e-Fólio A

U.C. 21103

Sistemas de Gestão de Bases de Dados

2025-2026

Resolução e Critérios de Correção

INSTRUÇÕES

1) O e-fólio é constituído por 3 perguntas. A cotação global é de 3 valores.

2) O e-fólio deve ser entregue num único ficheiro PDF, não zipado, com fundo branco, com perguntas numeradas e sem necessidade de rodar o texto para o ler. Cada pergunta com uma ou mais páginas, deve ser iniciada numa nova página. Penalização de 10% a 100%.

3) Não são aceites e-fólios manuscritos, i.e., tem penalização de 100%.

4) O nome do ficheiro: <nome estudante> + "eFolioA".

5) Na primeira página do e-fólio deve constar o nome completo do estudante bem como o seu número. Penalização de 10% a 100%.

6) Durante a realização do e-fólio, os estudantes devem concentrar-se na resolução do seu trabalho individual, não sendo permitida a colocação de perguntas ao professor ou entre colegas.

7) Nesta avaliação, não deve utilizar ferramentas de IA generativa, como o ChatGPT.

8) A interpretação das perguntas também faz parte da sua resolução, se encontrar alguma ambiguidade deve indicar claramente como foi resolvida.

9) A legibilidade, a objetividade e a clareza nas respostas serão valorizadas, pelo que, a falta destas qualidades será penalizada.

10) Critérios de correção gerais: todas as respostas devem ser justificadas, incluir imagens e exemplos com vista a clarificar os argumentos expostos. Devem ser utilizadas referências das páginas da bibliografia adotada e recomendada.

Vetor Cotações:
1ab 2ab 3ab pergunta
10 10 10 décimas

Critérios de correção gerais: todas as respostas devem ser justificadas, incluir imagens e exemplos com vista a clarificar os argumentos expostos. Devem ser utilizadas referências das páginas da bibliografia adotada e recomendada.

1.a) Escreva em álgebra relacional e em SQL: Encontrar os clientes que alugaram todos os filmes da categoria *"Action"*.

Resposta:
A base de dados Sakila tem as seguintes tabelas:
film(film_id -> title, ...)
category(category_id -> name, ...)

film_category(film_id, category_id ->...)

customer(customer_id -> …)
inventory(inventory_id -> film_id, …)
rental(rental_id -> rental_date, inventory_id, customer_id, …)

Em álgebra relacional vamos usar ϭ para seleção, Π para projeção e |><| para junção.

1. Selecionar a tabela de todos os filmes de ação:
filmesAcao = Π_{film_id} [$\sigma_{name=Action}$ (category |><| film_category)]

2. Selecionar a tabela (cliente, filme) com os filmes Ação alugados pelos clientes:
filmesAlugados = $\Pi_{customer\_id,\ film\_id}$ (inventory |><| rental |><| filmesAcao)

3. Encontrar os clientes que alugaram todos os filmes da categoria “Action”:
clientesTodosFilmesAcao = filmesAlugadosAcao ÷ filmesAcao

Em SQL teremos:

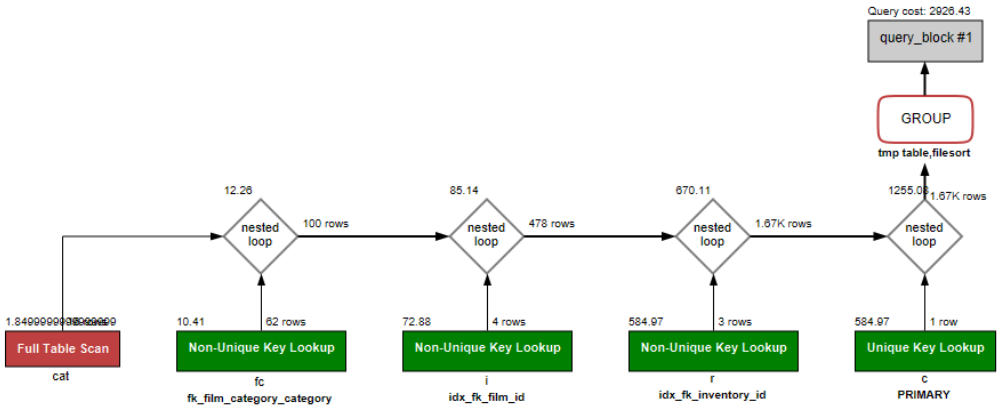
```
use sakila;
SELECT c.customer_id, c.first_name, c.last_name
FROM customer c
JOIN rental r ON r.customer_id = c.customer_id
JOIN inventory i ON i.inventory_id = r.inventory_id
JOIN film_category fc ON fc.film_id = i.film_id
JOIN category cat ON cat.category_id = fc.category_id
WHERE cat.name = 'Action'
GROUP BY c.customer_id, c.first_name, c.last_name
HAVING COUNT(DISTINCT i.film_id) =
(
  SELECT COUNT(*)
  FROM film_category fc2
  JOIN category cat2 ON cat2.category_id = fc2.category_id
  WHERE cat2.name = 'Action'
);
```

