Sistemas de Banco de Dados

Fundamentos em Bancos de Dados Relacionais

Wladmir Cardoso Brandão
www.wladmirbrandao.com

Fevereiro, 2020

Seção 01

Introdução aos Sistemas de Banco de Dados

Banco de Dados (BD)

- Coleção de dados estruturados
- Atende as necessidades de grupos de usuários
- Dados → Símbolos, sinais, códigos passíveis de registro
- Essencial na vida moderna
- Presente em diferentes ambientes de negócio:
  - Saque ou depósito de dinheiro no banco
  - Reserva de hotel
  - Reserva de livros na biblioteca
  - Compra de produtos em supermercado

Exemplo de Aplicação de BD

Amazon.com

- Milhões de livros, CDs, vídeos, DVDs, jogos eletrõnicos, roupas e outros produtos
- Dezenas de milhões de acessos diários para efetivação de compras
- Atualização constante de:
  - Clientes cadastrados
  - Pedidos de compra
  - Produtos em estoque
  - Vendas de produtos

Propriedades de BD

- Reflete a realidade do "mundo real" dos usuários
- "Mundo real" $\rightarrow$ Minimundo ou Universo de Discurso
- Construído e abastecido (populado) para uma finalidade específica
- Coerência na coleção lógica dos dados
- Provê compartilhamento de dados

Taxonomia de BD: Utilização

- Manual
  - Criado e mantido sem o uso de computadores
  - Exemplo → Lista telefônica
- **Computadorizado**
  - Criado e mantido por um grupo de programas
  - Exemplo → The Human Genome Database (GDB)

Taxonomia de BD: Aplicação

- **Tradicional** → Dados na forma textual ou numérica
- Multimídia → Imagens, áudios e vídeos
- Geográficos → Mapas, imagens de satélite e registros climáticos.
- *Data Warehouse* → "Armazém" de dados utilizados no processamento analítico online (OLAP) para auxílio à tomada de decisão
- Ativo e de Tempo Real → Processos industriais de manufatura

Abordagens de BD

- Processamento em arquivo
  - Usuário define e implementa os arquivos necessários para uma aplicação específica como parte da programação da aplicação.
- **Sistema Gerenciador de Banco de dados (SGBD)**
  - **Repositório único → Dados definidos uma vez e depois acessados por vários usuários.**
  - Abstração de dados
  - Compartilhamento de dados
  - Isolamento entre programas e dados
  - Natureza autodescritiva
  - Suporte à múltiplas visões dos dados
  - Processamento de transação multiusuário

Projeto de BD

- Construção de modelos para implementação de bancos de dados
- Modelos → Representações de objetos e eventos do mundo real
- Etapas na implementação de bancos de dados:
  1. Especificação → Descrição do minimundo
  2. Análise de requisitos → Restrições de operação
  3. Projeto conceitual → Estruturas e restrições conceituais
  4. Projeto lógico → Estruturas e restrições lógicas
  5. Projeto físico → Estruturas e restrições físicas
- BDs devem ser implementados, alimentados e mantidos continuamente para refletir o estado do minimundo

Atores em BD

- Administradores → *DataBase Administrators (DBAs)*
- Projetistas
- Analistas
- Programadores
- Usuários

Sistema Gerenciador de Banco de Dados (SGBD)

- Coleção de programas que permitem aos usuários criar e manter um banco de dados
- Definição de BD → Especificar tipos, estruturas e restrições armazenadas sob a forma de metadados no catálogo (dicionário) do sistema
- Construção de BD → Armazenar dados em meio controlado pelo SGBD
- Manipulação de BD → Consulta, atualização, inserção e exclusão de dados
- Compartilhamento de BD → Acesso simultâneo por múltiplos usuários ao banco de dados

SGBD: Componentes Básicos

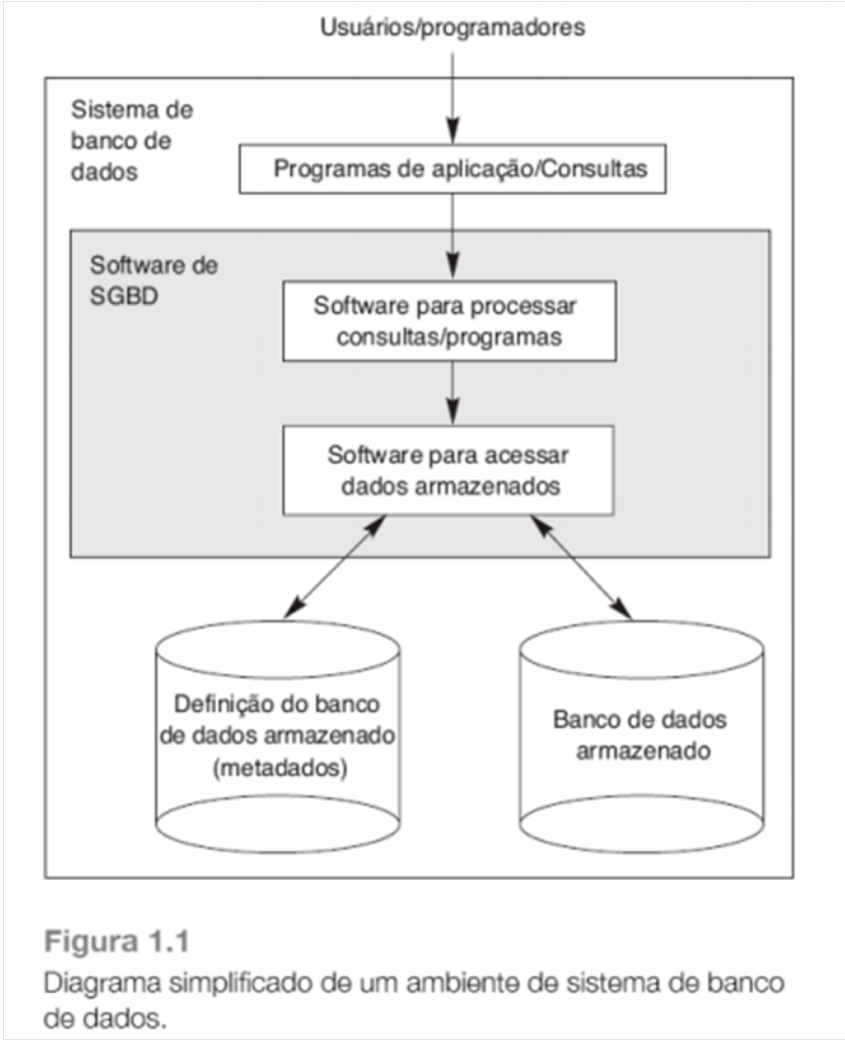


Figura 1.1
Diagrama simplificado de um ambiente de sistema de banco de dados.

SGBD: Vantagens de Utilização

- Controle de Redundância → Dados armazenados uma única vez, sem duplicatas

Figura 1.1: Sistemas isolados

SGBD: Vantagens de Utilização

- Controle de Redundância → Dados armazenados uma única vez, sem duplicatas

Figura 1.2: Sistemas integrados com dados compartilhados

SGBD: Vantagens de Utilização

- *Backup* e recuperação
- Múltiplas interfaces ao usuário
- Representação de relacionamentos complexos entre os dados
- Restrições de acesso
- Restrições de integridade

SGBD: Vantagens de Utilização

- Disponibilidade de dados atualizados
- Economias de escala
- Flexibilidade
- Potencial para garantia de padrões
- Tempo reduzido para desenvolvimento de aplicações

SGBD: Quando não utilizar?

- Monousuário → Acesso por múltiplos usuários não requerido
- Baixa complexidade → Aplicações de BD muito simples e bem definidas
- Requisitos rigorosos → Aplicações de tempo real e sistemas embarcados com capacidade de armazenamento limitada
- Alta especialização → Aplicações que demandam recursos que a generalidade que um SGBD oferece para a definição e processamento de dados não suporta. Por exemplo, funções de segurança sofisticadas
- Custo proibitivo → Alto investimento inicial em hardware, software e treinamento

Seção 02

Conceitos e Arquitetura de SGBDs

Estrutura de BDs

- Conjunto de tipos, relacionamentos e restrições que se aplicam aos dados
- Abstração de dados → Provida pela abordagem SGBD
  - Diferentes usuários percebem a estrutura do banco de dados de acordo com diferentes níveis de detalhamento
  - O SGBD suprime detalhes de organização e armazenamento dos dados
  - Fornece recursos essenciais para a compreensão dos dados e de seus relacionamentos
- Modelo de dados → Oferece meios necessários para se alcançar a abstração.
- Modelos → Representações de objetos e eventos reais

Modelo de Dados

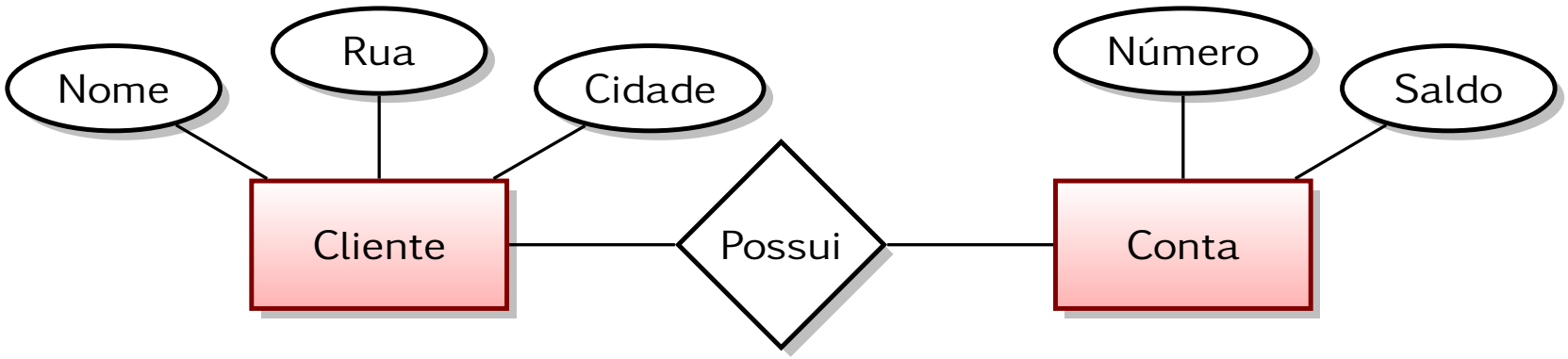
- Coleção de conceitos usados para descrever a estrutura do banco de dados
- Incorpora conjuntos de operações básicas para especificar atualização e recuperação de dados a partir do banco de dados:
  - Inserir, excluir, modificar ou recuperar
- Define o comportamento de uma determinada aplicação
- Modelagem de dados → Ato de construir modelos de dados, comumente utilizando linguagens textuais e gráficas

Modelos de Dados: Categorias

1. Conceituais → Alto nível de abstração
   - *Representam a estrutura do banco de dados como os usuários a percebem*
   - Conceitos → Entidades, Atributos e Relacionamentos
2. Representativos → Nível intermediário de abstração
   - Também conhecidos como modelos de implementação
   - Representam a estrutura do BD detalhando aspectos de implementação em um sistema computadorizado
   - Ocultam detalhes de armazenamento físico
   - Conceitos → Objetos e Relações...
3. Físicos → Baixo nível de abstração
   - Representam a estrutura do banco de dados detalhando aspectos de armazenamento físico em um sistema computadorizado
   - Conceitos → Arquivos, Registros, Índices...

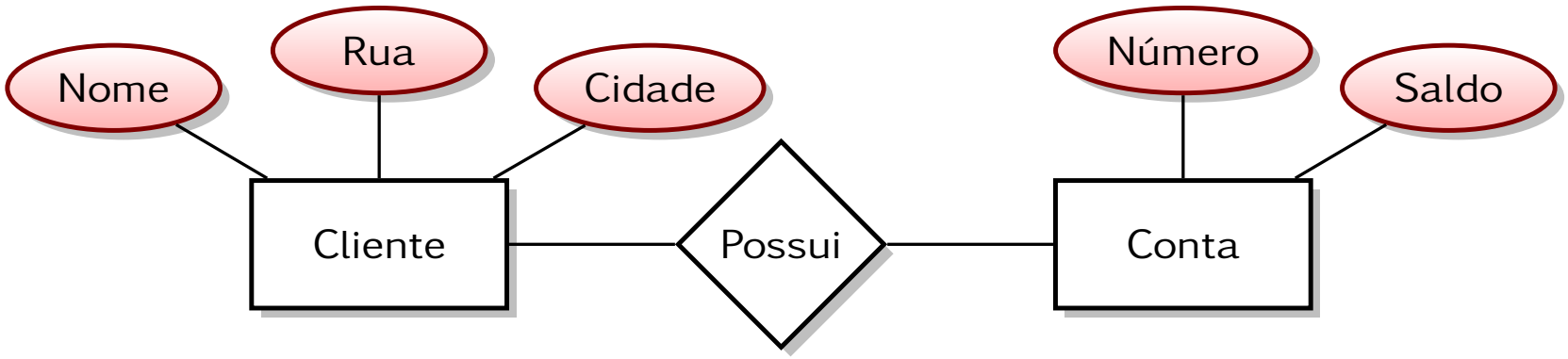
Modelos de Dados: Modelos Conceituais

- Entidade → Ente (objeto) do universo de discurso



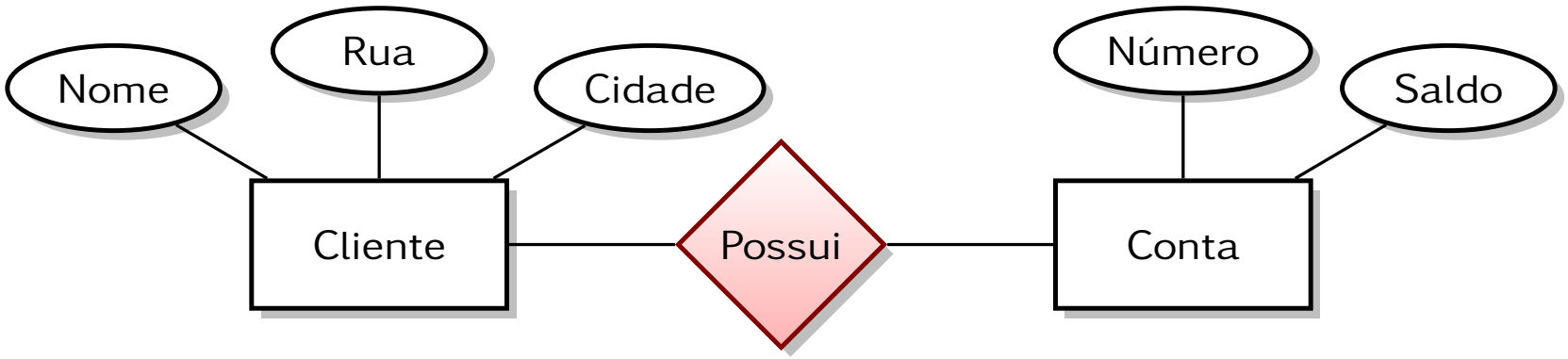
Modelos de Dados: Modelos Conceituais

- Atributo → Propriedade que caracteriza uma entidade



Modelos de Dados: Modelos Conceituais

- Relacionamento → Associação entre duas ou mais entidades



Modelos de Dados: Modelos Representativos

- Hierárquico
  - BD → Coleção de árvores formando uma floresta
  - Registro → Nó da árvore
  - Associação entre registros → Aresta da árvore
  - Um nó filho só pode ter um pai (*1:N*)
- Rede
  - Extensão do modelo hierárquico
  - Permitem associações *N:N*
  - Sistemas de navegação → Aplicações "atravessam" um conjunto de registros interligados
- Objeto
  - BD → Coleção de objetos
  - Registro → Objeto
  - Associação entre registros → Ligação
  - Próximos aos modelos de dados conceituais

Modelos de Dados: Modelos Representativos

- Relacional
  - BD → Coleção de relações (tabelas)
  - Registro → Tupla
  - Associação entre registros → Relacionamento
  - Definição teórica baseada na lógica de predicados e na teoria dos conjuntos
  - Modelo mais frequente adotado nos SGBDs comerciais baseados em transações (SGBDRs)
  - Consolidado, proporcionando alto desempenho na execução das operações básicas no BD

Modelos de Dados: Modelos Físicos

- Descrevem detalhes de armazenamento físico dos dados em memória:
  - Organização dos dados em arquivos armazenados em memória secundária
  - Formatos e ordenação de registros em arquivos
  - Indexação → Caminhos de acesso alternativos para recuperação rápida de registros

Esquema de BD

- Consiste na descrição (metadados) do banco de dados
- Especificado no projeto e não muda com frequência
- Existem convenções para se representar esquemas usando diagramas
- Diagrama de Esquema → Representação de um esquema
  - Representa aspectos do esquema, como os tipos de restrições e os nomes de tipos de registros e de itens de dados

EMPREGADO

CPF	Nome	Sexo	Salario	CPFSupervisor

- Construtor do esquema → Cada elemento que compõe o esquema. Por exemplo, EMPREGADO

Esquema de BD

- O diagrama apresenta a estrutura de cada tipo de registro, mas **NÃO** as instâncias dos registros

EMPREGADO

CPF	Nome	Sexo	Salario	CPFSupervisor
955	Roberto	M	2.000,00	
967	Amanda	F	3.000,00	955
983	Thales	M	1.000,00	983

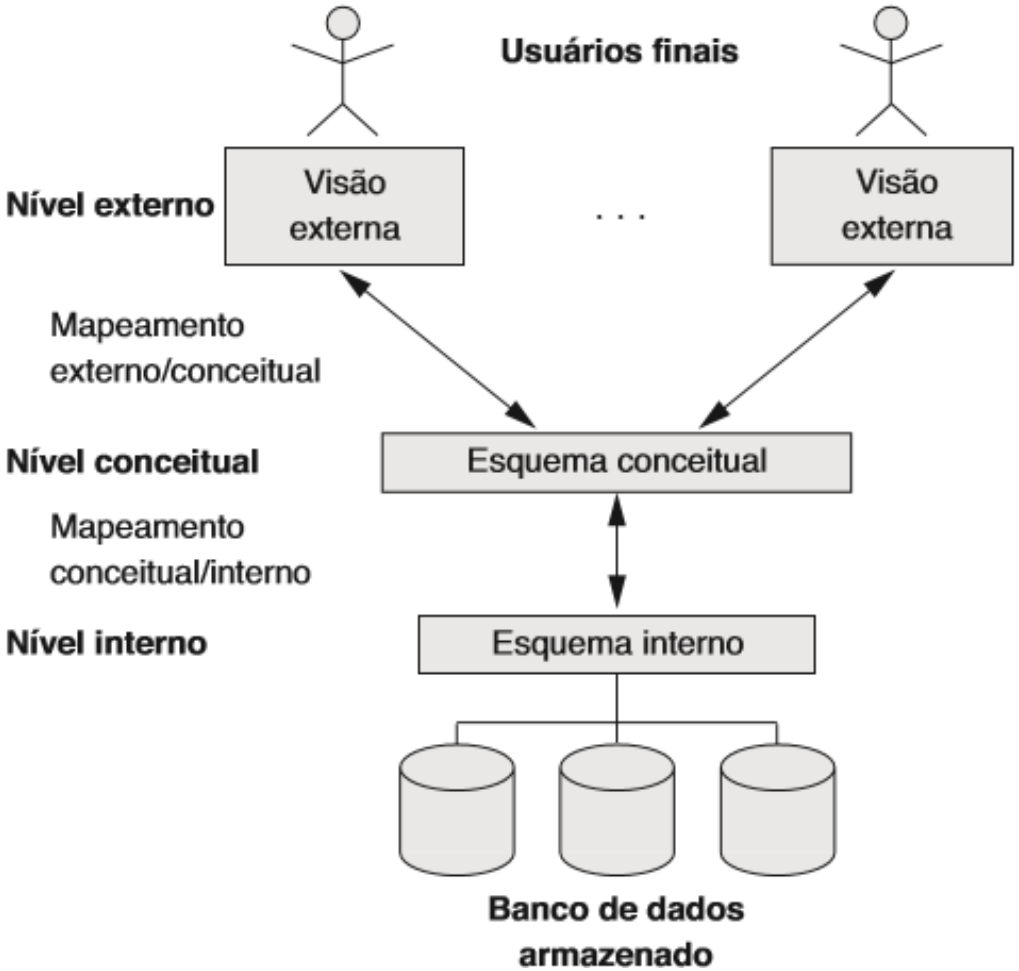
Instância de BD

- Conjunto de dados armazenados no banco de dados em determinado momento
- Estado Vazio → Esquema especificado, mas nenhum dado armazenado no BD
- Estado Inicial → BD é carregado (populado) com os dados inicias
- O estado do banco de dados é alterado ao se **inserir, excluir ou alterar** o valor de um item em um registro.

Arquitetura de Três Esquemas

- Abordagem na qual o usuário pode visualizar o esquema em diferentes níveis
- Visa alcançar:
  - Autodescrição
  - Independência entre dados e operações
  - Suporte a múltiplas visões do usuário

Arquitetura de Três Esquemas



Arquitetura de Três Esquemas

- Nível Externo
  - Esquemas externos → Visões de usuário
  - Cada visão descreve a parte do BD em que um grupo de usuários está interessado, ocultando o restante
  - Implementado utilizando um modelo de dados representativo
- Nível Conceitual
  - Esquemas conceituais → Descrevem a estrutura do BD para uma comunidade de usuários
  - Descrição de entidades, tipos de dados, relacionamentos, operações do usuário e restrições
  - Oculta detalhes de armazenamento físico
- Nível Interno
  - Esquemas físicos → Descrevem a estrutura do armazenamento físico do banco de dados
  - Detalhes do armazenamento de dados e dos caminhos de acesso para o BD

Arquitetura de Três Esquemas

- São apenas descrições dos dados, os dados armazenados estão apenas no nível físico
- A transformação de requisições e os resultados entre níveis são chamados de **mapeamentos**
- O SGBD transforma uma solicitação especificada por um grupo de usuários em uma solicitação no esquema conceitual e, em seguida, em uma solicitação no esquema interno para o processamento no banco de dados

Independência de Dados

- Capacidade de se alterar o esquema em um nível sem ter de alterar o esquema no nível adjacente mais elevado
- Independência lógica
  - Capacidade de alterar o esquema conceitual sem ter que alterar o esquema externo
  - Ao acrescentar ou remover um tipo de registro somente o mapeamento entre os níveis e a definição da visão são alterados
- Independência física
  - Capacidade de alterar o esquema interno sem ter que alterar o esquema conceitual
  - Ao organizar arquivos físicos criando estruturas de acesso adicionais somente o mapeamento entre os níveis é alterado

Independência de Dados

- A independência lógica de dados é mais difícil de ser alcançada porque permite alterações estruturais e de restrição sem afetar os programas de aplicação
- A arquitetura de três esquemas facilita a independência de dados
- Poucos SGBDs implementam a arquitetura completa de três esquemas por haver uma sobrecarga levando a baixa eficiência do SGBD

Linguagens em SGBDs

- O SGBD precisa oferecer linguagens e interfaces apropriadas para cada tipo de usuário
- Linguagem de Definição de Visão (VDL) → Especifica o esquema externo, as visões de usuário e seus mapeamentos ao esquema conceitual
- Linguagem de Definição de Dados (DDL) → Especifica o esquema conceitual
- Linguagem de Definição de Armazenamento (SDL) → Especifica o esquema interno
- Linguagem de Manipulação de Dados (DML) → Utilizada para manipulaccão de dados, tais como inserção, exclusão, modificação e recuperação de dados

Linguagens em SGBDs: DML

- Alto Nível → Não procedural
  - Especifica operações complexas de forma concisa
  - Podem recuperar muitos registros em uma única instrução
  - Declarativas → Especifica quais dados recuperar, em vez de como recuperá-los
  - Também denominadas de **linguagem de consulta** por poderem ser usadas de maneira interativa
- Baixo Nível → Procedural
  - Deve ser embutida em uma linguagem de programação de uso geral (**linguagem hospedeira**), sendo assim chamada de **sublinguagem de dados**
  - Recupera registros individuais ou objetos do banco de dados e os processa separadamente

Linguagens em SGBDs: SQL

- Nos SGBDs atuais as diferentes linguagens geralmente não são considerados linguagens distintas
- Linguagem de Consulta Estruturada (SQL) → Combinação de VDL, DDL e DML, bem como as instruções para especificação de restrição, evolução de esquema e outros recursos

Seção 04

SQL (Structured Query Language)

SQL

- ***Structured Query Language***: Linguagem de Consulta Estruturada.
- Oferece uma interface de linguagem *declarativa* de nível mais alto.
- O usuário apenas especifica *qual* deve ser o resultado, deixando a otimização real e as decisões sobre como executar a consulta para o SGBD.
- Linguagem padrão para SGBDs relacionais comerciais.

SQL

Linguagem abrangente de banco de dados:

- Contém instruções para definição de dados, consultas e atualizações.

Facilidades oferecidas:

- Definição de visões sobre o banco de dados;
- Especificação de segurança e autorizações;
- Definição de restrições de integridade;
- Especificação de controles de transações.

Definições e tipos de dados em SQL

Termos equivalentes ao modelo relacional:

- Relação → Tabela
- Tupla → Linha
- Atributo → Coluna

O principal comando para definição de dados é o **CREATE**, que pode ser usado para criar **esquemas, tabelas** e **domínios**.

Definições e tipos de dados em SQL

ESQUEMA

- Agrupa tabelas e outras construções que pertencem a mesma aplicação de banco de dados.

Identificado por:

- **Nome do esquema**;
- **Identificador de autorização** (indica o usuário proprietário do esquema);
- Descritores para cada **elemento** (*tabelas, restrições, views, domínios e outras construções*).

Criação de esquema EMPRESA que pertence ao usuário identificado por 'Jsilva':

```
CREATE SCHEMA EMPRESA AUTHORIZATION 'Jsilva';
```

Definições e tipos de dados em SQL

CATÁLOGO

- Coleção nomeada de esquemas em um ambiente SQL.
- Sempre contém o esquema **INFORMATION_SCHEMA** (*padrão*): oferece informações sobre todos os esquemas no catálogo e os descritores dos elementos.

