

Curso de Especialização:
Engenharia e Administração de Sistemas de Banco de Dados

Fundamentos de Sistemas de Banco de Dados



Transact SQL – DML

Profa. Dra. Gisele Busichia Baioco

gisele@ft.unicamp.br



UNICAMP



FACULDADE DE TECNOLOGIA



Conteúdo

- Alteração de Dados de Tabelas – Inserção, Atualização e Exclusão
- Consulta a Dados de Tabelas
- Gerenciamento de Transações
- Bibliografia



Alteração de Dados de Tabelas

Esquema Físico de Dados Exemplo

```
create table cliente
(
  codigo int not null,
  nome char(20) not null,
  endereco char(30) null,
  cidade char(15) null,
  cep char(8) null,
  uf char(2) null,
  cnpj char(20) null,
  ie char(20) null
  primary key(codigo)
)
```

```
create table vendedor
(
  codigo int not null,
  nome char(20) not null,
  salario_fixo money not null,
  faixa_comissao char(1) not null
  primary key(codigo)
)
```

```
create table pedido
(
  num_pedido int not null,
  prazo_entrega int not null,
  codigo_c int not null,
  codigo_v int not null
  primary key (num_pedido)
  foreign key (codigo_c)
  references cliente,
  foreign key (codigo_v)
  references vendedor
)
```

```
create table produto
(
  codigo int not null,
  unidade char(3) not null,
  descricao char(30) null,
  val_unit money not null
  primary key (codigo)
)

create table item_do_pedido
(
  num_pedido int not null,
  codigo_p int not null,
  quantidade int not null
  primary key (num_pedido,
  codigo_p)
  foreign key (num_pedido)
  references pedido,
  foreign key (codigo_p)
  references produto
)
```



Alteração de Dados de Tabelas

Inserção de Dados

- Comando INSERT:

- Adiciona registros (linhas) em uma tabela;

- Sintaxe:

```
insert [into] nome_tabela [(nome_coluna1 [, nome_coluna2, ...,  
    nome_colunan])]  
values (valor1 [, valor2, ..., valorn])
```

- Exemplos:

- Omitindo o nome das colunas:

- a inserção será em todas as colunas de uma linha;
- os valores devem ser especificados na ordem em que as colunas foram definidas na tabela. Por exemplo, inserção de um novo produto na tabela **produto**:

```
insert into produto  
values (108, 'kg', 'parafuso', 1.25)
```



Alteração de Dados de Tabelas

Inserção de Dados

■ Exemplos:

- Especificação explícita do nome das colunas – os nomes das colunas devem ser explicitamente especificados no INSERT quando:

a) existe uma ou mais colunas que permitem valores nulos e não deseja-se inserir dados nessas colunas. Por exemplo, considerando que a coluna *descricao* da tabela **produto** permite NULL:

```
insert into produto (codigo, unidade, val_unit)
values (109, 'cm', 1.50)
```

b) não se sabe ao certo a ordem em que colunas foram definidas na tabela e quer-se garantir a inserção correta. Por exemplo:

```
insert into produto (codigo, descricao, unidade,
val_unit)
values (110, 'elástico', 'm', 2.00)
```



Alteração de Dados de Tabelas

Inserção de Dados

- Inserção do resultado de um SELECT em uma tabela:

- Sintaxe:

```
insert [into] nome_tabela [(nome_coluna1 [, nome_coluna2, ...,  
    nome_colunan])]  
select nome_coluna1 [, nome_coluna2, ..., nome_colunan]  
from nome_tabela  
[where condição]
```

- Exemplo:

- cadastrar como clientes os vendedores cujo nome comece com a letra J:

```
insert into cliente (codigo, nome)  
select codigo, nome  
from vendedor  
where nome like 'J%'
```



Alteração de Dados de Tabelas

Atualização de Dados

- Comando UPDATE:

- Altera valores de colunas de tabelas;

- Sintaxe:

```
update nome_tabela  
set nome_coluna1 = valor1 [, nome_coluna2 = valor2, ...,  
    nome_colunan = valorn]  
[where condição]
```

- Exemplos:

- alteração do valor de uma coluna de uma ou mais linhas específicas da tabela. Por exemplo, alterar o valor unitário do produto 'parafuso' de 1.25 para 1.62:

```
update produto  
set val_unit = 1.62  
where descricao = 'parafuso'
```



Alteração de Dados de Tabelas

Atualização de Dados

■ Exemplos:

- alteração do valor de uma coluna para todos as linhas da tabela. Por exemplo, aumentar o salário fixo de todos os vendedores em 27%:

```
update vendedor  
set salario_fixo = (salario_fixo * 1.27)
```

- alteração do valor de mais de uma coluna da tabela ao mesmo tempo. Por exemplo, alterar o valor unitário do produto de código 108 de 1.62 para 2.30 e sua descrição de 'parafuso' para 'parafuso de aço':

```
update produto  
set val_unit = 2.00,  
descricao = 'parafuso de aço'  
where codigo = 108
```




