

Laboratório de Bases de Dados

Prof. José Fernando Rodrigues Júnior

Aula 6 – Transações, Visões, e Privilégios

Material original editado: Profa. Elaine Parros Machado de Sousa



Transações - Revisão

Transação: Unidade lógica de trabalho

- abrange um conjunto de operações de manipulação de dados que executam uma única tarefa

Conecta ao Banco de Dados

Começa transação

Operações de consulta/atualização

...

Finaliza transação

Começa transação

Operações de consulta/atualização

...

Finaliza transação

Desconecta

Transações – Propriedades ACID

Atomicidade: as operações de uma transação devem ser efetivadas; ou, na ocorrência de uma falha, nada deve ser efetivado

- “**tudo ou nada**” – não se admite parte de uma operação

Consistência: transações preservam a consistência da base

- Estado inicial consistente \Rightarrow Estado final consistente

Isolamento: a maneira como várias transações em paralelo interagem (o que pode ser lido e o que pode ser escrito por cada uma) deve ser bem definido

Durabilidade: uma vez consolidada (committed) a transação, suas alterações permanecem no banco até que outras transações aconteçam

Transações – Propriedades ACID

Atomicidade: as transações devem ser efetivadas; ou, nada deve ser efetivado

Recuperação de falhas
via log

- “**tudo ou nada**” – não se admite parte de uma operação

Consistência: transações preservam a consistência da base

- Estado inicial consistente \Rightarrow Estado final consistente

Isolamento: a maneira como várias transações em paralelo interagem (o que pode ser lido e o que pode ser escrito por cada uma) deve ser bem definido

Durabilidade: uma vez que uma transação tenha sido efetuada (commitada), a transação, suas alterações permanecem permanentemente armazenadas, não aconteçam outras transações

Recuperação de falhas
via log

Transações – Propriedades ACID

Atomicidade: as transações devem ser executadas de forma atômica; ou, nada deve ser efetivado

**Recuperação de falhas
via log**

- “**tudo ou nada**” – não se admite parte de uma operação

Consistência: transações não podem violar as regras de integridade da base de dados

**Controle de Concorrência
via Locks**

- Estado inicial consistente

Isolamento: a manipulação de dados por várias transações em paralelo não deve interferir umas nas outras (o que cada transação escreve deve ser bem escrito por cada uma)

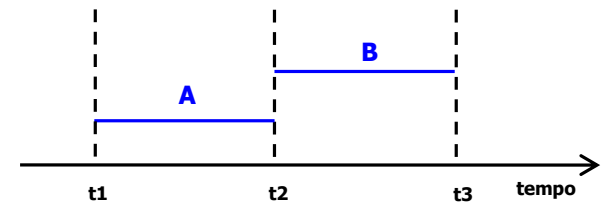
**Controle de Concorrência
via Locks**

Durabilidade: uma vez que uma transação tenha sido confirmada (committed) a transação, suas alterações persistem no banco de dados mesmo que ocorram falhas ou outras transações aconteçam

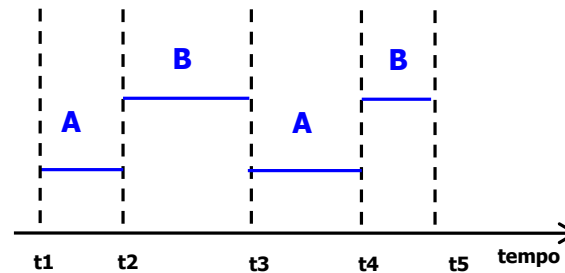
**Recuperação de falhas
via log**

Controle de Concorrência

Execução Serial (sequencial):



Execução Intercalada:



Controle de Concorrência

Execução Serial (sequencial): diversas transações executadas em sequência

- deixa a base de dados em estado correto e consistente

Execução Intercalada: comandos de diversas transações são intercalados

- pode levar a inconsistências

	Isolamento	Concorrência	Inconsistências
Serial	SIM	NÃO	NÃO
Intercalada	NÃO	SIM	SIM

Execução Serial X Intercalada

Execução Intercalada

- Toda execução serial é consistente
- Mas uma execução intercalada só é consistente se for igual ao resultado de uma execução em sequência (em ordem conhecida)
- esta execução é dita **serializável**

Problemas de Execução Intercalada

Ocorrência de anomalias

1. leitura inválida
2. leitura não repetível
3. leitura fantasma

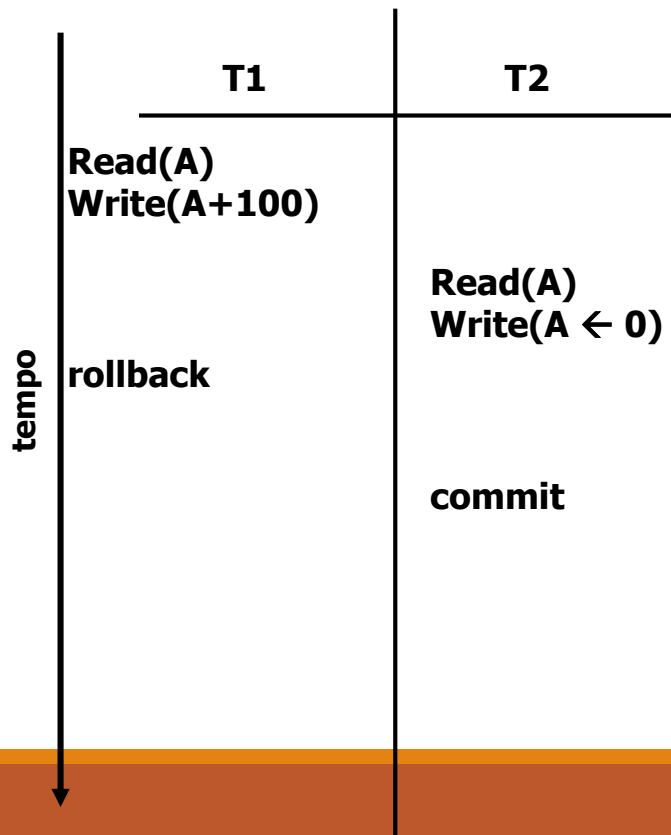
Problemas de Execução Intercalada

1) Leitura inválida (*Dirty Read*):

- transação T' lê um dado modificado por uma transação T que ainda não terminou;
- permite que outras transações possam ver os dados que ainda não foram consolidados (committed), isto é, mudanças que podem ser descartadas em seguida, por causa de uma instrução ROLLBACK por exemplo.

Problemas de Execução Intercalada

Ex: Leitura inválida (*Dirty Read*):



Exemplo 1:

- **Transação T1:** deposita R\$100,00 na conta A.
- **Transação T2:** saca tudo de A.
- **T1** é cancelada

Problemas de Execução Intercalada

Ex: Leitura

Resultado: foi possível sacar R\$ 100,00 a mais.

