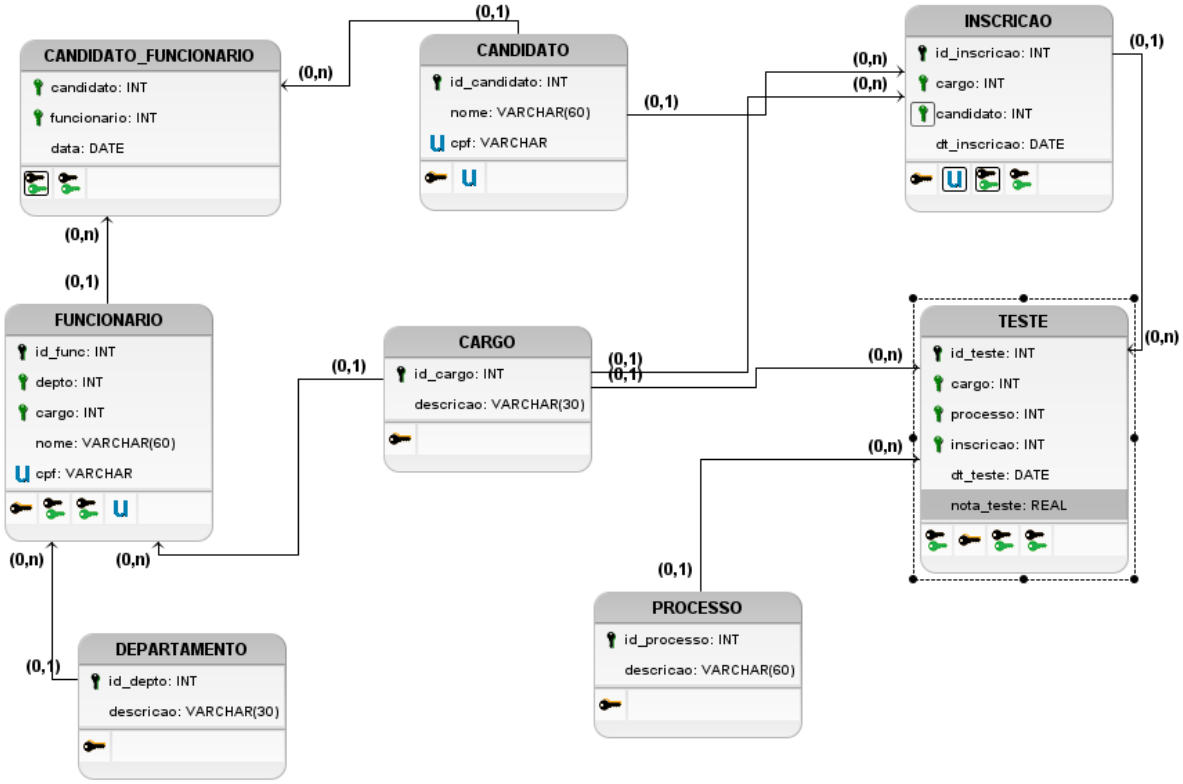
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA PARAÍBA Campus Campina Grande	INSTITUTO FEDERAL DA PARAÍBA. CURSOS: ENGENHARIA DE COMPUTAÇÃO/TELEMÁTICA DISCIPLINA: BANCO DE DADOS

ROTEIRO PRÁTICO – SQL: LINGUAGEM DE DEFINIÇÃO DE DADOS (DDL)

OBJETIVO: Neste roteiro vamos explorar os comandos SQL DDL e como eles auxiliam a definir e gerenciar a estrutura dos objetos do banco de dados. Vamos explorar as principais funcionalidades dos comandos CREATE e ALTER.

Considere o seguinte modelo de dados para gerenciar o armazenamento de dados sobre **processos seletivos**. Nesse modelo são apresentadas as tabelas com seus campos, tipos e chaves, assim como seus relacionamentos.



Utilizando um ambiente de **SQL Shell,** conecte em um **servidor MySQL** como administrador e faça o que se pede. Ao final deste roteiro deve ser entregue através da sala de aula virtual, um **documento em formato PDF** com as capturas de tela (*screenshot*) do resultado de alguns comandos solicitados ao longo deste roteiro. Esse documento deve conter o cabeçalho com nome da instituição, curso, disciplina, período letivo, nome completo do discente e número de matrícula e ser enviado pela sala de aula virtual dentro do prazo estabelecido pelo docente responsável.

