AULA 01

- Bancos de Dados Relacionais
- Arquitetura
- Linguagem SQL
- Componentes: DDL, DML, DCL

Prof. Msc. Célio R. Castelano

TÓPICOS

- Projeto de Bancos de Dados Relacionais
- Arquitetura
  - Modelagem Conceitual
  - Modelagem Lógica
  - Modelagem Física
- Mapeamento do Modelo Conceitual para o Modelo Relacional
- Linguagem SQL
  - Componentes: DDL, DML, DCL

Projeto de Banco de Dados:

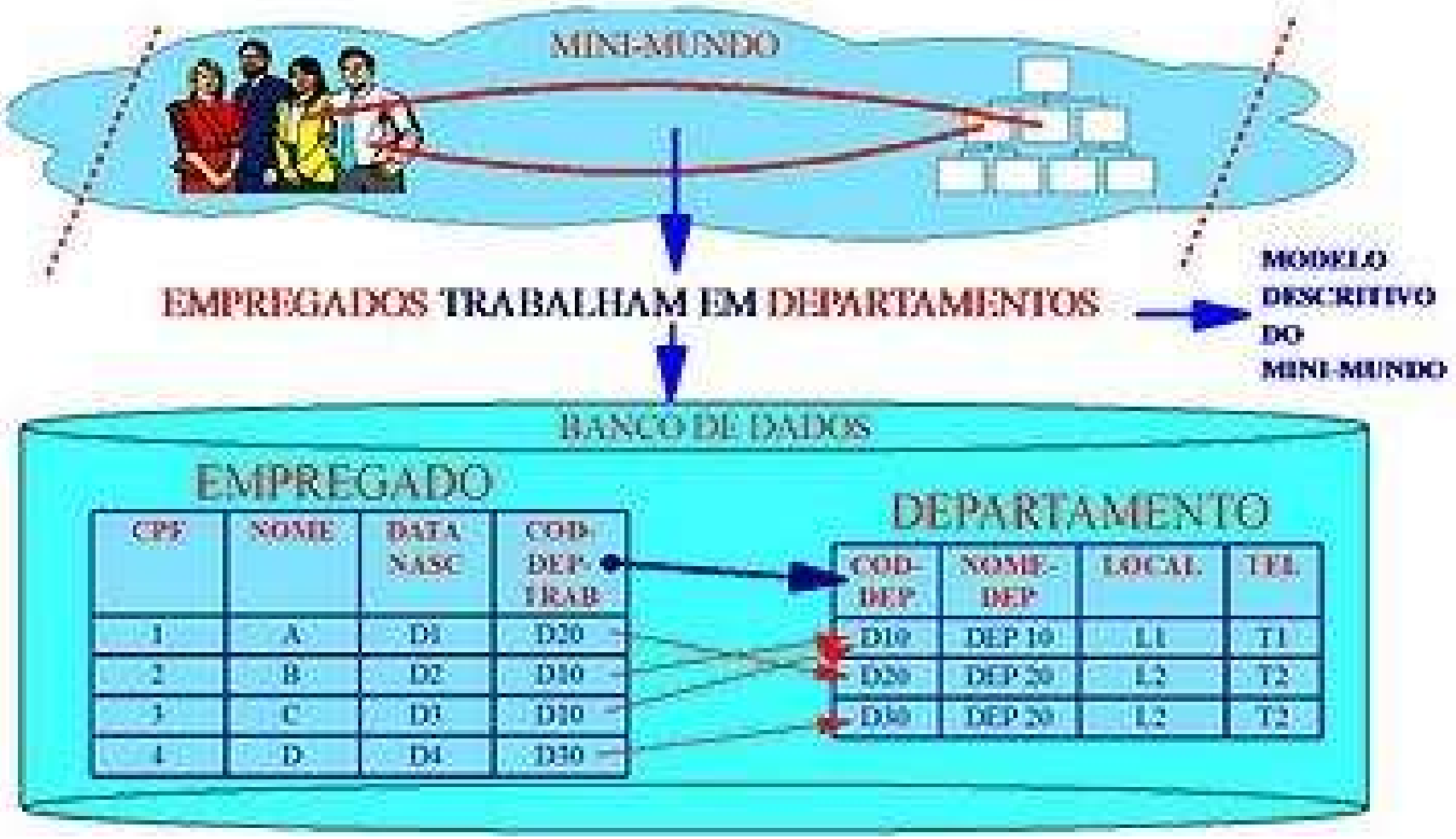
- É a atividade que tem como propósito especificar a Estrutura e o Comportamento de um Banco de Dados (Modelo do Banco de Dados), tendo como ponto de partida os Requisitos de Informação e as Regras de Negócio (Modelo descritivo) inerentes a um determinado Domínio do Problema (Mini-Mundo, Domínio de Conhecimento, Parcela do Mundo Real), com a utilização de Ferramentas de Projeto ou Modelagem (MER, MC-UML, MDR, ...), procurando atender a uma série de critérios de qualidade (Requisitos de Qualidade de Projeto ou Modelagem).

Como um Banco de Dados é projetado?

