**Name: Chitradevi Maruthavanan**

**CS486/586 Introduction to Databases**

**Summer 2022 Quarter**

Assignment 5 – Storage and Indexing; Query Evaluation

Due: Monday, August 8th, 11:59 pm

**Instructions & Notes:**

* Ensure that each group member’s name is listed on the assignment, and in the notes field of Canvas to ensure credit.
* Submit your assignment in PDF format.
* Submit your completed assignment on Canvas, including both of your names for each group.
* 100 points total.
* This assignment uses the postgresql EXPLAIN command. You can find info on the EXPLAIN command at: <https://www.postgresql.org/docs/12/performance-tips.html>
* **The instructions for this assignment are a bit more complex than previous assignments. Be sure to read through the questions carefully and completely.**

**Part I - Index Matching (20 points total)**

Schema: Stadiums(id, name, maximum\_capacity, field\_size)

Assume a clustered index on id and a multi-attribute index on (maximum\_capacity, field\_size).

For each selection predicate below, say if the index “matches” the predicate. If the index does not match the predicate, give a brief explanation as to why the index does not match the predicate. You can find the appropriate capacity and square foot information for the two records contained in the schema here:

* Providence Park at <https://en.wikipedia.org/wiki/Providence_Park>
* CenturyLink Field at <https://en.wikipedia.org/wiki/CenturyLink_Field>

For questions c and d, the field\_size attribute is measured in square feet (**not yards**).

***Question 1 (20 points)***

**Schema:** Stadiums(id, name, maximum\_capacity, field\_size)

Assume a clustered index on id and a multi-attribute index on (maximum\_capacity, field\_size)

**Answer:**

1. **name = 'Providence Park'**

**T**he index **does not match** the name predicate. Since, we have the index on the attributes are id, maximum\_capacity, field\_size, but we do not have index on the name attribute.

1. **id < 3**

**T**he index **does not match** the predicate.

1. **field\_size < 18000 AND maximum\_capacity > 25000**

**T**he index **matches** the predicate.

1. **maximum\_capacity < 35000 OR field\_size > 17000**

**T**he index **does not match** the predicate.

1. **Which of the above predicates will Providence Park satisfy? Which will the CenturyLink Field satisfy?**

**Providence Park Predicates are:**

* 1. name = ‘Providence Park’
  2. id < 3
  3. maximum\_capacity < 35000 OR field\_size > 17000

**Century Link Predicates are:**

1. id < 3
2. maximum\_capacity < 35000 OR field\_size > 17000

**Part II: Query Plans – (80 points total)**

Some questions in this section ask you to use the pg\_class table which contains information about the number of pages and tuples in a relation. You can see information about the pg\_class table here: [Documentation: 14: 52.11. pg\_class](https://www.postgresql.org/docs/14/catalog-pg-class.html) (Also see [supplementary video](https://media.pdx.edu/media/t/1_q62sb9eb))

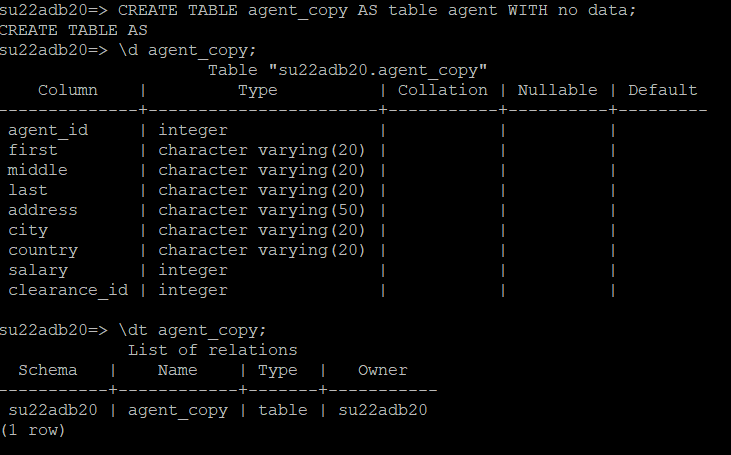
***Question 2 (20 points):***

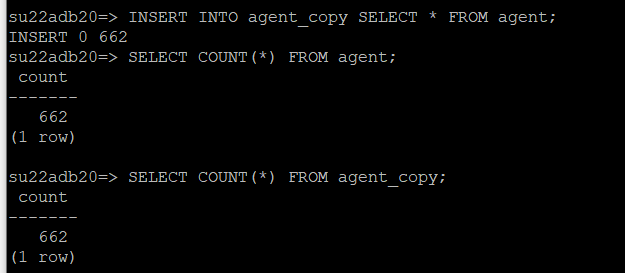
Make a copy of the agent table using the following set of commands. By running the first command and excluding the line 'WITH NO DATA' you can copy the schema and the data in one command; however, when dealing with data you are unfamiliar with it's

