CS486/586 Introduction to Databases
Spring 2022 Quarter
Submitted by: - Parth Parashar

Assignment 6 – Query Optimization, Transactions, Recovery
Due: Monday, May 31 at 11:59PM.

Instructions & Notes:

- Ensure that each group member's name is listed on the assignment.
- Submit your assignment in PDF format.
- Submit your completed assignment on Canvas.
- 100 points total.

Part I: Join Implementation – 15 points

Question 1 (15 points):

1. Write a SQL query using the spy schema for which you believe it would be efficient to use hash join. Include the query here.

Answer: - The required query which would be efficient for using hash join is given below: -
select s.skill, r.agent_id from skill s join skillrel r ON s.skill_id = r.skill_id;

2. Explain why you believe the hash join algorithm could or would be used.

Answer: - The skillrel table is the second largest table in the spy schema. It contains 9 pages and 1954 tuples.
On the other hand, skill table is the smallest table in the schema with 1 page and 66 tuples.
Now as we know that hash join is an efficient choice when one of the tables contains a lot of more tuples than the other i.e., one of the tables is considerably larger than the other table.
Because all of the tables in the spy schema will fit in the memory, we didn't consider whether or not a table would fit in the memory when selecting tables to include in the query.

3. Use EXPLAIN to see (and report) which join algorithm postgresql actually uses. (Note: It's fine if the join algorithm postgresql uses does not match the join algorithm named in the question.)

Answer: - Postgre sql uses hash join as given in the screenshot below: -

```
spr2022adb35=> explain select s.skill, r.agent_id from skill s join skillrel r ON s.skill_id = r.skill_id;
                            QUERY PLAN
------------------------------------------------------------------
 Hash Join  (cost=2.49..36.49 rows=1954 width=19)
   Hash Cond: (r.skill_id = s.skill_id)
   ->  Seq Scan on skillrel r  (cost=0.00..28.54 rows=1954 width=8)
   ->  Hash  (cost=1.66..1.66 rows=66 width=19)
         ->  Seq Scan on skill s  (cost=0.00..1.66 rows=66 width=19)
(5 rows)

spr2022adb35=>
```

Part II: Statistics – 25 points

Question 2 (5 points): Determine the min and max salary values in the agent table, and the number of rows in that table.

Answer: - Min Salary = 50008
Max Salary = 366962
Row count in agent table = 662

The queries used for finding these results is mentioned below: -
select min(salary), max(salary) from agent;
select relname, reltuples from pg_class where relname='agent';

```
spr2022adb35=> select min(salary), max(salary) from agent;
  min   |   max
--------+--------
 50008  | 366962
(1 row)

spr2022adb35=> select relname, reltuples from pg_class where relname='agent';
 relname | reltuples
---------+-----------
 agent   |       662
(1 row)

spr2022adb35=>
```

Question 3 (5 points): Give an estimate for the number of rows in agent with salary < 92000, assuming a uniform distribution of salaries between min and max salary. Explain how you derived your estimate.

Answer: -
Step-1: - First we find the range (T) between the minimum salary and maximum salary:
T = max - min = 366,962 - 50,008 = 316,954

Step-2: -
Given that we are assuming a uniform distribution, we can divide T by the number of rows in the table plus one to find out the range between each salary (S).

Step-3: -
We need to add one to account for the fact that T is inclusive of both the min and the max:
S = T / total rows = 316,954 / (662 + 1) = 478

Step-4: -
Then we find the range (R) between our target value of 92,000 and the minimum salary:
R = target value - min = 92,000 - 50,008 = 41,992

Step-5: -
Finally, we divide R by S to find the number of rows with a salary lower than 92,000:
R / S = 41,992 / 478 = 87.85

Since we can only have whole values for rows, we ignored the decimal and estimated that there are 87 rows in the agent table with a salary < 92,000.

Question 4 (5 points): Find the 25th, 50th and 75th percentile values for salaries in the agent table. (The 50th percentile value, for instance, is the smallest number such that 50% of the rows have salary value less than or equal to s.)

Answer: - 25th Percentile: 54,802
50th Percentile: 58,430
75th Percentile: 89,643

The query used for finding out the result is given below: -

```
spr2022adb35=> SELECT
spr2022adb35-> PERCENTILE_DISC(.25) WITHIN GROUP (ORDER BY agent.salary),
spr2022adb35-> PERCENTILE_DISC(.5) WITHIN GROUP (ORDER BY agent.salary),
spr2022adb35-> PERCENTILE_DISC(.75) WITHIN GROUP (ORDER BY agent.salary)
spr2022adb35-> FROM agent;
 percentile_disc | percentile_disc | percentile_disc
-----------------+-----------------+-----------------
           54802 |           58430 |           89643
(1 row)

spr2022adb35=>
```

Question 5 (5 points): Give an estimate of the number of rows in agent with salary < 92000, assuming in a uniform distribution in each quartile determined in Question 4. Explain how you derived your estimate. (Uniform distribution means values are evenly distributed between the min and max.)

