# Modelo Relacional

André Luis Schwerz
andreluis@utfpr.edu.br

Universidade Tecnológica Federal do Paraná

Banco de Dados 1
2019/1

# Agenda

# Objetivos

Entender:

- Como os dados são representados no modelo relacional
- Quais restrições de integridade podem ser expressas
- Como os dados podem ser criados e modificados
- Como os dados podem ser manipulados e consultados
- Como obter um projeto de BD relacional com base em um diagrama ER
- O que são visões e porque elas são usadas

# Agenda

# Contextualização

- O modelo relacional foi proposto em 1970
  - Codd
- Contrapôs o modelo hierárquico e de rede
- DB2, Informix, Oracle, Sybase, Access (eca...), SQL Server, FoxBase, MySql, PostGreSQL
- SQL
  - Linguagem de Consulta Estruturada do System-R
    - Padrões SQL: ANSI SQL-86, ANSI SQL-89, ISO/ANSI SQL-92, ISO/ANSI SQL:1999, ISO/ANSI SQL:2011
    - Porém os fabricantes adicionam convenções próprias

# Introdução ao Modelo Relacional

- Um BD é uma coleção de **relações**
  - Informalmente, cada relação é uma tabela
    - Representação Tabular = colunas e linhas
- Cada linha da tabela representa uma coleção de valores relacionados
  - Uma linha $\approx$ uma entidade
  - Uma linha $\approx$ um relacionamento
- Nome da tabela e da colunas facilitam a interpretação do significado de cada linha

# Agenda

# Conceitos do Modelo Relacional

## Relação

- A **relação** é o principal construtor para representar dados
  - **Esquema** de Relação
    - Cabeçalhos de colunas da tabela
  - **Instância** da Relação
    - Tabela com linhas (e respectivos valores)

- Especifica:
  - O nome da relação
  - O nome da cada **campo**
    - ou **coluna** ou **atributo**
  - O **domínio** de cada campo
    - Descrito pelo **nome de domínio**
    - Tem um conjunto de **valores** associados

ALUNOS (id-aluno: integer, nome: string, login: string, idade: integer, média: real)

| ID-ALUNO | NOME | LOGIN | IDADE | MÉDIA |
|---|---|---|---|---|
| 53666 | Jones | jones@cs | 18 | 3.4 |
| 53688 | Smith | smith@ee | 18 | 3.2 |
| 53650 | Smith | smith@math | 19 | 3.8 |
| 53831 | Madayan | madayan@music | 11 | 1.8 |
| 53832 | Guldu | guldu@music | 12 | 2.0 |

- Uma instância de relação é um conjunto de **tuplas**
  - ou conjunto de **registos**
- Cada tupla tem o mesmo número de campos que o esquema da relação

| ID-ALUNO | NOME | LOGIN | IDADE | MÉDIA |
|---|---|---|---|---|
| 53666 | Jones | jones@cs | 18 | 3.4 |
| 53688 | Smith | smith@ee | 18 | 3.2 |
| 53650 | Smith | smith@math | 19 | 3.8 |
| 53831 | Madayan | madayan@music | 11 | 1.8 |
| 53832 | Guldu | guldu@music | 12 | 2.0 |

- Relação = conjunto de tuplas, então
  - Não pode haver registros duplicados
    - Na prática os SGBDs comerciais ignoram essa afirmativa
- A ordem dos registros não importa
- A ordem dos campos não importa
  - A não ser que as colunas não tenham nome
    - Uso posicional

- Especificam que os valores que aparecem em uma coluna devem ser extraídos do domínio associado a essa coluna
- Restringem os valores dos campos
  - Semelhante a tipos de uma linguagem de programação
- As restrições de domínio no esquema da relação são tão importantes que consideraremos apenas instâncias de relação que as satisfazem

- Seja $R(f_1 : D_1, ..., f_n : D_n)$ um esquema de relação
  - e, para cada $f_1$, $1 \leq i \leq n$
- Seja $Dom_i$ o conjunto de valores associados ao domínio chamado $D_i$
- Uma instância de $R$ que satisfaça as restrições de domínio no esquema é um conjunto de tuplas com $n$ campos:

$$\{\langle f_1 : d_1, ..., f_n : d_n \rangle | d_1 \in Dom_1, ..., d_n \in Dom_n\}$$

- Exemplo:

$\{\langle$ id-aluno:50000, nome:Dave, login:dave@cs, idade:19, media:3.3 $\rangle\}$

# Conceitos do Modelo Relacional

## Demais Termos

- **Grau** de uma relação = número de campos da relação
  - Também chamado de **Aridade** da relação
- **Cardinalidade** de relação = número de tuplas da instância da relação
- **Banco de dados relacional** = coleção de relações com nomes distintos
- **Esquema de banco de dados relacional** = coleção de esquemas das relações
- **Instância de banco de dados relacional** = coleção de instâncias de relação, uma por esquema de relação no esquema de banco de dados relacional

# Agenda

# SQL e seus subconjuntos de linguagens

- Em **SQL** (Structured Query Language) uma relação é um `TABLE`
- Subconjunto da SQL
  - **DDL** (Data Definition Language)
    - Criação, alteração e exclusão de tabelas dentre outros
  - **DML** (Data Manipulation Language)
    - Inserção, alteração e exclusão de registros entre outros
  - **DQL** (Data Query Language)
    - Apenas o comando `SELECT`
  - **DCL** (Data Control Language)
    - `GRANT` e `REVOKE`
  - **TCL** (Transactional Control Language)
    - `COMMIT`, `ROLLBACK`, `SAVEPOINT` e `SETTRANSACTION`

# Criação e Modificação de Relações em SQL

## Criação de Relações

```
CREATE TABLE ALUNOS(id_aluno     INTEGER,
                    nome         CHAR(20),
                    login        CHAR(20),
                    idade        INTEGER,
                    media        REAL);
```

- Inserção
  - Alternativamente, os nomes dos campos podem ser omitidos.

```
INSERT
INTO    ALUNOS(id_aluno, nome, login, idade, media)
VALUES  (53777, 'Mike', 'mike@ee', 17, 3.4)
```

- Atualização

```
UPDATE  ALUNOS
SET     idade = idade + 1, media = media - 1
WHERE   id_aluno = 53688
```

- Remoção

```
DELETE
FROM  ALUNOS
WHERE nome = 'Smith'
```

# Restrições de Integridade sobre Relações

## Definições

- **Restrição de Integridade** (RI)
  - Condição sobre um esquema de BD que limita (restringe) os dados que podem ser armazenados
- **Instância válida** de BD
  - Instância de BD que satisfaz todas as RI especificadas em seu esquema
- Um SGBD **impõe RIs**
  - Garante apenas a existência de instâncias válidas no BD

- RI são especificadas e verificadas em diferentes ocasiões:
  - Quando o DBA define RI
  - Quando um aplicativo que usa o SGBD é executado
    - Proíbe alterações que violam as RIs
    - Ou faz compensações para garantir as RIs
- Pode-se especificar várias RIs
  - Ex: restrições de domínio, restrições de chave, etc

- É uma declaração de que um subconjunto **mínimo** de campos de uma relação é um identificar único da tupla
  - Dois alunos não podem ter o mesmo `id_aluno`
- Um conjunto de campos que identifica uma tupla de acordo com uma restrição de chave é chamado **chave candidata** da relação
  - Na relação `ALUNOS` é o `id_aluno`

