FIAP GRADUAÇÃO

FIAP

SISTEMAS DE INFORMAÇÃO

DESIGN E DESENVOLVIMENTO DE BANCO DE DADOS

PROF. Luciano Melo

FIAP

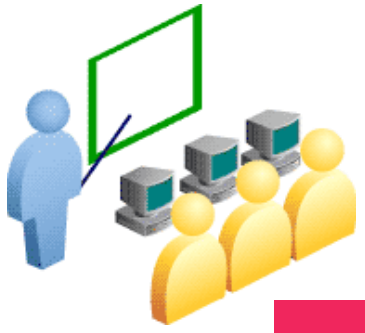
SQL

Comandos DDL

Criando e Gerenciando Tabelas no Banco de Dados

Agenda

- Tipos de Linguagens dentro do SQL
- Objetos típicos de um banco de dados
- Tipos de Dados
- Comandos DDL
- Tables
  - Criar / Alterar e Remover Tabelas
  - Constraints
  - Truncate
- Indexes



DDL

Data Definition Language

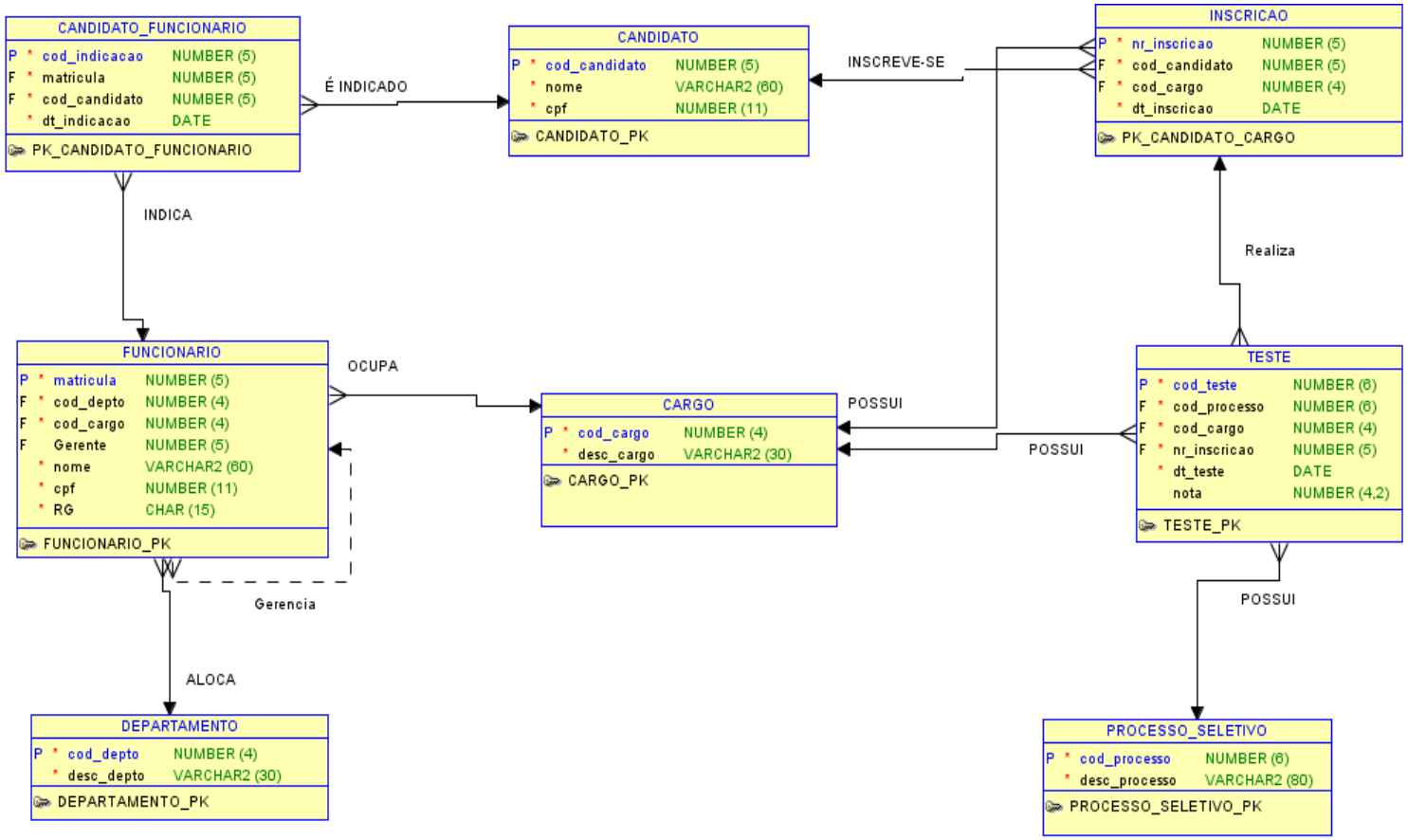
DDL – Data Definition Language

Linguagem de Definição de Dados

- **Comandos DDL são aqueles usados para definir objetos em um banco de dados, como tabelas e índices, por exemplo.**
- **Tipicamente envolvem os comandos:**
  - **CREATE: Para criar o objeto**
  - **ALTER: Para alterar a definição do objeto**
  - **DROP: Para remover o objeto**
  - **TRUNCATE*:* Para esvaziar o conteúdo de uma tabela**

DDL – Data Definition Language

São com os comandos DDL da linguagem SQL que criamos o banco de dados, alteramos estruturas (objetos) ou simplesmente removemos (“dropamos”)



Tipos de Objetos em um BD

Objetos Básicos de um Banco de Dados

Objeto	Descrição
Tabela	Unidade básica de armazenamento; composta de linhas
View	Representa logicamente subconjuntos de dados de uma ou mais tabelas
Seqüência	Gera valores numéricos (Oracle)
Índice	Melhora o desempenho de algumas consultas
Sinônimo	Fornece nomes alternativos a objetos

Neste tópico, vamos falar sobre **Tabelas e ìndices**

Objetos

Regras de Nomeação

Os nomes de tabelas e colunas:

- **Devem começar com uma letra**
- **Devem ter de 1 a 30 caracteres**
- **Devem conter apenas os caracteres A a Z, a a z, 0 a 9, _, $ e #**
- **Não devem duplicar o nome de outro objeto pertencente ao mesmo usuário**
- **Não devem ser palavras reservadas do servidor Oracle**

Atributos: Dicas de Nomeclatura

Nomenclatura Sugerida

Importante seguir uma padronização para sua nomenclatura:

COD	CD	= Código
NUM	NR	= Número
DES	DS	= Descrição
NOM	NM	= Nome
DAT	DT	= Data
VAL	VL	= Valor
QTD	QT	= Quantidade
SIG	SG	= Sigla

