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

Fundamentos de Sistemas de Banco de Dados

Transact SQL – Procedimentos Armazenados

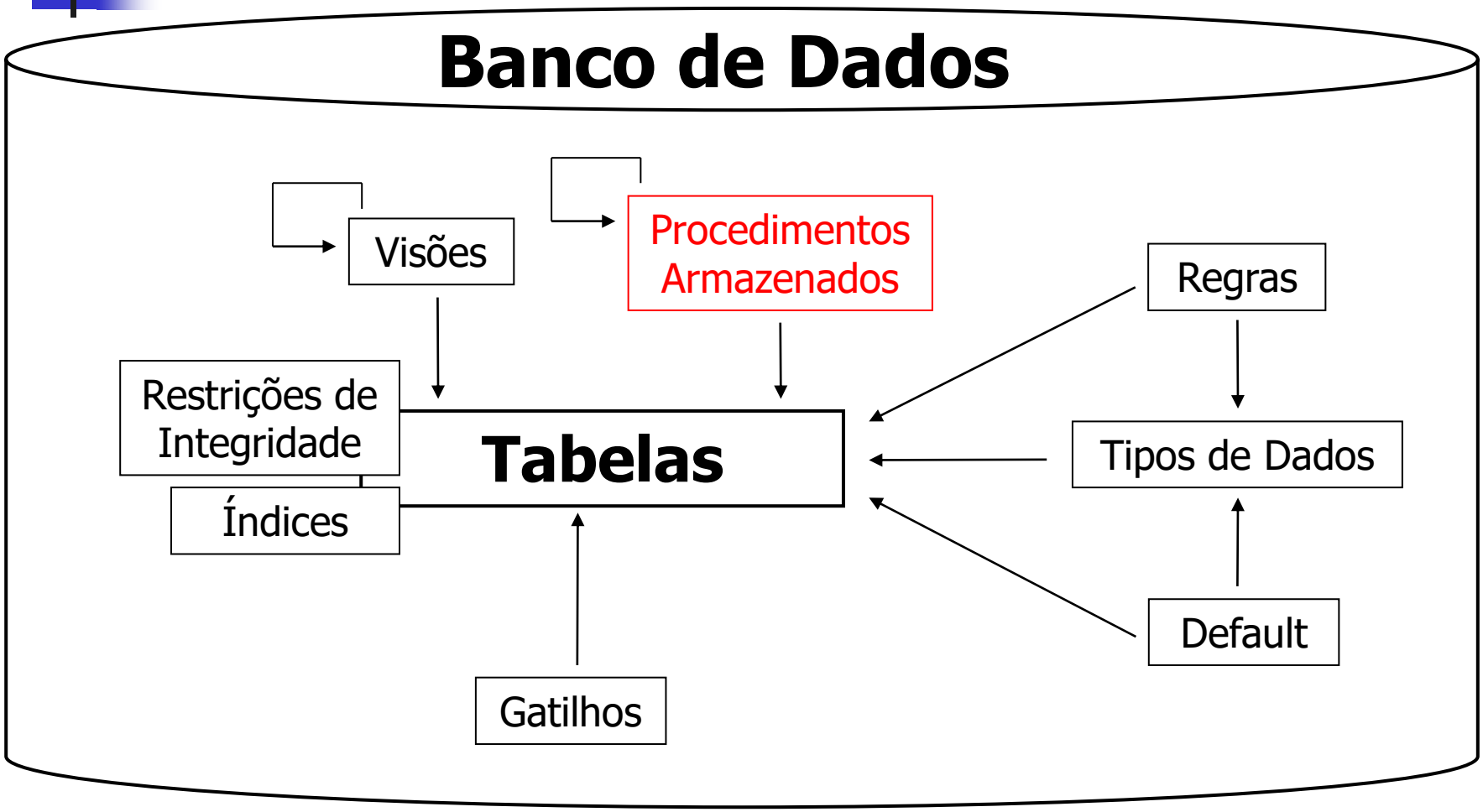
Profa. Dra. Gisele Busichia Baioco

gisele@ft.unicamp.br

UNICAMP

FACULDADE DE TECNOLOGIA

Procedimentos Armazenados – *Stored Procedures*



Procedimentos Armazenados – *Stored Procedures*

- **Definição:** um procedimento armazenado (*stored procedure*) é um conjunto de comandos SQL que são compilados e armazenados no servidor em determinada BD.
- **Vantagens:**
  - Encapsulamento de rotinas de uso frequente no próprio servidor, as quais estarão disponíveis para todas as aplicações: a parte lógica do sistema pode ser armazenada no próprio BD, em vez de ser codificada em cada aplicação;
  - Redução de erros: teoricamente todos os procedimentos armazenados devem ser previamente testados;
  - Segurança: o acesso aos dados da BD fica limitado à funcionalidade do procedimento armazenado, independente de quantas tabelas, visões ou outros procedimentos sejam acessados.

Procedimentos Armazenados
Como criar um procedimento usando T-SQL?

- Comando CREATE PROCEDURE;
- Sintaxe básica:

```
create proc[edure] nome_procedimento
[
@nome_parâmetro1 tipo,
...
@nome_parâmetroN tipo
]
as
comandos_SQL
```

Procedimentos Armazenados
Como criar um procedimento usando T-SQL?

- Exemplo:

Esquema físico de dados	Procedimento Armazenado
create table cliente (codigo numeric(10,0) not null, nome char(30) not null, endereco char(30) not null, telefone numeric(10,0) null, primary key (codigo)) go	create procedure buscacliente @nomebusca varchar(30) as select codigo, nome from cliente where nome like '%' + @nomebusca + '%'

Procedimentos Armazenados
Parâmetros

- Os parâmetros de procedimentos armazenados podem ser passados por valor ou referência:
  - A passagem por referência exige a colocação da palavra **OUTPUT** após o parâmetro desejado.
  - Exemplo:

```
create procedure buscacodigo
@codigobusca numeric(10,0),  /* por valor */
@nomecliente char(30) output /* por referência */
as
