



Instituto Federal de Educação, Ciência e Tecnologia da Bahia - IFBA
Departamento de Informática
Análise e Desenvolvimento de Sistemas / Licenciatura em Computação

Restrições de Integridade

André L. R. Madureira <andre.madureira@ifba.edu.br>
Doutorando em Ciência da Computação (UFBA)
Mestre em Ciência da Computação (UFBA)
Engenheiro da Computação (UFBA)

Restrições de Integridade

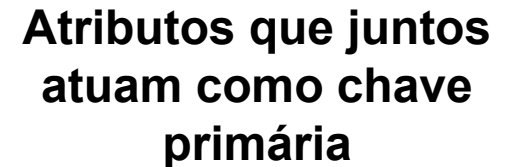
- Podemos definir várias **restrições de integridade** tais como:
 - **PRIMARY KEY**
 - **FOREIGN KEY**
 - **NOT NULL, DEFAULT e UNIQUE**
 - **CHECK**
 - **CREATE ASSERTION**
 - **CREATE DOMAIN**

Comando **PRIMARY KEY**

- Define a chave primária de uma tabela
- **Sintaxe (que funciona no MySQL)**

- **CREATE TABLE** <nome_tabela> (
 <nome_atributo1> <tipo> [**NOT NULL**],
 <nome_atributo2> <tipo> [**NOT NULL**],
 ...
 PRIMARY KEY(atributo1,[atributo2, ...])
);

**Atributos que juntos
atuam como chave
primária**



A chave primária deve conter valores únicos e não nulos

Exemplo de comando **PRIMARY KEY**

- Crie uma tabela *disciplina* para armazenar os IDs dos alunos, notas e status (aprovado ou reprovado)

```
CREATE TABLE disciplina (  
    id_aluno INT NOT NULL,  
    nota DOUBLE,  
    aprovado_reprovado BOOLEAN,  
    PRIMARY KEY (id_aluno)  
);
```

Exemplo de comando **PRIMARY KEY**

- Crie uma tabela *conta_corrente* para armazenar uma conta corrente de um sistema bancário

```
CREATE TABLE conta_corrente (  
    numero INT NOT NULL,  
    agencia INT NOT NULL,  
    saldo DOUBLE NOT NULL,  
    PRIMARY KEY (numero, agencia)  
);
```

Comando FOREIGN KEY

- Define a chave estrangeira de uma tabela

- **Sintaxe (que funciona no MySQL)**

- **CREATE TABLE** <nome_tabela> (
 <nome_atributo1> <tipo> [**NOT NULL**],
 <nome_atributo2> <tipo> [**NOT NULL**],
 ...

FOREIGN KEY (nome_atributo) **REFERENCES** nome_tabela2
(nome_da_pk_da_tabela2)
);

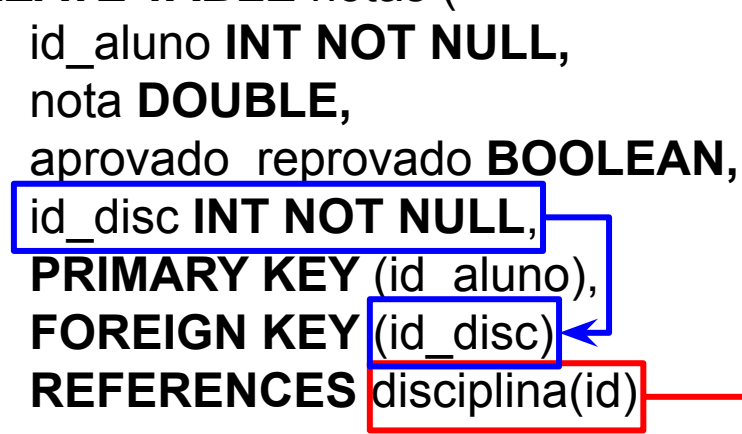
Atributo desta tabela que atua
como chave estrangeira

Nome da tabela que contém
a chave primária

Chave primária associada à chave estrangeira

Comando FOREIGN KEY

```
CREATE TABLE notas (  
  id_aluno INT NOT NULL,  
  nota DOUBLE,  
  aprovado reprovado BOOLEAN,  
  id_disc INT NOT NULL,  
  PRIMARY KEY (id_aluno),  
  FOREIGN KEY (id_disc)  
  REFERENCES disciplina(id)  
);
```



```
CREATE TABLE disciplina (  
  id INT NOT NULL,  
  nome_disc VARCHAR(20) NOT  
  NULL,  
  id_prof INT,  
  PRIMARY KEY (id)  
);
```

