UDESC

Banco de Dados I

Prof. Diego Buchinger
diego.buchinger@outlook.com
diego.buchinger@udesc.br

Profa. Rebeca Schroeder Freitas
Prof. Fabiano Baldo

PostgreSQL

SQL – Structured Query Language

Microsoft® SQL Server®

UDESC

SQL – O que é

- Utiliza uma base Formal combinando álgebra relacional e cálculo relacional

- Linguagem comercial para BD relacional
  - Desenvolvida pela IBM no início dos anos 70
  - Tornou-se padrão ISO desde a década de 80
    - SQL-1 (86), SQL-2 (92), SQL-3 (99)

- Se estabeleceu como a linguagem padrão para banco de dados relacionais

UDESC

SQL – Do que é composta

Apesar de ser chamada de “linguagem de consulta” possui outras funcionalidades também:

- **Linguagem de Definição de Dados (DDL)**
  - Comandos para definição, remoção e modificação de tabelas, índices, chaves *etc*.
- **Linguagem de Manipulação de Dados (DML)**
  - Comandos de consulta, inserção, exclusão e modificação de dados
- **DML Embutida:**
  - Projetada para utilização em linguagens de programação de uso geral (Cobol, C, JAVA *etc*.)

UDESC

SQL – Do que é composta

- **Definição de Visões:** comandos para a definição de visões
- **Autenticação:** comandos para especificação de autorização de acesso as tabelas e as visões
- **Integridade:** comandos para especificação de regras de integridade
- **Controle de Transações:** comandos para especificação de início e fim de transações, e bloqueios de dados para controle de concorrência

Qual tecnologia vamos usar?

- Vamos utilizar o **Postgre SQL [**ou **pgSQL**]

- Vamos acessar o SGBD através do programa: **pgAdmin**

  Mas ... pode-se usar o modo terminal /prompt
  SQL Shell (psql)

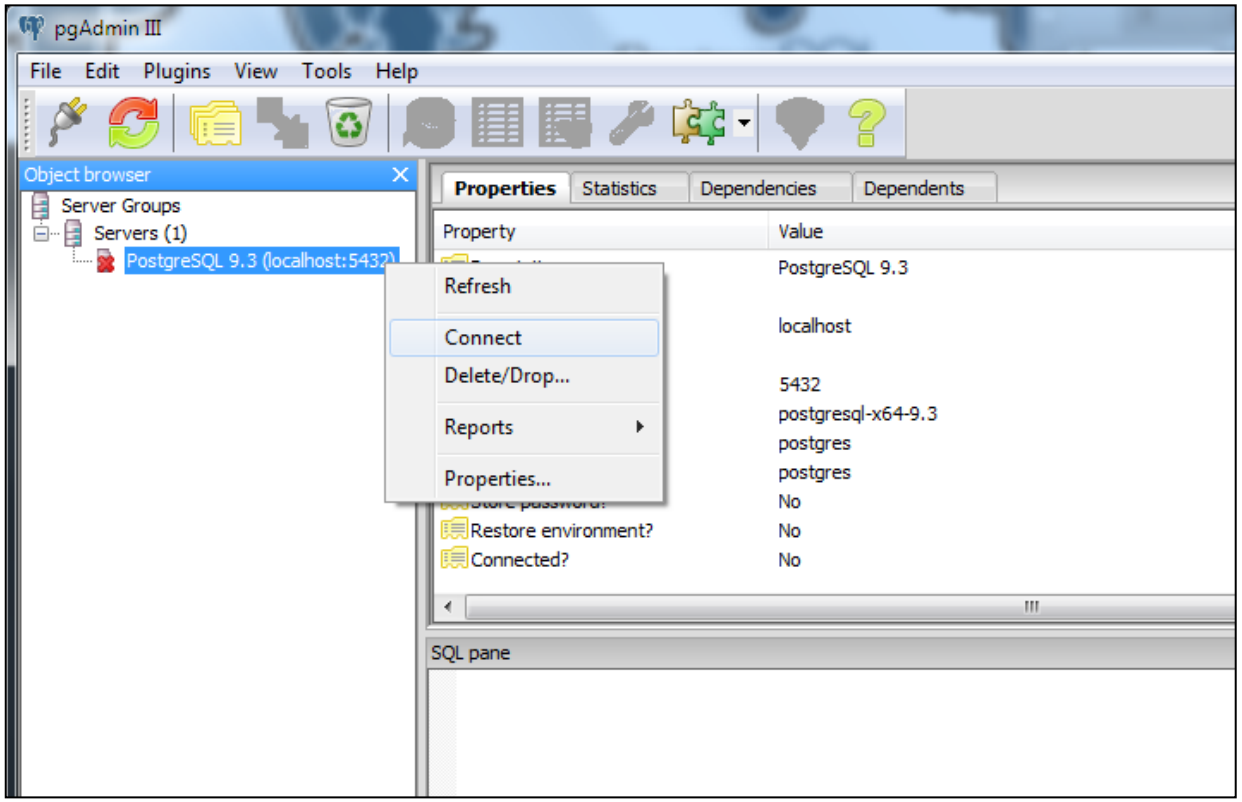
  – Qual é a senha? udesc (versão antiga)
  (na versão nova está sem senha)