Alteração de Dados de Tabelas

Exclusão de Dados

- Comando DELETE:

- Exclui registros (linhas) de uma tabela de um banco de dados;

- Sintaxe:

```
delete from nome_tabela  
[where condição]
```

- Exemplos:

- exclusão de um ou mais registros de uma tabela. Por exemplo, exclusão de todos os vendedores cujo salário fixo seja menos que 2500.00:

```
delete from vendedor  
where salario_fixo < 2500.00
```



Alteração de Dados de Tabelas

Exclusão de Dados

- Exemplos:

- CUIDADO: exclusão de todas as linhas de uma tabela. Por exemplo, exclusão de todas as linhas da tabela **cliente**:

```
delete from cliente
```

- Outra maneira de excluir todas as linhas de uma tabela é usando o comando **TRUNCATE TABLE**, que utiliza menos recursos do sistema e log de transações do que o **DELETE**, sendo mais rápido. Não pode ser executado em tabelas referenciadas por chaves estrangeiras. Por exemplo, exclusão de todas as linhas da tabela **item_do_pedido**:

```
truncate table item_do_pedido
```



Consulta a Dados de Tabelas

- Comando SELECT:

- Permite especificar quais as colunas (campos) de quais tabelas farão parte da consulta, quais critérios de consulta serão utilizados, qual a ordem de classificação, etc;

- Sintaxe simplificada:

Tabela a ser consultada

Colunas a serem retornadas a partir de uma ou mais tabelas

```
select coluna1, coluna2, ..., colunan  
from tabela  
[where ...]  
[group by ...]  
[having ...]  
[order by ...]
```

