BANCO DE DADOS 1

FÁBIO ROBERTO OCTAVIANO

SILVANA MARIA AFFONSO DE LARA

Sumário

- Restrições de Integridade
- DDL
- Tipos de Domínio em SQL
- Definição de Esquemas em SQL
- Violação das Regras de Integridade
- Remoção de Esquemas em SQL
- Modificação de Relações em SQL

Restrições de Integridade

- Restrições de Integridade **são regras** a respeito dos **valores que podem ser armazenados** nas relações e que devem ser sempre satisfeitas, em quaisquer das relações de uma base de dados.

Restrições de Integridade

- Existem **3 formas de restrições** de integridade que são consideradas **necessárias** a uma BD relacional:
  - Restrições de unicidade da chave: uma chave qualquer (primária ou candidata) **não pode ter o mesmo valor em duas tuplas distintas** da mesma relação.
  - Restrições de integridade da entidade: a chave primária de qualquer relação **não pode ser NULA** em nenhuma tupla dessa relação.
  - Restrições de integridade referencial: chave estrangeira.

Chave Estrangeira

- A restrição de integridade referencial determina que o valor dos atributos que são **chaves estrangeiras** numa tupla qualquer t[C] da relação R1:
  - é igual ao valor t[D] na relação R2 onde D é *chave primária*