Solução: lock (trava) de leitura nos valores sendo escritos.

T1

Read(A)
Write(A+100)

rollback

commit

T1 é cancelada

0

tempo

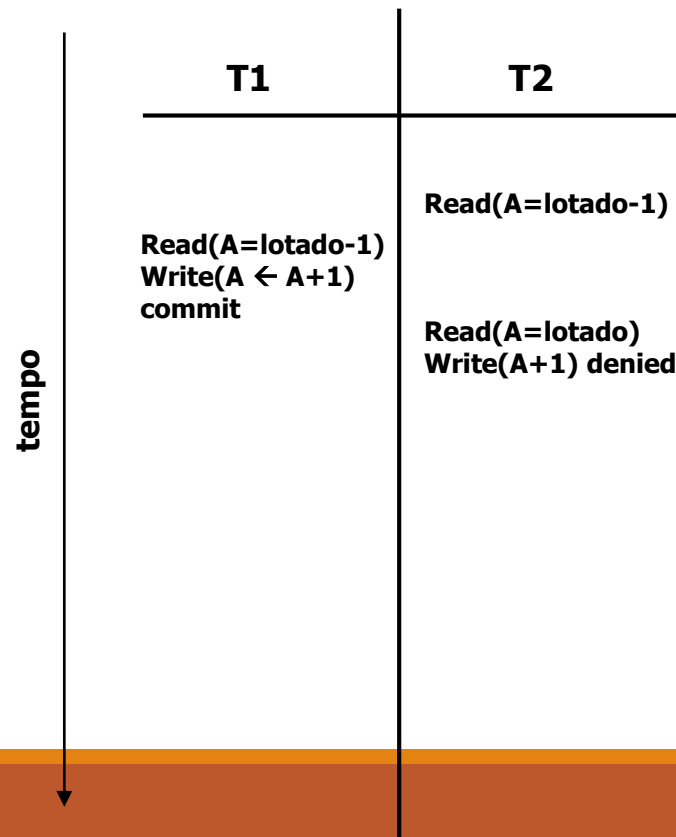
Problemas de Execução Intercalada

2) Leitura não repetível (*Nonrepeatable Read*):

- transação T lê um dado
- esse dado é modificado por uma transação T' que começou depois de T
- T é efetivada
- se T' tentar reler o mesmo dado, obterá valores diferentes (*nonrepeatable read*)

Problemas de Execução Intercalada

Ex: Leitura não repetível (*Nonrepeatable Read*):

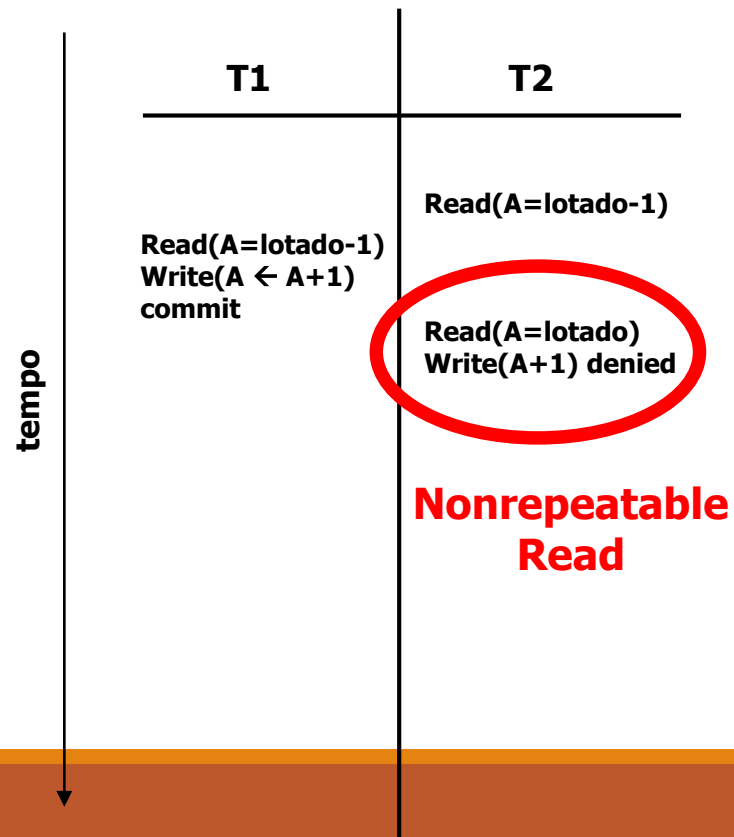


Exemplo:

- **Transação T2:** lê reservas de um voo e verifica que há apenas um lugar disponível.
- **Transação T1:** lê a mesma coisa
- **T1** reserva o último lugar e é efetivada.
- **T2** tenta reservar o lugar e ocorre um erro: **o software ou um trigger impedem a escrita.**

Problemas de Execução Intercalada

Ex: Leitura não repetível (*Nonrepeatable Read*):



Exemplo:

- **Transação T2:** lê reservas de um voo e verifica que há apenas um lugar disponível.
- **Transação T1:** lê a mesma coisa
- **T1** reserva o último lugar e é efetivada.
- **T2** tenta reservar o lugar e ocorre um erro: **o software ou um trigger impedem a escrita.**

Problemas de Execução Intercalada

Ex: Leitura

Resultado: um usuário foi informado de que ainda havia lugares. Após preencher o cadastro, clicou confirma e recebeu um erro (de software ou de trigger) de que o voo estava lotado.

T1

Read(A=lotado)
Write(A ← A + 1)
commit

Antes de terminar a transação, ele leu valores diferentes para um mesmo elemento – não foi possível repetir a leitura.

Solução: lock (trava) de escrita nos valores sendo lidos.