“*id_disc*” é a chave estrangeira (**FK**) de “*notas*”

“*id_disc*” está associado a chave primária (**PK**) “disciplina(id)”

Comando NOT NULL

- Define que um valor de atributo não pode ser NULO
- **Sintaxe:**
 - **CREATE TABLE** <nome_tabela> (
 <atributo> <tipo_dados> **NOT NULL**,
 ...
);
- **Ex: CREATE TABLE** pessoa (cpf **VARCHAR(11) NOT NULL**);

Comando **DEFAULT**

- Define que um valor padrão para um atributo não informado
- **Sintaxe:**
 - **CREATE TABLE** <nome_tabela> (
 <atributo> <tipo_dados> **DEFAULT** valor,
 ...
);
- **Ex: CREATE TABLE** conta_corrente (
 saldo **DOUBLE NOT NULL DEFAULT 0**
);

Exemplo de Comando **DEFAULT**

- **CREATE TABLE** conta_corrente (
 numero **INT NOT NULL**,
 agencia **INT NOT NULL**,
 saldo **DOUBLE NOT NULL DEFAULT 0**
);
- **INSERT INTO** conta_corrente(numero, agencia) **VALUES** (50217, 7744)
 - Crie uma conta de numero 50217 na agencia 7744 com saldo 0

Comando UNIQUE

- Define que um valor de atributo deve ser único (exclusivo) dentre os registros de uma tabela
 - **Diferença:** PRIMARY KEY define uma chave primária, **UNIQUE define uma chave candidata** (que poderia ser usada como chave primária)
- **Sintaxe:**
 - **CREATE TABLE** <nome_tabela> (
 <atributo> <tipo_dados>,
 ...,
 UNIQUE(<atributo1> [, <atributo2>, ...]),
);

Exemplo de Comando **UNIQUE**

- **CREATE TABLE** funcionario (
 id **INT PRIMARY KEY**,
 cpf **VARCHAR(11) NOT NULL**,
 UNIQUE (cpf)
);
- **'id' é chave primária** => Nenhum funcionário pode ter mesmo 'id'
(ID identifica unicamente cada registro)
- **'cpf' é chave candidata** => 'cpf' poderia ser usado como chave primária
 - CPF é único para cada funcionário
(não devem existir funcionários com mesmo CPF)

Exemplo de Comando **UNIQUE**

- **CREATE TABLE** funcionario (
 id **INT PRIMARY KEY**,
 cpf **VARCHAR(11) NOT NULL**,
 rg **VARCHAR(10) NOT NULL**,
 UNIQUE (cpf),
 UNIQUE (rg)
);

IMPORTANTE:

Podemos ter vários atributos **UNIQUE**, porém apenas um atributo **PRIMARY KEY**

Comando **CHECK**

- Define uma condição que deve ser testada sempre que um registro é inserido ou atualizado em uma tabela (relação)
- **Sintaxe:**
 - **CREATE TABLE** <nome_tabela> (
 <atributo1> <tipo_dados1>,
 ...,
 <atributoN> <tipo_dadosN>,
 CHECK (<condição>)
);

Exemplo de Comando **CHECK**

- **CREATE TABLE** conta_corrente (
 numero **INT NOT NULL**,
 agencia **INT NOT NULL**,
 saldo **DOUBLE NOT NULL**,
 PRIMARY KEY (numero, agencia),
 CHECK(saldo >= 0)
);

Toda vez que uma conta for criada ou for alterada, o DBMS vai verificar se saldo >= 0 (se essa condição for falsa, o DBMS irá emitir um erro)

Exemplo de Comando **CHECK**

- **CREATE TABLE** aluno (
 id **INT PRIMARY KEY**,
 nome **VARCHAR (15) NOT NULL**,
 grau_escolaridade **VARCHAR(15)**,
 CHECK(grau_escolaridade **IN** ('Bacharelado', 'Mestrado', 'Doutorado'))
);

Toda vez que um ALUNO for matriculado ou sofrer uma alteração no DB,
o DBMS vai verificar sua escolaridade
(o ALUNO precisa ter grau de 'Bacharelado', 'Mestrado' ou 'Doutorado' somente)

Exemplo de Comando **CHECK**

- **CREATE TABLE** conta_corrente (
 num_conta **INT NOT NULL**,
 num_agencia **INT NOT NULL**,
 saldo **DOUBLE NOT NULL**,
 PRIMARY KEY (num_conta, num_agencia),
 CHECK (num_agencia **IN** (**SELECT** num_agencia **FROM**
 agencia))
);

Somente permita a inserção de agências se elas existirem na tabela agencia. Nesse caso, o **CHECK** é mais exigente que a **FOREIGN KEY**.