1) Crie um banco de dados com o nome **bdselecaoMATRICULA**, por meio do seguinte comando: (Substitua MATRICULA por seu número de matricula, por exemplo, para a matricula 202311250099 o banco de dados se chamará **bdselecao202311250099**)

```
CREATE DATABASE bdselecaoMATRICULA;
```

2) Para verificar a lista de banco de dados existentes no servidor, execute o comando:

```
SHOW DATABASES;
```

Capture e salve a tela com a saída deste comando.

Obs: este comando é específico para o SGBD MySQL (ou MariaDB) e pode não funcionar em outras implementações.

3) A partir daqui, vamos nos referir ao banco de dados criado anteriormente somente como **bdselecao** (ao invés de bdselecaoMATRICULA). Agora conecte ao banco de dados **bdselecao** com o seguinte comando:

```
USE bdselecao;
```

Obs: este pode não funcionar em outras implementações de SGBD.

4) Certifique que está conectado ao banco **bdselecao** para iniciar a criar as tabelas no banco de dados. Agora, crie a tabela CANDIDATO_FUNCIONARIO:

```
CREATE TABLE CANDIDATO_FUNCIONARIO(
    funcionario integer,
    candidato integer,
    dt_indicacao DATE
);
```

5) Agora exiba a estrutura da tabela CANDIDATO_FUNCIONARIO com o seguinte comando:

```
DESCRIBE CANDIDATO_FUNCIONARIO;
```

Obs: este pode não funcionar em outras implementações de SGBD.

6) Crie a tabela CANDIDATO e em seguida descreva a estrutura da tabela:

```
CREATE TABLE CANDIDATO (
    id_candidato integer,
    nome VARCHAR(60),
    cpf VARCHAR(14)
);

desc CANDIDATO;
```

7) Crie a tabela INSCRICAO e em seguida descreva a estrutura da tabela:

```
CREATE TABLE INSCRICAO (
    id_inscricao integer,
    candidato integer,
    cargo integer,
    dt_inscricao date
);

desc INSCRICAO;
```

8) Crie as tabelas TESTE, CARGO, FUNCIONARIO, DEPARTAMENTO, PROCESSO_SELETIVO. Veja os comandos a seguir:

```
CREATE TABLE TESTE (
    id_teste            integer,
    processo          integer,
    cargo             integer,
    inscricao          integer,
    dt_teste             DATE
);
```

```
CREATE TABLE CARGO (
    id_cargo            integer,
    descricao           VARCHAR(30)
);
```

```
CREATE TABLE DEPARTAMENTO (
    id_depto            integer,
    descricao           VARCHAR(30)
);
```

```
CREATE TABLE FUNCIONARIO (
    id_func             integer,
    depto           integer,
    cargo           integer,
    nome                VARCHAR(60),
    cpf                 VARCHAR(14),
);
```

```
CREATE TABLE PROCESSO_SELETIVO (
    id_processo         integer,
    descricao       VARCHAR(60)
);
```

9) Agora liste as tabelas existentes no banco de dados **bdselecao**. Execute o comando: (certifique-se que todas as tabelas estejam criadas).

```
SHOW TABLES;
```

Capture e salve a tela com a saída deste comando.

Obs: este pode não funcionar em outras implementações de SGBD.

10) Note que até o momento as tabelas foram criadas apenas com as definições dos campos e tipos sem nenhuma restrição. A partir de agora vamos adicionar os diferentes tipos de restrições (*constraints*) estudados em sala de aula. Uma vez que as tabelas já estão criadas vamos utilizar variações do comando ALTER TABLE.
Adicione as chaves primárias nas tabelas FUNCIONARIO e CANDIDATO.

```
ALTER TABLE FUNCIONARIO ADD CONSTRAINT PK_func PRIMARY KEY (id_func);
ALTER TABLE CANDIDATO ADD CONSTRAINT PK_candidato PRIMARY KEY(id_candidato);
```

