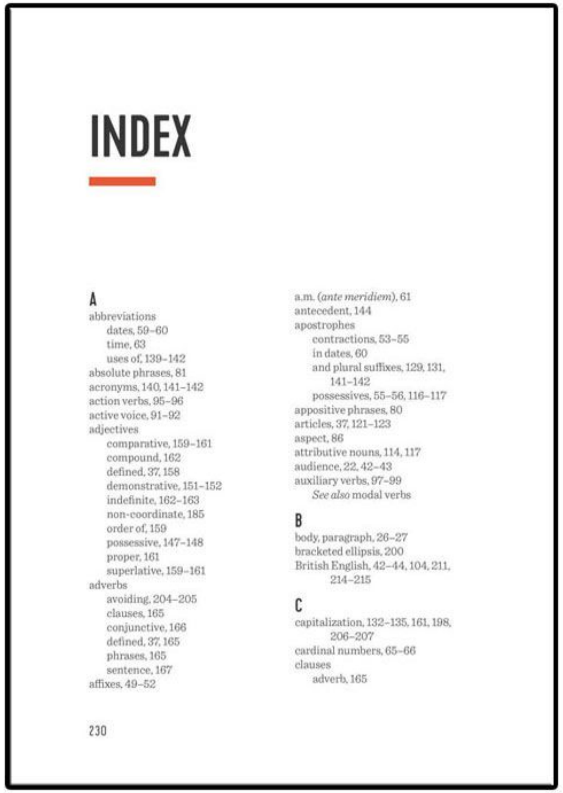
- Today’s SQL engines come with sophisticated query planners that optimize many aspects of query execution.
- Some “tricks” or advice you’ll find online may be outdated or less relevant than they used to be.
- We’ll focus on understanding what the engine does for you and where your choices still matter for performance.

Why Write Efficient SQL?

- Faster queries means faster results, faster iteration, faster dashboards and reports.
- More efficient queries reduce the load on shared databases and avoid slowing down teammates.
- Efficient SQL means more time for analysis and less time waiting for results.

Indexes

- In general, the less data we can scan the faster the results will be.
- An **index** is a data structure which lets the database quickly find rows in a table without having to scan the entire table.
- Think: index in a book. Rather than having to read all of the pages to find a particular topic, you can instead check the index to go directly to the relevant pages.


INDEX

A

abbreviations
    dates, 59–60
    time, 63
    uses of, 139–142
absolute phrases, 81
acronyms, 140, 141–142
action verbs, 95–96
active voice, 91–92
adjectives
    comparative, 159–161
    compound, 162
    defined, 37, 158
    demonstrative, 151–152
    indefinite, 162–163
    non-coordinate, 185
    order of, 159
    possessive, 147–148
    proper, 161
    superlative, 159–161
adverbs
    avoiding, 204–205
    clauses, 165
    conjunctive, 166
    defined, 37, 165
    phrases, 165
    sentence, 167
affixes, 49–52
a.m. (*ante meridiem*), 61
antecedent, 144
apostrophes
    contractions, 53–55
    in dates, 60
    and plural suffixes, 129, 131, 141–142
    possessives, 55–56, 116–117
appositive phrases, 80
articles, 37, 121–123
aspect, 86
attributive nouns, 114, 117
audience, 22, 42–43
auxiliary verbs, 97–99
    See also modal verbs

B

body, paragraph, 26–27
bracketed ellipsis, 200
British English, 42–44, 104, 211, 214–215

C

capitalization, 132–135, 161, 198, 206–207
cardinal numbers, 65–66
clauses
    adverb, 165