- Mais para frente, vamos acessar o banco de dados com a linguagem de programação JAVA

PostgreSQL

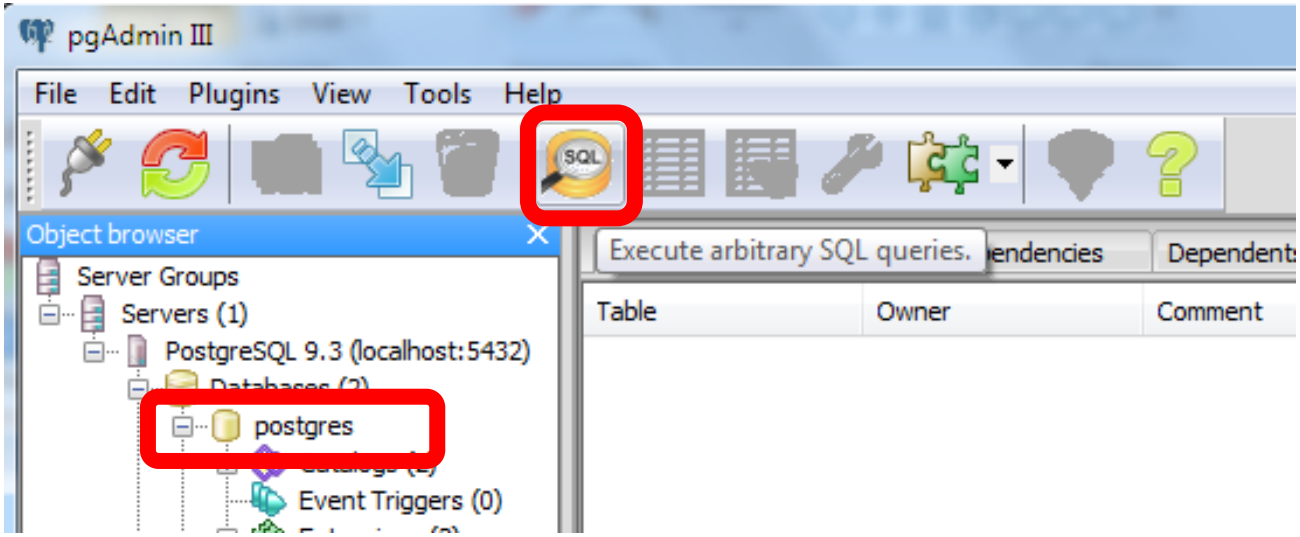
Qual tecnologia vamos usar?

- Instruções iniciais:
  - Conectar ao BD



Qual tecnologia vamos usar?