- Toda relação **deve** ter uma chave
  - Uma **superchave** é um conjunto de campos que contém uma chave
  - O conjunto de **todos** os campos é sempre uma **superchave**
    - Será?
- Podem haver várias chaves candidatas
  - Um DBA deve especificar uma **chave primária** dentre as candidatas
- Uma tupla pode ser referenciada em qualquer lugar do BD por sua chave primária
  - Qualquer chave candidata pode ser usada para isso, no entanto o SGBD espera que se use a chave primária

# Restrições de Integridade sobre Relações

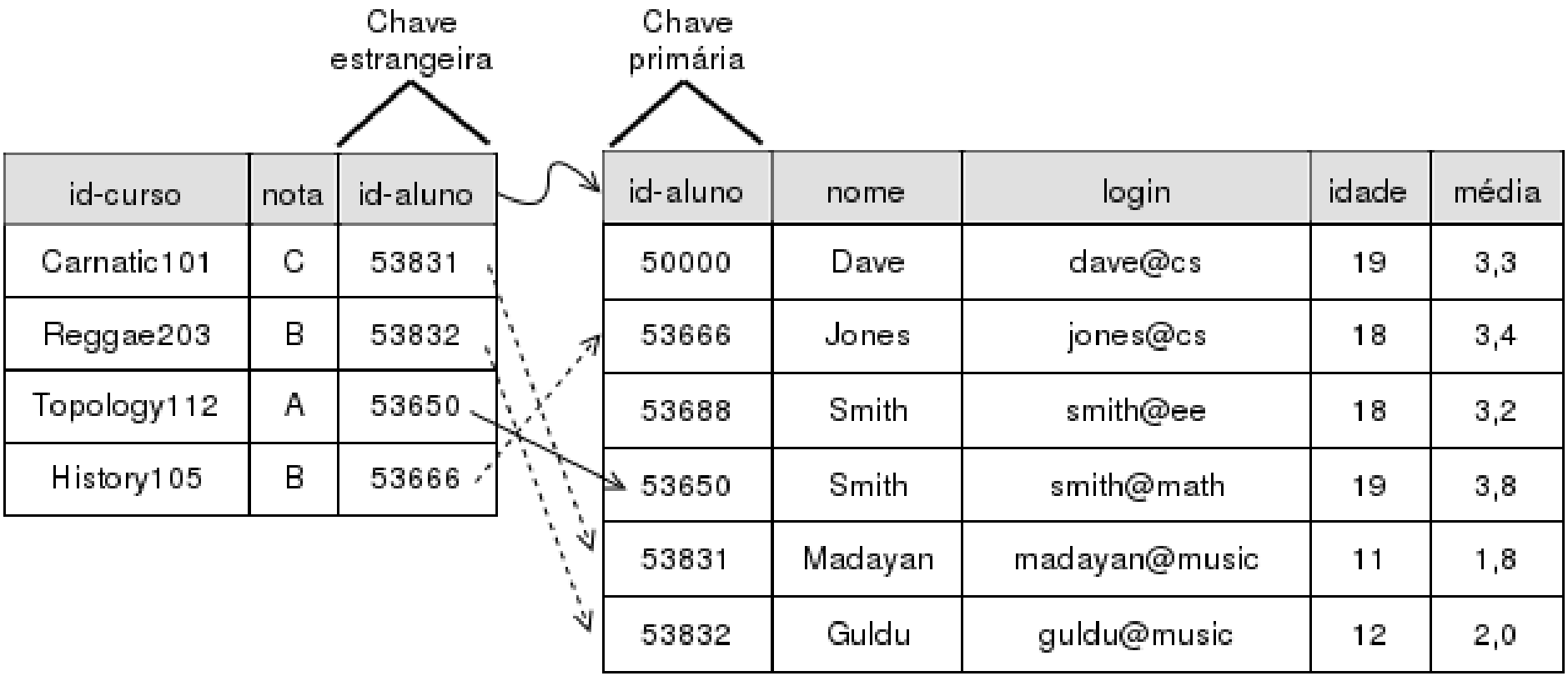
## Especificando Restrições de Chave

```
CREATE TABLE ALUNOS (id_aluno INTEGER,
                     nome CHAR(20),
                     login CHAR(20),
                     idade INTEGER,
                     media REAL,
    UNIQUE (nome, idade),
    CONSTRAINT Chave_Alunos PRIMARY KEY (id_aluno))
```

- `UNIQUE` define chaves
- `CONSTRAINT` define restrições
  - Inclusive de chave primária com o auxilio de `PRIMARY KEY`
  - Em uma violação da restrição, um erro irá retornar o nome da restrição

- Relações podem estar ligadas a outras relações
  - Manter a consistência do dados caso uma RI envolva as relações envolvidas
- **Chave estrangeira**
  - RI mais comum entre duas relações

MATRICULADO(id-aluno: integer, id-curso: string, nota: char)




| id-curso | nota | id-aluno |
|---|---|---|
| Carnatic101 | C | 53831 |
| Reggae203 | B | 53832 |
| Topology112 | A | 53650 |
| History105 | B | 53666 |

| id-aluno | nome | login | idade | média |
|---|---|---|---|---|
| 50000 | Dave | dave@cs | 19 | 3,3 |
| 53666 | Jones | jones@cs | 18 | 3,4 |
| 53688 | Smith | smith@ee | 18 | 3,2 |
| 53650 | Smith | smith@math | 19 | 3,8 |
| 53831 | Madayan | madayan@music | 11 | 1,8 |
| 53832 | Guldu | guldu@music | 12 | 2,0 |

# Restrições de Integridade sobre Relações

## Especificando Restrições de Chave Estrangeira

```
CREATE TABLE MATRICULADO(id_aluno INTEGER,
                         id_curso VARCHAR(100),
                         nota CHAR(1),
        PRIMARY KEY (id_aluno, id_curso),
        FOREIGN KEY (id_aluno) REFERENCES ALUNOS(id_aluno)))
```

# Restrições de Integridade sobre Relações

## Restrições Gerais

- Restrições de domínio, de chave primária e de chave estrangeira são fundamentais ao Modelo Relacional
- **Restrições gerais** (ou **regras de negócio**):
  - Por exemplo, alunos com idade mínima de 16 anos
    - Restrição de domínio estendida
- Restrições de Tabelas
  - Restrições que afetam apenas uma tabela
  - Por exemplo, alunos maiores de 18 anos devem ter média maior do que 3
- Assertivas
  - Restrições que envolvem várias tabelas
- Restrições de tabelas e assertivas são discutidas nas próximas aulas.

