

# Infraestrutura para Gestão de Dados

## Trabalho T2

Alunos: **Paulo Carnovale** e **Rodrigo Oliveira Rosa**

**2024/2**

### Criação e população das tabelas

#### create

```
--- create table salas
CREATE TABLE SALAS (
    sala_id INT PRIMARY KEY,
    nome_sala VARCHAR(100) NOT NULL,
    capacidade INT NOT NULL
);
--- create table reservas
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)
);
```

#### insert

```
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);

-- Inserir algumas reservas
INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id) VALUES (1, 1, TO_TIMESTAMP('2024-10-23 10:00:00', 'YYYY-MM-DD HH24:MI:SS'), 101);
INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id) VALUES (2, 2, TO_TIMESTAMP('2024-10-23 12:00:00', 'YYYY-MM-DD HH24:MI:SS'), 102);
INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id) VALUES (3, 3, TO_TIMESTAMP('2024-10-24 14:00:00', 'YYYY-MM-DD HH24:MI:SS'), 103);

COMMIT;
```

### Relatório de execução dos testes

A seguir temos o roteiro de execução, intercalando as operações entre as sessões para garantir que os problemas de concorrência sejam gerados. Vamos listar as sessões e as queries na ordem em que devem ser executadas para conseguirmos produzir **leitura não repetível**, **leitura fantasma** e **deadlock**.

Aqui está a versão preenchida com o objetivo:

## Problema de Dead-Lock

### Cenário com Quatro Sessões

#### Objetivo:

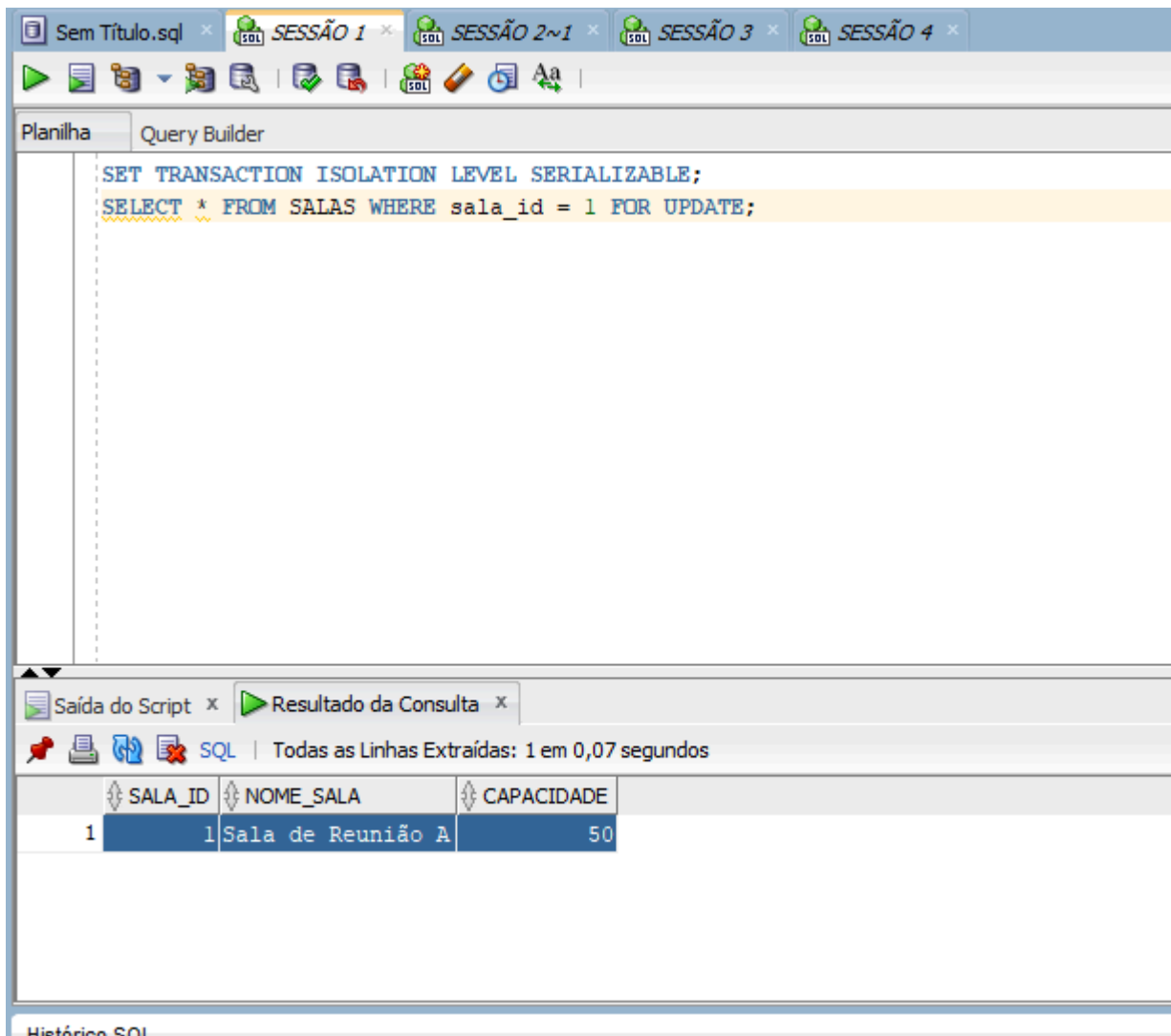
1. **Sessão 1:** Bloquear a sala 1 para atualização e manter a transação aberta até que o deadlock ocorra.
2. **Sessão 2:** Fazer leitura inicial e tentar inserir uma reserva na sala 1 para causar conflito com a Sessão 1.
3. **Sessão 3:** Inserir uma nova reserva na sala 2, sem conflitos, para simular transações concorrentes.
4. **Sessão 4:** Verificar o impacto do deadlock ao tentar atualizar a capacidade da sala 1, já bloqueada na Sessão 1.
5. **Sessão 1:** Tentar concluir a transação, demonstrando o deadlock e a resolução pelo sistema.

#### Sequência de Operações

1. Sessão 1: Inicia transação serializável e bloqueia sala 1 para atualização

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;  
SELECT * FROM SALAS WHERE sala_id = 1 FOR UPDATE;
```

- A operação bloqueia a sala 1 para modificações, prevenindo que outras transações a modifiquem até o commit.



## 2. Sessão 2: Leitura inicial da reserva da sala 1

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  
SELECT * FROM RESERVAS WHERE sala_id = 1;
```

- A leitura não interfere no bloqueio da Sessão 1 porque o nível de isolamento "read committed" permite leitura de dados confirmados.

The screenshot shows a SQL IDE interface. The top window, titled 'Query Builder', contains the following SQL code:

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  
SELECT * FROM RESERVAS WHERE sala_id = 1;
```

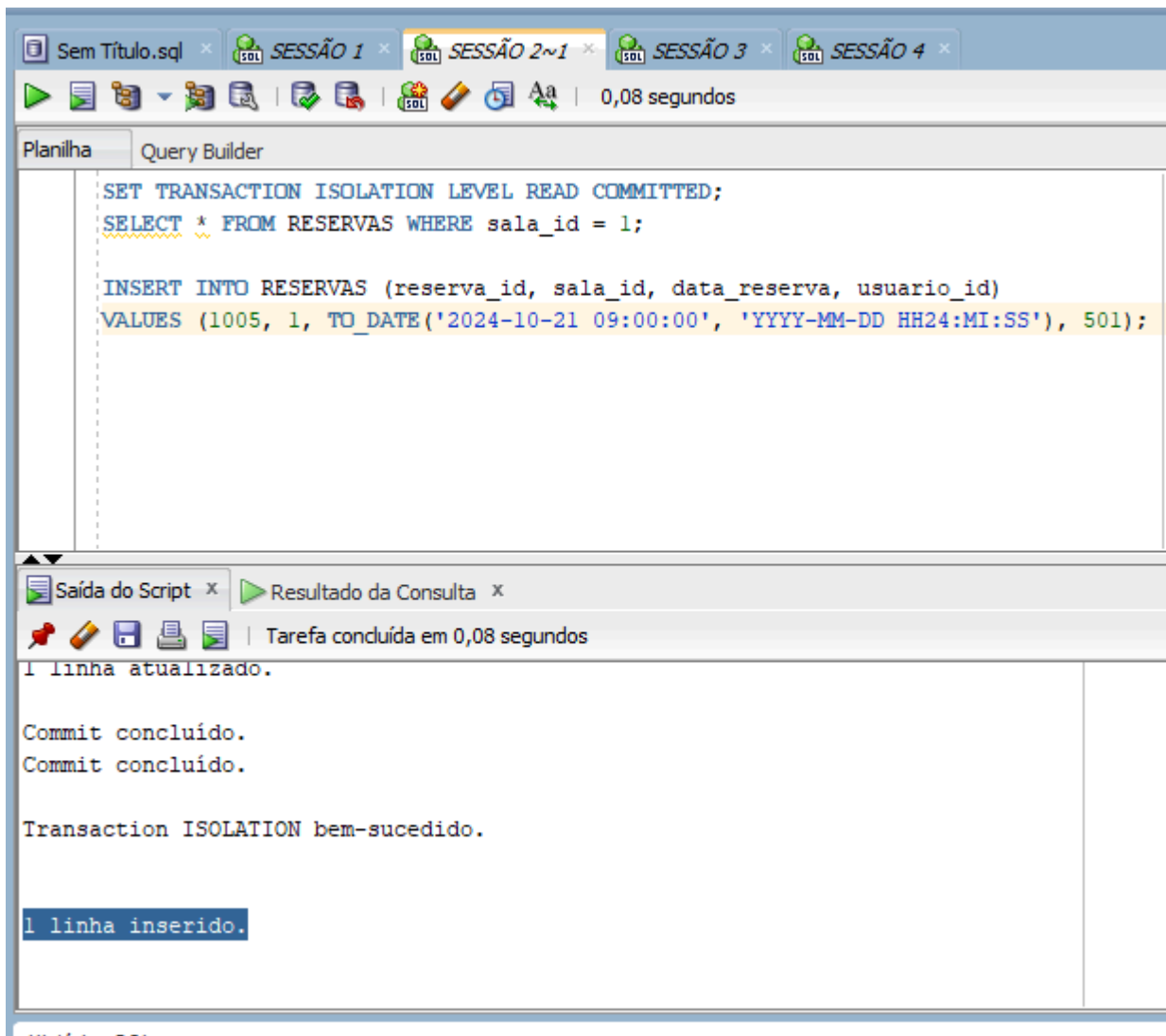
The bottom window, titled 'Resultado da Consulta', displays the results of the query. It shows a table with 5 rows and 4 columns: RESERVA\_ID, SALA\_ID, DATA\_RESERVA, and USUARIO\_ID. The first row is highlighted in blue.

	RESERVA_ID	SALA_ID	DATA_RESERVA	USUARIO_ID
1	1	1	23/10/24 10:00:00,0000000000	101
2	4	1	23/10/24 15:00:00,0000000000	104
3	1004	1	22/10/24 14:00:00,0000000000	502
4	1001	1	21/10/24 12:00:00,0000000000	502
5	1003	1	22/10/24 12:00:00,0000000000	501

### 3. Sessão 2: Insere reserva para sala 1