11) Adicione a chave primária na tabela CANDIDATO_FUNCIONARIO. Note que se trata de uma chave concatenada.

```
ALTER TABLE CANDIDATO_FUNCIONARIO ADD CONSTRAINT PK_cand_func PRIMARY KEY
(funcionario,candidato);
```

12) Agora vamos implementar o relacionamento N:N entre tabelas CANDITATO e FUNCIONARIO que em nível lógico é implementado com a criação da tabela CANDIDATO_FUNCIONARIO relacionada as tabelas anteriormente citadas (dois relacionamentos 1:N). A implementação de um relacionamento em nível físico se dá por meio da criação de uma chave estrangeira. Execute os comandos a seguir.

```
ALTER TABLE CANDIDATO_FUNCIONARIO ADD CONSTRAINT FK_cand_func FOREIGN KEY (funcionario)
REFERENCES FUNCIONARIO(id_func);
ALTER TABLE CANDIDATO_FUNCIONARIO ADD CONSTRAINT FK_cand_func_cand FOREIGN KEY (candidato)
REFERENCES CANDIDATO(id_candidato);
```

13) Agora verifique na descrição da tabela CANDIDATO_FUNCIONARIO a inclusão das chaves estrangeiras. Você consegue compreender onde essa informação localizada? Inclua a resposta à pergunta no documento de entrega.

```
DESC CANDIDATO_FUNCIONARIO;
```

Capture e salve a tela com a saída deste comando.

14) Adicione as chaves primárias nas tabelas INSCRICAO e CARGO.

```
ALTER TABLE INSCRICAO ADD CONSTRAINT PK_inscricao PRIMARY KEY (id_inscricao);
ALTER TABLE CARGO ADD CONSTRAINT PK_cargo PRIMARY KEY (id_cargo);
```

15) Implemente o relacionamento entre as tabelas INSCRICAO e CANDIDATO, e em seguida, o relacionamento entre INSCRICAO e CARGO.

```
ALTER TABLE INSCRICAO ADD CONSTRAINT FK_INSC_candidato FOREIGN KEY (candidato) REFERENCES
CANDIDATO(id_candidato);
ALTER TABLE INSCRICAO ADD CONSTRAINT FK_INSC_cargo FOREIGN KEY (cargo) REFERENCES
CARGO(id_cargo);
```

16) Adicione as chaves primárias nas tabelas TESTE e PROCESSO_SELETIVO.

```
ALTER TABLE TESTE ADD CONSTRAINT PK_Teste PRIMARY KEY (id_teste);
ALTER TABLE PROCESSO_SELETIVO ADD CONSTRAINT PK_processo PRIMARY KEY (id_processo);
```

17) Implemente os relacionamentos entre seguintes tabelas: TESTE e PROCESSO, em seguida, TESTE e CARGO, e, depois, TESTE e INSCRICAO.

```
ALTER TABLE TESTE ADD CONSTRAINT FK_teste_processo FOREIGN KEY (processo) REFERENCES
PROCESSO_SELETIVO(id_processo);
ALTER TABLE TESTE ADD CONSTRAINT FK_teste_cargo FOREIGN KEY (cargo) REFERENCES
CARGO(id_cargo);
ALTER TABLE TESTE ADD CONSTRAINT FK_teste_inscricao FOREIGN KEY (inscricao) REFERENCES
INSCRICAO(id_inscricao);
```

18) Adicione a chave primária na tabela DEPARTAMENTO e em seguida implemente os relacionamentos entre as seguintes tabelas: FUNCIONARIO e DEPARTAMENTO, e, FUNCIONARIO e CARGO.

```
ALTER TABLE DEPARTAMENTO ADD CONSTRAINT PK_depto PRIMARY KEY (id_depto);
ALTER TABLE FUNCIONARIO ADD CONSTRAINT FK_func_depto FOREIGN KEY (depto) REFERENCES
DEPARTAMENTO(id_depto);
ALTER TABLE FUNCIONARIO ADD CONSTRAINT FK_func_depto FOREIGN KEY (cargo) REFERENCES
CARGO(id_cargo);
```

19) Neste ponto, foram implementadas todas as tabelas com as devidas restrições de chave primária e as restrições de integridade referencial (chaves estrangeiras), isto é, os relacionamentos entre as tabelas. Agora, vamos acrescentar as tabelas outros tipos de restrição.
Por padrão, ao adicionarmos um campo em uma tabela, esse campo opcional, com exceção do campo que é chave primária. Vamos adicionar obrigatoriedade a campos de algumas tabelas.

```
ALTER TABLE CANDIDATO_FUNCIONARIO MODIFY dt_indicacao DATE NOT NULL;
```

O comando anterior, modifica a coluna dt_indicacao e a torna obrigatória, ou seja, não nula.

