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

- 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 as colunas que compõem a chave primária devem ser obrigatórias e as 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´)`
  - `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 (***550***, “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

Viola a restrição →cidade 5 não existe

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

Cidade

codigoCidade	Descricao	Estado
1	Contagem	MG
2	Divinópolis	MG

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 - 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	Contagem	MG
2	Divinópolis	MG

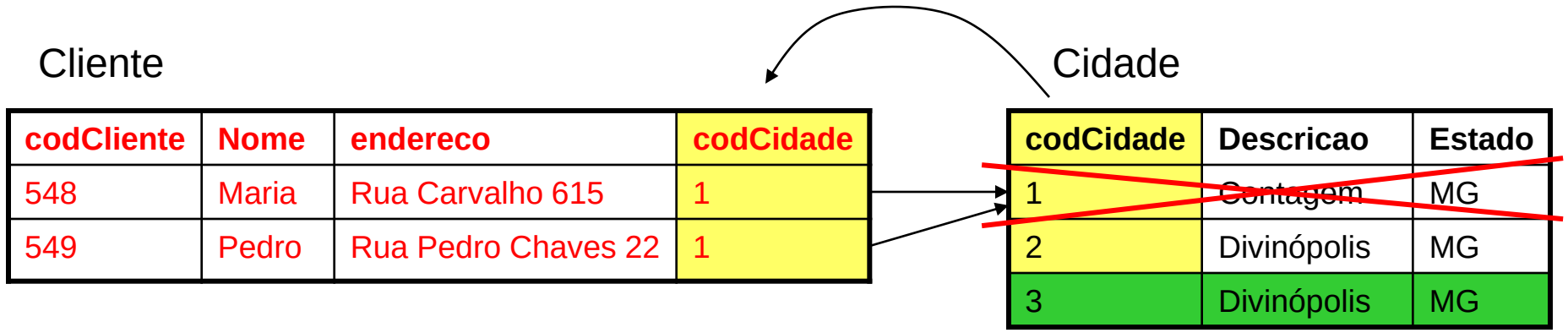
Restrições de Integridade Referencial - Exclusão

- AÇÕES:
  - (opção 1) NÃO permite alteração ou exclusão (**NO ACTION (default) ou Restrict;**
    - não permite a exclusão/alteração enquanto houver dependência;
      - Ex: só permite excluir a cidade quando nenhum cliente referenciar esta cidade
  - **(opção 2) SET DEFAULT** : se houver um valor default para a coluna da chave estrangeira, ela recebe este valor
  - **(opção 3) CASCADE** : propaga a exclusão/alteração;

Restrições de Integridade Referencial - EXCLUSÃO

(opção 1) 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))
```

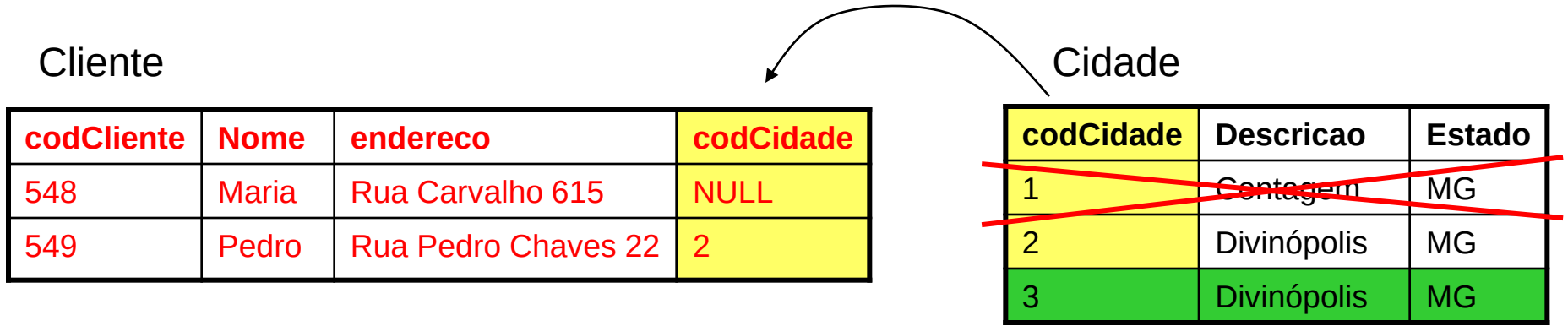


Restrições de Integridade Referencial - EXCLUSÃO

- ***(opção 2)*** assumir um valor default. Ao excluir uma cidade da tabela **CIDADE**, o SGBD precisa garantir que não exista nenhum cliente na tabela CLIENTE referenciando esta cidade