230

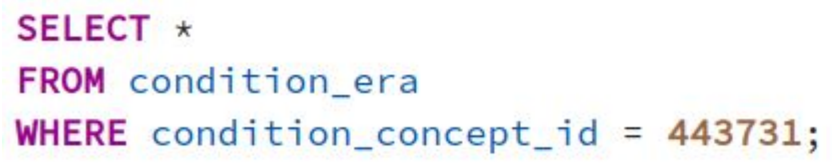
NASHVILLE SOFTWARE SCHOOL

Indexes

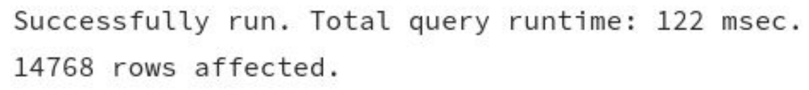
- Indexes improve query performance, especially for WHERE filters, JOIN conditions, and ORDER BY or GROUP BY.
- Useful for checking both equality and inequalities (<, >, or BETWEEN)
- Can be used with LIKE if there's no wildcard at the beginning (LIKE "text%")
- Indexes can cover one or more columns, and in postgres can even include data from other columns so that you can filter on one column quickly and return data from another column without having to retrieve the entire row.
- As an analyst, you typically cannot create or drop indexes, but can use and benefit from existing indexes.

Indexes

This example query runs more than 4 times faster using the index versus not using the index.

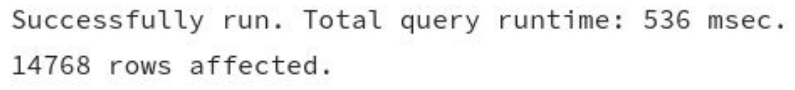


```sql
SELECT *
FROM condition_era
WHERE condition_concept_id = 443731;
```



```
Successfully run. Total query runtime: 122 msec.
14768 rows affected.
```

with index



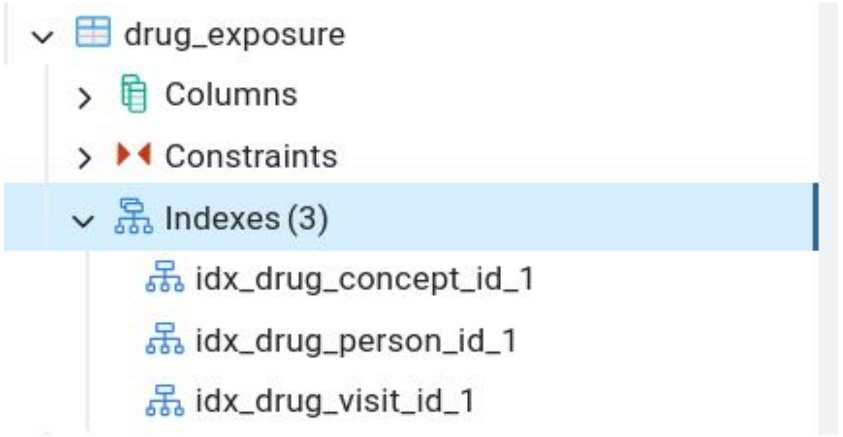
```
Successfully run. Total query runtime: 536 msec.
14768 rows affected.
```

without index

NASHVILLE SOFTWARE SCHOOL

Indexes

To see what indexes are available, you can either check the Indexes dropdown in pgadmin for the table of interest or run the following query:

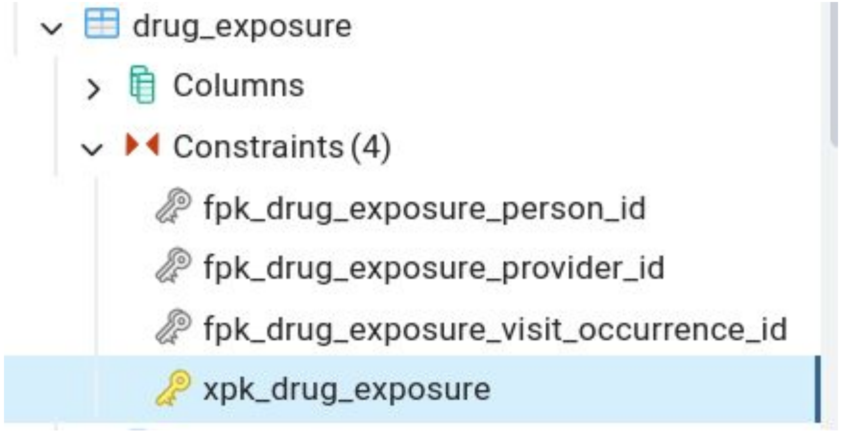

drug_exposure
Columns
Constraints
Indexes (3)
idx_drug_concept_id_1
idx_drug_person_id_1
idx_drug_visit_id_1

```
SELECT indexname, indexdef
FROM pg_indexes
WHERE tablename = 'drug_exposure';
```