- Verificação de RIs são feitas quando relações são modificadas
  - Inserção, alteração e exclusão que violam RIs são rejeitadas
  - Violações são verificadas ao final de instruções SQL ou, quando adiadas, no final da transação

```
INSERT
INTO    ALUNOS(id_aluno, nome, login, idade, media)
VALUES  (53688, 'Mike', 'mike@ee', 17, 3.4)
```

Exemplo 1: Violação de Chave Primária: 53688 já existe

```
INSERT
INTO    ALUNOS(id_aluno, nome, login, idade, media)
VALUES  (null, 'Mike', 'mike@ee', 17, 3.4)
```

Exemplo 2: Violação de Chave Primária: Não pode ser null

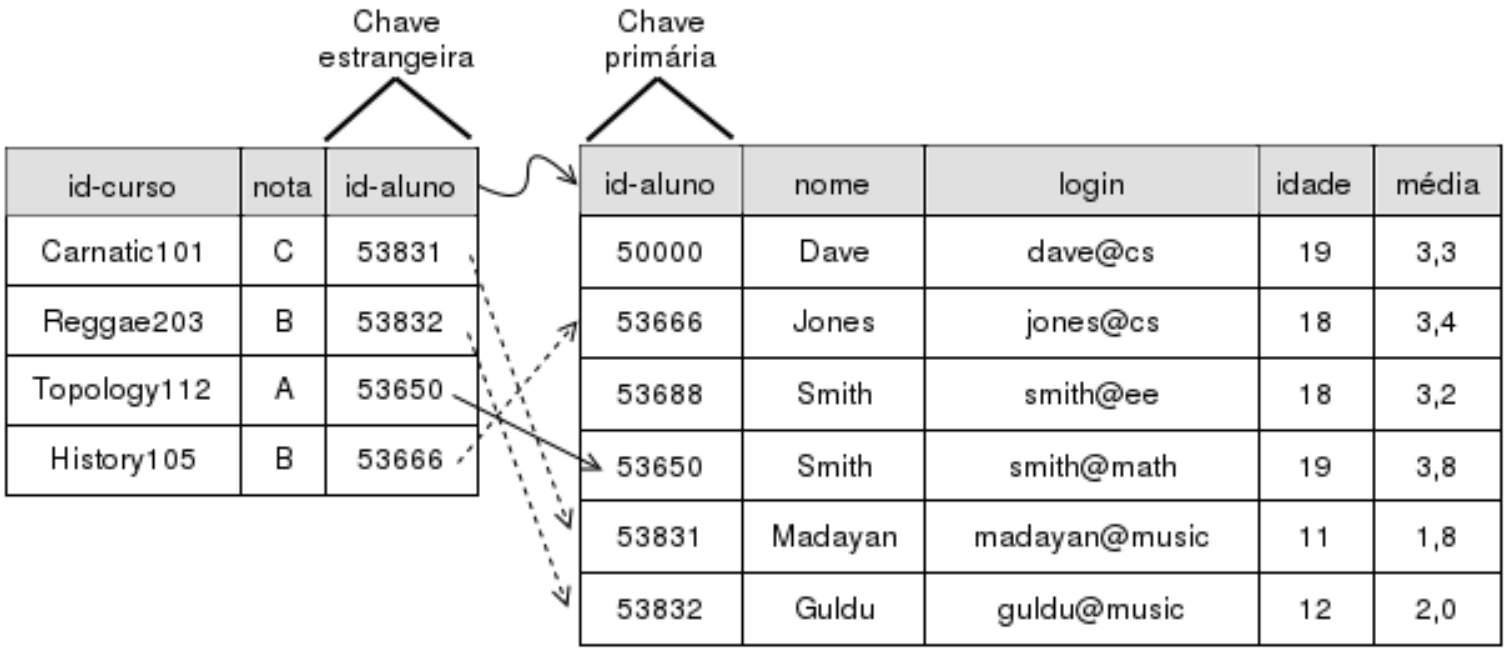
```
UPDATE  ALUNOS A
SET     A.id_aluno = 50000
WHERE   A.id_aluno = 53688
```

Exemplo 3: Violação de Chave Primária: 50000 já existe

- Inserções e atualizações com domínios errados causam violações de restrições de domínio
- Exclusões não causam violações de restrições de domínio, chave primária e de chave (`UNIQUE`)

- O impacto de restrições de chave estrangeira é mais complexo
  - Ao invés de rejeitar o comando e gerar uma violação pode-se retificar a violação

MATRICULADO(id-aluno: integer, id-curso: string, nota: char)




| id-curso | nota | id-aluno |
|---|---|---|
| Carnatic101 | C | 53831 |
| Reggae203 | B | 53832 |
| Topology112 | A | 53650 |
| History105 | B | 53666 |

| id-aluno | nome | login | idade | média |
|---|---|---|---|---|
| 50000 | Dave | dave@cs | 19 | 3,3 |
| 53666 | Jones | jones@cs | 18 | 3,4 |
| 53688 | Smith | smith@ee | 18 | 3,2 |
| 53650 | Smith | smith@math | 19 | 3,8 |
| 53831 | Madayan | madayan@music | 11 | 1,8 |
| 53832 | Guldu | guldu@music | 12 | 2,0 |

- O que fazer se uma inserção na tabela MATRICULADO refere-se a um `id_aluno` que não existe na tabela ALUNOS?

- O que fazer se uma inserção na tabela MATRICULADO refere-se a um `id_aluno` que não existe na tabela ALUNOS?
  - Rejeitar o comando

- O que fazer se uma linha de ALUNOS é excluída?

- O que fazer se uma linha de ALUNOS é excluída?
  - Excluir todas as linhas de MATRICULADO que referenciam a linha excluída em ALUNOS

- O que fazer se uma linha de ALUNOS é excluída?
  - Excluir todas as linhas de MATRICULADO que referenciam a linha excluída em ALUNOS
  - Rejeitar o comando

- O que fazer se uma linha de ALUNOS é excluída?
  - Excluir todas as linhas de MATRICULADO que referenciam a linha excluída em ALUNOS
  - Rejeitar o comando
  - Colocar um valor "padrão" para o campo `id_aluno` em cada linha de Matriculado que referencia a linha excluída em ALUNOS

- O que fazer se uma linha de ALUNOS é excluída?
  - Excluir todas as linhas de MATRICULADO que referenciam a linha excluída em ALUNOS
  - Rejeitar o comando
  - Colocar um valor "padrão" para o campo `id_aluno` em cada linha de Matriculado que referencia a linha excluída em ALUNOS
  - Colocar `null` para o campo `id_aluno` em cada linha de MATRICULADOS que referencia a linha excluída em ALUNOS
    - Especificamente para esse exemplo, essa última solução viola a restrição de chave primária de MATRICULADOS

- O que fazer se o valor de chave primária de uma linha de ALUNOS for atualizada?

- O que fazer se o valor de chave primária de uma linha de ALUNOS for atualizada?
  - As soluções são semelhantes à perguntar anterior

# Restrições de Integridade sobre Relações

## Detalhes da Verificação de RIs Referencial

```
CREATE TABLE MATRICULADOS(id_aluno INTEGER,
                          id_curso CHAR(20),
                          nota CHAR(1),
        PRIMARY KEY (id_aluno, id_curso),
        FOREIGN KEY (id_aluno) REFERENCES ALUNOS(id_aluno))
            ON DELETE CASCADE
            ON UPDATE NO ACTION)
```

- Pode-se escolher uma das quatro alternativas de solução para comandos DELETE e UPDATE:
  - NO ACTION é a opção padrão.
  - Pode-se especificar ON DELETE SET DEFAULT para falar que deverá ser colocado um valor "padrão" em `id_aluno`.
    - Esse valor padrão é colocado no campo no momento da criação da tabela:

      ```
      id_aluno INTEGER DEFAULT 53666
      ```
  - Pode-se especificar ON DELETE SET NULL

