

SQL

(Structured Query Language)

INE5423 – Banco de Dados I

Carina F. Dorneles
dorneles@inf.ufsc.br

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)

Manipulação das tabelas

- ▶ CREATE TABLE

- ▶ Cria tabelas no Banco de Dados

- ▶ DROP TABLE

- ▶ Apaga tabelas já existentes no Banco de Dados

- ▶ ALTER TABLE

- ▶ Altera tabelas já existentes no Banco de Dados



Deleção de tabelas

- ▶ Comando

- ▶ `DROP TABLE nomeTabela;`


- ▶ Apaga toda a tabela e seu conteúdo

- ▶ Não há como recuperar a tabela removida

```
DROP TABLE medico;
```

```
DROP TABLE paciente;
```

```
DROP TABLE cidade;
```



**Cidade é referenciada por
medico e paciente
DEVE ser a última a ser deletada**

Alteração de tabelas

- ▶ Comando

- ▶ `ALTER TABLE nomeTabela`

- ▶ Alterações possíveis

- ▶ Adicionar/apagar coluna
 - ▶ Adicionar/apagar definição de valores *default*
 - ▶ Adicionar/apagar definição de PK e FK

- ▶ Demais alterações (mudança de nome de coluna, mudança de tipo) são dependentes do SGBD



Alteração de tabelas - colunas

▶ Exemplos

Medico

<i>codigo</i>	<i>nome</i>	<i>email</i>	<i>CRM</i>	<i>codCid</i>

▶ Adicionando coluna

```
ALTER TABLE medico  
ADD especializacao VARCHAR(40),  
ADD area VARCHAR(30);
```

Medico

<i>codigo</i>	<i>nome</i>	<i>email</i>	<i>CRM</i>	<i>codCid</i>	<i>especializacao</i>	<i>area</i>

▶ Apagando coluna

```
ALTER TABLE medico  
DROP especializacao;
```

Medico

<i>codigo</i>	<i>nome</i>	<i>email</i>	<i>CRM</i>	<i>codCid</i>	<i>especializacao</i>	<i>area</i>

Alteração de tabelas - *default*

▶ Exemplos

▶ Nova definição de *default*

```
ALTER TABLE cidade  
ALTER UF SET DEFAULT 'RS';
```

▶ Apagando definição de *default*

```
ALTER TABLE cidade  
ALTER UF DROP DEFAULT;
```



Alteração de tabelas – PK e FK

▶ Exemplos

▶ Adicionando definição de PK

```
ALTER TABLE paciente  
ADD PRIMARY KEY (codigo);
```

OU

```
ALTER TABLE medico  
ADD CONSTRAINT pk_paciente PRIMARY KEY (codigo);
```

▶ Apagando definição de PK

```
ALTER TABLE paciente  
DROP pk_paciente;
```

**Só funciona se tiver sido criada
uma *CONSTRAINT* para PK**

Alteração de tabelas – PK e FK

▶ Exemplos

▶ Adicionando definição de FK

```
ALTER TABLE paciente  
ADD FOREIGN KEY (codCid) REFERENCES cidade (codigo);
```

ou

```
ALTER TABLE paciente  
ADD CONSTRAINT mora FOREIGN KEY (codCid) REFERENCES  
cidade (codigo);
```

▶ Apagando definição de FK

```
ALTER TABLE paciente  
DROP mora;
```

**Só funciona se tiver sido criada
uma *CONSTRAINT* para FK**

Ordem na criação das tabelas

- ▶ Tabelas referenciadas:
 - ▶ Devem ser criadas antes

Exemplo:

Errado:

```
paciente (codigo, nome, codcid#)  
          codcid referencia cidade (codigo)  
cidade (codigo, nome, uf)
```

Correto:

```
cidade (codigo, nome, uf)  
paciente (codigo, nome, codcid#)  
          codcid referencia cidade (codigo)
```



DICA na criação de tabelas

Evita referências a tabelas ainda não criadas

1. Cria todas as tabelas sem chaves estrangeiras
 - Escrever todos os comandos de
`CREATE TABLE nomeTabela`
`(`
`...`
`);`
2. Alterar todas as tabelas que possuem referencias a outras tabelas, adicionando chaves estrangeiras
 - Escrever todos os comandos de

```
ALTER TABLE nomeTabela  
ADD CONSTRAINT nome  
FOREIGN KEY (nomeColunaChaveEstrangeira)  
REFERENCES tabela (nomeColunaChavePrimaria);
```

SQL: DDL e DML

- ▶ **DDL:** possui comandos para *implementar* o modelo lógico
 - ▶ Comando para **criação** das tabelas
 - ▶ Comando para **alteração** das tabelas
 - ▶ Comando para **remoção** das tabelas
- ▶ **DML:** possui comandos para *manipular* os dados armazenados no Banco de Dados
 - ▶ Comando para **inserção** de dados nas tabelas
 - ▶ Comando para **alteração** de dados nas tabelas
 - ▶ Comando para **remoção** de dados nas tabelas
 - ▶ Comando para **consulta** aos dados nas tabelas



SQL/DML: manipulação de dados

- ▶ INSERT INTO . . .
 - ▶ Comando para **inserção** de dados nas tabelas
- ▶ UPDATE . . .
 - ▶ Comando para **alteração** de dados nas tabelas
- ▶ DELETE FROM . . .
 - ▶ Comando para **remoção** de dados nas tabelas
- ▶ SELECT . . .
 - ▶ Comando para **consulta** aos dados nas tabelas



Inserção de Dados

■ INSERT INTO

- ✓ **Sem informar** os atributos

Na tabela: Cidade (codigo, nome, uf)

```
INSERT INTO cidade VALUES (1, 'Passo Fundo', 'RS');
```

```
INSERT INTO cidade VALUES (2, 'Joenville', 'SC');
```