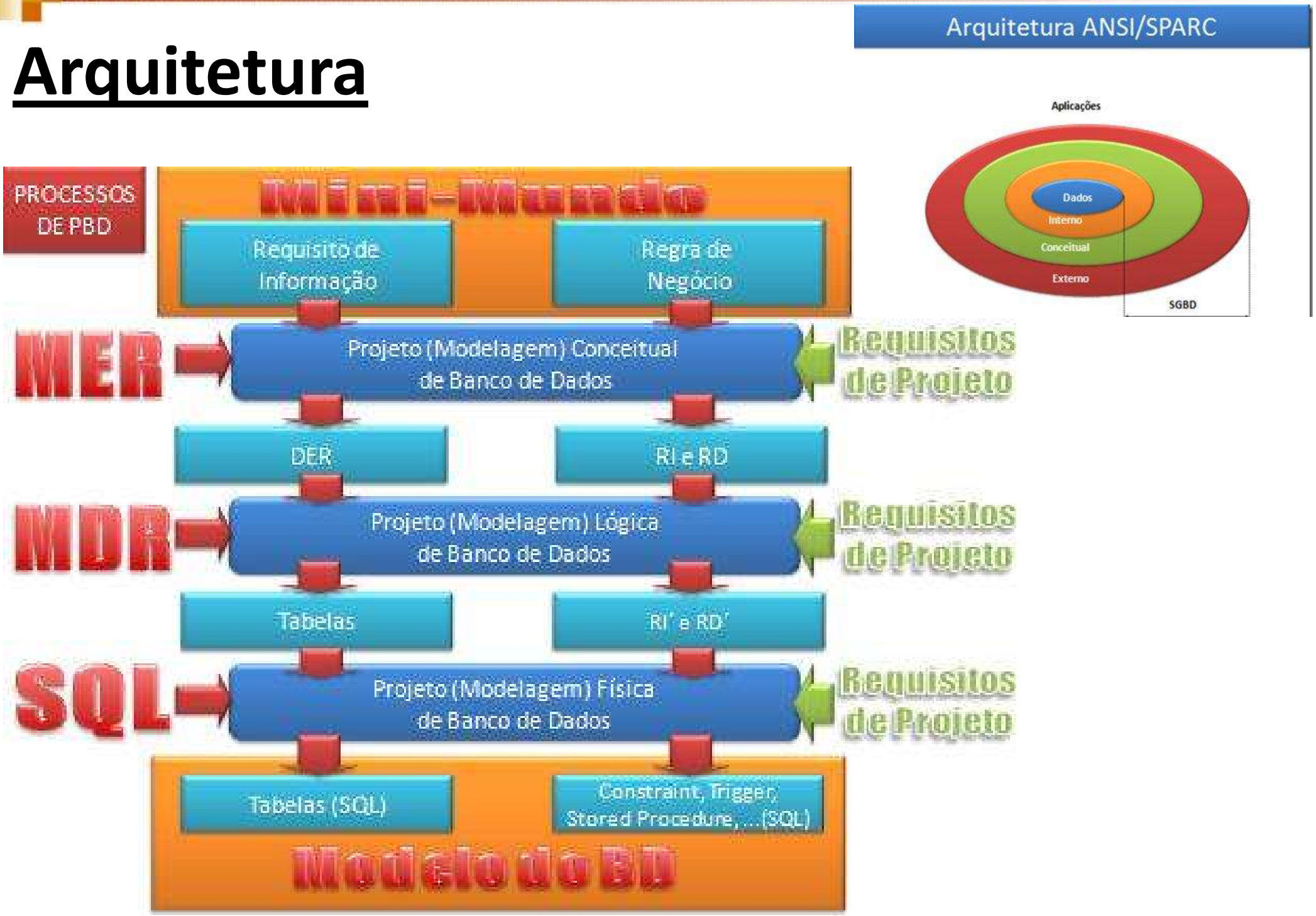
COMO UM BANCO DE DADOS É PROJETADO ?

Domínio do Problema

Domínio do Conhecimento



Arquitetura



Arquitetura

- Mini Mundo
  (Nível Externo)

A empresa deseja controlar suas categorias de materiais de acordo com cada fornecedor, onde vários fornecedores podem fornecer diversas categorias de materiais.

- Projeto(Modelagem) Conceitual
  (Nível Conceitual)

- Projeto(Modelagem) Lógico
  (Nível Conceitual)

- Projeto(Modelagem) Físico
  (Nível Interno)

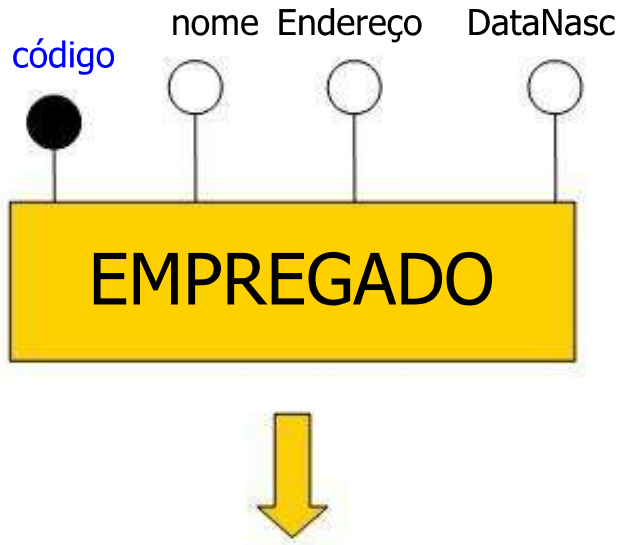
- create table fornecedor (cgc integer not null, ...)
- create table fornece(cgc_fornecedor intenger...)