Definições e tipos de dados em SQL

CREATE TABLE

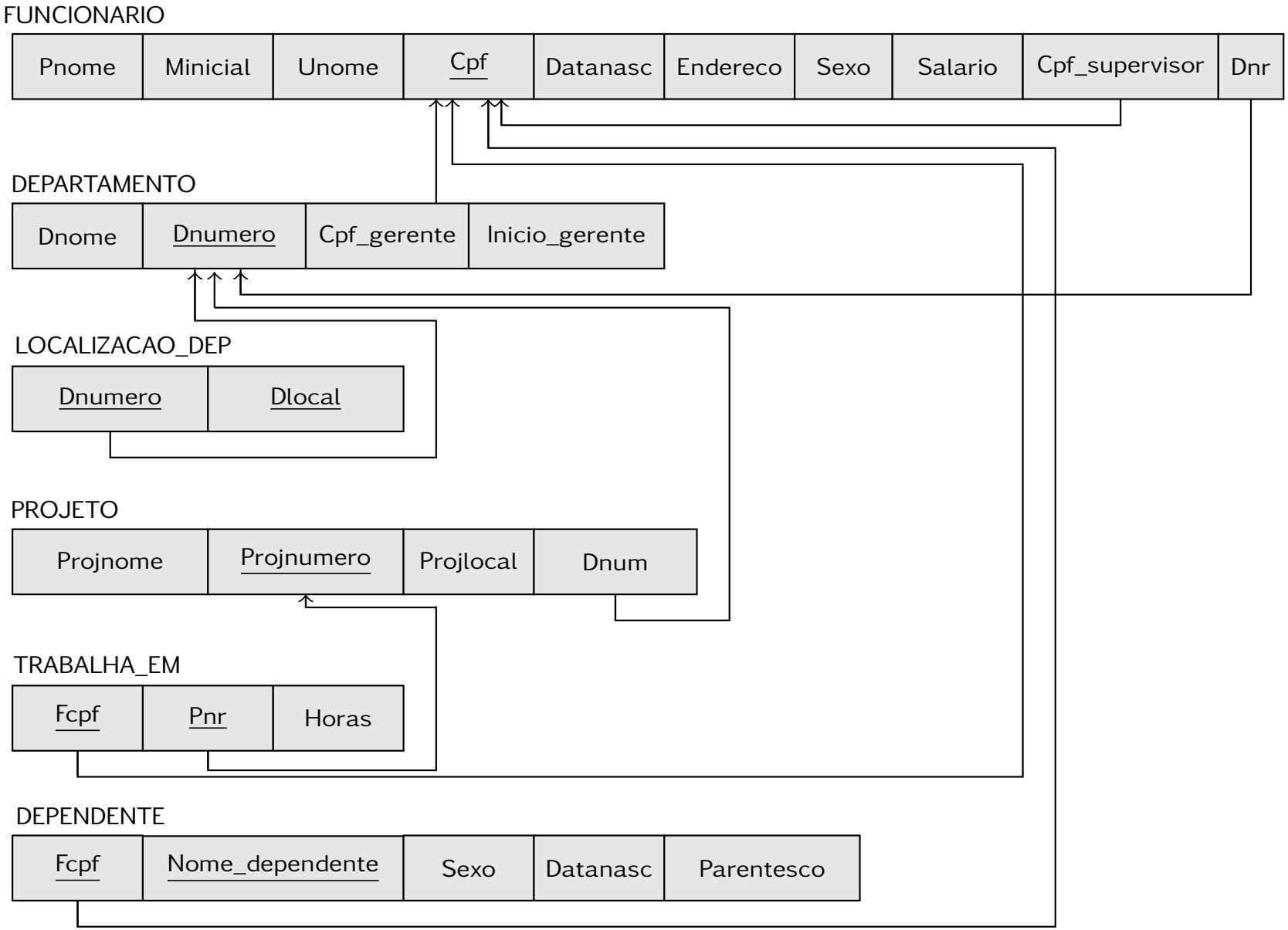
- Especifica uma nova *relação*, dando-lhe um *nome* e especificando seus *atributos e relações iniciais*.
- As restrições de **chave, integridade de entidade** ou **referencial** podem ser especificados com esta cláusula. Depois que os atributos forem criados, são definidos a partir do **ALTER TABLE.**

Exemplo:

```
CREATE TABLE FUNCIONARIO;
```

O esquema em que as relações são declaradas é especificado implicitamente no ambiente em que as instruções CREATE TABLE são executadas.

BD Relacional Exemplo



Definições e tipos de dados em SQL

TABELAS BASE

- Relações declaradas por meio das instruções CREATE TABLE.
- A relação e suas tuplas são realmente criadas e armazenadas como um arquivo pelo SGBD.

RELAÇÕES VIRTUAIS

- Relações declaradas por meio das instruções CREATE VIEW.
- A relação e suas tuplas podem ou não corresponder a um arquivo físico real.

Tipos de dados de atributos

NUMÉRICO

- Incluem números inteiros de vários tamanhos (INTEGER e SMALLINT) e números de ponto flutuante (reais) de várias posições (FLOAT/REAL e DOUBLE PRECISION).

CADEIA DE CARACTERES

- **Tamanho fixo** - CHAR(n), onde n é a quantidade de caracteres.
- **Tamanho variável** - VARCHAR(n), onde n é a quantidade máxima de caracteres.
- Para especificar um valor literal de cadeia de caracteres, este é colocado entre aspas simples e é *case sensitive*.

Tipos de dados de atributos

CADEIA DE BITS

- **Tamanho fixo** - BIT(n), onde n é a quantidade de bits.
- **Tamanho variável** - BIT VARYING(n), onde n é a quantidade máxima de bits.
- O valor padrão para n é 1.
- Os literais de cadeia de bits literais são colocados entre apóstrofos, mas precedidos por um B para distingui-los das cadeias de caracteres. Por exemplo: **B'10101'**.

BOOLEANO

- Valores tradicionais TRUE (verdadeiro) ou FALSE (falso).

Tipos de dados de atributos

DATE

- Possui dez posições e seus componentes são DAY (dia), MONTH (mês) e YEAR (ano) na forma **DD-MM-YYYY**.
- **Tamanho variável** - BIT VARYING(n), onde n é a quantidade máxima de bits.
- O tipo **TIME** tem ao menos oito posições, com os componentes HOUR (hora), MINUTE (minuto) e SECOND (segundo) na forma **HH:MM:SS**.
- Somente datas e horas válidas devem ser permitidas pela implementação SQL.

Tipos de dados de atributos

TIMESTAMP

- Inclui os campos DATE e TIME, mais um mínimo de seis posições para frações decimais de segundos e um qualificador opcional WITH TIME ZONE.

- Valores literais são representados por cadeias entre apóstrofos precedidos pela palavra-chave TIMESTAMP na forma: **TIMESTAMP '27-09-2008 09:12:47.648302'**.

Restrições em SQL

- Restrições básicas podem ser especificadas em SQL como parte da criação de tabela.
- Incluem **restrições de chave e integridade referencial, restrições sobre domínios de atributos e NULLs e restrições sobre tuplas individuais dentro de uma relação.**

Restrições em SQL

Restrição NOT NULL

- Pode ser especificada se o valor NULL não for permitido para determinado atributo.
- Sempre especificado de maneira implícita para os atributos que fazem parte da chave primária.

Exemplo:

```
CREATE TABLE FUNCIONARIO
(Pnome VARCHAR(15) NOT NULL,
 Unome VARCHAR(15) NOT NULL,
 Cpf CHAR(11), NOT NULL,
 Cpf_supervisor CHAR(11), NOT NULL,
PRIMARY KEY (Cpf));
```

Restrição DEFAULT

- Define um valor padrão para um atributo em sua definição.
- O valor padrão será incluído em qualquer nova tupla se um valor explícito não for fornecido para esse atributo.
- Se nenhuma cláusula default for especificada, o *valor padrão* será NULL para atributos *que não possuem* a restrição NOT NULL.

Restrições em SQL

Restrição CHECK

- Limita os valores de atributo ou domínio após sua definição.

Exemplo:

Suponha que os números de departamento sejam restritos a números inteiros entre 1 e 20; então, podemos mudar a declaração de atributo de **Dnumero** na tabela DEPARTAMENTO para o seguinte:

```
DNUMERO INT NOT NULL CKECK
(DNUMERO > 0 AND Dnumero < 21);
```

Restrições em SQL

Restrição PRIMARY KEY

- Especifica um ou mais atributos que compõem a chave primária de uma relação.
- Se a chave primária tiver apenas um atributo, a cláusula pode acompanhar o atributo diretamente.

Exemplo:

```
CREATE TABLE FUNCIONARIO
(Cpf CHAR(11), NOT NULL,
PRIMARY KEY (Cpf));
```

Restrição UNIQUE

- Especifica chaves alternativas (secundárias).
- Pode ser especificada diretamente para uma chave secundária se esta for um único atributo, como exemplo:

```
CREATE TABLE DEPARTAMENTO
(Dnome VARCHAR(15) NOT NULL,
 Dnumero INT NOT NULL,
PRIMARY KEY (Dnumero),
UNIQUE (Dnome));
```

Restrições em SQL

Restrição FOREIGN KEY

- Integridade referencial - chave estrangeira.

Exemplo:

```
CREATE TABLE LOCALIZACAO_DEP
(Dnumero INT NOT NULL,
 Dlocal VARCHAR(15) NOT NULL,
PRIMARY KEY (Dnumero, Dlocal),
FOREIGN KEY (Dnumero) REFERENCES DEPARTAMENTO(Dnumero));
```

- Uma restrição de integridade pode ser violada quando tuplas são inseridas ou excluídas, ou quando um valor de atributo de chave estrangeira ou chave primária é modificado.
- A ação default para evitar a violação da integridade é **rejeitar** a operação de atualização - opção RESTRICT.

Restrição FOREIGN KEY

- Ação de disparo referencial - definido pelo projetista de banco de dados. Especifica uma ação alternativa para os casos de violação de integridade:

Opções: **SET NULL, CASCADE** e **SET DEFAULT**

As opções acima devem ser escolhidas com **ON DELETE** (remoção) ou **ON UPDATE** (atualização).

Restrição FOREIGN KEY - Exemplo

- Se o projetista escolhe ON DELETE SET NULL e ON UPDATE CASCADE para a chave estrangeira Cpf_supervisor de FUNCIONARIO:

```
FOREIGN KEY (Cpf_supervisor) REFERENCES FUNCIONARIO(Cpf)
ON DELETE SET NULL ON UPDTE CASCADE);
```

*Se a tupla para um **funcionário superior é excluída**, o valor de Cpf_supervisor será automaticamente definido como NULL para todas as tuplas de funcionários que estavam referenciando a tupla do funcionário excluído.*

*Se o valor de Cpf para um funcionário supervisor é **atualizado**, o novo valor será **propagado em cascata** de Cpf_superior para todas as tuplas de funcionário que referencia a tupla de funcionário atualizada.*

Definição de nome a restrições

- Uma restrição pode receber um **nome de restrição**, seguindo a palavra-chave **CONSTRAINT**;
- Os nomes de todas as restrições de um esquema precisam ser exclusivos/únicos.

Exemplo:

```
CONSTRAINT CHPFUNC
  PRIMARY KEY (Cpf);

CONSTRAINT CHESUPERFUNC
  FOREIGN KEY (Cpf_supervisor) REFERENCES FUNCIONARIO(Cpf)
  ON DELETE SET NULL ON UPDTE CASCADE);
```

Recuperação de dados em SQL

- O **SELECT** é uma instrução básica para recuperar informações de banco de dados.
- A seguir, apresentaremos os recursos da SQL para *consultas de recuperação simples*.

Estrutura SELECT-FROM-WHERE

- A forma básica do comando SELECT, também chamado do **mapeamento** ou **bloco select-from-where**, é composta por três cláusulas, no seguinte formato:

SELECT <lista atributos>

FROM <lista tabelas>

WHERE <condição>;

onde:

- **<lista atributos>** é uma lista de nomes de atributo cujos valores devem ser recuperados pela consulta.
- **<lista tabelas>** é uma lista dos nomes de relação exigidos para processar a consulta.
- **<condição>** é uma expressão condicional (booleana) que identifica as tuplas a serem recuperadas pela consulta.

Estrutura SELECT-FROM-WHERE

- Os operadores básicos de comparação lógicos para comparar valores de atributo entre si e com constantes literais são =, <, <=, >, >= e <>.

Estrutura SELECT-FROM-WHERE

EXEMPLO 01

- Recuperar a data de nascimento e o endereço do(s) funcionário(s) cujo nome seja 'João B. Silva':

```
SELECT Datanasc, Endereco
FROM   Funcionario
WHERE  Pnome = 'Joao'
       AND Minicial = 'B'
       AND Unome = 'Silva';
```

Resultado da consulta:

Datanasc	Endereco
09-01-1965	Rua das Flores, 751, São Paulo, SP

Estrutura SELECT-FROM-WHERE

EXEMPLO 02

- Recuperar o nome e o endereço de todos os funcionários que trabalham para o departamento 'Pesquisa':

```
SELECT Pnome, Unome, Endereco
FROM   Funcionario, Departamento
WHERE  Dnome = 'Pesquisa' AND Dnumero = Dnr;
```

Resultado da consulta:

Pnome	Unome	Endereco
João	Silva	Rua das Flores, 751, São Paulo, SP
Fernando	Wong	Rua da Lapa, 34, São Paulo, SP
Ronaldo	Lima	Rua Rebouças, 65, Piracicaba, SP
Joice	Leite	Av. Lucas Obes, 74, São Paulo, SP

Estrutura SELECT-FROM-WHERE

EXEMPLO 03

- Para cada projeto localizado em 'Mauá', liste o número do projeto, o número do departamento que o controla e o sobrenome, endereço e data de nascimento do gerente do departamento:

```
SELECT Projnumero, Dnum, Unome, Endereco, Datanasc
FROM   Projeto, Departamento, Funcionario
WHERE  Dnum = Dnumero
       AND Cpf_gerente = Cpf
       AND Projlocal = 'Maua'
```

Resultado da consulta:

Projnumero	Dnum	Unome	Endereco	Datanasc
10	4	Souza	Av. Artur de Lima, 54, Santo André, SP	20-06-1941
30	4	Souza	Av. Artur de Lima, 54, Santo André, SP	20-06-1941

Ambiguidade de nomes de atributos

- O mesmo nome pode ser usado para dois (ou mais) atributos, desde que estes estejam em *relações diferentes*;

- É preciso *qualificar* o nome do atributo com o nome da relação para evitar ambiguidade - isso é feito prefixando o nome da relação (ou apelido) ao nome do atributo.

Ambiguidade de nomes de atributos

EXEMPLO 04

- Para cada funcionário, recupere o primeiro e o último nome do funcionário e o primeiro e último nome de seu supervisor imediato:

```
SELECT F.Pnome, F.Unome, S.Pnome, S.Unome
FROM   Funcionario AS F,
       Funcionario AS S
WHERE  F.Cpf_supervisor = S.Cpf;
```

Cláusula WHERE não especificada

- A falta de uma cláusula WHERE indica que **não há condições sobre a seleção das tuplas**.
- Todas as tuplas da relação especificada na cláusula FROM são selecionadas para o resultado da consulta.
- Se mais de uma relação for especificada e não houver uma cláusula WHERE, serão selecionadas todas as combinações possíveis entre as tuplas.

Uso de asteriscos

- Usado para recuperar todos os atributos de todas as tuplas selecionadas, não precisando listar seus nomes explicitamente.

EXEMPLO 05

- Recuperar todos os valores de atributo de qualquer FUNCIONARIO que trabalha no DEPARTAMENTO número 5:

```
SELECT *
FROM   Funcionario
WHERE  Dnr = 5
```

Tabelas como conjuntos

- As tabelas não são tratadas como um conjunto, mas como um **multiconjunto**.
- *Tuplas duplicadas podem aparecer mais de uma vez* em uma tabela e no resultado de uma consulta. A SQL não elimina automaticamente tuplas duplicadas nos resultados das consultas.
- Para eliminar as tuplas duplicadas no resultado da consulta, utiliza-se a palavra-chave **DISTINCT** na cláusula **SELECT**.

Apenas as tuplas distintas deverão permanecer no resultado.

Tabelas como conjuntos

EXEMPLO 06

- Recuperar o valor do salário de cada funcionário (sem tratamento de valores duplicados):

```
SELECT Salario
FROM   Funcionario
```

- Resultado da consulta:

Salário
30.000
40.000
25.000
43.000
38.000
25.000
25.000
55.000

EXEMPLO 06

- Recuperar o valor do salário de cada funcionário (com tratamento de valores duplicados):

```
SELECT DISTINCT Salario
FROM   Funcionario
```

- Resultado da consulta:

Salário
30.000
40.000
25.000
43.000
38.000
55.000

Comparação de subcadeias

Recursos adicionais SQL:

1. **LIKE**
   - Condição de comparação apenas sobre partes de uma cadeia de caracteres;
   - Cadeias parciais são especificadas usando dois caracteres reservados: **%** substitui um número qualquer, de zero ou mais caracteres, e o sublinhado (_) substitui um único caracter.

EXEMPLO 07

- Recuperar todos os funcionários cujo endereço esteja em São Paulo, SP:

```
SELECT Pnome, Unome
FROM   Funcionario
WHERE  Endereco LIKE '%SaoPaulo,SP%';
```

Comparação de subcadeias

Recursos adicionais SQL:

2 **BETWEEN** *(entre)*

EXEMPLO 08

- Recuperar todos os funcionários no departamento 5 cujo salário esteja entre R$30.000 e R$40.000:

```
SELECT *
FROM   Funcionario
WHERE  (Salario BETWEEN 30.000 AND 40.000)
       AND Dnr = 5;
```

Ordenação de resultados

- A SQL permite que o usuário ordene as tuplas do resultado de uma consulta pelos valores de um ou mais atributos que aparecem.
- Cláusula **ORDER BY**.
- A ordem default é a crescente de valores.
- Pode-se especificar a palavra-chave **DESC** caso queira exibir o resultado em ordem descrescente de valores.

Ordenação de resultados

EXEMPLO 09

- Recuperar uma lista dos funcionários e dos projetos em que estão trabalhando, ordenada por departamento e, dentro de cada departamento, ordenada alfabeticamente pelo sobrenome, depois pelo nome.

```
SELECT   D.nome, F.Unome, F.Pnome, P.Projnome
FROM     Departamento D, Funcionario F,
         Trabalha_em T, Projeto P
WHERE    D.numero = F.Dnr
         AND F.Cpf = T.Fcpf
         AND T.Pnr = P.Projnumero
ORDER BY D.Dnome, F.Unome, F.Pnome;
```

Comparações com NULL

Significados de **NULL**:

- Valor desconhecido;
- Valor indisponível ou retido;
- Atributo não aplicável.

A SQL permite consultas que verificam se o valor de um atributo é NULL.

- **IS** ou **IS NOT NULL**

Comparações com NULL

Exemplo: Recuperar os nomes de todos os funcionários que não possuem supervisores.

```
SELECT Pnome, Unome
FROM FUNCIONARIO
WHERE Cpf_supervisor IS NULL;
```

Consultas aninhadas

- São blocos **SELECT-FROM-WHERE** completos dentro da cláusula WHERE de outra consulta.
- Consulta externa.

Operador de comparação IN

- Compara um valor *v* com um conjunto (ou multiconjunto) de valores *V*.
- Avalia como TRUE se *v* for um dos elementos em *V*.

```
SELECT DISTINCT Fcpf
FROM TRABALHA_EM
WHERE (Pnr, Horas) IN (SELECT Pnr, Horas
                       FROM TRABALHA_EM
                       WHERE Fcpf = '12345678966');
```

Função EXISTS

Operador de comparação IN

- Verificar se o resultado de uma consulta aninhada correlacionada é vazio ou não.

Operador de comparação IN

- Costumam ser usados em conjunto com uma consulta aninhada correlacionada.

Exemplo: Recuperar os nomes de funcionários que não possuem dependentes.

```
SELECT Pnome, Unome
FROM FUNCIONARIO
WHERE NOT EXISTS (SELECT *
                  FROM DEPENDENTE
                  WHERE Cpf = Fcpf);
```

Funções de agregação

Utilizadas para resumir informações de várias tuplas em uma síntese de tupla única.

Funções de agregação embutidas:

- **COUNT**, **SUM**, **MAX**, **MIN** e **AVG**

Essas funções podem ser usadas na cláusula **SELECT** ou em uma cláusula **HAVING**.

Valores NULL são descartados quando as funções de agregação são aplicadas a um atributo.

Funções de agregação

Exemplo: Achar a soma dos salários de todos os funcionários do departamento 'Pesquisa', bem como o salário máximo, o salário mínimo e a média dos salários nesse departamento.

```
SELECT SUM (Salario), MAX (Salario),
       MIN (Salario), AVG (Salario)
FROM (FUNCIONARIO JOIN DEPARTAMENTO ON Dnr = Dnumero)
WHERE Dnome = 'Pesquisa';
```

Funções de agregação

Exemplo: Recuperar o número de funcionários no departamento 'Pesquisa'.

```
SELECT COUNT(*)
FROM FUNCIONARIO, DEPARTAMENTO
WHERE Dnr = Dnumero AND Dnome = 'Pesquisa';
```

Exemplo: Contar o número de valores de salário distintos no BD.

```
SELECT COUNT (DISTINCT Salario)
FROM FUNCIONARIO;
```

Cláusulas GROUP BY e HAVING

Particionam a relação em subconjuntos de tuplas, com base no atributo(s) de agrupamento.

Cláusula GROUP BY

- Especifica os atributos de agrupamento.
- Se há NULLs no atributo de agrupamento, um grupo separado é criado para todas as tuplas que satisfazem tal condição.

Cláusula HAVING

- Oferece uma condição sobre a informação de resumo.

Cláusulas GROUP BY e HAVING

Exemplo: Para cada departamento, recuperar o número do departamento, o número de funcionários no departamento e seu salário médio.

```
SELECT Dnr, COUNT (*), AVG (Salario)
FROM FUNCIONARIO
GROUP BY Dnr;
```

Cláusulas GROUP BY e HAVING

Exemplo: Para cada projeto em que mais de dois funcionários trabalham, recupere o número e o nome do projeto e o número de funcionários que trabalham no projeto.

```
SELECT Projnumero, Projnome, COUNT (*)
FROM PROJETO, TRABALHA_EM
WHERE Projnumero=Pnr
GROUP BY Projnumero, Projnome
HAVING COUNT (*) > 2
```

Cláusulas GROUP BY e HAVING

Em resumo:

```
SELECT
FROM
[WHERE <condicao>]
[GROUP BY <atributo(s) de agrupamento>]
[HAVING <condicao de grupo>]
[ORDER BY <lista de atributos>];
```

O comando INSERT

- É usado para acrescentar uma única tupla a uma relação.
- É necessário especificar o nome da relação e uma lista de valores para a tupla.
- Valores devem ser listados na mesma ordem em que os atributos correspondentes são especificados na relação.

O comando INSERT

- Acrescentar uma nova tupla a relação **FUNCIONARIO**.

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Endereco	Sexo	Salario	Cpf_supervisor	Dnr

```
INSERT INTO FUNCIONARIO
VALUES      ( 'Ricardo' , 'K', 'Marini',
            '65329865388', '30-12-
            1962', 'Rua Itapira, 44,
            Santos, SP', 'M', 37.000,
            '65329865388', 4 );
```

O comando INSERT

- Também é possível que o usuário especifique nomes de atributos que correspondem aos valores fornecidos no comando INSERT.
- É necessário especificar o nome da relação e uma lista de valores para a tupla.
- *Inserir uma tupla para um novo **FUNCIONARIO** do qual conhecemos apenas os atributos **Pnome**, **Unome**, **Dnr** e **Cpf**.*

```
INSERT INTO  FUNCIONARIO (Pnome, Unome,
             Dnr, Cpf)
VALUES       ( 'Ricardo', 'Marini', 4,
             '65329865388' );
```

O comando INSERT

- Atributos não especificados são definidos como seu valor DEFAULT ou NULL, os valores são listados na mesma ordem que os atributos são listados no próprio comando INSERT.

O comando INSERT

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Sexo	Salario	Cpf_supervisor	Dnr
João	B	Silva	12345678966	09-01-1965	M	30.000	33344555587	5
Fernando	T	Wong	33344555587	08-12-1955	M	40.000	88866555576	5
Alice	J	Zelaya	99988777767	19-01-1968	F	25.000	98765432168	4
Jennifer	S	Souza	98765432168	20-06-1941	F	43.000	88866555576	4
Ronaldo	K	Lima	66688444476	15-09-1962	M	38.000	33344555587	5
Joice	A	Leite	45345345376	31-07-1972	F	25.000	33344555587	5
André	V	Pereira	98798798733	29-03-1969	M	25.000	98765432168	4
Jorge	E	Brito	88866555576	10-11-1937	M	55.000	NULL	1

Não existe uma tupla **DEPARTAMENTO** no banco de dados com **Dnumero = 2**.

```
INSERT INTO  FUNCIONARIO (Pnome, Unome,
             Cpf, Dnr)
VALUES       ( 'Roberto', 'Gomes',
             '98076054011', 2 );
```

Operação rejeitada

O comando INSERT

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Sexo	Salario	Cpf_supervisor	Dnr
João	B	Silva	12345678966	09-01-1965	M	30.000	33344555587	5
Fernando	T	Wong	33344555587	08-12-1955	M	40.000	88866555576	5
Alice	J	Zelaya	99988777767	19-01-1968	F	25.000	98765432168	4
Jennifer	S	Souza	98765432168	20-06-1941	F	43.000	88866555576	4
Ronaldo	K	Lima	66688444476	15-09-1962	M	38.000	33344555587	5
Joice	A	Leite	45345345376	31-07-1972	F	25.000	33344555587	5
André	V	Pereira	98798798733	29-03-1969	M	25.000	98765432168	4
Jorge	E	Brito	88866555576	10-11-1937	M	55.000	NULL	1

Nenhum valor de ***Cpf*** é fornecido e essa é a chave primária, que não pode ser NULL.

```
INSERT INTO  FUNCIONARIO (Pnome, Unome,
             Dnr)
VALUES       ( 'Roberto', 'Gomes', 5 );
```

Operação rejeitada

O comando INSERT

- *Criar uma tabela temporária que possui o sobrenome do funcionário, o nome do projeto e as horas por semana para cada funcionário que trabalha em um projeto.*

```
CREATE TABLE   TRABALHA_EM_INFO
( Func_nome VARCHAR(15),
( Proj_nome VARCHAR(15),
  Horas_semanal DECIMAL(3, 1) );
INSERT INTO    TRABALHA_EM_INFO
               ( Func_nome, Proj_nome,
               Horas_por_semana)
SELECT         F.Unome, P.Projnome, T.Horas
FROM           PROJETO P, TRABALHA_EM T,
               FUNCIONARIO F
WHERE          P.Projnumero = T.Pnr AND T.Fcpf = F.Cpf;
```

O comando DELETE

- Remove tuplas de uma relação.
- As tuplas são explicitamente excluídas de apenas uma tabela por vez.
- No entanto, a exclusão pode se propagar para as tuplas em outras relações, de acordo com as restrições de integridade.

