

SQL

(Structured Query Language)

INE5423 – Banco de Dados I

Carina F. Dorneles
dorneles@inf.ufsc.br

Introdução

- ▶ **DDL - “*Data Definition Language*”**

- ▶ Linguagem usada para descrever (implementar) o modelo lógico no SGBD
 - ▶ Trabalha com os [meta-dados](#)

- ▶ **DML - “*Data Manipulation Language*”**

- ▶ Linguagem usada trabalhar sobre a base de dados (acesso e alteração de dados) Integrada com a DDL.
 - ▶ Trabalha com as [instâncias](#)



Modelos Lógicos e Linguagens

<u>Modelo</u>	<u>Linguagem</u>
Relacional	SQL
Orientado a Objeto	OQL
Objeto-Relacional	SQL3
XML	XQuery



Exemplo

► Modelo Lógico – relacional

```
cidade (codigo, nome, UF)
medico (codigo, nome, email, CRM, codCid#)
      codCid REFERENCIA cidade (codigo)
paciente (codigo, nome, email, fone, codCid#)
      codCid REFERENCIA cidade (codigo)
consulta (data, hora, codPac#, codMed#)
      codPac REFERENCIA paciente (codigo)
      codMed REFERENCIA medico (codigo)
medicamento (codigo, descricao)
cons_medicame (data, hora, codPac#, codMedica#)
      codMedica REFERENCIA medicamento (codigo)
      (data, hora, codPac) REFERENCIA consulta (data, hora, codPac))
```



Representação textual informal

SQL - DDL

(SQL – Data Definition Language)

SQL - DDL

- ▶ Comandos para *implementar* o modelo lógico
 - ▶ Comando para **definição** de estruturas no BD
 - ▶ **CREATE**
 - **Ex.:** CREATE TABLE, CREATE INDEX, CREATE DATABASE, ...
 - ▶ Comando para **alteração** de estruturas no BD
 - ▶ **ALTER**
 - **Ex.:** ALTER TABLE, ALTER INDEX, ALTER VIEW, ...
 - ▶ Comando para **remoção** de estruturas no BD
 - ▶ **DROP**
 - **Ex:** DROP TABLE, DROP INDEX, DROP SEQUENCE, ...



Manipulação das tabelas

- ▶ CREATE TABLE

- ▶ Cria tabelas no Banco de Dados

- ▶ ALTER TABLE

- ▶ Altera tabelas já existentes no Banco de Dados

- ▶ DROP TABLE

- ▶ Apaga tabelas já existentes no Banco de Dados



Criação de tabelas

- ▶ Exemplo:
 - ▶ a seguinte definição:

cidade (codigo, nome, UF)

- ▶ Em SQL, fica:

```
CREATE TABLE cidade
(
    codigo integer not null,
    nome varchar (40),
    UF char(2)
);
```



Tabela não tem chave primária

Chave primária (PK)

- ▶ Definição da chave primária (PK)
 1. definir os atributos
 2. daqueles atributos, qual faz parte da chave primária (PK)

cidade (codigo, nome, UF)

- ▶ Em SQL, fica:

```
CREATE TABLE cidade
(
    codigo integer not null,
    nome varchar (40),
    UF char(2),
    PRIMARY KEY(codigo)
);
```



Chave primária (PK)

► Definição da chave primária (PK)

1. definir os atributos
2. daqueles atributos, qual faz parte da chave primária (PK)

cidade (codigo, nome, UF)

► Em SQL, fica:

```
CREATE TABLE cidade
```

```
(  
  codigo integer not null,  
  nome varchar (40),  
  UF char(2),  
  PRIMARY KEY(codigo)  
);
```

Definir todos os atributos

Dos atributos definidos, definir Qual(is) faz(em) parte da PK

Criação de tabelas

Nome do atributo Tipo do atributo

Informar se o atributo pode ter valor nulo ou não. Se não pode, colocar NOT NULL. Caso contrário, deixar sem nada.

```
CREATE TABLE cidade  
(  
  [codigo] integer not null,  
  nome varchar (40) ,  
  UF char (2) ,  
  PRIMARY KEY (codigo)  
) ;
```

Usar quando há outro comando SQL abaixo

Alternativa para definição de PK

```
CREATE TABLE cidade  
(  
    codigo integer not null PRIMARY KEY,  
    nome varchar (40),  
    UF char(2)  
);
```

Obs.:

- Primeiro o atributo é definido,
depois é feita a definição de PK



Tipos

- ▶ Os tipos, normalmente, são dependentes do SGBD. Alguns são padrão SQL.
- ▶ Numéricos Inteiros
 - ▶ `int` (ou `integer`)
 - ▶ `longint`
 - ▶ `smallint`
- ▶ Numéricos Decimais
 - ▶ `numeric (total, nr. de dígitos depois da vírgula)` - **exato**
 - ▶ `decimal (total, nr. de dígitos depois da vírgula)` - **aproximado**
- ▶ Caracteres
 - ▶ `char(total)` - string de tamanho exato total
 - ▶ `varchar(total)` - string de tamanho máximo total
- ▶ Data e hora
 - ▶ `date`
 - ▶ `time`
 - ▶ `timestamp`