Mapeamento do Modelo Conceitual para o Modelo Relacional

- Um modelo conceitual construído utilizando o M.E.R. (Modelo Entidade-Relacionamento) pode ser mapeado para um modelo lógico Relacional
  - A equivalência mais direta é
    cada Entidade = uma Tabela
    cada Atributo = uma Coluna
    cada Ocorrência = uma Linha

Mapeamento de Entidades

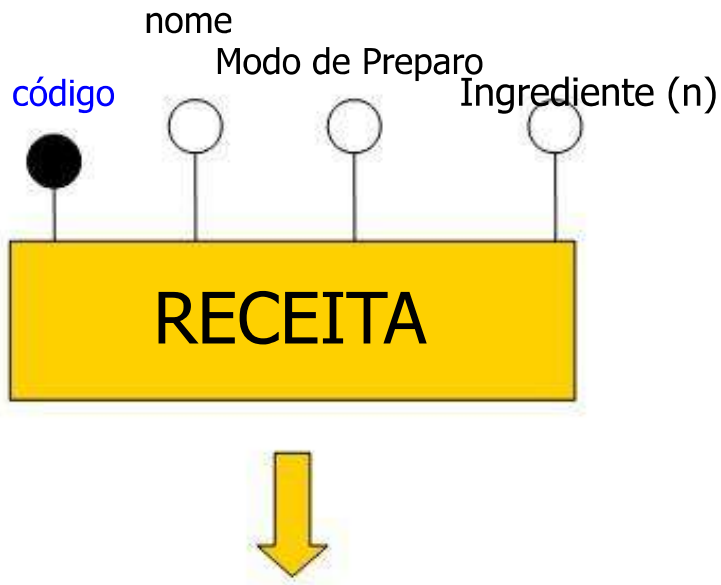
- Para cada Entidade deve ser criada uma relação (tabela)
- Para cada atributo simples incluir uma coluna na tabela
- No caso de atributo composto, incluir somente os atributos simples que o compõe (Endereço)



Empregado(Codigo, Nome, Logradouro, Numero, Bairro, Cidade, Estado, DataNasc)

Mapeamento de Atributos Multivalorados

- Para cada atributo multivalorado deve ser criada uma tabela formada pela chave primária da Tabela/Entidade e pelo atributo multivalorado
- A chave primária da nova tabela será o par de atributos
- Se o atributo multivalorado for composto. Por ex.: Ingrediente formado por Nome do ingrediente e quantidade, todo o grupo vai para a nova tabela



Receita(Codigo, Nome, Modo_Preparo)
Ingrediente_Receita (CodReceita, Ingrediente)

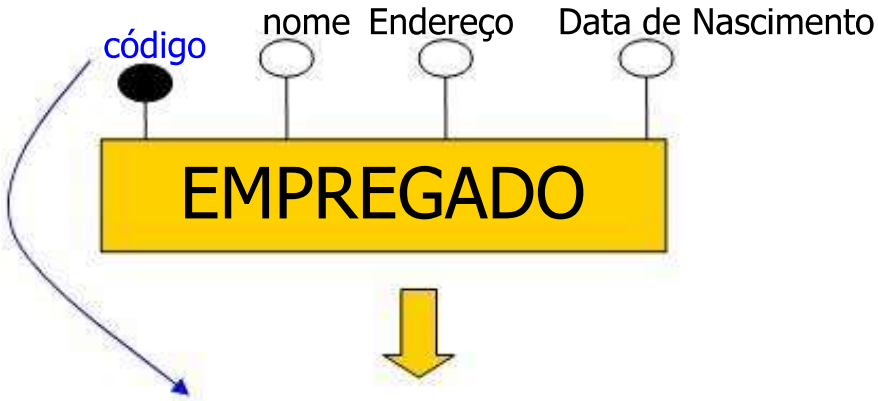
Nome das colunas

- O nome do atributo(modelo conceitual) pode ser diferente do nome do campo da tabela (coluna)
- Objetivo
  - Facilidade de programação como o uso de nomes curtos
  - Evitar nomes iguais. Ex.: Todas entidades com o atributo nome
  - Evitar espaços no nome da coluna, pois são proibidos nos BD relacionais
- Dica
  - Defina um padrão para converter o nome dos atributos, principalmente dos nomes compostos que necessitam abreviação.

Ex.: Nome do Responsável → NomeRes

Nome para a chave primária

- É boa prática compor o nome da chave primária com uma identificação da tabela a qual ela pertence



Empregado(CodEmp, Nome, Endereco, DataNasc)

- Como geralmente, elas se tornam chaves estrangeiras de outras tabelas, esta prática facilita a programação e compreensão dos campos