a) good to know how to copy the schema and data separately, and

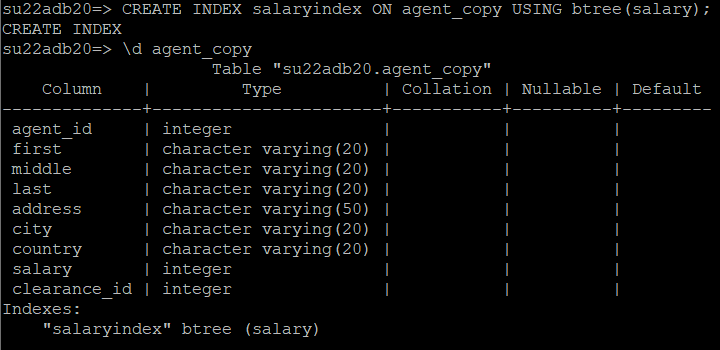
b) generally considered a best practice to do so.

**CREATE TABLE <new table name> AS TABLE <existing table> WITH NO DATA;**

****

**INSERT INTO <new table name> SELECT \*FROM <existing table>;**

Create an index on the salary attribute of the **copy** of the agent table. For the purposes of the SQL statements below, it will be referred to as **'agent\_copy’** though you may have named it something different.(You can use \help in the command line interface to get the syntax for CREATE INDEX).

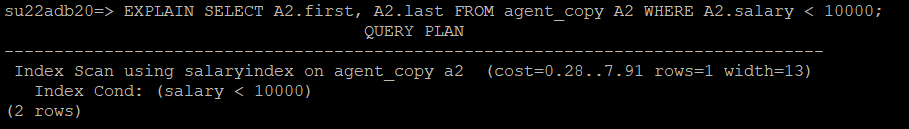


For each SQL statement below, use the EXPLAIN command to find the query plan that postgresql uses.

For your answer, indicate if **Postgres chooses an index** or not.

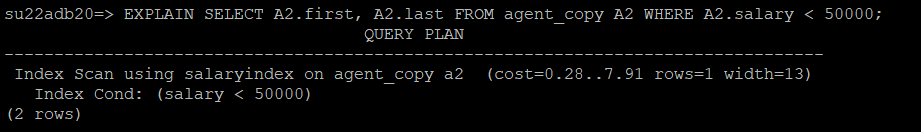
SELECT A2.first, A2.last FROM agent\_copy A2 WHERE A2.salary < 10000;

**Postgres chose an index**



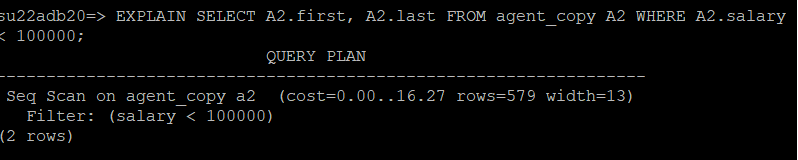
SELECT A2.first, A2.last FROM agent\_copy A2 WHERE A2.salary < 50000;

**Postgres chose an index**

****

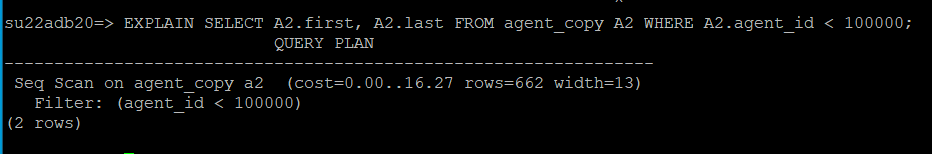
SELECT A2.first, A2.last FROM agent\_copy A2 WHERE A2.salary < 100000;