- Instruções iniciais:
  - Vamos escrever e executar scripts SQL na área própria: Selecione a base de dados que deseja utilizar e clique no botão destacado na figura



OBS: a conexão com o banco é feita sobre uma base de dados. Para executar scripts sobre outra base é necessário fazer nova conexão

UDESC

DDL – *Data Definition Language*

Mantendo Bases e Esquemas

- Especifica uma nova base de dados

```
CREATE DATABASE exemplo
    [WITH OWNER = postgres]
      [ENCODING = 'UTF8'];
```

[...] = opcional

- Especifica um novo esquema de tabelas (em uma base)

```
CREATE SCHEMA teste
       AUTHORIZATION nome;
```

- Exclui uma base de dados: `DROP DATABASE exemplo;`
- Exclui uma base de dados: `DROP SCHEMA teste;`

Criando Tabelas

- Para definir uma nova tabela e sua estrutura:

```
CREATE TABLE nome_tabela(
    nome_atributo_1 tipo_1 [*RESTRIÇÕES*],
    [{nome_atributo_n tipo_n [*RESTRIÇÕES*]},]
    [PRIMARY KEY (nome(s)_atributo(s)),]
    [FOREIGN KEY (nome_atributo)
       REFERENCES nome_tabela (nome_atributo)]
  );
```

Criando Tabelas

Exemplo

```
CREATE TABLE Cursos(
    id INT,
    nome VARCHAR(30) NOT NULL,
    PRIMARY KEY (id)
);
```

```
CREATE TABLE Alunos(
    matricula INT,
    nome VARCHAR(50) NOT NULL,
    curso_id INT,
    PRIMARY KEY (matricula),
    FOREIGN KEY (curso_id)
        REFERENCES Cursos (id)
  );
```

UDESC

Criando Tabelas

- **Principais tipos de dados:**
  - Numéricos: englobam números inteiros de vários tamanhos e pontos flutuantes (ver tabela a seguir)
  - Cadeia de Caracteres: podem ser de tamanho fixo [`char(n)`] ou variável [`varchar(n)`, `text`]
  - Booleano: assume valores TRUE e FALSO, mas pode ser também um valor NULL
  - Date: formato `aaaa-mm-dd`
  - Time: formato `hh:mm:ss`
  - Timestamp: formado por data e hora

Criando Tabelas

- **Detalhes sobre dados numéricos:**

nome	tamanho armazenamento	faixa de valores
tinyint	1 bytes	-128 a 127
smallint	2 bytes	-32.768 a +32.767
int	4 bytes	-2.147.483.648 a +2.147.483.647
bigint	8 bytes	-9.223.372.036.854.775.808 a ...
decimal	Variável	Sem limite
numeric	Variável	Sem limite
real	4 bytes	Precisão de 6 dígitos decimais
double precision	8 bytes	Precisão de 15 dígitos decimais

Criando Tabelas

- É possível utilizar atributos opcionais para dados numéricos:
  - `UNSIGNED`: indica que um tipo numérico não terá sinal e por consequência a faixa de valores é alterada
  - `ZEROFILL`: indica que os espaços vazios do campo serão preenchidos com zeros [PostgreSQL não implementa]
  - `(n)`: É possível especificar o "tamanho de apresentação", ex: `int`(5) => 00032 (OBS: faixa de valores não muda)
  - `(i,d)`: para os tipos `decimal` e `numeric` é possível especificar o número de dígitos totais e decimais, ex: `decimal`(6,2) => -9.999,99 até +9.999,99

Criando Tabelas

– O “tamanho de representação” para dados numéricos não é o número de bits utilizados para representar o número!

MySQL Docs, Numeric Type Attributes

MySQL supports an extension for optionally specifying the display width of integer data types in parentheses following the base keyword for the type. For example, **INT(4)** specifies an INT with a **display width of four digits**. This optional display width **may be used by applications to display** integer **values** having a width less than the width specified for the column **by left-padding them with spaces.** (That is, this width is present in the metadata returned with result sets. Whether it is used or not is up to the application.)

