Restrições de Integridade

Restrições de Integridade

- Objetivo primordial de um SGBD
  - garantir a integridade de dados
- Para garantir a integridade de um banco de dados
  - SGBDs oferecem o mecanismo de restrições de integridade
- *Uma restrição de integridade é uma regra de consistência de dados que é garantida pelo próprio SGBD*
  - *Precisa ser testada quando um registro é incluído, alterado ou excluído do BD*

Restrições de Integridade (RI)

- RI garantem que mudanças feitas no banco de dados por usuários autorizados não resultem na perda da consistência dos dados

Restrições de Integridade Básicas

- Restrição de vazio
- Restrições de domínio
- Restrições de chave primária
- Integridade referencial

Garantidas pelo SGBD

O programador Não se preocupa Com estas restrições

- Check constraints
- Gatilhos
- Asserções

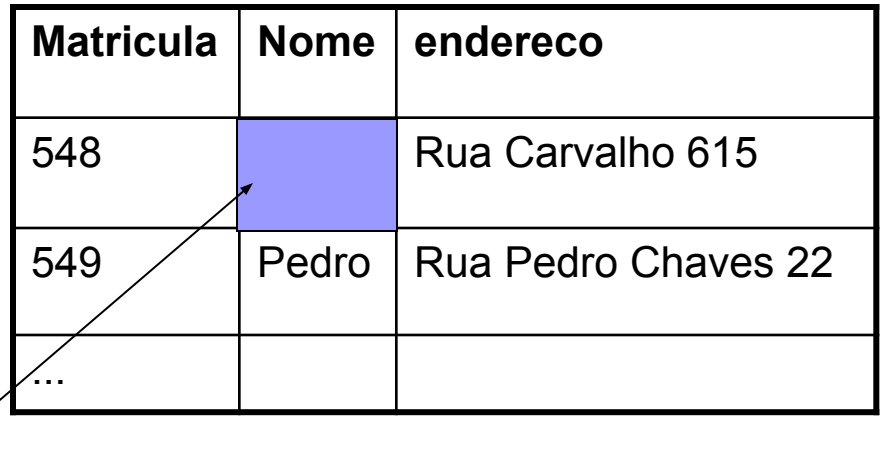
Restrições de Integridade Semântica

- Há muitas restrições de integridade que não se encaixam nas categorias básicas
  - Essas restrições são chamadas de restrições semânticas (ou regras de negócio)
- Exemplos de restrições semânticas:
  - Um empregado do departamento “Financeiro” não pode ter a categoria funcional “Engenheiro”.
  - Um empregado não pode ter um salário maior que seu superior imediato.
- Também chamadas de regras de negócio

Restrições de Vazio

Restrições de Valor Vazio

- O cliente 548 não tem nome
- Esta tupla se refere a um cliente anônimo, o que não tem muito sentido no BD
- Este pode ser um caso em que se deseja proibir valores vazios, restringindo o domínio do atributo nome para *not null*



Matricula	Nome	endereco
548		Rua Carvalho 615
549	Pedro	Rua Pedro Chaves 22
...		

Restrições de Valor Vazio

- Um valor de campo pode assumir o valor vazio ("null" em inglês)
  - Colunas nas quais não são admitidos valores vazios chamadas de colunas obrigatórias
  - Colunas nas quais podem aparecer valores vazios chamadas de colunas opcionais
- Abodagem relacional
  - todas colunas que compõem a chave primária devem ser obrigatórias
  - demais chaves podem conter colunas opcionais

Restrições de Valor Vazio

- Regra “Nulo”
  - **Permite**, **ou não**, que um atributo de uma tabela tenha valor nulo (ausência de valor)

- Exemplo em SQL:
  - ```
    Create table funcionario
      (matricula integer not null,
      nome varchar(30) not null,
      telefone varchar(20))
    ```

- ```
  Insert into funcionario values(568, ´´, ´48-33542519´)
  ```
  - ```
    postgreSQL Erro: Null value in column “nome” violates not-null constraint
    ```
    ```