**Postgres did not choose an index**

****

SELECT A2.first, A2.last FROM agent\_copy A2 WHERE A2.agent\_id < 100000;

**Postgres did not choose an index**

****

**When you have answered this question, delete the copied table using the following command:**

DROP TABLE <name of copy of agent table>;



***Questions 3 - 5***

For each SQL query below, do the following:

1. Write a query for the pg\_class table to get the number of pages and tuples in each relation. Show the query and the results. No need to repeat answers if a table is used multiple times in the questions below. That is, for each question, just get information about “new” tables.
2. List two types of joins that could be used for the query.
3. Using the formulas provided in the slides, calculate the cost of doing each type of join listed in b. **Note:** Keep in mind number of scans is a number of times of scanning; therefore, it needs to be in integer.
4. Use EXPLAIN to identify which join algorithm postgresql uses. (See [Documentation: 14: EXPLAIN](https://www.postgresql.org/docs/14/sql-explain.html) Also, see slide 19 in Slides 12 for and Activities for Slides 12 for information about Explain) (Also see supplementary video)
5. Report if your calculation of which join is cheapest matches postgresql’s choice or not
6. Use work\_mem / 8k to find # of buffer pages available to join ([Documentation: 13: 19.4. Resource Consumption](https://www.postgresql.org/docs/14/runtime-config-resource.html)) (Also see supplementary video)

***Question 3 (20 points):***

SELECT A.first, A.last, A.clearance\_id

FROM agent A, securityclearance S

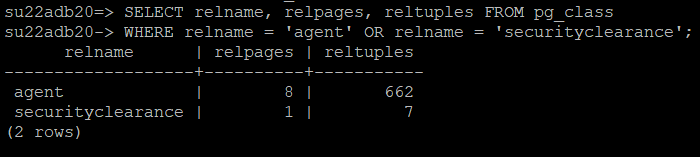
WHERE A.clearance\_id = S.sc\_id

**Answer:**

1. **Query:**

SELECT relname, relpages, reltuples FROM pg\_class

WHERE relname = 'agent' OR relname = 'securityclearance';

****

Number of pages in agent: 8

Number of tuples in agent: 662

Number of pages in security clearance: 1

Number of tuples in security clearance: 7

1. Index nested loop or Sort-Merge
2. **Index Nested Loop**

M = 8 pages in agent

M \* PA = 662 tuples in agent

Cost = M + ((M \* PA) \* 2)

= 8 + (662 \* 2)

= 1332 I/Os

**Sort-Merge**

M = 8 pages in agent

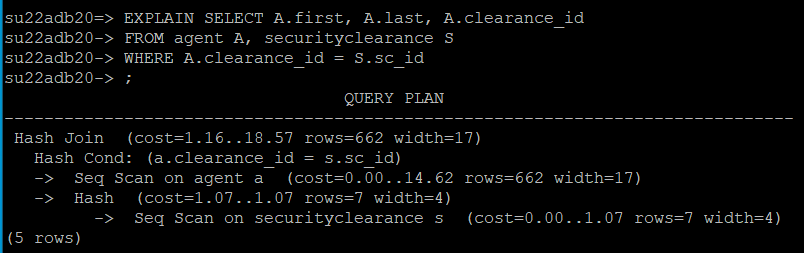
N = 1 page in securityclearance

Cost = 3 \* (M + N)

= 3 \* (8 + 1)

= 27 I/Os

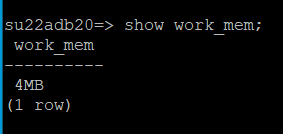
1. postgresql uses a hash join.



1. Here Postgre hash join was the cheapest at 18.57 I/Os.

Of my 2 choices Sort-Merge was the cheaper at 27 I/Os than Index-nested loop at 1332 I/Os

1. **To find # of buffer pages available to join**



Number of buffer pages = show work\_mem = 4MB = 512 Buffer Pages

8KB 8KB

***Question 4 (20 points):***

SELECT L.language, L.lang\_id

FROM language L, languagerel LR

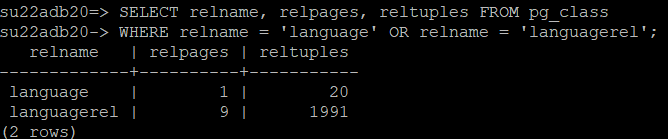
WHERE L.lang\_id = LR.lang\_id;