Tipos de Dados

Tipo de Dados	Descrição
VARCHAR2(*size*)	Dados de caractere de tamanho variável
CHAR(*size*)	Dados de caractere de tamanho fixo
NUMBER(*p*,*s*)	Dados numéricos de tamanho variável
DATE	Valores de datas e horários
LONG	Dados de caractere de tamanho variável (até 2 GB)
CLOB	Dados de caractere (até 4 GB)
RAW e LONG RAW	Dados binários brutos
BLOB	Dados binários (até 4 GB)
BFILE	Dados binários armazenados em um arquivo externo (até 4 GB)
ROWID	Um sistema numérico de base 64 que representa o endereço exclusivo de uma linha na tabela correspondente

Tabela com principais tipos de Dados e SGBDRs

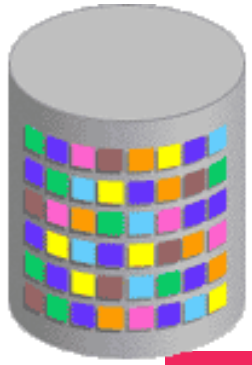
Padrão / Tipo	SQL Ansi 92	DBF III+	Paradox 7	MS-Access 2000	Oracle 8	MS-SQL Server 7	Interbase 4.2	MySQL 3.23
Texto	Char Varchar	Character	Alpha	Text Memo	Char Varchar2 Long	Char Varchar	Char Varchar	Char Varchar
Data	Date	Date	Date	Datetime	Date	Datetime	Date	Datetime
Hora	Time	-	Time	Datetime	Date	Datetime	Date	Datetime
Número	Numeric Integer Float	Numeric	- Long Integer Number	- Long Double	Number	Numeric Int Real	Numeric Integer Float	Numeric Integer Double
Moeda	-	-	Money	Currency	-	Money	-	-
Lógico	-	Logic	Logical	Bit	-	Bit	-	-
Contador	-	-	Autoincrement	Counter	-	-	-	-
Binários	-	-	Bynary	LongBinary	BLOB	Varbinary	BLOB	BLOB

TABELAS

Criando Tabelas

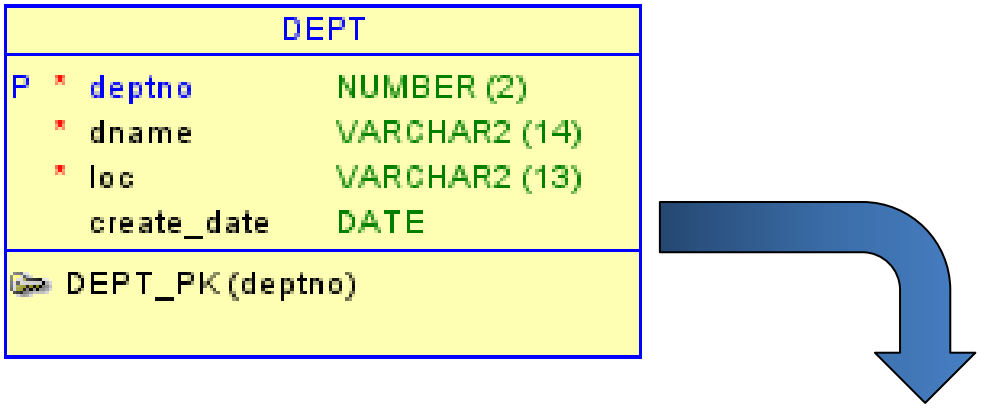
- **Comando SQL utilizado: CREATE TABLE**
- **A criação das tabelas faz parte do modelo físico do banco de dados.**
- **São criadas a partir da documentação do projeto conceitual do banco de dados – mais especificamente do modelo relacional**
- **Sintaxe para o comando CREATE TABLE**

```
CREATE TABLE <nome da tabela>
        (coluna1      <tipo de dado>,
         coluna2      <tipo de dado>,
         …
         );
```



Criando Tabelas

- **Modelo Entidade Relacionamento: Relação DEPT**



- **Criando a tabela DEPT**

```
CREATE TABLE dept
        (deptno      NUMBER(2),
         dname       VARCHAR2(14),
         loc         VARCHAR2(13),
         create_date DATE);
Table created.
```

Criando Tabelas

- **Definindo valores DEFAULT para colunas nas tabelas**

```
CREATE TABLE dept
        (deptno      NUMBER(2),
         dname       VARCHAR2(14),
         loc         VARCHAR2(13),
         create_date DATE DEFAULT SYSDATE);
Table created.
```

- **Descrevendo as colunas de uma tabela**

```
DESCRIBE dept
```

Name	Null?	Type
DEPTNO		NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)
CREATE_DATE		DATE

Alterando estruturas das Tabelas

- Comando SQL utilizado: ALTER TABLE

Use a instrução `ALTER TABLE` para:

- Adicionar uma nova coluna
- Modificar uma coluna existente
- Definir um valor default para a nova coluna
- Eliminar uma coluna

Alterando estruturas das Tabelas

1. Adicionando uma nova Coluna

```
ALTER TABLE DEPT ADD Total_sal number(10,2);
```

2. Modificando uma coluna existente [1]

```
ALTER TABLE DEPT Modify total_sal number(12,2)
```

3. Renomeando uma coluna

```
ALTER TABLE DEPT RENAME COLUMN total_sal TO Total_Salario;
```

4. Removendo uma coluna

```
ALTER TABLE DEPT DROP COLUMN total_Salario;
```

(1) Para modificar uma coluna existem regras. Consulte a documentação do SGBD

Removendo uma tabela

- **Comando SQL utilizado: DROP TABLE**

Use a instrução `DROP TABLE` para:

- **Remover uma tabela do banco de dados**
- **Não pode ser desfeito (1)**
- **Se existirem constraints de Foreign Key referenciando a tabela será necessário usar a cláusula "*cascade constraints*"**

(1) No caso do Oracle 10g ou superior, por default, ao remover uma tabela do banco de dados ela vai para a "lixeira" podendo ser restaurada depois. Para não usar a lixeira use a cláusula PURGE.