NASHVILLE SOFTWARE SCHOOL

Indexes

The primary key of a table will also automatically be indexed by postgres.



drug_exposure
Columns
Constraints (4)
fpk_drug_exposure_person_id
fpk_drug_exposure_provider_id
fpk_drug_exposure_visit_occurrence_id
xpk_drug_exposure

Indexes

- Just because a column has an index on it does not mean that it will always be used when filtering on that column.
  - If a filter returns a large proportion of rows, the index may not be used since there won't be an advantage of just scanning all rows. This would be called a "non-selective filter"
    - WHERE sex = 'M'
    - WHERE status = 'Active' (when 90% of records are active)
  - Indexes will not be used if an expression is used on an indexed column
    - WHERE lower(name) = 'bob'
    - Where EXTRACT(YEAR from date) = 2024

NASHVILLE SOFTWARE SCHOOL

Indexes

- When joining multiple tables, indexes can also help.

- Avoid applying function to join keys, as this can block index uses.

NASHVILLE SOFTWARE SCHOOL

Query Planner

Databases don’t just blindly scan tables. They use a **query planner** to decide how to retrieve the results of a query.

The query planner examines different possible plans and goes with the one it thinks will be the most efficient.

This query planner relies on statistics, or metadata about the data: number of rows, value distributions, how common NULL values are, etc.

EXPLAIN and EXPLAIN ANALYZE

How do we know whether the query planner decided to use an index or not?

Postgres has tools to show execution plans: EXPLAIN and EXPLAIN ANALYZE

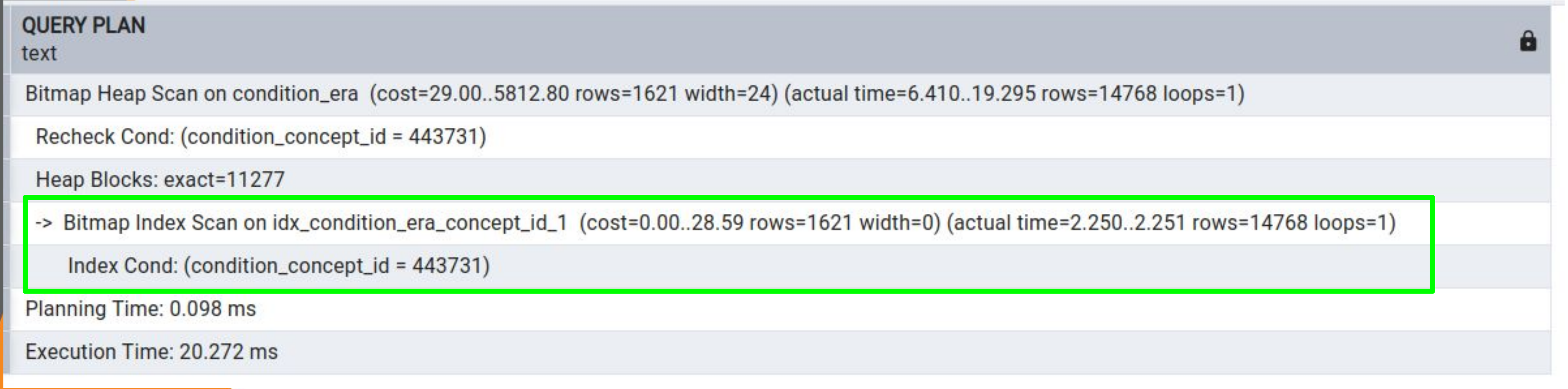
EXPLAIN shows the planner’s estimated plan without executing the query.