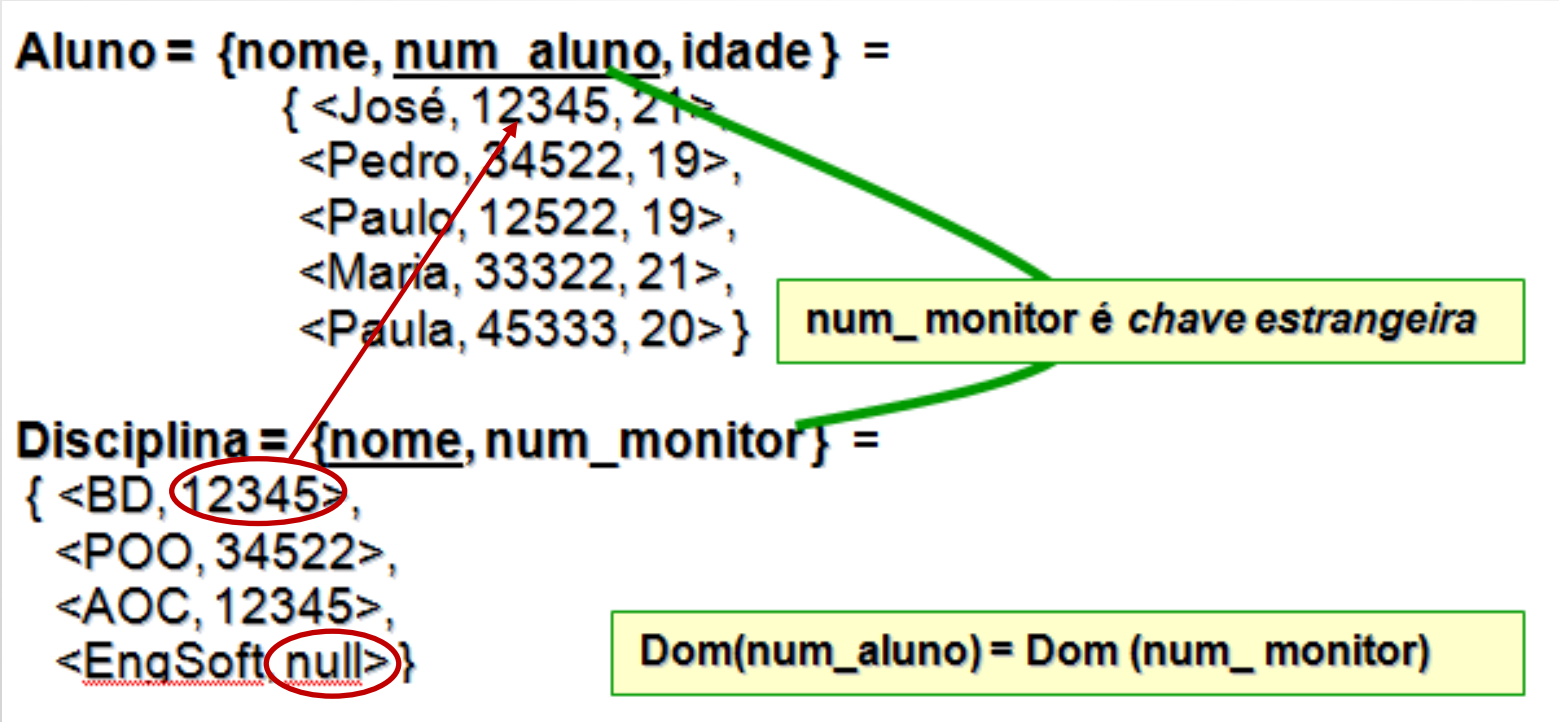
    ou
  - *é nulo*

- Em resumo: o valor de uma coluna de uma tabela que é chave estrangeira para outra tabela, deve ser:
  - Um valor existente da chave primária da relação a que referencia

    ou
  - *nulo*

Chave Estrangeira

- **Exemplo:**



SQL - DDL
Linguagem de Definição de Dados

- A Linguagem de Definição de Dados (DDL) é utilizada para a definição, exclusão e modificação de esquemas de relações em um banco de dados relacional

- Ela também permite que se definam o domínio dos valores associados a cada atributo, bem como as regras de integridade

Tipos de Domínio em SQL

- O padrão SQL-2003 aceita uma variedade de tipos de domínios embutidos, incluindo os seguintes:
- **char***(n):* cadeia de *n* caracteres (tamanho fixo)
- **varchar***(n):* cadeia de até *n* caracteres (tamanho variável)
- **integer,smallint:** números inteiros
- **numeric***(p,d):* número de *p* dígitos, com *d* casas decimais

Tipos de Domínio em SQL

continuação...

- **real, double:** números de ponto flutuante
- **float*(n)*:** número de ponto flutuante com precisão de *n* dígitos
- **date:** data de calendário
- **time:** horário
- **timestamp:** data completa especificando até fração de segundos.

Tipos de Domínio em SQL

- O padrão SQL:1999 introduz o tipo de domínio **blob** *(Binary Large Object)*
- Tipo utilizado quando se quer armazenar um atributo cujo valor típico é **grande** (vários Kbytes), em formato binário
- Exemplo: armazenar dentro do banco de dados um arquivo de imagem ou um arquivo de vídeo (uma foto de um cliente ou de um imóvel)

Tipos de Domínio em SQL

- Alguns SGBDs permitem a criação de nossos próprios tipos de domínios, através do comando *create domain:*

create domain nome_pessoa **char(20)**

- Assim, define-se um domínio do tipo *nome_pessoa*, que é uma cadeia de 20 caracteres, e que pode ser utilizado na declaração de atributos

Tipos de Domínio em SQL

- **Importante:** os tipos de domínios disponíveis, bem como o nome dos mesmos, varia entre as diversas implementações de SGBDs

- Muitos SGBDs possuem tipos de domínio próprios, ou seja, que não são comuns nem existem na linguagem SQL

Tipos de Domínio no MySQL

Tipo	Precisão	Valores
BIT / BOOL	1 byte	0 ou 1
TINYINT	1 byte	-128 a 127
SMALLINT	2 bytes	-32.768 a 32.767
MEDIUMINT	3 bytes	-8.388.608 a 8.388.607
INT / INTEGER	4 bytes	-2.147.483.648 a 2.147.483.647
BIGINT	8 bytes	...
FLOAT	4 bytes	-3,4028234E+38 a 1,175494E-38
DOUBLE / REAL	8 bytes	-1,7976931E+308 a -2,225073E-308
DECIMAL (M,D)	M + 2 bytes	...
NUMERIC (M,D)	M + 2 bytes	...