Restrições de Domínio

Restrições de Domínio

- Refere-se ao domínio de um atributo
  - Conjunto de valores que podem aparecer em uma coluna (atributo)
  - Domínio de valores válidos para um atributo
- Restrições de domínio são as mais elementares
  - Facilmente verificadas pelo sistema

Restrições de Domínio

- Similar aos tipos de variáveis em linguagens de programação
- Vários atributos podem ter o mesmo domínio, mas tem casos em que não faz sentido
- Exemplo: o atributo ***idade*** é numérico, precisa ser de domínio **inteiro,** e não do tipo character, como é o caso do atributo **nome**

```
Create table funcionario
(matricula integer not null,
nome varchar(30) not null,
idade integer,
endereco varchar(35))
```

Restrições de Domínio

- O padrão SQL suporta um conjunto restrito de tipos de domínio:
  - Cadeia com comprimento de caracteres fixo, com comprimento especificado pelo usuário
  - Número de casas decimais
  - Inteiro (conjunto finito de números inteiros)
  - Data
  - ...

```
Create table funcionario
 (matricula integer not null,
  nome varchar(30) not null,
  dataNascimento date,
  endereco varchar(35))
```

Insert into funcionario values (***5AB***, "Paulo", ***20/15/1999***, "Av Ipiranga 1900")

Matricula	Nome	dataNascimento	endereco
548	Maria	25/02/1973	Rua Carvalho 615
549	Pedro	14/06/1990	Rua Pedro Chaves 22

Restrições de Chave

Restrições de Chave - Primária

- Regra “chave primária”
  - restringe que cada linha de uma tabela deve ser identificada por um valor único
- Pode ser simples ou composta

Chave simples

```
Create table medico
     (codigoM integer  not null,
      nome varchar(30)  not null,
      endereco varchar(35),
      PRIMARY KEY (matricula) )
```

chave composta

```
Create table consulta
     (codigoMedico integer  not null,
      codigoPaciente integer  not null,
      data date  not null,
      PRIMARY KEY (codigoMedico,
     codigoPaciente, data))
```

Restrições de Chave – Chave Candidata

- Restrições ***Unique*** garantem que os dados contidos em uma coluna ou um grupo de colunas é único em relação a todas as linhas da tabela

  - ***Sintaxe: quando escrita como uma restrição de coluna***

```
CREATE TABLE produto (nroProduto integer UNIQUE,
                      nome varchar(30), preco real);
```

  - ***Sintaxe: quando escrita como uma restrição de tabela***

```
CREATE TABLE produto (nroProduto integer,
                      nome varchar(30), preco real,
                      UNIQUE (product_no));
```

Restrições de Integridade Referencial

Restrições de Integridade Referencial

- Uma das restrições mais importantes em BD
- **Definição**: é a garantia de que um valor que aparece em uma relação R1, para um conjunto de atributos, deve obrigatoriamente corresponder a valores de um conjunto de atributos em uma relação R2; OU
  - valores de atributos que são **"chave estrangeira" em uma relação R1** possuem valores correspondentes em **chaves primárias** da tabela referenciada R2

Restrições de Integridade Referencial

- Exemplo

```
CREATE TABLE cidade
(codigoCidade integer NOT NULL,
 descricao varchar(40) NOT NULL,
 estado char(2),
PRIMARY KEY (codigoCidade))
```

```
CREATE TABLE cliente
(codigoCliente integer NOT NULL,
 nome varchar(30) NOT NULL,
 codCidade integer,
 PRIMARY KEY (codigoCliente),
 FOREIGN KEY (codCidade)  REFERENCES Cidade (codigoCidade))
```

Cliente

codigoCliente	Nome	endereco	codigoCidade
548	Maria	Rua Carvalho 615	1
549	Pedro	Rua Pedro Chaves 22	5

Viola a restrição cidade 5 não existe

Cidade

codigoCidade	Descricao	Estado
1	Florianópolis	SC
2	São José	SC

A restrição garante que não irá existir CLIENTE que more numa cidade que não exista na tabela CIDADE