Removendo uma tabela

1. Removendo uma tabela

```
DROP TABLE EMPREGADOS;
```

2. Removendo uma tabela sem enviar para lixeira (oracle)

```
DROP TABLE EMPREGADOS PURGE;
```

3. Removendo uma tabela com referência de FK

```
DROP TABLE DEPT CASCADE CONSTRAINTS;
```

Restaurando uma tabela da lixeira (oracle)

```
FLASHBACK TABLE <NOME> TO BEFORE DROP;
```

SQL

2

CONSTRAINTS

CONSTRAINTS

CONSTRAINTS são RESTRIÇÕES criadas no banco de dados para:

- ✓ Impedir que dados inválidos sejam cadastrados no banco de dados
- ✓ Garantir a qualidade dos dados
- ✓ Garantir que campos obrigatórios sejam preenchidos
- ✓ Garantir integridade referencial definida no modelo relacional
- ✓ Em geral fazem parte da especificação do projeto conceitual do banco de dados e estão descritas no modelo relacional

A adição de constraints em uma tabela pode ser feita de duas maneiras:

1. Como parte do `CREATE TABLE` ao criar a tabela
2. Como parte do `ALTER TABLE` quando se adiciona as constraints depois de ter criado a tabela (é o mais comum)

Definindo Constraints

- **Sintaxe:**

```
CREATE TABLE [schema.]table
      (column datatype [DEFAULT expr]
      [column_constraint],
      ...
      [table_constraint][,...]);
```

- **Constraint no nível da coluna:**

```
column [CONSTRAINT constraint_name] constraint_type,
```

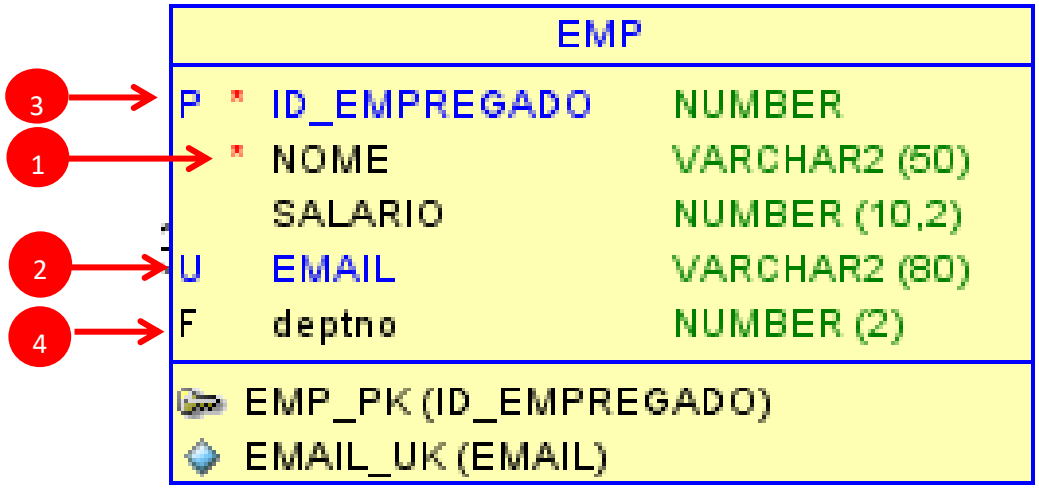
- **Constraint no nível da tabela:**

```
column,...
  [CONSTRAINT constraint_name] constraint_type
  (column, ...),
```

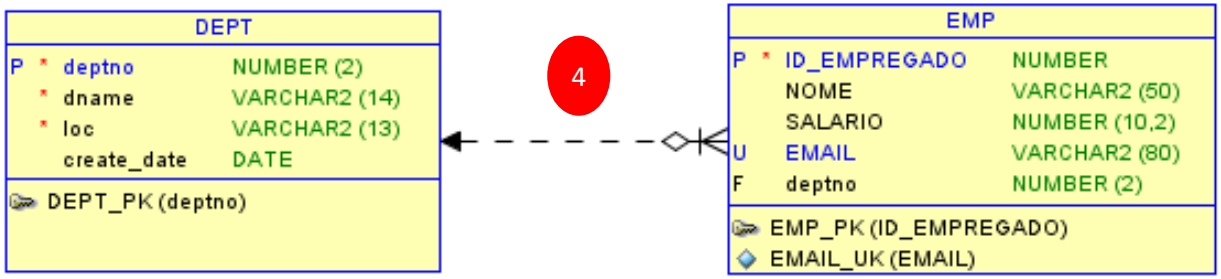
CONSTRAINTS

Tipos de Constraints

1. NOT NULL
2. UNIQUE
3. PRIMARY KEY
4. FOREIGN KEY
5. CHECK



- Algumas regras podem estar implícitas no Modelo ER.



Constraint NOT NULL

➔ **Garante que a coluna não aceite valores nulos:**

EMPLOYEE_ID	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID
100	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000	90
101	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000	90
102	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000	90
103	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000	60
104	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	6000	60
178	Grant	KGRANT	011.44.1644.429263	24-MAY-99	SA_REP	7000	
200	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	4400	10

. . .

20 rows selected.

Constraint `NOT NULL` (Nenhuma linha desta coluna pode conter um valor nulo.)

Constraint `NOT NULL`

Ausência da constraint `NOT NULL` (Qualquer linha desta coluna pode conter um valor nulo.)

Constraint NOT NULL

Ao criar a tabela

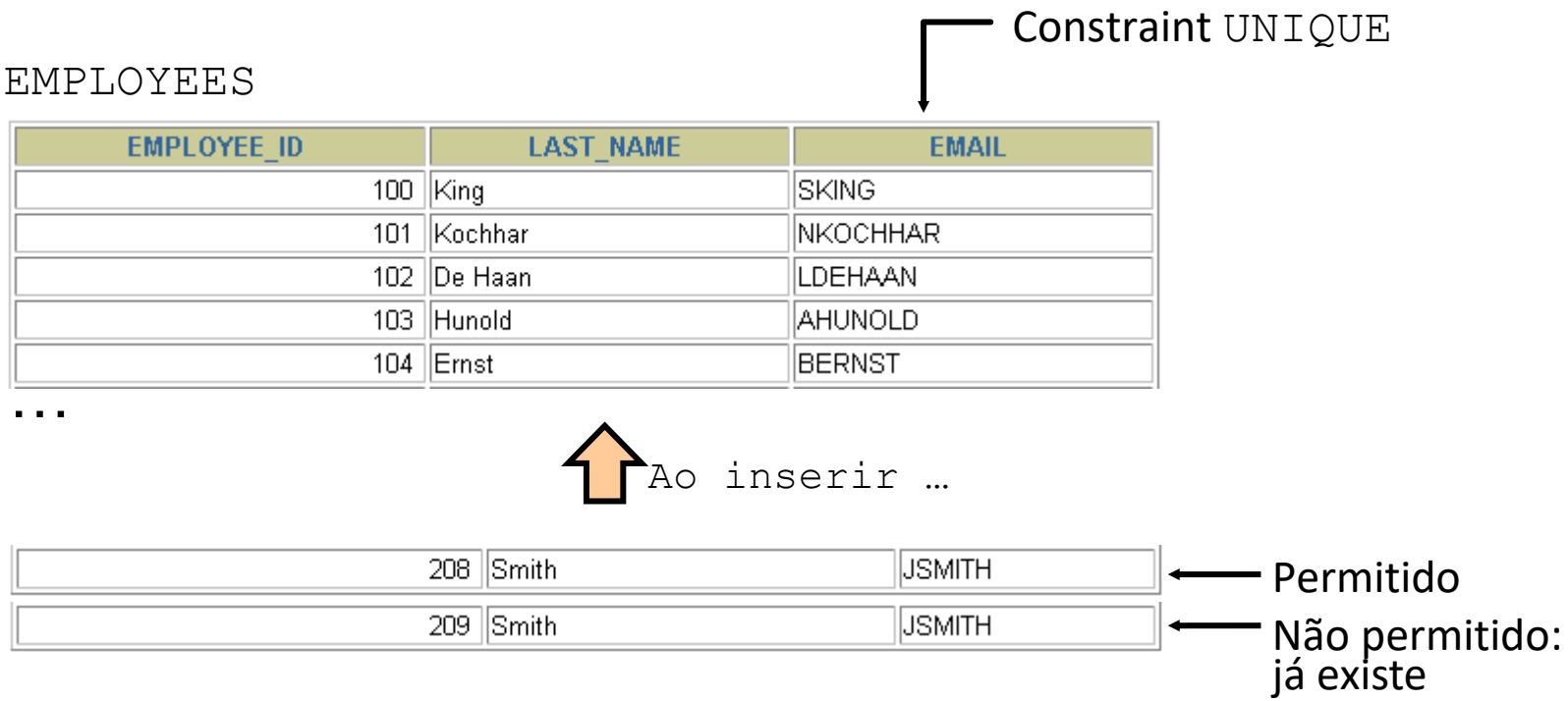
```
CREATE TABLE employees(
    employee_id      NUMBER(6),
    last_name        VARCHAR2(25) NOT NULL,
    email            VARCHAR2(25) NOT NULL,
    salary           NUMBER(8,2),
    commission_pct   NUMBER(2,2),
    hire_date        DATE NOT NULL,
...  );
```