EXPLAIN ANALYZE shows the actual time and rows alongside the plan.

NASHVILLE SOFTWARE SCHOOL

EXPLAIN and EXPLAIN ANALYZE

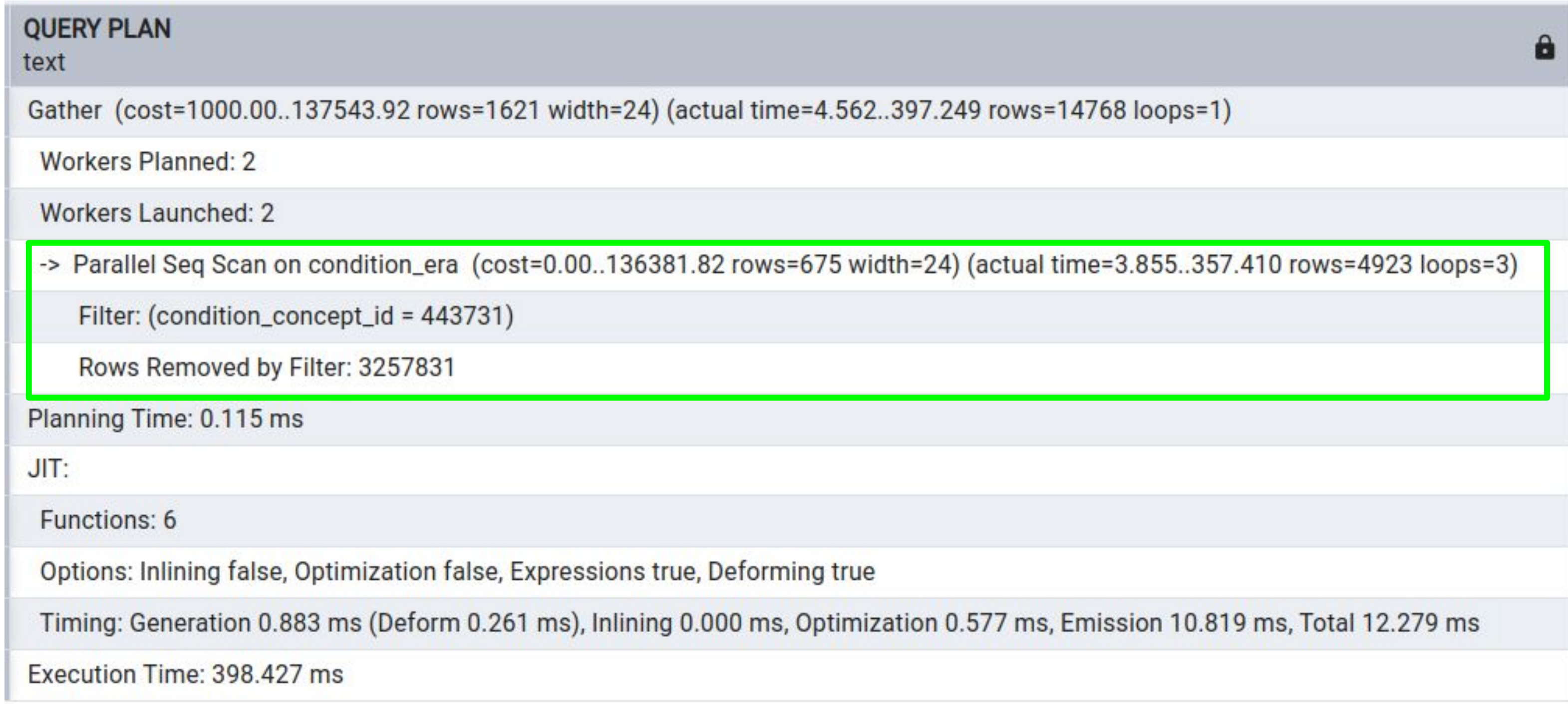
Here is the query plan for the earlier query using the index.



```
QUERY PLAN
text
Bitmap Heap Scan on condition_era  (cost=29.00..5812.80 rows=1621 width=24) (actual time=6.410..19.295 rows=14768 loops=1)
  Recheck Cond: (condition_concept_id = 443731)
  Heap Blocks: exact=11277
  -> Bitmap Index Scan on idx_condition_era_concept_id_1  (cost=0.00..28.59 rows=1621 width=0) (actual time=2.250..2.251 rows=14768 loops=1)
        Index Cond: (condition_concept_id = 443731)
Planning Time: 0.098 ms
Execution Time: 20.272 ms
```