Tabelas com FK

- ▶ Com **chave estrangeira** (FK)

- ▶ Exemplo: a seguinte definição:

```
medico (codigo, nome, email, CRM, codCid#)  
      codCid REFERENCIA cidade (codigo)
```



Tabelas com FK

► Com **chave estrangeira** (FK)

► Exemplo: a seguinte definição:

```
medico (codigo, nome, email, CRM, codCid#)  
      codCid REFERENCIA cidade (codigo)
```

► Em SQL, fica:

```
CREATE TABLE medico  
(  
    codigo integer not null,  
    nome varchar (40),  
    email varchar(20),  
    CRM integer,  
    codcid integer,  
    PRIMARY KEY(codigo)
```



Definição de um atributo

Tabelas com FK

▶ Com **chave estrangeira** (FK)

▶ Exemplo: a seguinte definição:

```
medico (codigo, nome, email, CRM, codCid#)  
      codCid REFERENCIA cidade (codigo)
```

▶ Em SQL, fica:

```
CREATE TABLE medico  
(  
    codigo integer not null,  
    nome varchar (40),  
    email varchar(20),  
    CRM integer,  
    codcid integer,  
    PRIMARY KEY(codigo),  
    FOREIGN KEY(codcid) REFERENCES cidade (codigo)  
)
```



Tabelas com FK

► Com chave estrangeira (FK)

► Exemplo: a seguinte definição:

```
medico (codigo, nome, email, CRM, codCid#)  
      codCid REFERENCIA cidade (codigo)
```

► Em SQL, fica:

```
CREATE TABLE medico  
(  
    codigo integer not null,  
    nome varchar (40),  
    email varchar(20),  
    CRM integer,  
    codcid integer,  
    PRIMARY KEY(codigo),  
    FOREIGN KEY(codcid) REFERENCES cidade (codigo)  
)
```

Definir o atributo da FK

► Depois dizer que este atributo é FK, e qual tabela ele referencia

Alternativa para definição de FK

```
CREATE TABLE medico
(
    codigo integer not null,
    nome varchar (40),
    email varchar(20),
    CRM integer,
    codcid integer FOREIGN KEY(codcid) REFERENCES cidade (codigo),
    PRIMARY KEY(codigo)
)
```



Obs.:

**- Primeiro o atributo é definido,
depois é feita a definição de FK**

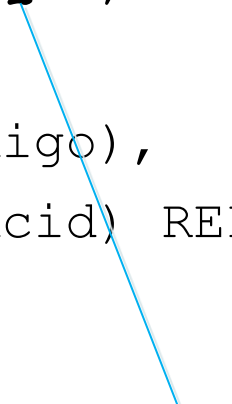
Tabelas com chave alternativa

```
CREATE TABLE medico
(
    codigo integer not null,
    nome varchar (40),
    email varchar(20),
    CRM integer,
    codcid integer,
    PRIMARY KEY(codigo),
    FOREIGN KEY(codcid) REFERENCES cidade (codigo),
    UNIQUE (CRM)
)
```

Terá as mesmas restrições da PRIMARY KEY; só que não poderá ser referenciada por tabelas

Tabelas com chave alternativa

```
CREATE TABLE medico
(
    codigo integer not null,
    nome varchar (40),
    email varchar(20),
    CRM integer UNIQUE,
    codcid integer,
    PRIMARY KEY(codigo),
    FOREIGN KEY(codcid) REFERENCES cidade (codigo)
)
```



Alguns SGBDs permitem esta sintaxe
Postgresql aceita as duas sintaxes

Tabelas com PK composta

- ▶ Exemplo: a seguinte definição:

```
consulta (data, hora, codPac#, codMed#)  
    codPac REFERENCIA paciente (codigo)  
    codMed REFERENCIA medico (codigo)
```



Tabelas com PK composta

- ▶ Exemplo: a seguinte definição:

```
consulta (data, hora, codPac#, codMed#)  
    codPac REFERENCIA paciente (codigo)  
    codMed REFERENCIA medico (codigo)
```

- ▶ Em SQL, fica:

```
CREATE TABLE consulta  
(  
    data DATE not null,  
    hora TIME not null,  
    codPac INTEGER not null,  
    codMed INTEGER not null,  
    PRIMARY KEY(data, hora, codPac)  
    FOREIGN KEY (codpac) REFERENCES paciente (codigo),  
    FOREIGN KEY (codmed) REFERENCES medico (codigo)  
)
```

Observe: os 3 são NOT NULL

Referência a PK composta

- ▶ Exemplo: a seguinte definição:

```
cons_medicame (data, hora, codPac#, codMedica#)
```

```
codMedica REFERENCIA medicamento (codigo)
```

```
(data, hora, codPac) REFERENCIA consulta (data, hora, codPac)
```



Referência a PK composta

- ▶ Exemplo: a seguinte definição:

```
cons_medicame (data, hora, codPac#, codMedica#)  
  codMedica REFERENCIA medicamento (codigo)  
  (data, hora, codPac) REFERENCIA consulta (data, hora, codPac)
```