Após ter criado a tabela

```
ALTER TABLE EMPLOYEES MODIFY SALARY NOT NULL;
```

Constraint UNIQUE

➔ **Garante que não existam valores não nulos repetidos**



Constraint UNIQUE

Ao criar a tabela

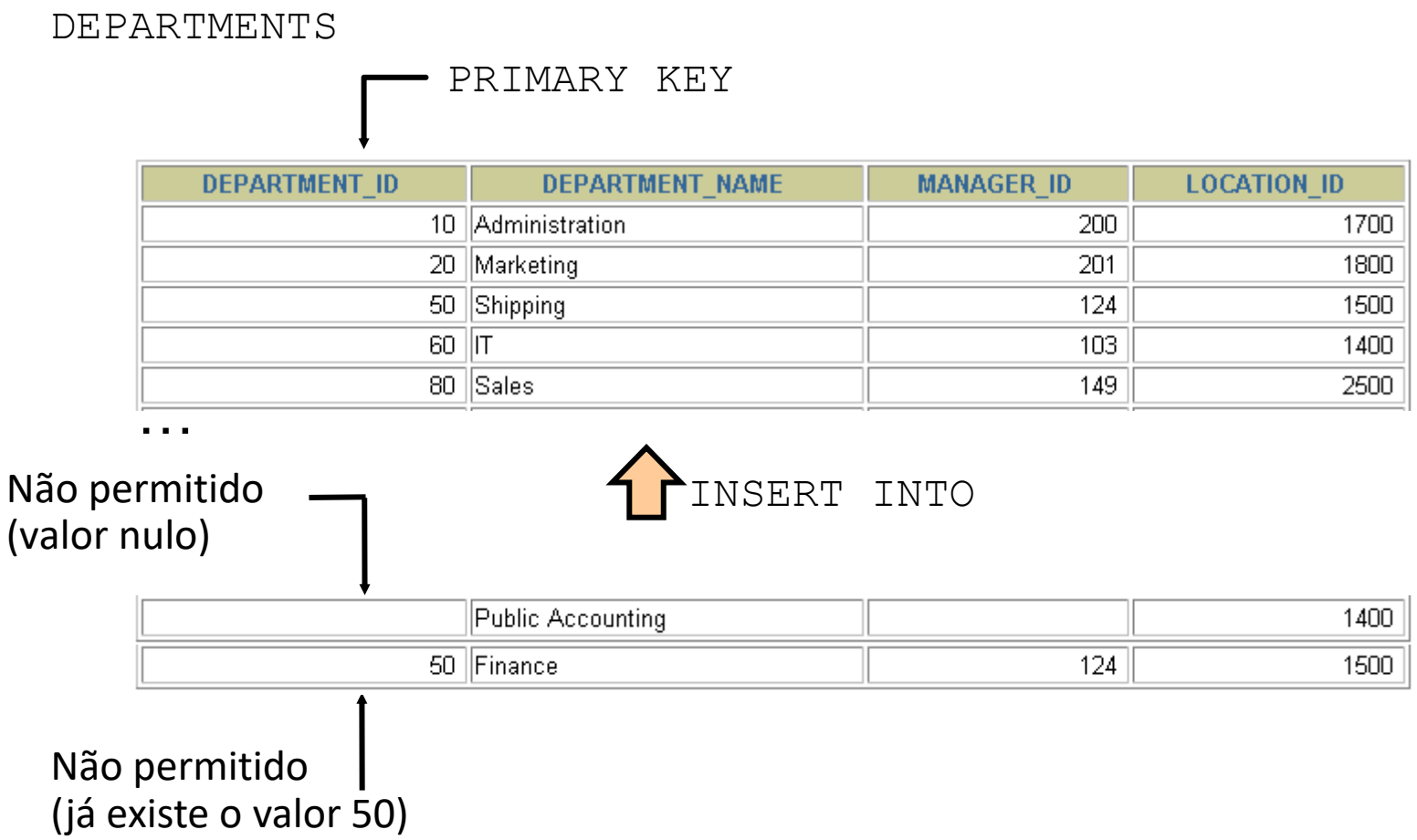
```
CREATE TABLE employees(
    employee_id       NUMBER(6),
    last_name         VARCHAR2(25) NOT NULL,
    email             VARCHAR2(25),
    salary            NUMBER(8,2),
    commission_pct    NUMBER(2,2),
    hire_date         DATE NOT NULL,
...
    CONSTRAINT emp_email_uk UNIQUE(email));
```

Após ter criado a tabela

```
ALTER TABLE EMPLOYEES ADD CONSTRAINT
emp_email_uk UNIQUE (email);
```

Constraint PRIMARY KEY

➔ **Identifica unicamente cada registro:** Não pode ter nulos nem valores repetidos



Constraint PRIMARY KEY

Ao criar a tabela

```
CREATE TABLE employees(
  employee_id  NUMBER(6) CONSTRAINT PRIMARY KEY,
  first_name   VARCHAR2(20),
  ...);
```

ou

```
CREATE TABLE employees(
  employee_id  NUMBER(6),
  first_name   VARCHAR2(20),
  ...
  CONSTRAINT emp_emp_id_pk PRIMARY KEY (EMPLOYEE_ID));
```