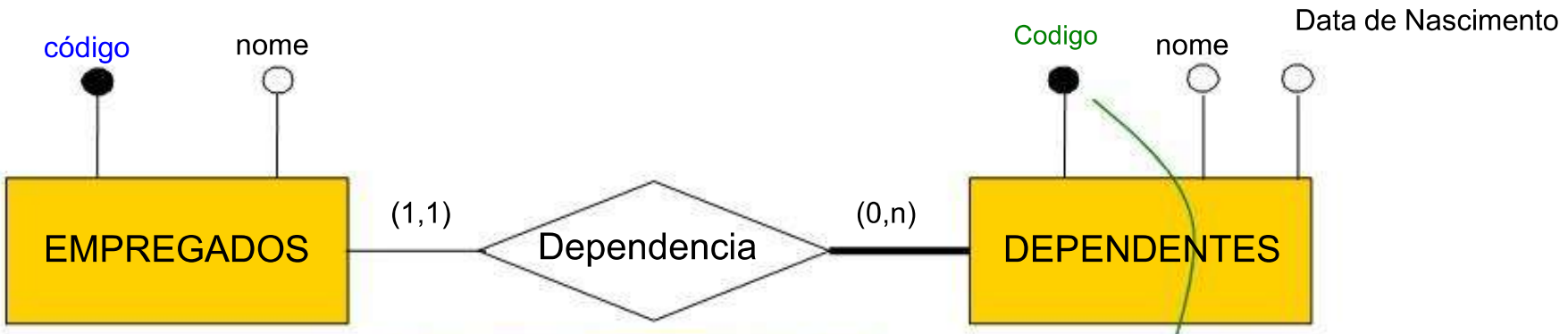
Mapeamento de Relacionamentos

- Um relacionamento pode ser transformado em:
  - Uma tabela
  - Uma coluna de uma das tabelas envolvidas
  - Fusão de duas tabelas

- Esta decisão depende da cardinalidade mínima e máxima dos relacionamentos
- No caso da fusão devemos considerar também outros relacionamentos da entidade

Entidades Fracas

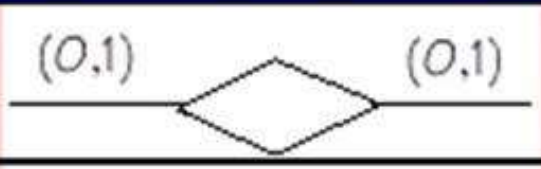
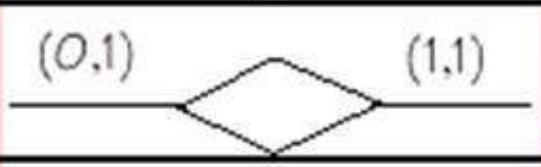
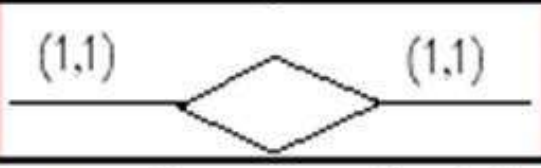
- Criar uma tabela para cada entidade fraca
- Nessa tabela incluir como chave estrangeira a chave primária da tabela que representa a entidade possuidora/identificadora
- As entidades fracas têm chave primária composta de duas partes:
  - A chave primária da tabela (entidade) possuidora
  - A chave parcial da tabela(entidade) fraca



Empregados(CodEmp, Nome)

Dependentes(CodEmp, CodDep, Nome, DataNasc)

Implementação de Relacionamentos 1:1

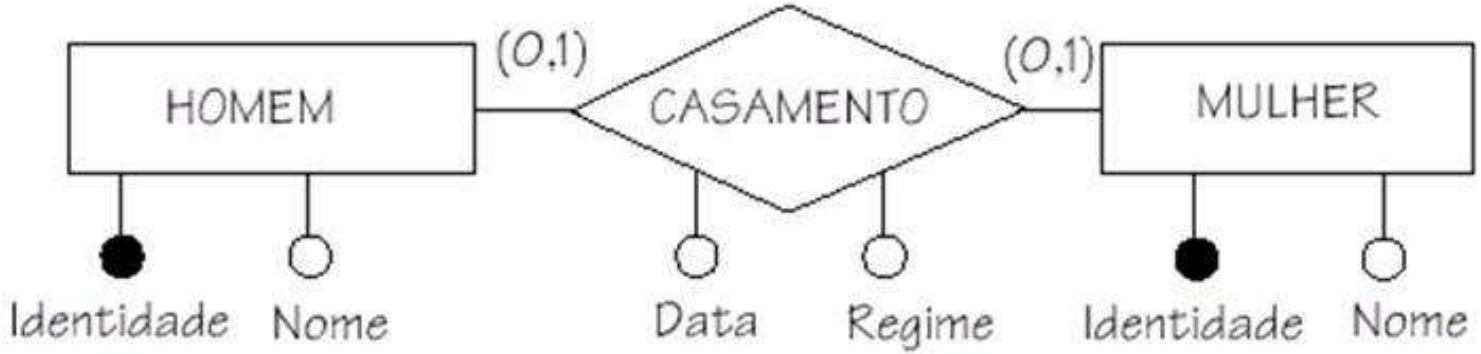
Tipo de relacionamento	Regra de implementação		
	Tabela própria	Adição coluna	Fusão tabelas
	±	✓	×
	×	±	✓
	×	×	✓