Criando Tabelas

– Cada SGBD possui as suas peculiaridades, podendo ter notações ou tipos de dados diferentes (específicos)

PostgreSQL disponibiliza tipos de dados para armazenamento de endereços IPv4, IPv6 e MAC:

`cidr` e `inet` – 12 ou 24 bytes
`macaddr` – 6 bytes

PostgreSQL não implementa o tamanho de apresentação para atributos numéricos

(…)

Criando Tabelas

- **Restrições / Atributos Opcionais**
  - `NOT NULL`: indica que o campo não pode ser nulo, ou seja, precisa necessariamente ter um valor
  - `PRIMARY KEY`: indica que um campo é chave primária;
  - `REFERENCES tabela (campo)`: indica que um campo é chave estrangeira
  - `UNIQUE`: indica que um campo deve ser único para cada registro / linha da tabela (não pode haver duplicatas

OBS: restrições PK e FK podem ser definidas no final

https://www.postgresql.org/docs/9.4/static/ddl-constraints.html

Criando Tabelas

- **Restrições / Atributos Opcionais**
  - `AUTO INCREMENT`: indica que será utilizado um valor incrementável automaticamente para o campo

    OBS: no PostgreSQL não há está restrição, mas sim um tipo de dado serial que possui este comportamento
  - `DEFAULT valor`: indica um valor padrão fixo para o campo caso um valor não seja especificado
  - `CHECK (predicado)`: indica que deverá ser realizar verificação de integridade de acordo com o predicado

Criando Tabelas

- Exemplos com Restrições

```
CREATE TABLE Materiais(
     id SERIAL PRIMARY KEY,
     nome VARCHAR(50) UNIQUE NOT NULL,
     preco NUMERIC(6,2)
);
```

```
CREATE TABLE Disciplinas(
       sigla CHAR(3) PRIMARY KEY,
       curso_id INT REFERENCES cursos (id),
       ch SMALLINT
);
```

Criando Tabelas

- Exemplos com Restrições (forma alternativa)

```
CREATE TABLE Materiais(
    id SERIAL,
    nome VARCHAR(50) NOT NULL,
    preco NUMERIC(6,2) CHECK (preco > 0),
    preco_vista NUMERIC(6,2),
    PRIMARY KEY (id),
--  CONSTRAINT pk PRIMARY KEY (id),
    UNIQUE (nome),
--  CONSTRAINT nome_diferente UNIQUE (nome),
    CHECK(preco_vista <= preco AND preco_vista > 0)
--  CONSTRAINT avista CHECK (preco_vista <= preco)
);
```

-- => comentário de linha

Criando Tabelas

- Exemplos com Restrições (forma alternativa)

```
CREATE TABLE Disciplinas(
        sigla CHAR(3),
        curso_id INT,
        ch SMALLINT,
        PRIMARY KEY (sigla),
        CONSTRAINT fk FOREIGN KEY (curso_id)
           REFERENCES cursos (id)
);
```

➢ **OBS**: Múltiplas chaves primárias, ou campos únicos podem ser definidos separando os nomes das colunas com vírgula

Atividade

Guarde/salve os comandos SQL. Vamos reutilizar esses dados!

1. Crie um BD com o nome Clinica
2. Crie as seguintes tabelas neste BD:
   - **Ambulatorios:** #nroa (int), andar (numeric(3), não nulo)
   - **Médicos:** #codm (int), nome (varchar(50), não nulo), idade (smallint, não nulo), especialidade (char(20)), CPF (numeric(11), único), cidade (varchar(30)), nroa (int)
   - **Pacientes:** #codp (int), nome (varchar(40), não nulo), idade (smallint), CPF (numeric(11), único), doenca (varchar(40), não nulo)
   - **Funcionarios**: #codf (int), nome (varchar(40), não nulo), idade (smallint), CPF (numeric(11), único), cidade (varchar(30)), salario (numeric(10)), cargo (varchar(20))
   - **Consultas**: #&*codm* (int), &*codp* (int), #data (date), #hora (time)
3. Crie uma restrição de verificação que garanta que não possa ser registrada uma consulta antes das 06:00 nem depois das 21:00.