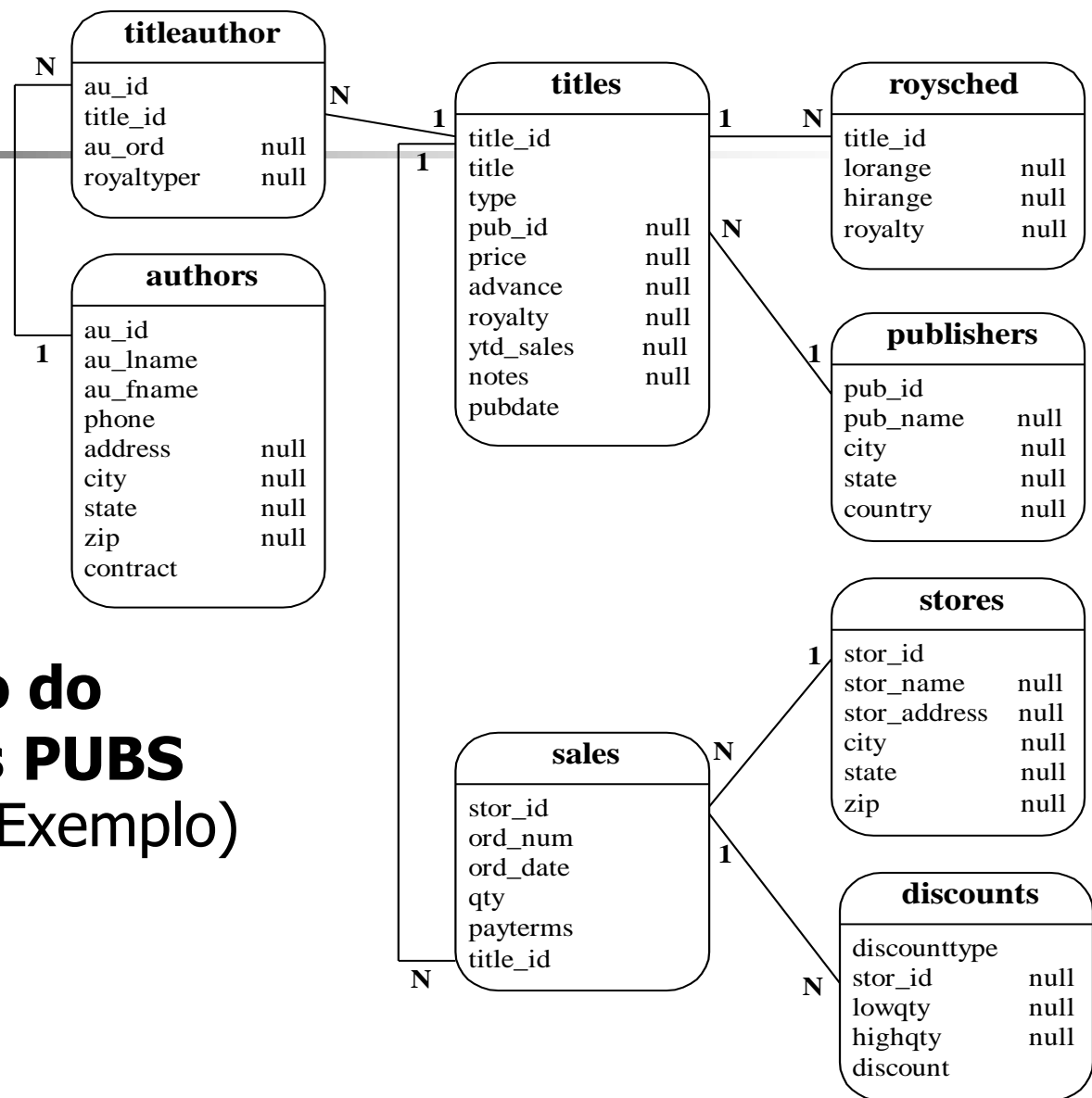
Especificação de critérios de consulta

Ordenação dos resultados com base em uma ou mais colunas

Agrupamento dos resultados em torno de uma ou mais colunas

Especificação de critérios sobre agrupamentos

Consulta a Dados de Tabelas



- **Esquema Lógico do Banco de Dados PUBS**
(Banco de Dados Exemplo)



Consulta a Dados de Tabelas

- **Escolha do banco de dados desejado:**

- Antes de iniciar uma consulta, deve-se escolher o banco de dados usando o comando:

```
use <nome_banco_de_dados>
```

- Exemplo:

```
use pubs
```

- **Consultas simples:**

- Consulta de todos os registros de uma tabela com todas as colunas:

```
select *  
from publishers
```

- Consulta de todos os registros de uma tabela com colunas específicas:

```
select pub_name, state  
from publishers
```



Consulta a Dados de Tabelas

- **Eliminação de linhas replicadas do resultado - opção *distinct*:**

```
select distinct city, state
from authors
```

- **Consultas qualificadas:**

- **Uso de operadores relacionais: >, <, >=, <=, =, !=**

```
select stor_name, state
from stores
where state = 'CA'
```

- ***between...and* e *not between...and***

```
select stor_id, stor_name
from stores
where stor_id between '6380' and '7100'
```

```
select stor_id, stor_name
from stores
where stor_id not between '6380' and '7100'
```



Consulta a Dados de Tabelas

■ Consultas qualificadas (continuação):

■ *in e not in*

```
select stor_name, city, state
from stores
where state in ('CA', 'UT')
```

■ *like e not like*

<pre>select stor_name from stores where stor_name like 'B%'</pre>	<pre>select stor_name from stores where stor_name not like 'B%'</pre>
<pre>select au_fname, au_lname from authors where au_fname like '%en%'</pre>	<pre>select stor_id, stor_name from stores where stor_id like '70_6'</pre>
<pre>select stor_name from stores where stor_name like '[CK]%'</pre>	<pre>select stor_name from stores where stor_name like '[A-E]%'</pre>



Consulta a Dados de Tabelas

- **Consultas com várias condições:**

- Uso de operadores lógicos: ***and*** e ***or***

```
select stor_name, city, state
from stores
where state = 'CA' and city = 'Fremont'
```

```
select stor_name, city, state
from stores
where state = 'CA' or city = 'Portland'
```

```
select title_id, type, advance
from titles
where (type = 'business' or type = 'psychology') and
      advance > 5000
```




Consulta a Dados de Tabelas

- **Atribuição de nomes às colunas da tabela-resultado:**

```
select au_fname 'Primeiro nome', au_lname Sobrenome
from authors
```

- **Uso de operadores aritméticos na consulta: +, -, *, /**

```
select title_id, advance + price
from titles
where type = 'business'
```

```
select title Titulo, (price * ytd_sales) Vendas
from titles
where price * ytd_sales > 80000 and type = 'business'
```

- **Tratamento de valor nulo: *is null* e *is not null***

```
select title, price
from titles
where price is null
```



Consulta a Dados de Tabelas

- **Ordenação de linhas da tabela-resultado:**

- Ordem ascendente – ***asc*** e ordem descendente – ***desc***. Quando nada for especificado o padrão é ordem ascendente:

```
select title, type, price
from titles
order by price
```

```
select title, type, price
from titles
order by type asc, price desc
```

```
select title, type, price
from titles
where type = 'business'
order by price desc
```

```
select title Titulo, (ytd_sales*price) Vendas
from titles
where type = 'business'
order by ytd_sales*price
```



Consulta a Dados de Tabelas

Uso de funções de agregação em consultas:

count(*)	Retorna o número de linhas selecionadas
count(nome_coluna)	Retorna o número de valores não-nulos em uma coluna
max(nome_coluna)	Retorna o maior valor da coluna
min(nome_coluna)	Retorna o menor valor da coluna
sum(nome_coluna)	Retorna a totalização dos valores da coluna
avg(nome_coluna)	Retorna a média dos valores da coluna

<pre>select count(*) from titles</pre>	<pre>select max(advance) from titles</pre>
<pre>select sum(ytd_sales) from titles where type = 'psychology'</pre>	<pre>select avg(price) from titles</pre>
<pre>select avg(price) from titles where type = 'business'</pre>	<pre>select avg(distinct price) from titles where type = 'business'</pre>