Restrições de Integridade Referencial

- A restrição de integridade é testada quando:
  - Inclusão: se uma tupla t2 é inserida em uma relação r2, o sistema precisa assegurar que existe uma tupla t1 em uma relação r1 tal que t1[r1]=t2[r2]
    - Ex: inclui novo cliente, testa se a cidade existe
  - Exclusão: Uma chave primária referenciada **é removida**
    - ON DELETE
      - Ex: Remove uma cidade referenciada por algum cliente
  - Alteração: Uma chave primária referenciada é alterada
    - ON UPDATE
      - Ex: Altera a chave primaria da cidade referenciada em cliente

Restrições de Integridade Referencial

- AÇÕES:
  - NÃO permite alteração ou exclusão (**NO ACTION** (default)):
    - não permite a exclusão/alteração enquanto houver dependência;
      - Ex: só permite excluir a cidade quando nenhum cliente referenciar esta cidade
  - **SET DEFAULT** : se houver um valor default para a coluna da chave estrangeira, ela recebe este valor
  - **CASCADE** : propaga a exclusão/alteração;
  - **SET NULL** : atribui o valor nulo.

Restrições de Integridade Referencial - INCLUSÃO

- **Inclusão**: ao inserir um novo cliente, é preciso garantir que o código da cidade na tabela **cliente** EXISTA na tabela **cidade**
- Para garantir isso cria-se a tabela de cliente com a chave estrangeira **codCidade**

```
CREATE TABLE cliente
(codigoCliente integer NOT NULL,
 nome varchar(30) NOT NULL,
 codCidade integer,
 PRIMARY KEY (codigoCliente),
 FOREIGN KEY (codCidade) REFERENCES Cidade (codigoCidade))
```

Cliente

codCliente	Nome	endereco	codCidade
548	Maria	Rua Carvalho 615	1
549	Pedro	Rua Pedro Chaves 22	2

Cidade

codCidade	Descricao	Estado
1	Florianópolis	SC
2	São José	SC

Restrições de Integridade Referencial - EXCLUSÃO

- Exclusã**o**: ao excluir uma cidade da tabela **CIDADE**, o SGBD precisa garantir que não exista nenhum cliente na tabela CLIENTE referenciando esta cidade
- várias opções para garantir a integridade:

(opção 1) Setar para nulo o codigo da cidade na tabela CLIENTE

```
CREATE TABLE cliente
 (codigoCliente integer NOT NULL, nome varchar(40) NOT NULL, codigoCidade integer,
  PRIMARY KEY (codigoCliente),
  FOREIGN KEY (codigoCidade) REFERENCES Cidade (codigoCidade) ON DELETE SET NULL))
```

(opção 2) assumir um valor default

```
CREATE TABLE cliente
 (codigoCliente integer NOT NULL, nome varchar(40) NOT NULL, codigoCidade integer,
  PRIMARY KEY (codigoCliente),
  FOREIGN KEY (codigoCidade) REFERENCES Cidade (codigoCidade) ON DELETE SET DEFAULT))
```

Cliente

codCliente	Nome	endereco	codCidade
548	Maria	Rua Carvalho 615	NULL
549	Pedro	Rua Pedro Chaves 22	2