- O que é uma transação?
  - Pode conter várias instruções (consultas, inserções, etc.)
  - Geralmente é executada em um programa aplicativo que usa um SGBD
- Detecção antecipada ou tardia de comandos que violam restrições em transações?
  - Verificação após cada instrução (padrão)
  - Verificação no final da transação

# Restrições de Integridade sobre Relações

## Transações e Restrições

```
CREATE TABLE ALUNOS (id_aluno INTEGER,
                     nome CHAR(20),
                     login CHAR(20),
                     idade INTEGER,
                     distincao CHAR(10) NOT NULL,
                     media REAL,
          PRIMARY KEY (id_aluno),
          FOREIGN KEY (distincao) REFERENCES CURSOS(id_curso))
```

```
CREATE TABLE CURSOS (id_curso CHAR(10),
                     nomec CHAR(10),
                     creditos INTEGER,
                     monitor INTEGER NOT NULL,
          PRIMARY KEY (id_curso),
          FOREIGN KEY (monitor) REFERENCES ALUNOS(id_aluno))
```

```
CREATE TABLE ALUNOS (id_aluno INTEGER,
                     nome CHAR(20),
                     login CHAR(20),
                     idade INTEGER,
                     distincao CHAR(10) NOT NULL,
                     media REAL,
            PRIMARY KEY (id_aluno),
            FOREIGN KEY (distincao) REFERENCES CURSOS(id_curso))
```

## O que fazer?

```
SET CONSTRAINT nome_restrição DEFERRED
```

Pode ser DEFERRED ou IMMEDIATE

# Consultas de Dados Relacionais

- **SQL** é uma **linguagem de consulta** de SGBDs relacionais

```
SELECT      *
FROM        ALUNOS A
WHERE       A.idade < 18
ORDER BY    A.nome DESC
```

```
SELECT  A.nome, A.login
FROM    ALUNOS A
WHERE   A.idade < 18
```

```
SELECT  A.nome, M.id_curso
FROM    ALUNOS A, MATRICULADOS M
WHERE   A.id_aluno = M.id_aluno and M.nota = 'A'
```

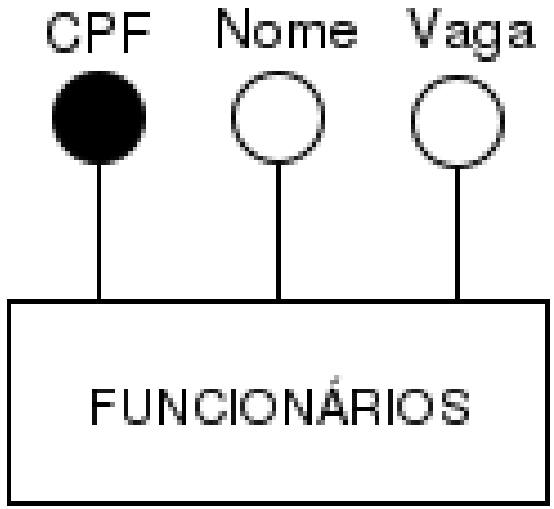
# Agenda

# ER –> Relacional

- ER é alto nível
- É possível gerar um modelo relacional (SQL) a partir de ER usando regras
  - A transformação é aproximada
    - A não ser que utilizemos técnicas avançadas (e dispendiosas) de SQL

# ER –> Relacional

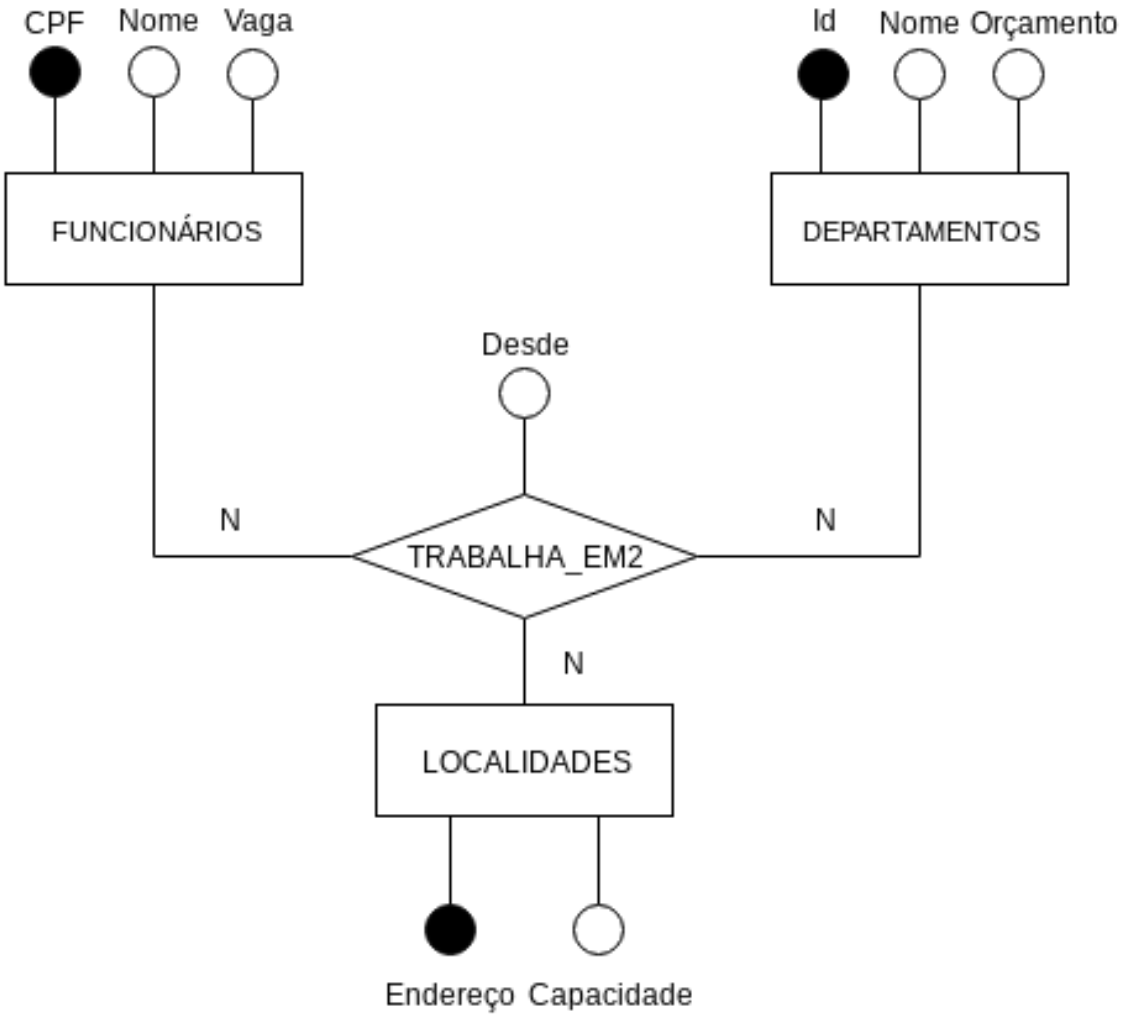
## Entidades –> Tabelas




```
CREATE TABLE FUNCIONARIOS(cpf CHAR(11),
                          nome CHAR(30),
                          vaga INTEGER,
              PRIMARY KEY(cpf))
```