```
INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id)  
VALUES (1001, 1, TO_DATE('2024-10-21 09:00:00', 'YYYY-MM-DD HH24:MI:SS'), 501);
```

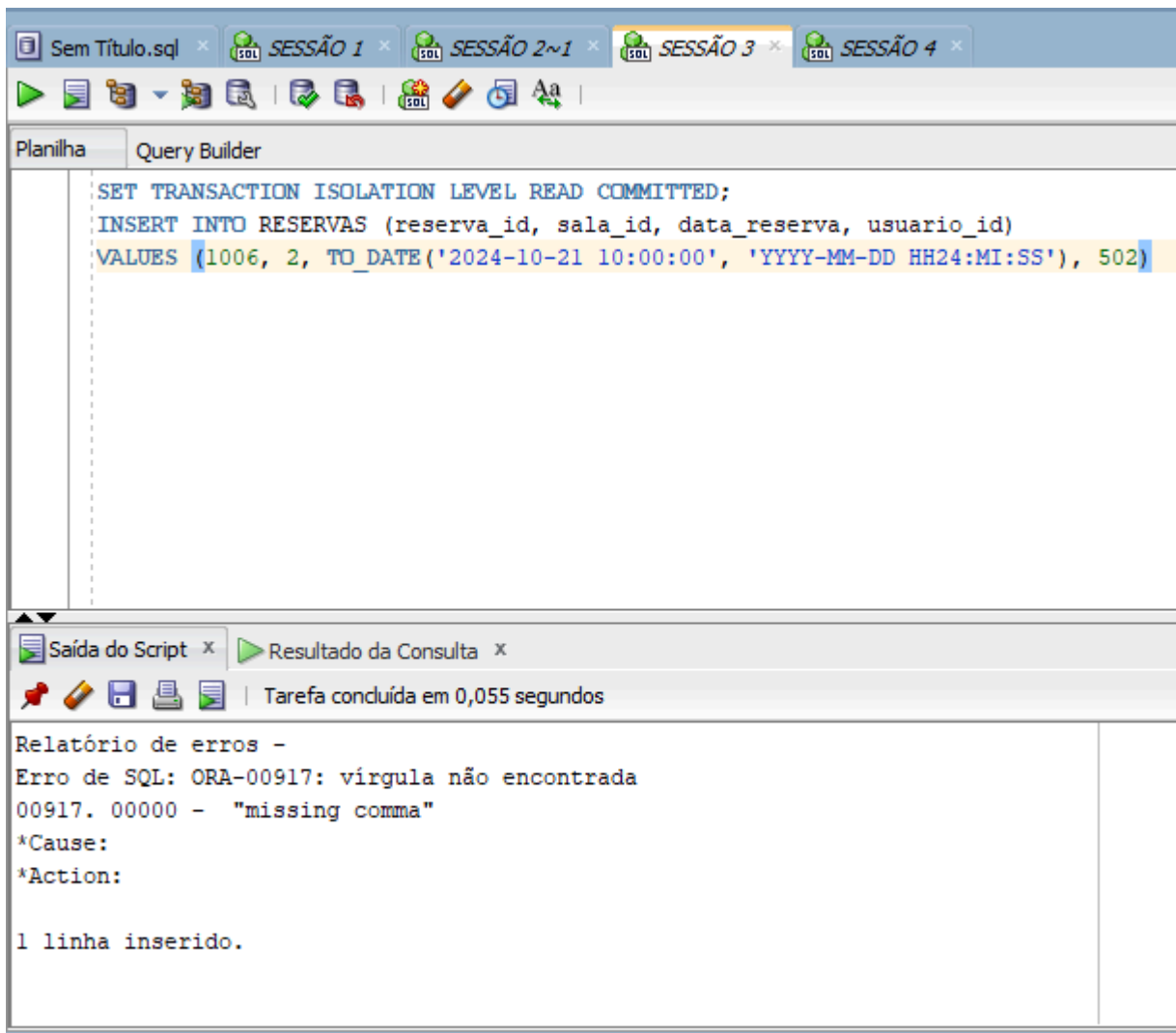
- Aqui ocorre o início do conflito, pois a Sessão 2 tenta inserir uma nova reserva na sala 1, que está bloqueada pela Sessão 1.



#### 4. Sessão 3: Insere nova reserva na sala 2

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  
INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id)  
VALUES (1002, 2, TO_DATE('2024-10-21 10:00:00', 'YYYY-MM-DD HH24:MI:SS'), 502);
```

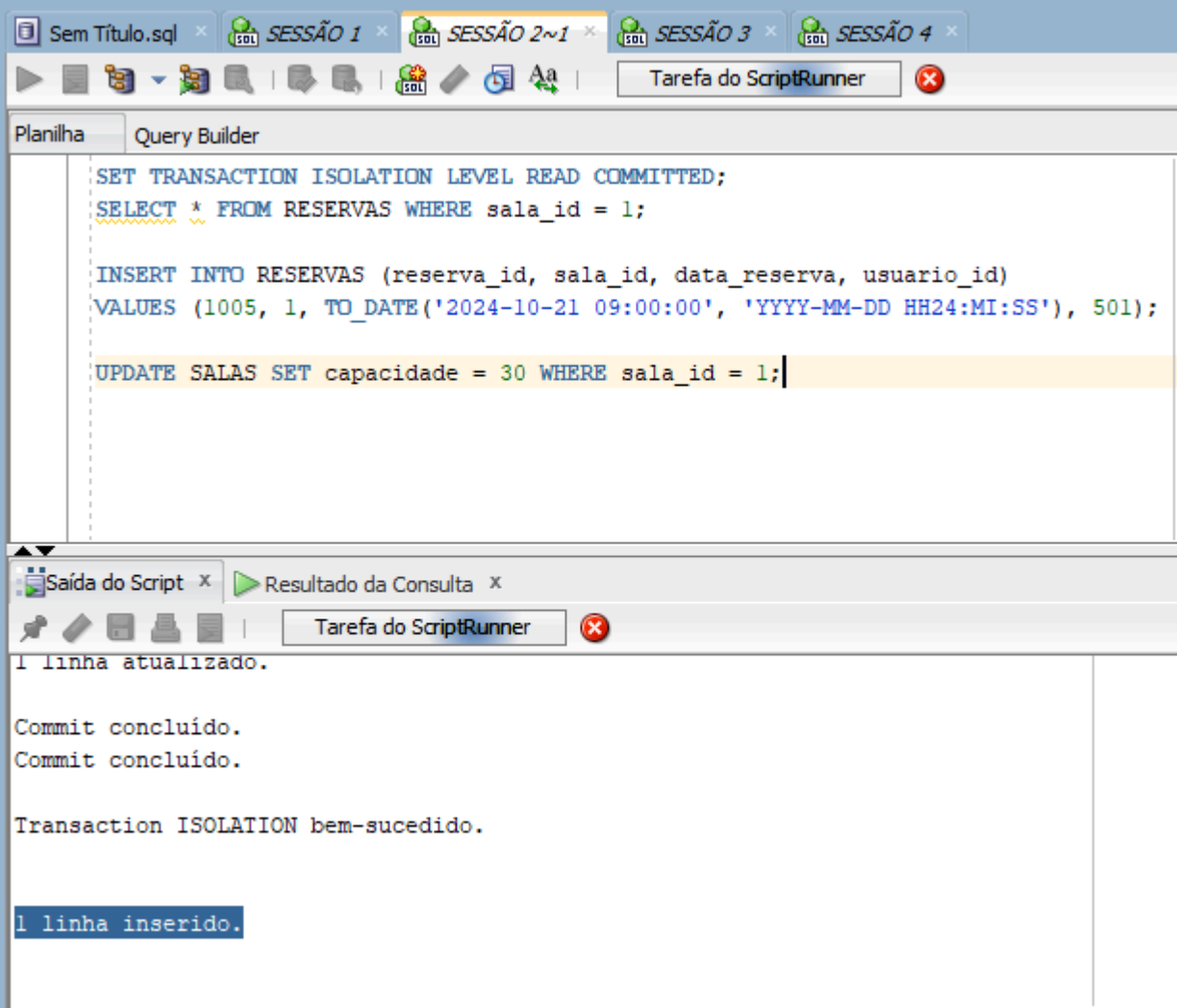
- Essa transação ocorre sem conflitos, pois a sala 2 não está bloqueada.