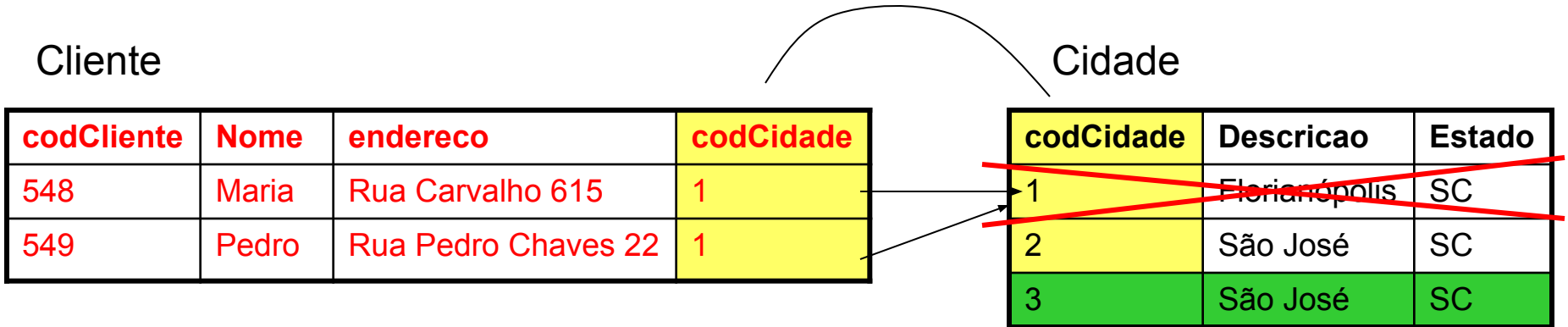
Cidade

codCidade	Descricao	Estado
1	Florianópolis	SC
2	São José	SC
3	São José	SC

Restrições de Integridade Referencial - EXCLUSÃO

(opção 3) NÃO permitir a exclusão da cidade 1 porque tem clientes morando nesta cidade. A cidade 3 pode ser excluída

```
CREATE TABLE cliente
 (codCliente integer NOT NULL, nome varchar(40) NOT NULL,endereco varchar(40), codCidade
   integer,
  PRIMARY KEY (codCliente),
  FOREIGN KEY (codCidade) REFERENCES Cidade (codCidade) ON DELETE RESTRICT))
```



Restrições de Integridade Referencial - EXCLUSÃO

(opção 4) remove as referencias (remove a cidade e todos os clientes da cidade)

```
CREATE TABLE cliente
 (codCliente integer NOT NULL, nome varchar(40) NOT NULL, endereco varchar (40),
    codCidade integer,
  PRIMARY KEY (codCliente),
  FOREIGN KEY (codCidade) REFERENCES Cidade (codigoCidade) ON DELETE CASCADE))
```

ISTO NÂO É permitido neste contexto: significa que AO REMOVER A CIDADE, REMOVA TAMBEM O CLIENTE

Cliente

codCliente	Nome	endereco	codCidade
548	Maria	Rua Carvalho 615	1
549	Pedro	Rua Chaves 22	1

Cidade

codCidade	Descricao	Estado
1	Florianópolis	SC
2	São José	SC

Restrições de Integridade Referencial - EXCLUSÃO

- **Exclusão**: on delete cascade é util quando: ao eliminar um cliente que encerrou sua conta em um banco, A remoção do cliente removerá automaticamente todas as tuplas na tabela CONTA e demais tabelas que referenciam o cliente

```
CREATE TABLE ITENS_NOTA
 (numero integer NOT NULL, codProduto integer NOT NULL, qtde float,
  PRIMARY KEY (numero),
  FOREIGN KEY (codProduto) REFERENCES Produto (codProduto) ON DELETE RESTRICT),
 FOREIGN KEY (numero) REFERENCES NOTA (numero) ON DELETE CASCADE))
```

Nota

numero	codCliente	Data
001	1	20/02/2006
002	2	20/05/2008
003	1	20/04/2008

Produto

codProduto	Nome	valor
548	Pao	3,00
549	Leite	2,00

Itens_nota

Numero	codProduto	qtde
001	548	2
001	549	5

Restrições de Integridade Referencial

- Especificando restrições de exclusão e alteração

```
CREATE TABLE cliente
(codigoCliente integer NOT NULL,
nome varchar(40) NOT NULL,
codigoCidade integer,
PRIMARY KEY (codigoCliente),
FOREIGN KEY (codigoCidade) REFERENCES Cidade (codigoCidade)
    ON DELETE SET NULL
    ON UPDATE CASCADE)
```

- Na exclusão de uma cidade, **setar para nulo** o valor do atributo onde ela é referenciada (codigoCidade na tabela CLIENTE)
  - obs: a coluna precisa aceitar NULL
- Na alteração do valor de uma chave da tabela cidade, alterar em **cascata** os valores em CLIENTE que referenciam esta chave