# ER –> Relacional

## Relacionamentos (sem restrições) –> Tabelas



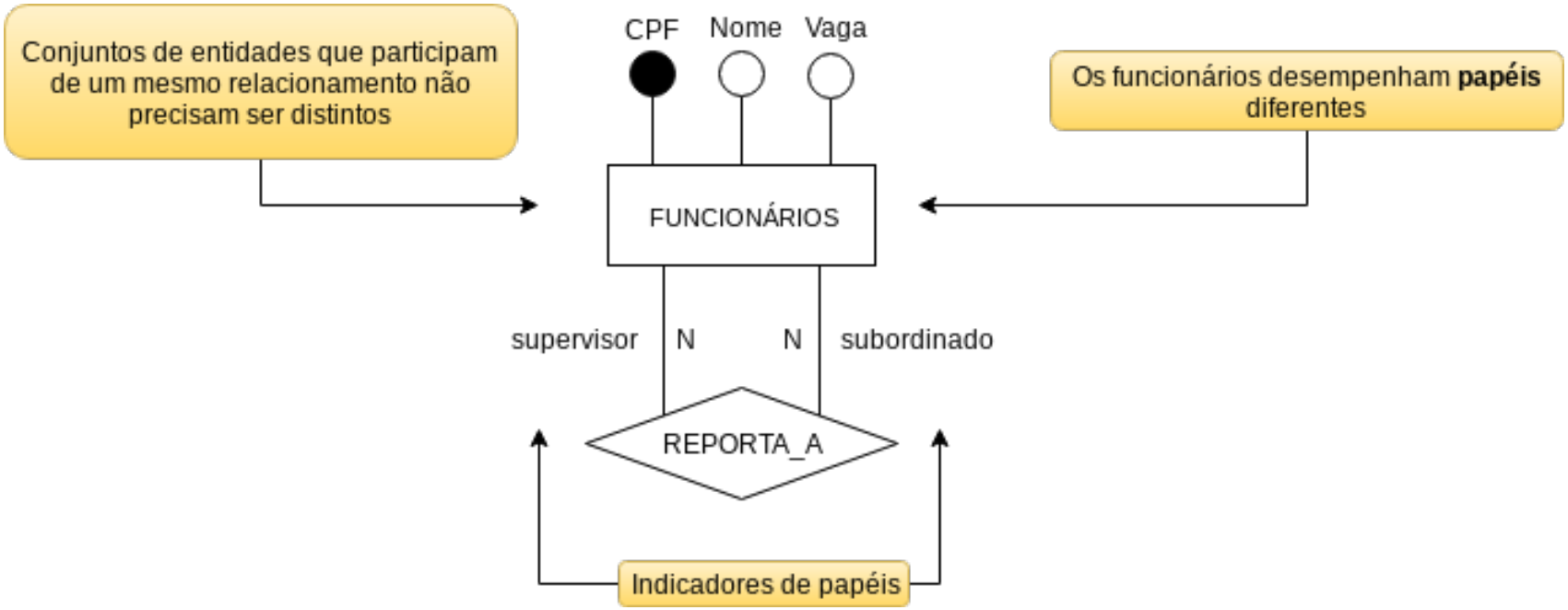
# ER –> Relacional

## Relacionamentos (sem restrições) –> Tabelas

```
CREATE TABLE TRABALHA_EM2(cpf CHAR(11),
                          id-depto INTEGER,
                          endereco CHAR(20),
                          desde DATE,
      PRIMARY KEY (cpf, id-depto, endereco),
      FOREIGN KEY (cpf) REFERENCES FUNCIONARIOS(cpf),
      FOREIGN KEY (endereco) REFERENCES LOCALIDADES(endereco),
      FOREIGN KEY (id-depto) REFERENCES DEPARTAMENTOS(id-depto))
```

# ER –> Relacional

## Relacionamentos (sem restrições) –> Tabelas



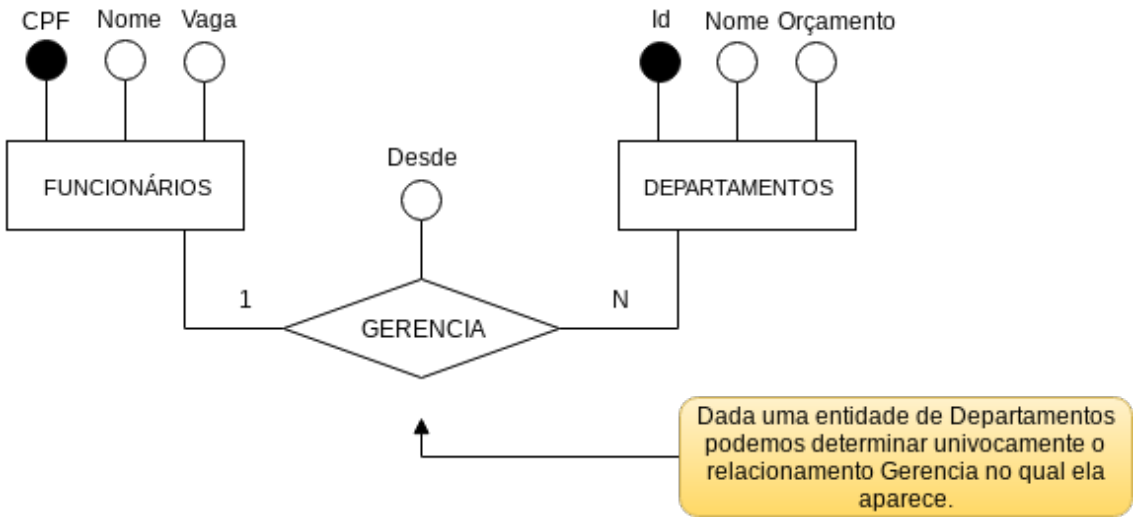
# ER –> Relacional

## Relacionamentos (sem restrições) –> Tabelas

```
CREATE TABLE REPORTA_A (supervisor_cpf CHAR(11),
                        subordinado_cpf CHAR(11),
    PRIMARY KEY (supervisor_cpf, subordinado_cpf),
    FOREIGN KEY (supervisor_cpf) REFERENCES FUNCIONARIOS(cpf),
    FOREIGN KEY (subordinado_cpf) REFERENCES FUNCIONARIOS(cpf))
```