FK – Chaves estrangeiras

Criando Tabelas

- ***Storage Engine*** **ou** ***Engine****:* são componentes do SGBD que são responsáveis pela realização das operações SQL

- PostgreSQL suporta e utiliza apenas um único e*ngine* que recebe o seu nome: PostgreSQL
- MySQL suporta e possibilita o uso de diversos *engines*, entre eles: InnoDB, MyISAM, MEMORY, CSV, ARCHIVE, BLACKHOLE, MERGE, FEDERATED, EXAMPLE

```
CREATE TABLE Disciplinas(
      [...]
) engine=MyISAM;
```

```
CREATE TABLE Disciplinas(
      [...]
) engine=InnoDB;
```

UDESC

Criando Tabelas

- No SQL os *engines* mais comuns são MyISAM e InnoDB:
  - *MyISAM*: modelo simples e eficiente
    - ✓ **contras**: não garante restrições de integridade referenciais
    - ✓ **prós**: simples e bom desempenho;
      bom para muitas leituras e poucas escritas;
      bom quando o controle de restrições está no código
  - *InnoDB:* modelo mais robusto
    - ✓ **prós**: garante restrições de integridade referenciais;
      bom para escritas concorrentes;
    - ✓ **contras**: um pouco mais lento do que o MyISAM

Apagando Tabelas

- Para excluir uma tabela existente pode-se utilizar:

```
DROP TABLE nome_tabela;
```

Alterando Tabelas

- Existem diversas alterações possíveis que podem ser feitas em uma tabela já existente:

```
ALTER TABLE nome_tabela
  RENAME TO novo_nome
  SET SCHEMA new_schema
  ADD [COLUMN] nome tipo [{Ris}]
  RENAME [COLUMN] nome_antigo TO nome_novo
  DROP [COLUMN] nome
  ALTER [COLUMN] nome [{SET|DROP} DEFAULT
      | {SET|DROP} NOT NULL | SET DATA TYPE tipo|...]
  ADD CONSTRAINT ri [PK|FK|CHECK|...]
  DROP CONSTRAINT nome_ri
  ...
```

https://www.postgresql.org/docs/9.1/static/sql-altertable.html

Alterando Tabelas

- Alguns exemplos

```
ALTER TABLE alunos
  RENAME TO graduandos

ALTER TABLE graduandos
  SET SCHEMA public

ALTER TABLE graduandos
  ADD COLUMN dt_nasc
  DATE NOT NULL

ALTER TABLE graduandos
  RENAME COLUMN dt_nasc
  TO data_nascimento
```

```
ALTER TABLE graduandos
  DROP COLUMN data_nascimento

ALTER TABLE graduandos
  ADD CONSTRAINT nroMat
  CHECK (matricula > 10000)

ALTER TABLE graduandos
  DROP CONSTRAINT nroMat

ALTER TABLE disciplinas
ALTER COLUMN ch
  SET DEFAULT (32)
```

Criando Índices

- **Índices** são utilizados para acelerar consultas a dados (assim como ocorre em um livro). Entretanto, índices ocupam espaço e devem ser utilizados com parcimônia
  - Índices são definidos automaticamente para PKs

```
CREATE [UNIQUE] INDEX nome_indice
  ON nome_tabela (nome_atrib [{, nome_atrib_n}])

DROP INDEX nome_indice ON nome_tabela
```

UDESC

Criando Sequencias

- O PostgreSQL utiliza **sequências** para valores do tipo serial (auto incrementáveis). As sequências podem ser alteradas com os seguintes comandos

```
ALTER SEQUENCE nome_sequencia
  INCREMENT BY valor -- escolhe o valor de inc.
  RESTART WITH valor -- reinicia com um valor
  MINVALUE valor -- define o valor mínimo
  MAXVALUE valor -- define o valor máximo
```

Atividade

4. Altere a tabela **Ambulatorios** criando a coluna `capacidade (smallint)` e a tabela **Pacientes** criando a coluna `cidade (varchar(30));`
5. Altere a tabela **Funcionarios** removendo a coluna `cargo`
6. Altere a tabela **Medicos**:
   a. crie uma FK para a coluna `nroa` que referencia a coluna `nroa` de **Ambulatórios**
   b. crie uma nova coluna `ativo (bool)` com valor padrão inicial verdadeiro (`true`)
7. O campo `doenca` foi mal interpretado e está no lugar errado. Corrija removendo este campo na tabela **Pacientes** e adicione-o na tabela **Consultas**

Guarde/salve os comandos SQL.
Vamos reutilizar esses dados!

