Nome: Camylla Lima Dias RA: 222170052

Nome: Patricia Helena Soares Medeiros RA: 221200421

Lista de Exercícios (08/10)

*** Ao executar duas consultas na tabela cliente, onde numero_cliente é primary key, se obteve dois resultados de explain, conforme mostrado a seguir. Porque?**
Resp.: Obteve-se resultados diferentes devido ao filtro "=1", que não precisa de ordenação e por ser uma pesquisa mais objetiva, de índice único;
Utilizando o filtro ">1" precisamos de dados sequenciais, devido ao seu tamanho e a necessidade de ordenação.

1. **Qual a estratégia utilizada na consulta: "Recupere o nome de todos os professores do departamento de Biologia"**
   Resp.: A estratégia utilizada foi Seq Scan, onde a tabela é ordenada por ordem alfabética (leitura sequencial).

```
1  explain (analyze)
2  select * from cc6240.instructor
3  where dept_name = 'Biology'
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Seq Scan on instructor (cost=0.00..15.50 rows=2 width=154) (actual time=0.017..0.021 rows=2 loops=1)
2	[...] Filter: ((dept_name)::text = 'Biology'::text)
3	[...] Rows Removed by Filter: 48
4	Planning Time: 0.413 ms
5	Execution Time: 0.038 ms

2. **Recupere o nome dos cursos(disciplinas) que pertencem ao departamento de Ciência da Computação (Comp.Sci.) e que possuam 4 créditos. Qual estratégia utilizada para o join?**
   Resp.: A estratégia utilizada foi Nested Loop;

```
1  explain (analyze)
2  select title
3  from cc6240.course , cc6240.department
4  where department.dept_name='Comp.Sci.' and course.credits='4'
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Nested Loop (cost=0.15..13.59 rows=92 width=17) (actual time=0.107..0.147 rows=92 loops=1)
2	[...] -> Index Only Scan using department_pkey on department (cost=0.15..8.17 rows=1 width=0) (actual time=0.091..0.092 rows=1 loops=1)
3	[...] Index Cond: (dept_name = 'Comp.Sci.'::text)
4	[...] Heap Fetches: 1
5	[...] -> Seq Scan on course (cost=0.00..4.50 rows=92 width=17) (actual time=0.014..0.047 rows=92 loops=1)
6	[...] Filter: (credits = '4'::numeric)
7	[...] Rows Removed by Filter: 108
8	Planning Time: 0.467 ms
9	Execution Time: 0.179 ms

3. **Para o estudante com ID=24746 mostre todos os ids das disciplinas (courses) e seus respectivos nomes já registrados para ele. Quais foram as caracteristicas que levaram o Postgres a escolher o Nested Loop?**

Resp.: Essa operação foi utilizada pelo Otimizador já que é o mais adequado para pequenos dados e transações menores. Quando um pequeno conjunto de linhas de cada uma das tabelas são utilizadas na junção das duas tabelas, ou mais, e o método de acesso normalmente utilizado para recuperar as linhas da tabela interna é um índice.

```
1  explain (analyze)
2  select course_id
3  from cc6240.student, cc6240.department, cc6240.course
4  where student.id = '24746'
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Nested Loop (cost=0.28..1376.19 rows=108000 width=4) (actual time=0.157..0.726 rows=4000 loops=1)
2	[...] -> Seq Scan on department (cost=0.00..15.40 rows=540 width=0) (actual time=0.019..0.021 rows=20 loops=1)
3	[...] -> Materialize (cost=0.28..11.29 rows=200 width=4) (actual time=0.007..0.019 rows=200 loops=20)
4	[...] -> Nested Loop (cost=0.28..10.29 rows=200 width=4) (actual time=0.130..0.167 rows=200 loops=1)
5	[...] -> Index Only Scan using student_pkey on student (cost=0.28..4.29 rows=1 width=0) (actual time=0.115..0.115 rows=1 loops=1)
6	[...] Index Cond: (id = '24746'::text)
7	[...] Heap Fetches: 0
8	[...] -> Seq Scan on course (cost=0.00..4.00 rows=200 width=4) (actual time=0.013..0.034 rows=200 loops=1)
9	Planning Time: 0.975 ms
10	Execution Time: 0.856 ms

4. **Utilizando a consulta 3, qual o total de créditos feito pelo estudante? (use agregação)**

Resp.: 29 créditos

```
1  select sum (credits)
2  from cc6240.student inner join cc6240.course on student.dept_name = course.dept_name
3  where student.id = '24746'
```

Data Output | Explain | Messages | Notifications

	sum numeric
1	29

5. **Na consulta 3, adicione uma ordenação ascendente para course_id. O que mudou no explain?**
Resp.: O otimizador utilizou mais um Nested Loop do que na consulta 3 (que retornou Seq Scan) para realizar a ordenação de course_id;

```
1  explain (analyze)
2  select course_id
3  from cc6240.student, cc6240.department, cc6240.course
4  where student.id = '24746'
5  order by course_id
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Nested Loop (cost=0.42..1392.69 rows=108000 width=4) (actual time=0.196..0.768 rows=4000 loops=1)
2	[...] -> Nested Loop (cost=0.42..25.94 rows=200 width=4) (actual time=0.181..0.288 rows=200 loops=1)
3	[...] -> Index Only Scan using course_pkey on course (cost=0.14..19.14 rows=200 width=4) (actual time=0.148..0.212 rows=200 loops=1)
4	[...] Heap Fetches: 200
5	[...] -> Materialize (cost=0.28..4.30 rows=1 width=0) (actual time=0.000..0.000 rows=1 loops=200)
6	[...] -> Index Only Scan using student_pkey on student (cost=0.28..4.29 rows=1 width=0) (actual time=0.027..0.027 rows=1 loops=1)
7	[...] Index Cond: (id = '24746'::text)
8	[...] Heap Fetches: 0
9	[...] -> Materialize (cost=0.00..18.10 rows=540 width=0) (actual time=0.000..0.001 rows=20 loops=200)
10	[...] -> Seq Scan on department (cost=0.00..15.40 rows=540 width=0) (actual time=0.012..0.013 rows=20 loops=1)
11	Planning Time: 0.201 ms
12	Execution Time: 0.915 ms

