E-fólio A | Folha de resolução para E-fólio

UNIDADE CURRICULAR: Sistemas de Gestão de Bases de Dados

CÓDIGO: 21103

DOCENTE: Luís Cavique

A preencher pelo estudante

NOME: Paulo Jorge Oliveira Gomes

N.º DE ESTUDANTE: 1500480

CURSO: Licenciatura em Engenharia Informática

DATA DE ENTREGA: 20/11/2022

TRABALHO / RESOLUÇÃO:

Pergunta 1:

Alínea a)

Para devolver o nome dos 10 principais clientes por ordem decrescente dos pagamentos efetuados e a soma de pagamentos de cada um efetuei a seguinte consulta:

```
Select concat(c.first_name,' ', c.last_name) as Nome, sum(p.amount) as Pagamentos

from sakila.payment as p, sakila.customer as c

where c.customer_id = p.customer_id

group by c.customer_id

order by sum(p.amount) desc

limit 10
```

Os resultados obtidos foram os que se seguem:

```
1 •  Select concat(c.first_name,' ', c.last_name) as Nome, sum(p.amount) as Pagamentos
2    from sakila.payment as p, sakila.customer as c
3    where c.customer_id = p.customer_id
4    group by c.customer_id
5    order by sum(p.amount) desc
6    limit 10
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Nome	Pagamentos
KARL SEAL	221.55
ELEANOR HUNT	216.54
CLARA SHAW	195.58
RHONDA KENNEDY	194.61
MARION SNYDER	194.61
TOMMY COLLAZO	186.62
WESLEY BULL	177.60
TIM CARY	175.61
MARCIA DEAN	175.58
ANA BRADLEY	174.66

Alínea b)

Sabemos que:

Customer c tem 599 linhas.

Payment p tem 16044 linhas.

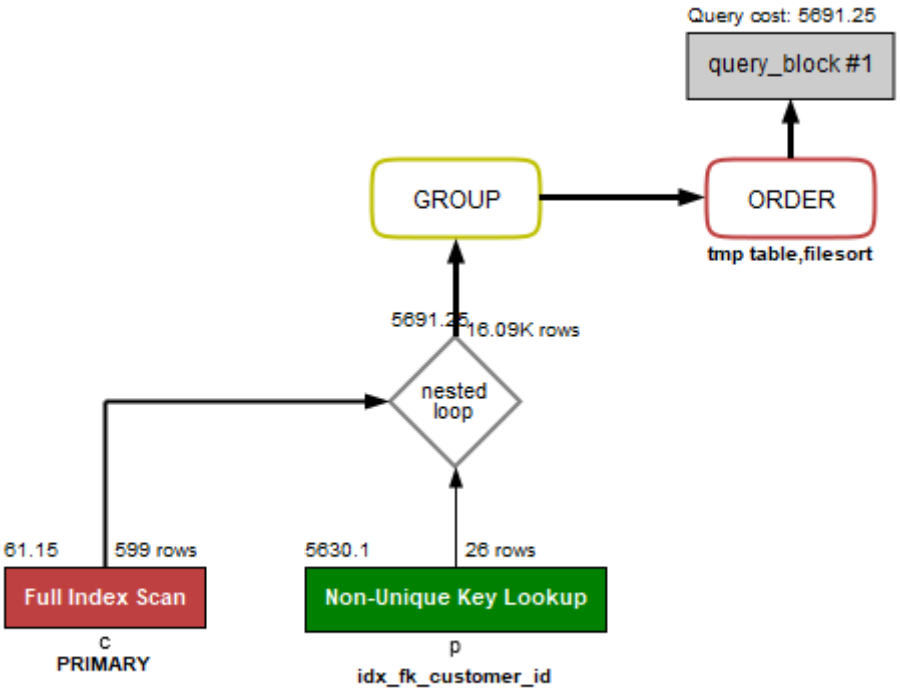
Payment tem um rácio de 26,78 linhas em relação a Customer (16044/599).