```
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))
```

OBS: Não muito recomendado/usado



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

codCidade	Descricao	Estado
1	~~Contagem~~	MG
2	Divinópolis	MG
3	Divinópolis	MG

Restrições de Integridade Referencial - EXCLUSÃO

(opção 3) 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	Contagem	MG
2	Divinópolis	MG

Restrições de Integridade Referencial - EXCLUSÃO

- **Exclusão**: on delete cascade é útil quando ao remover a tupla de uma tabela queremos excluir todas as tuplas relacionadas de outra tabela

```
CREATE TABLE Preco_Produto
 (codProduto integer NOT NULL, data Date, preco float,
  PRIMARY KEY (codProduto, data),
  FOREIGN KEY (codProduto) REFERENCES Produto (codProduto) ON DELETE RESTRICT),
 FOREIGN KEY (codProduto) REFERENCES Preco (codProduto) ON DELETE CASCADE))
```

Produto

codProduto	Nome
548	Pao
549	Leite

Preco

codProduto	data	preco
548	01/12/2016	2,00
548	06/06/2017	5,00
549	01/02/2017	3,00

Restrições Semânticas

Check constraints

```
CREATE TABLE products ( product_no integer, name text, price integer CHECK
   (price > 2 );
```

Teste: insert into products values (1,'teste',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 > 2),
   discounted_price numeric
                    CHECK (discounted_price > 0),
   CHECK (price > discounted_price) );
```

Teste1: insert into products values (1,'teste',0,0)

Teste2: insert into products values (1,'teste',100,200)

Teste3: insert into products values (1,'teste',100,50)

Gatilhos

Gatilhos

- Triggers (Gatilhos) são unidades PL / SQL armazenadas que são executadas automaticamente (“disparadas") em resposta a eventos especificados

- É armazenada no banco de dados e (se estiver no estado habilitado) é executada automaticamente

Gatilhos

- Um uso de triggers é impor regras de negócios que se aplicam a todos os aplicativos clientes.
- Por exemplo, suponha que os dados adicionados à tabela EMPREGADOS tenham um determinado formato e que muitos aplicativos clientes possam adicionar dados a essa tabela.
- Um gatilho na tabela pode garantir o formato adequado de todos os dados adicionados a ele.
- Como o gatilho é executado sempre que qualquer cliente adiciona dados à tabela, nenhum cliente pode burlar as regras
- O código que impõe as regras pode ser armazenado e mantido apenas no gatilho, em vez de em cada aplicativo cliente.

Gatilhos

- Criando gatilhos
  - Para criar gatilhos use a instrução DDL CREATE TRIGGER .
- Alterando Triggers
  - Para alterar um trigger, use a instrução DDL CREATE TRIGGER com a cláusula OR REPLACE .

Gatilhos

- Útil quando a inclusão, alteração ou exclusão de um atributo em alguma tabela tiver algum efeito sobre um atributo de outra tabela

- Exemplo: quando o saldo da conta do cliente for negativo, insira automaticamente um registro na tabela empréstimo e faça o valor do saldo na conta receber o valor zero

Gatilhos

- 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

Gatilhos

- Um gatilho tem essa estrutura:

```
TRIGGER trigger_name
  triggering_event
  [ trigger_restriction ]
Begin
  triggered_action ;
End;
```

Gatilhos

- O trigger_name deve ser único para todos os gatilhos no esquema do BD.
- Um trigger pode ter o nome de outro componente do esquema (uma tabela, por exemplo), é recomendado usar uma convenção de nomes para evitar confusão
- Se o trigger está no modo habilitado, o triggering_event faz com que o SGBD execute o triggered_action se a trigger_restriction é TRUE ou omitida.

Gatilhos

- O triggering_event a uma tabela , uma view, um schema, ou um banco de dados, e pode ser uma destas opções:
  - Declaração DML
  - Declaração DDL
  - Operação na base de dados (SERVERERROR, LOGON, LOGOFF, STARTUP, ou SHUTDOWN)
- Se o gatilho está no estado desabilitado, o triggering_event não faz o SGBD executar a triggered_action, mesmo se a trigger_restriction é TRUE ou omitida.

Gatilhos

- Por padrão, um trigger é criado no estado de habilitado. Você pode alternar ele entre habilitado e desabilitado;.

- Um trigger não pode ser chamado diretamente. Ele é chamado pelo evento de triggering_event, causado por um usuário ou aplicação.

- Um trigger pode disparar a partir de um desses eventos:

  - Antes do triggering event executar (declaração BEFORE)
  - Depois do triggering event executar (declaração AFTER)
  - Antes de cada linha que o trigger afetar (BEFORE à nível de tupla)
  - Depois de cada linha que o trigger afetar (AFTER à nível de tupla)

Exercícios

- Da lista de exercícios, verifique pelo menos 4 exercícios que podem ser afetados por Restrições de Integridade

- Identifique como seria possível implementar Triggers em alguma tabela do esquemas apresentados na disciplina