## 5. Sessão 2: Tenta atualizar a capacidade da sala 1

```
UPDATE SALAS SET capacidade = 30 WHERE sala_id = 1;
```

- Nesse momento, a Sessão 2 fica travada, aguardando a liberação do bloqueio que a Sessão 1 mantém na sala 1, o que resulta no deadlock.



Histórico SQL

SQL	Conexão	TimeStamp	Tipo	Executado	Duração
UPDATE SALAS SET capacidade = 50 WHERE sala_id = 1;	SESSÃO 1	1729487004281	SQL	1	0.034
INSERT INTO RESERVAS ( reserva_id, sala_id, data_reserva, usuario_id) VALUES ( 1006, 2, TO_DATE('2024-10-21 10:00:00', 'YYYY-MM-DD HH24:MI:SS'), 502)	SESSÃO 3	1729486932840	SQL	1	0.038
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;	SESSÃO 3	1729486886127	SQL	1	0.044
INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id) VALUES (1005, 1, TO_DATE('2024-10-21 09:00:00', 'YYYY-MM-DD HH24:MI:SS'), 501)	SESSÃO 2	1729486764033	SQL	1	0.068
SELECT * FROM RESERVAS WHERE sala_id = 1;	SESSÃO 2	1729486668243	SQL	1	0.043
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;	SESSÃO 2	1729486664389	SQL	1	0.036
SELECT * FROM SALAS WHERE sala_id = 1 FOR UPDATE;	SESSÃO 1	1729486592888	SQL	1	0.07
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;	SESSÃO 1	1729486587844	SQL	1	0.036

Ordem de execução	T1	T2	T3	T4
1	SET TRANSACTION ISOLATION LEVEL SERIALIZABLE; SELECT * FROM SALAS WHERE sala_id = 1 FOR UPDATE;			
Resultado	1Sala de Reunião A50			
2		SET TRANSACTION ISOLATION LEVEL READ COMMITTED; SELECT * FROM RESERVAS WHERE sala_id = 1;		
Resultado		1123/10/24 10:00:00,000000000101 4123/10/24 15:00:00,000000000104 1004122/10/24 14:00:00,0000000000502 1001121/10/24 12:00:00,0000000000502 1003122/10/24 12:00:00,0000000000501		
3		INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id) VALUES (1001, 1, TO_DATE('2024-10-21 09:00:00', 'YYYY-MM-DD HH24:MI:SS'), 501);		
Resultado		1 linha inserido.		
4			SET TRANSACTION ISOLATION LEVEL READ COMMITTED; INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id) VALUES (1002, 2, TO_DATE('2024-10-21 10:00:00', 'YYYY-MM-DD HH24:MI:SS'), 502);	
Resultado			1 linha inserido.	
5		UPDATE SALAS SET capacidade = 30 WHERE sala_id = 1;		
Resultado		erro		

Non-Repeatable Read

A seguir apresentamos um cenário envolvendo **leitura não repetível** (non-repeatable read) usando quatro sessões, onde uma sessão faz uma leitura e, em seguida, outra sessão altera os dados antes que a primeira sessão faça uma nova leitura.

Cenário com Quatro Sessões

Objetivo:

- 1. **Sessão 1:** Lê uma reserva.
- 2. **Sessão 2:** Atualiza a reserva lida pela Sessão 1.
- 3. **Sessão 3:** Lê a mesma reserva que a Sessão 1.
- 4. **Sessão 4:** Realiza outra atualização e finaliza a transação.
- 5. **Sessão 1:** Faz uma nova leitura da reserva para observar a alteração.

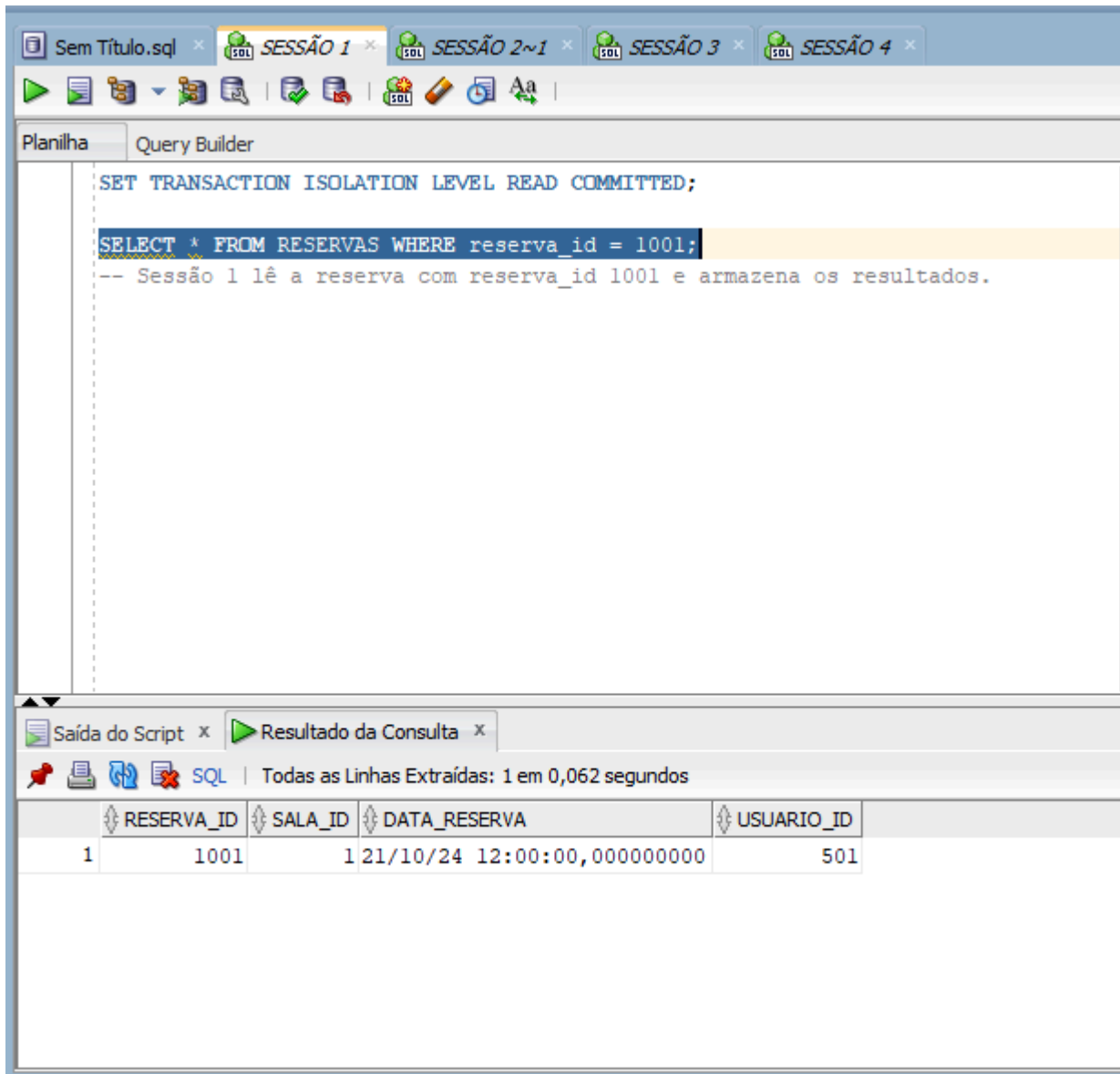
Sequência de Operações

- 1. Sessão 1: Inicia a transação e lê uma reserva específica

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
```



```
SELECT * FROM RESERVAS WHERE reserva_id = 1001;
-- Sessão 1 lê a reserva com reserva_id 1001 e armazena os resultados.
```



The screenshot shows a SQL IDE interface. The top toolbar includes icons for running queries, saving, and other database operations. The main window is titled 'Query Builder' and contains the following SQL code:

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

SELECT * FROM RESERVAS WHERE reserva_id = 1001;
-- Sessão 1 lê a reserva com reserva_id 1001 e armazena os resultados.
```