Tipos de Domínio no MySQL

Data e Hora:

Tipo	Precisão	Valores	Formato
Date	3 bytes	01/01/1001 a 31/12/9999	ano-mês-dia
DateTime	8 bytes	01/01/1001 00:00:00 a 31/12/9999 23:59:59	ano-mês-dia horas:min:seg
TimeStamp	4 bytes	01/01/1970 00:00:00 a 31/12/2037 23:59:59	aaaammddhhmmss
Year	1 byte	1901 a 2175	aaaa

Tipos de Domínio no MySQL

Caracteres:

Tipo	Precisão	Valores
Char (n)	n bytes	0 a 255 caracteres
Varchar (n)	n + 1 bytes	0 a 255 caracteres
TinyText / TinyBlob	Longitude + 1 byte	0 a 255 caracteres
Text / Blob	Longitude + 2 bytes	0 a 65.535 caracteres
MediumText / MediumBlob	Longitude + 3 bytes	0 a 16.777.215 caracteres
LongText / LongBlob	Longitude + 4 bytes	0 a 4.294.967.295 (limite de 16 Mb)
Enum / Set	1 a 8 bytes	Enum aceita até 65535 valores e Set aceita até 64 valores distintos

Criação de um BD no MySQL

- Para criarmos um novo BD em MySQL (chamado por ele de esquema), utilizamos:

  CREATE DATABASE nome_bd;

  onde:

  - **nome_bd** é o nome do banco de dados a ser criado;

- Para nos conectarmos a um BD existente em MySQL, utilizamos:

  USE nome_bd;

  onde:

  - **nome_bd** é o nome do banco de dados desejado.

Definição de Tabelas em SQL (DDL)

- Define-se uma relação em SQL da seguinte forma:

  CREATE TABLE R (

  $A_1D_1, A_2D_2, \ldots, A_nD_n,$

  <regra de integridade $_1$ >

  ...,

  <regra de integridade $_k$ >

  );

- onde:
  - **R** é o nome da relação;
  - cada $\mathbf{A_i}$ é o nome de um atributo da relação;
  - cada $\mathbf{D_i}$ é o tipo de domínio do atributo correspondente A_i;

Definição de Tabelas em SQL (DDL)

- As **regras de integridade** podem ser:
  - Definição de chaves primárias (**primary key**)
  - Definição de chaves estrangeiras (**foreign key**)
  - Definição de checagens (**check**)

Definição de Tabelas em SQL (DDL)

- Para cada domínio $\mathbf{D_i}$ também podemos especificar:
  - Não permissão de valores nulos (**not null**)
  - Não permissão de valores repetidos (**unique**) – *chaves candidatas!*
  - Definição de valores padrão (**default**) – em algumas implementações

Definição de Tabelas em SQL (DDL)

- Em resumo, podemos definir para as tabelas:
  - **PRIMARY KEY**
  - **FOREIGN KEY**
  - **NOT NULL**
  - **UNIQUE**
  - **DEFAULT**
  - **CHECK**

- Elas são chamadas constraints, pois são restrições a serem verificadas pelo SGBD quando ocorre inclusão, alteração ou remoção de tuplas na tabela.

DDL - Exemplo

- Criando uma relação e definindo restrições de domínio e chave primária:

```
create table estudante (
        id_estudante        integer         not null,
        nome                varchar(25)     not null unique ,
        nivel               varchar(15)     not null,
        periodo             integer         default 1,
        primary key (id_estudante),
        check (nivel in ('Graduacao', 'Mestrado', 'Doutorado'))
);
```

DDL - Exemplo

- Uma vez que uma tabela foi criada, podemos utilizar o comando DESC para descrever a sua estrutura de colunas.
- Exemplo:

  DESC estudante;

Field	Type	Null	Key	Default	Extra
id_estudante	int(11)	NO	PRI	NULL	
nome	varchar(25)	NO	UNI	NULL	
nivel	varchar(15)	NO		NULL	
periodo	int(11)	YES		1	

DDL - Exemplo

- **Observação Importante**: O comando **check** é aceito mas só começou a funcionar no MySQL a partir da versão 8.0.16.

- Quando se utiliza versão anterior do MySQL, as regras de verificação com check devem ser feitas na aplicação.

DDL – Auto Increment

- No MySQL é possível criar uma coluna numérica como auto increment, isto é, o próprio SGBD se encarregará de popular essa coluna incrementando seu valor em 1 a cada inserção de um novo registro.