Consulta a Dados de Tabelas

- **Organização de dados em grupos:**

```
select type Gênero, avg(price) 'Preço Médio'
from titles
group by type
```

```
select title_id, stor_id, sum(qty)
from sales
group by title_id, stor_id
```

- Quando se usa a cláusula *where* as linhas são selecionadas antes do agrupamento:

```
select title_id, avg(price)
from titles
where advance < 3000
group by title_id
```



Consulta a Dados de Tabelas

- **Estabelecimento de condições sobre grupos:**

- **Funções de agregação não podem ser usadas na cláusula *where*;**
- Para aplicar condições a grupos usa-se a cláusula ***having***:

```
select type Gênero, avg(price) 'Preço Médio'
from titles
group by type
having avg(price) > 12.00
```

- **Produto cartesiano de tabelas – *cross join*:**

```
select type, pub_name
from publishers cross join titles
```

- **Operação de junção (*join*):**

- Relaciona tabelas, geralmente comparando os valores das chaves primárias e das chaves estrangeiras das tabelas, pois são essas colunas que indicam os **relacionamentos**.



Consulta a Dados de Tabelas

- **Junção Interior – *inner join*:**

- Relaciona duas tabelas e retorna apenas as linhas que satisfazem a condição de junção, ou seja, caso não exista relacionamento a linha não é retornada;

- Exemplo:

```
select stores.stor_id, stor_name, title_id, ord_num, qty
from sales inner join stores
      on sales.stor_id = stores.stor_id
```

- Usando *alias* (apelido) para o nome das tabelas:

```
select st.stor_id, stor_name, title_id, ord_num, qty
from sales sa inner join stores st
      on sa.stor_id = st.stor_id
```



Consulta a Dados de Tabelas

- **Junção Interior – *inner join*** (continuação):

- Junção contendo outros critérios:

```
select s.stor_id, stor_name, title_id, ord_num, qty
from sales sl inner join stores s
    on sl.stor_id = s.stor_id
where qty < 35
```

- Junção envolvendo mais de duas tabelas:

```
select au_fname, au_lname, title
from authors a inner join titleauthor ta
    on a.au_id = ta.au_id
inner join titles t
    on ta.title_id = t.title_id
```



Consulta a Dados de Tabelas

- **Junção Exterior – *outer join*:**

- Relaciona duas tabelas e retorna todas as linhas, mesmo quando não satisfazem a condição de junção, ou seja, caso não exista relacionamento *null* é retornado;

- ***left outer join*** – retorna todas as linhas da tabela à esquerda:

```
select stores.stor_id, stor_name, title_id, ord_num, qty
from stores left outer join sales
      on stores.stor_id = sales.stor_id
```

- ***right outer join*** – retorna todas as linhas da tabela à direita:

```
select sales.stor_id, stor_name, title_id, ord_num, qty
from stores right outer join sales
      on stores.stor_id = sales.stor_id
```

- ***full outer join*** – retorna todas as linhas de ambas as tabelas:

```
select stores.stor_id, sales.stor_id, stor_name, title_id,
      ord_num, qty
from stores full outer join sales
      on stores.stor_id = sales.stor_id
```




Consulta a Dados de Tabelas

■ Subconsultas:

- Uma subconsulta (*subquerie*) é um comando SELECT usado dentro de outro SELECT, ou seja, é a utilização de comandos SELECT aninhados;
- O comando SELECT mais interno é resolvido primeiro e seu resultado é utilizado pelo SELECT mais externo.
- Exemplo:

```
select title
  from titles
 where pub_id = (select pub_id
                  from publishers
                  where pub_name = 'New Moon Books')
```



Consulta a Dados de Tabelas

- **Subconsultas** (continuação):

- Uma subconsulta pode conter outras subconsultas:

```
select title
from titles
where title_id = (select title_id
                  from titleauthor
                  where au_id = (select au_id
                                from authors
                                where au_lname = 'Blotchet-Halls'))
```

- Se o resultado da subconsulta retornar mais de um valor (uma coluna e mais de uma linha), o comparador = precisa ser substituído por ***in***:

```
select distinct stor_id, title_id
from sales
where stor_id in (select stor_id
                  from stores
                  where state = 'CA')
```



Gerenciamento de Transações

Conceito de Transação

- O que é uma **transação**?

- Conjunto de várias operações a serem executadas em uma Base de Dados (BD), considerado uma única **unidade** lógica de execução do ponto de vista do usuário.

Início da transação

operação 1

operação 2

...

operação N

Fim da transação

- Sistemas de processamento de transações:

- Sistemas de BD com grande volume de dados;
- Centenas de usuários executando transações concorrentes;
- Exemplos: sistemas de reservas de passagens, bancos, controle de venda de produtos, entre outros.