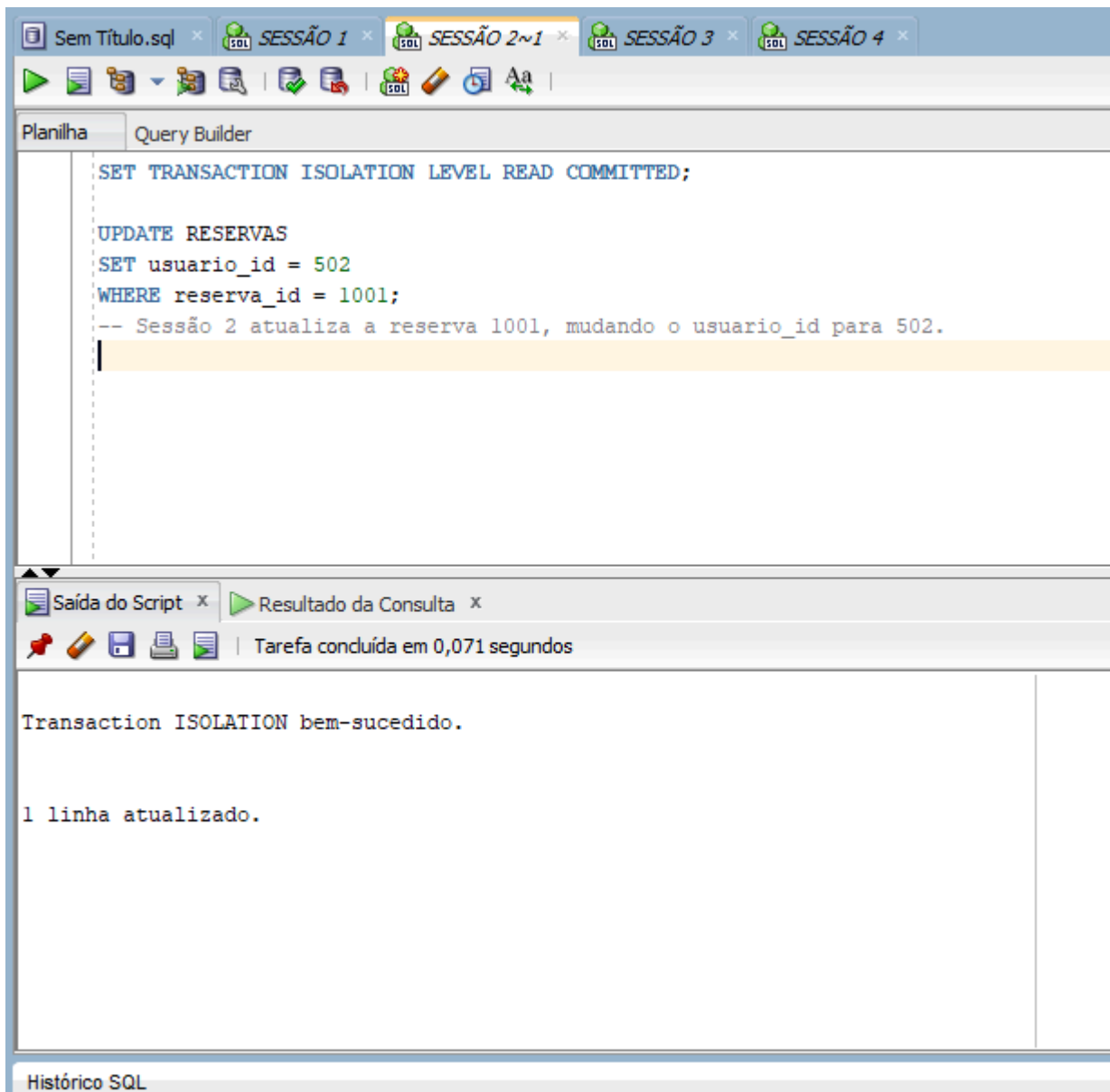
Below the query window, there is a 'Resultado da Consulta' (Query Result) window. It displays the results of the query in a table format. The table has four columns: RESERVA\_ID, SALA\_ID, DATA\_RESERVA, and USUARIO\_ID. The first row shows the results for the query.

	RESERVA_ID	SALA_ID	DATA_RESERVA	USUARIO_ID
1	1001	1	21/10/24 12:00:00,0000000000	501

## 2. Sessão 2: Inicia uma nova transação e atualiza a reserva lida pela Sessão 1

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

UPDATE RESERVAS
SET usuario_id = 502
WHERE reserva_id = 1001;
-- Sessão 2 atualiza a reserva 1001, mudando o usuario_id para 502.
```



### 3. Sessão 3: Inicia uma transação e lê a mesma reserva que a Sessão 1

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

SELECT * FROM RESERVAS WHERE reserva_id = 1001;
-- Sessão 3 lê a reserva 1001, deverá ver o valor original do usuario_id.
```

The screenshot shows a SQL IDE interface. The top window, titled 'Query Builder', contains the following SQL code:

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  
  
SELECT * FROM RESERVAS WHERE reserva_id = 1001;  
-- Sessão 3 lê a reserva 1001, deverá ver o valor original do usuario_id.
```

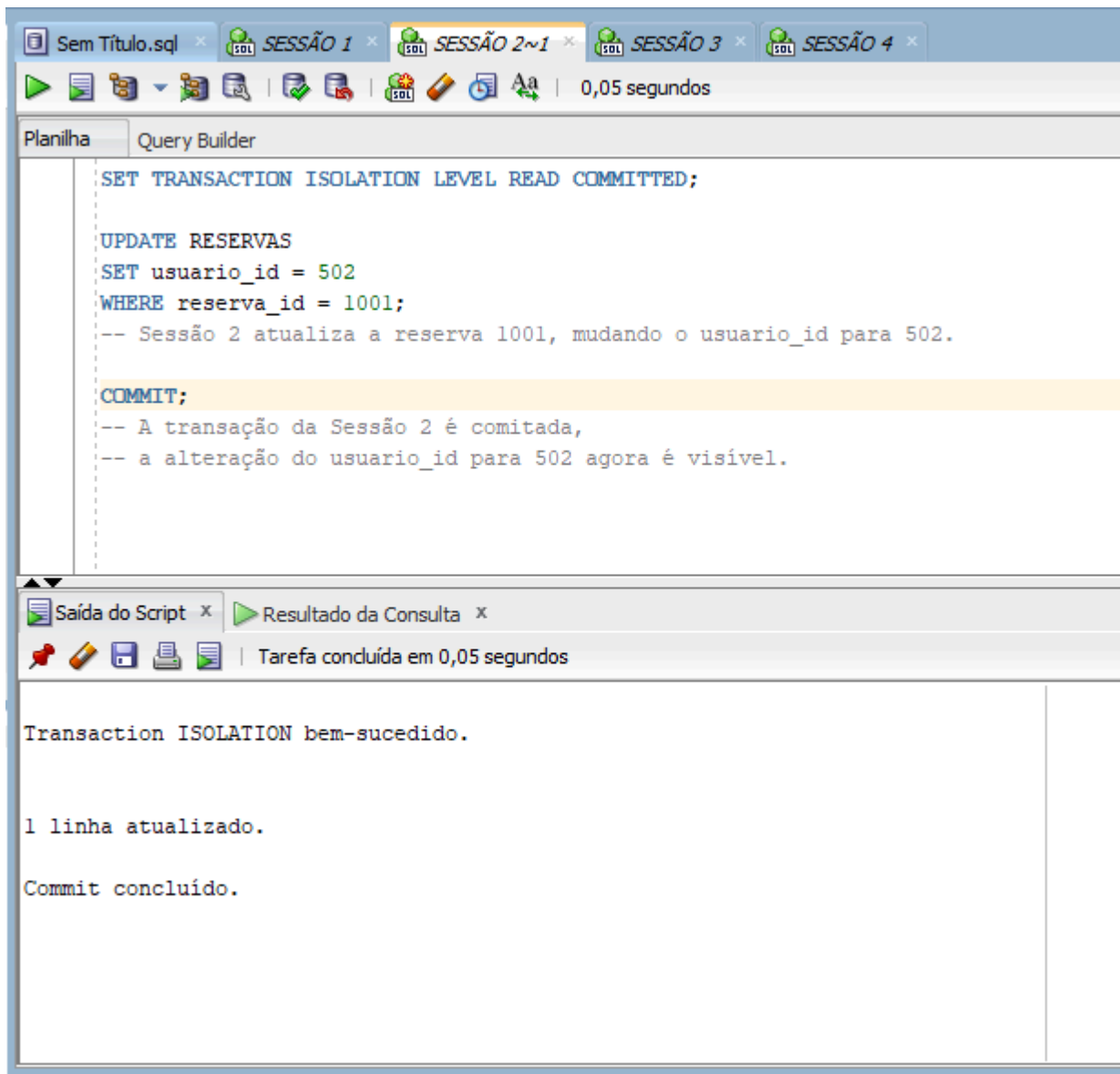
The bottom window, titled 'Resultado da Consulta', displays the results of the query in a table format. The table has four columns: RESERVA\_ID, SALA\_ID, DATA\_RESERVA, and USUARIO\_ID. The first row shows the reservation details for reservation\_id 1001, which was made by user\_id 501 on 21/10/24 at 12:00:00.

	RESERVA_ID	SALA_ID	DATA_RESERVA	USUARIO_ID
1	1001	1	21/10/24 12:00:00,000000000	501

#### 4. Sessão 2: Comita a transação

```
COMMIT;
```

-- A transação da Sessão 2 é comitada, a alteração do usuario\_id para 502 agora é visível.



## 5. Sessão 1: Faz uma nova leitura da reserva

```
SELECT * FROM RESERVAS WHERE reserva_id = 1001;
-- Sessão 1 lê novamente a reserva 1001.
-- Deve observar que o usuario_id mudou para 502, mostrando a leitura não repetível.
```

The screenshot shows a SQL IDE window with multiple tabs labeled 'SESSÃO 1' through 'SESSÃO 4'. The 'Query Builder' tab is active, displaying the following SQL script:

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

SELECT * FROM RESERVAS WHERE reserva_id = 1001;
-- Sessão 1 lê a reserva com reserva_id 1001 e armazena os resultados.

SELECT * FROM RESERVAS WHERE reserva_id = 1001;
-- Sessão 1 lê novamente a reserva 1001.
-- Deve observar que o usuario_id mudou para 502,
-- mostrando a leitura não repetível.
```

Below the script, the 'Resultado da Consulta' (Query Result) tab is active, showing a table with the following data:

	RESERVA_ID	SALA_ID	DATA_RESERVA	USUARIO_ID
1	1001	1	21/10/24 12:00:00,000000000	502

## Resumo do Cenário

1. **Sessão 1** lê a reserva 1001 e armazena os resultados, por exemplo, `usuario_id = 501`.
2. **Sessão 2** inicia uma transação e atualiza a reserva 1001, mudando o `usuario_id` para 502.
3. **Sessão 3** lê a reserva 1001 antes de a Sessão 2 ser comitada e vê o valor original, `usuario_id = 501`.
4. **Sessão 2** comita a transação, tornando a nova alteração visível.
5. **Sessão 1** faz uma nova leitura da reserva 1001 e observa que o `usuario_id` agora é 502, demonstrando o fenômeno de leitura não repetível.