\ Não existe nenhum cliente que tenha visto todos os clientes de ação.

1.b) Mostre o plano de execução disponível no MySQL idêntico ao da figura. Recolha os dados da opção gráfica Execution Plan. Analise e comente os resultados.

Resposta:

Foi realizada a seguinte sequência de junções:
((((category |><| film_category) |><| inventory) |><| rental)|><| customer)



Detalhe de cada etapa:

Etapa	Ação	Porquê?	Linhas aproximadas	Custo Estimado	Custo Acumulado
1	Full Table Scan em category	A tabela category=Action	1	1,84	1,84
2	Non-Unique Key Lookup em film_category (índice: fk_film_category_category)	Encontrar todos os filmes da categoria *Action*	62	10,41	12,25
3	Non-Unique Key Lookup em inventory (índice: idx_fk_film_id)	Encontrar todas as cópias dos filmes da categoria Action	100	72,88	85,13
4	Non-Unique Key Lookup em rental (índice: idx_fk_inventory_id)	Localizar alugueres feitos dessas cópias	478	584,97	670,1
5	Unique Key Lookup em customer (índice PRIMARY)	Encontra o cliente de cada aluguer	1670	584,97	1255,07
6	GROUP BY (tmp table + filesort)	Necessário agrupar cliente × filmes alugados	1670	1673,36	2928,43
7	Query Block final	Resultado consolidado	0	2928,43	

Critérios de correção:
alínea a) 4 décimas, álgebra relacional, consulta SQL
alínea b) 6 décimas, plano de execução e explicação dos custos
- erros, omissões, redundâncias ou indentação desadequada: -20% a -100%

2) (1 valor) Relativo a Concorrência. Considere duas transações concorrentes num sistema bancário:

T1	T2
READ(A); A=A-100; WRITE(A);	
	READ(A); A=A*1.1;
READ(B); B=B+100; WRITE(B)	
	WRITE(A); READ(B); B=B*1.1; WRITE(B)

2.a) Desenhe um sequenciamento (schedule) aplicando o protocolo 2PL básico, indicando quando cada transação adquire e liberta locks. O sequenciamento é serializável?

Resposta: Existe possibilidade do sequenciamento T1->T2

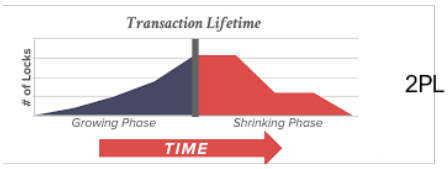
T1	T2	comentários 2PL
LOCK-X(A)		
LOCK-X(B)		
READ(A)		
A=A-100		
WRITE(A)		
UNLOCK(A)		
	LOCK-X(A)	T2 pode avançar
	READ(A)	
	A=A*1.1	
READ(B)		
B=B+100		
WRITE(B)		
UNLOCK(B)		
COMMIT		
	LOCK-X(B)	
	WRITE(A)	
	READ(B)	
	B=B*1.1	
	WRITE(B)	
	UNLOCK(A)	
	UNLOCK(B)	
	COMMIT	

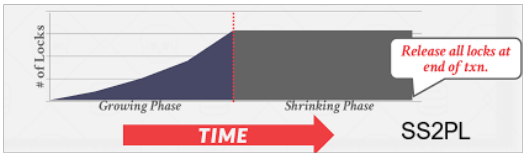
2.b) Repita usando SS2PL (strong Strict 2PL). Que diferença observa no momento de libertação dos locks?