# ER –> Relacional

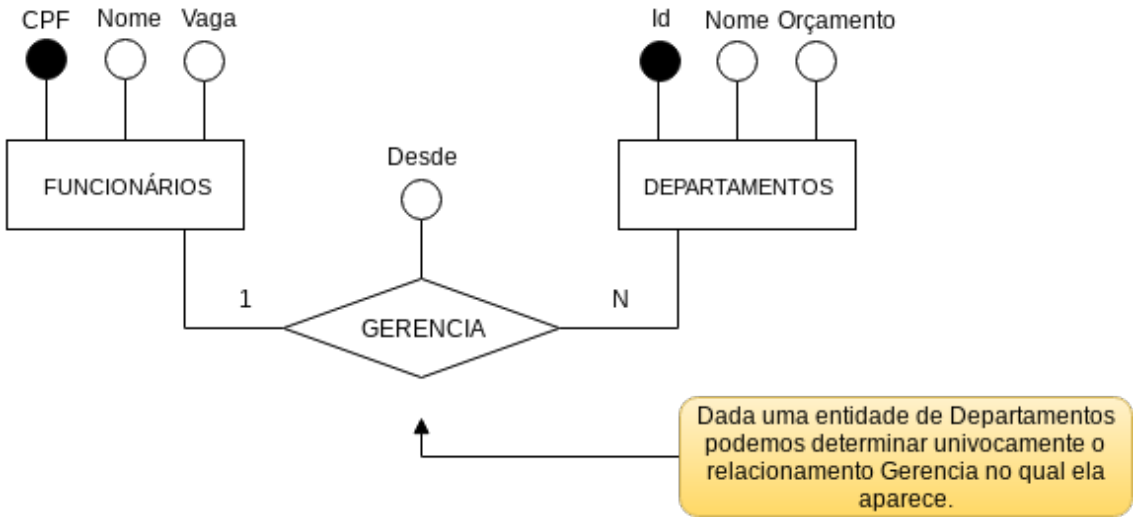
## Relacionamentos (com restrições de chave) –> Tabelas - Estratégia 1




```
CREATE TABLE GERENCIA(cpf CHAR(11),
                      id-depto INTEGER,
                      desde DATE,
            PRIMARY KEY (id-depto),
            FOREIGN KEY (cpf) REFERENCES FUNCIONARIOS(cpf),
            FOREIGN KEY (id-depto) REFERENCES DEPARTAMENTOS(id-depto))
```

# ER –> Relacional

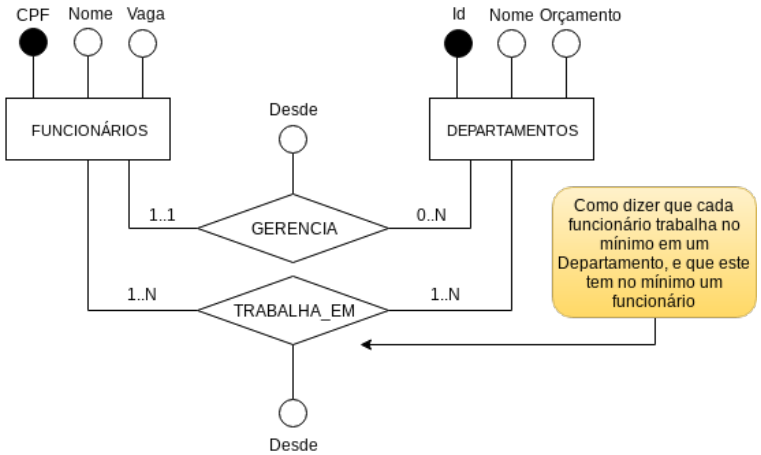
## Relacionamentos (com restrições de chave) –> Tabelas - Estratégia 2




```
CREATE TABLE DEPTO_GERENCIA(id-depto INTEGER,
                            nome-depto CHAR(20),
                            orcamento REAL,
                            cpf CHAR(11),
                            desde DATE,
                PRIMARY KEY (id-depto),
                FOREIGN KEY (cpf) REFERENCES FUNCIONARIOS(cpf))
```

# ER –> Relacional

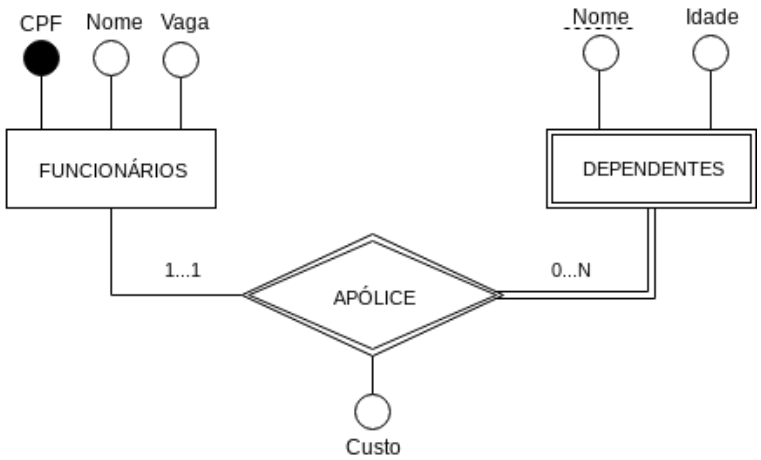
## Relacionamentos (com restrições de participação) –> Tabelas




```
CREATE TABLE DEPTO_GERENCIA(id-depto INTEGER,
                            nome-depto CHAR(20),
                            orcamento REAL,
                            cpf CHAR(11) NOT NULL,
                            desde DATE,
                PRIMARY KEY (id-depto),
                FOREIGN KEY (cpf) REFERENCES FUNCIONARIOS(cpf) ON DELETE
                    NO ACTION)
```

# ER –> Relacional

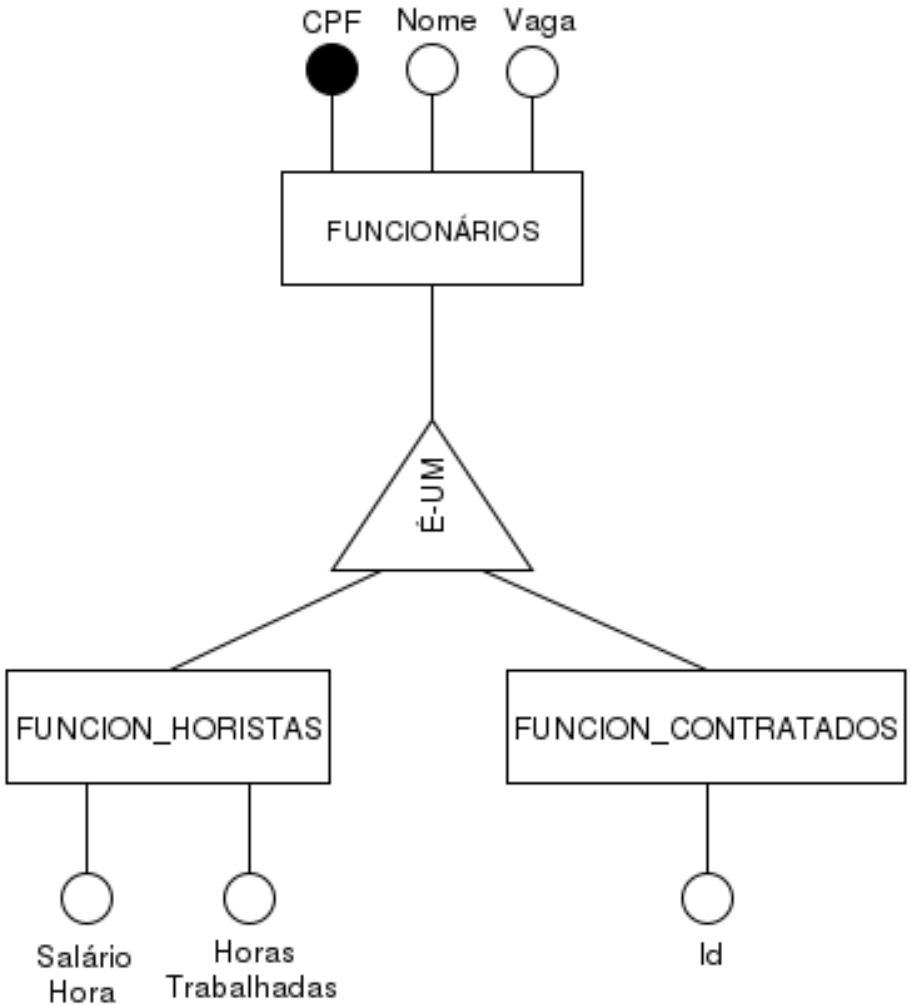
## Relacionamentos (entidades fracas) –> Tabelas




```
CREATE TABLE APOLICE_DEP(nome CHAR(20),
                         idade INTEGER,
                         custo REAL,
                         cpf CHAR(11),
                PRIMARY KEY (nome, cpf),
                FOREIGN KEY (cpf) REFERENCES FUNCIONARIOS(cpf) ON DELETE
                     CASCADE)
```