UDESC

DML – *Data Manipulation Language*
Parte I

UDESC

DML

- A **DML** define operações de manipulação de dados:
  - Inserção, atualização, exclusão e seleção

- As instruções são declarativas, ou seja, especificam o que deve ser feito mas não como fazer

- **ATENÇÃO:** deve-se ter muito cuidado com as operações de atualização e exclusão pois podem causar danos irreversíveis nos registros das tabelas!
  (palavra a não esquecer: WHERE)

Inserção de Registros

- Para inserir valores podem-se utilizar dois modelos:
  - ❑ Apresentando todos os valores explicitamente:

```
INSERT INTO nome_tabela
  VALUES ( valores_separados_por_virgula )
```

  - ❑ Identificando quais colunas estão sendo listadas (demais colunas recebem valor padrão ou valor nulo)

```
INSERT INTO nome_tabela ( lista_colunas )
  VALUES ( valores_na_mesma_ordem )
```

OBS: caso algum campo com restrição não nulo não apareça na operação, o SGBD acusará um erro

Inserção de Registros

- Exemplos:

```
INSERT INTO materiais
   VALUES (1, 'livro BAN', 75.00, 70.00);
```

```
INSERT INTO materiais (nome, preco, preco_vista)
  VALUES ('teste', 125.50, 100.00);
```

OBS: caso o campo de sequencia não tenha sido atualizado, a primeira tentativa resultará em erro pois o SGBD tentará utilizar id=1, mas este valor já foi utilizado

```
INSERT INTO materiais (nome, preco, preco_vista)
  VALUES ('mouse', 45.00, 39.99),
         ('teclado', 75.50, 65.00);
```

OBS: nem todo SGBD permite a inserção de múltiplos registros em uma única cláusula

Inserção de Registros

- Exemplos das Restrições de Integridade na inserção:

OBS: o campo nome é obrigatório (NOT NULL). Vamos adicionar um novo registro sem nome e ver o que acontece

```
INSERT INTO materiais (preco, preco_vista)
  VALUES (99.90, 95.00);
```

OBS: a coluna nome foi marcada como UNIQUE. Vamos adicionar um novo registro com o mesmo nome e ver o que acontece

```
INSERT INTO materiais (nome, preco, preco_vista)
  VALUES ('teste', 175.50, 150.00);
```

OBS: preco_vista tem uma restrição assertiva (check) que verifica se este valor é menor do que o preço. E se não for, o que acontece?

```
INSERT INTO materiais (nome, preco, preco_vista)
  VALUES ('caneta atômica', 4.50, 5.00);
```

Alteração/Atualização de Registros

- Para alterar / atualizar registros utiliza-se:

```
UPDATE nome_tabela
   SET nome_atributo = valor
       [{, nome_atributo_n = valor}]
   [WHERE condição]
```

CUIDADO: caso um condicional WHERE não for definido, todos os registros da tabela serão alterados!!!

- Exemplos:

```
UPDATE materiais
 SET valor_vista = 65.00
 WHERE id = 1;
```

```
UPDATE materiais
 SET nome = 'livro EDA',
  preco_vista = preco * 0.9
 WHERE id = 2;
```

Alteração/Atualização de Registros

- Pode-se concatenar valores a colunas de tipo textual com o operador || ou a função CONCAT

```
UPDATE materiais
 SET nome = nome || 'ed.1'
 WHERE id = 2;
```

```
UPDATE materiais
 SET nome = CONCAT(nome, 'v.1', ', n.2')
 WHERE id = 1;
```

Exclusão de Registros

- Para excluir registros utiliza-se:

```
DELETE FROM nome_tabela
   [WHERE condição]
```

CUIDADO: caso um condicional WHERE não for definido, todos os registros da tabela serão excluídos!!!