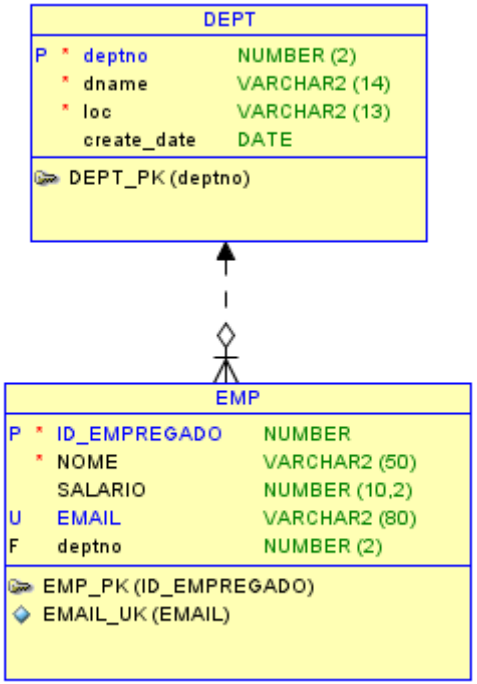
Após criar a tabela

```
ALTER TABLE EMPLOYEES ADD CONSTRAINT
emp_emp_id_pk PRIMARY KEY (EMPLOYEE_ID);
```

Constraint FOREIGN KEY

➔ **Estabelece o RELACIONAMENTO entre duas tabelas**

- É sempre criada no lado “MUITO” do relacionamento
- É criada na coluna que é chave estrangeira
- Deve sempre referenciar uma coluna em uma tabela que obrigatoriamente é UNIQUE ou PRIMARY KEY



Usada para garantir

INTEGRIDADE REFERENCIAL

Constraint FOREIGN KEY

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500

...

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
102	De Haan	90
103	Hunold	60
104	Ernst	60
107	Lorentz	60

...

FOREIGN KEY

INSERT INTO

200	Ford	9
201	Ford	60

Não permitido (o valor 9 não existe)

Permitido

Constraint FOREIGN KEY

```
CREATE TABLE employees(
    employee_id      NUMBER(6),
    last_name        VARCHAR2(25) NOT NULL,
    email            VARCHAR2(25),
     ...
    department_id    NUMBER(4),
    CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)
    REFERENCES departments(department_id),
    CONSTRAINT emp_email_uk UNIQUE(email));
```

```
ALTER TABLE EMPLOYEES
ADD CONSTRAINT emp_dept_fk FOREIGN KEY (DEPARTMENT_ID)
REFERENCES departments (DEPARTMENT_ID);
```

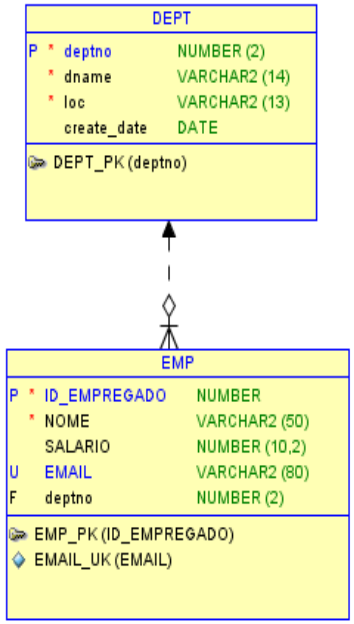
Constraint FOREIGN KEY

INTEGRIDADE REFERENCIAL

Um banco de dados impõe restrições referenciais para garantir a integridade dos dados quando as linhas de uma tabela são alteradas ou excluídas.
Se as linhas dependentes nas tabelas referenciadas ainda existem, essas referências devem ser consideradas.

1) Não da para apagar um registro pai se existem filhos referenciando ele => Isto geraria um problema de INTEGRIDADE dos dados
2) Se o modelo permitir e o negócio aceitar, você pode remover o registro pai e o SGBDR irá automaticamente remover TODOS os registros filhos ou modificar a coluna FK com valor NULL. Nestes casos, a constraint FK deve obrigatoriamente ter a cláusula:

- **ON DELETE CASCADE** => Remove os registros filhos
- **ON DELETE SET NULL** => Muda o valor da coluna FK para NULL (desde que ela não tenha uma constraint NOT NULL)



Verifique em cada SGBDR a implementação das integridades referenciais

Constraint FOREIGN KEY

- **Com deleção em cascata**

```
ALTER TABLE EMPLOYEES
ADD CONSTRAINT emp_dept_fk FOREIGN KEY (DEPARTMENT_ID)
REFERENCES departments (DEPARTMENT_ID) ON DELETE CASCADE;
```

- **Modificando a coluna FK da tabela filha para NULL**

```
ALTER TABLE EMPLOYEES
ADD CONSTRAINT emp_dept_fk FOREIGN KEY (DEPARTMENT_ID)
REFERENCES departments (DEPARTMENT_ID) ON DELETE SET NULL;
```

Constraint CHECK

- Define uma condição que cada linha deve atender
- As seguintes expressões não são permitidas:
  - Referências às pseudocolunas **CURRVAL**, **NEXTVAL**, **LEVEL** e **ROWNUM**
  - Chamadas de functions
  - Consultas que fazem referência a outros valores em outras linhas

```
..., salary   NUMBER(2)
     CONSTRAINT emp_salary_min CHECK (salary > 0),...
```

CREATE TABLE: Exemplo

```
CREATE TABLE employees
    ( employee_id    NUMBER(6)
        CONSTRAINT     emp_employee_id   PRIMARY KEY
    , first_name     VARCHAR2(20)
    , last_name      VARCHAR2(25)
        CONSTRAINT     emp_last_name_nn  NOT NULL
    , email          VARCHAR2(25)
        CONSTRAINT     emp_email_nn      NOT NULL
        CONSTRAINT     emp_email_uk      UNIQUE
    , phone_number   VARCHAR2(20)
    , hire_date      DATE
        CONSTRAINT     emp_hire_date_nn  NOT NULL
    , job_id         VARCHAR2(10)
        CONSTRAINT     emp_job_nn        NOT NULL
    , salary         NUMBER(8,2)
        CONSTRAINT     emp_salary_ck     CHECK (salary>0)
    , commission_pct NUMBER(2,2)
    , manager_id     NUMBER(6)
    , department_id  NUMBER(4)
        CONSTRAINT     emp_dept_fk       REFERENCES
           departments (department_id));
```

Violando Constraints

Uma vez criada as constraint de FK, a integridade referencial será automaticamente verificada pelo SGBD

```
UPDATE employees
SET department_id = 55
WHERE       department_id = 110;
```

```
UPDATE employees
       *
ERROR at line 1:
ORA-02291: integrity constraint (HR.EMP_DEPT_FK)
violated - parent key not found
```

O departamento 55 não existe.