O comando DELETE

- Uma cláusula WHERE inexistente especifica que todas as tuplas na relação deverão ser excluídas;
- A tabela permanece no banco de dados como uma tabela vazia.

O comando DELETE

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Sexo	Salario	Cpf_supervisor	Dnr
João	B	Silva	12345678966	09-01-1965	M	30.000	33344555587	5
Fernando	T	Wong	33344555587	08-12-1955	M	40.000	88866555576	5
Alice	J	Zelaya	99988777767	19-01-1968	F	25.000	98765432168	4
Jennifer	S	Souza	98765432168	20-06-1941	F	43.000	88866555576	4
Ronaldo	K	Lima	66688444476	15-09-1962	M	38.000	33344555587	5
Joice	A	Leite	45345345376	31-07-1972	F	25.000	33344555587	5
André	V	Pereira	98798798733	29-03-1969	M	25.000	98765432168	4
Jorge	E	Brito	88866555576	10-11-1937	M	55.000	NULL	1

```
DELETE FROM  FUNCIONARIO
WHERE        Unome = 'Braga';
```

Excluirá zero tuplas

O comando DELETE

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Sexo	Salario	Cpf_supervisor	Dnr
João	B	Silva	12345678966	09-01-1965	M	30.000	33344555587	5
Fernando	T	Wong	33344555587	08-12-1955	M	40.000	88866555576	5
Alice	J	Zelaya	99988777767	19-01-1968	F	25.000	98765432168	4
Jennifer	S	Souza	98765432168	20-06-1941	F	43.000	88866555576	4
Ronaldo	K	Lima	66688444476	15-09-1962	M	38.000	33344555587	5
Joice	A	Leite	45345345376	31-07-1972	F	25.000	33344555587	5
André	V	Pereira	98798798733	29-03-1969	M	25.000	98765432168	4
Jorge	E	Brito	88866555576	10-11-1937	M	55.000	NULL	1

```
DELETE FROM  FUNCIONARIO;
WHERE        Cpf = '12345678966';
```

Excluirá uma tupla

O comando DELETE

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Sexo	Salario	Cpf_supervisor	Dnr
João	B	Silva	12345678966	09-01-1965	M	30.000	33344555587	5
Fernando	T	Wong	33344555587	08-12-1955	M	40.000	88866555576	5
Alice	J	Zelaya	99988777767	19-01-1968	F	25.000	98765432168	4
Jennifer	S	Souza	98765432168	20-06-1941	F	43.000	88866555576	4
Ronaldo	K	Lima	66688444476	15-09-1962	M	38.000	33344555587	5
Joice	A	Leite	45345345376	31-07-1972	F	25.000	33344555587	5
André	V	Pereira	98798798733	29-03-1969	M	25.000	98765432168	4
Jorge	E	Brito	88866555576	10-11-1937	M	55.000	NULL	1

```
DELETE FROM  FUNCIONARIO;
WHERE        Dnr = 5;
```

Excluirá quatro tuplas

O comando DELETE

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Sexo	Salario	Cpf_supervisor	Dnr
João	B	Silva	12345678966	09-01-1965	M	30.000	33344555587	5
Fernando	T	Wong	33344555587	08-12-1955	M	40.000	88866555576	5
Alice	J	Zelaya	99988777767	19-01-1968	F	25.000	98765432168	4
Jennifer	S	Souza	98765432168	20-06-1941	F	43.000	88866555576	4
Ronaldo	K	Lima	66688444476	15-09-1962	M	38.000	33344555587	5
Joice	A	Leite	45345345376	31-07-1972	F	25.000	33344555587	5
André	V	Pereira	98798798733	29-03-1969	M	25.000	98765432168	4
Jorge	E	Brito	88866555576	10-11-1937	M	55.000	NULL	1

```
DELETE FROM  FUNCIONARIO;
```

Excluirá todas tuplas

O comando UPDATE

- Usado para modificar valores de atributo de uma ou mais tuplas selecionadas.
- Cada comando UPDATE refere-se explicitamente a apenas uma única relação.
- A atualização de uma chave primária pode ser propagada para os valores de chave estrangeira das tuplas em outras relações de acordo com as restrições de integridade.
- Uma cláusula SET adicional no comando UPDATE especifica os atributos a serem modificados e seus novos valores.

O comando UPDATE

- *Alterar o local e o número de departamento que controla o número de projeto 10 para 'Santo André' e 5, respectivamente.*

```
UPDATE  PROJETO
SET     Projlocal = 'Santo Andre', Dnum = 5
WHERE   Projnumero = 10;
```

O comando UPDATE

- *Dar a todos os funcionários no departamento 'Pesquisa' um aumento de 10 por cento no salário.*

```
UPDATE   FUNCIONARIO
SET      Salario = Salario * 1,1
WHERE    Dnr = 5;
```

Seção 06

Álgebra Relacional

Álgebra Relacional: Introdução

- Conjunto básico de operações para manipular o banco de dados (modelo relacional).
- Essas operações permitem que um usuário especifique solicitações de recuperações básicas.
- O resultado da consulta (sequência de operações) é uma **nova relação**.

Álgebra Relacional: Introdução

- A álgebra oferece um importante alicerce formal para as operações do modelo relacional.
- Historicamente, a álgebra foi desenvolvida *antes* da linguagem SQL e seus conceitos são incorporados.
- Linguagem formal para o modelo relacional:

  Álgebra Relacional

- Linguagem prática para o modelo relacional:

  SQL, DDL, DML

Álgebra Relacional: Introdução

As operações da álgebra relacional podem ser divididas em *dois grupos*:

Operações da teoria do conjunto da matemática:

- União
- Intersecção
- Diferença
- Produto

Operações exclusivas para banco de dados:

- Seleção
- Projeção
- Junção

Álgebra Relacional - Introdução

Outras classificações:

- **Operações unárias**: aplicadas em relações isoladas.
- **Operações binárias**: duas tabelas combinando tuplas baseadas em condições.
- **Funções de agregação**: operações que resumem dados das tabelas.

A operação SELEÇÃO

$$\sigma_{condicional}(R)$$

- O operador **SELEÇÃO** é unário: ele é aplicado a uma única relação.
- Seleciona todas as tuplas que satisfazem a condição de seleção de uma relação **R**.
- Funciona como um ***filtro*** que mantém apenas as tuplas que *satisfazem uma condição*.
- As tuplas que não satisfazem a condição são *descartadas*.

A operação SELEÇÃO

- O símbolo *sigma* (σ) é usado para indicar o operador seleção.
- A *relação resultante* da operação seleção tem os mesmos atributos de **R**.
- O número de tuplas na relação resultante é sempre menor ou igual ao número de tuplas em **R**.

A operação SELEÇÃO

Exemplos:

- Selecionar tuplas de funcionário cujo número do departamento seja igual a 4:

$$\sigma_{Dnr\ =\ 4}(FUNCIONARIO)$$

- Selecionar tuplas de funcionário cujo salário seja maior que 30.000,00:

$$\sigma_{Salario\ >\ 30.000,00}(FUNCIONARIO)$$

A operação SELEÇÃO

A condição de seleção é uma **expressão booleana**, conforme:

<nome atributo> <op comparação> <valor constante>

<nome atributo> <op comparação> <nome atributo>

onde:

- ***<nome atributo>*** é o nome de um atributo de R;
- ***<op comparação>*** é um dos seguintes operadores:

$$= \quad < \quad \leq \quad > \quad \geq \quad \neq$$

- ***<valor constante>*** valor constante do domínio do atributo.

As cláusulas podem ser conectadas pelos operadores booleanos ***and***, ***or*** e ***not***.

A operação SELEÇÃO

Selecionar as tuplas para todos os funcionários que ou trabalham no departamento 4 e ganham mais de 25.000,00 por ano, ou trabalham no departamento 5 e ganham mais de 30.000,00.

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Salario	Cpf_supervisor	Dnr
João	B	Silva	12345678966	30.000	33344555587	5
Fernando	T	Wong	33344555587	40.000	88866555576	5
Alice	J	Zelaya	99988777767	25.000	98765432168	4
Jennifer	S	Souza	98765432168	43.000	88866555576	4
Joice	A	Leite	45345345376	25.000	33344555587	5
Jorge	E	Brito	88866555576	55.000	null	1

A operação SELEÇÃO

Selecionar as tuplas para todos os funcionários que ou trabalham no departamento 4 e ganham mais de 25.000,00 por ano, ou trabalham no departamento 5 e ganham mais de 30.000,00.

- ***Especificação da operação SELEÇÃO:***

$$\sigma_{(Dnr = 4 \wedge Salario > 25.000) \vee (Dnr = 5 \wedge Salario > 30.000)}(FUNCIONARIO)$$

- ***Resultado da operação:***

Pnome	Minicial	Unome	Cpf	Salario	Cpf_supervisor	Dnr
Fernando	T	Wong	33344555587	40.000	88866555576	5
Jennifer	S	Souza	98765432168	43.000	88866555576	4

A operação SELEÇÃO

- A seleção é aplicada a cada tupla individualmente, portanto, as condições de seleção não podem envolver mais de uma tupla.
- A operação SELECT é **comutativa**, logo pode ser aplicada em qualquer ordem:

$$\sigma_{cond1}(\sigma_{cond2}(R)) = \sigma_{cond2}(\sigma_{cond1}(R))$$

- Pode-se combinar uma sequência de operações SELEÇÃO a uma única operação SELEÇÃO com uma condição conjuntiva (**AND**):

$$\sigma_{cond1}(\sigma_{cond2}(\ldots(\sigma_{cond1}(R))\ldots)) = \sigma_{cond1 \ \wedge \ cond2 \ \wedge \ \ldots \ \wedge \ condn}(R)$$

A operação PROJEÇÃO

- Produz uma nova relação com apenas alguns atributos de **R**, e remove tuplas duplicadas.
- Seleciona certas colunas da tabela e descarta outras, projetando a relação apenas por alguns atributos.
- O operador PROJEÇÃO é **unário**: ele é aplicado a uma única relação.
- A forma geral da projeção é:

$$\pi_{lista\ de\ atributos}(R)$$

A operação PROJEÇÃO

$$\pi_{lista\ de\ atributos}(R)$$

- **Pi** (π) é o símbolo que representa a **PROJEÇÃO**;

  <lista de atributos> se refere aos atributos desejados da relação;

- O resultado da operação PROJEÇÃO tem apenas os atributos especificados em ***<lista atributos>*** na mesma ordem.
- A operação PROJEÇÃO remove quaisquer tuplas duplicadas.

A operação PROJEÇÃO

Listar último nome, primeiro nome e o salário de cada funcionário.

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Endereco	Sexo	Salario	Cpf_supervisor	Dnr
João	B	Silva	12345678966	09-01-1965	Rua das Flors, 751, São Paulo	M	30.000	33344555587	5
Fernando	T	Wong	33344555587	08-12-1955	Rua da Lapa, 34, São Paulo	M	40.000	88866555576	5
Alice	J	Zelaya	99988777767	19-01-1968	Rua Souza Lima, 34, Curitiba, PR	F	25.000	98765432168	4
Jennifer	S	Souza	98765432168	20-06-1941	Av. Brigadeiro, 54, São Paulo,SP	F	43.000	88866555576	4
Joice	A	Leite	45345345376	31-07-1972	Av Lucas Obes, 74, São Paulo	F	25.000	33344555587	5
Jorge	E	Brito	88888555578	10-11-1937	Rua do Horto, 35, São Paulo, SP	M	55.000	null	1

A operação PROJEÇÃO

- ***Especificação da operação PROJEÇÃO:***

$$\pi_{Unome,\ Pnome,\ Salario}(FUNCIONARIO)$$

- ***Resultado da operação:***

Unome	Pnome	Salario
Silva	João	30.000
Wong	Fernando	40.000
Zelaya	Alice	25.000
Souza	Jennifer	43.000
Leite	Joice	25.000
Brito	Jorge	55.000

A operação PROJEÇÃO

Listar o sexo e o salário de cada funcionário.

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Endereco	Sexo	Salario	Cpf_supervisor	Dnr
João	B	Silva	12345678966	09-01-1965	Rua das Flors, 751, São Paulo	M	30.000	33344555587	5
Fernando	T	Wong	33344555587	08-12-1955	Rua da Lapa, 34, São Paulo	M	40.000	88866555576	5
Alice	J	Zelaya	99988777767	19-01-1968	Rua Souza Lima, 34, Curitiba, PR	F	25.000	98765432168	4
Jennifer	S	Souza	98765432168	20-06-1941	Av. Brigadeiro, 54, São Paulo,SP	F	43.000	88866555576	4
Joice	A	Leite	45345345376	31-07-1972	Av Lucas Obes, 74, São Paulo	F	25.000	33344555587	5
Jorge	E	Brito	88888555578	10-11-1937	Rua do Horto, 35, São Paulo, SP	M	55.000	null	1

A operação PROJEÇÃO

- ***Especificação da operação PROJEÇÃO:***

$$\pi_{Sexo,\ Salario}(FUNCIONARIO)$$

- ***Resultado da operação:***

Sexo	Salario
M	30.000
M	40.000
F	25.000
F	43.000
M	55.000

A tupla <F, 25.000> aparece apenas uma vez na relação resultante, embora essa combinação de valores apareça duas vezes na relação FUNCIONARIO. Logo, ocorreu a **eliminação de duplicatas**.

A operação PROJEÇÃO

- O número de tuplas em uma relação resultante de uma operação PROJEÇÃO é menor ou igual ao número de tuplas em **R**.

- Se a lista de projeção inclui alguma chave de **R**, a relação resultante tem o mesmo número de tuplas que **R**.

$$\pi_{lista1}(\pi_{lista2}(R)) = \pi_{lista1}(R)$$

A *<lista2>* deve conter os mesmos atributos que a *<lista3>*.

- A comutatividade não é mantida na projeção.

Sequência de operações

Para a maioria das consultas, precisamos aplicar várias operações da álgebra relacional uma após a outra, a partir das seguintes possibilidades:

- Alinhando as operações e gerando uma única expressão da álgebra relacional (expressão em linha).
- Aplicando uma operação de cada vez e criando relações de resultados intermediários.

Sequência de operações

Recuperar o primeiro nome, sobrenome e salário de todos os funcionários que trabalham no departamento número 5.

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Endereco	Sexo	Salario	Cpf_supervisor	Dnr
João	B	Silva	12345678966	09-01-1965	Rua das Flors, 751, São Paulo	M	30.000	33344555587	5
Fernando	T	Wong	33344555587	08-12-1955	Rua da Lapa, 34, São Paulo	M	40.000	88866555576	5
Alice	J	Zelaya	99988777767	19-01-1968	Rua Souza Lima, 34, Curitiba, PR	F	25.000	98765432168	4
Jennifer	S	Souza	98765432168	20-06-1941	Av. Brigadeiro, 54, São Paulo,SP	F	43.000	88866555576	4
Joice	A	Leite	45345345376	31-07-1972	Av Lucas Obes, 74, São Paulo	F	25.000	33344555587	5
Jorge	E	Brito	88888555578	10-11-1937	Rua do Horto, 35, São Paulo, SP	M	55.000	null	1

Sequência de operações

Recuperar o primeiro nome, sobrenome e salário de todos os funcionários que trabalham no departamento número 5.

- ***Operações alinhadas:***

$$\pi_{Pnome,\ Unome,\ Salario}(\sigma_{Dnr\ =\ 5}(FUNCIONARIO))$$

- ***Sequência de operações:***

$$A \longleftarrow \sigma_{Dnr\ =\ 5}(FUNCIONARIO)$$
$$B \longleftarrow \pi_{Pnome,\ Unome,\ Salario}(A)$$

Sequência de operações

Recuperar o primeiro nome, sobrenome e salário de todos os funcionários que trabalham no departamento número 5.

- ***Resultado da operação:***

Pnome	Unome	Salario
João	Silva	30.000
Fernando	Wong	40.000
Joice	Leite	25.000

A operação RENOMEAR

- Pode-se renomear os atributos nas relações intermediárias e de resultado.
- Para renomear os atributos em uma relação, é necessário listar os novos nomes de atributo entre parênteses.
- ***Especificação da operação RENOMEAR:***

$$TEMP \longleftarrow \sigma_{Dnr\ =\ 5}(FUNCIONARIO)$$
$$R(Primeiro\_nome,\ Ultimo\_nome,\ Salario) \longleftarrow$$
$$\pi_{Pnome,\ Unome,\ Salario}(TEMP)$$

A operação RENOMEAR

- ***Especificação da operação RENOMEAR:***

$$TEMP \longleftarrow \sigma_{Dnr\ =\ 5}(FUNCIONARIO)$$

$$R(Primeiro\_nome,\ Ultimo\_nome,\ Salario) \longleftarrow$$

$$\pi_{Pnome,\ Unome,\ Salario}(TEMP)$$

- ***Resultado da primeira operação - relação TEMP:***

TEMP

Pnome	Minicial	Unome	Cpf	Datanasc	Endereco	Sexo	Salario	Cpf_supervisor	Dnr
João	B	Silva	12345678966	09-01-1965	Rua das Flors, 751, São Paulo	M	30.000	33344555587	5
Fernando	T	Wong	33344555587	08-12-1955	Rua da Lapa, 34, São Paulo	M	40.000	88866555576	5
Joice	A	Leite	45345345376	31-07-1972	Av Lucas Obes, 74, São Paulo	F	25.000	33344555587	5

A operação RENOMEAR

- ***Especificação da operação RENOMEAR:***

$$TEMP \longleftarrow \sigma_{Dnr\ =\ 5}(FUNCIONARIO)$$

$$R(Primeiro\_nome,\ Ultimo\_nome,\ Salario) \longleftarrow \pi_{Pnome,\ Unome,\ Salario}(TEMP)$$

- ***Resultado da segunda operação - relação R:***

R

Primeiro_nome	Ultimo_nome	Salario
João	Silva	30.000
Fernando	Wong	40.000
Joice	Leite	25.000

A operação RENOMEAR

- O operador unário **RENOMEAR** formal pode renomear o nome da **relação** ou os nomes de **atributos**, ou ambos.

$$\rho_{S(B1,\ B2,\ \ldots\ Bn)}(R) \text{ ou } \rho_{S}(R) \text{ ou } \rho_{B1,\ B2,\ \ldots\ Bn}(R)$$

- O símbolo **rho** (ρ) é usado para indicar o operador RENOMEAR.
- S é o nome da nova relação.
- Entre parênteses são os novos nomes de atributo.

Operações de teoria de conjunto

Operações de teoria de conjunto são usadas para **mesclar os elementos de dois conjuntos**, através das operações binárias.

- *União*
- *Intersecção*
- *Diferença*

Estas operações precisam ser aplicadas a relações com o **mesmo formato de tuplas (compatibilidade)**: mesmo número de atributos e mesmo domínio.

A operação UNIÃO

$$R \cup S$$

- O resultado dessa operação é uma relação que inclui todas as tuplas que estão em **R** ou em **S**, tanto em **R** ou em **S**.
- As tuplas duplicadas são eliminadas.
- É aplicável a qualquer quantidade de relações (**operação associativa**);
- Consiste em uma operação **comutativa**:

$$R \cup S = S \cup R$$

A operação UNIÃO

$$FUNCS\_DEP5 \longleftarrow \sigma_{Dnr\ =\ 5}(FUNCIONARIO)$$
$$RESULTADO1 \longleftarrow \pi_{Cpf}(FUNCS\_DEP5)$$
$$RESULTADO2(Cpf) \longleftarrow \pi_{Cpf\_supervisor}(FUNCS\_DEP5)$$
$$RESULTADO \longleftarrow RESULTADO1 \cup RESULTADO2$$

- **RESULTADO1** tem o CPF de todos os funcionários que trabalham no departamento 5.
- **RESULTADO2** tem o CPF de todos os funcionários que supervisionam diretamente um funcionário que trabalha no departamento 5.

RESULTADO1

Cpf
12345678966
33344555587
45345345376

RESULTADO2

Cpf
33344555587
88866555576

RESULTADO

Cpf
12345678966
33344555587
45345345376
88866555576

A operação UNIÃO

As relações ALUNO e PROFESSOR são *compatíveis*.

ALUNO

Pn	Un
Susana	Yao
José	Goncalves
Barbara	Pires
Ana	Tavares
Jonas	Wang
Ernesto	Gilberto

PROFESSOR

Pnome	Unome
João	Silva
Ricardo	Braga
Susana	Yao
Francisco	Leme

A operação UNIÃO

- ***Especificação da operação UNIÃO:***

$$ALUNO \cup PROFESSOR$$

- ***Resultado da operação:***

Pn	Un
Susana	Yao
José	Goncalves
Barbara	Pires
Ana	Tavares
Jonas	Wang
Ernesto	Gilberto
João	Silva
Ricardo	Braga
Francisco	Leme

A operação INTERSECÇÃO

$$R \cap S$$

- O resultado é uma relação que inclui todas as tuplas que estão tanto em R quanto em S.
- É aplicável a qualquer quantidade de relações (**operação associativa**).
- Consiste em uma **operação comutativa**:

$$R \cap S = S \cap R$$

A operação INTERSECÇÃO

As relações ALUNO e PROFESSOR são compatíveis.

ALUNO

Pn	Un
Susana	Yao
José	Goncalves
Barbara	Pires
Ana	Tavares
Jonas	Wang
Ernesto	Gilberto

PROFESSOR

Pnome	Unome
João	Silva
Ricardo	Braga
Susana	Yao
Francisco	Leme

A operação INTERSECÇÃO

- ***Especificação da operação INTERSECÇÃO:***

$$ALUNO \cap PROFESSOR$$

- ***Resultado da operação:***

Pn	Un
Susana	Yao

Observe que a relação mostra apenas aqueles que são tanto alunos quanto professores.

A operação DIFERENÇA DE CONJUNTO

$$R - S$$

- O resultado é uma relação que inclui todas as tuplas que estão em R, mas não em S.
- Não é uma operação comutativa:

$$R - S \neq S - R$$

A operação DIFERENÇA DE CONJUNTO

As relações ALUNO e PROFESSOR são compatíveis.

ALUNO

Pn	Un
Susana	Yao
José	Goncalves
Barbara	Pires
Ana	Tavares
Jonas	Wang
Ernesto	Gilberto

PROFESSOR

Pnome	Unome
João	Silva
Ricardo	Braga
Susana	Yao
Francisco	Leme

A operação DIFERENÇA DE CONJUNTO

- ***Especificação da operação DIFERENÇA DE CONJUNTO:***

$$ALUNO - PROFESSOR$$

- ***Resultado da operação:***

Pn	Un
José	Goncalves
Barbara	Pires
Ana	Tavares
Jonas	Wang
Ernesto	Gilberto

Observe que são exibidos os nomes dos alunos que não são professores.

A operação DIFERENÇA DE CONJUNTO

- ***Especificação da operação DIFERENÇA DE CONJUNTO:***

$$PROFESSOR - ALUNO$$

- ***Resultado da operação:***

Pnome	Unome
João	Silva
Ricardo	Braga
Francisco	Leme

Observe que são exibidos os nomes dos professores que não são alunos.

A operação PRODUTO

- Produz uma relação que têm os atributos de **R1** e **R2** e inclui como tuplas todas as possíveis combinações de tuplas de **R1** e **R2**.

$$R_1 \times S_2$$

- A operação é indicada por x.

- Consiste em uma operação de conjunto binária, no entanto, as relações sobre as quais ela é aplicada não precisam ser compatíveis na união.

A operação PRODUTO

Suponha que deseja-se recuperar uma lista de dos nomes dos dependentes de cada funcionário.

- ***Especificação de operações:***

$$A \longleftarrow \sigma_{Sexo = 'F'}(Funcionario)$$
$$B \longleftarrow \pi_{Pnome,\ Unome,\ Cpf}(A)$$
$$C \longleftarrow B \times Dependente$$
$$D \longleftarrow \sigma_{Cpf = Fcpf}(C)$$
$$E \longleftarrow \pi_{Pnome,\ Unome,\ Nome\_dependente}(D)$$

A operação PRODUTO

- ***Resultado final das operações:***

Pnome	Unome	Cpf	Fcpf	Nome dependente	Sexo	Datanasc
Alice	Zelaya	99988777767	33344555587	Alicia	F	05-04-1986
Alice	Zelaya	99988777767	33344555587	Tiago	M	25-10-1983
Alice	Zelaya	99988777767	33344555587	Janaina	F	03-05-1958
Alice	Zelaya	99988777767	98765432168	Antonio	M	20-02-1942
Alice	Zelaya	99988777767	12345678966	Michael	M	04-01-1988
Alice	Zelaya	99988777767	12345678966	Alicia	F	30-12-1988
Alice	Zelaya	99988777767	12345678966	Elisabeth	F	05-05-1967
Jennifer	Souza	98765432168	33344555587	Alicia	F	05-04-1986
Jennifer	Souza	98765432168	33344555587	Tiago	M	20-10-1983
Jennifer	Souza	98765432168	33344555587	Janaina	F	03-05-1958
Jennifer	Souza	98765432168	98765432168	Antonio	M	28-02-1942

Observe que cada dependente é combinado com cada funcionário.