## Histórico SQL

SQL	Conexão	TimeStamp	Tipo	Executado	Duração
-----	-----	-----	-----	-----	-----
SELECT \* FROM RESERVAS WHERE reserva_id = 1001;	SESSÃO 1	1729477047023	SQL	2	0.054
COMMIT;	SESSÃO 2	1729476982203	SQL	1	0.031
SELECT \* FROM RESERVAS WHERE reserva_id = 1001;	SESSÃO 3	1729476914976	SQL	1	0.059
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;	SESSÃO 3	1729476907563	SQL	1	0.043
UPDATE RESERVAS					
SET usuario_id = 502					
WHERE reserva_id = 1001;	SESSÃO 2	1729476855502	SQL	1	0.045
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;	SESSÃO 2	1729476851228	SQL	1	0.072
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;	SESSÃO 1	1729476766264	SQL	1	0.072

## Tabela de execução

Ordem de execução	T1	T2	T3	T4
1	SET TRANSACTION ISOLATION LEVEL READ COMMITTED; SELECT * FROM RESERVAS WHERE reserva_id = 1001;			
Resultado	1001 1 21/10/24 12:00:00,00000000 501			
Observação				
2		SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  UPDATE RESERVAS SET usuario_id = 502 WHERE reserva_id = 1001;		
Resultado		Transaction ISOLATION bem-sucedido.  1 linha atualizado.		
Observação				
3			SET TRANSACTION ISOLATION LEVEL READ COMMITTED; SELECT * FROM RESERVAS WHERE reserva_id = 1001;	
Resultado			1001 1 21/10/24 12:00:00,00000000 501	
Observação				
4		COMMIT;		
Resultado		Commit concluído.		
Observação				
5	SELECT * FROM RESERVAS WHERE reserva_id = 1001;			
Resultado	1001 1 21/10/24 12:00:00,00000000 502			
Observação				

## Problema de Phantom Read

Vamos criar um cenário envolvendo quatro sessões diferentes para gerar o problema de **leitura fantasma** (phantom read). O objetivo é que essas quatro sessões interajam entre si de forma a provocar diferentes estados de leitura e escrita, demonstrando claramente o fenômeno.

Cenário com Quatro Sessões

### Objetivo:

1. **Sessão 1:** Inicia uma transação e lê as reservas.
2. **Sessão 2:** Insere uma nova reserva.
3. **Sessão 3:** Inicia uma transação e também lê as reservas.
4. **Sessão 4:** Insere uma nova reserva e finaliza a transação.

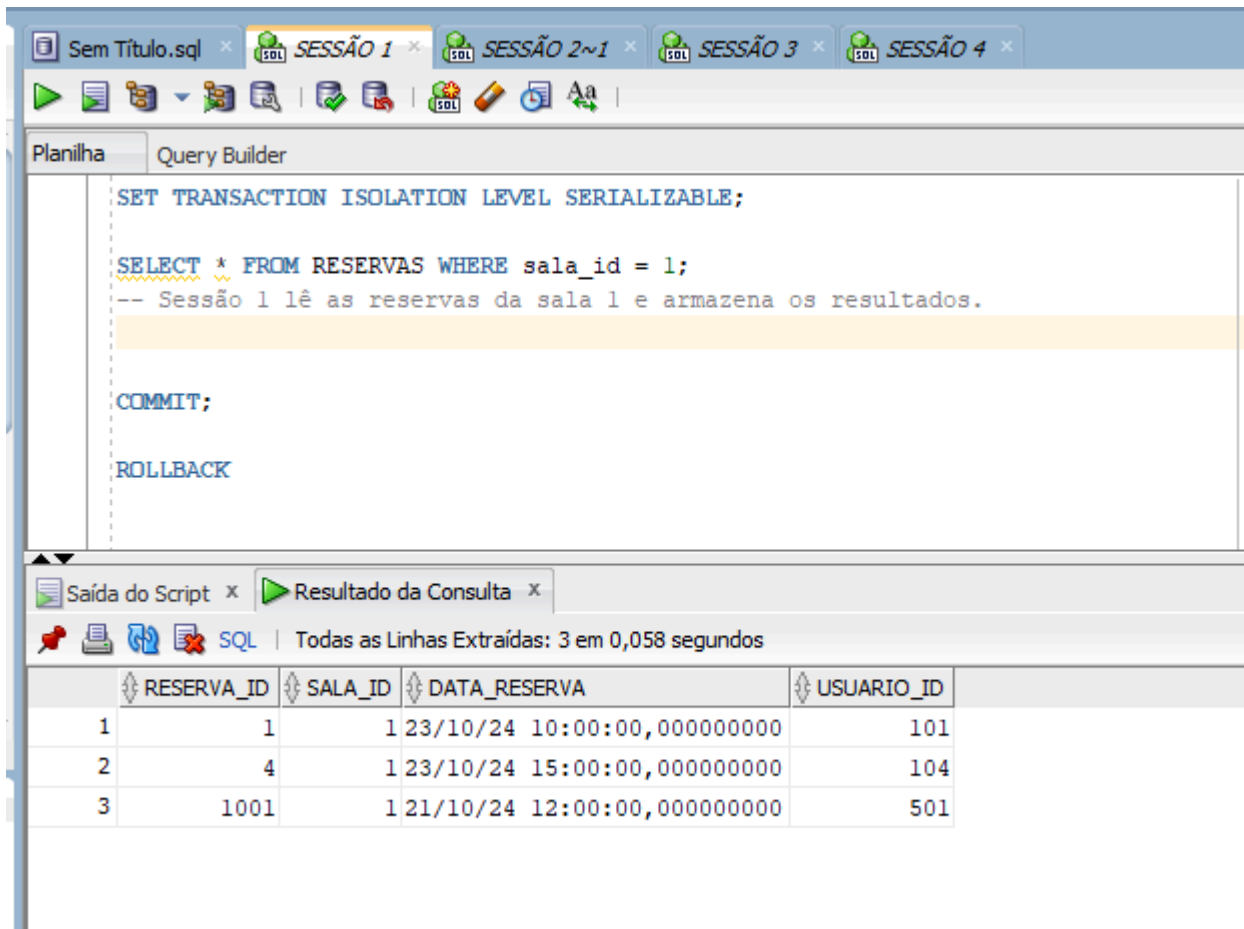
5. **Sessão 3:** Realiza uma nova leitura para observar a mudança.

## Sequência de Operações

1. Sessão 1: Inicia a transação e lê reservas da sala 1

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

SELECT * FROM RESERVAS WHERE sala_id = 1;
-- Sessão 1 lê as reservas da sala 1 e armazena os resultados.
```



The screenshot shows a SQL IDE interface with a query script editor and a results pane. The query script editor contains the following SQL code:

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

SELECT * FROM RESERVAS WHERE sala_id = 1;
-- Sessão 1 lê as reservas da sala 1 e armazena os resultados.

COMMIT;

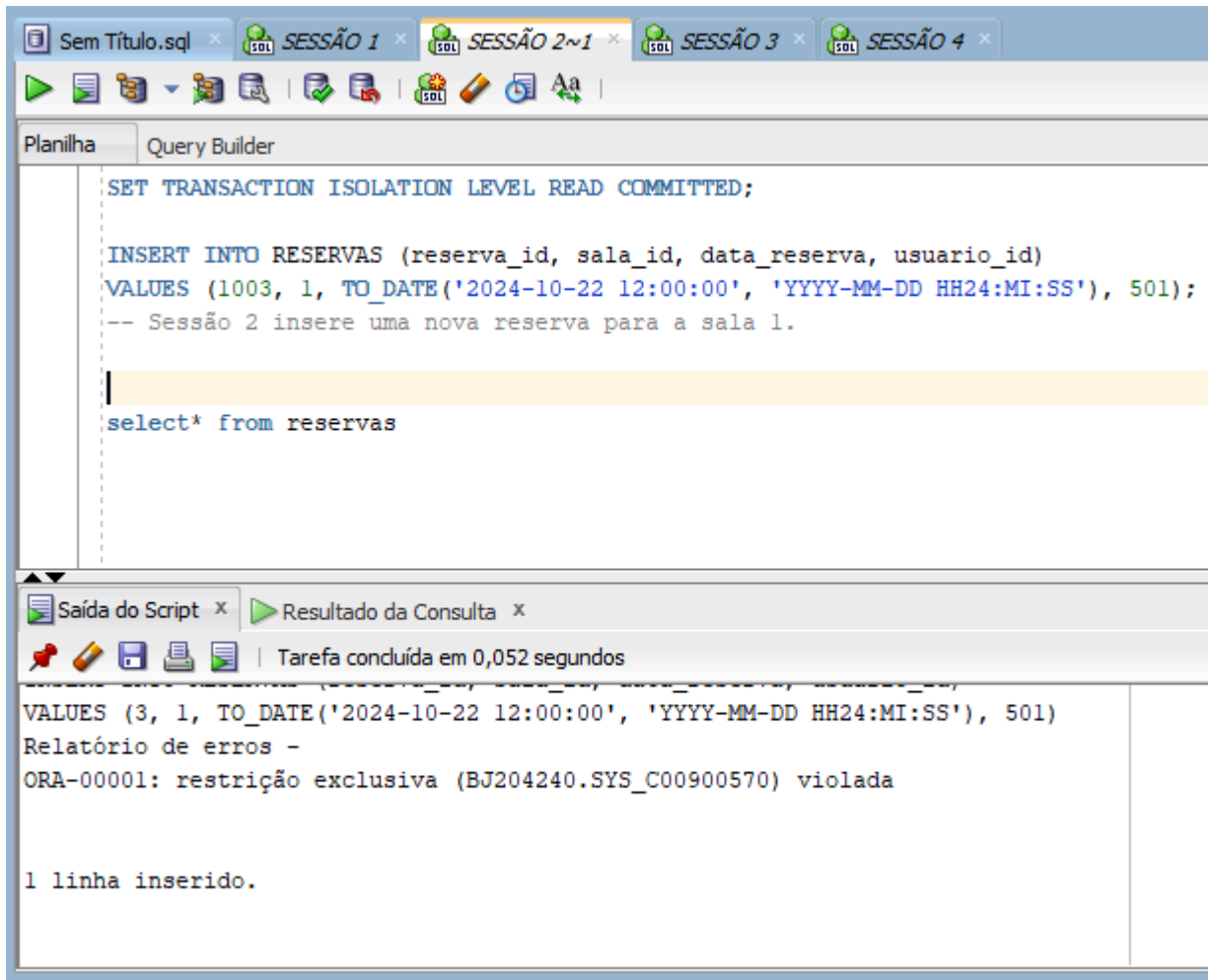
ROLLBACK
```