Gerenciamento de Transações

Conceito de Transação

- Exemplos de transações:
 - Transferência de fundos de uma conta corrente para uma poupança: débito da conta corrente e crédito na poupança;
 - Controle de estoque: confirmação da venda de um produto e baixa no estoque desse produto;
 - etc.
 - Inserção/alteração em itens de dados resultantes de abstrações de generalização.

Observação: em todos esses casos, é essencial a conclusão de todo o conjunto de operações, ou que, no caso de uma falha, nenhuma operação ocorra.



Gerenciamento de Transações

Operações e Estados da Transação

- A fim de possibilitar o gerenciamento de transações são necessárias as seguintes operações:
 - BEGIN_TRANSACTION: marca o início da execução da transação;
 - READ(X): transfere o item de dado X da BD para um *buffer* na memória principal;
 - WRITE(X): transfere o item de dado X do *buffer* na memória principal de volta para a BD;
 - COMMIT_TRANSACTION: indica o término bem-sucedido da transação e as mudanças causadas por ela serão efetivadas;
 - ROLLBACK (ou ABORT): indica o término mal-sucedido da transação e as mudanças causadas por ela serão desfeitas;
 - END_TRANSACTION: marca o fim da execução da transação. Ainda deve ser verificado se a transação será efetivada ou abortada.



Gerenciamento de Transações

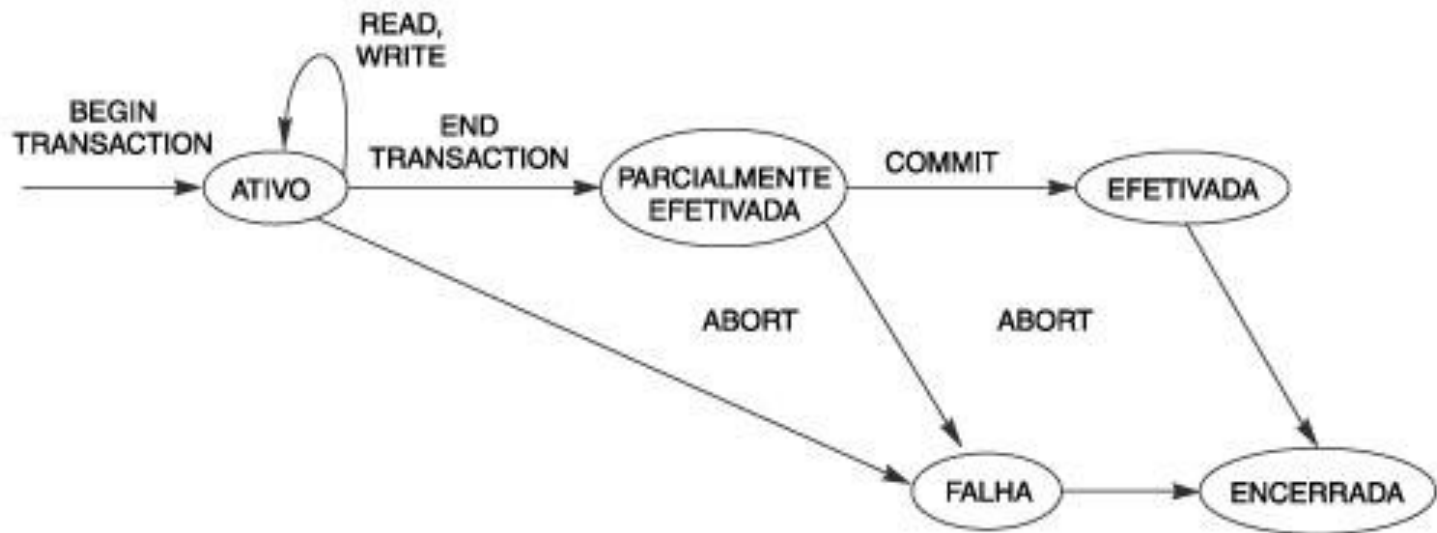
Operações e Estados da Transação

- Uma transação é dita **encerrada** se estiver efetivada ou abortada (em falha). Os seguintes estados podem ser considerados para uma transação, até que a mesma seja considerada encerrada:
 - ATIVO: a transação permanece nesse estado enquanto estiver executando;
 - PARCIALMENTE EFETIVADA: após a execução da última declaração;
 - EFETIVADA: após a conclusão com sucesso. Neste ponto as mudanças promovidas pela transação na BD são permanentes.
 - FALHA: após a descoberta de que a execução normal já não pode se realizar;
 - ENCERRADA: após a transação ter sido efetivada ou abortada.

Gerenciamento de Transações

Operações e Estados da Transação

- Diagrama de transição de estado ilustrando os estados de execução de uma transação:





Gerenciamento de Transações

Operações e Estados da Transação

- O *log* de transações:
 - a operação WRITE não resulta necessariamente na atualização imediata dos dados nas tabelas no disco. A alteração é salva primeiro em um arquivo de ***log de transações*** até que toda a transação seja completada.
 - ***log de transações*** \Rightarrow permite recuperar a BD a um estado consistente em caso de uma falha do sistema (falta de energia, falhas de hardware e erros de software).