- Nesse caso, não informamos a coluna auto increment no commando insert.

DDL – Auto Increment

- Exemplo:

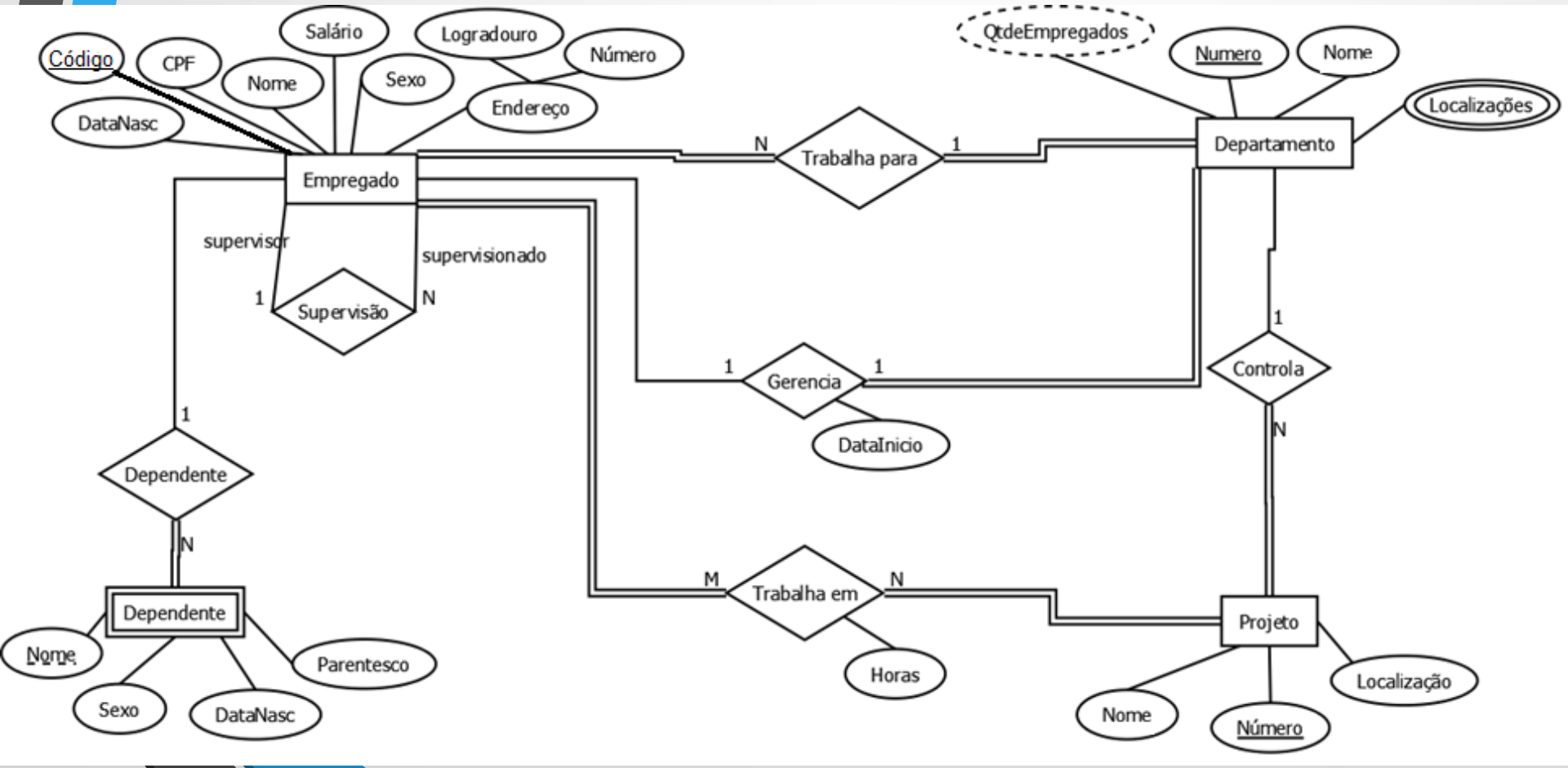
```
CREATE TABLE teste (
        codigo          int         AUTO_INCREMENT,
        descricao       varchar(30),
        observacao varchar(100),
        primary key(codigo)
);

INSERT INTO teste (descricao, observacao) VALUES ('xxxxx', 'yyyyy');
```

A tupla com descrição xxxxx e observação yyyyy será inserida com o código 1.