- Exemplos:

```
DELETE FROM Ambulatorios
```

```
DELETE FROM materiais
WHERE id = 1
```

```
DELETE FROM materiais
WHERE nome = 'livro EDA'
   OR nome = 'livro BAN'
```

Operadores SQL

- Para as expressões condicionais é possível utilizar os seguintes operadores:
  - Operadores de comparação

    `= ; <> ; != ; > ; >= ; < ; <=`
  - Operadores lógicos

    `and; or ; not`

Chaves Estrangeiras (DDL)

- Para as `FOREIGN KEYS`, pode-se especificar o que deve ser feito caso uma operação sobre suas referencias seja realizada:
  - Definir qual a operação considerada:
    - `ON DELETE ; ON UPDATE`
  - Definir o comportamento:
    - NO ACTION: **padrão**, previne a alteração do registro se este estiver referenciado por outro registro.
    - RESTRICT: similar ao NO ACTION
    - SET NULL / SET DEFAULT: altera os registros que estiverem o referenciando para um valor nulo ou padrão;
    - CASCADE: propaga a alteração ou remoção aos registros que o estiverem referenciando

Chaves Estrangeiras (DDL)

- O comportamento deve ser especificado na criação da tabela ou então um comando de alteração:

```
CREATE TABLE Disciplinas(
        sigla CHAR(3) PRIMARY KEY,
        curso_id INT,
        ch SMALLINT,
        CONSTRAINT fk FOREIGN KEY (curso_id)
            REFERENCES cursos (id)
            ON DELETE SET NULL
            ON UPDATE CASCADE
);
```

Tente **atualizar** um id de curso antes e depois dessa operação
Tente **deletar** um registro de curso antes e depois dessa operação

Atividade

8. Popular as tabelas

Ambulatórios		
nroa	**andar**	**capacidade**
1	1	30
2	1	50
3	2	40
4	2	25
5	2	55

Funcionários					
codf	**nome**	**idade**	**cidade**	**salário**	**cpf**
1	Rita	32	Florianopolis	1200	20000100000
2	Maria	55	Palhoca	1220	30000110000
3	Caio	45	Biguacu	1100	41000100000
4	Denise	26	Florianopolis	1300	51000110000
5	Paula	33	Florianopolis	2500	61000111000
6	Carlos	44	Joinville	1200	30120120231

Atividade

Pacientes				
codp	**nome**	**idade**	**cidade**	**cpf**
1	Ana	20	Florianopolis	20000200000
2	Paulo	24	Palhoca	20000220000
3	Lucia	30	Biguacu	22000200000
4	Carlos	28	Joinville	11000110000

Medicos						
codm	**nome**	**idade**	**especialidade**	**cidade**	**cpf**	**nroa**
1	João	40	ortopedia	Florianopolis	10000100000	1
2	Maria	42	traumatologia	Blumenau	10000110000	2
3	Pedro	51	pediatria	São Jose	11000100000	2
4	Carlos	28	ortopedia	Joinville	11000110000	(nulo)
5	Márcia	33	neurologia	Biguacu	11000111000	3

UDESC

Atividade

Consultas				
codm	codp	data	hora	doenca
1	1	2016 / 06 / 12	14:00	gripe
1	4	2016 / 06 / 13	10:00	tendinite
2	1	2016 / 06 / 13	09:00	fratura
2	2	2016 / 06 / 13	11:00	fratura
2	3	2016 / 06 / 14	14:00	traumatismo
2	4	2016 / 06 / 14	17:00	checkup
3	1	2016 / 06 / 19	18:00	gripe
3	3	2016 / 06 / 12	10:00	virose
3	4	2016 / 06 / 19	13:00	virose
4	4	2016 / 06 / 20	13:00	tendinite
4	4	2016 / 06 / 22	19:30	dengue

Atividade

9. Realize as seguintes atualizações no BD:

   a. O paciente Paulo mudou-se para Ilhota
   b. A consulta do médico 1 com o paciente 4 foi remarcada para às 12:00 do dia 4 de Julho de 2006
   c. A paciente Ana fez aniversário – não precisa dar parabéns, apenas adicione 1 a sua idade (pode-se usar idade = idade+1)
   d. A consulta do médico Pedro (codm=3) com o paciente Carlos (codp=4) foi postergada para uma hora e meia depois
   e. O funcionário Carlos (codf=6) deixou a clínica (remover)
   f. Todas as consultas marcadas após as 19 horas foram canceladas (remova estes registros)

Atividade

9. Realize as seguintes atualizações no BD:

   g. Os médicos que residem em Biguacu e Joinville foram transferidos para outra clínica. Registrar como inativos. (utilize a coluna ativo da tabela de médicos)
   h. Adicione o sobrenome “Dantas” para a médica Maria