The results pane displays the output of the query, showing 3 rows of data. The columns are RESERVA\_ID, SALA\_ID, DATA\_RESERVA, and USUARIO\_ID.

	RESERVA_ID	SALA_ID	DATA_RESERVA	USUARIO_ID
1	1	1	23/10/24 10:00:00,000000000	101
2	4	4	23/10/24 15:00:00,000000000	104
3	1001	1	21/10/24 12:00:00,000000000	501

2. Sessão 2: Inicia uma nova transação e insere uma nova reserva na sala 1

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
BEGIN;
INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id)
VALUES (1001, 1, TO_DATE('2024-10-21 12:00:00', 'YYYY-MM-DD HH24:MI:SS'), 501);
-- Sessão 2 insere uma nova reserva para a sala 1.
```



### 3. Sessão 3: Inicia uma transação e lê as reservas da sala 1

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

```
SELECT * FROM RESERVAS WHERE sala_id = 1;
```

```
-- Sessão 3 lê as reservas da sala 1. Neste ponto, não deve ver a nova reserva da Sessão 2.
```



Sem Título.sql x SESSÃO 1 x SESSÃO 2~1 x SESSÃO 3 x SESSÃO 4 x

Planilha Query Builder

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;  
SELECT * FROM RESERVAS WHERE sala_id = 1;  
-- Sessão 3 lê as reservas da sala 1. Neste ponto, não deve ver a nova reserva da Sessão 2.
```

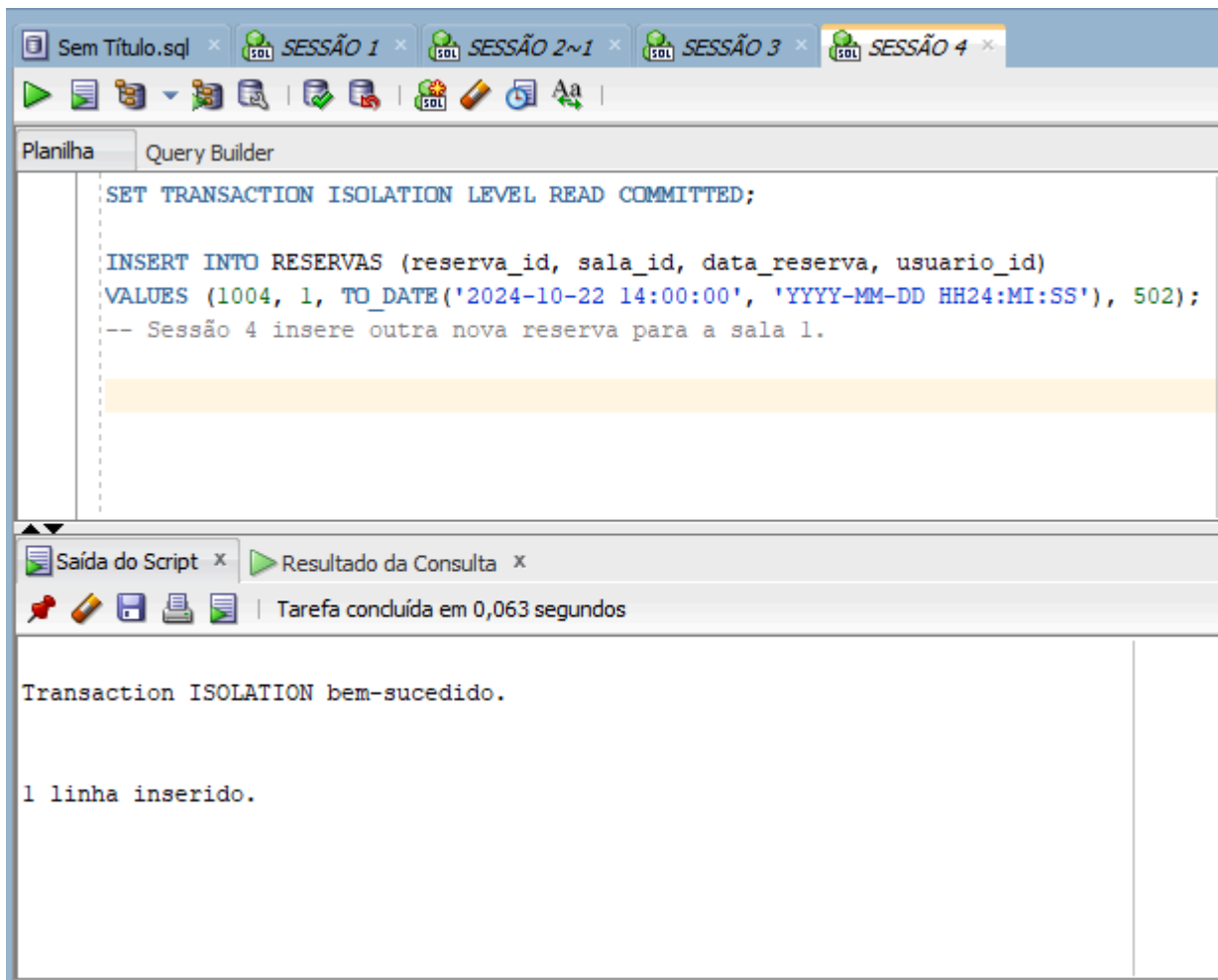
Saída do Script x Resultado da Consulta x

Todas as Linhas Extraídas: 3 em 0,052 segundos

	RESERVA_ID	SALA_ID	DATA_RESERVA	USUARIO_ID
1	1	1	23/10/24 10:00:00,000000000	101
2	4	4	23/10/24 15:00:00,000000000	104
3	1001	1	21/10/24 12:00:00,000000000	501