✓ Alternativa preferida ± Pode ser usada

× Não usar

Relacionamentos Binários - Um para Um

- Ambas entidades têm participação opcional
  - adição de colunas



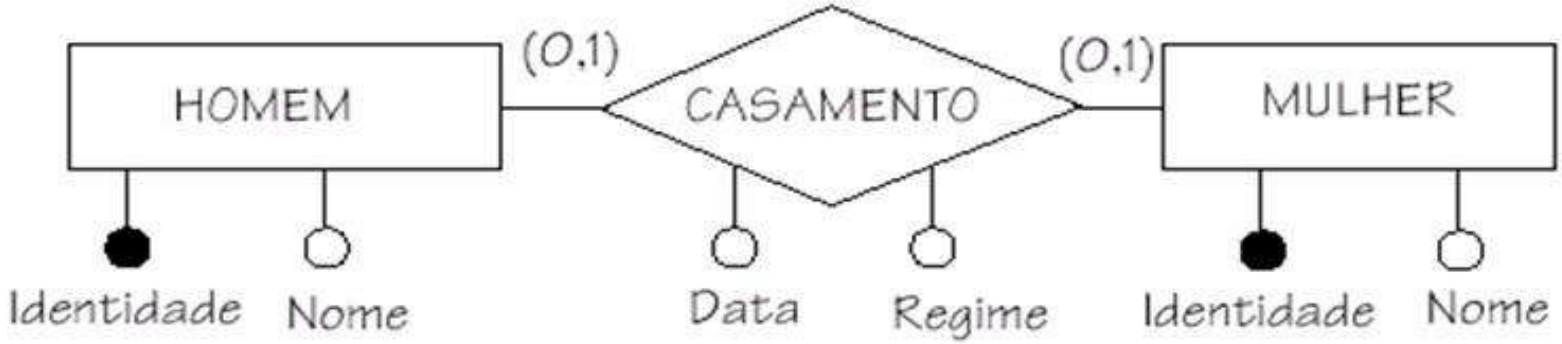
Mulher (IdentM,Nome,**IdentH,Data,Regime)**

IdentH referencia Homem

Homem (IdentH,Nome)

Relacionamentos Binários - Um para Um

- Ambas entidades têm participação opcional
  - tabela própria



Mulher (IdentM,Nome)

Homem (IdentH,Nome)

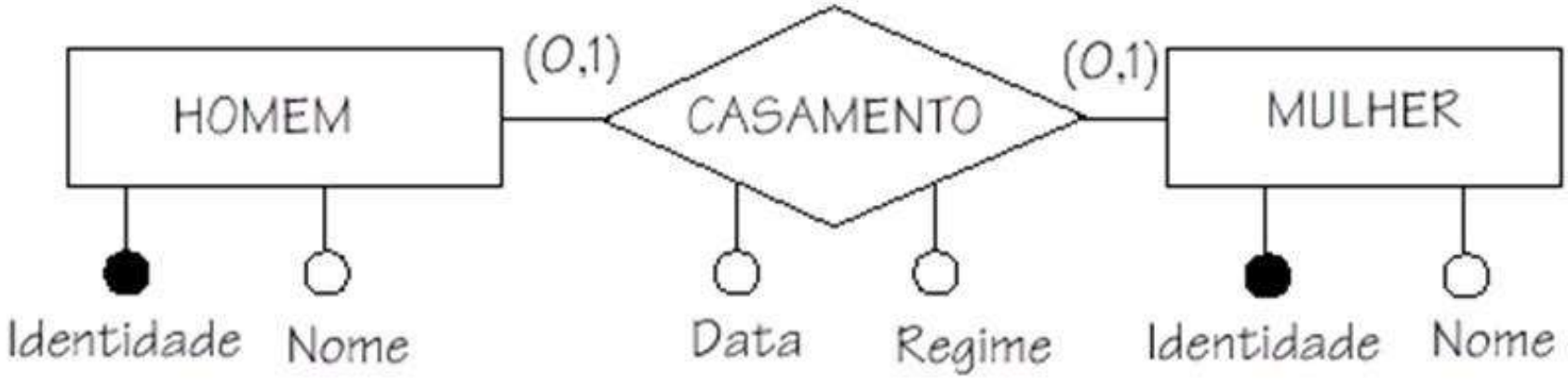
Casamento (IdentM,IdentH,Data,Regime)

IdentM referencia Mulher

IdentH referencia Homem

Relacionamentos Binários - Um para Um

- Ambas entidades têm participação opcional
  - fusão de tabelas

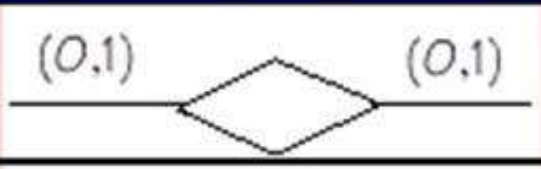
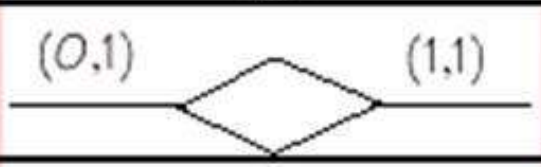
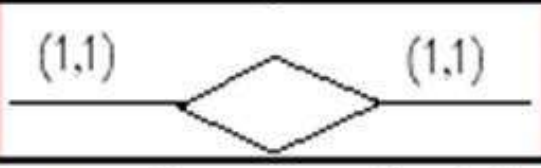


Casamento (IdentM, IdentH, Data, Regime, **NomeH, NomeM**)

Relacionamentos Binários - Um para Um

- Ambas entidades têm participação opcional
- Solução por fusão de tabelas é inviável
  - Chave primária artificial (já que pode não estar completa)
- Solução por adição de colunas: melhor
  - Menor número de junções
  - Menor número de chaves
- Solução por tabela própria: aceitável