Inserção de Dados

■ INSERT INTO

- ✓ **Sem informar** os atributos
- ✓ **Informando** os atributos

Na tabela: Cidade (codigo, nome, uf)

```
INSERT INTO cidade VALUES (1, 'Passo Fundo', 'RS');
```

```
INSERT INTO cidade VALUES (2, 'Joenville', 'SC');
```

```
INSERT INTO cidade (codigo, nome) VALUES (3, 'Marau');
```

```
INSERT INTO cidade (codigo, nome) VALUES (4, 'Carazinho');
```



Inserção de Dados

```
INSERT INTO cidade VALUES (1, 'Passo Fundo', 'RS');
```

```
INSERT INTO cidade VALUES (2, 'Joenville', 'SC');
```

```
INSERT INTO cidade (codigo, nome) VALUES (3, 'Marau');
```

```
INSERT INTO cidade (codigo, nome) VALUES (4, 'Carazinho');
```

Resultado:

	CODIGO	NOME	UF
▶	1	Passo Fundo	RS
	2	Joenville	SC
	3	Marau	<null>
	4	Carazinho	<null>



Inserção de Dados

- **Inserção de chave estrangeira**
 - valor já deve existir na PK da outra tabela

paciente (codigo, nome, email, fone, codCid#)
codCid REFERENCIA cidade (codigo)

```
INSERT INTO paciente VALUES (1, 'Juca', 'ju@abc', '9999.9999', 1);  
INSERT INTO paciente VALUES (2, 'Linda', 'll@abc', '9898.9898', 2);
```

```
INSERT INTO paciente (codigo, nome) VALUES (3, 'Luis');  
INSERT INTO paciente (codigo, nome) VALUES (4, 'Lana');
```



Inserção de Dados

- **Inserção de chave estrangeira**
 - valor já deve existir na PK da outra tabela

paciente (codigo, nome, email, fone, codCid#)
codCid REFERENCIA cidade (codigo)

```
INSERT INTO paciente VALUES (1, 'Juca', 'ju@abc', '9999.9999', 1);  
INSERT INTO paciente VALUES (2, 'Linda', 'll@abc', '9898.9898', 2);
```



Inserção de Dados

```
INSERT INTO paciente VALUES (1, 'Juca', 'juc@abc', '9999.9999', 1);
```

```
INSERT INTO paciente VALUES (2, 'Linda', 'll@abc', '9898.9898', 2);
```

```
INSERT INTO paciente (codigo, nome) VALUES (3, 'Luis');
```

```
INSERT INTO paciente (codigo, nome) VALUES (4, 'Lana');
```

Resultado:

	CODIGO	NOME	EMAIL	FONE	CODCID
▶	1	Juca	juc@abc	9999.9999	1
	2	Linda	ll@abc	9898.9898	2
	3	Luis	<null>	<null>	<null>
	4	Lana	<null>	<null>	<null>



Alteração de dados

- Supondo a tabela, com estes dados:

	codigo integer	nome character var	curriculo xml	idade integer	email character var
1	1	Aninha		30	ana@123.456
2	3	Juli		30	jul@abc.def
3	4	Lia		30	li@abc
4	2	Luca		17	luc@email.com



Alteração de dados

- Supondo a tabela, com estes dados:

	codigo integer	nome character var	curriculo xml	idade integer	email character var
1	1	Aninha		30	ana@123.456
2	3	Juli		30	jul@abc.def
3	4	Lia		30	li@abc
4	2	Luca		17	luc@email.com

UPDATE pessoa

SET idade = 18, email = 'jul@email.com'

WHERE nome = 'Juli';

	codigo integer	nome character var	curriculo xml	idade integer	email character var
1	1	Aninha		30	ana@123.456
2	4	Lia		30	li@abc
3	2	Luca		17	luc@email.com
4	3	Juli		18	jul@email.com

Exclusão de dados

Supondo a tabela:

	codigo integer	nome character var	curriculo xml	idade integer	email character var
1	1	Aninha		30	ana@123.456
2	4	Lia		30	li@abc
3	2	Luca		17	luc@email.com
4	3	Juli		18	jul@email.com



Exclusão de dados

Supondo a tabela:

	codigo integer	nome character var	curriculo xml	idade integer	email character var
1	1	Aninha		30	ana@123.456
2	4	Lia		30	li@abc
3	2	Luca		17	luc@email.com
4	3	Juli		18	jul@email.com

```
DELETE FROM pessoa  
WHERE nome = 'Lia';
```

	codigo integer	nome character var	curriculo xml	idade integer	email character var
1	1	Aninha		30	ana@123.456
2	2	Luca		17	luc@email.com
3	3	Juli		18	jul@email.com



Exclusão de dados

Supondo a tabela:

	codigo integer	nome character var	curriculo xml	idade integer	email character var
1	1	Aninha		30	ana@123.456
2	4	Lia		30	li@abc
3	2	Luca		17	luc@email.com
4	3	Juli		18	jul@email.com

DELETE FROM pessoa

	codigo integer	nome character var	curriculo xml	idade integer	email character var
--	---------------------------	-------------------------------	--------------------------	--------------------------	--------------------------------



Alteração de **dados** (não de esquema)

- Para que as atualizações no BD, feitas com os comandos DML, sejam visualizados por outros usuários:
 - COMMIT
- Para que elas sejam desfeitas:
 - ROLLBACK
 - Funciona apenas se o comando COMMIT não foi executado
 - Assunto tratado em detalhes em “Processamento de Transações”, na disciplina de BD-2



Resumo dos comandos

DDL

DML

<i>criação</i>	CREATE	INSERT
<i>alteração</i>	ALTER	UPDATE
<i>remoção</i>	DROP	DELETE



Exercícios