Answer: - Using the quartiles from Question 4, we can say that the 75th is at 89,643 which is very close to the target value of 92,000.

Because of this, we can estimate that at least 75% of the rows in agent have a salary < 92,000:

Therefore, we can say that
total rows * .75 = 662 * .75 = 496.5

Since we can only have whole values for rows, we ignored the decimal and estimated that there are 496 rows in the agent table with a salary < 92,000.

Question 6 (5 points): How many rows in the agent table actually have salary < 92000?

Answer: - The rows with salary< 92000 are 572.
This is given by the query and its result
select count(*) from agent where salary < 92000;

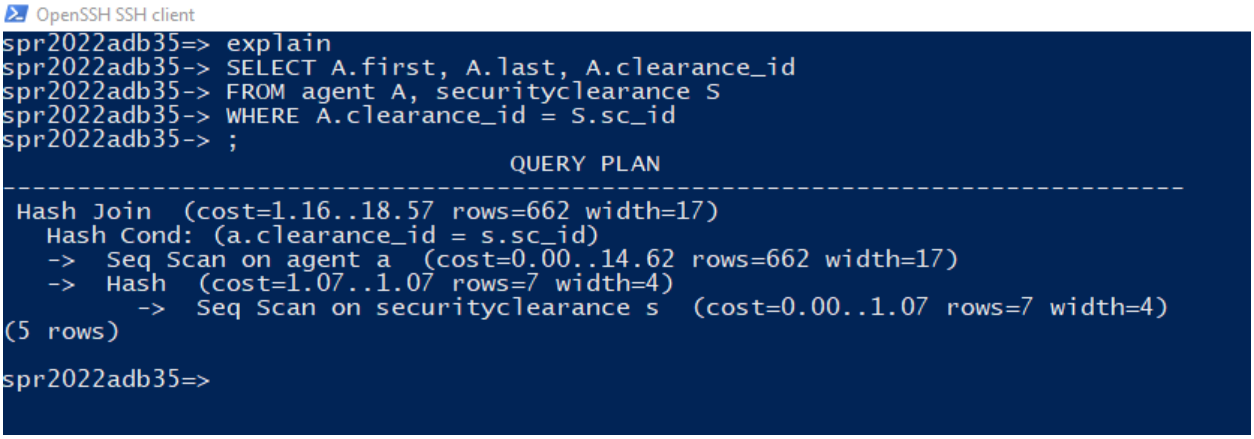
```
spr2022adb35=> select count(*) from agent where salary < 92000;
 count
-------
   572
(1 row)
```