Read

o software ou um trigger impedem a escrita.

erro:

tempo

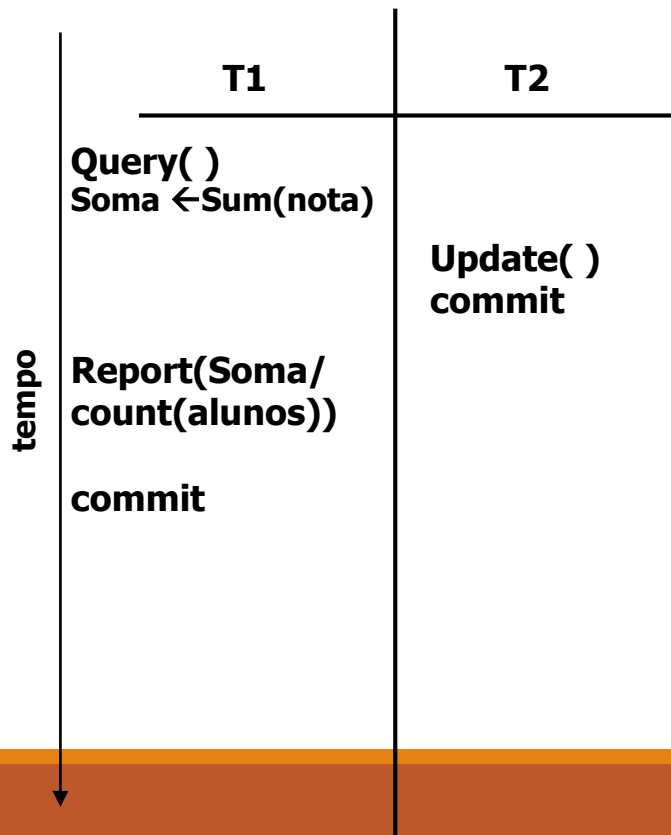
Problemas de Execução Intercalada

3) Leitura fantasma (*Phantom Read*):

- transação T lê um conjunto de tuplas que atendam a uma condição de consulta
- transação T' insere/remove/atualiza uma tupla que atenderia a essa condição e é efetivada
- se T refizer a mesma consulta, obterá um conjunto diferente de tuplas (*phantom read*)

Problemas de Execução Intercalada

Ex: Leitura fantasma (*Phantom Read*):



Exemplo:

- **Transação T1:** faz uma consulta que retorna a média geral dos alunos que têm média ponderada acima de 5.0, e gera um relatório
 - **Transação T2:** acrescenta e/ou deleta alguns alunos
 - **T1** refaz a consulta para gerar relatório com nro de alunos por faixa de média
- ⇒ **relatórios inconsistentes.**

Problemas de Execução Intercalada

Ex: Leitura

T1

Query()
Soma ← Sum(

Report(Soma
count(alunos)
commit

Resultado: o número de alunos mudou, mas o somatório das notas não.

A média gerada considera tuplas que não existem mais, ou desconsidera tuplas que chegaram enquanto as transações foram intercaladas.

Solução: lock de escrita nas tuplas inteiras que satisfazem a um WHERE – lock de predicado. Ou snapshot de dados, isto é, para uma dada transação, os dados não mudam independentemente do que ocorrer em transações concorrentes.

alunos

Problemas de Execução

Repeatable read vs Phantom read

- **Repeatable read**: lê valores diferentes de um mesmo dado.
- **Phantom read**: lê conjuntos de dados diferentes, sendo que um dos conjuntos possui dados que não existem no(s) outro(s) conjunto(s) – fantasmas.
- Phantom reads estão intimamente ligados a **predicados** que determinam **conjuntos** de tuplas; é caracterizado pelo surgimento/desaparecimento de tuplas.

tempo

Problemas de Execução Intercalada → Isolamento

Ocorrência de anomalias

1. **leitura inválida:** leitura de um dado que não foi consolidado, cujo valor pode ser alterado causando inconsistência;
2. **leitura não repetível:** leitura de um dado consolidado, mas cujo valor foi alterado e consolidado ao longo da transação, causando a leitura de um dado diferente durante a mesma transação;
3. **leitura fantasma:** leitura de um conjunto (definido por um predicado) de tuplas consolidadas cujos elementos não se repetem ao longo da transação – tuplas novas surgem e tuplas existentes desaparecem.

Transações – Propriedades ACID

Atomacidade: todas as operações de uma transação devem ser efetivadas; ou, na ocorrência de uma falha, nada deve ser efetivado

- “**tudo ou nada**” – não se admite parte de uma operação

Consistência: transações preservam a consistência da base

- Estado inicial consistente \Rightarrow Estado final consistente

Isolamento: a maneira como várias transações em paralelo interagem (o que pode ser lido e o que pode ser escrito por cada uma) deve ser bem definido

Durabilidade: uma vez consolidada (committed) a transação, suas alterações permanecem no banco até que outras transações aconteçam

Problemas de Execução Intercalada → Isolamento

Ocorrência de anomalias

1. leitura inválida
2. leitura não repetível
3. leitura fantasma

Solução via isolamento em diferentes graus

- Read uncommitted
- Read committed
- Repeatable read
- Serializable

Interpretação

Nível de isolamento	Anomalias que PODEM ocorrer		
	1) Leitura inválida	2) Leitura não repetível	3) Leitura fantasma
Leitura mesmo do que NÃO FOI committed	Sim	Sim	Sim
Leitura apenas do que FOI committed	Não	Sim	Sim
Leitura apenas se a leitura repetida for garantida	Não	Não	Sim
Torna a execução equivalente à execução em série	Não	Não	Não

Níveis de Isolamento em SQL99

Nível de isolamento	Anomalias que PODEM ocorrer		
	1) Leitura inválida	2) Leitura não repetível	3) Leitura fantasma
Read uncommitted	Sim	Sim	Sim
Read committed	Não	Sim	Sim
Repeatable read	Não	Não	Sim
Serializable	Não	Não	Não

Níveis de Isolamento em SQL99

Nível de isolamento	Anomalias que PODEM ocorrer		
	1) Leitura inválida	2) Leitura não repetível	3) Leitura fantasma
Read uncommitted	NÃO EXISTE EM ORACLE <i>Sim</i>	NÃO EXISTE EM ORACLE <i>Sim</i>	<i>Sim</i>
Read committed	MÍNIMO ACEITO EM ORACLE <i>Não</i>	MÍNIMO ACEITO EM ORACLE <i>Sim</i>	<i>Sim</i>
Repeatable read	NÃO EXISTE EM ORACLE <i>Não</i>	NÃO EXISTE EM ORACLE <i>Não</i>	<i>Sim</i>
Serializable	Não	Não	Não

Níveis de Isolamento

Tanto o PostgreSQL quanto o Oracle não implementam todos os quatro níveis de isolamento previstos pelo padrão SQL

Em Oracle, apenas os níveis READ COMMITTED e SERIALIZABLE são aceitos; em PostgreSQL, todos os quatro níveis podem ser enunciados, no entanto, o READ UNCOMMITTED opera da mesma maneira que o READ COMMITTED e o REPEATABLE READ opera com as mesmas restrições do SERIALIZABLE.