A operação PRODUTO

- ***Resultado final das operações - continuação da tabela:***

Pnome	Unome	Cpf	Fcpf	Nome dependente	Sexo	Datanasc
Jennifer	Souza	98765432168	12345678966	Michael	M	04-01-1988
Jennifer	Souza	98765432168	12345678966	Alicia	F	30-12-1988
Jennifer	Souza	98765432168	12345678966	Elizabeth	F	05-05-1967
Joice	Leite	45345345376	33344555587	Alicia	F	05-04-1986
Joice	Leite	45345345376	33344555587	Tiago	M	25-10-1983
Joice	Leite	45345345376	33344555587	Janaina	F	03-05-1958
Joice	Leite	45345345376	98765432168	Antonio	M	28-02-1942
Joice	Leite	45345345376	12345678966	Michael	M	04-01-1988
Joice	Leite	45345345376	12345678966	Alicia	F	30-12-1988
Joice	Leite	45345345376	12345678966	Elizabeth	F	05-05-1967

Observe que cada dependente é combinado com cada funcionário.

A operação JUNÇÃO

- Operação utilizada para combinar tuplas relacionadas de duas relações em uma única tupla *maior*.
- Produz todas as combinações de tuplas R1 e R2 que satisfazem a condição de junção, em **tuplas únicas combinadas**.

$$R_1 \bowtie_{condicional} R_2$$

- As tuplas cujos atributos de junção são *NULL* ou dos quais a condição de junção é FALSE não aparecem no resultado.

A operação JUNÇÃO

- A operação JUNÇÃO pode ser especificada como uma operação PRODUTO seguida por uma operação SELEÇÃO.

- A diferença é que a JUNÇÃO apresenta no resultado apenas combinações de tuplas que satisfazem a condição de junção, enquanto no PRODUTO todas as combinações de tuplas são incluídas no resultado.

A operação JUNÇÃO

Recuperar o nome do gerente de cada departamento.

- ***Especificação das operações:***

$$A \leftarrow Departamento \bowtie_{Cpf\_gerente = Cpf} (FUNCIONARIO)$$

$$B \leftarrow \pi_{Dnome,\ Unome,\ Pnome}(A)$$

- ***Resultado das operações:***

B

Dnome	Dnumero	Cpf gerente	Pnome	Minicial	Unome	Cpf
Pesquisa	5	33344555587	Fernando	T	Wong	33344555587
Administracao	4	98765432168	Jennifer	S	Souza	98765432168
Matriz	1	88866555576	Jorge	E	Brito	88866555576

A operação JUNÇÃO

Variações de junção - Equijunção:

- Utilização apenas de comparações de igualdade.
- No resultado de um EQUIJUNÇÃO, sempre há um ou mais pares de atributos que possuem valores idênticos em cada tupla.
- No exemplo anterior, *Cpf_gerente* e *Cpf* são idênticos em cada tupla de *DEP_GER*.

A operação JUNÇÃO

Variações de junção - Junção natural:

- Elimina o segundo atributo (desnecessário) na condição de EQUIJUNÇÃO.
- Essa operação requer que os dois atributos de junção tenham o mesmo nome nas duas relações.
- Caso necessário uma operação de renomeação é aplicada primeiro.

A operação JUNÇÃO

Variações de junção - Junção natural:

Combinar cada tupla PROJETO com a tupla DEPARTAMENTO que controla o projeto.

$A \leftarrow \sigma_{Dnome,\ Dnum,\ Cpf\_gerente,\ Data\_inicio\_gerente}(Departameto)$
$B \longleftarrow Projeto\ *\ A$

- ***Dnum*** é o atributo de junção, pois é o único atributo com o mesmo nome nas duas relações.

A operação DIVISÃO

- Utilizada quando se deseja extrair de uma relação **R1** uma determinada parte que possui de atributos da relação **R2**.

$$R_1 \div R_2$$

A operação DIVISÃO

Recuperar os nomes dos funcionários que trabalham em todos os projetos em que 'João Silva' trabalha.

- ***Primeiro passo:***

$$SILVA \leftarrow \sigma_{Pnome = 'Joao' \wedge Unome = 'Silva'}(Funcionario)$$
$$SILVA\_PNRS \leftarrow \pi_{Pnr}(TRABALHA\_EM \bowtie_{Fcpf = Cpf} SILVA)$$

SILVA_PNRS

Pnr
1
2

A operação DIVISÃO

Recuperar os nomes dos funcionários que trabalham em todos os projetos em que 'João Silva' trabalha.

- ***Segundo passo:***

$$CPF\_PNRS \longleftarrow \pi_{Fcpf,\ Pnr}(TRABALHA\_EM)$$

CPF_PNRS

Fcpf	Pnr
12345678966	1
12345678966	2
66688444476	3
45345345376	1
45345345376	2
99988777767	30
99988777767	10
98798798733	10
98798798733	30
98765432168	30
98765432168	20
88866555576	20

A operação DIVISÃO

Recuperar os nomes dos funcionários que trabalham em todos os projetos em que 'João Silva' trabalha.

- ***Terceiro passo:***

$$CPFS(Cpf) \leftarrow CPF\_PNRS \div SILVA\_PNRS$$
$$RESULTADO \leftarrow \pi_{Pnome,\ Unome}(CPFS * FUNCIONARIO)$$

CPFS

Cpf
12345678966
45345345376

Projeção generalizada

- Estende a operação de projeção, permitindo que as funções dos atributos sejam incluídas na lista de projeção.

$$\pi_{F1,F2,...,FN}(R)$$

F1, F2, ..., FN, são funções sobre os atributos na relação R e podem envolver operações aritméticas e valores constantes.

Projeção generalizada

Exemplo 01 - Considere a relação:

FUNCIONARIO(Cpf, Salario, Deducao, Anos_em_servico)

A partir da relação acima, deseja-se extrair as seguintes informações:

- Salário líquido = Salario - Deducao
- Bonus = 2.000 * Anos_em_servico
- Imposto = 0.25 * Salario

Neste caso, a *projeção generalizada* pode ser utilizada da seguinte maneira:

$A \leftarrow \pi_{Cpf,\ Salario-Deducao,\ 2.000*Anos\_em\_servico,\ 0.25*Salario}(FUNCIONARIO)$

$B \leftarrow \rho_{Cpf,\ Salario\_liquido,\ Bonus,\ Imposto}\ (A)$

Funções de agregação e agrupamento

- **Funções de agregação** matemáticas sobre coleções de valores do banco de dados.

- SOMA (*sum*), MÉDIA (*avg*), MÁXIMO (*max*), MÍNIMO (*min*), CONTA (*Contagem de tuplas - count*).

- É representada a partir de:

$$\text{<atributos agrupamento>}\ \mathfrak{Im}_{\text{<lista de funcoes>}}(R)$$

Funções de agregação e agrupamento

Exemplo 02 - Recuperar o número do departamento, o número de funcionários no departamento e seu salário médio:

$$A \leftarrow Dnr\ \mathfrak{Im}_{COUNT(Cpf),\ AVERAGE(Salario)}(FUNCIONARIO)$$
$$B \leftarrow \rho_{Dnr,\ Numero\_Funcionarios,\ Media\_Salarial}(A)$$

- **Resultado da operação:**

Dnr	Nr_de_funcionarios	Media_sal
5	3	31.600
4	2	34.000
1	1	55.000

Funções de agregação e agrupamento

Exemplo 02 - Recuperar o número do departamento, o número de funcionários no departamento e seu salário médio:

Se não houver renomeação, os atributos da relação resultante serão concatenados no nome da função com o nome do atributo.

- ***Operação sem a renomeação seria:***

$$\text{Dnr}\ \mathfrak{Im}_{COUNT(Cpf),\ MEDIA(Salario)}(FUNCIONARIO)$$

- ***Resultado da operação:***

Dnr	Contador_cpf	Media_salario
5	3	31.600
4	2	34.000
1	1	55.000

Funções de agregação e agrupamento

Exemplo 02 - Recuperar o número do departamento, o número de funcionários no departamento e seu salário médio:

Se nenhum atributo de agrupamento for especificado, as funções são aplicadas a ***todas*** *as tuplas, resultando em uma única tupla.*

- ***Operação sem o atributo de agrupamento:***

$$\gamma_{CONTA\ Cpf,\ MEDIA\ Salario}(FUNCIONARIO)$$

- ***Resultado da operação:***

Contador_cpf	Media_salario
6	36.300

Operações de fechamento recursivo

- Aplicada a um **relacionamento recursivo** entre tuplas do mesmo tipo.
- Exemplo: *Relacionamento entre funcionário e supervisor:*

Observe a chave estrangeira **Cpf_supervisor**: relaciona cada tupla de funcionário (**papel de supervisionado**) a outra tupla de funcionário (**papel de supervisor**).

Operações de fechamento recursivo

Exemplo 03 - Especificar os Cpfs de todos os funcionários *f'* supervisionados diretamente pelo funcionário *f* cujo nome é 'Jorge Brito'. Passos para resolução:

1. Extrair o Cpf do funcionário com o nome 'Jorge Brito':

Especificação da operação:

$$A \leftarrow \sigma_{Pnome='Jorge' \ AND \ Unome='Brito'}(FUNCIONARIO)$$

$$B \leftarrow \pi_{Cpf}(A)$$

Resultado da operação:

Cpf
88866555576

Operações de fechamento recursivo

Exemplo 03 - Especificar os Cpfs de todos os funcionários *f'* supervisionados diretamente pelo funcionário *f* cujo nome é 'Jorge Brito'. Passos para resolução:

2. Projetar o Cpf e o Cpf_supervisor de todas as tuplas:

Especificação da operação:

$$\text{SUPERVISAO} \leftarrow \pi_{Cpf,\ Cpf\_supervisor}\ (FUNCIONARIO)$$

Resultado da operação:

Cpf	Cpf_supervisor
12345678966	33344555587
33344555587	88866555576
99988777767	98765432168
98765432168	88866555576
45345345376	33344555587
88866555576	null

Operações de fechamento recursivo

Exemplo 03 - Especificar os Cpfs de todos os funcionários *f'* supervisionados diretamente pelo funcionário *f* cujo nome é 'Jorge Brito'. Passos para resolução:

3. Efetuar a junção dos resultados obtidos nos itens 1 e 2, a partir da chave Cpf e Cpf_supervisor:

Especificação da operação:

$$A \leftarrow SUPERVISAO \bowtie_{Cpf\_supervisor=Cpf} (CPF\_BRITO)$$

$$B \leftarrow \pi_{Cpf}(A)$$

Resultado da operação:

Cpf
33344555587
98765432168

Operações OUTER JOIN (junção externa)

Junção interna (*inner join*):

- As operações JUNÇÃO descritas anteriormente combinam com tuplas que satisfazem a condição de junção.
- As tuplas sem uma tupla correspondente são eliminadas do resultado.
- Tuplas com valores NULL nos atributos de junção também são eliminadas.

Operações OUTER JOIN (junção externa)

Exemplo 04 - Lista de todos os nomes de funcionários, bem como o nome dos departamentos que eles gerenciam, **se eles gerenciarem um departamento**:

- Para este caso, aplicamos a operação JUNÇÃO EXTERNA A ESQUERDA, para recuperar o resultado.
- Mantém a tupla da *primeira* relação (ou da *esquerda*).

$$R \ltimes S$$

- Se nenhuma tupla correspondente for encontrada em S, então os atributos de S no resultado da junção serão preenchidos com valores NULL.

Operações OUTER JOIN (junção externa)

Exemplo 04 - Lista de todos os nomes de funcionários, bem como o nome dos departamentos que eles gerenciam, **se eles gerenciarem um departamento**:

Especificação da operação:

$$A \leftarrow \text{FUNCIONARIO} \rtimes_{Cpf = Cpf\_gerente} (DEPARTAMENTO)$$

$$\text{RESULTADO} \leftarrow \pi_{Pnome,\ Minicial,\ Unome,\ Dnome} (A)$$

Resultado:

Pnome	Minicial	Unome	Dnome
João	B	Silva	null
Fernando	T	Wong	Pesquisa
Alice	J	Zelaya	null
Jennifer	S	Souza	Administracão
Joice	A	Leite	null
Jorge	E	Brito	Matriz

Operações OUTER JOIN (junção externa)

JUNÇÃO EXTERNA A DIREITA:

- Indicada por:

$$R \ltimes S$$

- Mantém a tupla da *segunda* relação (ou da *direita*);

JUNÇÃO EXTERNA COMPLETA:

- Indicada por:

$$R \bowtie S$$

- Mantém todas as tuplas nas relações da esquerda e da direta quando nenhum tupla correspondente for encontrada, preenchendo-as com valores NULL conforme a necessidade.

Exemplos de consulta na Álgebra Relacional

- A seguir, exibiremos exemplos adicionais para ilustrar o uso das operações de álgebra relacional.
- Os exemplos se referem ao banco de dados da figura anexada ao próximo slide.
- Em geral, a mesma consulta pode ser indicada de várias maneiras usando as diversas operações.

Exemplos de consulta na Álgebra Relacional

Um estado de banco de dados possível para o esquema de banco de dados relacional EMPRESA - Parte 1.

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Endereco	Sexo	Salario	Cpf_supervisor	Dnr
João	B	Silva	12345678966	09-01-1965	Rua das Flors, 751, São Paulo	M	30.000	33344555587	5
Fernando	T	Wong	33344555587	08-12-1955	Rua da Lapa, 34, São Paulo	M	40.000	88866555576	5
Alice	J	Zelaya	99988777767	19-01-1968	Rua Souza Lima, 34, Curitiba, PR	F	25.000	98765432168	4
Jennifer	S	Souza	98765432168	20-06-1941	Av. Brigadeiro, 54, São Paulo,SP	F	43.000	88866555576	4
Joice	A	Leite	45345345376	31-07-1972	Av Lucas Obes, 74, São Paulo	F	25.000	33344555587	5
Jorge	E	Brito	88888555578	10-11-1937	Rua do Horto, 35, São Paulo, SP	M	55.000	null	1

Exemplos de consulta na Álgebra Relacional

Um estado de banco de dados possível para o esquema de banco de dados relacional EMPRESA - Parte 2.

DEPARTAMENTO

Dnome	Dnumero	Cpf_gerente	Data_inicio_gerente
Pesquisa	5	33344555587	22-06-1988
Administracao	4	98765432168	01-01-1995
Matriz	1	88866555578	19-06-1981

LOCALIZACAO_DEP

Dnumero	Dlocal
1	São Paulo
4	Mauá
5	Santo André
5	Itu
5	São Paulo

Exemplos de consulta na Álgebra Relacional

Um estado de banco de dados possível para o esquema de banco de dados relacional EMPRESA - Parte 3.

TRABALHA_EM

Fcpf	Pnr	Horas
12345678966	1	32,5
12345678966	2	7,5
66688444476	3	40,0
45345345376	1	20,0
45345345376	2	20,0
33344555587	2	10,0
33344555587	3	10,0
33344555587	10	10,0
33344555587	20	10,0
99988777767	30	30,0
99988777767	10	10,0
98798798733	10	35,0
98798798733	30	5,0
98798798733	30	20,0
98798798733	20	15,0
88866555576	20	null

PROJETO

ProjNome	ProjNumero	ProjLocal	Dnum
ProdutoX	1	Santo Andre	5
ProdutoY	2	Itu	5
ProdutoZ	3	São Paulo	5
Informatizacão	10	Mauá	4
Reorganizacão	20	São Paulo	1
Novos Benefícios	30	Mauá	4

DEPENDENTE

Fcpf	Nome dependente	Sexo	Datanasc	Parentesco
33344555587	Alicia	F	05-04-1986	Filha
33344555587	Tiago	M	25-10-1983	Filho
33344555587	Janaina	F	03-05-1958	Esposa
98765432168	Antonio	M	28-02-1942	Marido
12345678966	Michael	M	04-01-1988	Filho
12345678966	Alicia	F	30-12-1988	Filha
12345678966	Elizabeth	F	05-05-1987	Esposa

Exemplos de consulta na Álgebra Relacional

CONSULTA 01:

- Recuperar o nome e o endereço de todos os funcionários que trabalham para o departamento ′Pesquisa′.

$$A \leftarrow \sigma_{Dnome='Pesquisa'}(DEPARTAMENTO)$$

$$B \leftarrow A \bowtie_{Dnumero=Dnr} (FUNCIONARIO)$$

$$C \leftarrow \pi_{Pnome,Unome,Endereco}(B)$$

Exemplos de consulta na Álgebra Relacional

CONSULTA 02:

- Para cada projeto localizado em ′Mauá′, liste o número do projeto, o número do departamento que o controla e o último nome, endereço e data de nascimento do gerente de departamento.

$$A \leftarrow \sigma_{Projlocal='Maua'}(PROJETO)$$

$$B \leftarrow A \bowtie_{Dnum=Dnumero} (DEPARTAMENTO)$$

$$C \leftarrow B \bowtie_{Cpf\_gerente=Cpf} (FUNCIONARIO)$$

$$D \leftarrow \pi_{Projnumero,\ Dnum,\ Unome,\ Endereco,\ Datanasc}(C)$$

Exemplos de consulta na álgebra Relacional

CONSULTA 02:

- Para cada projeto localizado em ′Mauá′, liste o número do projeto, o número do departamento que o controla e o último nome, endereço e data de nascimento do gerente de departamento.

Neste exemplo, primeiro selecionamos os projetos que se localizam em Mauá, depois os juntamos a seus departamentos de controle, em seguida juntamos o resultado com os gerentes de departamento. Finalmente, aplicamos uma operação de projeção aos atributos desejados.

Exemplos de consulta na Álgebra Relacional

CONSULTA 03:

- Recuperar os nomes dos funcionários que não possuem dependentes.

$$A \leftarrow \pi_{Cpf}(FUNCIONARIO)$$

$$B(Cpf) \leftarrow \pi_{Fcpf}(DEPENDENTE)$$

$$C(Cpf) \leftarrow A - B$$

$$D \leftarrow \pi_{Unome,\ Pnome}(C * FUNCIONARIO)$$

Exemplos de consulta na álgebra Relacional

CONSULTA 03:

- Recuperar os nomes dos funcionários que não possuem dependentes.

Primeiro, recuperamos uma relação com todos os Cpfs de funcionários em A. Depois, criamos uma tabela com os Cpfs dos funcionários que possuem pelo menos um dependente em B. Então, aplicamos a operação DIFERENÇA DE CONJUNTO para recuperar os Cpfs de funcionários sem dependentes em C, e finalmente juntamos isso com FUNCIONARIO para recuperar os atributos desejados.

Seção 07

Modelo Entidade-Relacionamento

Projeto de Banco de Dados

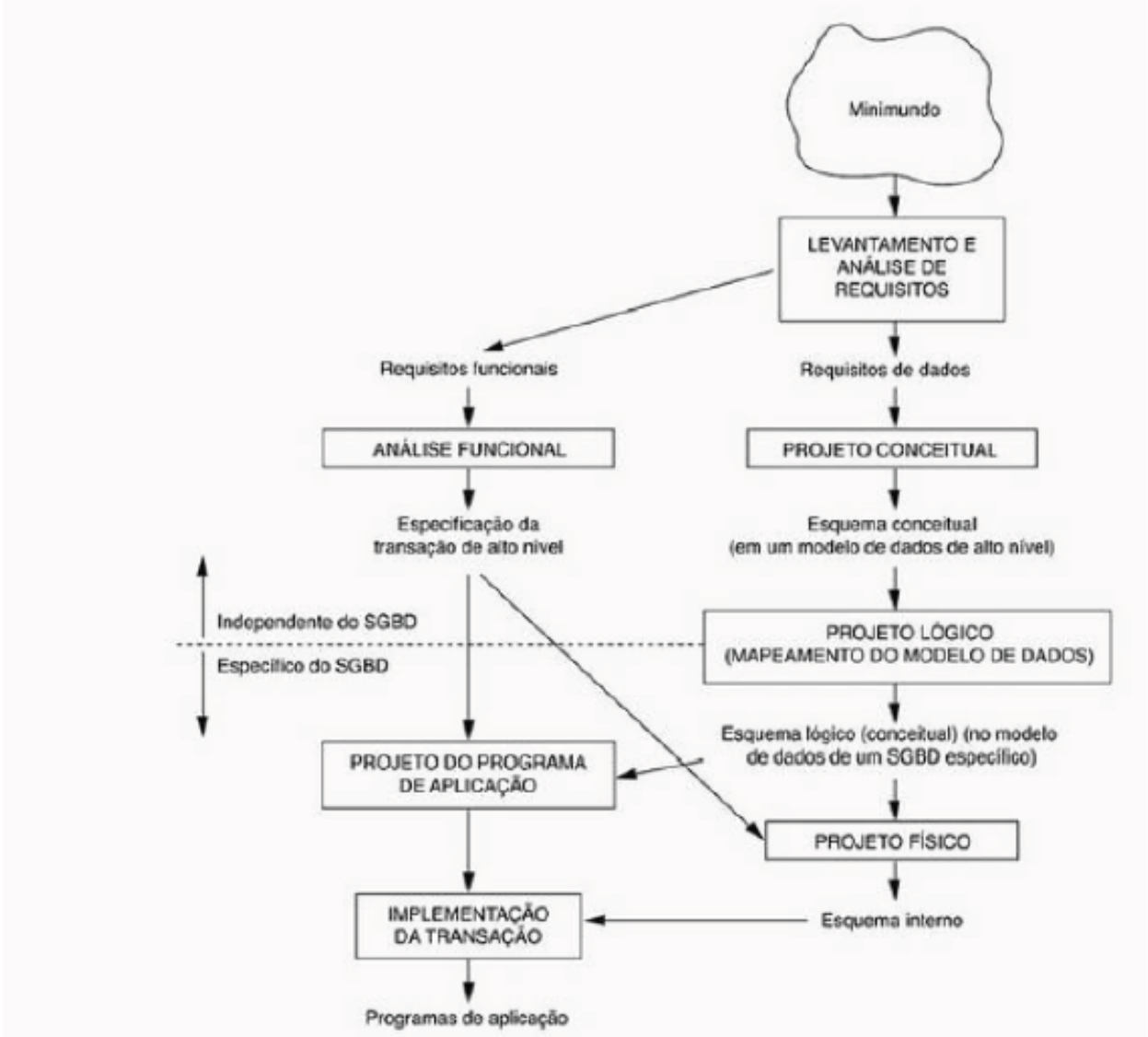
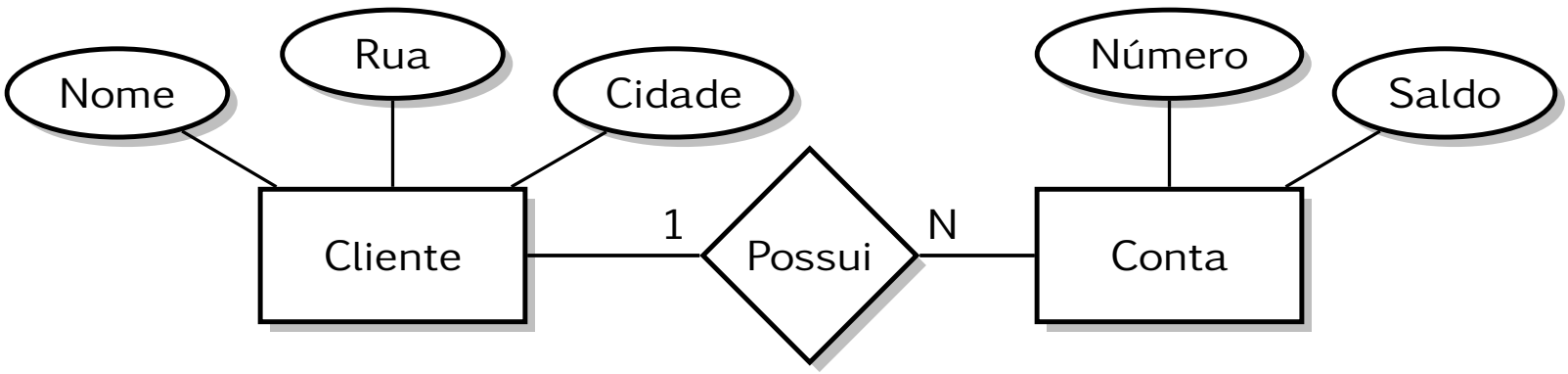


Figura 7.1
Um diagrama simplificado para ilustrar as principais fases do projeto de banco de dados.