Part III: Query Plans, cont'd – 60 points

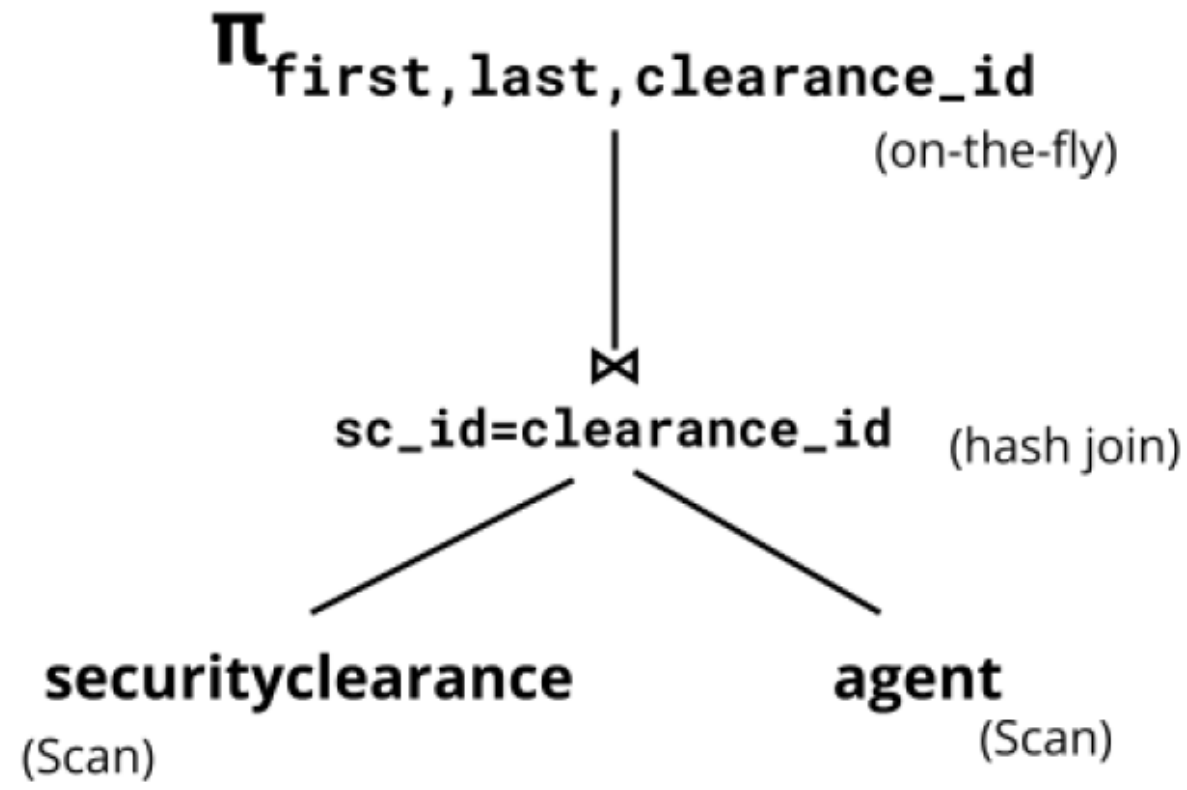
Following on the query plans work from HW 5 and using the same set of queries, please do the following. For each SQL statement below, draw the query plan PostgreSQL uses (you can find the query plan with the EXPLAIN command See Documentation: 14: EXPLAIN Also, see slide 19 in Slides 12 for and Activities for Slides 12 for information about Explain). For each plan, suggest a reason that the particular join algorithms were chosen.

Question 7 (20 points):
SELECT A.first, A.last, A.clearance_id
FROM agent A, securityclearance S
WHERE A.clearance_id = S.sc_id

Answer: -



Query Plan: -



Reason to choose Hash Join Algorithm: -
securityclearance is a very small table (7 rows), and agents is one of the larger tables in the spy schema (662 rows).
Therefore, Hash join is a good choice for joining these two relations.
This can be attributed to the fact that hash join works better when one of the relations is small and the other relation is large.

Question 8 (20 points):
SELECT L.language, L.lang_id
FROM language L, languagerel LR
WHERE L.lang_id = LR.lang_id;

