Databases II Assignment 2 Report

Part A

A.Q1. Filter: (“position” = 1)
Sequential Scan on played_in (cost=0.00..1195.00 rows=11065 width=18)

A.Q2. Estimated Cost of the plan in A.Q1 = 1195.00

A.Q3. Index Cond: (“position” = 1)
→Bitmap Index Scan on in1 (cost=0.00..207.28 rows= 11065 width=0)
Recheck Cond: (“position” = 1)
Bitmap Heap Scan on played_in (cost=210.04..806.36 row= 11065 width-18)

A.Q4. Estimated Cost of the plan in A.Q3 = 806.36

A.Q5. Yes. It changed because we were first scanning without an index, therefore, we had to use iteration scan over the tuples to match the condition. After creating the index, we got to use the index scanning which reduced the cost of the query.

Part B

B.Q1. Filter: ((name)::text ~~ ‘%pele%’::text)
Sequential Scan on played_in (cost=0.00..1195.00 rows=102 width=18)

B.Q2. Estimated Cost of the plan in B.Q1 = 1195.00

B.Q3. Filter: ((name)::text ~~ ‘%pele%’::text)
Sequential Scan on played_in (cost=0.00..1195.00 rows=102 width=18)

B.Q4. Estimated Cost of the plan in B.Q3 = 1195.00

B.Q5. No, there’s no difference because we are searching with a “like” condition, not an “equality” condition, therefore, we had to search the table using sequential, iterative, scanning. This happened because indexing needs specific values to match tuples to indexes, but in the case of “like” condition, there’s no specific value it an map to, it has to search the whole table sequentially.

Part C

C.Q1. Filter: ((rating * ‘3’::double precision) > ‘20’::double precision)
Sequential Scan on cup_matches (cost=0.00..59.20 rows=893 width=24)

C.Q2. Estimated Cost of the plan in C.Q1 = 59.20

C.Q3. Filter: ((rating * '3'::double precision) > '20'::double precision)
Sequential Scan on cup_matches (cost=0.00..59.20 rows=893 width=24)

C.Q4. Estimated Cost of the plan in C.Q3 = 59.20

C.Q5. No, they are the same because we are searching with a "multiplication" condition , not an "equality" condition, therefore, we had to search the table using sequential, iterative, scanning.

Part D

D.Q1. Seq Scan on cup_matches (cost=0.00..45.80 rows=2680 width=24),
Hash (cost=45.80..45.80 rows=2680 width=24),
Seq Scan on played_in (cost=0.00..1047.60 rows=58960 width=18),
Hash Cond: (played_in.year = cup_matches.year),
Hash Join (cost=79.30..373417.19 rows=32195319 width=42)

Estimated cost : 373417.19

D.Q2. Seq Scan on cup_matches (cost=0.00..45.80 rows=2680 width=24),
Hash (cost=45.80..45.80 rows=2680 width=24),
Seq Scan on played_in (cost=0.00..1047.60 rows=58960 width=18),
Hash Cond: (played_in.year = cup_matches.year),
Hash Join (cost=79.30..373417.19 rows=32195319 width=42)

Estimated cost : 373417.19

D.Q3. No, the execution time is still the same because, the execution time of the Hash join is faster than than of the Index join, and that is why the execution time is still the same.

D.Q4. Estimated cost : 373417.19

D.Q5. No, the execution time is still the same because the execution time of the Hash join is faster than
t hat of the Index join even though we are using two indexes on two different Relations.

Part E

E.Q1. Seq Scan on cup_matches (cost=0.00..45.80 rows=2680 width=24),
Hash (cost=45.80..45.80 rows=2680 width=24),
Seq Scan on played_in (cost=0.00..1047.60 rows=58960 width=18),
Hash Cond: (played_in.mid = cup_matches.mid),
Hash Join (cost=79.30..1937.60 rows=58960 width=42)

Estimated cost : 1937.60

Join algorithm used: Hash Join

E.Q2. Index Scan using played_in_pkey on played_in (cost=0.29..2853.50 rows=58960 width=18)
Index Scan using cup_matches_pkey on cup_matches (cost=0.28..102.48 rows=2680 width=24)
Merge Cond: (cup_matches.mid = played_in.mid)
Merge Join (cost=0.57..3699.68 rows=58960 width=42)

Estimated cost : 3699.68

Join algorithm used: Merge Join

E.Q3. Index Cond: (mid = cup_matches.mid),
Index Scan using played_in_pkey on played_in (cost=0.29..1.70 rows=22 width=18),
Seq Scan on cup_matches (cost=0.00..45.80 rows=2680 width=24),
Nested Loop (cost=0.29..5184.40 rows=58960 width=42)

Estimated cost : 5184.40

Join algorithm used: Sequential join