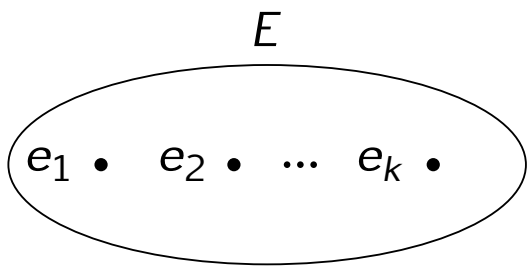
Modelo Entidade-Relacionamento (MER)

- Modelo conceitual elaborado a partir da especificação do minimundo
- Minimundo → Comumente especificado de forma textual, estabalecendo os requisitos de dados fruto da fase de levantamento e análise de requisitos
- Diagrama Entidade-Relacionamento (DER) → Representa graficamente as entidades, atributos, relacionamentos e restrições do MER



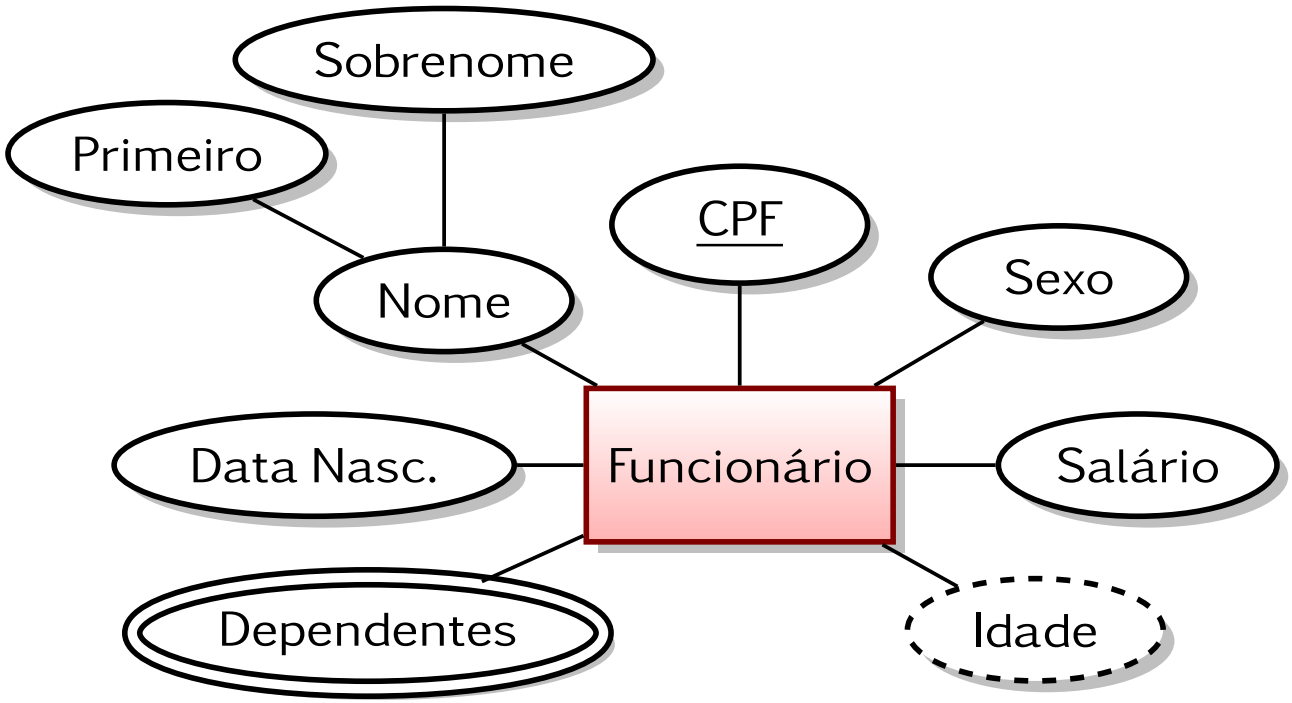
MER: Entidade

- Ente com existência real no minimundo especificado
- Seja $E = \{e_1, e_2, \cdots, e_k\}$ um conjunto de k entidades de mesmo tipo. Define-se:
  - Tipo de Entidade $(E) \rightarrow$ Conjunto de instâncias de entidades do mesmo tipo
  - Instância de Entidade $(e_i) \rightarrow$ Um ente específico de um tipo de entidade E, tal que $e_i \in E$



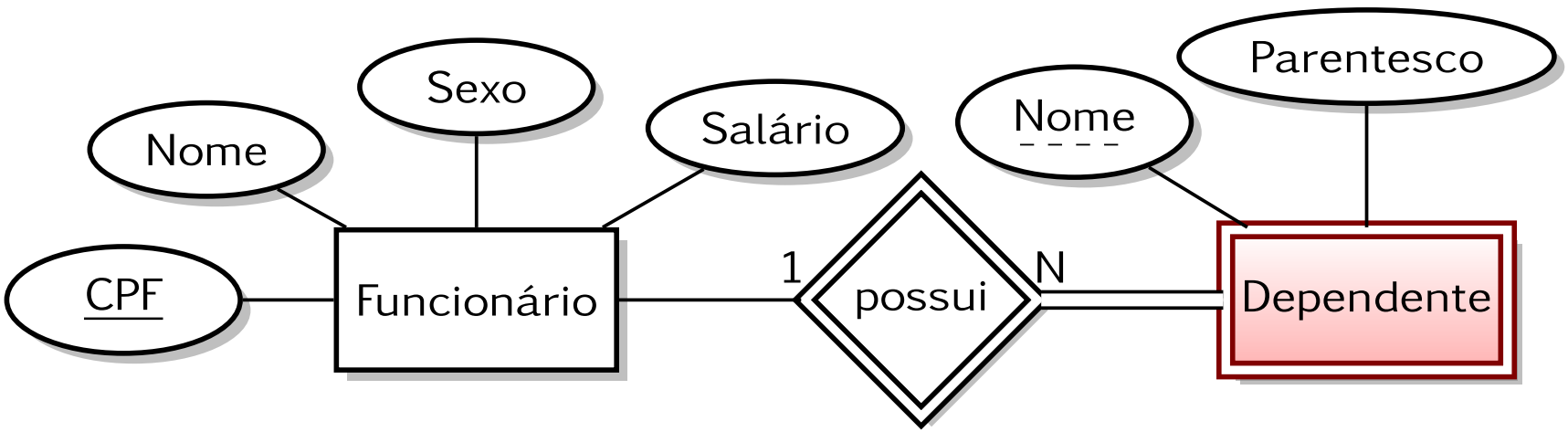
MER: Entidade

- DER → Representa-se um tipo de entidade, ou simplesmente entidade, como um retângulo rotulado



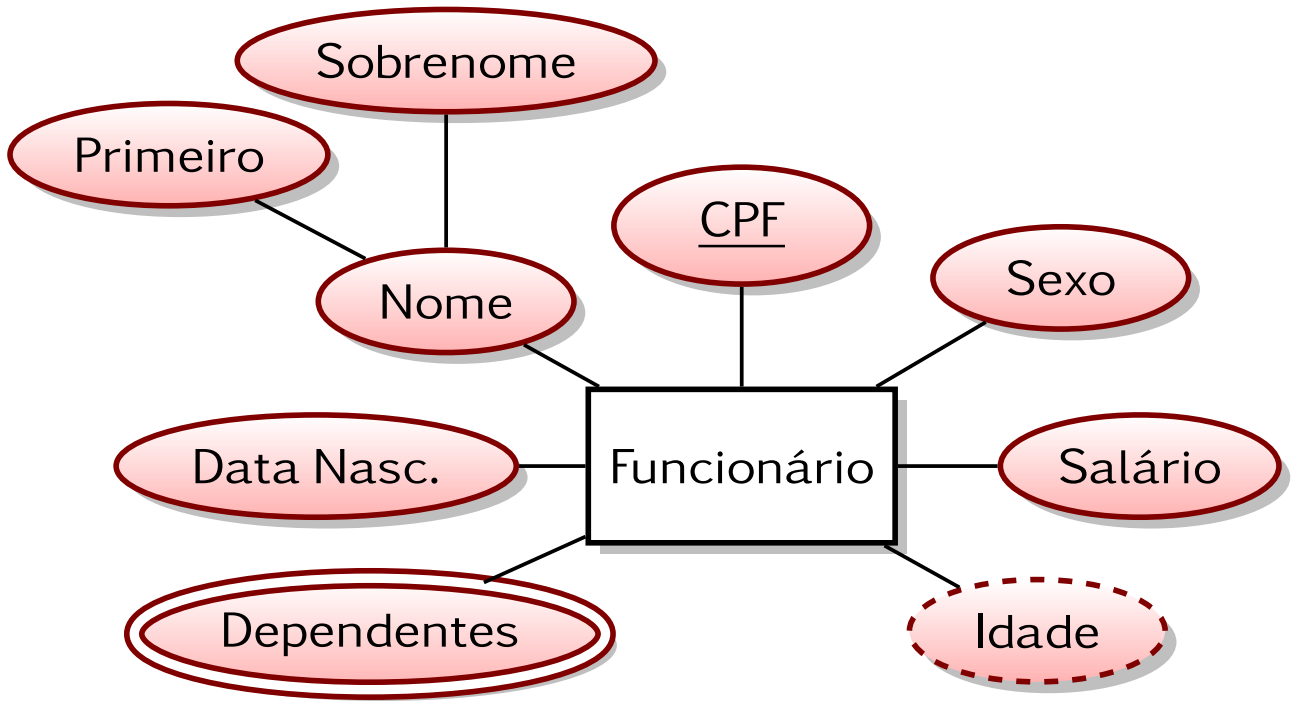
MER: Entidade

- Entidade Fraca → Entidade cuja existência depende da existência de outra entidade



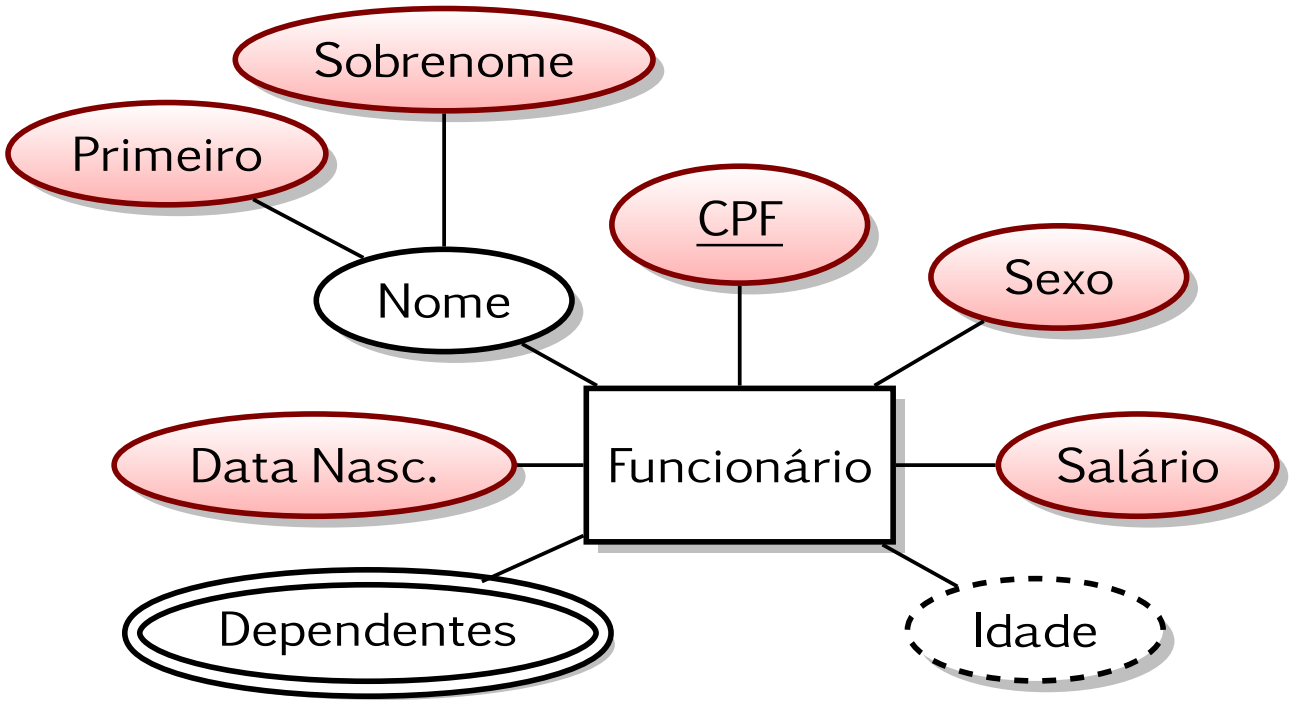
MER: Atributo

- Propriedade que descreve uma característica específica de uma entidade
- DER → Representa-se como uma elipse rotulada e ligada à entidade que ele caracteriza



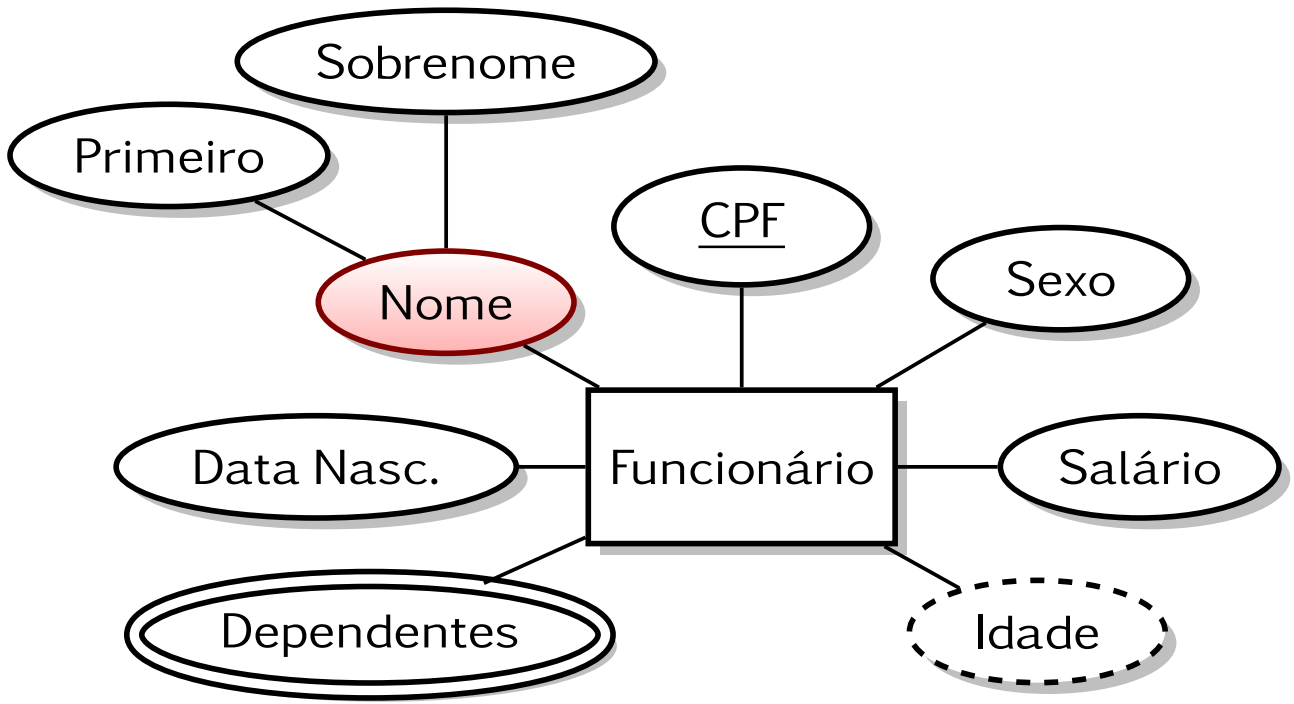
MER: Atributo

- Simples → Indivisível. Representado por uma elipse simples rotulada



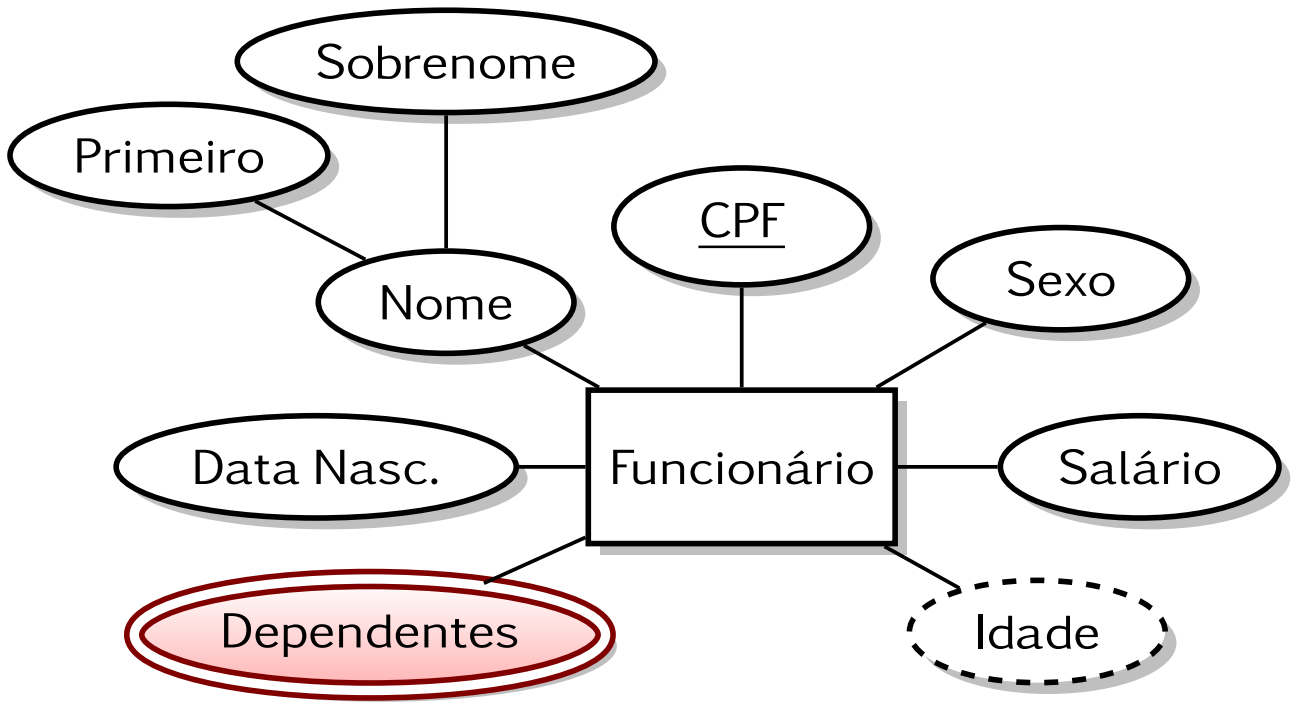
MER: Atributo

- Composto → Desmembra-se em outros atributos. Representado por uma elipse simples rotulada com outros atributos ligados a ele



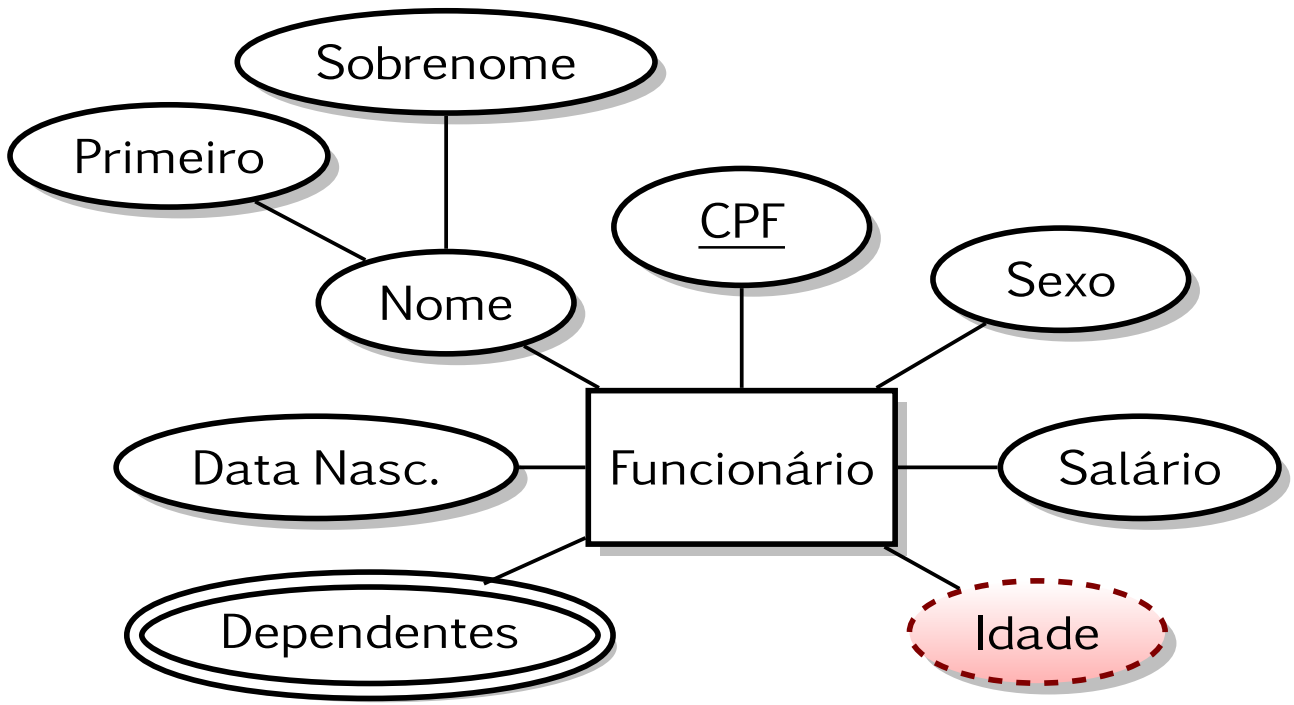
MER: Atributo

- Multivalorado → Conteúdo formado por mais de um valor. Representado por uma elipse rotulada com borda dupla



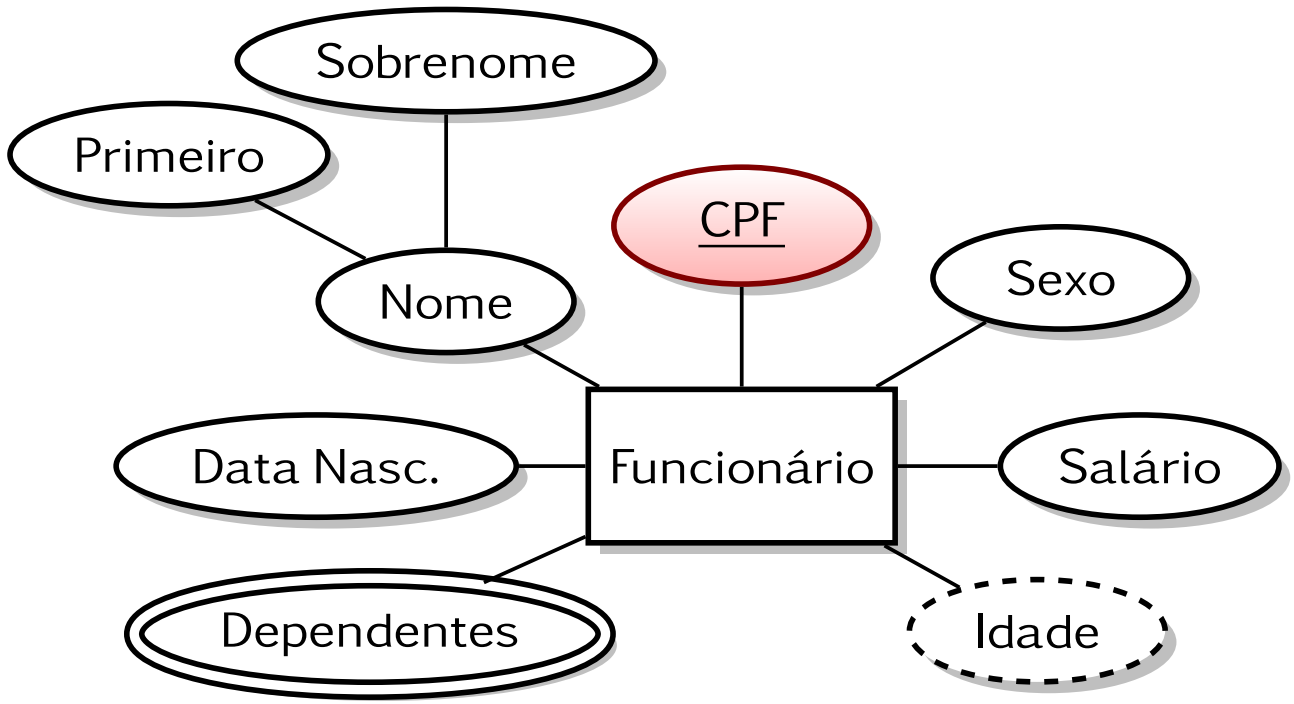
MER: Atributo

- Derivado → Valor pode ser obtido a partir de valores de outros atributos ou relacionamentos entre entidades. Representado por uma elipse rotulada com borda tracejada



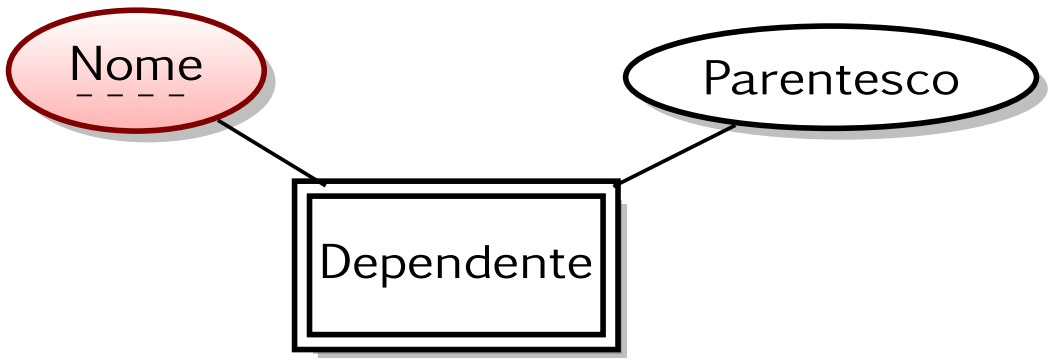
MER: Atributo

- Chave → Atributo ou conjunto de atributos que juntos identificam cada instância de entidade de maneira exclusiva. Representado por uma elipse simples rotulada, com rótulos sublinhados



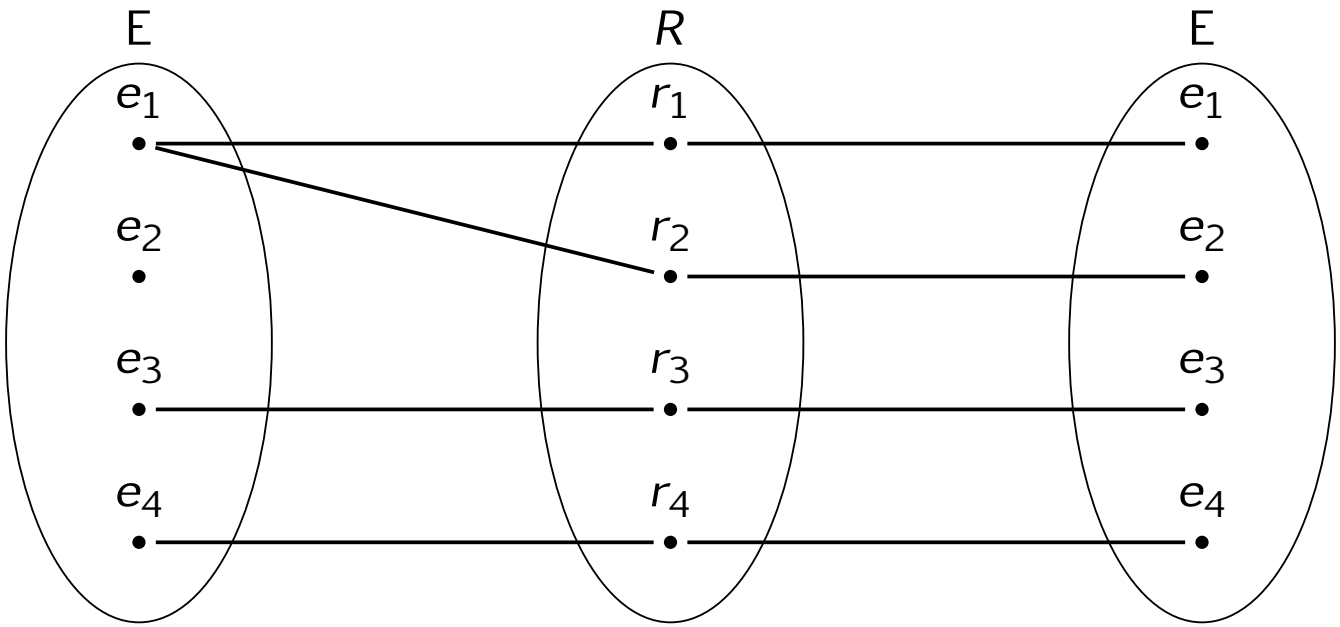
MER: Atributo

- Chave Parcial → Também conhecido como discriminador, trata-se de um atributo ou conjunto de atributos que juntos potencialmente identificam cada instância de entidade (fraca) de maneira exclusiva. Representado por uma elipse simples rotulada, com rótulos sublinhados de maneira tracejada