DDL - Exemplo

- Criar uma base de dados completa, definindo restrições de domínio, chave primária e integridade referencial (*foreign keys*) para o MER:



DDL - Exemplo

- Mapeamento para o Modelo Relacional:

EMPREGADO (codigo, cpf, sexo, datanasc, nome, salário, logradouro, nro, depto_num, codigo_ger)

depto_num: FK DEPARTAMENTO (numero), codigo_ger: FK EMPREGADO (codigo)

DEPARTAMENTO (numero, nome, qtde_empregados, codigo_ger, data_inicio_ger)

codigo_ger: FK EMPREGADO (codigo)

PROJETO (numero, nome, localização, num_dep)

num_dep: FK DEPARTAMENTO (numero)

DEPENDENTE (codigo_emp, nome, cpf, sexo, parentesco, datanasc)

codigo_emp: FK EMPREGADO (codigo)

DEPTO_LOCAL (depto_num, localizacao)

depto_num: FK DEPARTAMENTO (numero)

TRABALHA_EM (codigo_emp, projeto_num, horas)

codigo_emp: FK EMPREGADO (codigo), projeto_num: FK PROJETO (numero)

DDL - Exemplo

- Criando a tabela Empregado...

```
CREATE TABLE empregado (
        codigo          int NOT NULL,
        nome            varchar(60) NOT NULL,
        cpf             varchar(14) NOT NULL,
        sexo            char(1),
        datanasc        date,
        logradouro      varchar(50),
        nro             int,
        salario         decimal(8,2) NOT NULL,
        depto_num       int,
        codigo_ger      int,
        CONSTRAINT empregado_pk PRIMARY KEY (codigo),
        CONSTRAINT empregado_sexo_ck CHECK (sexo in ('M','F','m','f')),
        CONSTRAINT empregado_salario_ck CHECK (salario > 0)
);
```

DDL - Exemplo

- Criando a tabela Departamento...

```
CREATE TABLE departamento (
        numero              int NOT NULL,
        nome                varchar(20) NOT NULL,
        codigo_ger          int,
        data_inicio_ger     date,
        CONSTRAINT depto_pk PRIMARY KEY (numero),
        CONSTRAINT depto_nome_uk UNIQUE (nome)

);
```

Modificação de Tabelas em SQL

- Modificamos o esquema de uma tabela (isto é, adicionamos ou removemos atributos) por meio dos comandos:

ALTER TABLE R **ADD** A D;

- Adiciona um novo atributo A, de domínio D na tabela R.
- O valor inicial do novo atributo em todas as tuplas será nulo (**null**).

ALTER TABLE R **DROP** A;

- Remove um atributo **A** existente na tabela **R** (nem todos SGBDs permitem isso).

Modificação de Tabelas em SQL

- Algumas implementações (SGBD) também permitem que se **altere o domínio dos atributos** por meio do comando **modify** :

ALTER TABLE R **MODIFY** A D1;

- Modifica o domínio do atributo já existente **A** da tabela **R** para o novo domínio **D1** especificado.
- Caso existam valores na tabela, eles devem ser compatíveis com o novo domínio.
- Para modificar o nome do atributo de A para A1, utilizamos:

ALTER TABLE R **CHANGE** A A1 D;

- Onde D é o domínio do atributo renomeado.

Modificação de Tabelas em SQL

- Exemplos:
  - Adicionar a coluna fone à tabela Empregado:

    alter table empregado **add** fone **varchar(20)**;

  - Modificar o tipo (domínio) da coluna fone:

    alter table empregado **modify** fone **varchar(16)**;

  - Renomear a coluna nome para fone_emp:

    alter table empregado **change** fone fone_emp varchar(16);

  - Remover a coluna fone da tabela Empregado:

    alter table empregado **drop** fone_emp;