Implementação de Relacionamentos 1:1

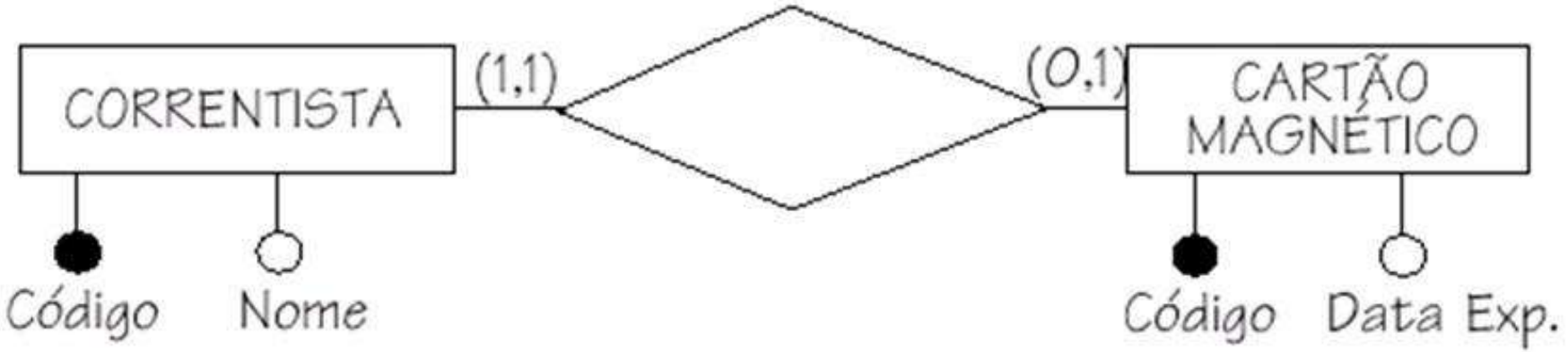
Tipo de relacionamento	Regra de implementação		
	Tabela própria	Adição coluna	Fusão tabelas
	±	✓	×
	×	±	✓
	×	×	✓

✓ Alternativa preferida ± Pode ser usada

× Não usar

Relacionamentos Binários - Um para Um

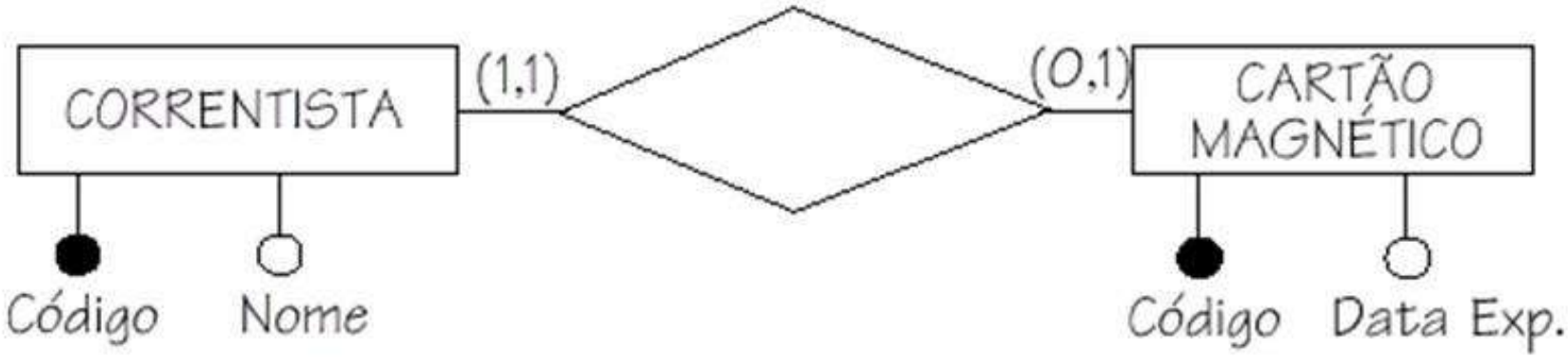
- Participação opcional/obrigatória
  - Fusão de tabelas



Correntista (CodCorrent,Nome,CodCartão,DataExp)

Relacionamentos Binários - Um para Um

- Participação opcional/obrigatória
  - Adição de colunas

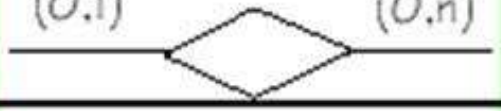
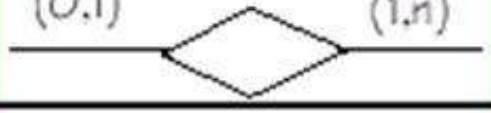
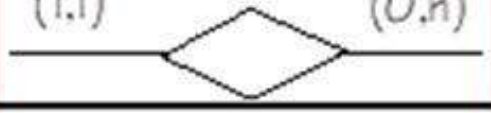
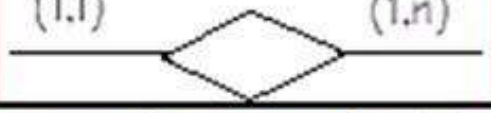


Correntista (CodCorrent,Nome)
Cartão(CodCartão,DataExp,**CodCorrent**)
CodCorrent referencia Correntista