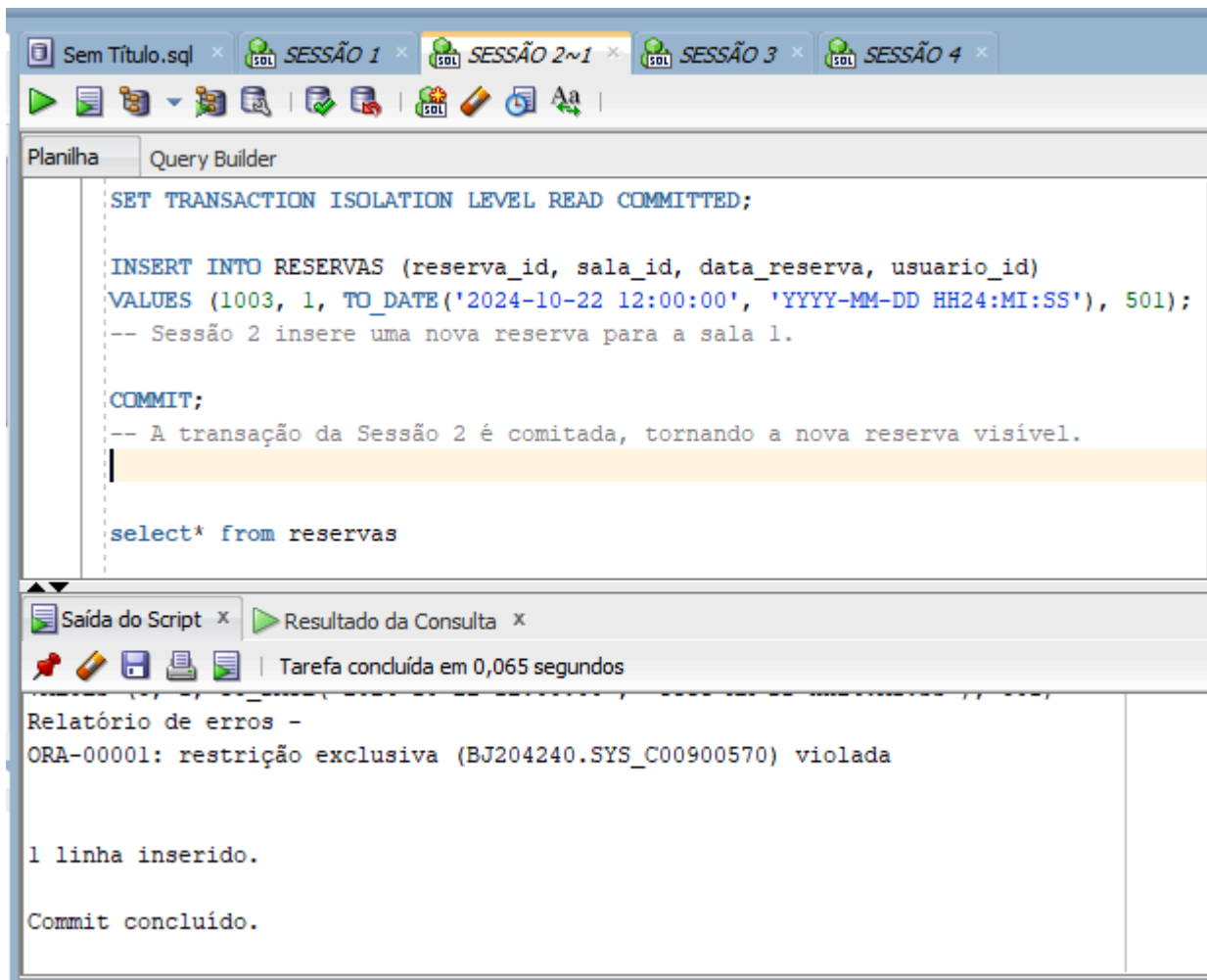
4. Sessão 4: Inicia uma nova transação e insere outra nova reserva na sala 1

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  
  
INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id)  
VALUES (1002, 1, TO_DATE('2024-10-22 14:00:00', 'YYYY-MM-DD HH24:MI:SS'), 502);  
-- Sessão 4 insere outra nova reserva para a sala 1.
```



## 5. Sessão 2: Comita a transação

```
COMMIT;
-- A transação da Sessão 2 é comitada, tornando a nova reserva visível.
```



## 6. Sessão 3: Realiza uma nova leitura das reservas da sala 1

```
SELECT * FROM RESERVAS WHERE sala_id = 1;
-- Sessão 3 tenta ler novamente as reservas da sala 1.
-- Agora deve retornar o mesmo conjunto de resultados que teve na primeira leitura,
pois a transação ainda está em SERIALIZABLE.
```

Sem Título.sql x SESSÃO 1 x SESSÃO 2~1 x SESSÃO 3 x SESSÃO 4 x

Planilha Query Builder

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;  
SELECT * FROM RESERVAS WHERE sala_id = 1;  
-- Sessão 3 lê as reservas da sala 1. Neste ponto, não deve ver a nova reserva da  
  
SELECT * FROM RESERVAS WHERE sala_id = 1;  
-- Sessão 3 tenta ler novamente as reservas da sala 1.  
-- Agora deve retornar o mesmo conjunto de resultados que teve na primeira leitura
```

Saída do Script x Resultado da Consulta x

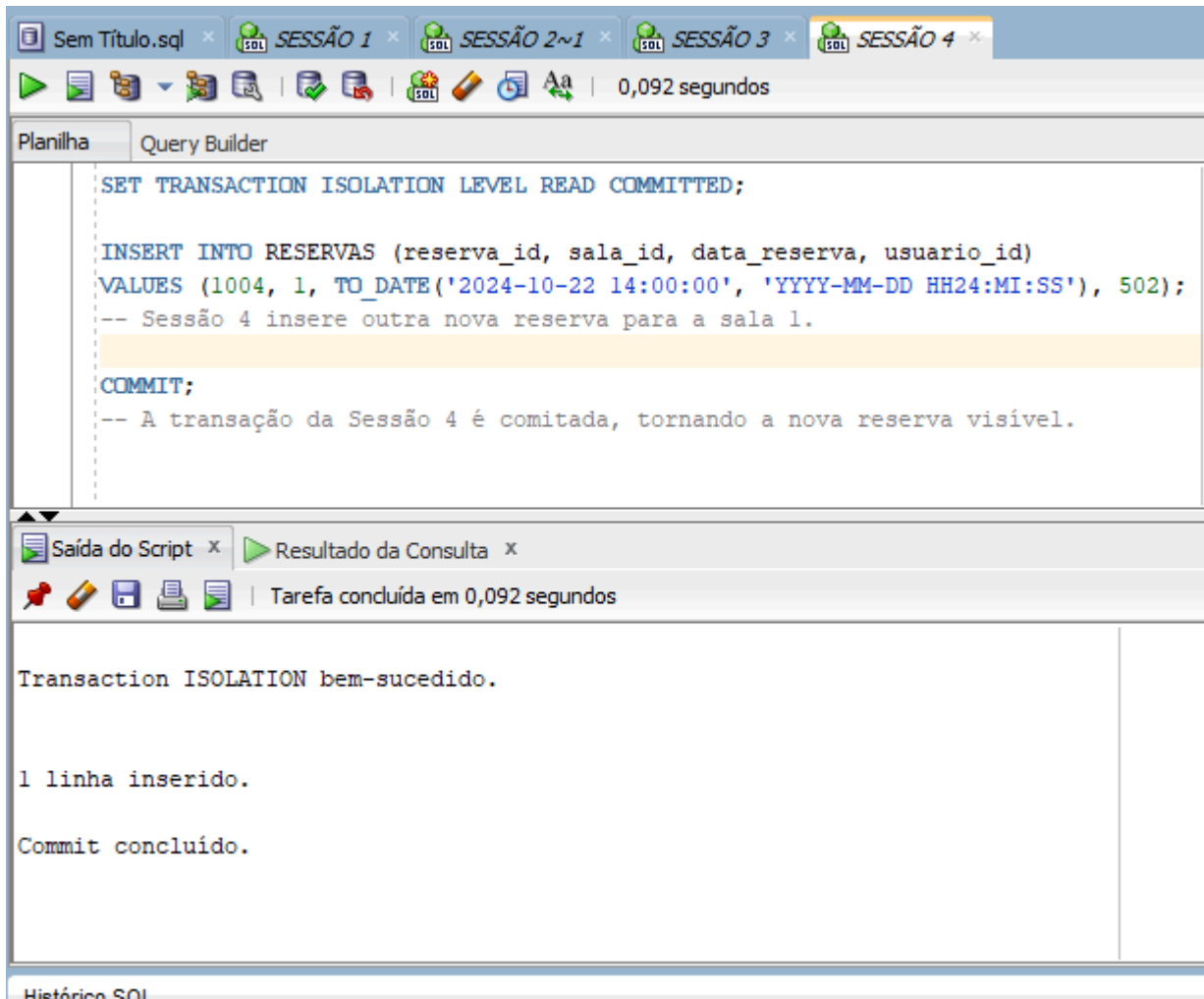
Todas as Linhas Extraídas: 3 em 0,049 segundos

	RESERVA_ID	SALA_ID	DATA_RESERVA	USUARIO_ID
1	1	1	23/10/24 10:00:00,000000000	101
2	4	1	23/10/24 15:00:00,000000000	104
3	1001	1	21/10/24 12:00:00,000000000	501

## 7. Sessão 4: Comita a transação

```
COMMIT;
```

```
-- A transação da Sessão 4 é comitada, tornando a nova reserva visível.
```



## 8. Sessão 3: Realiza uma nova leitura das reservas da sala 1

```
SELECT * FROM RESERVAS WHERE sala_id = 1;
-- Sessão 3 agora deve ver o conjunto de resultados que inclui as novas reservas
inseridas pela Sessão 2 e Sessão 4.
-- Isso demonstra a leitura fantasma, pois os resultados mudaram após a primeira
leitura.
```

A seguir demonstramos diferentes valores nas sessões.

sessão 1

Sem Título.sqlSESSÃO 1SESSÃO 2~1SESSÃO 3SESSÃO 4

PlanilhaQuery Builder

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

SELECT * FROM RESERVAS WHERE sala_id = 1;
-- Sessão 1 lê as reservas da sala 1 e armazena os resultados.

