+Jod e Obenson

1.)
   SELECT * FROM public.supplier splr, public.supplies spls WHERE splr.sid = spls.sid AND splr.sid IN(3,5,9);

2.)

   CREATE INDEX tp3_pg_index1 on public.supplier(sid);
   CREATE INDEX tp3_pg_index on public.supplies(sid);

3 .)

   Uma consulta sem índice

```
postgres=# SELECT  * FROM public.supplier splr, public.supplies spls WHERE splr.sid = spls.sid AND splr.sid IN(3,5,9);
 sid |        name         |      street       |    city     | zipcode | eid | sid
-----+---------------------+-------------------+-------------+---------+-----+-----
   5 | Everett Chemical    | 8932 174th Ave W  | Worcester   | 01386   |  14 |   5
   3 | Intercon. Linens    | 15521 188th Ave W | Chicopee    | 01952   |   4 |   3
   5 | Everett Chemical    | 8932 174th Ave W  | Worcester   | 01386   |   4 |   5
   3 | Intercon. Linens    | 15521 188th Ave W | Chicopee    | 01952   |  17 |   3
   5 | Everett Chemical    | 8932 174th Ave W  | Worcester   | 01386   |   9 |   5
   9 | Hippocratic Systems | 26417 27th Ave N  | Springfield | 01478   |   5 |   9
   3 | Intercon. Linens    | 15521 188th Ave W | Chicopee    | 01952   |   1 |   3
(7 rows)

postgres=# explain(analyse) SELECT  * FROM public.supplier splr, public.supplies spls WHERE splr.sid = spls.sid AND splr.sid IN(3,5,9);
                                                      QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=1.16..2.57 rows=8 width=60) (actual time=0.060..0.078 rows=7 loops=1)
   Hash Cond: (spls.sid = splr.sid)
   ->  Seq Scan on supplies spls  (cost=0.00..1.24 rows=24 width=8) (actual time=0.019..0.025 rows=24 loops=1)
   ->  Hash  (cost=1.12..1.12 rows=3 width=52) (actual time=0.022..0.023 rows=3 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 9kB
         ->  Seq Scan on supplier splr  (cost=0.00..1.12 rows=3 width=52) (actual time=0.012..0.016 rows=3 loops=1)
               Filter: (sid = ANY ('{3,5,9}'::integer[]))
               Rows Removed by Filter: 6
 Planning Time: 0.259 ms
 Execution Time: 0.130 ms
(10 rows)

postgres=#
```

Com índice

```
             Table "public.supplies"
 Column |  Type   | Collation | Nullable | Default
--------+---------+-----------+----------+---------
 eid    | integer |           |          |
 sid    | integer |           |          |
Indexes:
    "tp3_pg_index" btree (sid)

postgres=# \d supplier;
                  Table "public.supplier"
 Column  |         Type          | Collation | Nullable | Default
---------+-----------------------+-----------+----------+---------
 sid     | integer               |           | not null |
 name    | character varying(20) |           |          |
 street  | character varying(20) |           |          |
 city    | character varying(15) |           |          |
 zipcode | character varying(5)  |           |          |
Indexes:
    "tp3_pg_index1" btree (sid)

postgres=# explain(analyse) SELECT  * FROM public.supplier splr, public.supplies spls WHERE splr.sid = spls.sid AND splr.sid IN(3,5,9);
                                                                 QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=1.16..2.57 rows=8 width=60) (actual time=0.072..0.111 rows=7 loops=1)
   Hash Cond: (spls.sid = splr.sid)
   ->  Seq Scan on supplies spls  (cost=0.00..1.24 rows=24 width=8) (actual time=0.017..0.029 rows=24 loops=1)
   ->  Hash  (cost=1.12..1.12 rows=3 width=52) (actual time=0.030..0.031 rows=3 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 9kB
         ->  Seq Scan on supplier splr  (cost=0.00..1.12 rows=3 width=52) (actual time=0.015..0.021 rows=3 loops=1)
               Filter: (sid = ANY ('{3,5,9}'::integer[]))
               Rows Removed by Filter: 6
 Planning Time: 0.625 ms
 Execution Time: 0.178 ms
(10 rows)

postgres=#
```

A consulta não está acionando o índice porque o SGBD acha mais fácil percorrer a tabela e retornar as informações. Sendo assim, como na primeira não tinha índices e na segunda não está acionando eles, o custo fica igual para as duas situações.

4- Qual algoritmo foi usado para realizar o Join? Explique o seu funcionamento.

Foi usado o hash join que, na realidade, é uma forma de optimização de dados geralmente quando as tabelas são grandes .

Todas as variantes de algoritmos de junção de hash envolvem a construção de tabela a partir das tuplas de uma ou ambas as relações unidas e, subsequentemente, sondar essas tabelas para que apenas tuplas com o mesmo código de hash precisem ser comparadas quanto à igualdade em equijoins.

As junções de hash são normalmente mais eficientes do que as junções de loops aninhados, exceto quando o lado da investigação da junção é muito pequeno.