Relacionamentos Binários - Um para Um

- Adição de colunas versus fusão de tabelas
  - Fusão de tabelas é melhor em termos de número de junções e número de chaves
  - Adicão de colunas é melhor em termos de campos opcionais
  - Fusão de tabelas é considerada a melhor e adição de colunas é aceitável

Relacionamentos Binários - Um para Muitos

Tipo de relacionamento	Regra de implementação		
	Tabela própria	Adição coluna	Fusão tabelas
	±	✓	×
	±	✓	×
	×	✓	×
	×	✓	×

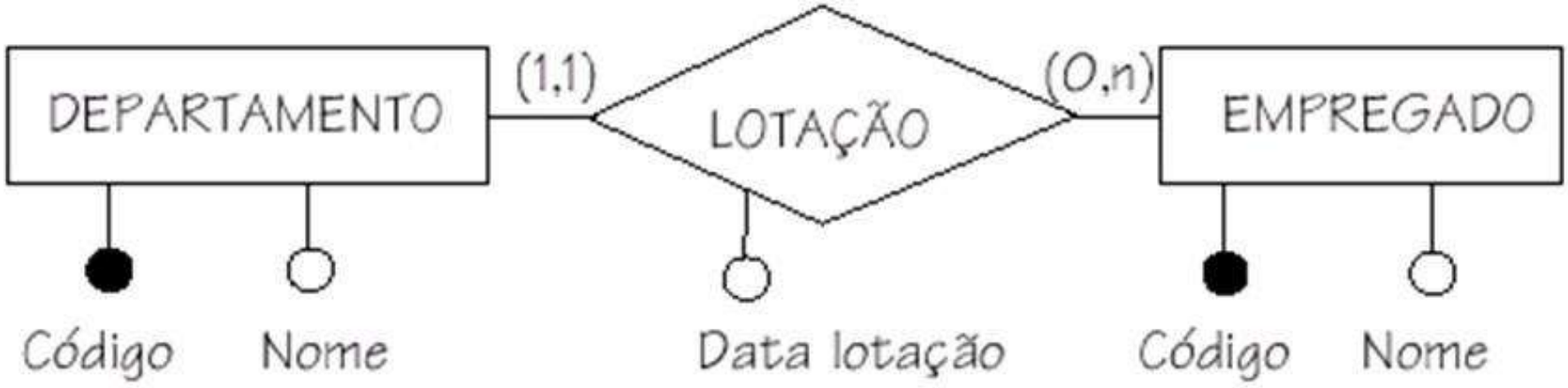
✓ Alternativa preferida ± Pode ser usada

× Não usar

Relacionamentos Binários - Um para Muitos

- Já temos duas tabelas, uma para cada conjunto de entidades que participam do relacionamento
- Incluir como chave estrangeira, na tabela do “lado muitos” (o “lado N”), a chave primária da tabela do “lado um”
- Incluir também colunas com os atributos do relacionamento

Relacionamentos Binários - Um para Muitos

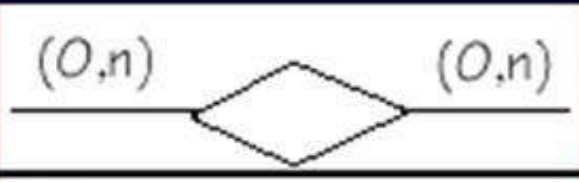
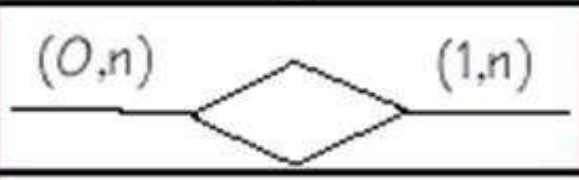
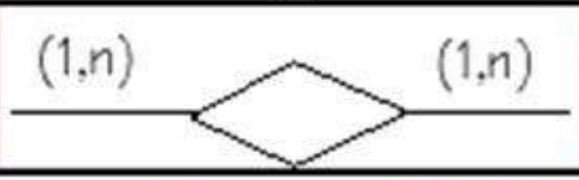


Departamento (CodDept,Nome)

Empregado (CodEmp,Nome,**CodDept,DataLota**)

CodDept referencia Departamento

Relacionamentos Binários - Muitos para Muitos

Tipo de relacionamento	Regra de implementação		
	Tabela própria	Adição coluna	Fusão tabelas
	✓	×	×
	✓	×	×
	✓	×	×

✓ Alternativa preferida × Não usar

Relacionamentos Binários - Muitos para Muitos

- Já temos duas tabelas, uma para cada conjunto de entidades que participa do relacionamento
- Criar uma nova tabela contendo, como chaves estrangeiras, as chaves primárias dessas duas tabelas
  - A combinação dessas chaves estrangeiras forma a chave primária da nova tabela
- Incluir também colunas com os atributos do relacionamento

Relacionamentos Binários - Muitos para Muitos

Engenheiro (CodEng,Nome)

Projeto (CodProj,Título)

Atuação (CodEng,CodProj,Função)

CodEng referencia Engenheiro

CodProj referencia Projeto

A Linguagem SQL

- O SQL foi desenvolvido originalmente no início dos anos 70 nos laboratórios da IBM em San Jose, dentro do projeto System R(Sistema R), que tinha por objetivo demonstrar a viabilidade da implementação do modelo relacional proposto por Edgar Frank Codd.
- O nome original da linguagem era *SEQUEL*, acrônimo para *"Structured English Query Language"* (*Linguagem de Consulta Estruturada em Inglês*), vindo daí o fato de, até hoje, a sigla, em inglês, ser comumente pronunciada "síquel" ao invés de "és-kiú-él", letra a letra.
- No entanto, em português, a pronúncia mais corrente é a letra a letra: "ése-quê-éle".