Transações em ORACLE

Modo de isolamento: para transações com atualizações

- READ COMMITTED (padrão):
 - LEITURA: a transação “vê” apenas dados consolidados (committed) antes do início de uma dada operação
 - ESCRITA: antes de uma operação, a transação **aguarda** até que quaisquer tuplas sendo atualizadas sejam liberadas e prossegue
- SERIALIZABLE:
 - LEITURA: a transação “vê” apenas dados modificados pela própria transação e dados efetivados antes do início da transação
 - ESCRITA: uma tupla é alterada (por outra transação) após o início da transação serializable, se a transação serializable tentar alterar esta mesma tupla, ela receberá a exceção:

ORA-08177: Can't serialize access for this transaction.

ou seja, o Oracle informa que não é capaz de tornar a concorrência semelhante a um processamento em série

Transações em ORACLE

Modo de isolamento: para transações com atualizações

- READ COMMITTED (padrão):
 - LEITURA: a transação “vê” apenas dados consolidados (committed) antes do início de uma dada operação
 - ESCRITA: uma tupla é alterada (por outra transação) após o início da transação serializável, ela
Ideia de Snapshot dos dados antes de uma operação – vários snapshots do banco.
- SERIALIZABLE:
 - LEITURA: a transação “vê” apenas dados modificados pela própria transação e dados efetivados antes do início da transação
 - ESCRITA: uma tupla é alterada (por outra transação) após o início da transação serializável, ela
Ideia de Snapshot dos dados antes de uma transação inteira – um único snapshot.

ou seja, o Oracle informa que não é capaz de tornar a concorrência semelhante a um processamento em série

Transação em ORACLE

Comando **SET TRANSACTION**

```
SET TRANSACTION ISOLATION LEVEL  
    { SERIALIZABLE | READ COMMITTED } |  
  
NAME 'nome_da_transacao';
```

Transação em ORACLE

SET TRANSACTION READ ONLY;

Solução anterior ao padrão Serializable: lê tudo que está consolidado antes do início da transação; não pode fazer qualquer escrita. Ignora alterações em paralelo.

Comandos de INSERT, UPDATE, e DELETE causam erro → previne escrita acidental.

SET TRANSACTION READ WRITE;

Padrão. Admite INSERT, UPDATE, e DELETE na transação.

Transações em ORACLE

- Transações READ-ONLY aceitam apenas os seguintes tipos de comandos:
 - SELECT INTO
 - OPEN
 - FETCH
 - CLOSE
 - LOCK TABLE
 - COMMIT
 - ROLLBACK
- Não permitem INSERT, UPDATE e DELETE

Transação em ORACLE

Comando **COMMIT**

- termina a transação
- torna permanente as ações da transação
- libera os recursos bloqueados

Transação em ORACLE

Comando **ROLLBACK**

- desfaz todas as operações da transação
- libera todos os recursos bloqueados
- termina transação

Views

Exemplo

Listar para cada aluno, a disciplina em que está matriculado, a turma (número e ano), o nome do professor que ministra a disciplina, e as informações da matrícula (nota e frequência).

CREATE VIEW Cursando AS

**SELECT a.Nome AS Nome_Aluno, a.NUSP AS Numero_Aluno,
d.Nome AS Nome_Disciplina, t.Numero AS Numero_Turma, t.Ano AS Ano_Turma,
p.Nome AS Nome_Professor,
m.Nota AS Conceito, m.FrequenciaPorc AS Freq**

FROM Matricula m

JOIN Aluno a ON m.Aluno = a.NUSP

JOIN Turma t ON m.Sigla = t.Sigla AND m.Numero = t.Numero AND m.Ano = t.Ano

JOIN Disciplina d ON t.Sigla = d.Sigla

JOIN Professor p ON d.Professor = p.NFunc;

→ SELECT * FROM Cursando

Visão (*View*)

- Representação de dados contidos em outras tabelas (tabelas base) ou mesmo em outras visões
- Trata resultado de uma consulta como uma tabela
 - **consulta armazenada**
 - **tabela virtual**
- Espaço de armazenamento (no dicionário de dados) apenas para a consulta (select) que define a visão
- Consulta é **executada cada vez que a visão é acessada**

Visão (*View*)

Utilidade:

- **segurança** - restrição de acesso a tuplas e colunas
- armazenamento de consultas complexas ou executadas com muita frequência
 - **simplicidade** para usuário
 - **abstração**
- **apresentação** dos dados com menor complexidade ou em diferentes perspectivas
- **isolamento de aplicações** em relação a alterações de esquema

Visão (*View*)

```
CREATE VIEW Cursando AS  
  
SELECT a.Nome AS Nome_Aluno, a.NUSP AS Numero_Aluno,  
d.Nome AS Nome_Disciplina, t.Numero AS Numero_Turma, t.Ano AS Ano_Turma,  
p.Nome AS Nome_Professor,  
m.Nota AS Conceito, m.FrequenciaPorc AS Freq  
  
FROM Matricula m  
JOIN Aluno a ON m.Aluno = a.NUSP  
JOIN Turma t ON m.Sigla = t.Sigla AND m.Numero = t.Numero AND m.Ano = t.Ano  
JOIN Disciplina d ON t.Sigla = d.Sigla  
JOIN Professor p ON d.Professor = p.NFunc;
```

Atributos ocultos

Tabelas ocultas

Visão (*View*)

```
CREATE VIEW Cursando AS  
SELECT      AS Nome_Aluno,  
            AS Nome_Disciplina,  
            AS Nome_Professor,  
            AS Conceito,  
            AS Numero_Aluno,  
            AS Numero_Turma, AS Ano_Turma,  
            AS Freq  
FROM
```

Atributos ocultos

Tabelas ocultas

Abstração

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class ConsultaCursando {
    public static void main(String[] args) {
        String url = "jdbc:oracle:thin:@localhost:1521:xe"; // Ajuste a URL conforme necessário
        String user = "seu_usuario"; // Substitua pelo nome de usuário do banco de dados
        String password = "sua_senha"; // Substitua pela senha do banco de dados

        String query = "SELECT a.Nome AS Nome_Aluno, a.NUSP, d.Nome AS Nome_Disciplina,
                           t.Numero, t.Ano, p.Nome AS Nome_Professor, m.Nota, m.FrequenciaPorc
                           FROM Matricula m JOIN Aluno a ON m.Aluno = a.NUSP
                           JOIN Turma t ON m.Sigla = t.Sigla AND m.Numero = t.Numero AND m.Ano = t.Ano
                           JOIN Disciplina d ON t.Sigla = d.Sigla
                           JOIN Professor p ON d.Professor = p.NFunc;";

        try (Connection conn = DriverManager.getConnection(url, user, password);
            PreparedStatement stmt = conn.prepareStatement(query);
            ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                String nomeAluno = rs.getString("Nome_Aluno");
                String nomeDisciplina = rs.getString("Nome_Disciplina");
                String nomeProfessor = rs.getString("Nome_Professor");
                System.out.println("Aluno: " + nomeAluno + ", Disciplina: " + nomeDisciplina + ", Professor: " + nomeProfessor);
            }
        } catch (SQLException e) {
            e.printStackTrace(); // Tratar possíveis erros de conexão e consulta
        }
    }
}
```

Abstração

Novo requisito para a base de dados:

→ **ALTER TABLE Aluno RENAME COLUMN Nome TO Nome_Sobrenome;**

O que fará com que a aplicação não funcione mais, exigindo que ela seja atualizada e recompilada.

Solução:

- 1) Atualizar a view para o novo nome de atributo;
- 2) Reescrever a aplicação de modo que ela não acesse o atributo Nome diretamente, mas sim a view.

