centro universitário FEI

AULA 15 – VIEWS E ROLLBACK

PROFA. DRA. LEILA BERGAMASCO

CC5232 – Banco de Dados

centro universitário FEI

NA AULA DE HOJE

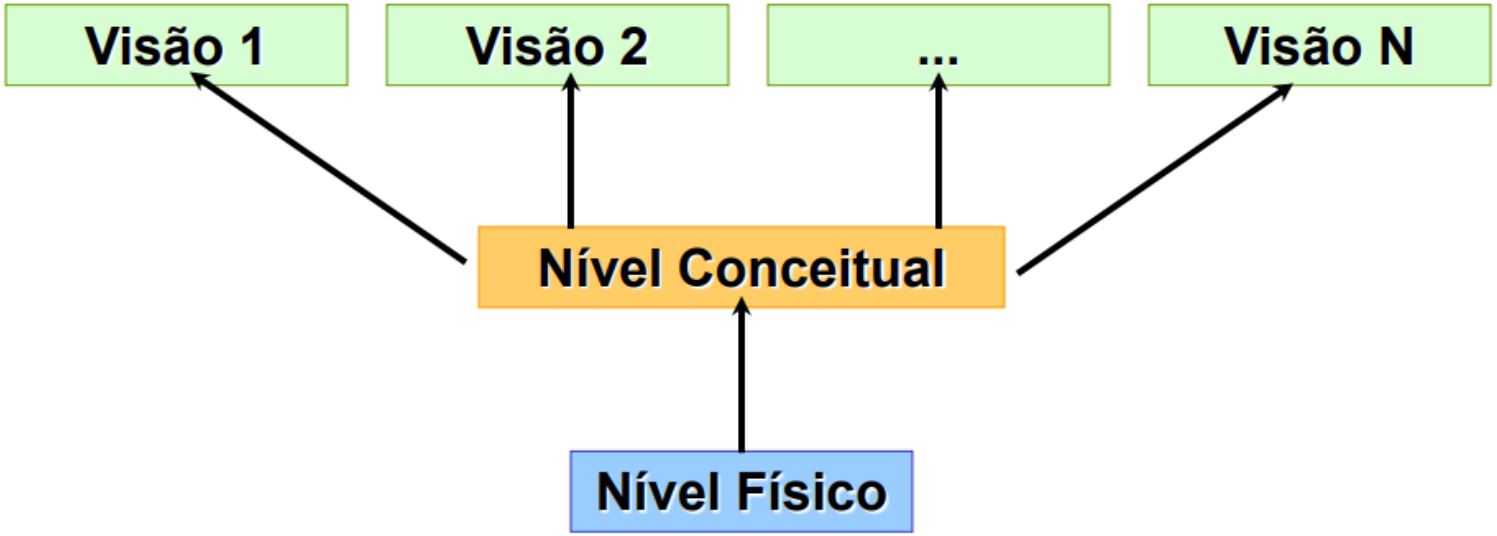
- Views e Rollback

VISÕES - VIEWS

- Nos exemplos mostrados nas aulas anteriores, consideramos o modelo lógico
  - relações usadas são aquelas reais, armazenadas no BD.
- Todos os usuários podem ver o modelo lógico inteiro?
  - NÃO !!!!!!!!!
  - Por exemplo: não posso permitir que qualquer funcionário saiba o salário de todos os funcionários da empresa!
- Qualquer relação que não seja parte do modelo lógico, mas que é visível a um usuário como uma relação virtual, é chamada de view.

VISÕES - VIEWS

- Segurança do BD
  - é desejável filtrar as informações do BD para que cada categoria de usuário somente "enxergue" a parte que lhe é permitida.
- Visões permitem esta diferenciação.



centro universitário FEI

VISÕES - VIEWS

- Comando create view
- Sintaxe

```
CREATE VIEW v AS <expressão>
```

- Onde:
  - v é o nome da view
  - <expressão> é uma consulta válida em SQL

centro universitário FEI

VISÕES - VIEWS

- Exemplos:

```
Conta (numero_conta, nome_cliente, saldo,codigo_agencia)
Agencia (codigo_agencia, nome_agencia, cidade)
Emprestimo (numero_emprestimo nome_cliente, valor, codigo_agencia)
```