Violando Constraints

Outro exemplo:

Não é possível deletar uma linha que contém uma chave primária usada como chave estrangeira em outra tabela.

```
DELETE FROM departments
WHERE       department_id = 60;
```

```
DELETE FROM departments
            *
ERROR at line 1:
ORA-02292: integrity constraint (HR.EMP_DEPT_FK)
violated - child record found
```

Esvaziando uma tabela

Comando TRUNCATE

- Remove todas as linhas de uma tabela, esvaziando a tabela e mantendo a estrutura intacta.
- É muito mais rápido que remover (usando delete) todas as linhas da tabela.
- É uma instrução DDL (Data Definition Language), e não DML; não pode ser desfeita facilmente
- Sintaxe:

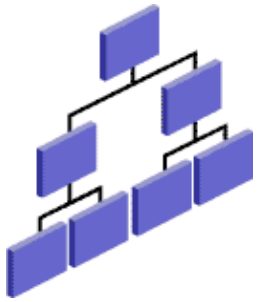
```
TRUNCATE TABLE table_name;
```

- Exemplo:

```
TRUNCATE TABLE copy_emp;
```

3 ÍNDICES

Índices

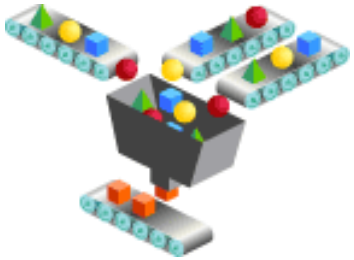


Um índice:

- **É um objeto do esquema (usuário)**
- **É usado pelo banco de dados para acelerar a recuperação de linhas.**
- **Pode reduzir a entrada/saída de disco por um método de acesso de caminho rápido para localizar dados rapidamente**
- **É independente da tabela que indexa**
- **É usado e mantido automaticamente pelo banco de dados**

Como Criar Índices?

- **Automaticamente: Um índice exclusivo é criado automaticamente quando você define uma constraint de chave `PRIMARY` ou `UNIQUE` em uma definição de tabela.**



- **Manualmente: Os usuários podem criar índices não exclusivos em colunas para acelerar o acesso às linhas.**

Criando um Índice

- Crie um índice em uma ou mais colunas:

```
CREATE INDEX index ON table (column[, column]...);
```

Exemplo:

Aumente a velocidade de acesso da consulta à coluna `LAST_NAME` *da tabela* `EMPLOYEES`

```
CREATE INDEX  emp_last_name_idx ON employees(last_name);
```

- Índices compostos são índices criados com mais de uma coluna.

```
CREATE INDEX emp_name_idx ON employees(last_name,first_name);
```

- Índices únicos não indexam valores repetidos. São similares a uma constraing UNIQUE

```
CREATE UNIQUE INDEX emp_email_idx ON employees(email);
```

Removendo um Índice

- **Para remover um índice do dicionário de dados, use o comando `DROP INDEX`:**

```
DROP INDEX index;
```

- **Remova o índice `UPPER_LAST_NAME_IDX` do dicionário de dados:**

```
DROP INDEX emp_last_name_idx;
Index dropped.
```

- **Para eliminar um índice, você precisa ser o proprietário dele ou ter o privilégio `DROP ANY INDEX`.**

Outros Objetos de Banco de Dados

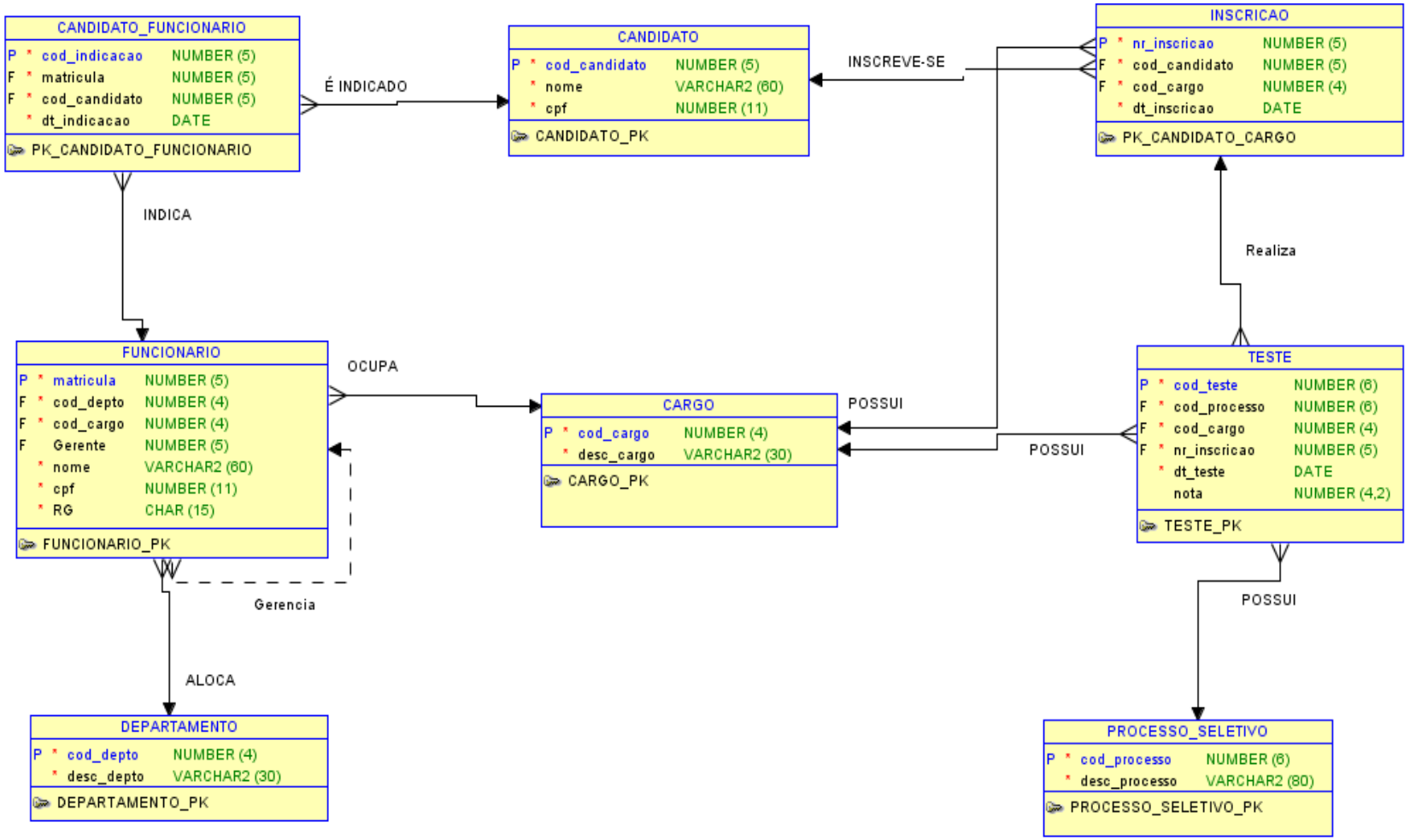
- **Sinônimos**
- **“Sequences” em banco de dados Oracle**

Pesquisem a respeito destes outros objetos.
Pesquisem tópicos avançados sobre os objetos vistos neste capítulo.

