DCC011:
Introdução a Banco de Dados

Rodrygo Santos
rodrygo@dcc.ufmg.br

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais

Programa

- Introdução
  - Conceitos básicos, características da abordagem de banco de dados, modelos de dados, esquemas e instâncias, arquitetura de um sistema de banco de dados, componentes de um sistema de gerência de banco de dados.
- Modelos de dados e linguagens
  - Modelo entidade-relacionamento (ER), modelo relacional, álgebra relacional, **SQL**.
- Projeto de bancos de dados
  - Fases do projeto de bancos de dados, projeto lógico de bancos de dados relacionais, normalização.
- Novas Tecnologias e Aplicações de Banco de Dados

SQL

1. Definição de Dados
2. Definição de Restrições de Integridade
3. Atualizações
4. Visões

1. Definição de Dados em SQL

- Comando CREATE SCHEMA

  CREATE SCHEMA schemaName
  AUTHORIZATION userName;

- Esquema tem nome e um usuário administrador

  CREATE SCHEMA EMPRESA
  AUTHORIZATION JohnSmith;

- DBA: cria esquemas, tabelas, e demais partes do banco de dados

1. Definição de Dados em SQL

- Comando CREATE TABLE

  CREATE TABLE <nome da tabela>
  (<definições de colunas>
  <definição da chave primária>
  <definições de chaves alternativas>
  <definições de chaves estrangeiras>);

  - Case sensitive em geral (depende de implementação)

Tipos de Dados

- Numérico (principais)
  - INTEGER/INT, FLOAT, REAL
- String de caractere
  - CHAR(n), VARCHAR(n), CLOB
- String de bits: BIT(n), BIT VARYING(n)
- Booleano: BOOLEAN, TRUE/FALSE/UNKNOWN
- Data: DATE, YEAR, MONTH, DAY
- Tempo: TIME, HOUR, MINUTE, SECOND, TIMESTAMP

TRUE/FALSE/UNKNOWN

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	--	--
FALSE	FALSE	FALSE	--
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	--	--
FALSE	TRUE	FALSE	--
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

Restrições de atributos e domínios

- NOT NULL
  - Implícita para chave primária
- DEFAULT <valor>
- CHECK <condição>
  - Dnumber INT **NOT NULL**
    CHECK (Dnumber > 0 **AND** Dnumber < 21);
  - **CREATE DOMAIN** D_NUM **AS** INTEGER
    CHECK (D_NUM > 0 **AND** D_NUM < 21);

```
CREATE TABLE EMPLOYEE
    (FNAME      VARCHAR(15)    NOT NULL,
    MINIT       CHAR,
    LNAME       VARCHAR(15)    NOT NULL,
    SSN         CHAR(9)        NOT NULL,
      …
    SUPERSSN    CHAR(9),
    DNO         INT            NOT NULL,
    PRIMARY KEY (SSN),
    FOREIGN KEY (SUPERSSN)
                REFERENCES EMPLOYEE (SSN)
    ON DELETE SET NULL,
    FOREIGN KEY (DNO)
                REFERENCES DEPARTMENT (DNUMBER));
```

```
CREATE TABLE EMPLOYEE
        ( FNAME          VARCHAR(15)      NOT NULL ,
          MINIT          CHAR ,
          LNAME          VARCHAR(15)      NOT NULL ,
          SSN            CHAR(9)          NOT NULL ,
          BDATE          DATE
          ADDRESS        VARCHAR(30) ,
          SEX            CHAR ,
          SALARY         DECIMAL(10,2) ,
          SUPERSSN       CHAR(9) ,
          DNO            INT              NOT NULL ,
        PRIMARY KEY (SSN) ,
        FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE(SSN) ,
        FOREIGN KEY (DNO) REFERENCES DEPARTMENT(DNUMBER) ) ;
```