```
CREATE or replace VIEW Cursando(Nome_Aluno, Numero_Aluno, Nome_Disciplina, Numero_Turma,
Ano_Turma, Nome_Professor, Conceito, Freq) AS
SELECT a.Nome, a.NUSP, d.Nome, t.Numero, t.Ano, p.Nome, m.Nota, m.FrequenciaPorc
FROM Matricula m
JOIN Aluno a ON m.Aluno = a.NUSP
JOIN Turma t ON m.Sigla = t.Sigla AND m.Numero = t.Numero AND m.Ano = t.Ano
JOIN Disciplina d ON t.Sigla = d.Sigla
JOIN Professor p ON d.Professor = p.NFunc;
```

Abstração

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class ConsultaCursando {
    public static void main(String[] args) {
        String url = "jdbc:oracle:thin:@localhost:1521:xe"; // Ajuste a URL conforme necessário
        String user = "seu_usuario"; // Substitua pelo nome de usuário do banco de dados
        String password = "sua_senha"; // Substitua pela senha do banco de dados

        String query = "SELECT Nome_Aluno, Nome_Disciplina, Nome_Professor FROM Cursando";
        try (Connection conn = DriverManager.getConnection(url, user, password);
            PreparedStatement stmt = conn.prepareStatement(query);
            ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                String nomeAluno = rs.getString("Nome_Aluno");
                String nomeDisciplina = rs.getString("Nome_Disciplina");
                String nomeProfessor = rs.getString("Nome_Professor");
                System.out.println("Aluno: " + nomeAluno + ", Disciplina: " + nomeDisciplina + ", Professor: " + nomeProfessor);
            }
        } catch (SQLException e) {
            e.printStackTrace(); // Tratar possíveis erros de conexão e consulta
        }
    }
}
```

Views no ORACLE

CREATE OR REPLACE VIEW *nome*

[(*NomeColuna* [, *NomeColuna* ...])]

AS <*select*>

[WITH CHECK OPTION | READ ONLY] ;

Escrita em views

Visões atualizáveis (*updatable*): preservam a chave

- inserção
- remoção
- atualização

Visões inerentemente NÃO atualizáveis:

- não preservam a chave
- operadores de conjunto (UNION, INTERSECT, MINUS...)
- operador DISTINCT
- GROUP BY (como parte da visão)
- ORDER BY..
- subconsulta na lista da cláusula SELECT
- *stored procedures*
- alguns casos de junções

Escrita em views

Preservação da chave:

```
CREATE VIEW AlunosAtivos AS  
SELECT NUSP, Nome, Idade  
FROM Aluno  
WHERE Idade > 18;
```

```
UPDATE AlunosAtivos SET Nome = 'João da Silva'  
WHERE NUSP = 1000000001;
```

➔ É possível determinar, exatamente, quais tuplas da tabela base serão atualizadas

Escrita em views

Sem preservação da chave:

```
CREATE VIEW NomeDosAlunos AS  
SELECT Nome AS NomeDoAluno  
FROM Aluno  
WHERE Idade > 18;
```

```
UPDATE NomeDosAlunos SET NomeDoAluno = 'João da Silva'  
WHERE NomeDoAluno LIKE 'J'
```

Alunos com Idade ≤ 18 , que não aparecem na view, também seriam atualizados, causando efeitos imprevisíveis; além disso, a ausência de chave levaria a uma busca sequencial para determinar a correspondência view-tabela.



→ Não é possível determinar, exatamente, quais tuplas da tabela base serão atualizadas

Views read-only

Opção para tornar a *view* *read-only*

```
create view view_disciplina as  
  select nome, sigla  
    from disciplina  
WITH READ ONLY;
```



Professor = {Nome, NFunc, Idade, Titulação}

Views CHECK OPTION

Exemplo

```
create or replace view professor_mestre as  
select * from professor  
where titulacao = 'Mestrado'
```

```
insert into professor_mestre values  
(999, 'Rogerio', 40, 'Doutorado');
```

 *Insere a tupla, mas a view não a mostra*

Views CHECK OPTION

WITH CHECK OPTION

- em visões atualizáveis, **WITH CHECK OPTION** não permite operações que violem a condição de seleção que define a visão

Professor = {Nome, NFunc, Idade, Titulação}

Views CHECK OPTION

Exemplo

```
create or replace view professor_mestre as  
select * from professor  
where titulacao = 'Mestrado'  
WITH CHECK OPTION;
```

```
insert into professor_mestre values  
(999, 'Rogerio', 40, 'Doutorado');
```



Visão Materializada (*materialized view*)

Visões **armazenadas** como tabelas

- dados provenientes de *master tables* (*tabelas base*)

Utilidade

- replicação de dados
- desempenho
 - *snapshot* local de dados remotos
 - armazenamento de resultados de consultas complexas e custosas
- armazenamento de informações sumarizadas
- **distribuição** de dados
- política de atualização automática

Visão Materializada (*materialized view*)

Comuns em *data warehousing*, sistemas distribuídos, computação móvel....

Principais desvantagens:

- ocupa espaço de armazenamento
- exige *refresh* quando as *master tables* são modificadas

Visões materializadas

- por *default* : *read-only*

Recursos Oracle de *Advanced Replication*

- permitem que as visões materializadas sejam atualizáveis

Visão Materializada no ORACLE

Tipos:

- Visões materializadas com agregações
- Visões materializadas apenas com junções
- Visões materializadas aninhadas

Disciplina = {Sigla, Nome, NCred, Professor, Livro}

Matrícula = {Sigla, Numero, Aluno, Ano, Nota}

Visão Materializada no ORACLE

Exemplo:

```
SELECT D.Sigla, count(M.Sigla) as  
           Nro_Matriculados  
FROM Disciplina D, Matricula M  
WHERE D.Sigla=M.Sigla  
GROUP BY D.Sigla;
```

Visão Materializada no ORACLE

-- logs nas tabelas *master* para o *refresh fast*

-- criados antes da visão

```
CREATE MATERIALIZED VIEW LOG ON Disciplina with ROWID;
```

```
CREATE MATERIALIZED VIEW LOG ON Matricula with ROWID;
```


Disciplina = {Sigla, Nome, NCred, Professor, Livro}

Matrícula = {Sigla, Numero, Aluno, Ano, Nota}

Visão Materializada no ORACLE

```
CREATE MATERIALIZED VIEW view_matriculados
BUILD IMMEDIATE
REFRESH FAST ON COMMIT
AS SELECT D.Sigla, count(M.Sigla) as
           Nro_Matriculados
FROM Disciplina D, Matricula M
WHERE D.Sigla=M.Sigla
GROUP BY D.Sigla;
```