Resposta: Existe também possibilidade do sequenciamento T1->T2

T1	T2	comentários SS2PL
LOCK-X(A)		
LOCK-X(B)		
READ(A)		
A=A-100		
WRITE(A)		
	READ(A)	espera pelo unlock
	A=A*1.1	T2 wait
READ(B)		T2 wait
B=B+100		T2 wait
WRITE(B)		T2 wait
COMMIT		T2 wait
UNLOCK(A)		T2 wait
UNLOCK(B)		T2 wait
	LOCK-X(A)	T2 pode avançar
	READ(A)	
	A=A*1.1	
	LOCK-X(B)	
	WRITE(A)	
	READ(B)	
	B=B*1.1	
	WRITE(B)	
	COMMIT	
	UNLOCK(A)	
	UNLOCK(B)	

A diferença da libertação dos locks entre 2PL e SS2PL reside na forma como se processam os Commits. O 2PL o Commit é depois dos Unlocks. No SS2PL primeiro realiza-se o Commit. Graficamente teremos:



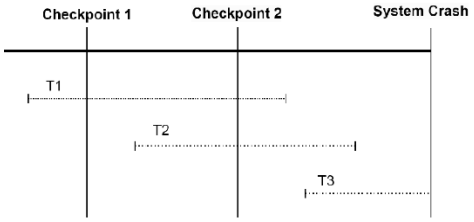


Critérios de correção:
alínea 2.a) 4 décimas no 2PL
alínea 2.b) 6 décimas no SS2PL
- erros, omissões, redundâncias ou indentação desadequada: -20% a -100%

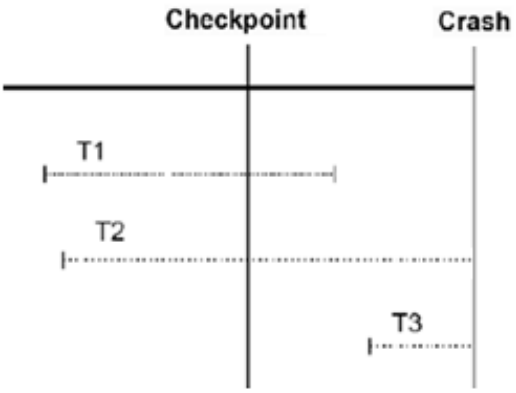
3) (1 valor) Relativo ao tema da Recuperação. Considere o seguinte log de transações:

Log	Redo Phase	Undo Phase
<T1, start>; <T1, A,100,150>; <T2, start>; <T2, B,200,250>; <T1, C,300,350>; <checkpoint T1, T2>; <T3, start>; <T1, commit>; <T3, D,400,450>; <T2, E,500,550>; <CRASH>.		

3.a) Represente as transações e os checkpoints na linha do tempo (diagrama temporal como o da figura). Quais os valores de A,B,C,D,E em disco no momento do Crash?



Resposta:



\Redo T1, Undo T2 e T3

Valores em Disco:
A=150; C=350; B = 250 gravados com o checkpoint
D = 400; E = 500

3.b) Identifique quais transações necessitam de UNDO e quais necessitam de REDO. Justifique. Acrescente os registos na recuperação. Complete a tabela e justifique a resposta.

Resposta:

Redo T1, Undo T2 e T3.

Log	Redo Phase		Undo Phase
<T1, start>;	undo-list =[T1]		
<T1, A,100,150>;	redo T1		undo-list =[]
<T2, start>;	undo-list =[T1,T2]		undo T2
<T2, B,200,250>;	redo T2		
<T1, C,300,350>;	redo T1		
<checkpoint T1, T2>;			undo-list =[T2]
<T3, start>;	undo-list =[T1,T2,T3]		undo T3
<T1, commit>;	undo-list =[T2,T3]		
<T3, D,400,450>;	redo T3		
<T2, E,500,550>;	redo T2		undo-list =[T2,T3]
<CRASH>.			Undo-Phase start
redo T1			

Critérios de correção:
alínea 3.a) 4 décimas, diagrama temporal
alínea 3.b) 6 décimas Redo e Undo Phases
- erros, omissões, redundâncias ou formatação desadequada: -20% a -100%