```
1 •  Explain Select concat(c.first_name,' ', c.last_name) as Nome, sum(p.amount) as Pagamentos
2    from sakila.payment as p, sakila.customer as c
3    where c.customer_id = p.customer_id
4    group by c.customer_id
5    order by sum(p.amount) desc
6    limit 10
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	c	NULL	index	PRIMARY,idx_fk_store_id,idx_fk_address_id,id...	PRIMARY	2	NULL	599	100.00	Using temporary; Using filesort
1	SIMPLE	p	NULL	ref	idx_fk_customer_id	idx_fk_customer_id	2	sakila.c.customer_id	26	100.00	NULL

Visualmente teremos:



No nested loop C-P resultam 16090 linhas, produto aproximado de 599 linhas de C com o rácio de 26,78 Customer/Payment.

1. Ciclo Externo C
2. Ciclo Interno P
3. Junção C e P
4. Fim Ciclo
5. Fim Ciclo

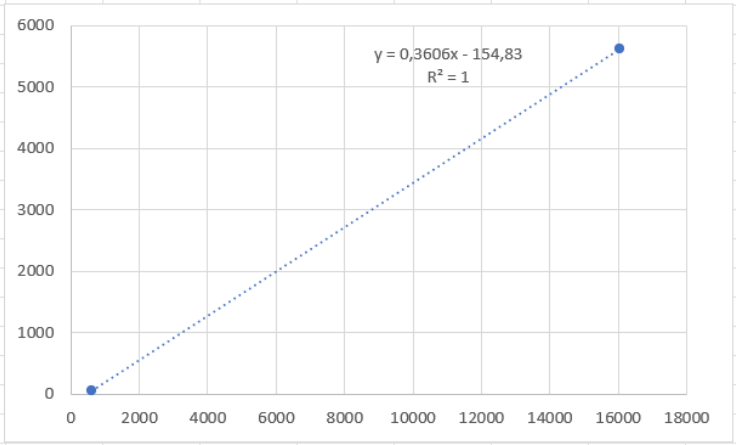
Sabendo que os custos são encontrados através de uma heurística, temos:

Custo(C) = 61,15 Custo(P)=5630,1

Custo(C)+Custo(P)=5691,25

A heurística reflete um comportamento linear entre o número de linhas e o custo:

Número Linhas	Custo
599	61,15
16044	5630,1



Pergunta 2:

Alínea a)

Analisemos as transações que nos são indicadas na tabela:

- T1 obtém uma Shared lock no recurso A

- T1 obtém uma Shared lock no recurso D

- T2 obtém uma eXclusive lock no recurso B

- T1 tenta obter uma Shared lock no recurso B mas vai ter de esperar que T2 liberte o recurso

- T3 obtém uma Shared Lock no recurso D

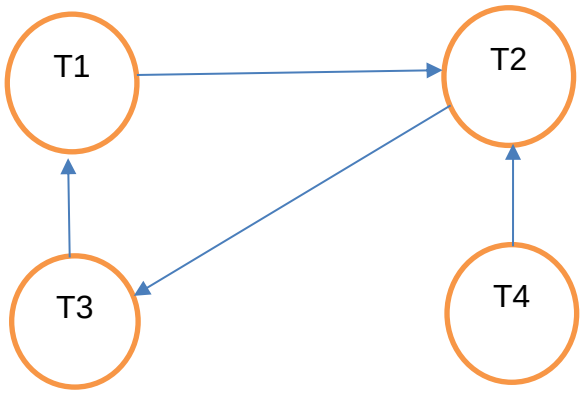
- T3 obtém uma Shared Lock no recurso C

- T2 tenta obter uma eXclusive Lock no recurso C mas vai ter de esperar que T3 liberte o recurso

- T4 tenta obter uma eXclusive lock no recurso B mas vai ter que esperar que T2 liberte o recurso

- T3 tenta obter uma eXclusive lock no recurso A, mas vai ter que esperar que T1 liberte o recurso

Podemos então criar o seguinte grafo wait-for:



Verificamos pelo grafo wait-for que existirá um deadlock uma vez que é possível verificar a existência de um ciclo no grafo wait-for entre as transações T1, T2 e T3.