EXPLAIN and EXPLAIN ANALYZE

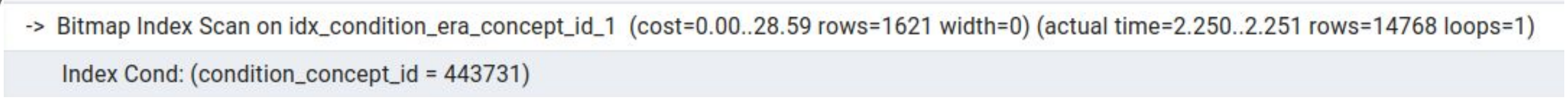
And the one without the index.



```
QUERY PLAN
text
Gather  (cost=1000.00..137543.92 rows=1621 width=24) (actual time=4.562..397.249 rows=14768 loops=1)
  Workers Planned: 2
  Workers Launched: 2
  ->  Parallel Seq Scan on condition_era  (cost=0.00..136381.82 rows=675 width=24) (actual time=3.855..357.410 rows=4923 loops=3)
        Filter: (condition_concept_id = 443731)
        Rows Removed by Filter: 3257831
Planning Time: 0.115 ms
JIT:
  Functions: 6
  Options: Inlining false, Optimization false, Expressions true, Deforming true
  Timing: Generation 0.883 ms (Deform 0.261 ms), Inlining 0.000 ms, Optimization 0.577 ms, Emission 10.819 ms, Total 12.279 ms
Execution Time: 398.427 ms
```

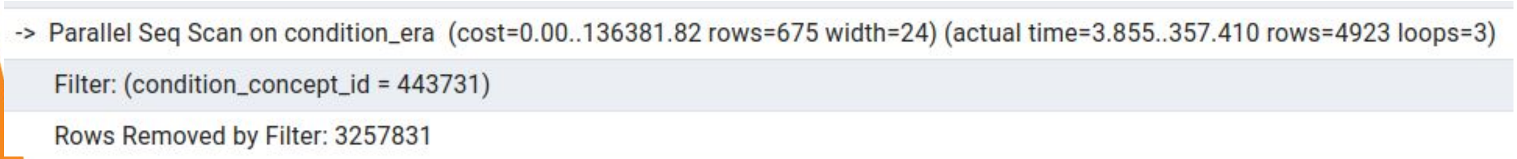
EXPLAIN and EXPLAIN ANALYZE

This one jumps directly to the relevant rows.



```
-> Bitmap Index Scan on idx_condition_era_concept_id_1  (cost=0.00..28.59 rows=1621 width=0) (actual time=2.250..2.251 rows=14768 loops=1)
     Index Cond: (condition_concept_id = 443731)
```

This one looks at all rows and has to discard those that don't match.



```
-> Parallel Seq Scan on condition_era  (cost=0.00..136381.82 rows=675 width=24) (actual time=3.855..357.410 rows=4923 loops=3)
     Filter: (condition_concept_id = 443731)
     Rows Removed by Filter: 3257831
```

NASHVILLE SOFTWARE SCHOOL

EXPLAIN and EXPLAIN ANALYZE

For more information about reading query plans, see

- https://www.postgresql.org/docs/current/sql-explain.html
- https://pganalyze.com/docs/explain/basics-of-postgres-query-planning
- https://www.pgmustard.com/blog/2018/09/21/reading-postgres-query-plans-for-beginners
- https://www.depesz.com/tag/unexplainable/