Gerenciamento de Transações

Propriedades ACID

- Para assegurar a integridade dos dados, exige-se que o SGBD mantenha as **propriedades ACID das transações**:
 - **Atomicidade – uma transação é uma unidade atômica de processamento:** ou todas as operações da transação são refletidas corretamente na BD ou nenhuma o será;
 - **Consistência – a execução de uma transação preserva a consistência da BD:** a BD passa de um estado consistente para outro;
 - **Isolamento – uma transação deve ser executada como se estivesse isolada das demais:** a execução de uma transação não deve sofrer interferência de quaisquer outras transações concorrentes;
 - **Durabilidade – após efetivada, as mudanças na BD são persistentes** (mesmo se houver falhas no sistema).



Gerenciamento de Transações

Execução de Transações Concorrentes

- Considerando que transações:
 - podem executar concorrentemente;
 - podem acessar e alterar os mesmos itens de dados.

⇒ o SGBD deve controlar a interação entre as transações garantindo o **Isolamento**;
- Um **plano de execução** define a ordem de execução das operações das várias transações executadas concorrentemente;
- Duas operações em um plano são ditas em conflito e não podem ser intercaladas se satisfizerem as três condições seguintes:
 1. pertencerem a diferentes transações;
 2. acessarem o mesmo item de dado;
 3. pelo menos uma das operações ser um WRITE.



Gerenciamento de Transações

Execução de Transações Concorrentes

- Se a intercalação das operações não for permitida, existem apenas duas possibilidades, usando **planos de execução seriais**:
 - executar todas as operações de T_1 (em sequência), seguidas de todas as operações de T_2 (em sequência).
 - executar todas as operações de T_2 (em sequência), seguidas de todas as operações de T_1 (em sequência).



Gerenciamento de Transações

Controle de Transações Concorrentes

- Uma maneira muito utilizada por um SGBD para controlar transações concorrentes é desautorizar o compartilhamento pelo **bloqueio** (*lock*) dos dados recuperados para atualização;
- Quanto ao **controle**, os bloqueios podem ser:
 - **Implícitos:** controlados automaticamente pelo SGBD.
 - **Explícitos:** controlados por comandos emitidos ao SGBD a partir do programa aplicativo ou de uma consulta do usuário.
- Quanto ao **tipo**, um bloqueio pode ser:
 - **Exclusivo:** protege itens de dado contra o acesso de qualquer tipo, ou seja, nenhuma outra transação pode ler ou alterar os itens de dados.
 - **Compartilhado:** protege itens de dado contra alterações, mas não contra leitura, ou seja, outras transações podem ler os itens de dados contanto que não tentem alterá-los.



Gerenciamento de Transações

Controle de Transações Concorrentes

- A **granularidade** refere-se a quantos itens de dados serão afetados por um bloqueio. De menor para maior granularidade, tem-se:
 - bloqueio de registros (linhas);
 - bloqueio de páginas;
 - bloqueio de tabelas;
 - bloqueio da BD.
- Bloqueios de maior granularidade \Rightarrow mais fáceis de serem administrados pelo SGBD, mas causam mais conflitos.
- Bloqueios de menor granularidade \Rightarrow mais difíceis de administrar (existem mais detalhes para o SGBD controlar e verificar), mas os conflitos são menores.



Gerenciamento de Transações

Transações no SQL Server

- Considerando um SGBD Relacional, por exemplo, no SQL Server:
 - Transação: conjunto de comandos SQL que são tratados de maneira única no momento da execução.
 - O gerenciamento de transações pode ser didaticamente dividido em três partes:
 - *Log* de Transações;
 - Mecanismos de *Lock*;
 - Comandos para Controle de Transações.



Gerenciamento de Transações

Transações no SQL Server – *Log*

- O ***log* de transações** é necessário para que o SQL Server garanta a propriedade de **Durabilidade** das transações;
- O processo de recuperação automática do SQL Server usa o *log* de transações para efetivar transações completas ou desfazer transações incompletas;
- Funcionamento: todas as modificações realizadas nos dados são gravadas no *log* de transações antes que sejam efetivamente escritas na BD.



Gerenciamento de Transações

Transações no SQL Server – *Lock*

- Os **mecanismos de *Lock*** são necessários para que o SQL Server garanta a propriedade de **Isolamento** das transações;
- O compartilhamento é desautorizado pelo **bloqueio** dos dados recuperados para atualização;
- Bloqueio implícito: controle feito automaticamente pelo SQL Server;
- Granularidade dos bloqueios: páginas ou tabelas;
- Além dos tipos de bloqueios básicos – *exclusive* e *shared locks* – existem outros tipos necessários ao gerenciamento:
 - *livelock*: ocorre quando uma transação não consegue adquirir um *exclusive lock* até que todos os *shared locks* sejam liberados;
 - *demand lock*: evita que mais algum *shared lock* seja obtido em um conjunto de dados, uma vez que a próxima transação a ser executada necessita de um *exclusive lock* no mesmo conjunto de dados;
 - entre outros.