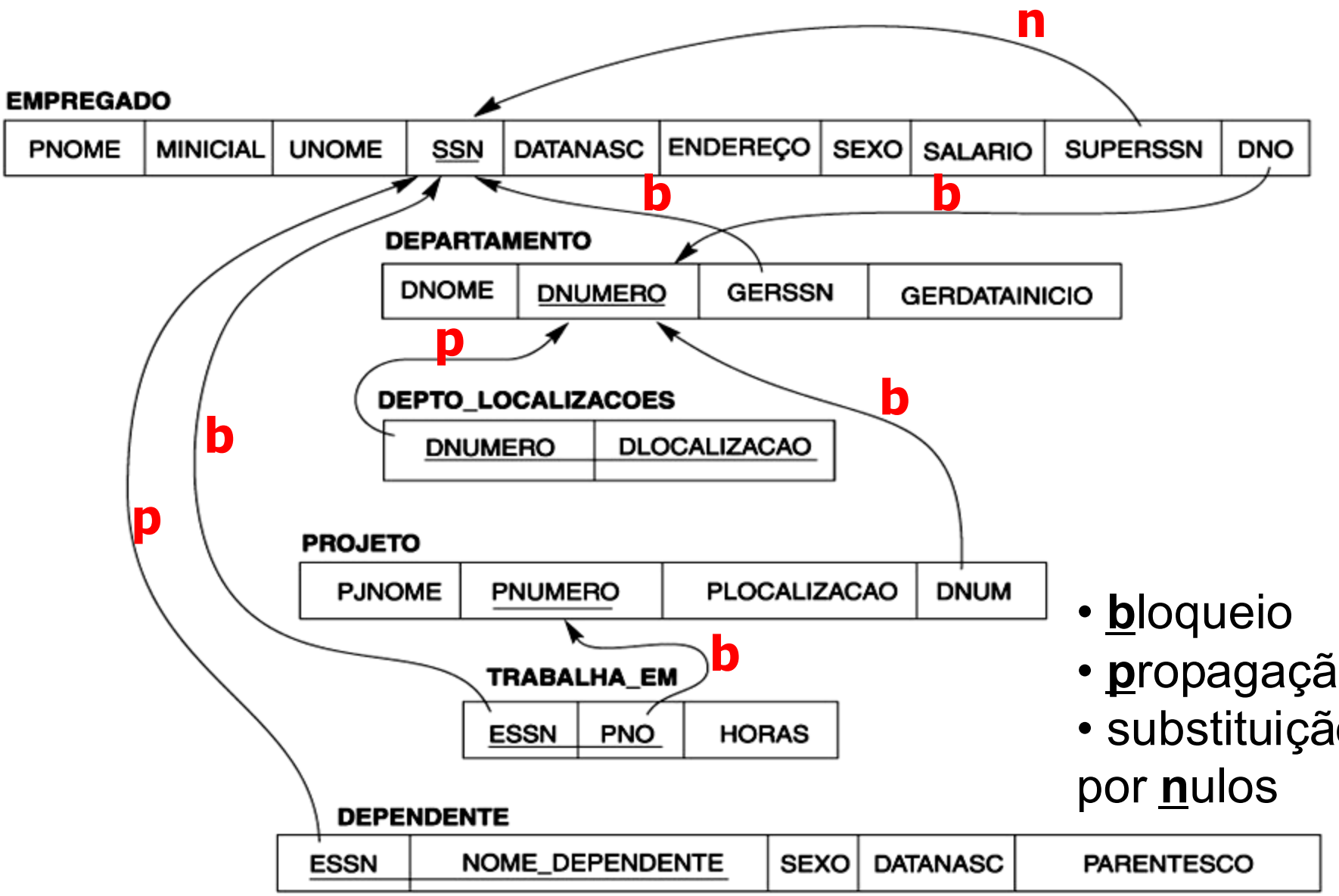
```
CREATE TABLE DEPARTMENT
        ( DNAME          VARCHAR(15)      NOT NULL ,
          DNUMBER        INT              NOT NULL ,
          MGRSSN         CHAR(9)          NOT NULL ,
          MGRSTARTDATE   DATE ,
        PRIMARY KEY (DNUMBER) ,
        UNIQUE (DNAME) ,
        FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE(SSN) ) ;
```

```
CREATE TABLE PROJECT
        ( PNAME          VARCHAR(15)      NOT NULL ,
          PNUMBER        INT              NOT NULL ,
          PLOCATION      VARCHAR(15) ,
          DNUM           INT              NOT NULL ,
        PRIMARY KEY (PNUMBER) ,
        UNIQUE (PNAME)
        FOREIGN KEY (DNUM) REFERENCES DEPARTMENT(DNUMBER) ) ;
```

```
CREATE TABLE WORKS_ON
        ( ESSN           CHAR(9)          NOT NULL ,
          PNO            INT              NOT NULL ,
          HOURS          DECIMAL(3,1)     NOT NULL ,
        PRIMARY KEY (ESSN, PNO) ,
        FOREIGN KEY (ESSN) REFERENCES EMPLOYEE(SSN) ,
        FOREIGN KEY (PNO) REFERENCES PROJECT(PNUMBER) ) ;
```