**Answer:**

**a) Query:**

SELECT relname, relpages, reltuples FROM pg\_class

WHERE relname = 'language' OR relname = 'languagerel';

****

Number of pages in language: 1

Number of tuples in language: 20

Number of pages in languagerel: 9

Number of tuples in languagerel: 1991

**b)** Simple Nested Loop or Page Nested Loop

**c) Simple Nested Loop**

M = 1 page in language

N = 9 pages in languagerel

PA \* M = 20 tuples in language

Cost = M + (PA \* M) \* N

= 1 + 20 \* 9

= 181 I/Os

**Paged Nested Loop**

M = 1 page in language

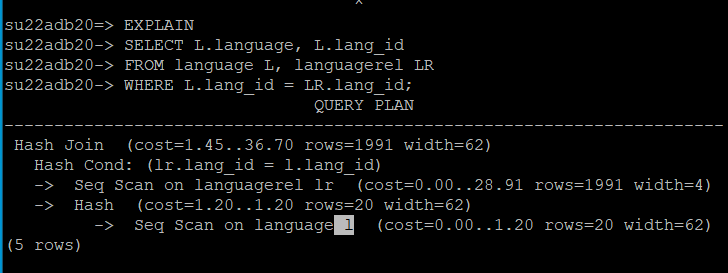
N = 9 pages in languagerel

Cost = M + M \* N

= 1 + 1 \* 9

= 10 I/Os

**d) postgresql uses a hash join.**

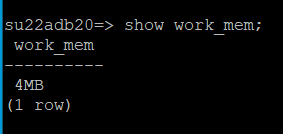


e) Here Postgresql chose hash join. The cost is more expensive at 36.70 I/Os.

Of my 2 choices, page nested loop was the cheapest at 10 I/Os.

(simple nested loop was 181 I/Os)

f)**To find # of buffer pages available to join**



Number of buffer pages = show work\_mem = 4MB = 512 Buffer Pages

8KB 8KB

***Question 5 (20 points):***

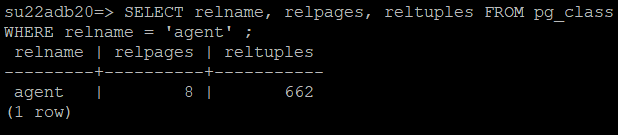
SELECT A1.agent\_id, A2.agent\_id

FROM agent A1, agent A2

WHERE A1.salary > A2.salary

**Answer:**

**a) Query:**

SELECT relname, relpages, reltuples FROM pg\_class WHERE relname = 'agent' ;

Number of pages in agent: 8

Number of tuples in agent: 662

**b) Block Nested loop** is the only option because this query does not use equi-join.

**c) Block Nested loop**

M = 8 pages in agent

N = 8 pages in agent

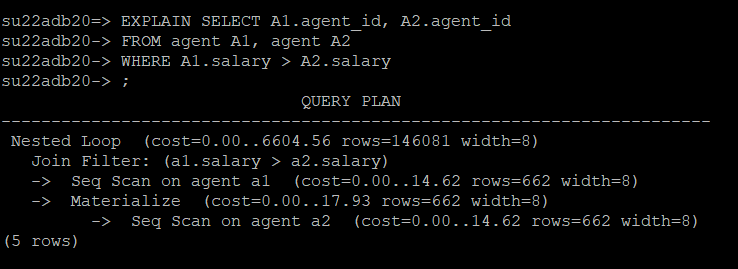
BP = 10 buffer pages

Cost = M + (M / (BP - 2)) \* N

= 8 + (8 / (10 - 2)) \* 8

= 16 I/Os

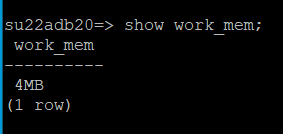
**d) postgresql uses a NESTED LOOP join.**



**e)** Here Postgre chose a nested loop. The cost was more expensive at 6604.56 I/Os

Of my choice, **block nested loop** was cheapest at 16 I/Os

f) **To find # of buffer pages available to join**



Number of buffer pages = show work\_mem = 4MB = 512 Buffer Pages

8KB 8KB

**NOTE: Make sure that you are running question 5 against the original agent table and not the copy you created for question 2 (this is why you were supposed to drop the copy of the table at the end of question 2!)**