A Linguagem SQL

- A SQL é uma linguagem de banco de dados abrangente: ela possui comandos para definição de dados, consultas e atualizações.
- Assim, ela tem ambas as DDL e DML. Além disso, tem funcionalidades para a definição de visões (views) no banco de dados, a fim de especificar a segurança e as autorizações para as definições de restrições de integridade e de controles de transação.
- Ela também possui regras para embutir os comandos SQL em linguagens de programação genérica como Java, COBOL ou C/C++.

A Linguagem SQL

- Como a especificação do padrão SQL está em expansão, com mais funcionalidades a cada versão, o último padrão SQL-99 foi dividido em uma especificação de núcleo (core) mais pacotes (packages) opcionais.
- O núcleo deve ser implementado por todos os vendedores de SGBDs relacionais compatíveis com o padrão.
- Os pacotes podem ser implementados como módulos opcionais, que podem ser adquiridos independentemente para as aplicações específicas de um banco de dados, como data mining (garimpagem de dados), dados espaciais, dados temporais, data warehousing, processamento analítico on-line (OLAP), dados multimídia, e assim por diante.

A Linguagem SQL

• A linguagem SQL é dividida em subconjuntos de acordo com as operações que queremos efetuar sobre um banco de dados. Neste curso aprenderemos apenas sobre os dois primeiros subconjuntos, mas é importante que o estudante tenha uma noção geral dos subconjuntos básicos.

1. **DDL - Linguagem de Definição de Dados**
2. **DML - Linguagem de Manipulação de Dados**
3. **DCL - Linguagem de Controle de Dados**

A Linguagem SQL

1 – DDL - Linguagem de Definição de Dados(*Data Definition Language*)

É um conjunto de comandos dentro da SQL usada para a definição das estruturas de dados, fornecendo as instruções que permitem a criação, modificação e remoção das tabelas, assim como criação de índices.

Estas instruções SQL permitem definir a estrutura de uma base de dados, incluindo as linhas, colunas, tabelas, índices, e outros metadados.

Entre os principais comandos DDL estão CREATE (Criar), DROP (deletar) e ALTER (alterar).

A Linguagem SQL

1 – DDL - Linguagem de Definição de Dados(de *Data Definition Language*)

Exemplos:

```
• CREATE TABLE ALUNO (
        alu_ra integer not null,
        alu_nome varchar(80)  );

• CREATE TABLE PROFESSOR (
        pro_codigo integer,
        pro_nome varchar(50),
        pro_admissao date );

• DROP TABLE ALUNO;

• DROP TABLE PROFESSOR;
```

A Linguagem SQL

2- DML – Linguagem de Manipulação de Dados (*Data Manipulation Language*)

É o grupo de comandos dentro da linguagem SQL utilizado para a recuperação, inclusão, remoção e modificação de informações em bancos de dados.

Os principais comandos DML são SELECT (Seleção de Dados), INSERT (Inserção de Dados), UPDATE (Atualização de Dados) e DELETE (Exclusão de Dados).

A Linguagem SQL

2 – DML – Linguagem de Manipulação de Dados (*Data Manipulation Language*)

Exemplos:

- `SELECT NUMERO, NOME FROM empregados;`
- `INSERT INTO empregados`

  `(nome, data_nascimento, salario, cidade, estado)`
  `VALUES`

  `('José','05/14/1965',1000,'Brasilia','DF');`
- `UPDATE empregados`

  `SET nome='João da Silva',cidade='São Paulo'`

  `WHERE codigo_empregado=2;`
- `DELETE FROM empregados where codigo_empregado=1;`

A Linguagem SQL

3 – DCL – Linguagem de Controle de Dados (*Data Control Language*)

É o grupo de comandos que permitem ao administrador de banco de dados controlar o acesso aos dados deste banco. Alguns exemplos de comandos DCL são:

GRANT: Permite dar permissões a um ou mais usuários e determinar as regras para tarefas determinadas;

REVOKE: Revoga permissões dadas por um GRANT.

As tarefas básicas que podemos conceder ou barrar permissões são:

- CONNECT
- SELECT
- INSERT
- UPDATE
- DELETE
- USAGE

A Linguagem SQL

3 – DCL – Linguagem de Controle de Dados (*Data Control Language*)

Exemplos:

- `GRANT SELECT ON CLIENTES TO PUBLIC`
- `GRANT SELECT,INSERT ON PESSOAS TO JOSE,PEDRO`
- `GRANT UPDATE(SALARIO) ON FOLPAGTO TO ANAMARIA`
- `REVOKE SELECT ON CLIENTES TO PUBLIC`
- `REVOKE SELECT,INSERT ON PESSOAS TO JOSE,PEDRO`
- `REVOKE UPDATE(SALARIO) ON FOLPAGTO TO ANAMARIA`
- `REVOKE EXECUTE ON PROCEDURE SOMATUDO TO ANA,JOSE`