6. **Similar a consulta 4, agrupe a soma de créditos para cada um dos estudantes e não só para o de id= 24746. O que mudou no explain, por que?**
Resp.: O que mudou é a utilização do Hash Join ao invés de Nested Loop, já que é uma pesquisa mais extensa do que a utilizada buscando por um único student.id

```
1  explain (analyze)
2  select sum (credits)
3  from cc6240.student inner join cc6240.course on student.dept_name = course.dept_name
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Aggregate (cost=345.82..345.83 rows=1 width=32) (actual time=5.136..5.138 rows=1 loops=1)
2	[...] -> Hash Join (cost=6.50..295.95 rows=19945 width=5) (actual time=0.104..3.300 rows=19945 loops=1)
3	[...] Hash Cond: ((student.dept_name)::text = (course.dept_name)::text)
4	[...] -> Seq Scan on student (cost=0.00..35.00 rows=2000 width=9) (actual time=0.017..0.196 rows=2000 loops=1)
5	[...] -> Hash (cost=4.00..4.00 rows=200 width=14) (actual time=0.077..0.077 rows=200 loops=1)
6	[...] Buckets: 1024 Batches: 1 Memory Usage: 18kB
7	[...] -> Seq Scan on course (cost=0.00..4.00 rows=200 width=14) (actual time=0.014..0.041 rows=200 loops=1)
8	Planning Time: 1.053 ms
9	Execution Time: 5.178 ms

7. **Retorne o nome de todos os estudantes que já cursaram alguma disciplina de Comp.Sci., ordene pelo nome de forma ascendente Aplique a clausula distinct, eliminando os duplicados. Há alguma diferença no explain com e sem duplicados?**
Resp.: A diferença e que com os duplicados o retorno é somente com o Sort, já sem duplicados temos o retorno com Unique e apenas em seguida o Sort.

Sem a clausula distinct

```
1  explain (analyze)
2  select student.name
3  from cc6240.student
4  where dept_name='Comp.Sci.'
5  order by student.name
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Sort (cost=43.65..43.92 rows=108 width=6) (actual time=0.485..0.489 rows=108 loops=1)
2	[...] Sort Key: name
3	[...] Sort Method: quicksort Memory: 30kB
4	[...] -> Seq Scan on student (cost=0.00..40.00 rows=108 width=6) (actual time=0.017..0.259 rows=108 loops=1)
5	[...] Filter: ((dept_name)::text = 'Comp.Sci.'::text)
6	[...] Rows Removed by Filter: 1892
7	Planning Time: 0.220 ms
8	Execution Time: 0.521 ms

Com a clausula distinct

```
1  explain (analyze)
2  select distinct student.name
3  from cc6240.student
4  where dept_name='Comp.Sci.'
5  order by student.name
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Unique (cost=43.65..44.19 rows=107 width=6) (actual time=0.472..0.495 rows=105 loops=1)
2	[...] -> Sort (cost=43.65..43.92 rows=108 width=6) (actual time=0.471..0.477 rows=108 loops=1)
3	[...] Sort Key: name
4	[...] Sort Method: quicksort Memory: 30kB
5	[...] -> Seq Scan on student (cost=0.00..40.00 rows=108 width=6) (actual time=0.018..0.221 rows=108 loops=1)
6	[...] Filter: ((dept_name)::text = 'Comp.Sci.'::text)
7	[...] Rows Removed by Filter: 1892
8	Planning Time: 0.148 ms
9	Execution Time: 0.522 ms

8. **Retorne todas as colunas e valores da tabela takes, ordenando por id ascendente. Observe o explain, qual método utiliza?**
Resp.: Utiliza o método Index Scan usando as chaves primárias (using takes_pkey)

```
1  explain (analyze)
2  select * from cc6240.takes
3  order by takes.id
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Index Scan using takes_pkey on takes (cost=0.29..2058.28 rows=30000 width=24) (actual time=0.016..10.776 rows=30000 loops=1)
2	Planning Time: 0.441 ms
3	Execution Time: 11.498 ms

9. **Delete a chave primária e o índice da tabela takes. Execute a consulta 10 novamente. Qual a diferença notada no explain?**
Resp.: A ordenação anterior era realizada sequencialmente com o Index Scan, desta vez temos o retorno com o Sort além do Seq Scan, aumentando o custo da pesquisa.

```
1  alter table cc6240.takes
2  drop constraint takes_pkey
```

Data Output | Explain | Messages | Notifications

```
ALTER TABLE

Query returned successfully in 55 msec.
```

*

```
1  explain (analyze)
2  select * from cc6240.takes
3  order by takes.id
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Sort (cost=2750.90..2825.90 rows=30000 width=24) (actual time=82.456..84.846 rows=30000 loops=1)
2	[...] Sort Key: id
3	[...] Sort Method: quicksort Memory: 3112kB
4	[...] -> Seq Scan on takes (cost=0.00..520.00 rows=30000 width=24) (actual time=0.011..1.789 rows=30000 loops=1)
5	Planning Time: 0.218 ms
6	Execution Time: 85.941 ms