Gerenciamento de Transações

Transações no SQL Server – Comandos

- Os **comandos para o controle de transações** do SQL Server são necessários para que programador possa escrever as transações, garantindo a **Atomicidade** e a **Consistência**.
- O SQL Server opera com três modos de transações:
 - **Autocommit transactions:** cada operação individual (*select*, *insert*, *update*, *delete*) é considerada uma transação que será automaticamente efetivada ou desfeita;
 - **Explicit transactions:** cada transação deve ser explicitamente iniciada com o comando *begin transaction* e explicitamente terminada com um comando *commit* (efetiva) ou *rollback* (desfaz);
 - **Implicit transactions:** uma nova transação é implicitamente iniciada quando a transação anterior completa sua execução, mas cada transação deve ser explicitamente terminada com um comando *commit* (efetiva) ou *rollback* (desfaz).



Gerenciamento de Transações

Transações no SQL Server – Comandos

- Comandos básicos para controle de transações no SQL Server:
 - **begin tran[saction]** – marca o início de uma transação de maneira explícita;
 - **commit [tran[saction]]** – marca o final bem-sucedido de uma transação iniciada de maneira implícita ou explícita;
 - **rollback [tran[saction]]** – marca o final mal-sucedido de uma transação iniciada de maneira implícita ou explícita;
 - **save tran[saction]** – marca um potencial ponto de *rollback* (*savepoint*) dentro de uma transação.

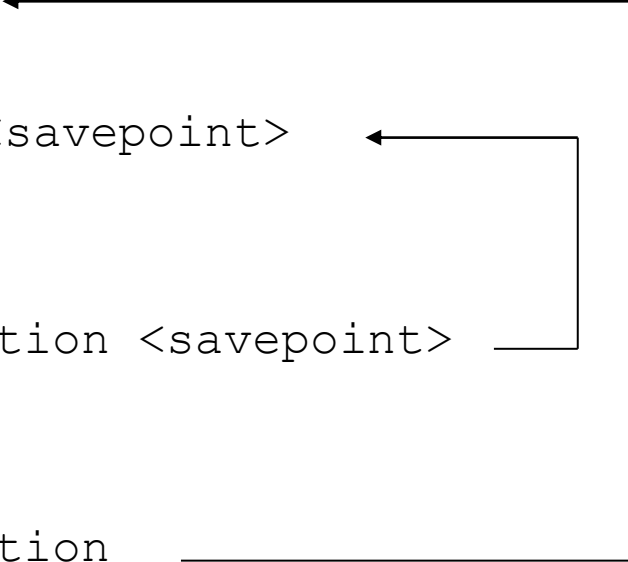


Gerenciamento de Transações

Transações no SQL Server – Comandos

- Esquema geral de definição de uma transação:

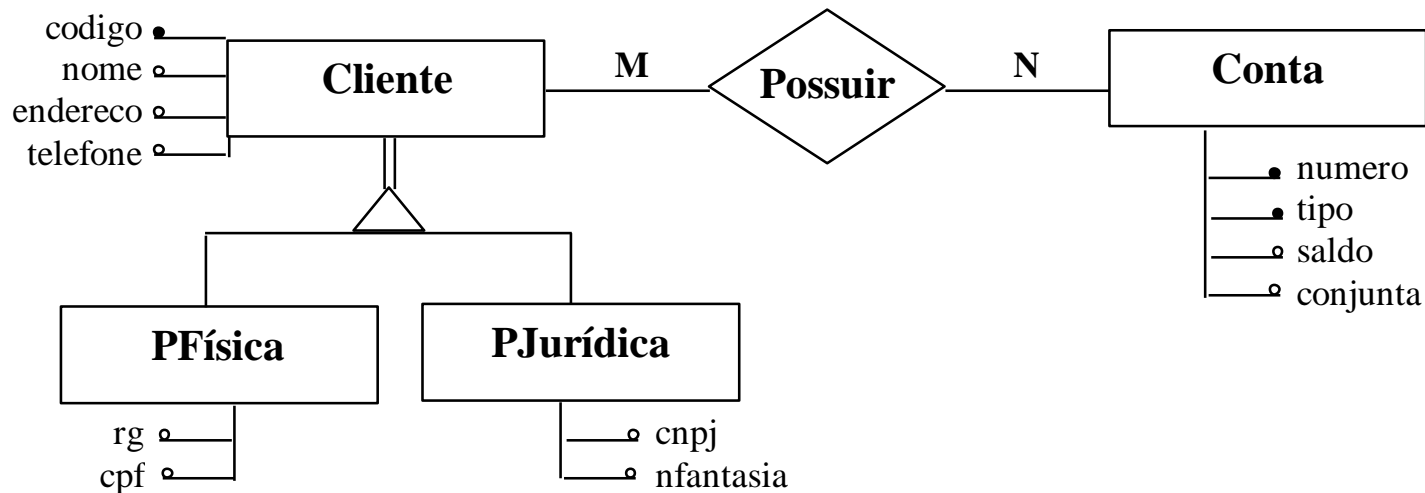
```
begin transaction
    <comandos sql 1>
save transaction <savepoint>
    <comandos sql 2>
if <condição1>
    rollback transaction <savepoint>
    <comandos sql 3>
if <condição2>
    rollback transaction
else
    commit transaction
```



