Linguagem de Definição de Dados (DDL) - Continuação

Prof. Jeferson Souza, MSc.

(jefecomp)

jefecomp.official@gmail.com

UDESC
UNIVERSIDADE DO ESTADO DE SANTA CATARINA
JOINVILLE
CENTRO DE CIÊNCIAS TECNOLÓGICAS

Integridade do dados

Restrições de integridade

Como visto nos slides anteriormente no material do curso é possível criar restrições na definição das tabelas do banco de dados. A definição de restrições permite especificar o modelo de consistência que os dados devem seguir, e portanto são denominadas **restrições de integridade**.

Exemplos

- O nome do usuário não pode ser nulo;
- Dois usuários não podem ter o mesmo cpf;
- O saldo da conta deve ser sempre maior do que R$50,00;

Restrições de integridade

Algumas das formas de especificar restrições de integridade já foram vistas anteriormente:

- **NULL**;
- **NOT NULL**;
- **UNIQUE**;
- **DEFAULT**.

Entretanto, vamos ver mais alguns exemplos da restrição **UNIQUE**.

Restrições de integridade: *UNIQUE*

Exemplo de candidatos a chave com *UNIQUE*

CREATE TABLE usuario(id serial *PRIMARY KEY*, nome varchar(20), email varchar(30), cpf char (11)., *UNIQUE*(email,cpf));

Restrições de integridade: *CHECK*

Com a restrição do tipo check é possível definir uma expressão (ex: preco $>$ 0) que deve ser satisfeita durante a manipulação dos dados.

Exemplo 1

CREATE TABLE conta(nr_conta bigserial, agencia bigint, saldo numeric, *PRIMARY KEY* (nr_conta,agencia), *FOREIGN KEY*(agencia) references agencia(id), ***CHECK***(saldo $>$ 0));

Exemplo 2

CREATE TABLE departamento (id bigserial *PRIMARY KEY*, nome varchar(30), sigla varchar(4), ***CHECK***(sigla **in** ('DCC','DMAT','DEC','DEE')));

Restrições de integridade: *CHECK* (Continuação)

Exemplo 3

CREATE TABLE departamento (id bigserial *PRIMARY KEY*, nome varchar(30), sigla varchar(4), ***CHECK***(sigla **in** (select sigla from siglas where tipo = 'departamento')));

Integridade Referencial

Conceito

Integridade referencial visa garantir que um conjunto de valores que apareçam em uma determinada tabela, sejam consistentes com os valores de sua tabela de referência.

Exemplo 1

CREATE TABLE usuario (id bigserial *PRIMARY KEY*, nome varchar(30), dept bigint ***references*** departamento);

Integridade Referencial (Continuação)

Exemplo 2

CREATE TABLE departamento (id bigserial *PRIMARY KEY*, nome varchar(30), sigla char(3) UNIQUE NOT NULL);

CREATE TABLE curso (id bigserial *PRIMARY KEY*, nome varchar(30), dept char(3) UNIQUE NOT NULL ***references*** departamento (sigla));

Importante

Colunas que não sejam chaves primárias para serem referenciadas devem ser, pelo menos, únicas.

Integridade Referencial: Operações em cascata

É possível realizar operações em cascata na base de dados para manter a integridade referencial. Para isso, basta definir as chaves estrangeiras com a cláusula **cascade**.

Exemplo

CREATE TABLE curso (id bigserial *PRIMARY KEY*, nome varchar(30), dept char(3) UNIQUE NOT NULL, *FOREIGN KEY*(dept) ***references*** departamento (sigla) **on delete cascade on update cascade**);

Visões (Views)

O que são visões (views)?

Visões são uma forma alternativa de acessar dados no modelo de dados. As visões são criadas a partir de consultas válidas realizadas sobre tabelas ou outras visões;

Visões (Views)

O que são visões (views)?

Visões são uma forma alternativa de acessar dados no modelo de dados. As visões são criadas a partir de consultas válidas realizadas sobre tabelas ou outras visões;

Para que servem as visões (views)?

- Fornecem um meio de acesso mais simples e direto ao dados;
- Permitem acesso controlado e limitado a dados com restrições de acesso;
- Permitem "estender" virtualmente o modelo de dados, e criar relações virtuais que podem ser utilizadas exatamente da mesma forma que tabelas.

Diferença entre tabelas e visões (views)

- Tabelas são estrturas criadas para armazenar dados;
- Visões são o resultado de consultas que podem ser manipuladas posteriormente da mesma forma que tabelas;
- Caso não exista a necessidade de relacionamentos, tabelas podem ser criadas sem a existência de outras estruturas na base de dados;
- A criação de uma visão depende da existência de, ao menos, uma tabela;

Diferença entre tabelas e visões (views) (Continuação)

- Os dados mostrados por uma visão podem ser modificados (em alguns casos) por comandos de inserção, remoção, e atualização. Entretanto, modificações não são recomendas, já que as mesmas devem refletir modificações nas tabelas que dão origem a visão alvo;

Criar visões (views)

CREATE VIEW NOME_DA VISÃO (nome das colunas) AS <Consulta>;

comando — argumento obrigatório — argumento opcional — argumento obrigatório

Criar visões (views)

Exemplos:

OBS: cargo_id = 1 é o identificador para o cargo de Professor.

