Infraestrutura para Gestão de Dados
Trabalho 2

Nome: Paulo Ricardo Carnovale Cardoso / Rodrigo Rosa

Limpeza de tabelas

```
BEGIN
  FOR r IN (SELECT object_name, object_type
          FROM user_objects
         WHERE object_type IN ('TABLE', 'VIEW', 'SEQUENCE', 'SYNONYM', 'PROCEDURE', 'FUNCTION',
'PACKAGE', 'TRIGGER', 'INDEX', 'CLUSTER'))
  LOOP
    BEGIN
      IF r.object_type = 'TABLE' THEN
        EXECUTE IMMEDIATE 'DROP ' || r.object_type || ' "' || r.object_name || '" CASCADE CONSTRAINTS';
      ELSIF r.object_type = 'CLUSTER' THEN
        EXECUTE IMMEDIATE 'DROP ' || r.object_type || ' "' || r.object_name || '" INCLUDING TABLES CASCADE
CONSTRAINTS';
      ELSE
        EXECUTE IMMEDIATE 'DROP ' || r.object_type || ' "' || r.object_name || '"';
      END IF;
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Erro ao excluir ' || r.object_type || ' ' || r.object_name || ': ' || SQLERRM);
    END;
  END LOOP;
END;
```

Ordem de Execução	T1 (RH)	T2 (Vendas)	T3 (Gerência)	T4 (administrador)
1				CREATE TABLE SALAS (sala_id INT PRIMARY KEY, nome_sala VARCHAR(100) NOT NULL, capacidade INT NOT NULL); CREATE TABLE RESERVAS (reserva_id INT PRIMARY KEY, sala_id INT NOT NULL, data_reserva TIMESTAMP NOT NULL, usuario_id INT NOT NULL, FOREIGN KEY (sala_id) REFERENCES SALAS(sala_id));
Resultado				Table SALAS criado. Table RESERVAS criado.
Observação				Administrador de BD cria as tabelas de sala e reserva
2				INSERT INTO SALAS (sala_id, nome_sala, capacidade) VALUES (1, 'Sala de Reunião A', 10); INSERT INTO SALAS (sala_id, nome_sala, capacidade) VALUES (2, 'Sala de Conferência B', 20); INSERT INTO SALAS (sala_id,

				nome_sala, capacidade) VALUES (3, 'Auditório C', 50); COMMIT;
Resultado				1 linha inserido. 1 linha inserido. 1 linha inserido. Commit concluído
Observação				BD insere nas tabelas as informações das salas disponíveis para reserva e suas características.
3	SET TRANSACTION READ ONLY; SELECT * FROM RESERVAS WHERE sala_id = 1 AND data_reserva BETWEEN TO_TIMESTAMP('2024-10-01 09:00:00', 'YYYY-MM-DD HH24:MI:SS') AND TO_TIMESTAMP('2024-10-01 10:00:00', 'YYYY-MM-DD HH24:MI:SS'); COMMIT;			
Resultado	Transaction READ bem-sucedido. nenhuma linha selecionada Commit concluído.			
Observação	RH quer ver a disponibilidade da sala de reunião A para			

	o dia 01/10/2024 às 9h para realização de uma entrevista que levará 1h. Pra isso ele usa um sistema de apenas consulta. Como resultado ele vê que a sala está disponível.			
4	SET TRANSACTION READ WRITE; INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id) VALUES (1001, 1, TO_TIMESTAMP('2024-10-01 09:00:00', 'YYYY-MM-DD HH24:MI:SS'), 101); COMMIT;			
Resultado	Transaction READ bem-sucedido. 1 linha inserido. Commit concluído.			
Observação	RH confirmou com o candidato e quer reservar a sala de reunião A para o dia 01/10/2024 das 9h às 10h. Reserva inserida com sucesso.			
5		SET TRANSACTION READ ONLY; -- SELECT * FROM RESERVAS WHERE sala_id = 1 AND data_reserva BETWEEN TO_TIMESTAMP('2024-10-01 10:15:00', 'YYYY-MM-DD HH24:MI:SS') AND TO_TIMESTAMP('2024-10-01 12:00:00', 'YYYY-MM-DD HH24:MI:SS');		
Resultado		Transaction READ bem-sucedido.		

		nenhuma linha selecionada		
Observação		A equipe de vendas precisa fazer uma reunião no mesmo dia, 01/10/2024, das 10h15 às 12h e faz uma consulta da disponibilidade. Tem como resposta que está disponível para reserva.		
6				SET TRANSACTION READ WRITE; LOCK TABLE RESERVAS IN EXCLUSIVE MODE;
Resultado				Transaction READ bem-sucedido. Lock bem-sucedido.
Observação				Administrador de BD, paralelamente, bloqueia as reservas para alterar o horário de início da reserva do RH, pois a sala estará indisponível até as 10h devido a manutenção.
7	SET TRANSACTION READ WRITE; UPDATE RESERVAS SET data_reserva = TO_TIMESTAMP('2024-10-01 14:00:00', 'YYYY-MM-DD HH24:MI:SS') WHERE reserva_id = 1001; COMMIT;			