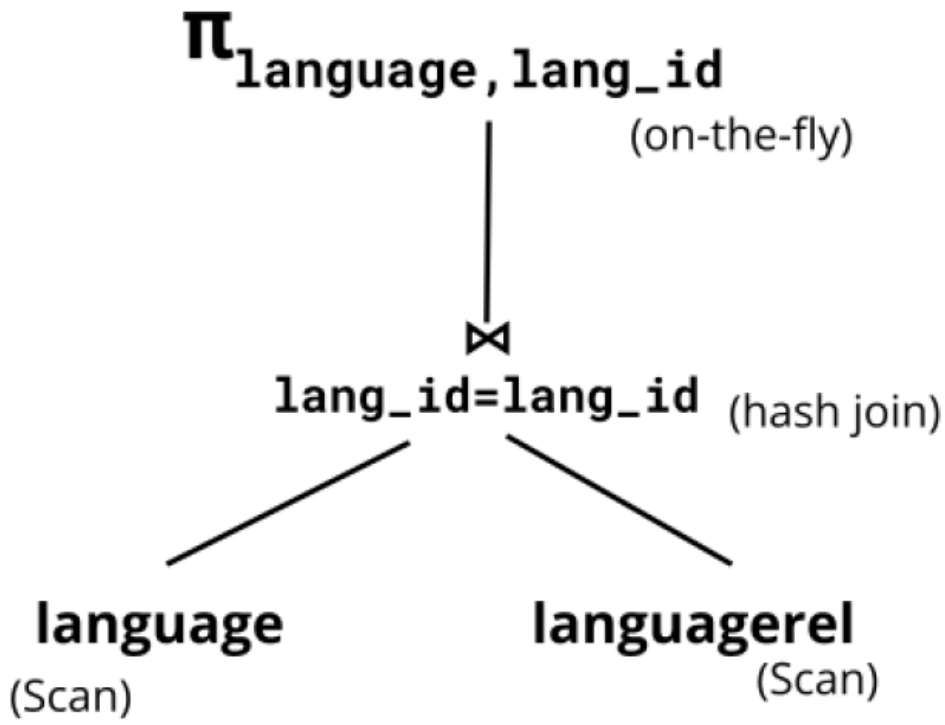
Answer: -

OpenSSH SSH client

```
spr2022adb35=> explain
spr2022adb35-> SELECT L.language, L.lang_id
spr2022adb35-> FROM language L, languagerel LR
spr2022adb35-> WHERE L.lang_id = LR.lang_id;
                              QUERY PLAN
-----------------------------------------------------------------------
 Hash Join  (cost=1.45..36.70 rows=1991 width=62)
   Hash Cond: (lr.lang_id = l.lang_id)
   ->  Seq Scan on languagerel lr  (cost=0.00..28.91 rows=1991 width=4)
   ->  Hash  (cost=1.20..1.20 rows=20 width=62)
         ->  Seq Scan on language l  (cost=0.00..1.20 rows=20 width=62)
(5 rows)

spr2022adb35=>
```

Query Plan: -



Reason to Choose Hash Join Algorithm:
language is a small table (20 rows), and *languagerel* is the largest table in the *spy* schema (1991 rows). Therefore, Hash join is chosen.
This is because hash join is generally a good choice for joining two relations when one of the relations is small and the other relation is large.

Question 9 (20 points):
SELECT A1.agent_id, A2.agent_id
FROM agent A1, agent A2
WHERE A1.salary < A2.salary

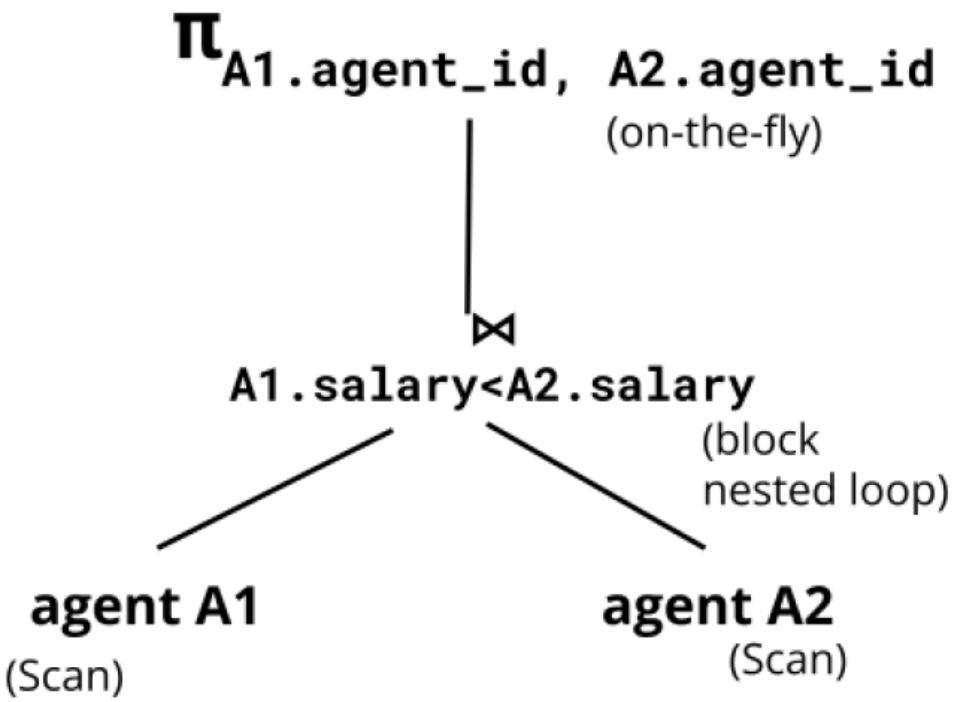
Answer: -

OpenSSH SSH client

```
spr2022adb35=> explain
spr2022adb35-> SELECT A1.agent_id, A2.agent_id
spr2022adb35-> FROM agent A1, agent A2
spr2022adb35-> WHERE A1.salary < A2.salary
spr2022adb35-> ;
                            QUERY PLAN
------------------------------------------------------------------------
 Nested Loop  (cost=0.00..6604.56 rows=146081 width=8)
   Join Filter: (a1.salary < a2.salary)
   ->  Seq Scan on agent a1  (cost=0.00..14.62 rows=662 width=8)
   ->  Materialize  (cost=0.00..17.93 rows=662 width=8)
         ->  Seq Scan on agent a2  (cost=0.00..14.62 rows=662 width=8)
(5 rows)

spr2022adb35=>
```

Query Plan: -



Reason to Choose Block Nested Loop Join Algorithm:
This query does not use an equi-join, and block nested loop is the go-to choice for queries that have JOINs that are not equi-joins.