➔ Checar a documentação Oracle para a sintaxe completa (rowids)

Visão Materializada no ORACLE

Refresh automático, exemplos:

/*Atualização automática completa de hora em hora*/

```
REFRESH COMPLETE
```

```
START WITH Sysdate NEXT SysDate + 1/24
```

/*Atualização automática incremental uma vez por dia*/

```
REFRESH FAST
```

```
START WITH Sysdate NEXT SysDate + 1
```

Refresh manual, exemplo:

```
EXECUTE DBMS_MVIEWS.REFRESH('nome_view', 'c');
```

c - complete

f - fast

Dicas

Para consultar informações do dicionário de dados, ou seja, consultar *views* do dicionário:

- tabelas
 - **SELECT * FROM user_tables**
- visões, atributos e colunas atualizáveis
 - **SELECT * FROM user_views**
 - **SELECT * FROM user_updatable_columns**

Documentação sobre dicionário de dados em *Oracle 11g Database Reference*

Conteúdo

Gerenciamento de Usuários no Oracle

- usuários
- papéis (atribuições)
- Privilégios

```
CREATE USER nome_usuario
```

```
    IDENTIFIED {BY senha | EXTERNALLY}
```

```
    DEFAULT TABLESPACE nome_tablespace
```

```
    TEMPORARY TABLESPACE nome_tablespace
```

```
    QUOTA {integer [K|M] | UNLIMITED} ON nome_tablespace
```

```
    PROFILE nome_profile
```

```
    PASSWORD EXPIRE
```

```
    ACCOUNT {LOCK | UNLOCK}
```

Usuários

Exemplo:

```
CREATE USER labbd  
  IDENTIFIED BY labbd  
  DEFAULT TABLESPACE users  
  QUOTA 10M ON users  
  ACCOUNT UNLOCK
```

Usuários

Exclusão:

```
DROP USER nome_usuario [CASCADE]
```

➔ **CASCADE:** todos os objetos do são removidos também

Privilégios, Papéis e Usuários

Concessão de privilégio e/ou papel:

- de sistema

```
GRANT privilegio [,privilegio,...] | papel  
TO usuario [,usuario,...] | papel | PUBLIC  
[WITH ADMIN OPTION]
```

- de objeto

```
GRANT privilegio [,privilegio,...] | papel  
ON objeto  
TO usuario [,usuario,...] | papel | PUBLIC  
[WITH GRANT OPTION]
```

Privilégios

Privilégio \Rightarrow autorização para que o usuário acesse e manipule objetos do BD.

Tipos:

- Sistema: permissão de executar ações no BD; DDL e gerenciamento
 - + de 100 tipos de privilégios distintos
 - ex: `create table`, `create session`, `create tablespace`, `create user`, `alter user`, ...
- Objeto: permissão para acessar e manipular objetos específicos
 - ex: `select on Aluno`, `insert on Aluno`,
`update (nusp) on Aluno ...`

Privilégios

Privilégios de sistema – criação, alteração e remoção de objetos de esquemas

- apenas em seu próprio esquema
 - ex: `create table, create view, create procedure, ...`
 - usuário é **dono** dos objetos que cria \Rightarrow possui permissão de alterar, remover, consultar, inserir, atualizar, executar, conceder privilégios sobre seus objetos, etc...
- em qualquer (any) esquema - dado por usuário DBA
 - ex: `create any table, alter any table, drop any table, select any table, insert any table, execute any procedure ...`

Usuários

Exemplo – privilégios de sistema:

Conectado como usuário dba:

```
GRANT CREATE SESSION, CREATE TABLE, CREATE  
SEQUENCE, CREATE SYNONYM TO labbd → apenas em seu  
próprio esquema
```

```
GRANT CREATE ANY TABLE, CREATE ANY SEQUENCE, CREATE  
ANY SYNONYM TO labbd → em qualquer esquema – grant  
precisa ser dado pelo DBA
```

Usuários

Exemplo – privilégios de objeto:

Conectado como usuário teste:

```
GRANT SELECT, UPDATE ON ALUNO TO labbd;
```

```
GRANT UPDATE, DELETE ON PROFESSOR TO labbd;
```

```
GRANT REFERENCES ON DISCIPLINA TO labbd;
```

```
//-----
```

```
GRANT UPDATE(ANO) ON TURMA TO labbd;
```

```
GRANT INSERT(SIGLA, NUMERO, ANO) ON MATRICULA TO labbd;
```

```
GRANT REFERENCES(ALUNO) ON ALUNO TO labbd;
```

```
GRANT ALL ON MATRICULA TO labbd
```

```
//-----
```

```
GRANT SELECT(NOME) ON PROFESSOR TO labbd;
```

→ não funciona em Oracle, usar views para essa finalidade

Usuários

GRANT ALL se refere a:

DELETE | INSERT | REFERENCES | SELECT | TRIGGER | UPDATE

Usuários

Exemplo – privilégios de objeto:

Conectado como usuário labbd:

```
SELECT * FROM teste.Aluno;
```

```
DELETE FROM teste.Matricula;
```

```
UPDATE teste.Professor set Nome = '';
```

Papéis (atribuições)