- Criar uma visão com os clientes de cada agência.

```
CREATE VIEW Clientes_agencia AS
    (SELECT nome_agencia,nome_cliente
     FROM conta, agencia
     WHERE conta.codigo_agencia = agencia.codigo_agencia)
        UNION
    (SELECT nome_agencia,nome_cliente
     FROM emprestimo, agencia
     WHERE emprestimo.codigo_agencia = agencia.codigo_agencia)
```

VISÕES - VIEWS

- Visão definida

```
Clientes_agencia (nome_agencia, nome_cliente)
```

- Usando a visão Clientes_agencia criada, selecionar todos os clientes da agência Centro.

```
SELECT nome_cliente
FROM Clientes_agencia
WHERE nome_agencia= 'CENTRO'
```

VISÕES - VIEWS

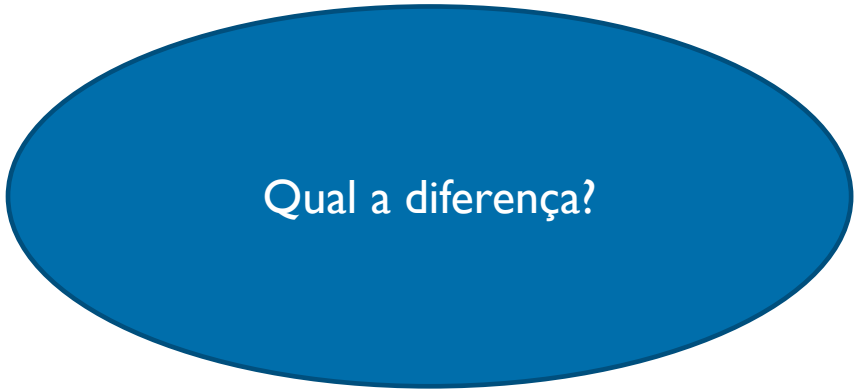
- Nomes de atributos de uma view podem ser especificados de forma explícita:

```
Agencia (codigo_agencia, nome_agencia, cidade)
Emprestimo (numero_emprestimo nome_cliente, valor, codigo_agencia)

CREATE VIEW emprestimo_total_agencia (nome_agencia, emprestimo_total) as
    (SELECT nome_agencia,sum(valor)
     FROM emprestimo,agencia
     WHERE emprestimo.codigo_agencia = agencia.codigo_agencia
     GROUP BY nome_agencia);
```

VISÕES - VIEWS

- A definição de visão é diferente da operação de criação de uma nova tabela a partir de outra.
- Exemplo:

```
CREATE TABLE empdep10 AS
    SELECT empno,ename,job
    FROM emp
    WHERE deptno=10;

CREATE VIEW vempdept10 AS
    SELECT empno,ename,job
    FROM emp
    WHERE deptno=10;
```



VISÕES - VIEWS

- Quando uma visão é definida, o SGBD armazena a sua definição propriamente dita, em vez do resultado da expressão que a gerou.

- Sempre que uma consulta é desenvolvida, a relação visão é recalculada.
- Vantagens e desvantagens?
  - view está sempre atualizada – mesmo que tabelas sejam modificadas;
  - view exige maior tempo de processamento.

VISÕES - VIEWS

- Nomes das views podem aparecer em qualquer lugar onde um nome de relação pode aparecer.
- Mas, como ficam as atualizações?
  - Em princípio, nenhuma operação de atualização pode ser executada em uma view.
  - Por que?
  - Devido à definição de visões, as atualizações podem gerar problemas no esquema do BD.
    - Atualização em uma visão reflete atualizações nas relações envolvidas.
    - Problema → atualizações indevidas ou incompletas

VISÕES - VIEWS

- `Conta (numero_conta, nome_cliente, saldo,codigo_agencia)`
- `Agencia (codigo_agencia, nome_agencia, cidade)`
- `Emprestimo (numero_emprestimo nome_cliente, valor, codigo_agencia)`

- `Clientes_agencia (nome_agencia, nome_cliente)`
- `INSERT INTO cc5232.clientes_agencia VALUES ('Baeta', 'Ana')`

- Relações envolvidas na definição da view:
  - Conta e Agencia

- Isto significa criar tuplas em conta e agência com alguns atributos nulos!