Gerenciamento de Transações

Transações no SQL Server – Exemplos

- Considere o seguinte esquema conceitual da BD:





Gerenciamento de Transações

Transações no SQL Server – Exemplos

- Considere o seguinte esquema físico da BD:

```
create table cliente (  
  codigo numeric(10,0) not null,  
  nome char(30) not null,  
  endereco char(30) not null,  
  telefone numeric(10,0) null,  
  tipo tinyint not null,  
  primary key (codigo)  
)  
go
```

```
create table pfisica (  
  codigo numeric(10,0) not null,  
  rg char(10) not null,  
  cpf numeric(10,0) not null,  
  primary key (codigo),  
  foreign key (codigo)  
    references cliente  
)  
go
```

```
create table pjuridica (  
  codigo numeric(10,0) not null,  
  cnpj numeric(10,0) not null,  
  nfantasia char(30) not null,  
  primary key (codigo),  
  foreign key (codigo)  
    references cliente  
)  
go
```

```
create table conta (  
  numero numeric(8,0) not null,  
  tipo tinyint not null,  
  saldo money not null,  
  conjunta char(1) not null,  
  primary key (numero, tipo)  
)  
go
```

```
create table cliente_conta (  
  codigo numeric(10,0) not null,  
  numero numeric(8,0) not null,  
  tipo tinyint not null,  
  primary key(codigo, numero,  
  tipo),  
  foreign key (codigo)  
    references cliente,  
  foreign key (numero, tipo)  
    references conta  
)  
go
```



Gerenciamento de Transações

Transações no SQL Server – Exemplos

- **Transação 1:** Cadastro de cliente pessoa física.

```
begin transaction
    insert into cliente
    values (1, 'Jose da Silva', 'Av. 1 n.1111 Rio Claro-SP', 5554444, 0)
    if @@rowcount > 0 /* insercao de cliente bem sucedida */
    begin
        insert into pfisica
        values (1, '33333333', 123456789)
        if @@rowcount > 0 /* insercao de pessoa física bem sucedida */
            commit transaction
        else
            rollback transaction
    end
else
    rollback transaction
```



Gerenciamento de Transações

Transações no SQL Server – Exemplos

- **Transação 2:** Cadastro de uma conta corrente com 500.00 de saldo inicial, para o cliente de código 1.

```
begin transaction
    insert into conta
    values (1, 0, 500.00, 'N')
    if @@rowcount > 0 /* inserção de conta corrente bem sucedida */
    begin
        insert into cliente_conta
        values (1, 1, 0)
        if @@rowcount > 0
            commit transaction
        else
            rollback transaction
    end
else
    rollback transaction
```



Gerenciamento de Transações

Transações no SQL Server – Exemplos

- **Transação 3:** Transferência de 50.00 da conta corrente para a conta poupança número 1.

```
begin transaction
    update conta
    set saldo  = saldo + 50.00
    where numero = 1 and tipo = 1
    if @@rowcount > 0  /* credito em poupança bem sucedido */
    begin
        update conta
        set saldo  = saldo - 50.00
        where numero = 1 and tipo = 0
        if  @@rowcount > 0  /* debito em conta corrente bem sucedido */
            commit transaction
        else
            rollback transaction
    end
else
    rollback transaction
```




Bibliografia

ELMASRI, R.; NAVATHE, S. B., Fundamentals of database systems. 7 ed., Pearson, 2016.

SQL-DML

<https://docs.microsoft.com/pt-br/sql/t-sql/queries/queries?view=sql-server-ver15>

Insert, Update e Delete

<https://docs.microsoft.com/pt-br/sql/t-sql/statements/insert-transact-sql?view=sql-server-ver15>

<https://docs.microsoft.com/pt-br/sql/t-sql/queries/update-transact-sql?view=sql-server-ver15>

<https://docs.microsoft.com/pt-br/sql/t-sql/statements/delete-transact-sql?view=sql-server-ver15>

Select

<https://docs.microsoft.com/pt-br/sql/t-sql/queries/select-transact-sql?view=sql-server-ver15>

Transações

<https://docs.microsoft.com/pt-br/sql/t-sql/language-elements/transactions-transact-sql?view=sql-server-ver15>