Papel: grupo de privilégios

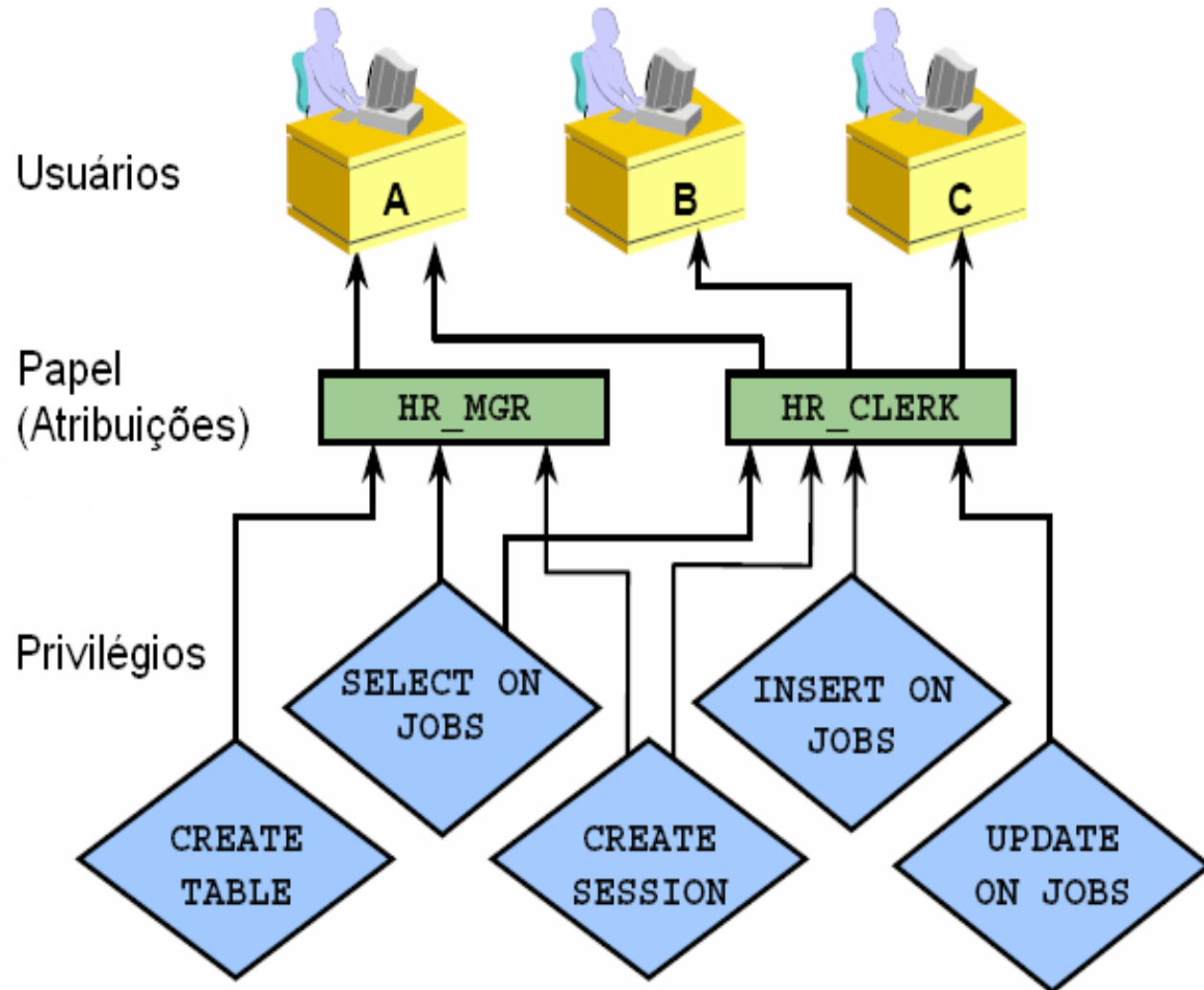
- simplifica a administração dos usuários

Criar papel:

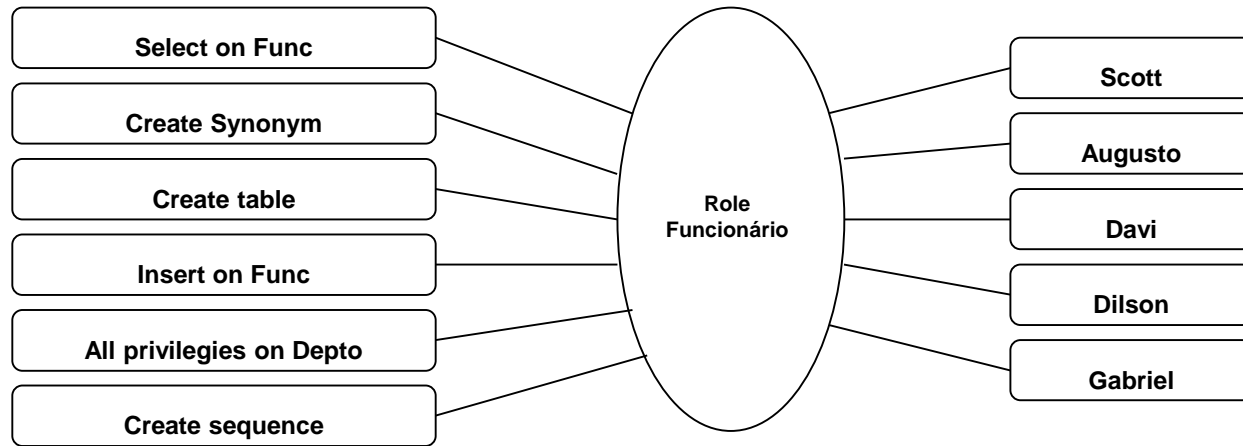
```
CREATE ROLE papel
```

Excluir papel:

```
DROP ROLE papel
```



Exemplo



Criar papel:

```
CREATE ROLE FUNCIONARIO;
```

Atribuir alguns privilégios ao papel:

```
GRANT CREATE SYNONYM, CREATE TABLE, CREATE SEQUENCE TO FUNCIONARIO
```

```
GRANT SELECT ON FUNC, INSERT ON FUNC, ALL ON DEPTO TO FUNCIONARIO;
```

Atribuir o papel ao usuário Scott:

```
GRANT FUNCIONARIO TO Scott, Augusto, Davi, Dilson, Gabriel;
```

Privilégios, Papéis e Usuários

Revogar privilégio e/ou papel:

```
REVOKE privilegio [,privilegio,...] | papel  
      [ON objeto]  
      FROM usuario [,usuario,...] | papel
```

Exemplo:

```
dba: REVOKE CREATE TABLE FROM labbd;
```

```
teste: REVOKE SELECT ON F1 FROM labbd;
```


Privilégios, Papéis e Usuários

WITH ADMIN OPTION

- opção para privilégios de sistema
- para usuários ou papéis
- Permite:
 - conceder/revogar o privilégio/papel para/de qualquer usuário ou papel
 - alterar ou remover o papel concedido
- ex:

-- conectado como usuário dba

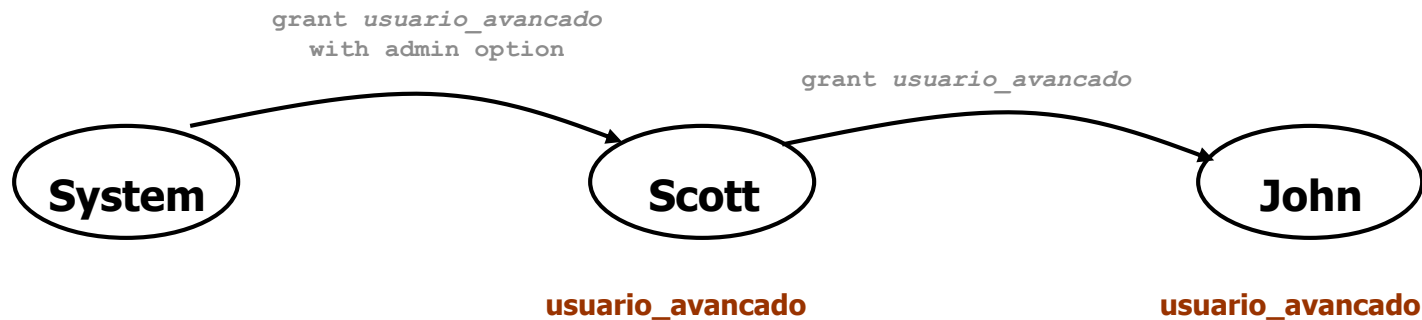
```
create role usuario_avancado;
```

```
grant create table to usuario_avancado;
```