Dúvidas

Exercício opcional para praticarem

Use os comandos DDL a seguir para criar o projeto físico do seguinte modelo relacional



Exercício opcional para praticarem

- ✓ Vejam a seguir como os comandos DDL são usados para criar as tabelas e constraints de acordo com o modelo relacional
- ✓ Olhem o modelo e vejam como os comandos SQL foram criados
- ✓ **Dica**: Criem um script!

Se você quer criar um script que recria as tabelas "do zero", como uma dica de ouro é adicionar no início do script o comando para remover todas as tabelas e em seguida começa a cria-las novamente

DROP TABLE <nome> CASCADE CONSTRAINTS;

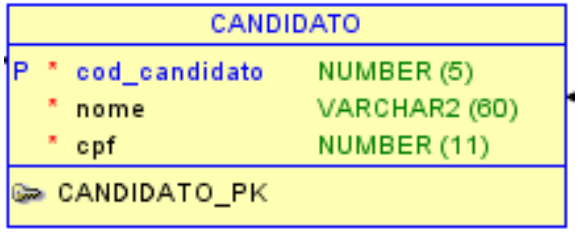
No final da apostila tem os comandos para este exercício

Solução

Transformação do Modelo para SQL – Criação de Tabelas

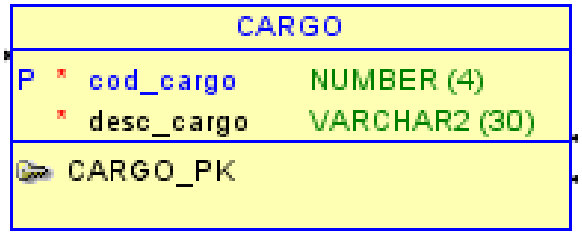
Exemplo

```
/*******  CRIAÇÃO DAS TABELAS E CHAVE PRIMÁRIA ********/
/* TABELA CANDIDATO */
CREATE TABLE CANDIDATO
    (
     cod_candidato  NUMBER (5)     NOT NULL ,
     nome           VARCHAR2 (60)  NOT NULL ,
     cpf            NUMBER (11)    NOT NULL
    )
;
/* Chave primária da tabela Candidato */
ALTER TABLE CANDIDATO
    ADD CONSTRAINT CANDIDATO_PK PRIMARY KEY ( cod_candidato ) ;
```



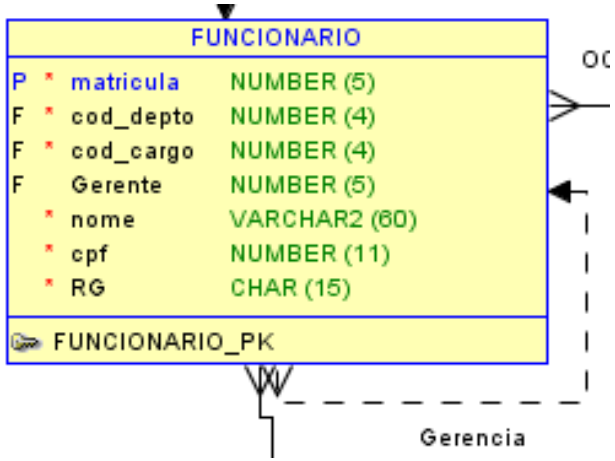
Solução

```
/* TABELA CARGO */
CREATE TABLE CARGO
    (
     cod_cargo  NUMBER (4)     NOT NULL ,
     desc_cargo VARCHAR2 (30) NOT NULL
    )
;
/* Chave primária da tabela Cargo */
ALTER TABLE CARGO
    ADD CONSTRAINT CARGO_PK PRIMARY KEY ( cod_cargo ) ;
```



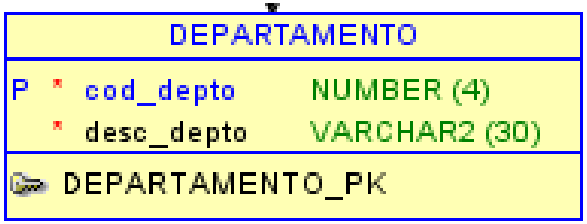
Solução

```
/* TABELA FUNCIONARIO */
CREATE TABLE FUNCIONARIO
    (
     matricula   NUMBER (5)     NOT NULL ,
     cod_depto   NUMBER (4)     NOT NULL ,
     cod_cargo   NUMBER (4)     NOT NULL ,
     Gerente     NUMBER (5)          NULL ,
     nome        VARCHAR2 (60)  NOT NULL ,
     cpf         NUMBER (11)    NOT NULL ,
     RG          CHAR (15)      NOT NULL
    )
;
/* Chave primária da tabela Funcionario */
ALTER TABLE FUNCIONARIO
    ADD CONSTRAINT FUNCIONARIO_PK PRIMARY KEY ( matricula ) ;
```



Solução

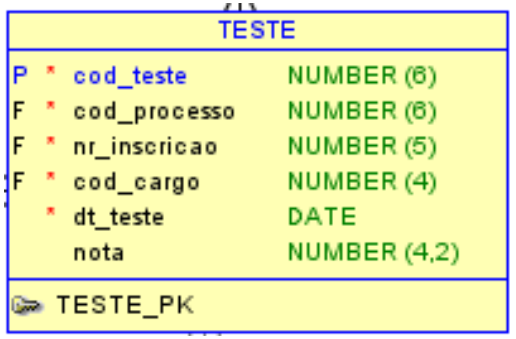
```
/* TABELA DEPARTAMENTO */
CREATE TABLE DEPARTAMENTO
    (
     cod_depto  NUMBER (4)     NOT NULL ,
     desc_depto VARCHAR2 (30) NOT NULL
    )
;
/* Chave primária da tabela Departamento */
ALTER TABLE DEPARTAMENTO
    ADD CONSTRAINT DEPARTAMENTO_PK PRIMARY KEY ( cod_depto ) ;
```



54

Solução

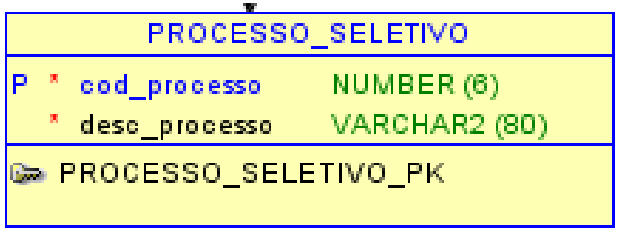
```
/* TABELA TESTE */
CREATE TABLE TESTE
    (
     cod_teste NUMBER (6)  NOT NULL ,
     cod_processo NUMBER (6)  NOT NULL ,
     nr_inscricao NUMBER (5)  NOT NULL ,
     cod_cargo NUMBER (4)  NOT NULL ,
     dt_teste DATE  NOT NULL ,
     nota NUMBER (4,2)
    )
;
/* Chave primária da tabela Teste */
ALTER TABLE TESTE
    ADD CONSTRAINT TESTE_PK PRIMARY KEY ( cod_teste ) ;
```