REVISÃO

Restrições de integridade referencial com opções de remoção



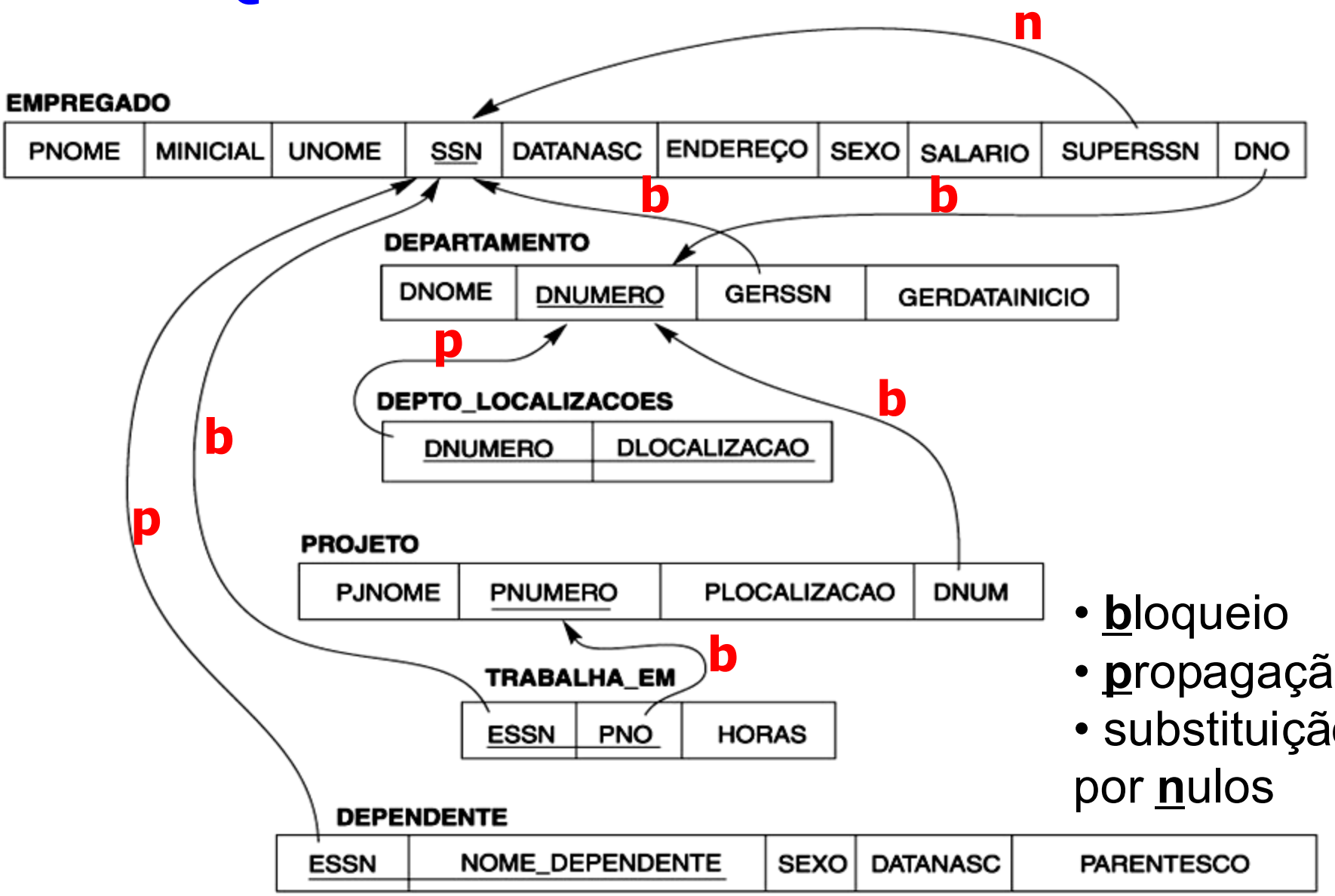
- **b**loqueio
- **p**ropagação
- substituição por **n**ulos

2. Restrição Integridade Referencial

- Opções de remoção (cláusula ON DELETE):
  - CASCADE (propagação)
  - SET NULL (substituição por nulos)
  - SET DEFAULT (substituição por um valor default)
  - RESTRICT: (bloqueia a operação – opção default)
- As mesmas opções se aplicam à cláusula ON UPDATE