Resultado	Transaction READ bem—sucedido.			
Observação	Estado de lock pois admin trancou a tabela de reservas.			
8				COMMIT;
Resultado				Commit concluído.
Observação				Administrador libera reservas e a transação do terminal 1 também é concluída.
9	SET TRANSACTION ISOLATION LEVEL READ COMMITTED; SELECT data_reserva FROM RESERVAS WHERE reserva_id = 1001;			
Resultado	Transaction ISOLATION bem-sucedido. DATA_RESERVA ---------- ---------- 01/10/24 14:00:00,000000000			
Observação	Leitura inicial do horário da reserva 1001 (sem bloqueio explícito)			
10			SET TRANSACTION ISOLATION LEVEL SERIALIZABLE; UPDATE RESERVAS SET data_reserva = TO_TIMESTAMP('2024-10-01 15:00:00', 'YYYY-MM-DD HH24:MI:SS') WHERE reserva_id = 1001; COMMIT;	
Resultado			Transaction ISOLATION bem-sucedido. 1 linha atualizado.	

			Commit concluído.	
Observação			Gerência altera o horário da reserva 1001 para as 15h e comita a alteração	
11	SELECT data_reserva FROM RESERVAS WHERE reserva_id = 1001;			
Resultado	01/10/24 15:00:00,000000000			
Observação	Leitura não repetível			
12		SET TRANSACTION ISOLATION LEVEL READ COMMITTED; SELECT * FROM RESERVAS WHERE sala_id = 1; LOCK TABLE RESERVAS IN SHARE MODE;		
Resultado		Transaction ISOLATION bem-sucedido. RESERVA_ID SALA_ID DATA_RESERVA USUARIO_ID ---------- ---------- ---------- ---------- 1001 1 01/10/24 15:00:00,000000000 101 Lock bem-sucedido.		
Observação		Time de Vendas faz uma consulta inicial para ver todas as reservas da Sala de Reunião A e bloqueia a tabela RESERVAS em modo compartilhado para garantir que ninguém modifique as reservas		
13			SET TRANSACTION ISOLATION LEVEL READ COMMITTED;	

			INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id) VALUES (1002, 1, TO_TIMESTAMP('2024-10-01 14:00:00', 'YYYY-MM-DD HH24:MI:SS'), 102);	
Resultado			Transaction ISOLATION bem—sucedido.	
Observação			Gerência tenta inserir uma nova reserva para a Sala de Reunião A e tenta comitar. Fica bloqueado o insert.	
14		SELECT * FROM RESERVAS WHERE sala_id = 1;		
Resultado		RESERVA_ID SALA_ID DATA_RESERVA USUARIO_ID ---------- ---------- ---------- ---------- 1001 1 01/10/24 15:00:00,000000000 101		
Observação		O Time de Vendas tenta consultar as reservas novamente, mas não verá a nova reserva, pois a inserção da Gerência foi bloqueada e vê o mesmo conjunto de reservas, sem novas inserções, porque sua transação ainda está aberta e a tabela RESERVAS está bloqueada.		
15		COMMIT;		
Resultado		Commit concluído.		
Observação		finaliza transação e libera o bloqueio sobre a tabela RESERVAS, o que permite outras		

		transações modifiquem ou insiram novas reservas		
16			COMMIT;	
Resultado			Commit concluído.	
Observação			Terminal é liberado e a reserva efetuada.	
17		SELECT * FROM RESERVAS WHERE sala_id = 1;		
Resultado		RESERVA_ID SALA_ID DATA_RESERVA USUARIO_ID ---------- ---------- ------- ---------- --- 1002 1 01/10/24 14:00:00,000000000 102 1001 1 01/10/24 15:00:00,000000000 101		
Observação		É possível ver a nova reserva inserida pela Gerência, devido fenômeno de Leitura de Fantasmas, pois a nova linha não estava presente nas leituras anteriores da mesma consulta		
18		SET TRANSACTION ISOLATION LEVEL SERIALIZABLE; LOCK TABLE RESERVAS IN EXCLUSIVE MODE;		
Resultado		Commit concluído. Transaction ISOLATION bem-sucedido. Lock bem-sucedido.		
Observação		Time de Vendas bloqueia a tabela RESERVAS em modo exclusivo para alterações, a		

		transação não é finalizada aqui, então a tabela RESERVAS permanece bloqueada		
19			SET TRANSACTION ISOLATION LEVEL SERIALIZABLE; LOCK TABLE SALAS IN EXCLUSIVE MODE;	
Resultado			Transaction ISOLATION bem-sucedido. Lock bem-sucedido.	
Observação			Gerência bloqueia a tabela SALAS em modo exclusivo para alteração. A transação não é finalizada aqui, então a tabela SALAS permanece bloqueada	
20		UPDATE SALAS SET capacidade = 40 WHERE sala_id = 1;		
Resultado		Bloqueio		
Observação		Vendas tenta acessar a tabela SALAS. O Time de Vendas fica bloqueado aguardando a Gerência liberar a tabela SALAS		
21			UPDATE RESERVAS SET data_reserva = TO_TIMESTAMP('2024-10-01 16:00:00', 'YYYY-MM-DD HH24:MI:SS') WHERE reserva_id = 1001;	
Resultado			Bloqueado	
Observação			Gerência tenta acessar a tabela RESERVAS fica bloqueada aguardando o Time de Vendas liberar a tabela RESERVAS	
22				