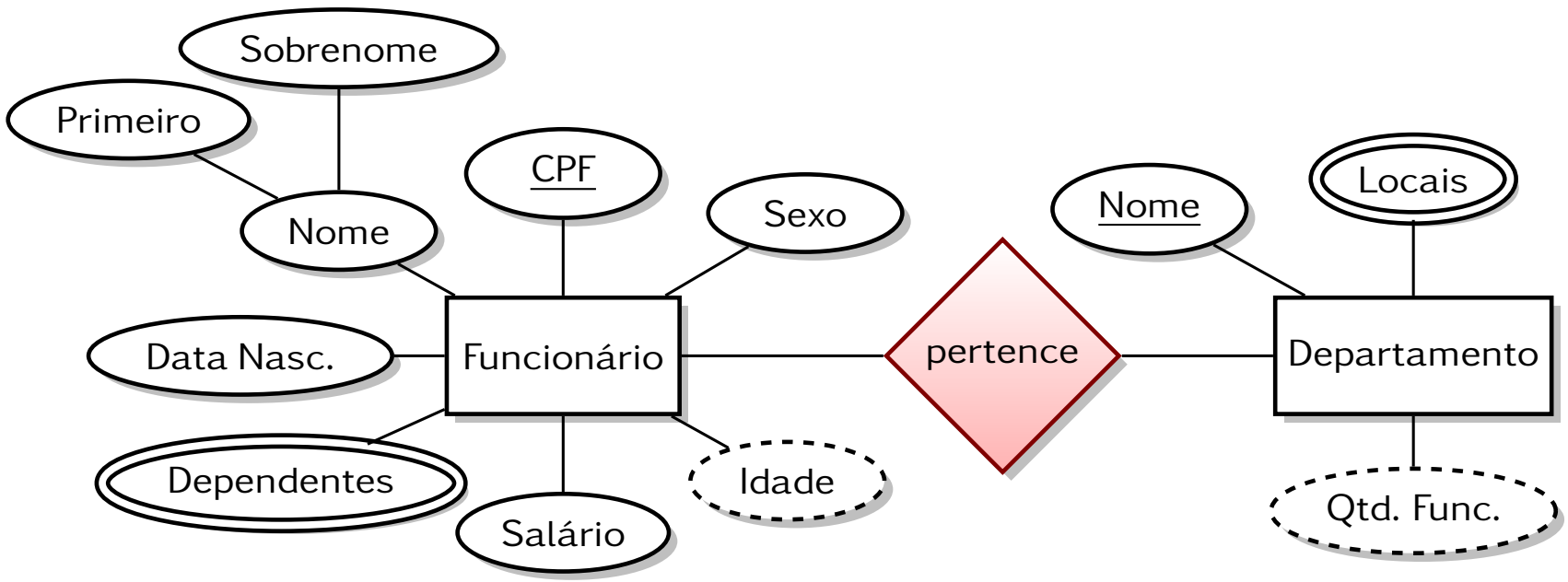
MER: Relacionamento

- Associação entre entidades
- Seja $R = \{r_1, r_2, \cdots, r_k\}$ um conjunto de k associações entre entidades. Define-se:
  - Tipo de Relacionamento (R) $\rightarrow$ Conjunto de instâncias de associações do mesmo tipo
  - Instância de Relacionamento (r_i) $\rightarrow$ Associação específica entre instâncias de entidades, tal que $r_i \in R$



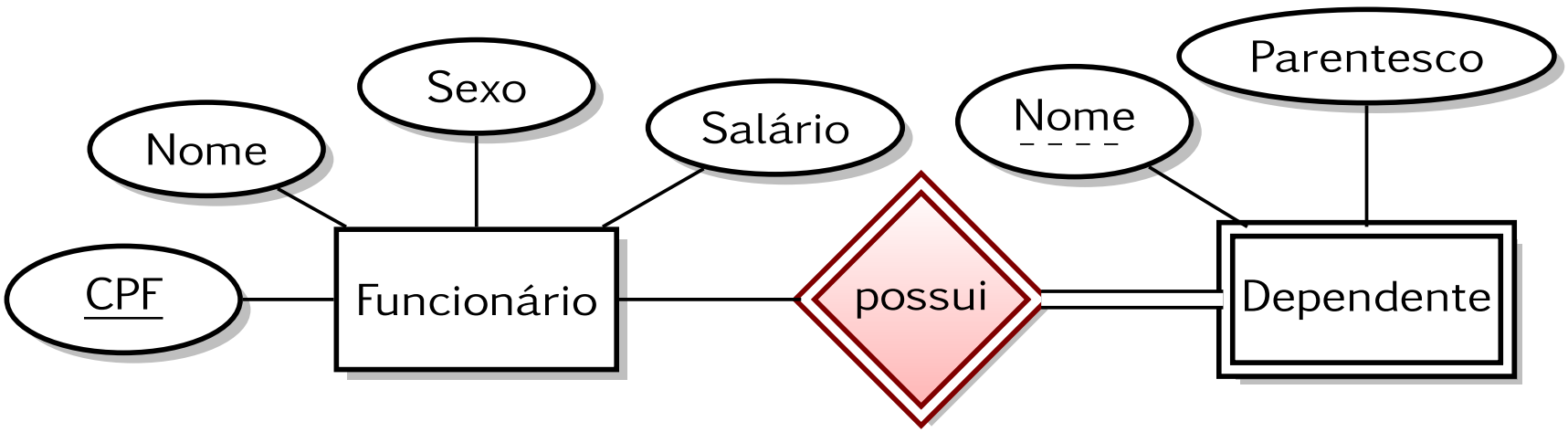
MER: Relacionamento

- DER → Representa-se um tipo de relacionamento, ou simplesmente relacionamento, como um losango rotulado



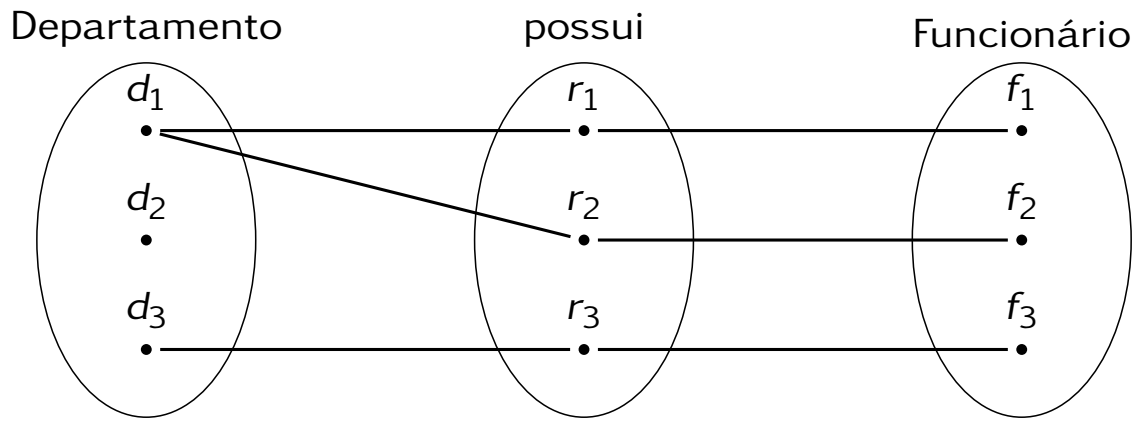
MER: Relacionamento

- Relacionamento Fraco → Também conhecido como RELACIONAMENTO DE DEPENDÊNCIA, consiste na associação envolvendo ao menos uma entidade fraca. Representado por um losango rotulado com borda dupla



MER: Relacionamento

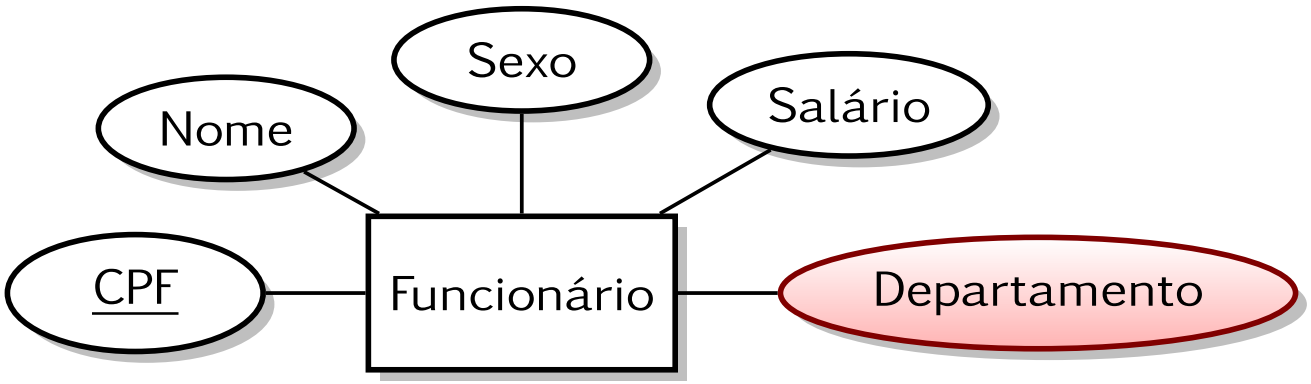
- Cada relacionamento $r_i \in R$ é uma associação entre entidades que inclui exatamente uma única entidade de cada tipo de entidade participante



- Grau do Relacionamento → Número de entidades participantes do relacionamento
  - Relacionamento Binário → Relacionamento de grau 2
  - Relacionamento Ternário → Relacionamento de grau 3
  - Relacionamento Quaternário → Relacionamento de grau 4

MER: Relacionamento

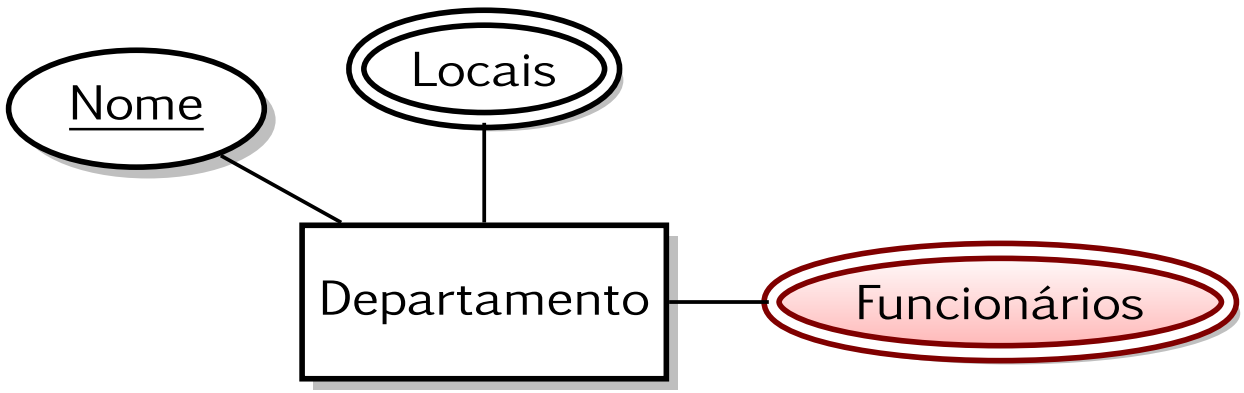
- Relacionamento como Atributo Simples → O valor do atributo de uma instância de entidade referencia o valor do atributo de uma instância de outra entidade



- Por exemplo, para uma instância de *Funcionário*, o valor do atributo *Departamento* contém o valor do atributo *Nome* de uma instância de *Departamento*

MER: Relacionamento

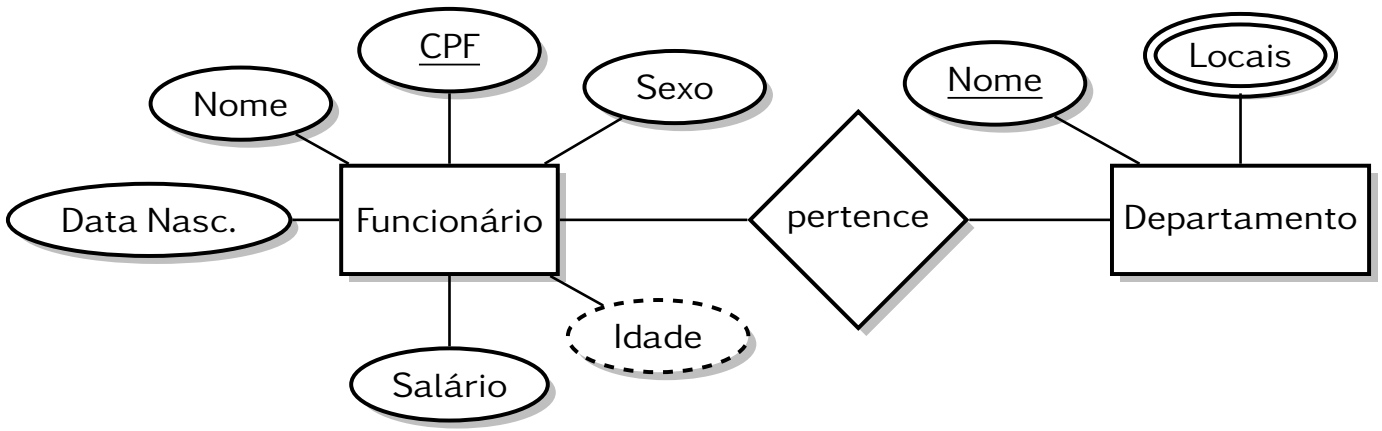
- Relacionamento como Atributo Multivalorado → O valor do atributo de uma instância de entidade referencia o valor do atributo de múltiplas instâncias de outra entidade



- Por exemplo, para uma instância de *Departamento*, o valor do atributo *Funcionários* contém os valores do atributo *Nome* de múltiplas instâncias de *Funcionários*

MER: Relacionamento

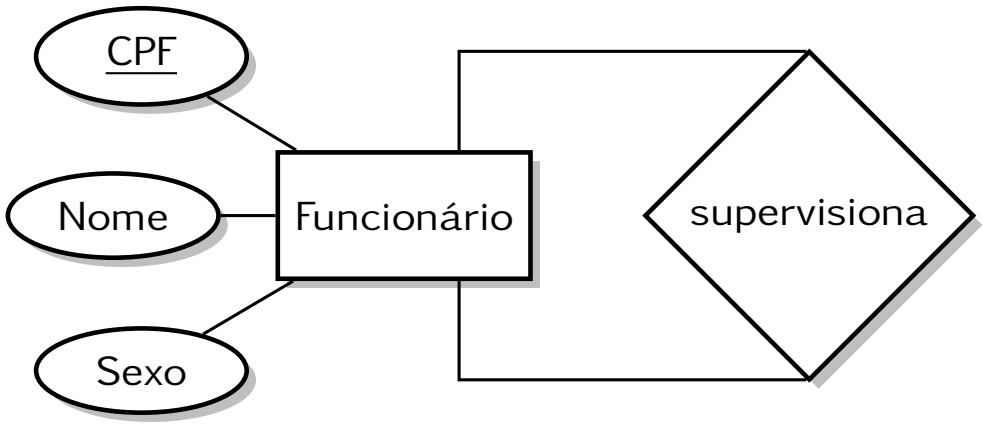
- Nome de Função → Rotula o tipo de relacionamento e representa a tarefa que uma entidade desempenha em cada relacionamento
- Auxilia no entendimento da semântica do tipo de relacionamento



- No relacionamento *pertence*, a entidade *Funcionário* desempenha a função "pertencedor", enquanto *Departamento* desempenha a função "possuidor"

MER: Relacionamento

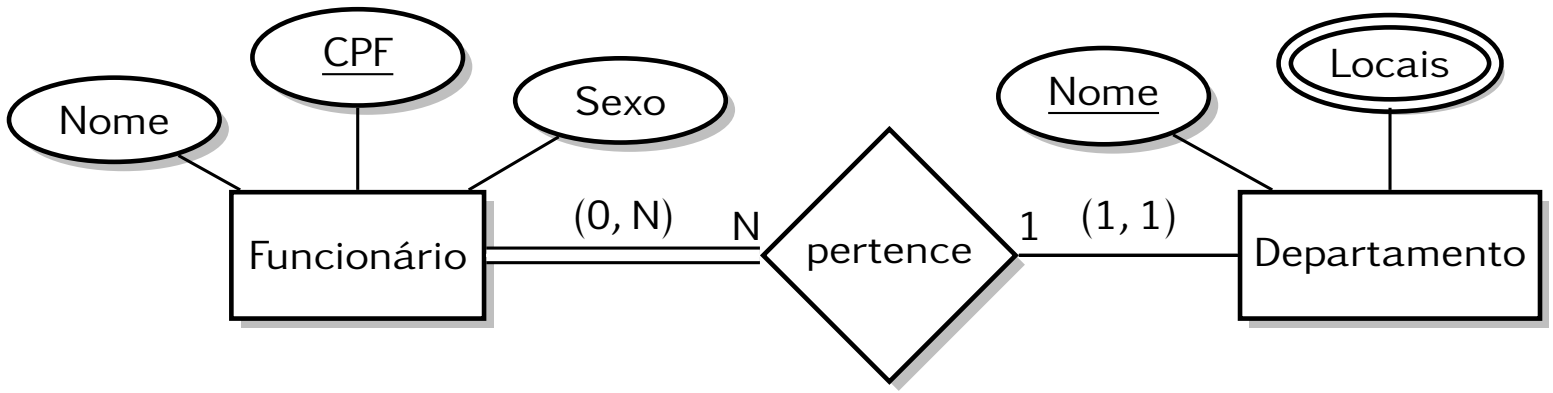
- Relacionamento Recursivo → A mesma entidade participa mais de uma vez, com funções diferentes, em um relacionamento



- No exemplo acima, a entidade *Funcionário* participa com as funções de "supervisor" e "supervisionado" no relacionamento *supervisiona*

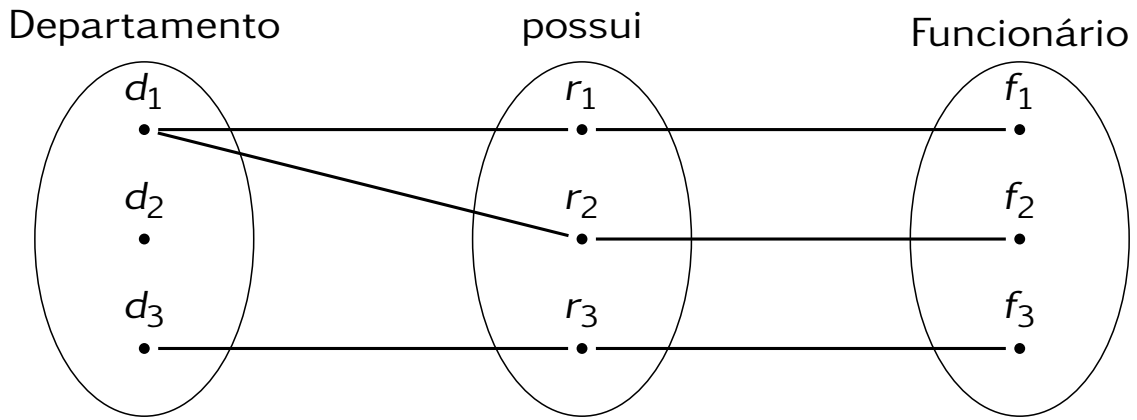
MER: Relacionamento

- Restrições → Limitam as possibilidades de associação entre entidades nos relacionamentos;
  - Razão de Participação → Especifica se a participação de uma entidade no relacionamento é parcial ou total
  - Razão de Cardinalidade → Especifica o número máximo de relacionamentos nos quais uma entidade pode participar: *1:1*, *1:N* ou *N:N*. Pode também indicar os limites mínimos e máximos: *(min, max)*

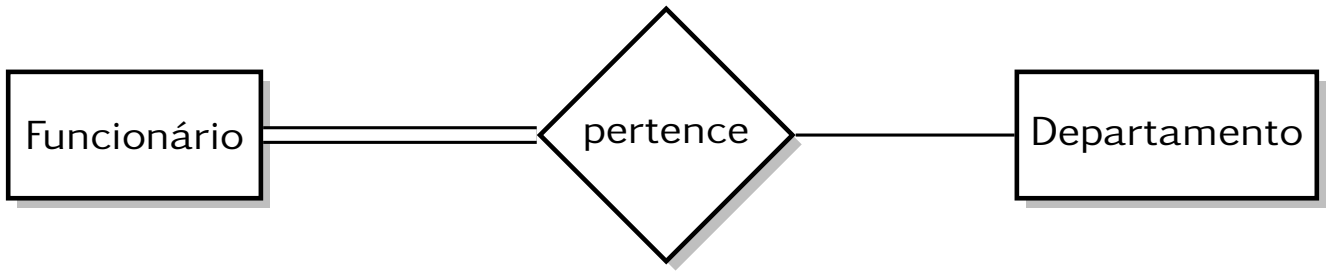


MER: Relacionamento

- Restrição de Participação Total → Todas as instâncias da entidade devem obrigatoriamente participar de relacionamentos, como a entidade *Funcionário* abaixo

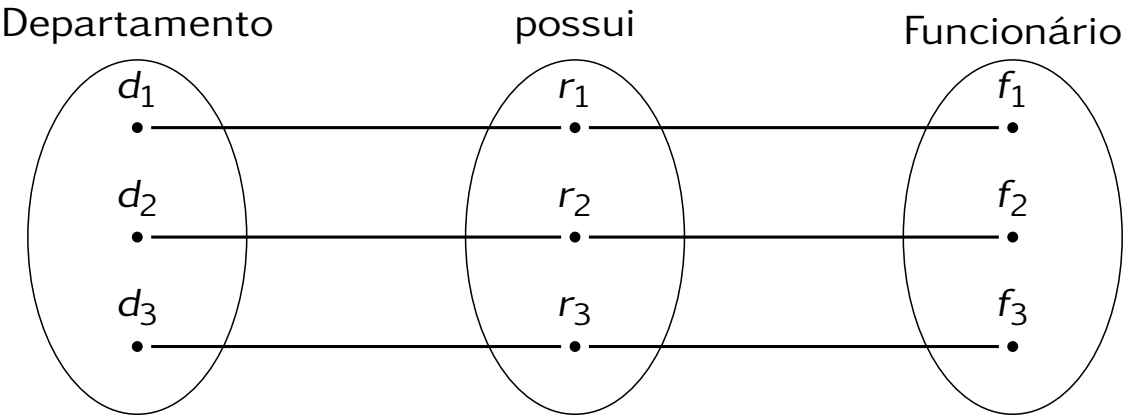


- DER → Representado por uma linha dupla entre a entidade e o relacionamento

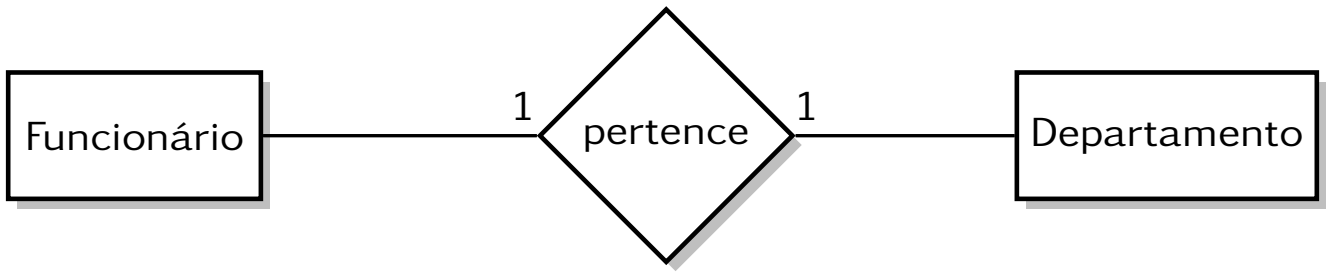


MER: Relacionamento

- Restrição de Cardinalidade *1:1* → Uma instância de cada entidade participante só pode participar de um único relacionamento