Cliente

codCliente	Nome	endereco	codCidade
548	Maria	Rua Carvalho 615	1
549	Pedro	Rua Pedro Chaves 22	1

Cidade

codCidade	Descricao	Estado
1	Florianópolis	SC
2	São José	SC

Restrições Semânticas

Check constraints

```
CREATE TABLE products ( product_no integer, name text,
   price integer CHECK (price > 0) );
```

```
CREATE TABLE products ( product_no integer, name text, price numeric
CONSTRAINT positive_price CHECK (price > 0) );
```

```
CREATE TABLE products ( product_no integer, name
   text, price numeric CHECK (price > 0),
   discounted_price numeric CHECK (discounted_price >
   0), CHECK (price > discounted_price) );
```

Check constraints

```
create table emp
(codigoEmp integer,
tipo char(10) null,
nome char(10) not null,
CIC char(10) not null,
codigoDepto integer not null,
cartHabil char(10) null,
CREA char(10) null,
codigoRamo integer null,
primary key (codigoEmp),
foreign key (codigoDepto) references depto,
foreign key (codigoRamo) references ramo,
check ((tipo='engenheiro' and CREA is not null and codigoRamo is not null) OR
(tipo='motorista' and cartHabil is not null) OR
(tipo<>'motorista'and tipo<>'engenheiro')));
```

Gatilhos

Gatilhos (triggers)

- Interessante quando a inclusão, alteração ou exclusão de um atributo em alguma tabela tiver algum efeito sobre um atributo de outra tabela
- Um Trigger é um bloco de comandos Transact-SQL que é automaticamente executado quando um comando INSERT , DELETE ou UPDATE for executado em uma tabela do banco de dados.

Gatilhos (triggers)

- Não são o modo recomendado para implementar restrições de integridade
- RI são normalmente suportadas pelos SGBD atuais

- Gatilhos podem ser disparados ANTES ou DEPOIS do evento (insert, delete, update) especificado
- Sintaxe:

```
CREATE TRIGGER nome { BEFORE | AFTER } { evento [ OR ... ] }
   ON tabela [ FOR [ EACH ] { ROW | STATEMENT } ]
   EXECUTE PROCEDURE nome_da_função ( argumentos )
```

Gatilhos (triggers)

- Uma TRIGGER é sempre associada a uma tabela, porém os comandos que formam a TRIGGER podem acessar dados de outras tabelas.
- Ex: dadas as tabelas
  - Nota_Fiscal(Num_nota, valor_total)
  - Produto(Cod_Prod, nome, preço, estoque)
  - Nota_Prod(Num_nota, Cod_Prod, quantidade)

  - Pode-se criar uma Trigger para a operação de INSERT na tabela Nota_Prod. Sempre que for inserido um novo item de pedido na tabela Nota_Prod será disparada um Trigger que atualiza o nível de estoque do produto que está sendo vendido

Asserções

- Predicado que expressa uma condição que deve ser sempre satisfeita no BD
- Exemplo:
  - Um aluno só pode se matricular se não tiver débitos pendentes na biblioteca
- Sintaxe
  - `create assertion <nome_restrição> check predicado`
  - `drop assertion <nome_restrição>`

Asserções

- Como o SQL não suporta PARA TODO, precisamos usar o NOT EXISTS
- Qdo uma asserção é criada
  - O sistema verifica sua validade
  - Se as asserções são válidas, qualquer modificação posterior no banco será permitida somente quando a asserção não for violada
- A Verificação de asserções pode gerar um aumento significativo no tempo de processamento

Asserções

- **Exemplos**:

1. O numero da conta do cliente precisa estar entre 10.000 e 99.999

```
create assertion restricao_conta check
   (not exists (select * from conta
          where  numero < 10.000 OR numero > 99.999))
```

2. Definir uma restrição de integridade que não permita saldos negativos.

```
create assertion restricao_saldo check
(not exists (select * from conta where saldo < 0))
```