Resultado		Erro a partir da linha : 2 no comando - UPDATE SALAS SET capacidade = 40 WHERE sala_id = 1 Erro na Linha de Comandos : 3 Coluna : 5 Relatório de erros - Erro de SQL: ORA-00060: conflito detectado ao aguardar recurso 00060. 00000 - "deadlock detected while waiting for resource" *Cause: Transactions deadlocked one another while waiting for resources. *Action: Look at the trace file to see the transactions and resources involved. Retry if necessary.		
Observação		Deadlock detectado. SQL derruba essa transação (cancela) e permanece o bloqueio no terminal 3		
23		COMMIT;		
Resultado		Commit concluído.		
Observação		A capacidade da sala é atualizada		
24			COMMIT;	
Resultado			1 linha atualizado. Commit concluído.	
Observação			A operação então é desbloqueada e a reserva atualizada.	

Sequência

1. TERMINAL 4: O Administrador de BD cria as tabelas SALAS e RESERVAS para gerenciar as reservas de salas.

2. TERMINAL 4: O Administrador de BD insere informações sobre as salas disponíveis (ex.: Sala de Reunião A, Sala de Conferência B).

3. TERMINAL 1 (RH): O RH verifica a disponibilidade da Sala de Reunião A para uma entrevista no dia 01/10/2024 das 9h às 10h usando uma consulta de leitura.

Início de Non-Repeatable Read

4. TERMINAL 1 (RH): O RH reserva a Sala de Reunião A das 9h às 10h no dia 01/10/2024, e a transação é confirmada.

5. TERMINAL 2 (Vendas): O Time de Vendas consulta a disponibilidade da Sala de Reunião A para uma reunião das 10h15 às 12h no mesmo dia.

7. TERMINAL 1 (RH): O RH tenta modificar a reserva para um novo horário (14h) devido a mudanças na agenda, mas é impedido pelo bloqueio anterior.

9. TERMINAL 1 (RH): O RH verifica novamente a reserva modificada e confirma que o novo horário é 14h.

10. TERMINAL 3 (Gerência): A Gerência altera o horário da reserva para as 15h e finaliza a transação.

11. TERMINAL 1 (RH): O RH verifica novamente o horário da reserva e vê o horário atualizado para as 15h, caracterizando uma leitura não repetível.

Fim de Non-Repeatable Read

12. TERMINAL 2 (Vendas): O Time de Vendas consulta as reservas da Sala de Reunião A e bloqueia a tabela RESERVAS em modo compartilhado.

Início de Phantom Read

13. TERMINAL 3 (Gerência): A Gerência tenta inserir uma nova reserva na Sala de Reunião A, mas é bloqueada pela transação do Time de Vendas.

14. TERMINAL 2 (Vendas): O Time de Vendas tenta ler as reservas novamente, mas não vê a nova reserva devido ao bloqueio da Gerência.

15. TERMINAL 2 (Vendas): O Time de Vendas finaliza sua transação e libera o bloqueio sobre a tabela RESERVAS.

16. TERMINAL 3 (Gerência): A Gerência finaliza a transação e a nova reserva é inserida com sucesso.

17. TERMINAL 2 (Vendas): O Time de Vendas consulta novamente as reservas e agora vê a nova reserva, evidenciando uma leitura de fantasmas.

Fim de Phantom Read

18. TERMINAL 2 (Vendas): O Time de Vendas bloqueia a tabela RESERVAS em modo exclusivo para realizar alterações.

Início de Deadlock

19. TERMINAL 3 (Gerência): A Gerência bloqueia a tabela SALAS em modo exclusivo para alterar a capacidade da sala.

20. TERMINAL 2 (Vendas): O Time de Vendas tenta acessar a tabela SALAS enquanto está bloqueada pela Gerência, ficando aguardando.

21. TERMINAL 3 (Gerência): A Gerência tenta acessar a tabela RESERVAS enquanto está bloqueada pelo Time de Vendas, ficando aguardando.

22. TERMINAL 3 (Gerência): O Oracle detecta um deadlock entre as transações e cancela a transação da Gerência.

23. TERMINAL 2 (Vendas): O Time de Vendas finaliza a transação e a capacidade da sala é atualizada.

24. TERMINAL 3 (Gerência): A transação bloqueada é resolvida e a reserva é atualizada com sucesso.

Fim de Deadlock