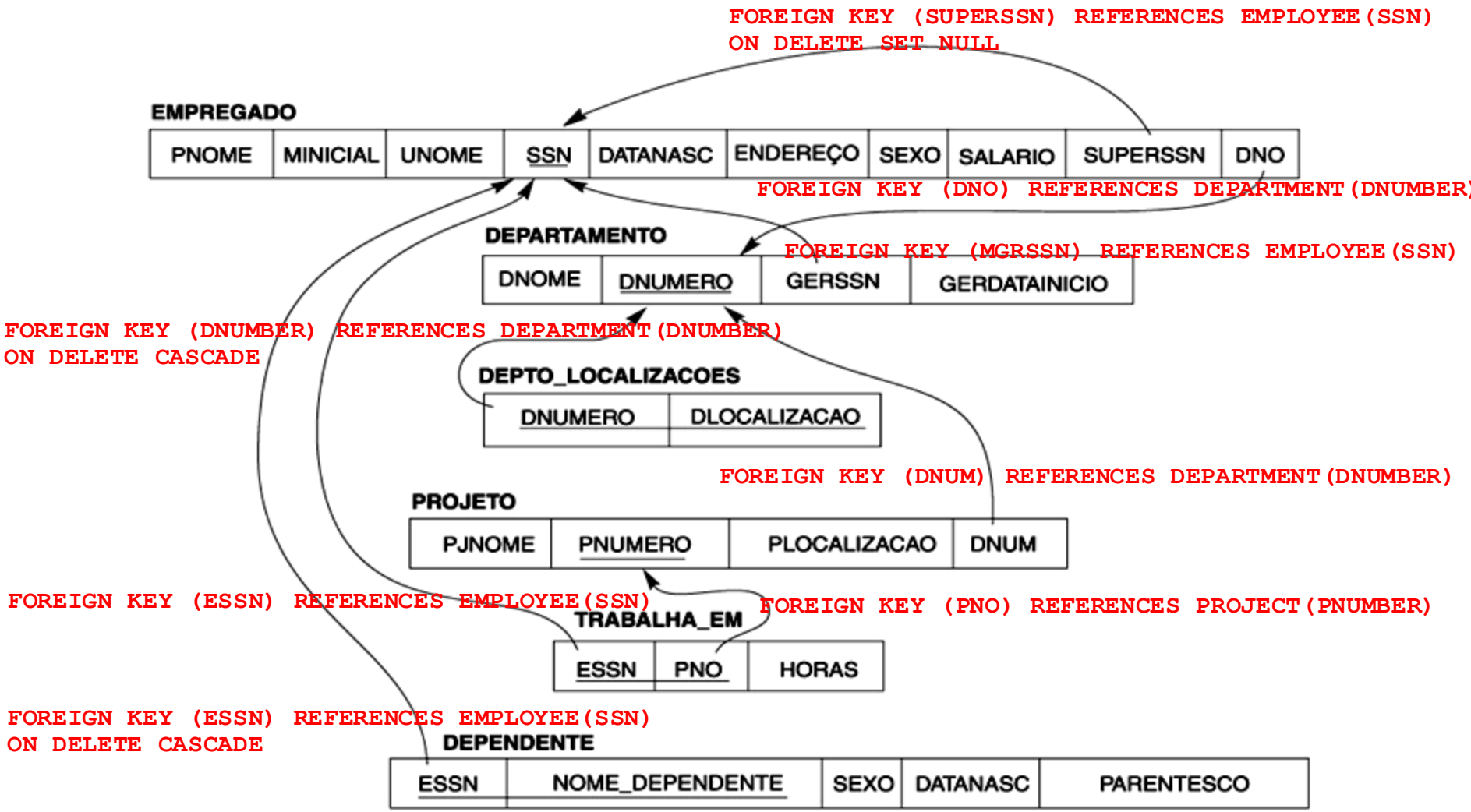
  Entretanto esta cláusula NÃO deve ser utilizada para se evitar que as chaves primárias sejam modificadas

Em SQL????