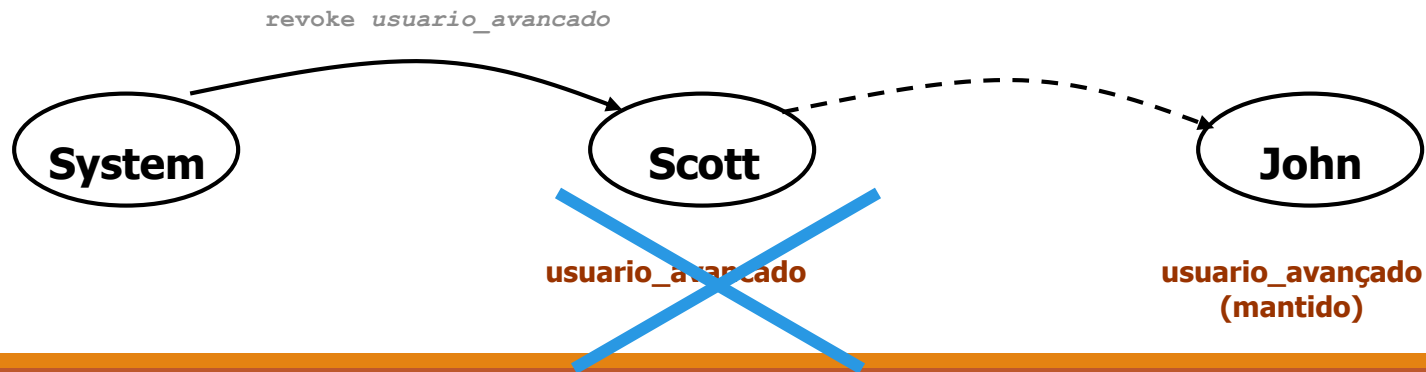
```
grant usuario_avancado to Scott WITH ADMIN OPTION;
```

-- conectado como Scott

```
grant usuario_avancado to John;
```



```
-- conectado como usuário SYSTEM  
revoke usuario_avancado from Scott;
```



Privilégios, Papéis e Usuários

WITH GRANT OPTION

- opção para privilégios de objetos
- somente para usuários
- permite ao usuário
 - conceder o privilégio/papel para qualquer usuário (com ou sem **GRANT OPTION**) ou papel

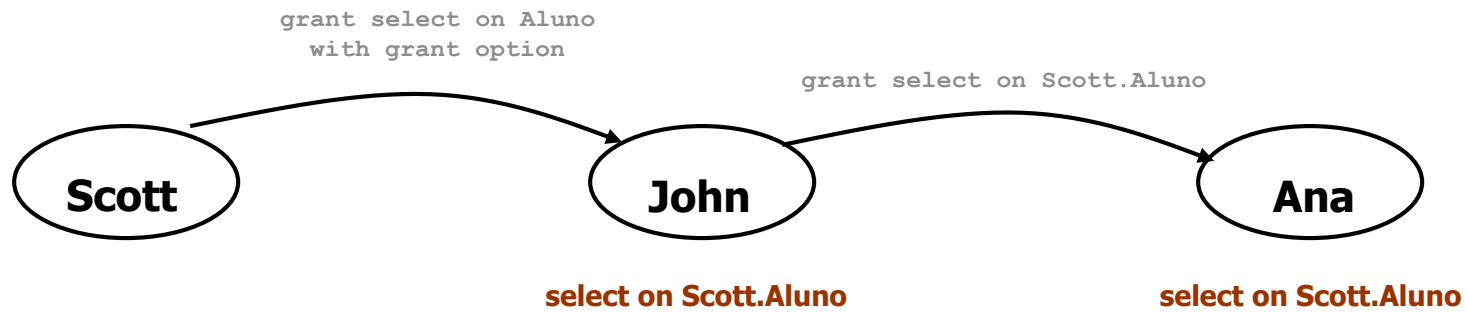
◦ ex:

-- conectado como *Scott* (dono da tabela *Aluno*)

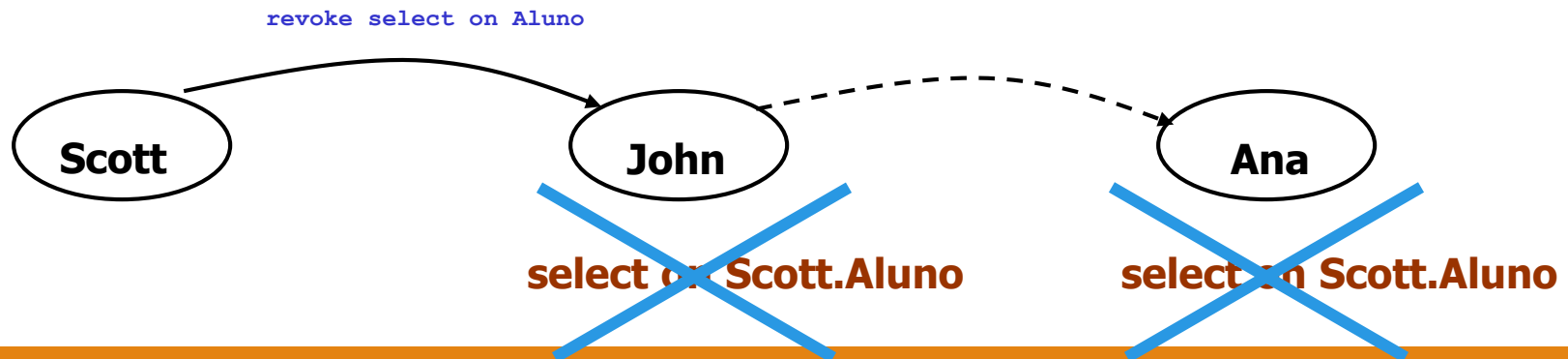
```
grant select on Aluno to John WITH GRANT OPTION;
```

-- conectado como *John*

```
grant select on Scott.Aluno to Ana;
```



```
-- conectado como usuário Scott  
revoke select on Aluno from John;
```



`grant select on Aluno
with grant option`

`grant select on Scott.Aluno`

No PostgreSQL, o REVOKE de objetos não é cascadeado.

~~`select on Scott.Aluno`~~

~~`select on Scott.Aluno`~~

Prática 6