55

Solução

```
/* TABELA PROCESSO_SELETIVO */
CREATE TABLE PROCESSO_SELETIVO
    (
     cod_processo   NUMBER (6)      NOT NULL ,
     desc_processo  VARCHAR2 (80)   NOT NULL
    )
;
/* Chave primária da tabela Processo_Seletivo */
ALTER TABLE PROCESSO_SELETIVO
    ADD CONSTRAINT PROCESSO_SELETIVO_PK PRIMARY KEY ( cod_processo )
  ;
```



56

Solução

```
/* TABELA INSCRICAO */
CREATE TABLE INSCRICAO
    (
     nr_inscricao   NUMBER (5)   NOT NULL ,
     cod_candidato  NUMBER (5)   NOT NULL ,
     cod_cargo      NUMBER (4)   NOT NULL ,
     dt_inscricao   DATE         NOT NULL
    )
;
/* Chave primária da tabela Inscricao */
ALTER TABLE INSCRICAO
    ADD CONSTRAINT PK_CANDIDATO_CARGO PRIMARY KEY ( nr_inscricao ) ;
```

57

Solução

```
/* TABELA CANDIDATO_FUNCIONARIO */
CREATE TABLE CANDIDATO_FUNCIONARIO
    (
     cod_indicacao  NUMBER (5)   NOT NULL ,
     matricula      NUMBER (5)   NOT NULL ,
     cod_candidato  NUMBER (5)   NOT NULL ,
     dt_indicacao   DATE         NOT NULL
    )
;
/* Chave primária da tabela Candidato_Funcionario */
ALTER TABLE CANDIDATO_FUNCIONARIO
    ADD CONSTRAINT PK_CANDIDATO_FUNCIONARIO PRIMARY KEY (
  cod_indicacao ) ;
```

58

Solução

```
/******* Chave ESTRANGEIRA - Tabela Funcionario ********/
ALTER TABLE FUNCIONARIO
    ADD ( FOREIGN KEY  (cod_depto)
          REFERENCES DEPARTAMENTO (cod_depto)
        );
/******* Chave ESTRANGEIRA - Tabela Funcionario ********/
ALTER TABLE FUNCIONARIO
    ADD ( FOREIGN KEY (Gerente)
          REFERENCES FUNCIONARIO (matricula)
          ON DELETE SET NULL
        );
```


59

Criação das Chaves Estrangeiras

```
/******* Chave ESTRANGEIRA - Tabela Funcionario ********/
ALTER TABLE FUNCIONARIO
    ADD ( FOREIGN KEY (cod_cargo)
          REFERENCES CARGO (cod_cargo)
        );
/******* Chave ESTRANGEIRA - Tabela Candidato_Funcionario
  ********/
ALTER TABLE CANDIDATO_FUNCIONARIO
    ADD ( FOREIGN KEY (matricula)
          REFERENCES FUNCIONARIO (matricula)
        );
```


60

Criação das Chaves Estrangeiras

```
/******* Chave ESTRANGEIRA - Tabela Candidato_Funcionario
  ********/
ALTER TABLE CANDIDATO_FUNCIONARIO
    ADD ( FOREIGN KEY (cod_candidato)
          REFERENCES CANDIDATO (cod_candidato)
          ON DELETE CASCADE
        );
/******* Chave ESTRANGEIRA - Tabela Inscricao ********/
ALTER TABLE INSCRICAO
    ADD ( FOREIGN KEY (cod_candidato)
          REFERENCES CANDIDATO (cod_candidato)
          ON DELETE CASCADE
         );
```


61

Criação das Chaves Estrangeiras

```
/******* Chave ESTRANGEIRA - Tabela Inscricao ********/
ALTER TABLE INSCRICAO
    ADD ( FOREIGN KEY (cod_cargo)
          REFERENCES CARGO  (cod_cargo)
        );
/******* Chave ESTRANGEIRA - Tabela Teste ********/
ALTER TABLE TESTE
    ADD ( FOREIGN KEY (cod_processo)
          REFERENCES PROCESSO_SELETIVO (cod_processo)
        );
```


62

Criação das Chaves Estrangeiras

```
/******* Chave ESTRANGEIRA - Tabela Teste ********/
ALTER TABLE TESTE
    ADD ( FOREIGN KEY (nr_inscricao)
          REFERENCES INSCRICAO    (nr_inscricao)
        );
/* Chave ESTRANGEIRA - Tabela Teste */
ALTER TABLE TESTE
    ADD ( FOREIGN KEY  (cod_cargo)
          REFERENCES CARGO   (cod_cargo)
        );
```


63

CRIANDO INDEXES ...

1. Criar um índice para a coluna nome data tabela funcionário

```
CREATE INDEX IX_NOME_FUNC ON FUNCIONARIO(NOME);
```

2. Criar um índice para a coluna NOTA e DT_TESTE da tabela pessoa TESTE

```
CREATE INDEX IX_DT_NOTA_TESTE ON TESTE(NOTA,DT_TESTE);
```

3. Criar um índice único para a coluna CPF da tabela CANDIDADTO

```
CREATE UNIQUE INDEX IX_CPF_CANDIDATO ON CANDIDATO(CPF);
```


64

FIAP

PARA REMOVER AS TABELAS ....

65

Removendo as tabelas

Comando DROP TABLE – Exemplos do modelo Recrutamento e Seleção

```
/*  EXCLUSÃO DE TABELAS E CONSTRAINTS */
DROP TABLE CANDIDATO CASCADE CONSTRAINTS ;
DROP TABLE CANDIDATO_FUNCIONARIO CASCADE CONSTRAINTS ;
DROP TABLE CARGO CASCADE CONSTRAINTS ;
DROP TABLE DEPARTAMENTO CASCADE CONSTRAINTS ;
DROP TABLE FUNCIONARIO CASCADE CONSTRAINTS ;
DROP TABLE INSCRICAO CASCADE CONSTRAINTS ;
DROP TABLE PROCESSO_SELETIVO CASCADE CONSTRAINTS ;
DROP TABLE TESTE CASCADE CONSTRAINTS ;
```


66

REFERÊNCIAS

- PUGA, Sandra; FRANÇA Edson; GOYA Milton. Banco de Dados – Implementação em SQL PL/SQL e Oracle 11g. São Paulo: Pearson, 2014
- NAVATHE, Shamkant B, ELMASRI, Ramez E. **Sistemas de banco de dados**. São Paulo: Pearson, 2005.

- Oracle SQL References: http://docs.oracle.com
- Manuais Oracle – Introdução ao Oracle e SQL I e II
- SQL Developer http://www.oracle.com/technetwork/developer-tools/sql-developer/overview/index-097090.html

FIAP