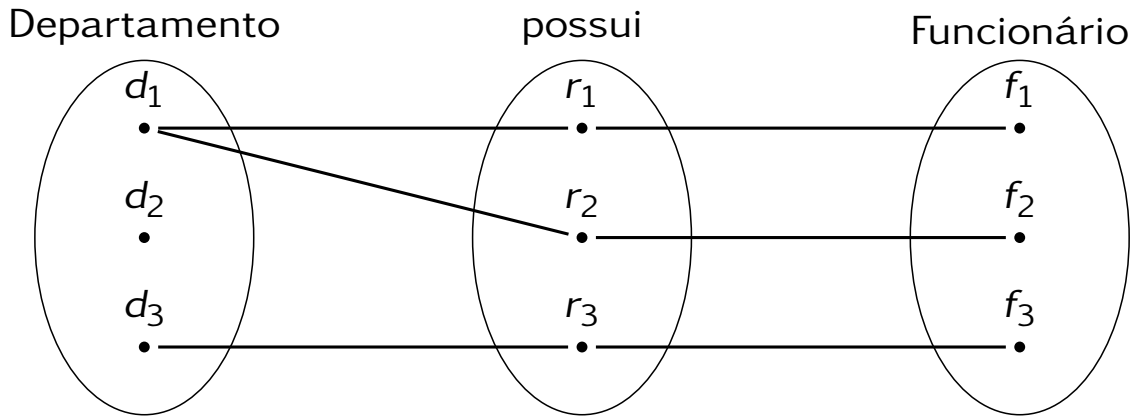


- DER → Representado por rótulos *1* nas duas extremidades do relacionamento

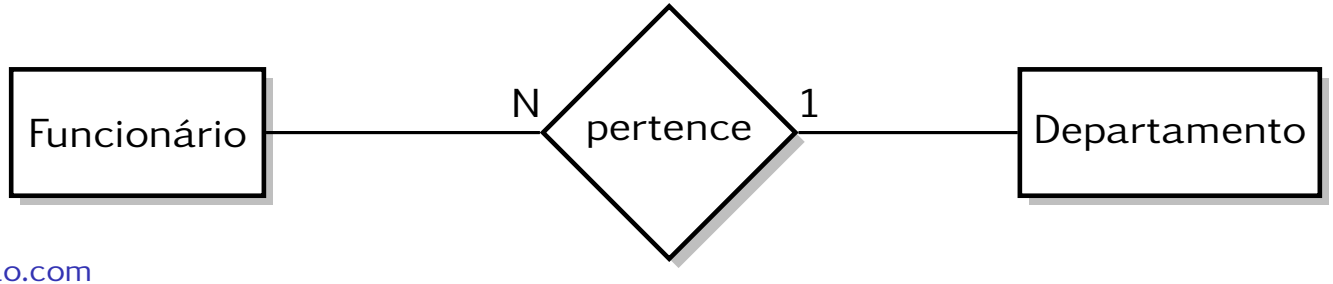


MER: Relacionamento

- Restrição de Cardinalidade *1:N* → Uma instância de uma entidade participante só pode participar de um único relacionamento, enquanto que uma instância da outra entidade participante pode participar de múltiplos

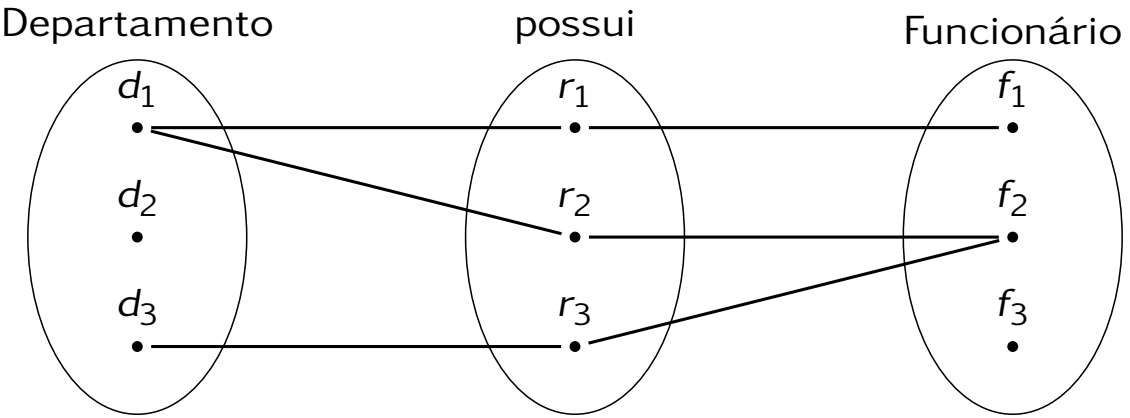


- DER → Representado por rótulos *1* em uma extremidade e *N* na outra extremidade do relacionamento

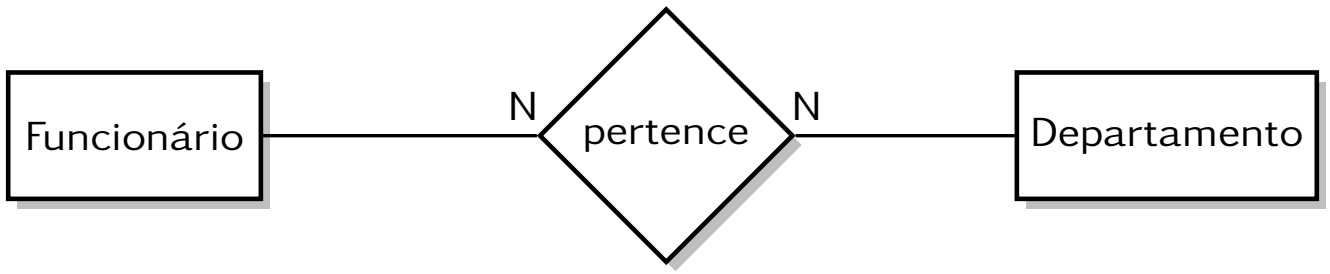


MER: Relacionamento

- Restrição de Cardinalidade *N:N* → Uma instância de cada entidade participante pode participar de múltiplos relacionamentos

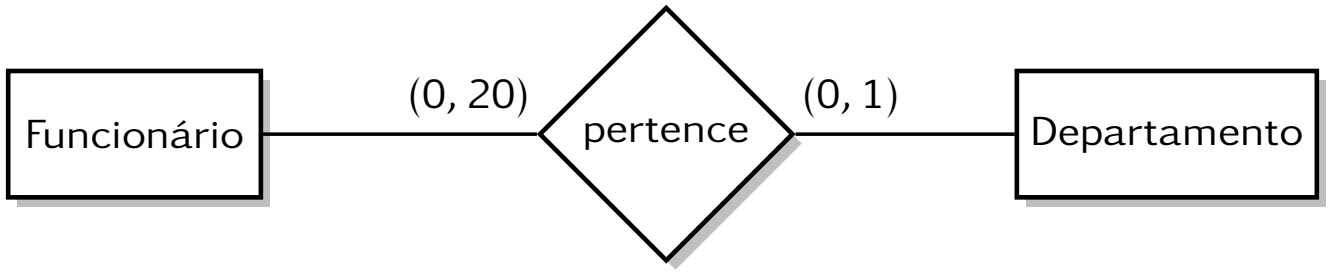


- DER → Representado por rótulos *N* nas duas extremidades do relacionamento



MER: Relacionamento

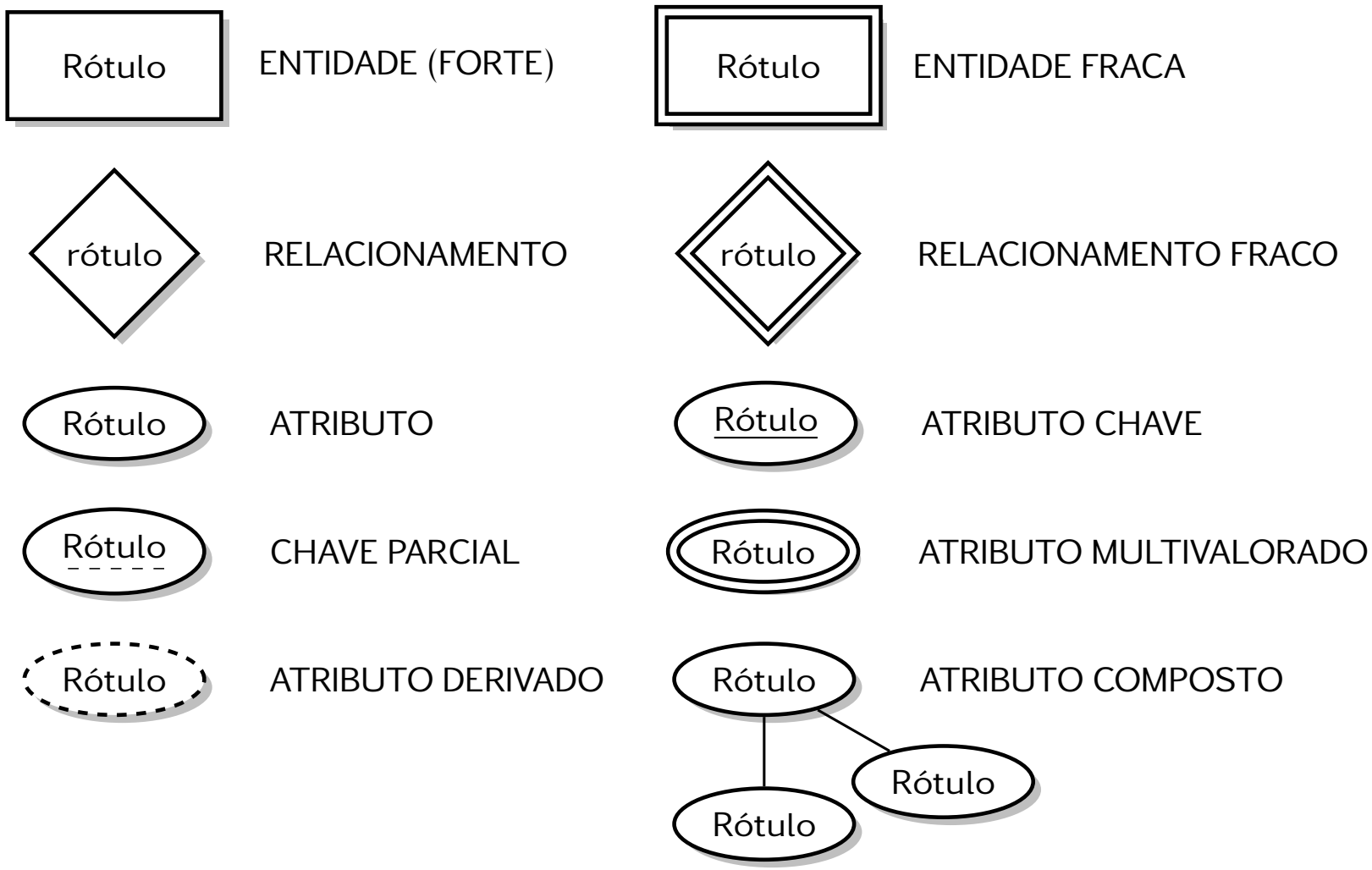
- Limites Mínimos e Máximos → Opcionalmente às notações *1:1*, *1:N* e *N:N* que definem apenas os limites máximos como *1* ou *N*, pode-se definir limites mínimos e máximos para os relacionamentos
- DER → Representado por rótulos *(min, max)* nas duas extremidades do relacionamento



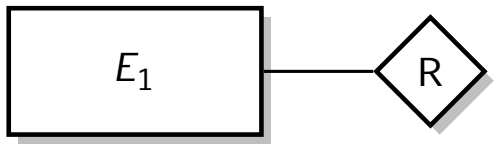
Convenção de Nomes

- Em uma especificação textual de minimundo:
  - Substantivos → Indicam tipos de entidade ou atributos
  - Verbos → Indicam tipos de relacionamento
- Para construção do DER deve-se adotar uma convenção. Por exemplo:
  - Tipo de Entidade → Nome no singular com letra inicial em maiúscula
  - Tipo de Relacionamento → Nome no singular com todas as letras minúsculas
  - Atributo → Nome com a letra inicial em maiúscula. Nome no plural para atributos multivalorados

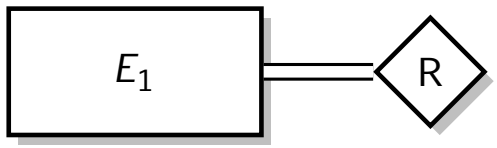
DER: Elementos



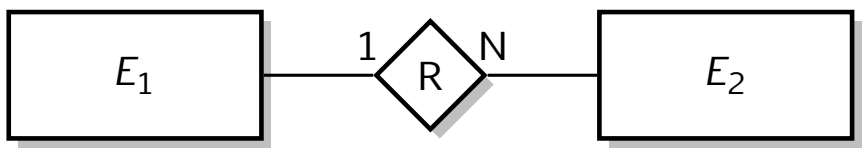
DER: Elementos



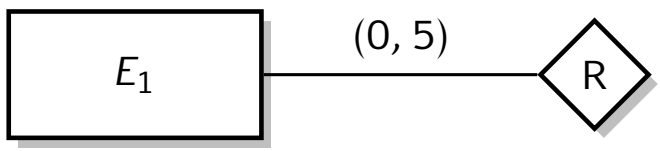
PARTICIPAÇÃO PARCIAL DE E_1 EM R



PARTICIPAÇÃO TOTAL DE E_1 EM R



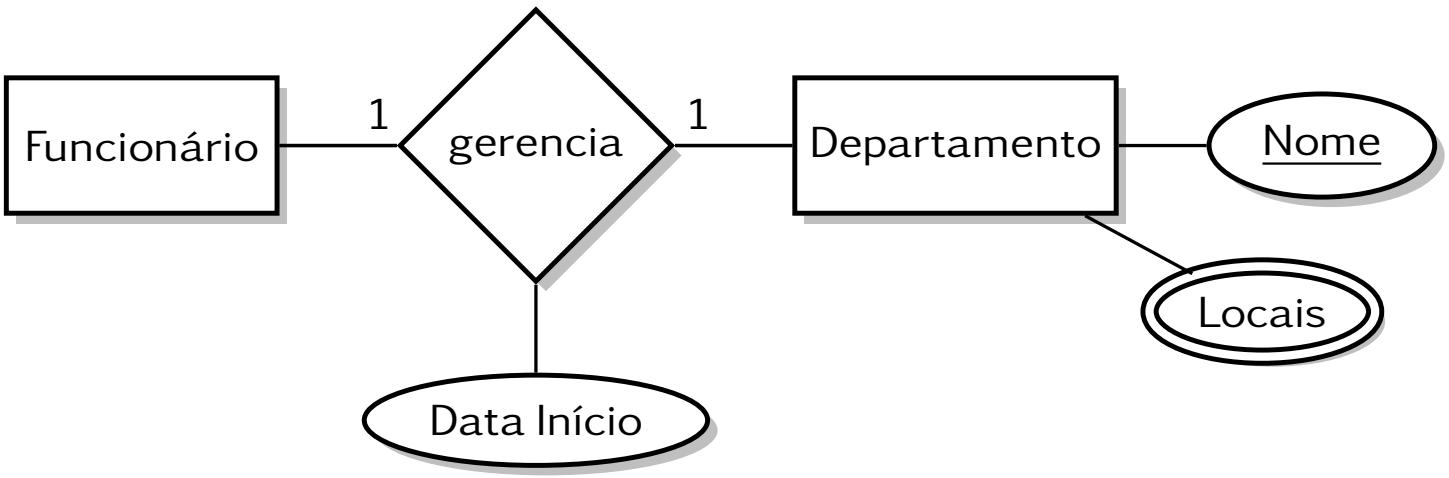
CARDINALIDADE 1:N DE E_1 PARA E_2



LIMITES DE CARDINALIDADE DE E_1 EM R

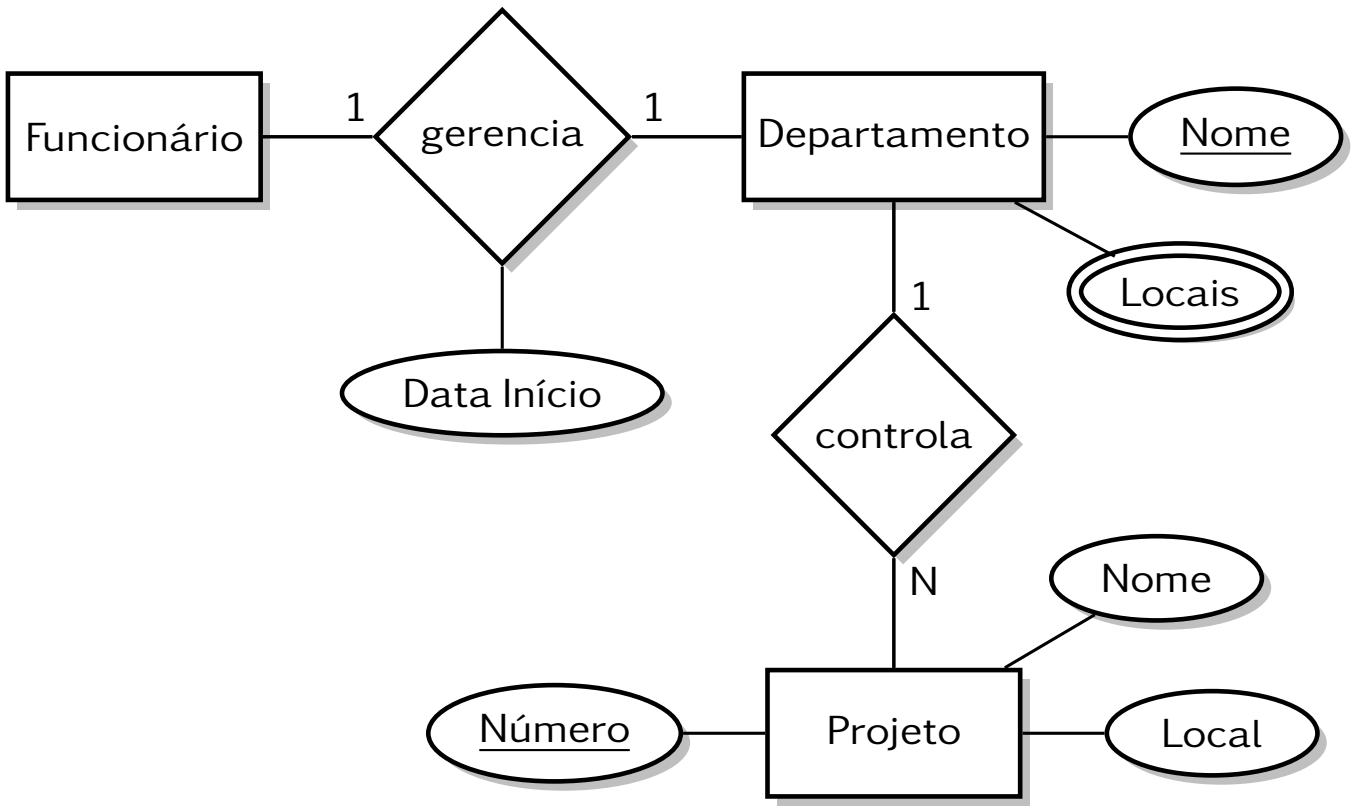
Exemplo de MER: DER - EMPRESA

- **Minimundo** → A empresa é organizada em departamentos, sendo que cada departamento tem um nome exclusivo e um funcionário em particular que o gerencia. Cada funcionário gerencia apenas um departamento e registramos a data inicial em que o funcionário começou a gerenciar o departamento. Um departamento pode ter vários locais.



Exemplo de MER: DER - EMPRESA

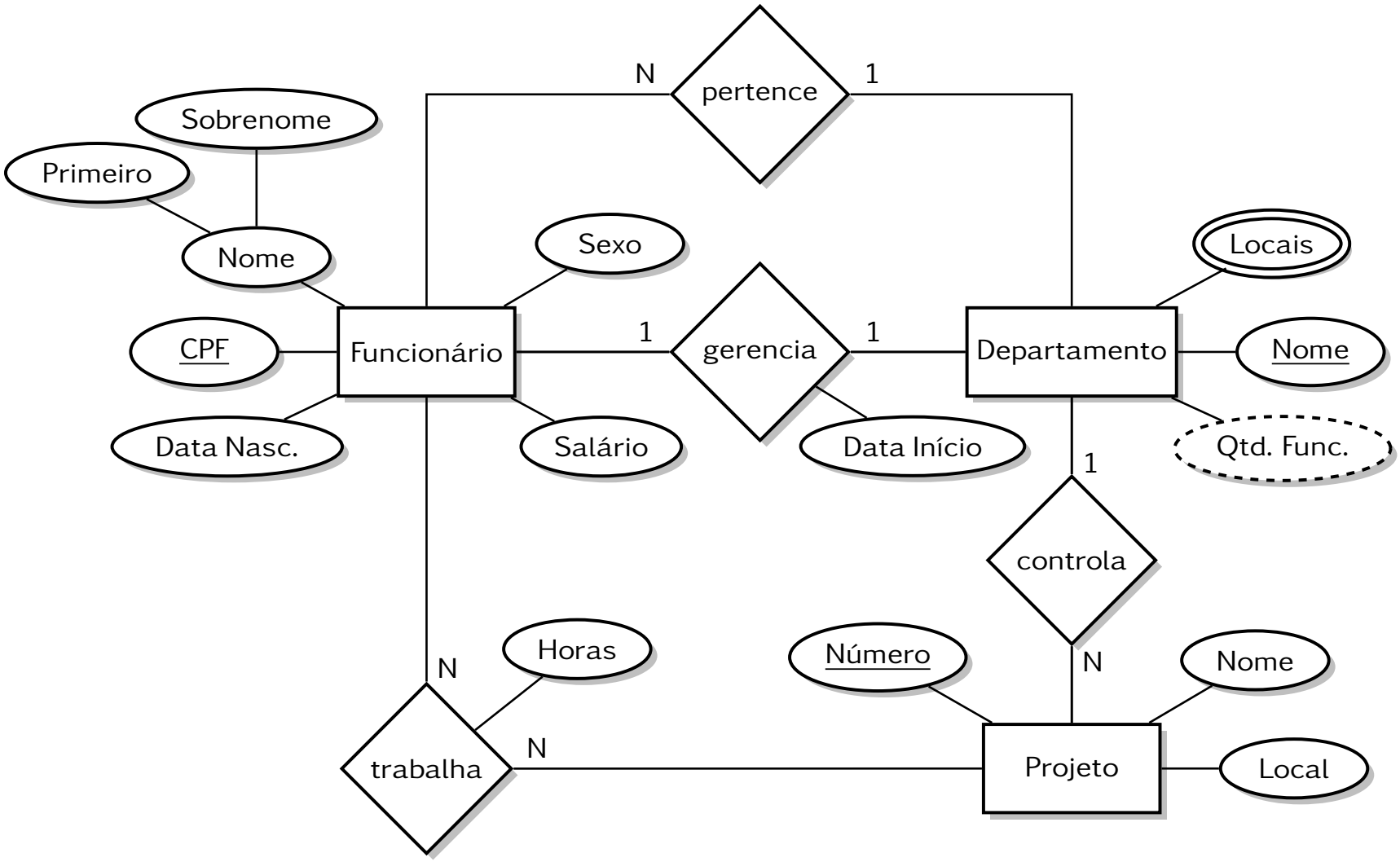
- **Minimundo** → Um departamento controla uma série de projetos, sendo que um projeto é controlado por apenas um departamento. Cada projeto possui um número exclusivo, um nome e um local



Exemplo de MER: DER - EMPRESA

- **Minimundo** → Para cada funcionário, registramos o CPF, nome (composto por primeiro nome e sobrenome), salário, sexo e data de nascimento. Um funcionário pertence a um departamento e um departamento possui vários funcionários, sendo que a quantidade de funcionários que um departamento possui deriva dessa relação e deve ser registrada no departamento. Um funcionário pode trabalhar em vários projetos e em um projeto podemos ter vários funcionários trabalhando. Para cada funcionário trabalhando em um projeto, registramos as horas trabalhadas.

Exemplo de MER: DER - EMPRESA



Seção 08

MER Estendido

MER-E: MER Estendido

- Consiste no modelo entidade relacionamento (ER) incorporando conceitos adicionais de modelagem semântica de dados.
- Modelo ER aprimorado que inclui todos os conceitos de modelagem do modelo ER.
- Possui requisitos mais complexos e precisos
  - Supertipos e subtipos
  - Generalização e Especialização
  - Herança
  - Restrições complexas

Subclasses e superclasses

Uma entidade pode possuir **subtipos** *significativos* e que precisam ser *representados explicitamente*:

→ **FUNCIONARIO**

- SECRETARIA
- ENGENHEIRO
- GERENTE
- TECNICO
- FUNCIONARIO_MENSAL
- FUNCIONARIO_HORISTA

Subclasses e superclasses

- **Subtipo** ou **subclasse** são subagrupamentos de uma entidade (chamada de **superclasse** ou **supertipo**).

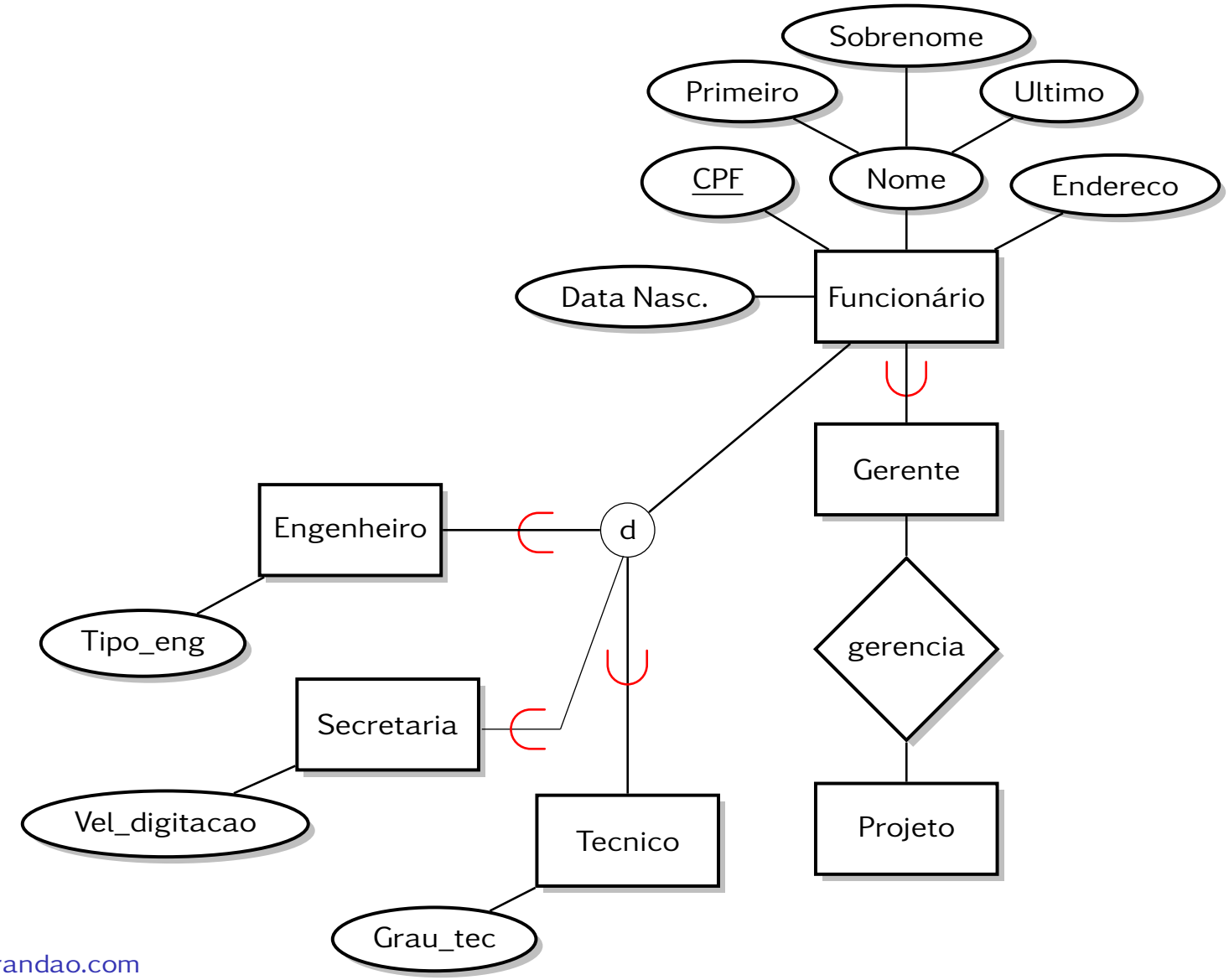
Esse relacionamento é denominado *superclasse/subclasse*, *supertipo/subtipo* ou *classe/subclasse*:

→ **FUNCIONARIO *(supertipo)***

- SECRETARIA *(subtipo)*
- ENGENHEIRO *(subtipo)*
- GERENTE *(subtipo)*
- TECNICO *(subtipo)*
- FUNCIONARIO_ENSAL *(subtipo)*
- FUNCIONARIO_HORISTA *(subtipo)*

Subclasses e superclasses

- **Representação nos diagramas ER Estendido:**



Herança de Tipo

- A entidade de uma subclasse **herda** todos os atributos e relacionamentos da superclasse.
- Uma entidade na subclasse possui *atributos específicos* assim como valores de *atributos da superclasse*.