# ER –> Relacional

## Hierarquia de Classes –> Tabelas

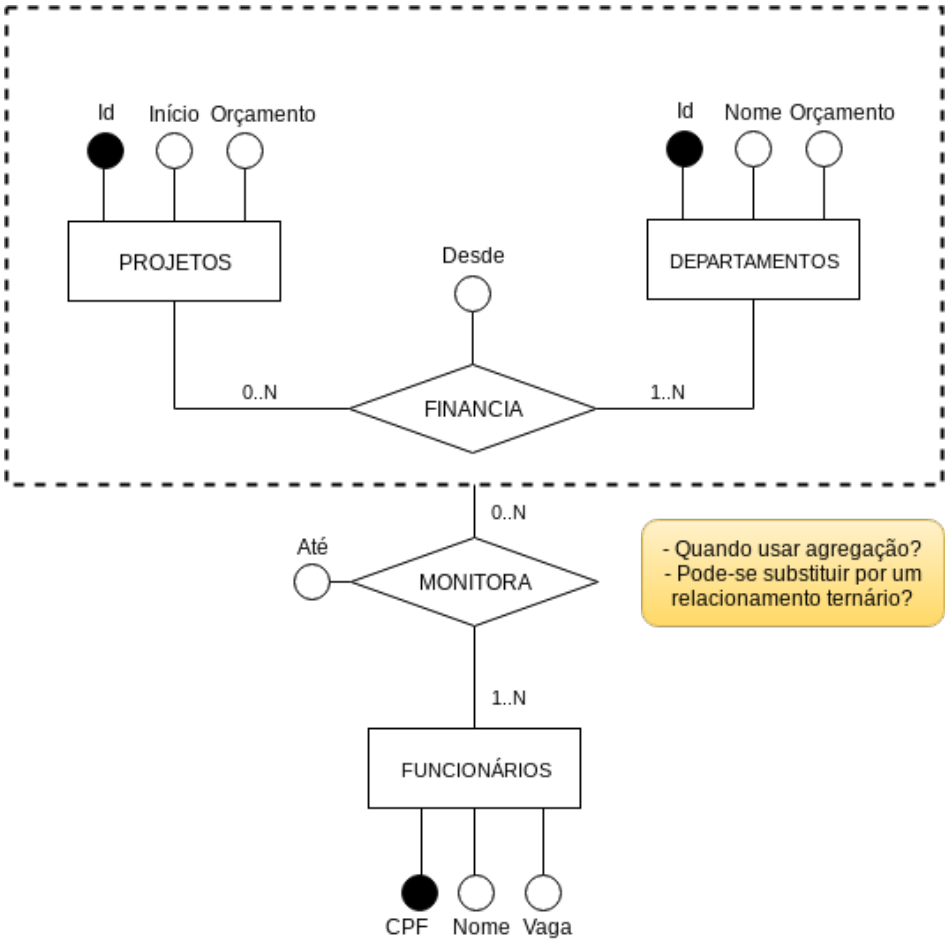


- **Estratégia 1:** mapear cada um dos conjuntos de entidades FUNCIONARIOS, FUNCION_HORISTAS e FUNCION_CONTRATADOS em relações distintas
  - Em geral é sempre aplicável
  - Quando um registro da superclasse é excluído, a exclusão deve ser propagada às demais classes

- **Estratégia 2:** criar apenas duas relações, FUNCION_HORISTAS e FUNCION_CONTRATATOS
  - Não aplicável caso haja funcionários não são horistas nem contratados
  - Duplicação de registros caso um funcionário seja horista e contratado ao mesmo tempo

# ER –> Relacional

## Agregação –> Tabelas



# ER –> Relacional

## Agregação –> Tabelas

- FUNCIONARIOS, DEPARTAMENTOS e PROJETOS e o relacionamento FINANCIA são mapeados normalmente
- MONITORA deve conter os atributos de chave de FUNCIONARIOS(cpf), de chave de FINANCIA (id-depto, id-projeto) e os atributos descritivos de MONITORA (até)