- **b**loqueio
- **p**ropagação
- substituição por **n**ulos

Restrição Integridade Referencial



```
CREATE TABLE EMPREGADO
        ( . . . ,
          DNO          INT      NOT NULL      DEFAULT 1,
        CONSTRAINT EMPPK
          PRIMARY KEY (SSN) ,
        CONSTRAINT EMPSUPERFK
          FOREIGN KEY (SUPERSSN) REFERENCES EMPREGADO(SSN)
                       ON DELETE SET NULL ON UPDATE CASCADE ,
        CONSTRAINT EMPDEPTFK
          FOREIGN KEY (DNO) REFERENCES DEPARTAMENTO(DNUMERO)
                       ON DELETE SET DEFAULT ON UPDATE CASCADE );

CREATE TABLE DEPARTAMENTO
        ( . . . ,
          GERSSN CHAR(9) NOT NULL DEFAULT '888665555' ,
          . . . ,
          CONSTRAINT DEPTPK
            PRIMARY KEY (DNUMERO) ,
          CONSTRAINT DEPTSK
            UNIQUE (DNOME),
          CONSTRAINT DEPTMGRFK
            FOREIGN KEY (GERSSN) REFERENCES EMPREGADO(SSN)
                 ON DELETE SET DEFAULT ON UPDATE CASCADE );

CREATE TABLE DEP_LOCALIZACOES
        ( . . . ,
          PRIMARY KEY (DNUMERO, DLOCALIZACAO),
          FOREIGN KEY (DNUMERO) REFERENCES DEPARTAMENTO(DNUMERO)
          ON DELETE CASCADE ON UPDATE CASCADE ) ;
```

Figura 8.2 Exemplo ilustrando como os valores do atributo *default* e as ações referenciais engatilhadas são especificados em SQL.

Definição de Dados em SQL

- Comandos DROP SCHEMA e DROP TABLE
  - **DROP SCHEMA** COMPANY [**CASCADE | RESTRICT**];
    - **RESTRICT:** apenas se não tem elementos
  - **DROP TABLE** DEPENDENT [**CASCADE | RESTRICT**];
    - **RESTRICT:** se a tabela não é referenciada em qualquer restrição ou visão
- Comando ALTER TABLE
  - **ALTER TABLE** COMPANY.EMPLOYEE
    ADD JOB VARCHAR(12);
      - Inicialmente Null para todas as tuplas
  - **ALTER TABLE** COMPANY.EMPLOYEE
    DROP ADDRESS [**CASCADE | RESTRICT**];
    - **RESTRICT:** se nenhuma visão ou restrição referencia a coluna

3. SQL: Atualizações

1. Insert
2. Delete
3. Update

1. INSERT

- INSERT: adiciona uma única linha a uma relação
- Nome da relação e os valores para os atributos
- Atributos na MESMA ORDEM do esquema (CREATE TABLE)

INSERT INTO EMPLOYEE

VALUES ('Richard' , 'K' , 'Marini' , '653258653' , '1962-12-30' , '98 Oak Forest, Katy, TX' ,37000,' 987654321' ,4);

INSERT

- Outra opção: especifica o nome dos atributos
  - Adicionar alguns valores
  - Valores NOT NULL, e sem DEFAULT

```
INSERT INTO
        EMPLOYEE(FNAME, LNAME, SSN, DNO)
VALUES ( 'Richard' , 'Marini' , '653258653' , 4);
```

- Adiciona o resultado de uma consulta

```
INSERT INTO
        EMPLOYEE(FNAME, LNAME, SSN, DNO)
SELECT * FROM INPUT;
```

2. DELETE

- Remove tuplas de uma relação + propagação
  - WHERE: seleciona tuplas a serem removidas

DELETE FROM EMPLOYEE **WHERE** LNAME= 'Brown' ;

DELETE FROM EMPLOYEE
WHERE DNO **IN**
(**SELECT** DNUMBER
FROM DEPARTMENT
WHERE DNAME= 'Research');

- Remover todas as tuplas da tabela (mas mantém o esquema):

DELETE FROM EMPLOYEE;

3. UPDATE

- Modifica valores de atributos em tuplas de UMA relação
  - WHERE: seleciona tuplas a serem modificadas
  - Alterar chave primária **(prática ruim)** pode propagar valores em chaves estrangeiras em outras relações
- **UPDATE** PROJECT

  SET PLOCATION= 'Bellaire' , DNUM=5

  WHERE PNUMBER=10;
- **UPDATE** EMPLOYEE

  SET SALARY=SALARY*1.1

  WHERE DNO **IN** (**SELECT** DNUMBER

  FROM DEPARTMENT

  WHERE DNAME= 'Research');