COMMIT;

ROLLBACK
```

Saída do ScriptResultado da Consulta

SQL | Todas as Linhas Extraídas: 5 em 0,052 segundos

	RESERVA_ID	SALA_ID	DATA_RESERVA	USUARIO_ID
1	1	1	1 23/10/24 10:00:00,000000000	101
2	4	4	1 23/10/24 15:00:00,000000000	104
3	1004	1004	1 22/10/24 14:00:00,000000000	502
4	1001	1001	1 21/10/24 12:00:00,000000000	501
5	1003	1003	1 22/10/24 12:00:00,000000000	501

sessão 2

Sem Título.sqlSESSÃO 1SESSÃO 2~1SESSÃO 3SESSÃO 4

PlanilhaQuery Builder

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id)
VALUES (1003, 1, TO_DATE('2024-10-22 12:00:00', 'YYYY-MM-DD HH24:MI:SS'), 501);
-- Sessão 2 insere uma nova reserva para a sala 1.

COMMIT;
-- A transação da Sessão 2 é comitada, tornando a nova reserva visível.

SELECT * FROM RESERVAS WHERE sala_id = 1;
```

Saída do ScriptResultado da Consulta

SQL | Todas as Linhas Extraídas: 5 em 0,051 segundos

	RESERVA_ID	SALA_ID	DATA_RESERVA	USUARIO_ID
1	1	1	1 23/10/24 10:00:00,000000000	101
2	4	4	1 23/10/24 15:00:00,000000000	104
3	1004	1004	1 22/10/24 14:00:00,000000000	502
4	1001	1001	1 21/10/24 12:00:00,000000000	501
5	1003	1003	1 22/10/24 12:00:00,000000000	501

Sem Título.sql x SESSÃO 1 x SESSÃO 2~1 x SESSÃO 3 x SESSÃO 4 x

Planilha Query Builder

```

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

SELECT * FROM RESERVAS WHERE sala_id = 1;
-- Sessão 3 lê as reservas da sala 1. Neste ponto, não deve ver a nova reserva da

SELECT * FROM RESERVAS WHERE sala_id = 1;
-- Sessão 3 tenta ler novamente as reservas da sala 1.
-- Agora deve retornar o mesmo conjunto de resultados que teve na primeira leitura

SELECT * FROM RESERVAS WHERE sala_id = 1;
-- Sessão 3 agora deve ver o conjunto de resultados que inclui as novas
-- reservas inseridas pela Sessão 2 e Sessão 4.
-- Isso demonstra a leitura fantasma, pois os resultados mudaram após a primeira

```

Saída do Script x Resultado da Consulta x

SQL | Todas as Linhas Extraídas: 3 em 0,055 segundos

	RESERVA_ID	SALA_ID	DATA_RESERVA	USUARIO_ID
1	1	1	1 23/10/24 10:00:00,000000000	101
2	4	4	1 23/10/24 15:00:00,000000000	104
3	1001	1	1 21/10/24 12:00:00,000000000	501

Histórico SQL

sessão 3

Sem Título.sql x SESSÃO 1 x SESSÃO 2~1 x SESSÃO 3 x SESSÃO 4 x

Planilha Query Builder

```

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id)
VALUES (1004, 1, TO_DATE('2024-10-22 14:00:00', 'YYYY-MM-DD HH24:MI:SS'), 502);
-- Sessão 4 insere outra nova reserva para a sala 1.

COMMIT;
-- A transação da Sessão 4 é comitada, tornando a nova reserva visível.

SELECT * FROM RESERVAS WHERE sala_id = 1;

```

Saída do Script x Resultado da Consulta x

SQL | Todas as Linhas Extraídas: 5 em 0,046 segundos

	RESERVA_ID	SALA_ID	DATA_RESERVA	USUARIO_ID
1	1	1	1 23/10/24 10:00:00,000000000	101
2	4	4	1 23/10/24 15:00:00,000000000	104
3	1004	1	1 22/10/24 14:00:00,000000000	502
4	1001	1	1 21/10/24 12:00:00,000000000	501
5	1003	1	1 22/10/24 12:00:00,000000000	501

sessão 4

## Resumo do Cenário

1. **Sessão 1** lê as reservas da sala 1 e armazena os resultados.
2. **Sessão 2** insere uma nova reserva para a sala 1, mas não a comita imediatamente.
3. **Sessão 3** lê as reservas da sala 1 e não vê a nova reserva inserida pela Sessão 2.
4. **Sessão 4** insere outra nova reserva para a sala 1.
5. **Sessão 2** comita sua transação, fazendo com que a nova reserva fique visível.
6. **Sessão 3** lê novamente as reservas, mas ainda deve ver o mesmo conjunto de resultados que teve na primeira leitura.
7. **Sessão 4** comita sua transação.
8. **Sessão 3** faz uma nova leitura e agora vê o novo conjunto de resultados que inclui as reservas inseridas pelas Sessões 2 e 4, ilustrando a leitura fantasma.

Observações Finais

- O fenômeno de leitura fantasma é demonstrado quando a **Sessão 3**, que estava em SERIALIZABLE, lê dados antes e depois das inserções das outras sessões. O resultado muda conforme as transações são comitadas.
- A interação entre as quatro sessões e suas transações ajuda a visualizar claramente como as alterações nas transações afetam as leituras subsequentes

Histórico SQL

SQL	Conexão	TimeStamp	Tipo	Executado	Duração
-----	-----	-----	-----	-----	-----
SELECT \* FROM RESERVAS WHERE sala_id = 1;	SESSÃO 1	1729476150588	SQL	5	0.052
SELECT \* FROM RESERVAS WHERE sala_id = 1;	SESSÃO 2	1729476107673	SQL	2	0.051
SELECT \* FROM RESERVAS WHERE sala_id = 1;	SESSÃO 3	1729476069829	SQL	7	0.055
SELECT \* FROM RESERVAS WHERE sala_id = 1;	SESSÃO 4	1729475999967	SQL	2	0.046
SELECT \* FROM RESERVAS	SESSÃO 3	1729475961242	SQL	1	0.088
COMMIT;	SESSÃO 1	1729475881100	SQL	2	0.035
COMMIT;	SESSÃO 4	1729475869720	SQL	5	0.032
COMMIT;	SESSÃO 2	1729475607320	SQL	3	0.035
INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id)					
VALUES (1004, 1, TO_DATE('2024-10-22 14:00:00', 'YYYY-MM-DD HH24:MI:SS'), 502);	SESSÃO 4	1729475551091	SQL	1	0.031
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;	SESSÃO 4	1729475532691	SQL	1	0.062
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;	SESSÃO 3	1729475461574	SQL	1	0.028
INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id)					
VALUES (1003, 1, TO_DATE('2024-10-22 12:00:00', 'YYYY-MM-DD HH24:MI:SS'), 501);	SESSÃO 2	1729475376448	SQL	1	0.031
select\* from reservas	SESSÃO 2	1729475346305	SQL	1	0.039
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;	SESSÃO 2	1729475204530	SQL	3	0.037
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;	SESSÃO 1	1729475135108	SQL	3	0.383
INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id)					
VALUES (1001, 1, TO_DATE('2024-10-21 12:00:00', 'YYYY-MM-DD HH24:MI:SS'), 501);	SESSÃO 2	1729474601803	SQL	1	0.061
LOCK TABLE SALAS IN EXCLUSIVE MODE;	SESSÃO 1	1729474518538	SQL	1	316.887
COMMIT;	SESSÃO 3	1729474503673	SQL	2	0.072
ROLLBACK;	SESSÃO 4	1729474476932	SQL	1	0.035
rollback	SESSÃO 4	1729474426439	SQL	2	0.04
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;	SESSÃO 4	1729474134932	SQL	1	0.067
UPDATE SALAS SET capacidade = 50 WHERE sala_id = 1;	SESSÃO 2	1729474069537	SQL	3	0.009
ROLLBACK	SESSÃO 1	1729474068200	SQL	1	0.046
INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id)					
VALUES (1002, 2, TO_DATE('2024-10-21 10:00:00', 'YYYY-MM-DD HH24:MI:SS'), 502);	SESSÃO 3	1729473668727	SQL	1	0.053
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;	SESSÃO 3	1729473662388	SQL	1	0.01
INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id)					
VALUES (1001, 1, TO_DATE('2024-10-21 09:00:00', 'YYYY-MM-DD HH24:MI:SS'), 501);	SESSÃO 1	1729473646214	SQL	1	0.044
SELECT \* FROM SALAS WHERE sala_id = 1 FOR UPDATE;	SESSÃO 1	1729473600155	SQL	1	0.046

Tabela de execução



Ordem de execução	T1	T2	T3	T4
1	SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;  SELECT * FROM RESERVAS WHERE sala_id = 1;			
Resultado	1 123/10/24 10:00:00,000000000 101 4 123/10/24 15:00:00,000000000 104 1001121/10/24 12:00:00,000000000 501			
Observação				
2		SET TRANSACTION ISOLATION LEVEL READ COMMITTED; BEGIN; INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id) VALUES (1001, 1, TO_DATE('2024-10-21 12:00:00', 'YYYY-MM-DD HH24:MI:SS'), 501);		
Resultado		1 linha inserido.		
Observação				
3			SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;  SELECT * FROM RESERVAS WHERE sala_id = 1;	
Resultado			1 123/10/24 10:00:00,000000000 101 4 123/10/24 15:00:00,000000000 104 1001121/10/24 12:00:00,000000000 501	
Observação				
4				SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  INSERT INTO RESERVAS (reserva_id, sala_id, data_reserva, usuario_id) VALUES (1002, 1, TO_DATE('2024- 10-22 14:00:00', 'YYYY-MM-DD HH24:MI:SS'), 502);
Resultado				Transaction ISOLATION bem- sucedido.  1 linha atualizado.
Observação				
5		COMMIT;		
Resultado		Commit concluído.		
Observação				
6			SELECT * FROM RESERVAS WHERE sala_id = 1;	
Resultado			1 123/10/24 10:00:00,000000000 101 4 123/10/24 15:00:00,000000000 104 1001121/10/24 12:00:00,000000000 501	
Observação				
7				COMMIT;
Resultado				Commit concluído.
8			SELECT * FROM RESERVAS WHERE sala_id = 1;	
8			123/10/24 10:00:00,000000000 101 4 123/10/24 15:00:00,000000000 104 1004122/10/24 14:00:00,000000000 502 1001121/10/24 12:00:00,000000000 501 1003122/10/24 12:00:00,000000000 501	

		XXXXXXXXXXXX	001
--	--	--------------	-----