select @nomecliente = nome
from cliente
where codigo = @codigobusca
```

- Especificação de valores default para parâmetros:

```
create procedure buscacliente2
@nomebusca varchar(30) = '%'
as
select codigo, nome
from cliente
where nome like @nomebusca
```

Procedimentos Armazenados
Execução (Chamada)

- Comando EXEC (ou EXECUTE);
  - Exemplos:

```
exec buscacliente 'Si'
exec buscacliente2
```

- A passagem de parâmetros para um procedimento pode ser posicional ou nomeada.
  - Exemplos:

```
exec buscacliente 'Si'    /* posicional */
exec buscacliente @nomebusca = 'Si'  /* por nome */
```

  - Obs: uma vez iniciada a passagem de parâmetros por nome, deve ser continuada por nome.
- Na chamada de procedimentos com parâmetros passados por referência, a palavra **OUTPUT** deve seguir o parâmetro.
  - Exemplo:

```
declare @result char(30)    /* declara uma variável para receber o valor */
exec buscacodigo 1, @result output
print @result  /* exibe o valor da variável */
```

Procedimentos Armazenados
Valores de Retorno

- Comando RETURN;
- É permitido apenas valores de retorno do tipo inteiro;
- Exemplo:

```
create procedure buscacliente3
@nomebusca varchar(30)
as
select codigo, nome
from cliente
where nome like '%' + @nomebusca + '%'
return @@rowcount
/* execução */
declare @retorno int
exec @retorno = buscacliente3 'Si'
print @retorno
```

Procedimentos Armazenados
Como alterar um procedimento?

- Comando ALTER PROCEDURE;
- Sintaxe básica:

```
alter proc[edure] nome_procedimento
[
@nome_parâmetro1 tipo [= valor_default] [output],
...
@nome_parâmetroN tipo [= valor_default] [output]
]
as
comandos_SQL
```

- A sintaxe de alteração de procedimentos armazenados é similar a de criação;
- O comando ALTER PROCEDURE sobrepõe a definição da visão original, mantendo todas as permissões anteriormente atribuídas;
- Para alterar apenas o nome de um procedimento armazenado, deve-se utilizar o procedimento armazenado do sistema **sp_rename.**
  - Sintaxe:

```
sp_rename nome_velho, nome_novo
```

Procedimentos Armazenados
Como excluir um procedimento?