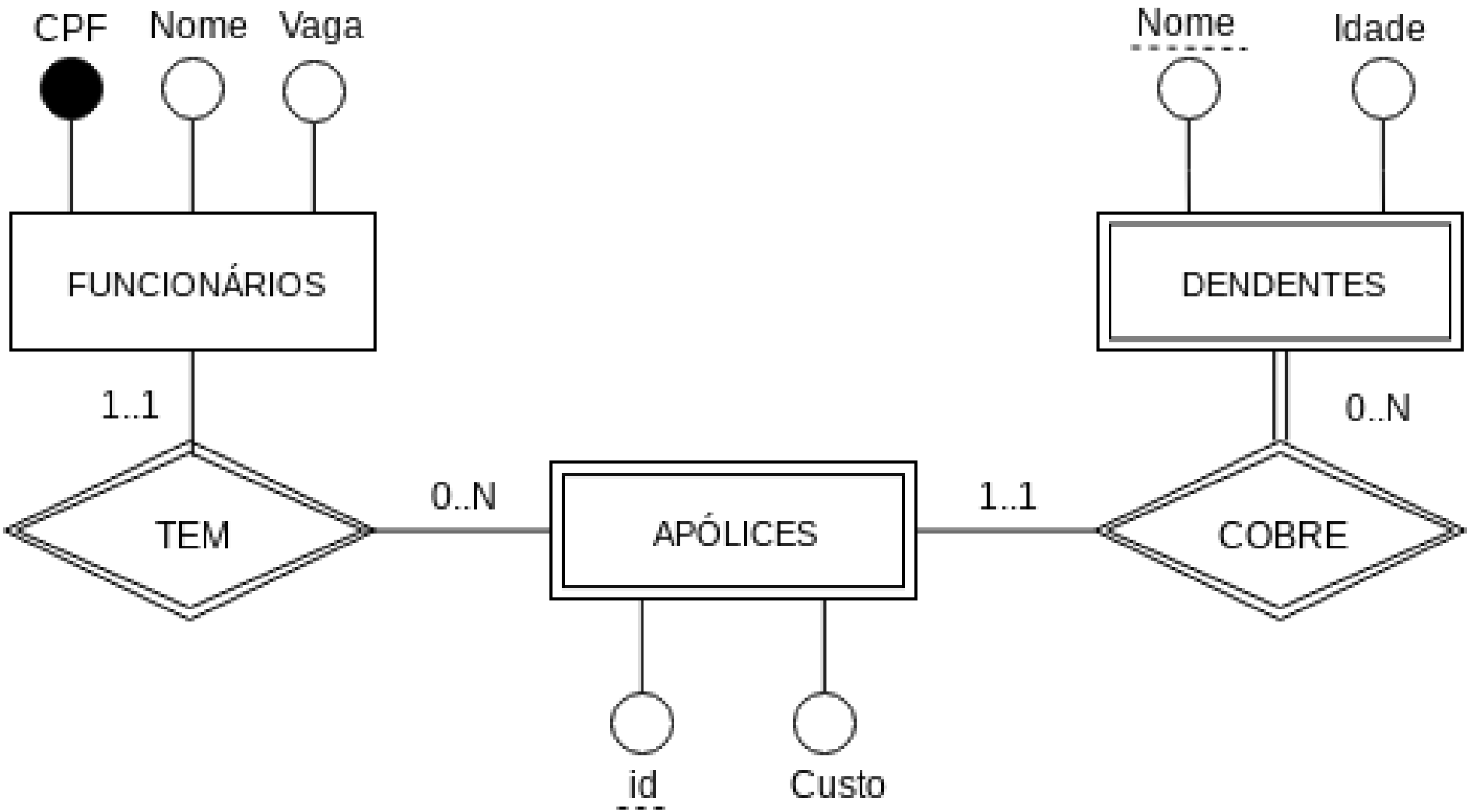
# ER –> Relacional

## Agregação –> Tabelas

- Em um caso especial, quando FINANCIA não tem atributos descritivos e tem participação total em MONITORA, pode-se:
  - Eliminar a relação FINANCIA, porque pode-se obter essa relação a partir `<id-projeto, id-depto>` de MONITORA

# ER –> Relacional

## Mais Exemplos



# ER –> Relacional

## Mais Exemplos

```
CREATE TABLE APOLICES(id_apolice INTEGER,
                      custo REAL,
                      cpf CHAR(11),
              PRIMARY KEY (cpf, id-apolice),
              FOREIGN KEY (cpf) REFERENCES FUNCIONARIOS(cpf) ON
                  DELETE CASCADE)
```

```
CREATE TABLE DEPENDENTES(nomed CHAR(20),
                         idade INTEGER,
                         id_apolice INTEGER
                         cpf CHAR(11),
               PRIMARY KEY(cpf, id-apolice, nomed),
               FOREIGN KEY (cpf, id_apolice) REFERENCES APOLICES(cpf,
                    id-apolice) ON DELETE CASCADE)
```

# Agenda

# Visões

- Uma **Visão** é uma tabela cujas linhas não são explicitamente armazenadas
  - São calculadas conforme necessário a partir de uma **definição de visão**

# Visões

## Exemplo

- Considerando que precisamos localizar frequentemente nomes e identificadores de aluno dos estudantes que tiram nota B, juntamente com o seu curso

```
CREATE VIEW Estudantes_B(nome, id_aluno, curso) AS
    SELECT  A.nome, A.id_aluno, M.id_curso
    FROM    ALUNOS A, MATRICULADOS M
    WHERE   A.id_aluno = M.id_aluno AND M.nota = '5'
```

- Esquema físico descreve como relações do esquema conceitual são armazenadas
- Esquema conceitual é a coleção de esquemas das relações armazenadas no BD
  - Algumas podem fazer parte do esquema externo por meio de visões
    - Independência lógica de dados no modelo relacional
    - Enfatizar segurança por meio da limitação de acesso às informações

- Pode-se atualizar os dados das visões:
  - Visões atualizáveis
  - SQL-92 apenas tabelas únicas
    - SQL:1999 muda essa afirmação e define outras regras

```
CREATE VIEW Estudantes_Bons(id_aluno, media) AS
    SELECT A.id_aluno, A.media
    FROM ALUNOS A
    WHERE A.media > 3,0
    WITH CHECK OPTION
```

# Visões

## Destruindo/Alterando tabelas e visões

- DROP TABLE
  - DROP TABLE ALUNOS RESTRICT
  - DROP TABLE ALUNOS CASCADE
- DROP VIEW
  - Funciona como DROP TABLE
- ALTER TABLE
  - ALTER TABLE ALUNOS ADD COLUMN nome-familia CHAR(10)

# Agenda

# Estudo de Caso

## A Loja na Internet

```
CREATE TABLE LIVROS(isbn CHAR(10),
                    titulo CHAR(80),
                    autor CHAR(80),
                    qtidade-em-estoque INTEGER,
                    preco REAL,
                    ano_publicacao INTEGER,
                PRIMARY KEY(isbn))
```

# Estudo de Caso

## A Loja na Internet

```
CREATE TABLE PEDIDOS(isbn CHAR(10),
                     id-cliente INTEGER,
                     num-cartao CHAR(16),
                     qtidade INTEGER,
                     data-pedido DATE,
                     data-remessa DATE,
        PRIMARY KEY (isbn,id-cliente),
        FOREIGN KEY (isbn) REFERENCES LIVROS(isbn),
        FOREIGN KEY (id-cliente) REFERENCES CLIENTES(id-cliente))
```

# Estudo de Caso

## A Loja na Internet

```
CREATE TABLE CLIENTES(id-cliente INTEGER,
                      nomec CHAR(80),
                      endereco CHAR(200),
                PRIMARY KEY (id-cliente))
```

# Estudo de Caso

## A Loja na Internet

### Novos requisitos foram adicionados:

- Os clientes poderão adquirir vários livros diferentes em um único pedido. Por exemplo, se um cliente quiser pedir três exemplares de "O professor de Inglês" e dois de "O Caráter da Lei da Física", ele deverá ser capaz de fazer um único pedido para ambos os livros.
- Os pedidos podem ser enviados parcialmente.
- Os cliente podem fazer mais de um pedido por dia e eles querem identificar os pedidos que fazem.
- Os cliente devem ter os números de cartão de crédito protegidos dos funcionário que informa o status dos pedidos quando os cliente ligam.

# Estudo de Caso

## A Loja na Internet

```
CREATE TABLE PEDIDOS(num-pedido INTEGER,
                     isbn CHAR(10),
                     id-cliente INTEGER,
                     num-cartao CHAR(16),
                     qtidade INTEGER,
                     data-pedido DATE,
                     data-remessa DATE,
        PRIMARY KEY (num-pedido, isbn),
        FOREIGN KEY (isbn) REFERENCES LIVROS(isbn),
        FOREIGN KEY (id-cliente) REFERENCES CLIENTES(id-cliente))
```

# Estudo de Caso

## A Loja na Internet

```
CREATE VIEW InfoPedido(isbn,id-cliente,
                       qtidade, data-pedido,
                       data-remessa) AS
    SELECT P.id-cliente, P.qtidade,
           P.data-pedido, P.data-remessa
    FROM PEDIDOS P
```

# Agenda

# O que aprendemos?

# Considerações Finais

- Como os dados são representados no modelo relacional
- Quais restrições de integridade podem ser expressas
- Como os dados podem ser criados e modificados
- Como os dados podem ser manipulados e consultados
- Como obter um projeto de BD relacional com base em um diagrama ER
- O que são visões e porque elas são usadas

# Atividades Extra-classe

- RAMAKRISHNAN, Raghu; GEHRKE, Johannes. **Sistemas de gerenciamento de banco de dados.** 3. ed. São Paulo, SP: McGraw-Hill do Brasil, 2008. 884 p. ISBN 9788577260270.
- Leitura do Capítulo 3 - O Modelo Relacional
- Exercícios 3.1 a 3.20.