DDL - Exemplo

- De acordo com o Modelo Relacional, a coluna depto_num da tabela Empregado deve ser uma FK que referencia um Departamento.
- Quando criamos a tabela Empregado, essa FK não pode ser criada porque a tabela Departamento ainda não existia (não tinha como referenciá-la).
- E agora para inserirmos a chave estrangeira do departamento na tabela de funcionários que já está criada?
- É preciso alterar a estratura da tabela, adicionando a FK. Para tanto, utilizamos o commando ALTER TABLE:

  ALTER TABLE empregado ADD CONSTRAINT EMP_DEP_FK FOREIGN KEY(depto_num) REFERENCES departamento(numero);

  ALTER TABLE empregado ADD CONSTRAINT EMP_GER_FK FOREIGN KEY(codigo_ger) REFERENCES empregado(codigo);

DDL – Modificando Constraints

- Para apagar uma PRIMARY KEY de uma tabela utilizamos:
  ALTER TABLE *nome_tabela* DROP PRIMARY KEY;
- Para apagar uma UNIQUE de uma tabela utilizamos:
  ALTER TABLE *nome_tabela* DROP INDEX *nome_index_unique*;
- Para apagar uma FOREIGN KEY de uma tabela utilizamos:
  ALTER TABLE *nome_tabela* DROP FOREIGN KEY *nome_foreign_key*;
- Para modificar uma coluna e torná-la NOT NULL ou NULL, utilizamos:
  ALTER TABLE *nome_tabela* MODIFY *nome_coluna* TIPO NOT NULL;
  ALTER TABLE *nome_tabela* MODIFY *nome_coluna* TIPO NULL;
- Para adicionar valor default a uma coluna, utilizamos:
  ALTER TABLE *nome_tabela* MODIFY *nome_coluna* TIPO DEFAULT *valor*;

DDL – Modificando Constraints

- Para adicionar uma PRIMARY KEY a uma tabela utilizamos:
  ALTER TABLE *nome_tabela* ADD {CONSTRAINT *nome*} PRIMARY KEY(*coluna*);
- Para adicionar uma UNIQUE a uma tabela utilizamos:
  ALTER TABLE *nome_tabela* ADD UNIQUE *(colunas)*;
- Para adicionar uma FOREIGN KEY a uma tabela utilizamos:
  ALTER TABLE *nome_tabela* ADD {CONSTRAINT *nome_foreign_key*}
  FOREIGN KEY(*coluna*) REFERENCES *nome_tabela(coluna)*;

Modificação de Tabelas em SQL

- **Observação Importante**: vale lembrar que, dependendo do SGBD a ser utilizado, algumas funcionalidades não são possíveis
- O contrário também pode existir, ou seja, um SGBD pode incluir definições próprias (a mais do que a SQL prevê)

Remoção de Tabelas em SQL

- Para remover **uma tabela** em SQL, usamos o comando:

  DROP TABLE R;

- Esse comando remove todos os dados inseridos na tabela, e também remove a definição do esquema da mesma, ou seja, para inserir dados nessa tabela será necessário criá-la novamente com o comando **create table.**

Exercícios

1. Crie a base de dados Clube e, em seguida, conecte-se nela ;
2. Crie as tabelas JOGADOR e EQUIPE, conforme especificação a seguir:

JOGADOR

Coluna	Tipo	Tamanho	Restrições
ID	SMALLINT		PK
NOME	VARCHAR	50	NOT NULL
CPF	VARCHAR	14	UNIQUE
DATA_NASC	DATE		
SALARIO	NUMERIC	8, 2	CHECK > 0
ID_EQUIPE	SMALLINT		FK (Equipe)

EQUIPE

Coluna	Tipo	Tamanho	Restrições
ID	SMALLINT		PK
NOME	VARCHAR	30	NOT NULL

Exercícios

3. Modifique a estrutura da tabelas JOGADOR, adicionando a coluna CONTATO, do tipo VARCHAR de tamanho 10;
4. Modifique a coluna CONTATO da tabela JOGADOR, passando para o tipo VARCHAR de tamanho 20;
5. Adicione a coluna EMAIL à tabela JOGADOR, sendo uma coluna de texto variável com tamanho máximo 60 caracteres.
6. Elimine a coluna EMAIL criada no passo anterior da tabela JOGADOR.