CREATE VIEW professores_view as SELECT nome, sobrenome, email, cpf, cargo_id, departamento_id FROM usuario where cargo_id = 1;

Criar visões (views)

Exemplos:

OBS: cargo_id = 1 é o identificador para o cargo de Professor.

CREATE VIEW professores_view as SELECT nome, sobrenome, email, cpf, cargo_id, departamento_id FROM usuario where cargo_id = 1;

Pode-se depois realizar consultas diretamente na Visão(View)

SELECT * FROM professores_view where departamento_id = 1;

OBS: departamento_id=1 é o identificador do DCC.

Visões Materializadas (Materialised Views)

Alguns SGBDs permitem que visões sejam armazenadas em disco. Esse tipo de visão é chamada de **visão materializada (materialised view)**. Caso dados sejam inseridos na base de dados, e o resultado da consulta que define a visão mude, a visão materializada também é atualizada.

Criar Visões Materializadas (Materialised Views)

CREATE MATERIALIZED VIEW NOME_DA VISÃO (nome das colunas) AS <Consulta>;

comando — argumento obrigatório — argumento opcional — argumento obrigatório

Criar Visões Materializadas (Materialised Views)

```
CREATE MATERIALIZED VIEW professores_view as SELECT nome, sobrenome, email, cpf, cargo_id, departamento_id FROM usuario where cargo_id = 1;
```

```
SELECT * FROM professores_view where departamento_id = 1;
```

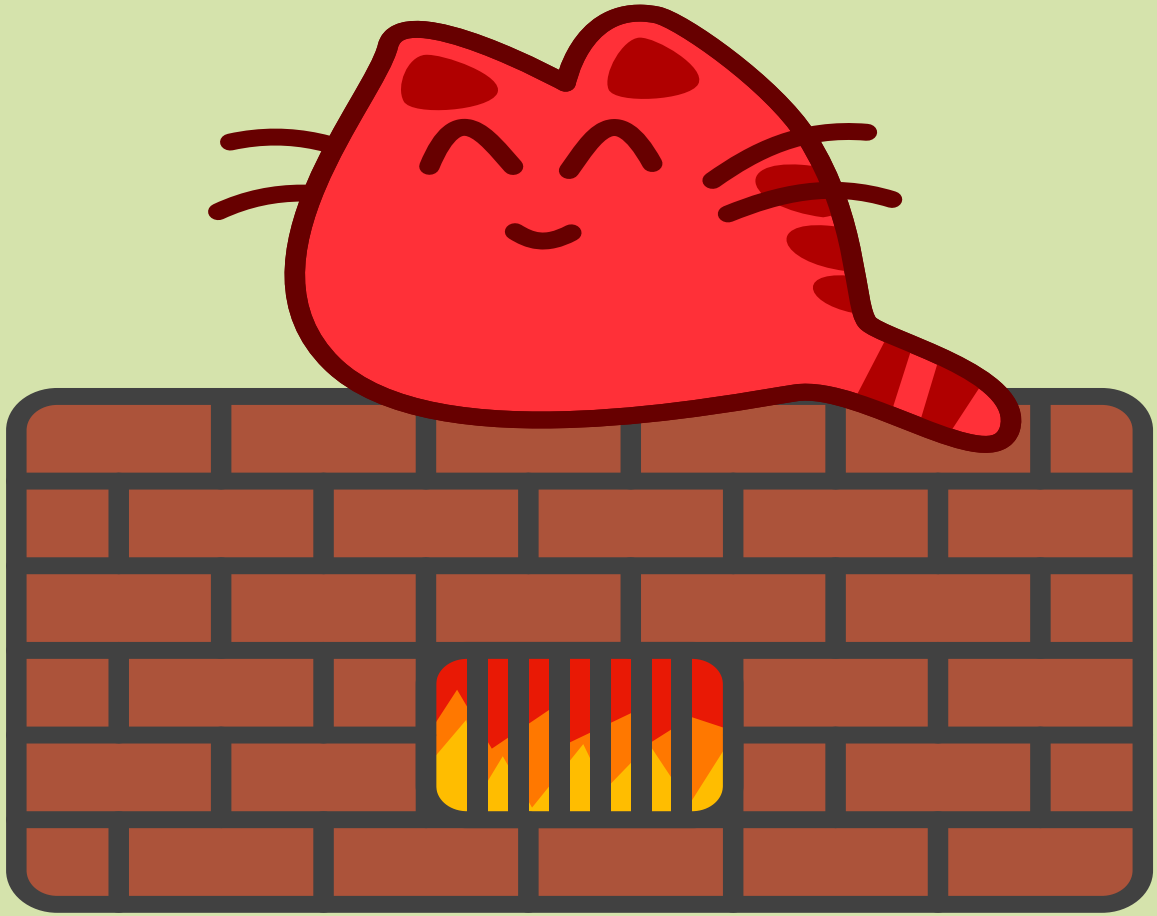
Bibliografia

Garcia-Molina, H. and Ullman, J. D. and Widom, J. *"Database Systems: The Complete Book"*. 2nd edition. Prentice Hall, 2008.

PostgreSQL Development Group. *"PostgreSQL 10.2 Documentation"*. 2018.

Silberschatz, A. and Korth, H.F. and Sudarshan, S. *"Database Systems"*. 6th edition. McGrawHill, 2011.

That's it folks!



Thank you for your attention!