Alínea b)

Para resolvermos o dealock que encontramos na alínea anterior um dos métodos que podemos utilizar consiste em escolher uma vitima entre as transações para realizarmos um rollback que possa eliminar o deadlock. A escolha da transação que deve sofrer rollback está relacionada com o custo de

cada transação de modo a escolhermos uma transação que incorra no menor custo possível para o sistema.

No caso que nos é apresentado verificamos que que o deadlock foi causado pela última tentativa da transação T3 adquirir um eXclusive lock no recurso A. O ideal seria escolher como vítima a transação T3 e efetuar um rollback parcial da transação T3 até ao momento anterior a esta tentativa de aquisição do eXclusive lock. De seguida a transação T3 iria esperar pela libertação dos recursos pelas outras transações e de seguida iria obter o eXclusive lock no recurso A de modo a conseguir completar-se também.

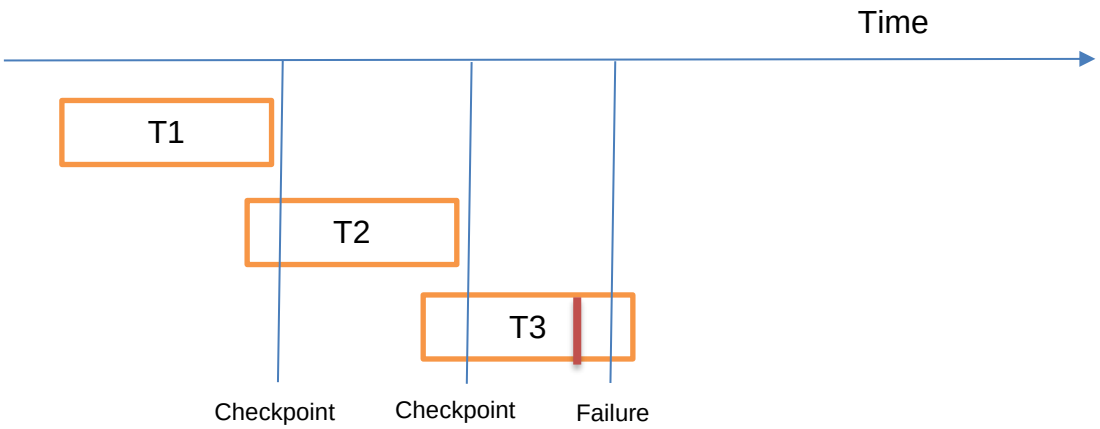
É, no entanto, necessário salientar que esta estratégia só iria funcionar caso o sistema estivesse preparado para suportar estes rollbacks parciais, bem como estar preparado para que as transações possam seguir a sua execução depois destes rollbacks.

Caso o sistema não estivesse preparado para isso, a opção seria de fazer rollback completo a uma das transições (neste caso a melhor opção seria a transação T2 uma vez que existem outras duas transações, T1 e T4, que estão á espera que esta transação se complete), sendo que depois das outras transações estarem completas, se seguiria a execução da transação T2.

Pergunta 3

Alínea a)

Em termos de diagrama temporal temos:



Alínea b)

Log	Redo-Phase	Undo-Phase
	undo-list = [ ]	Fim undo-phase
Start T1	undo-list = [T1]	
Read T1, A		
Write T1, A, 20, 30	Redo T1, A, 20, 30	
Start T2	undo-list = [T1, T2]	
Commit T1	undo-list = [T2]	
Checkpoint	Checkpoint	
Read T2, B		undo-list = [ ]
Start T3	undo-list = [T2, T3]	undo T3
Write T2, B, 10, 20	Redo T2, B, 10, 20	
Commit T2	Undo-list = [T3]	
Checkpoint	Checkpoint	
Read T3, C		
Write T3, C, 30, 40	Redo T3, C, 30, 40	undo-list = [T3]
System-Crash <------	----------	Início undo-phase
Redo T1, A, 20, 30		Registos acrescentados
Redo T2, B, 10, 20		Na redo-phase
Redo T3, C, 30, 40		E undo-phase
Undo T3		

Verificamos assim que teríamos o redo de T1 e T2 e o undo de T3.