- ▶ Em SQL, fica:

```
CREATE TABLE cons_medicame  
(  
  data DATE not null,  
  hora TIME not null,  
  codPac INTEGER not null,  
  codMedica INTEGER not null,  
  PRIMARY KEY(data, hora, codPac, codMedica)  
  FOREIGN KEY (data, hora, codPac) REFERENCES consulta (data, hora, codPac),  
  FOREIGN KEY (codmedica) REFERENCES medicamento (codigo)  
)
```



Referência a PK composta

- Exemplo: a seguinte definição:

```
cons_medicame (data, hora, codPac#, codMedica#)  
  codMedica REFERENCIA medicamento (codigo)  
  (data, hora, codPac) REFERENCIA consulta (data, hora, codPac)
```

- Em SQL, fica:

```
CREATE TABLE cons_medicame  
(  
  data DATE not null,  
  hora TIME not null,  
  codPac INTEGER not null,  
  codMedica INTEGER not null,  
  PRIMARY KEY(data, hora, codPac, codMedica)  
  FOREIGN KEY (data, hora, codPac) REFERENCES consulta (data, hora, codPac),  
  FOREIGN KEY (codmedica) REFERENCES medicamento (codigo)  
)
```

Mesma ordem e mesmo tipo

Definição de CONSTRAINT

- ▶ As restrições de PK e FK podem ser nomeadas
- ▶ Nomes **únicos** em todo o BD
 - ▶ A tabela abaixo tem uma PK sem nome:

```
CREATE TABLE cidade
(
    codigo integer not null,
    nome varchar (40),
    UF char(2),
    PRIMARY KEY (codigo)
)
```

- ▶ Já esta definição de tabela tem uma PK com nome “pk_cid”:

```
CREATE TABLE cidade
(
    codigo integer not null,
    nome varchar (40),
    UF char(2),
    CONSTRAINT pk_cid PRIMARY KEY (codigo)
)
```



Definição de CONSTRAINT

- ▶ A tabela abaixo tem uma FK sem nome:

```
CREATE TABLE medico
(
    codigo integer not null,
    nome varchar (40),
    codcid integer,
    CONSTRAINT pk_medico PRIMARY KEY (codigo),
    FOREIGN KEY (codcid) REFERENCES cidade (codigo)
)
```




Definição de CONSTRAINT

- ▶ A tabela abaixo tem uma FK sem nome:

```
CREATE TABLE medico
(
    codigo integer not null,
    nome varchar (40),
    codcid integer,
    CONSTRAINT pk_medico PRIMARY KEY (codigo),
    FOREIGN KEY (codcid) REFERENCES cidade (codigo)
)
```

- ▶ Já esta definição de tabela tem uma FK com nome “moradia”:

```
CREATE TABLE medico
(
    codigo integer not null,
    nome varchar (40),
    codcid integer,
    CONSTRAINT pk_medico PRIMARY KEY (codigo),
    CONSTRAINT moradia FOREIGN KEY (codcid) REFERENCES cidade (codigo)
)
```



Alternativas para definição de CONSTRAINT

► PK

```
CREATE TABLE cidade
(
    codigo integer not null CONSTRAINT pk_cid PRIMARY KEY,
    nome varchar (40),
    UF char(2)
)
```

► FK

```
CREATE TABLE medico
(
    codigo integer not null,
    nome varchar (40),
    email varchar(20),
    CRM integer,
    codcid integer CONSTRAINT fk_mc FOREIGN KEY(codcid) REFERENCES cidade (codigo),
    PRIMARY KEY(codigo)
)
```



Definição de valores *default*

- ▶ Toda coluna pode ter um valor *default*
 - ▶ Exemplo:

```
CREATE TABLE cidade
(
    codigo integer not null,
    nome varchar (40),
    UF char(2) DEFAULT 'SC',
    CONSTRAINT pk_cid PRIMARY KEY (codigo)
)
```



Checagem de valores – CHECK()

- ▶ Usado para criar restrições na criação da tabela
- ▶ Exemplo:

```
CREATE TABLE medico
(
    codigo integer not null,
    nome varchar (40),
    email varchar(20),
    CRM integer,
    codcid integer,
    PRIMARY KEY(codigo),
    FOREIGN KEY(codcid) REFERENCES cidade (codigo),
    CONSTRAINT const1 CHECK (email <> '' AND codigo > 100)
)
```



Checagem de valores – CHECK()

- ▶ Usado para criar restrições na criação da tabela
- ▶ Exemplo:

```
CREATE TABLE paciente
(
    codigo integer not null,
    nome varchar (40),
    email varchar(20),
    categoria char(1) CHECK (categoria in ('A', 'B', 'C')),
    codcid integer,
    PRIMARY KEY(codigo),
    FOREIGN KEY(codcid) REFERENCES cidade (codigo)
)
```



Comentários em SQL

- ▶ Comentário de uma linha --

```
-- CREATE TABLE cidade (...)
```

- ▶ Comentário de um bloco /* ... */

```
/*CREATE TABLE cidade  
(codigo integer not null,  
...)  
*/
```



Exercícios