Obs: Esse comando pode não funcionar em outro SGBD. Por exemplo, no PostgreSQL, o comando equivalente é o seguinte:
ALTER TABLE CANDIDATO_FUNCIONARIO ALTER COLUMN dt_indicacao SET NOT NULL;

20) Neste ponto, foram implementadas todas as tabelas com as devidas restrições de chave primária e as restrições de integridade referencial (chaves estrangeiras), isto é, os relacionamentos entre as tabelas. Agora, vamos acrescentar as tabelas outros tipos de restrição.
Por padrão, ao adicionarmos um campo em uma tabela, esse campo opcional, com exceção do campo que é chave primária. Vamos adicionar obrigatoriedade a campos de algumas tabelas.

```
ALTER TABLE CANDIDATO_FUNCIONARIO MODIFY dt_indicacao DATE NOT NULL;
```

21) Quais os comandos tornam obrigatório os campos nome e cpf da tabela CANDIDATO e dt_inscricao da tabela INSCRICAO?

Capture e salve a tela com a execução destes comandos.

22) Agora vamos adicionar uma coluna chamada **nota** na tabela TESTE.

```
ALTER TABLE TESTE ADD COLUMN nota real NOT NULL;
```

23) Para ficar coerente ao modelo apresentado no inicio deste roteiro, vamos renomear a coluna **nota** da tabela TESTE, adicionada no item anterior, para **nota_teste**.

```
ALTER TABLE TESTE CHANGE COLUMN nota nota_teste real NOT NULL;
```

Obs: Esse comando pode não funcionar em outro SGBD. Por exemplo, no PostgreSQL, o comando equivalente é o seguinte: ALTER TABLE TESTE RENAME COLUMN nota TO nota_teste;

24) Agora vamos adicionar algumas restrições semânticas. Isso será feito pela cláusula **CHECK**. Execute os comandos a seguir.

```
ALTER TABLE TESTE ADD CONSTRAINT CK_nota CHECK (nota_teste>0);
```

O comando anterior faz com que a coluna **nota_teste** só aceite valores positivos.

```
ALTER TABLE TESTE ADD CONSTRAINT CK_nota_2 CHECK (nota_teste BETWEEN 0 AND 100);
```

Esse comando faz com que a coluna **nota_teste** só aceite valores entre 0 e 100, inclusive.

25) O comando a seguir inclui um valor padrão para uma coluna de uma tabela. Nesse caso, é atribuído o valor 7 por padrão na coluna **nota_teste**. Vale ressaltar que esse comando é útil para o caso da coluna da tabela ser opcional (NULL). Explique o porquê dessa afirmação. Inclua a explicação no documento de entrega.

```
ALTER TABLE TESTE ALTER nota_teste SET DEFAULT 7;
```

Note que a palavra **COLUMN** (da expressão ALTER COLUMN) é opcional.

26) Execute o comando DESCRIBE para todas as tabelas do banco de dados.

Capture e salve a saída na tela resultado da execução destes comandos.

27) Renomeia a tabela PROCESSO_SELETIVO para PROCESSO. Use o seguinte comando:

```
RENAME TABLE PROCESSO_SELETIVO TO PROCESSO;
```

Obs: Esse comando pode não funcionar em outro SGBD. Por exemplo, no PostgreSQL, o comando equivalente é o seguinte: ALTER TABLE PROCESSO_SELETIVO RENAME TO PROCESSO;

28) Pesquise comandos para excluir uma coluna de uma tabela e para excluir uma tabela. Inclua um exemplo de cada comando no documento de entrega.