Conta: (nulo, “Manoel”,nulo,nulo)
Agencia: (nulo, “Acapulco”,nulo)

Problemático!!

VISÕES - VIEWS

- É possível criar novas visões a partir de visões já existentes.

```
Clientes_agencia (nome_agencia, nome_cliente)


CREATE VIEW Clientes_centro as
   (SELECT nome_cliente
    FROM Clientes_agencia
    WHERE nome_agencia ='CENTRO');
```

centro universitário FEI

EXPANSÃO DE VIEWS

- Ao processo de criar uma view a partir de outras views dá-se o nome de expansão de views.
- O processo considera que as definições de views não são recursivas:
  - nenhuma view é usada em sua própria definição – direta ou indiretamente.
- O que de fato acontece na expansão de views? O que o SGBD faz?

centro universitário FEI

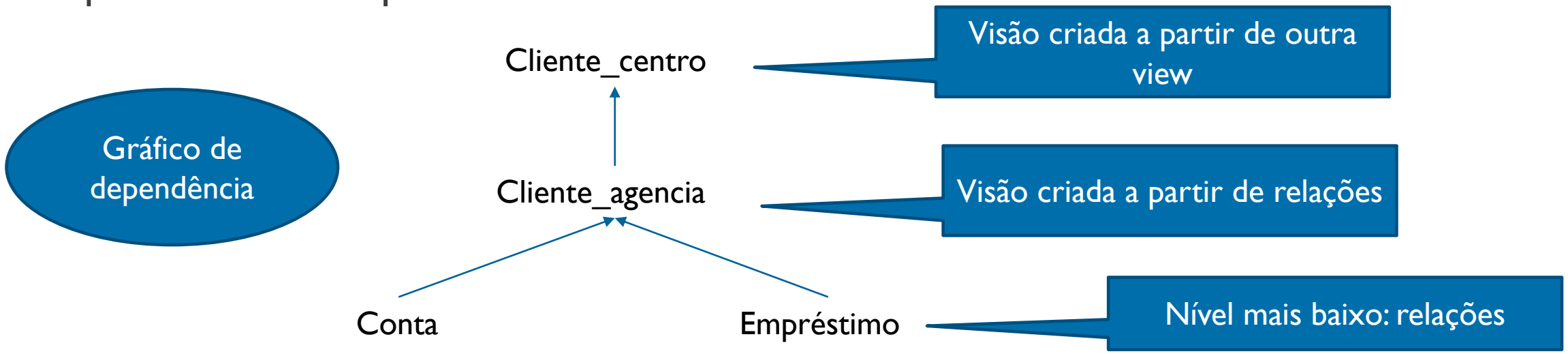
EXPANSÃO DE VIEWS

```
create view Clientes_centro as
       (select nome_cliente
       from Clientes_agencia
       where nome_agencia ='CENTRO');
```

```
create view Clientes_centro as
       (select nome_cliente from
       ((select nome_agencia,nome_cliente
       from conta, agencia
       where conta.codigo_agencia = agencia.codigo_agencia)
       UNION
       (select nome_agencia,nome_cliente
       from emprestimo, agencia
       where emprestimo.codigo_agencia= agencia.codigo_agencia))
       where nome_agencia ='CENTRO');
```

EXPANSÃO DE VIEWS

- Uma visão v1 é diretamente dependente de uma visão v2 se v2 é usada na expressão que define v1.
- Uma visão v1 é dependente de uma visão v2 se existir um caminho no gráfico de dependência de v2 para v1.