Revisão: INSERT/UPDATE/DELETE

- **INSERT INTO** EMPLOYEE
  VALUES (‘Richard’ , ‘K’ , ‘Marini’ , ‘653258653’ , ‘1962-12-30’ ,
  ‘98 Oak Forest, Katy, TX’ ,37000,’ 987654321’ ,4);
- **DELETE FROM** EMPLOYEE **WHERE** LNAME= ‘Brown’ ;
- **UPDATE** PROJECT
  SET PLOCATION= ‘Bellaire’ , DNUM=5
  WHERE PNUMBER=10;

4. Visões em SQL

1. Definição
2. Implementação
3. Atualização
4. Características adicionais

1. Definição: Visões em SQL

- Uma visão é uma **tabela derivada** de tabelas base (definidas no esquema) de um banco de dados ou de visões previamente definidas
  - Simplificam a **interface com o usuário** e constituem um mecanismo eficiente de **segurança**
- Comando CREATE VIEW
  - **CREATE VIEW** <nome da visão> [(<atributos>)] **AS** <comando SELECT>;

Visões em SQL

- WORKS_ON1 (FNAME, LNAME, PNAME, HOURS)

```
CREATE VIEW WORKS_ON1
AS SELECT FNAME, LNAME, PNAME, HOURS
    FROM EMPLOYEE, PROJECT, WORKS_ON
    WHERE SSN=ESSN AND PNO=PNUMBER;
```

- DEPT_INFO (DEPT_NAME, NO_OF_EMPS, TOTAL_SAL)

```
CREATE VIEW DEPT_INFO (DEPT_NAME,
                       NO_OF_EMPS, TOTAL_SAL)
AS SELECT DNAME, COUNT(*), SUM(SALARY)
    FROM DEPARTMENT, EMPLOYEE
    WHERE DNUMBER=DNO
   GROUP BY DNAME;
```

Visões em SQL

- Visões podem ser consultadas como qualquer outra tabela

```
SELECT LNAME, HOURS
FROM WORKS_ON1
WHERE PNAME= 'ProductX' ;
```

→Não precisa fazer os JOINs

- Visões sempre refletem o estado do banco de dados → não são materializadas quando definidas
- Quando não se precisa mais da visão:

DROP VIEW WORKS_ON1;

2. Implementação de Visões

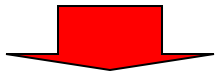
Processamento de uma consulta sobre uma visão → depende da estratégia adotada para sua implementação

1. Modificação da consulta (*query modification*)
   A consulta é modificada de acordo com a definição da visão e então executada sobre o banco de dados
2. Materização da visão (*view materialization*)
   A consulta é executada diretamente sobre a visão materializada

Implementação de Visões

- Estratégia de modificação da consulta

```
SELECT LNAME, HOURS
FROM WORKS_ON1
WHERE PNAME= 'ProductX' ;
```



```
SELECT LNAME, HOURS
FROM EMPLOYEE, PROJECT, WORKS_ON
WHERE SSN=ESSN AND
      PNO=PNUMBER AND
      PNAME= 'ProductX' ;
```

APLICAÇÃO: simplificação de consultas

Implementação de Visões

- Estratégia de materialização da consulta

  SELECT LNAME, HOURS
  FROM WORKS_ON1
  WHERE PNAME= ‘ProductX’ ;

- Define fisicamente uma tabela temporária quando a visão é consultada pela primeira vez
  - A tabela é mantida considerando que será consultada novamente
- APLICAÇÃO: eficiência de consultas
- PROBLEMA: Precisa desenvolver uma técnica para manter a visão atualizada
  - E.g., se a visão não é consultada por um tempo, ela é removida automaticamente e recriada quando consultada novamente

3. Atualização de Visões

- Visões definidas sobre uma única tabela são atualizáveis se seus atributos incluem a chave primária (ou uma das chave alternativas) da tabela base
- Visões definidas sobre múltiplas relações através de junções em geral não são atualizáveis
- Visões definidas usando-se agrupamento e funções de agregação não são atualizáveis

4. Características Adicionais

- SQL “físico”
  - Parâmetros de projeto, arquivos, métodos de acesso/índices ...
  - Específico para cada DBMS
- SQL para controle de transações (controle de concorrência e recuperação de dados)
- SQL para controle de usuários, privilégios
- SQL para criar triggers
- SQL para características objeto-relacionais
- SQL + XML...