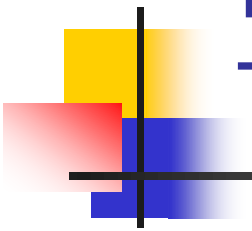
- Comando DROP PROCEDURE;
- Sintaxe:

```
drop procedure nome_procedimento1 [, …nome_procedimentoN]
```

- Exemplo:

```
drop procedure buscaclientenovo
```

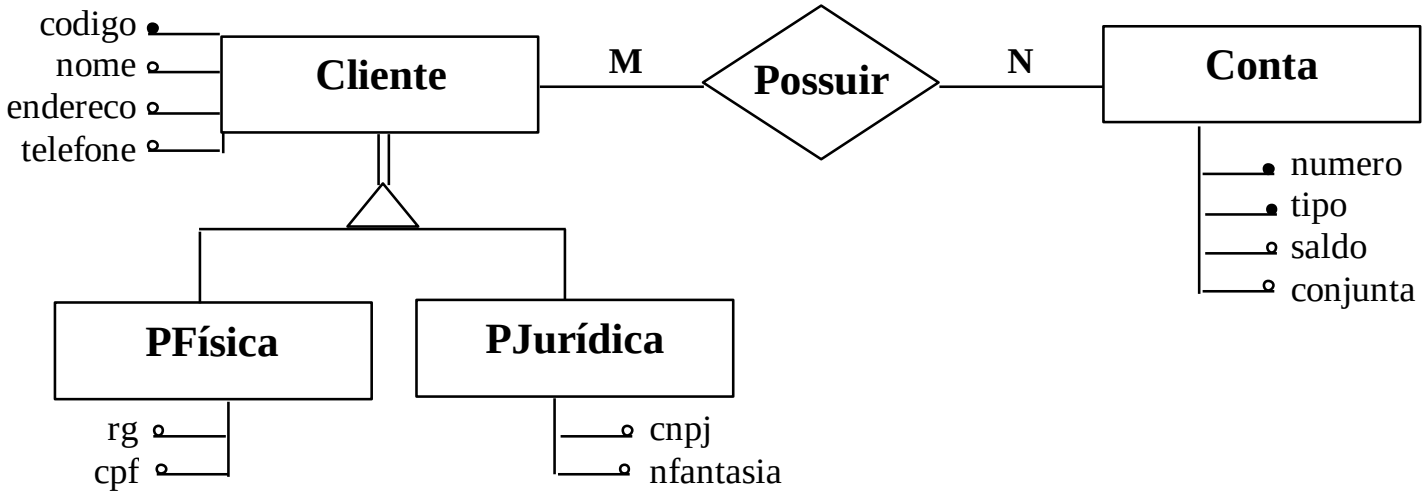
Procedimentos Armazenados
Transações e Aninhamento



- Transações podem ser especificadas dentro de procedimentos armazenados e o aninhamento de execução de procedimentos (chamada de procedimentos dentro de outros) é permitido.
  - Pode-se criar procedimentos para transações mais simples e utilizá-los dentro de procedimentos para transações mais complexas.

Procedimentos Armazenados
Transações e Aninhamento – Exemplos

- Considere o seguinte esquema conceitual da BD:



Procedimentos Armazenados
Transações e Aninhamento – 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
```

Procedimentos Armazenados
Transações e Aninhamento – Exemplos

- **Procedimento 1:**
  Transação de cadastro de pessoa física – retorna 1 se bem sucedida e 0 caso contrário.