CHECK é executado sempre que um registro é adicionado ou modificado em “*conta_corrente*” e também em “*agencia*”

Comando **CREATE ASSERTION**

- Define uma condição que deve ser testada sempre que um registro é inserido ou atualizado em uma das tabelas da afirmação (*assertion*)
 - Se a condição da afirmação for **FALSE**, a afirmação foi **violada**
 - O SGBD irá recusar a execução da transação que violou a afirmação
- **Sintaxe:**
 - **CREATE ASSERTION** <nome_assertion> **CHECK** <condição>;
- Afirmações (assertions) devem ser usadas com cuidado pois a afirmação é testada SEMPRE que uma das tabelas da consulta SQL sofre uma alteração (o que pode gerar problemas de desempenho)

Exemplo de Comando **CREATE ASSERTION**

- **CREATE ASSERTION** check_conta_corrente **CHECK** (
 EXISTS (
 SELECT num_agencia
 FROM conta_corrente
 GROUP BY num_agencia
 HAVING COUNT(numero) = 0
)
)

Essa afirmação será verificada sempre que a tabela “*conta_corrente*” sofrer uma alteração

- A afirmação é violada se existir alguma agência do banco que possui 0 (zero) contas correntes

Desempenho do comando **CREATE ASSERTION**

- Afirmações (assertions) devem ser usadas com cuidado pois uma afirmação é testada SEMPRE que uma das tabelas da afirmação sofre uma alteração
 - **Isto pode gerar sérios problemas de desempenho**
- Afirmações somente devem ser usadas quando outras restrições de integridade não puderem ser utilizadas. **Ex:**
 - **Restrições de domínio (Tipos de dados, NOT NULL, DEFAULT)**
 - **Chaves (UNIQUE, PRIMARY KEY, FOREIGN KEY)**
 - **CHECK**

Comando **CREATE DOMAIN**

- Cria um domínio de atributo que pode ser usado em qualquer lugar de uma consulta SQL onde tipos de dados são utilizados
- **Sintaxe:**
 - **CREATE DOMAIN** <nome_dominio> **AS** <tipo_dados> [**<restricoes_de_integridade>**] ;
- **Ex:**
 - **CREATE DOMAIN** Reais **AS** NUMERIC(12,2) **NOT NULL**;
 - **CREATE DOMAIN** CPF **AS** VARCHAR(11);

Exemplo de Comando **CREATE DOMAIN**

- **CREATE DOMAIN** Reais **AS NUMERIC(12,2) NOT NULL;**
- **CREATE TABLE** conta_corrente (
 numero **INT NOT NULL**,
 agencia **INT NOT NULL**,
 saldo Reais,
 PRIMARY KEY (numero, agencia)
);

○ **MySQL não suporta domínios**
(tipos de dados customizados)

PostgreSQL e o MS SQL Server
suportam esse recurso

Exemplo de Comando **CREATE DOMAIN**

- **CREATE DOMAIN** Salario **AS** NUMERIC(12,2) **NOT NULL CHECK** (**VALUE** >= 1320.0);
 - Todo atributo do tipo Salário precisa ter um valor maior ou igual a 1320,00 reais
- **CREATE TABLE** funcionario (
 id **INT PRIMARY KEY**,
 nome **VARCHAR(50) NOT NULL**,
 salario Salario
);

Exemplo de Comando **CREATE DOMAIN**

- **CREATE DOMAIN** Escolaridade **AS VARCHAR(50) CHECK (VALUE IN ('Bacharelado', 'Mestrado', 'Doutorado'))**;
 - Todo atributo do tipo Escolaridade somente assume um dos seguintes valores: Bacharelado, Mestrado ou Doutorado
- **CREATE TABLE** aluno (
 id **INT PRIMARY KEY**,
 nome **VARCHAR(50) NOT NULL**,
 nivel_escolaridade Escolaridade
);

Referencial Bibliográfico

- KORTH, H.; SILBERSCHATZ, A.; SUDARSHAN, S. **Sistemas de bancos de dados**. 5. ed. Rio de Janeiro: Ed. Campus, 2006.
- DATE, C. J. **Introdução a sistemas de bancos de dados**. Rio de Janeiro: Ed. Campus, 2004. Tradução da 8ª edição americana.