centro universitário FEI

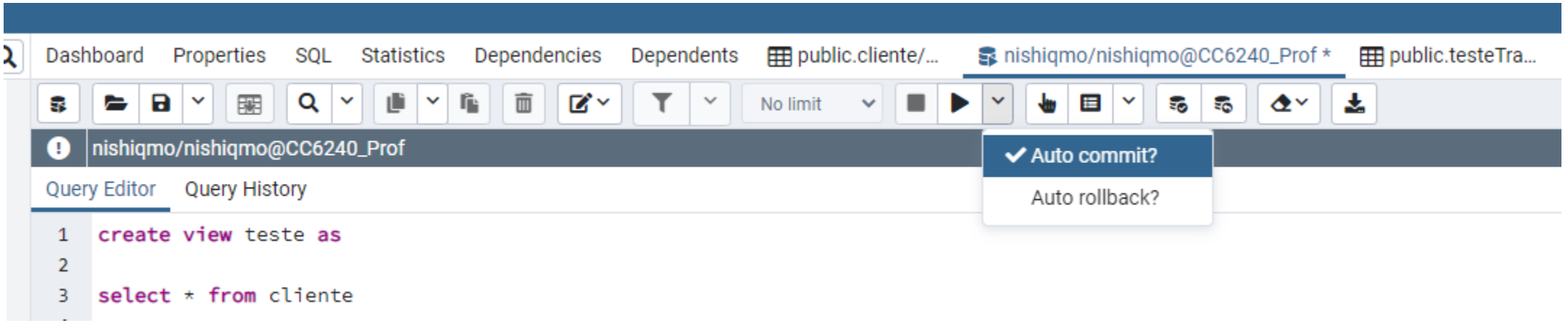
COMMIT/ROLLBACK

centro universitário FEI

COMMIT/ROLLBACK

- BEGIN;
  - Inicia uma transação. Tudo o que ocorrer entre o begin e o end (Postgres) poderá ser defeito
- END
  - Termina uma transação. No Postgres termina a transação e faz o commit!
- COMMIT
  - Concretiza os comandos dados anteriormente
- ROLLBACK
  - Desfaz qualquer update/delete feito

COMMIT/ROLLBACK



COMMIT/ROLLBACK

- Criei tabela testeTransaction (id)
- Desabilitei auto-commit

```
insert into public."testeTransaction" values (5);
select * from public."testeTransaction“;
rollback;
```

	id integer
1	1
2	2
3	3
4	4
5	5

	id integer
1	1
2	2
3	3
4	4

Atenção! Se houver mais consultas no editor o Postgres tentará executar e pode dar conflito

COMMIT/ROLLBACK

- Criei tabela testeTransaction (id)
- Desabilitei auto-commit

```
insert into public."testeTransaction" values (5);
select * from public."testeTransaction“;
end;
```

	id integer
1	1
2	2
3	3
4	4
5	5

	id integer
1	1
2	2
3	3
4	4
5	5

ERROR: current transaction is aborted, commands ignored until end of transaction block SQL state: 25P02

Executar commando end; e torcer para não ser nada grave

EXERCÍCIOS

- Considere os seguintes esquemas de relação:

```
Professor (número-prof, profnome, profrua, profcidade)
Aluno (número-aluno, alunome, alurua, alucidade)
Matrícula (número-aluno,código-disc,ano, nota)
Disciplina (código-disc,nome-disciplina,nome-curso, quantidade de aulas)
ProfDisc (código-disc,número-prof, ano)
```

1. Crie uma visão com todas as disciplinas oferecidas neste ano, constando os nomes das disciplinas, nomes e cidades dos professores responsáveis e nomes dos cursos das disciplinas que os professores ministram.
2. A partir da visão criada no exercício 1, mostre todos os nomes dos professores do curso X (escolha um curso cadastrado no seu BD) de um ano especifico e os nomes das disciplinas pelas quais são responsáveis.
3. A partir das visões criadas, mostre os nomes de todos os professores que não moram em São Paulo (ou outra cidade cadastrada) e que ministram alguma disciplina no curso utilizado no exercício anterior.
4. Crie uma visão com os dados dos professores (número e nome) e a quantidade total de aulas que ministram.
5. O que acontece com as visões criadas se atribuirmos a disciplina "Estruturas de Dados" para o professor X (escolha um professor cadastrado)? Quais visões seriam afetadas?
6. Crie um consulta delete em alguma tabela do seu banco de dados. Desabilite o auto commit, execute o comando e, em seguida faça um select * from X na tabela que foi modificada. Execute o comando rollback e faça o select novamente. Verifique o que aconteceu e execute o comando commit ao final

centro universitário FEI

OBRIGADO E ATÉ A PRÓXIMA AULA!