```
create procedure ins_pessoa_fisica
@codigo numeric(10,0),
@nome char(30),
@endereco char(30),
@telefone numeric(10,0),
@rg char(10),
@cpf numeric(10,0)
as
begin transaction
insert into cliente
values (@codigo, @nome, @endereco, @telefone, 0)
if @@rowcount > 0  /* insercao de cliente bem sucedida */
begin
      insert into pfisica
      values (@codigo, @rg, @cpf)
      if  @@rowcount > 0  /* insercao de pessoa física bem sucedida */
      begin
            commit transaction
            return 1
      end
     else
     begin
           rollback transaction
           return 0
     end
end
else
begin
      rollback transaction
      return 0
end
```

Procedimentos Armazenados
Transações e Aninhamento – Exemplos

- Executando:

```
declare @ret int
exec @ret =  ins_pessoa_fisica 1, 'Antonio', 'Av.0, 111', 222222,
   '000000', 12121212
print @ret
```

- Observação: A variável @@rowcount recebe o valor 0 após a execução de qualquer comando que não retorne linhas, como, por exemplo, um *if*, um *print* entre outros. Assim, no caso do procedimento armazenado anterior, não é possível utilizar um comando *return @@rowcount* apenas no final, devido aos comandos *if* usados antes.

Procedimentos Armazenados
Transações e Aninhamento – Exemplos

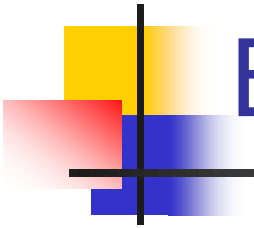
- **Procedimento 2:** Transação de cadastro de uma conta corrente ou poupança para um cliente, cujo código é passado como parâmetro – retorna 1 se bem sucedida e 0 caso contrário.

```
create procedure ins_conta
@codigo numeric(10,0),
@numero numeric(8,0),
@tipo tinyint,
@saldo money,
@conjunta char(1)
as
begin transaction
insert into conta
values (@numero, @tipo, @saldo, @conjunta)
if @@rowcount > 0 /* inserção de conta corrente bem sucedida */
begin
      insert into cliente_conta
      values (@codigo, @numero, @tipo)
      if @@rowcount > 0
      begin
            commit transaction
            return 1
     end
     else
     begin
            rollback transaction
            return 0
     end
end
else
begin
      rollback transaction
      return 0
end
```

Procedimentos Armazenados
Transações e Aninhamento – Exemplos

- **Procedimento 3:** Transação completa para o cadastro de um cliente pessoa física, uma conta corrente e uma conta poupança – retorna 1 se bem sucedida e 0 caso contrário:

```
create procedure trans_completa
@codigo numeric(10,0),
@nome char(30),
@endereco char(30),
@telefone numeric(10,0),
@rg char(10),
@cpf numeric(10,0),
@numero numeric(8,0),  /* atributo de conta */
@saldo_cc money,   /* atributo de conta corrente */
@saldo_p money,   /* atributos de conta poupança */
@conjunta char(1)  /* atributo de conta */
as
  declare @retorno int
  begin transaction
  exec @retorno = ins_pessoa_fisica @codigo, @nome, @endereco, @telefone, @rg, @cpf
  if @retorno > 0  /* inserção de cliente pessoa física bem sucedida */
  begin
    exec @retorno = ins_conta @codigo, @numero, 0, @saldo_cc, @conjunta
    if @retorno > 0  /* inserção de conta corrente bem sucedida */
    begin
      exec @retorno = ins_conta @codigo, @numero, 1, @saldo_p, @conjunta
      if @retorno > 0   /* inserção de conta poupança bem sucedida */
        commit transaction
      else      /* inserção de conta poupança mal sucedida */
        rollback transaction
    end
    else     /* inserção de conta corrente mal sucedida */
      rollback transaction
  end
  else
     rollback transaction
  return @retorno
```



Bibliografia

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

Procedimentos Armazenados no SQL Server

https://docs.microsoft.com/pt-br/sql/relational-databases/stored-procedures/create-a-stored-procedure?view=sql-server-ver15

https://docs.microsoft.com/pt-br/sql/t-sql/statements/create-procedure-transact-sql?view=sql-server-ver15

Linguagem de controle de fluxo de execução

https://docs.microsoft.com/pt-br/sql/t-sql/language-elements/control-of-flow?view=sql-server-ver15