Especialização

- **Especialização** é o processo de definir um *conjunto de subclasses* de uma entidade com base em alguma característica:

Caso I - Tipo de Cargo

→ **FUNCIONARIO**

- SECRETARIA
- ENGENHEIRO
- TECNICO

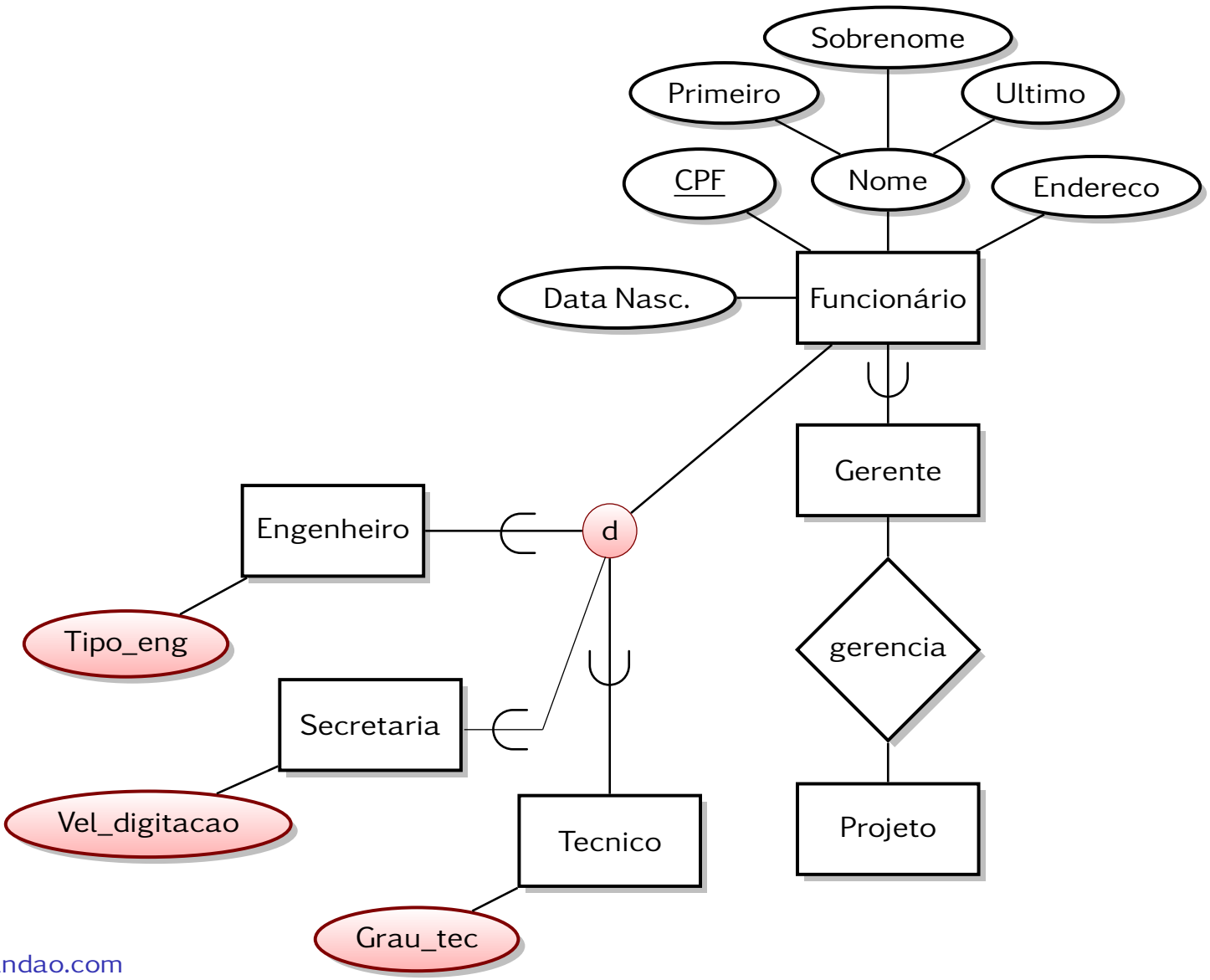
Caso II - Método de Pagamento

→ **FUNCIONARIO**

- FUNCIONARIO_MENSAL
- UNCIONARIO_HORISTA

Especialização

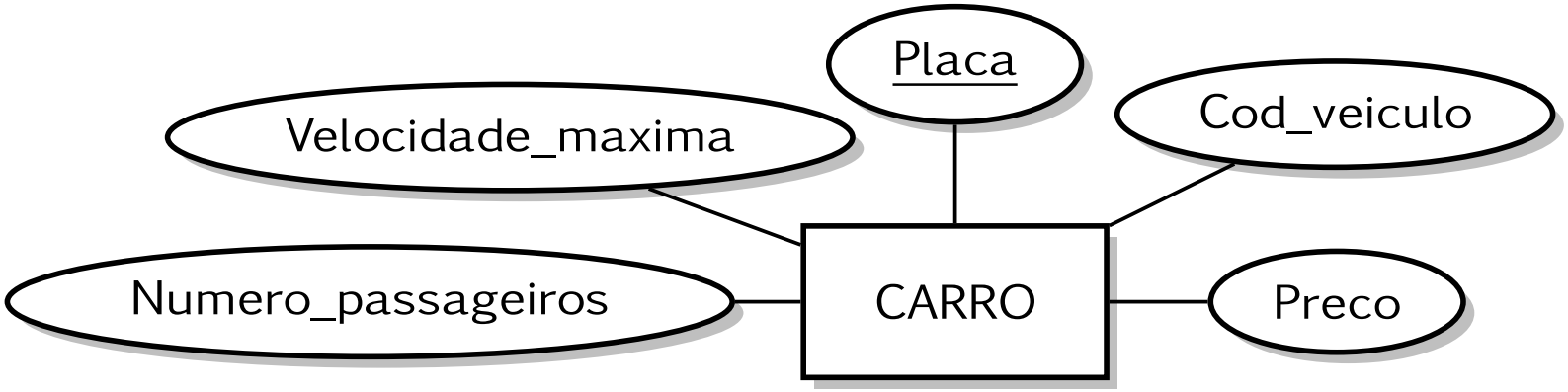
- **Representação nos diagramas ER Estendido:**

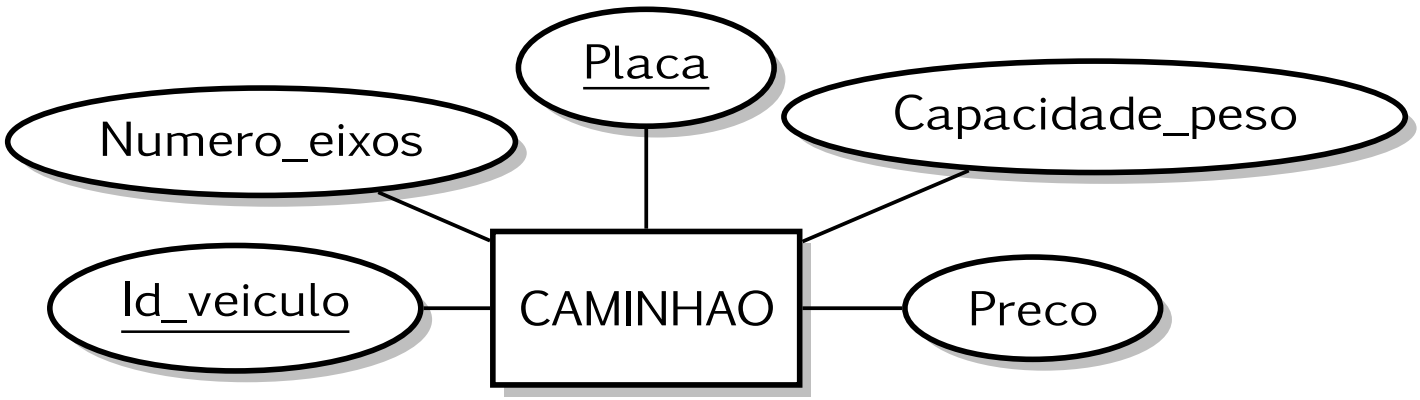


Generalização

- Generalização é a definição de um **tipo de entidade generalizado** com base nos tipos de entidades fornecidos.
- Identifica-se *características comuns* e generaliza-se em uma *única superclasse*.

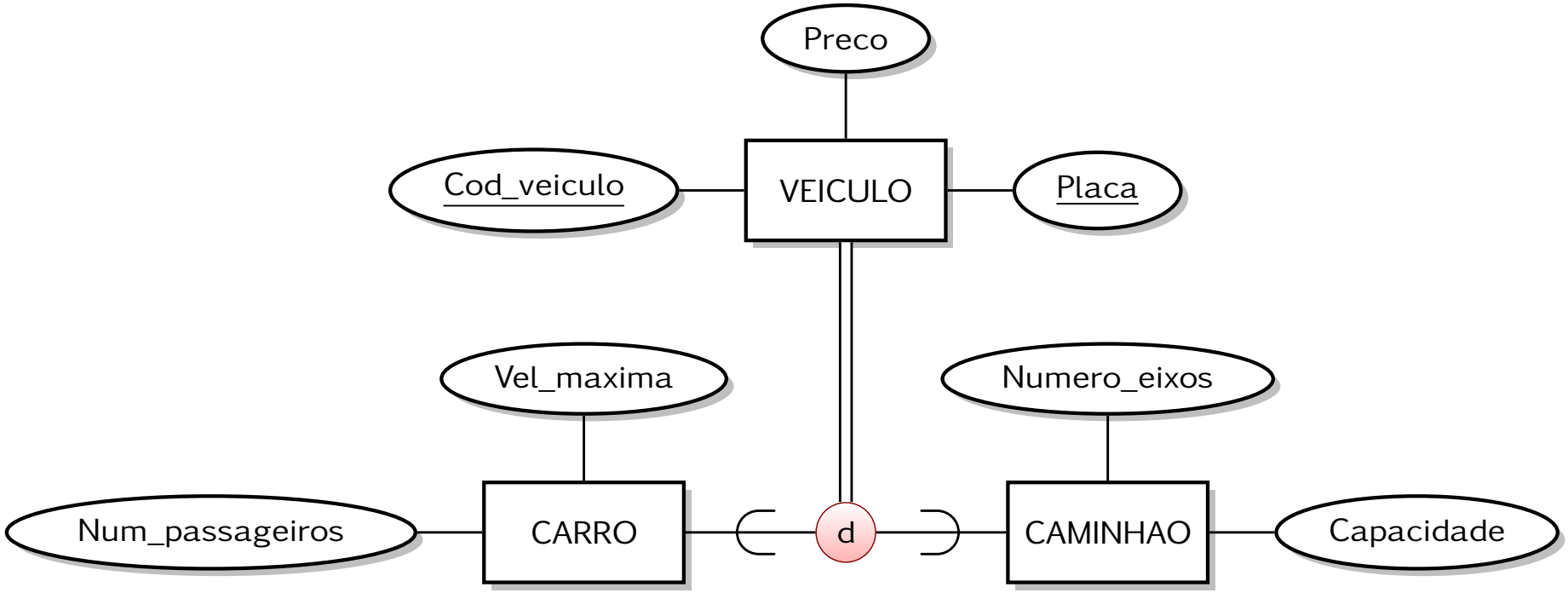
Generalização





CARRO e *CAMINHAO* possuem vários **atributos comuns.**

Generalização

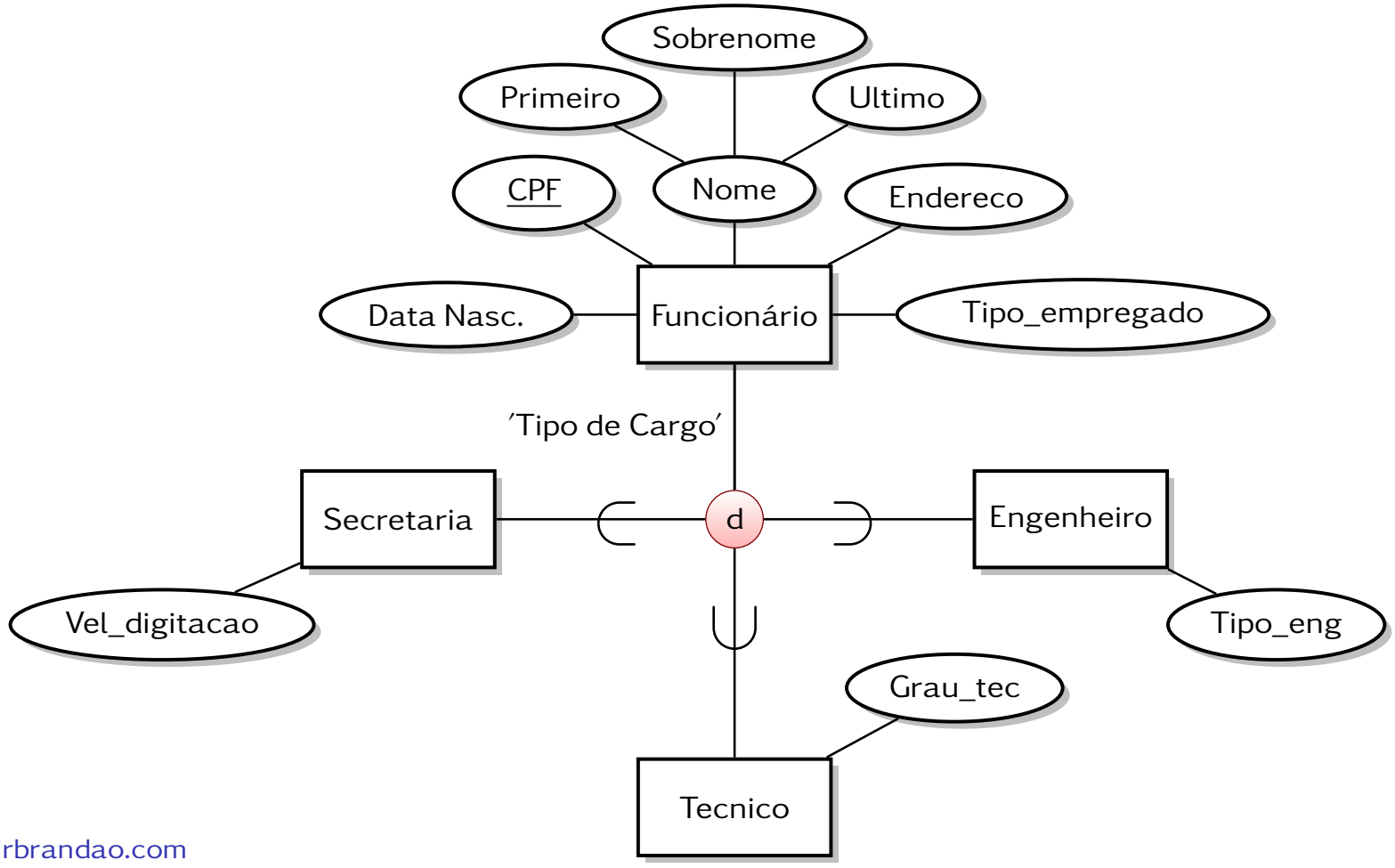


- *CARRO* e *CAMINHAO* podem ser generalizados no tipo de entidade *VEICULO*;
- *CARRO* e *CAMINHAO* são subclasses da **superclasse generalizada** *VEICULO*.

Restrições sobre especialização e generalização

Subclasse definida por condição

- Determinam-se as entidades que farão parte da subclasse ao definir uma condição em um atributo:



Restrições sobre especialização e generalização

Subclasse definida por condição

- A condição é determinada quando se aplica a operação para incluir uma entidade a subclasse.
- É especificada individualmente para cada entidade pelo usuário.
- Nesse caso, não se tem uma condição qualquer que possa ser avaliada automaticamente.

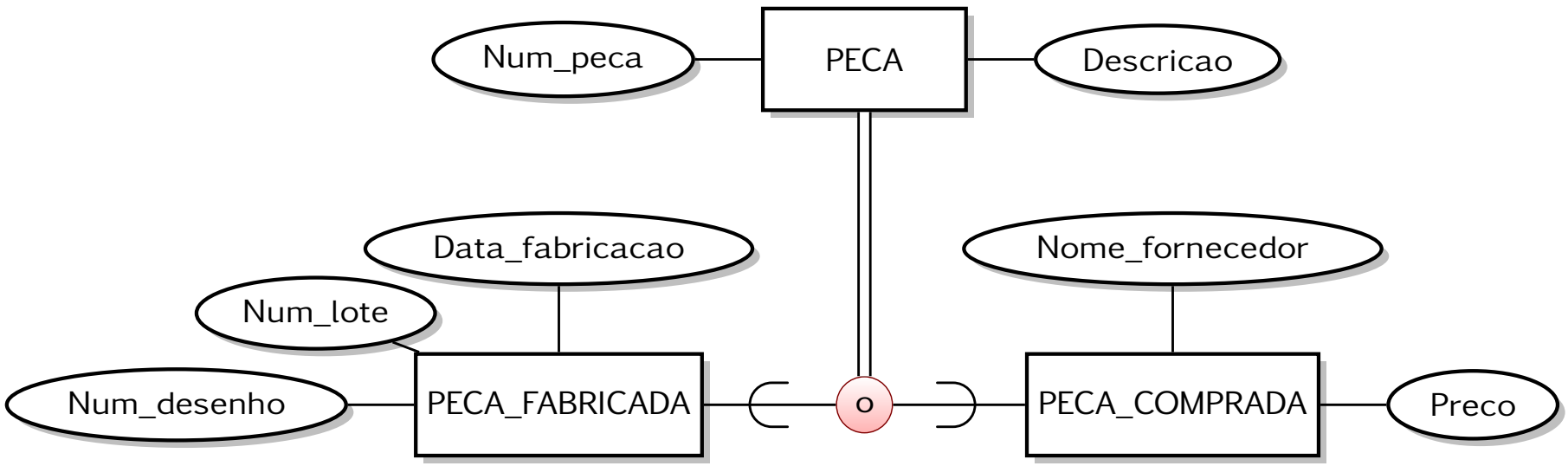
Restrições sobre especialização e generalização

Restrição de disjunção (ou desconexão)

- Uma entidade pode ser membro de *no máximo* uma das subclasses da especialização.
- Uma especialização que é definida por um *atributo de valor único* implica a restrição de disjunção.

Conjuntos de Entidades Sobrepostos

- Uma entidade como membro de mais de uma subclasse da especialização:



Restrições sobre especialização e generalização

Restrição de completude (totalidade)

- Restrição de ***especialização total*** especifica que toda entidade na superclasse precisa ser um membro de pelo menos uma subclasse na especialização.
- Restrição de ***especialização parcial*** permite que uma entidade não pertença a qualquer uma das subclasses.

Restrições sobre especialização e generalização

Temos quatro restrições possíveis da especialização:

- Disjunção, total.
- Disjunção, parcial.
- Sobreposição, total.
- Sobreposição, parcial.

Hierarquias e reticulado da especialização e generalização

A subclasse pode ter mais subclasses, formando uma **hierarquia** ou um **reticulado de especializações**:

- Cada subclasse tem apenas um pai, que resulta em uma ***estrutura de árvore*** ou ***hierarquia estrita***.
- Para um ***reticulado de especialização***, uma subclasse pode ser uma subclasse em mais de um relacionamento.

Diferenças entre herança múltipla e simples

- Na **herança múltipla** a subclasse herda diretamente atributos e relacionamentos de múltiplas classes (*subclasse compartilhada*).
- Enquanto na **herança simples** não existe qualquer subclasse compartilhada.

Diferenças entre especialização e generalização

- Na especialização inicia-se com uma entidade e depois é definido subclasses pela especialização sucessiva - **processo de refinamento conceitual de cima para baixo (*top-down*)**.

- Na generalização é possível chegar a mesma hierarquia ou reticulado de outra direção - **síntese conceitual de baixo para cima (*bottom-up*)**.

Em termos estruturais, o resultado de ambos processos podem ser idênticos.

Modelando tipos de UNIÃO usando categorias

- **Tipo de união** ou **categoria**: subclasse que representa uma *coleção de objetos* que é um *subconjunto da UNIAO de entidades* distintas.

Um único relacionamento de superclasse/subclasse com mais de uma superclasse que representam entidades distintas.

Banco de Dados para registro de veículo a motor

→ **Entidades**

- PESSOA
- BANCO
- EMPRESA

O proprietário de um veículo pode ser uma pessoa, um banco ou uma empresa.

Banco de Dados para registro de veículo a motor

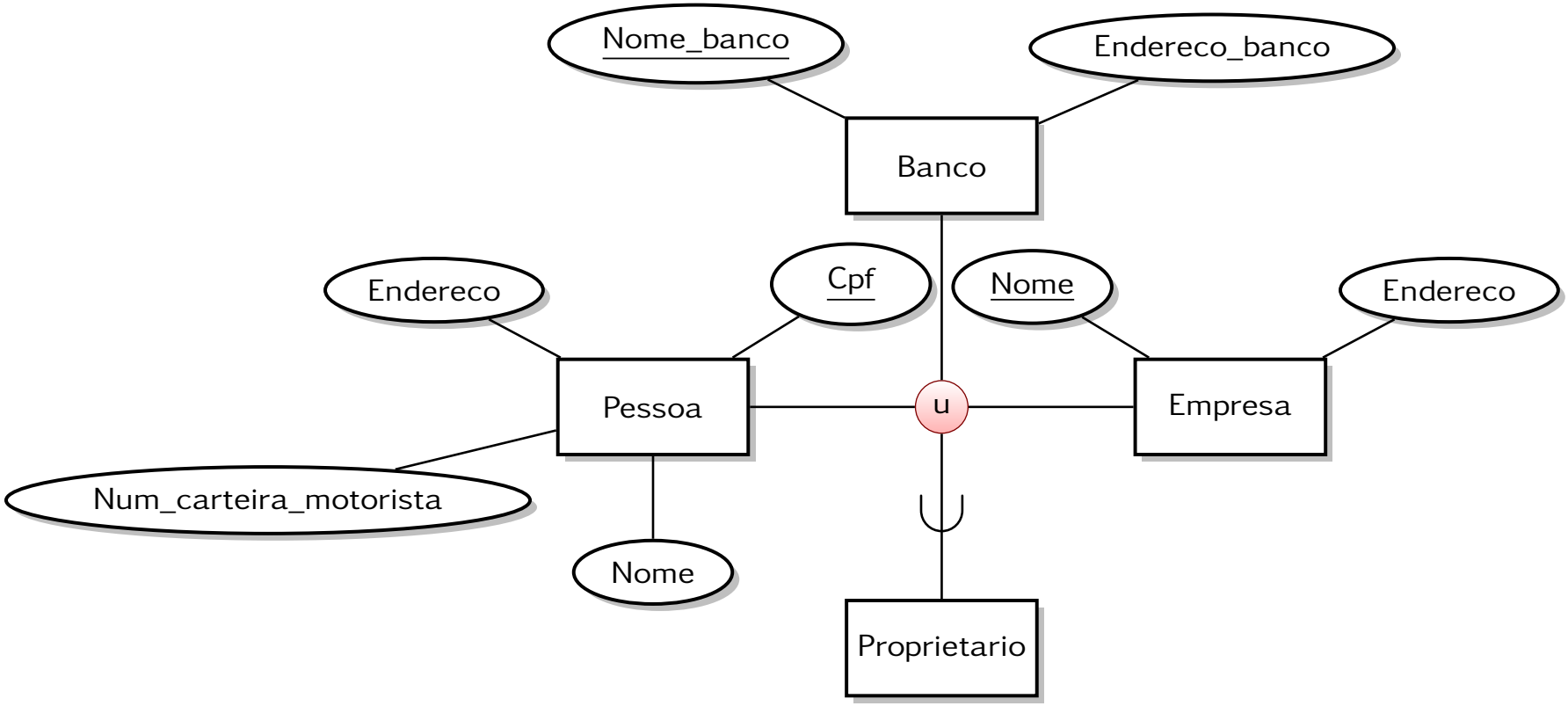
- Criar uma **classe (coleção de entidades)** que inclui os três tipos para desempenhar o papel de proprietário do veículo.

→ PROPRIETARIO ***(tipo de união)***

- PESSOA *(entidade)*
- BANCO *(entidade)*
- EMPRESA *(entidade)*

O proprietário de um veículo pode ser uma pessoa, um banco ou uma empresa.

Banco de Dados para registro de veículo a motor



Cada entidade PROPRIETARIO **herda** os atributos de uma EMPRESA, uma PESSOA ou um BANCO.

Modelando tipos de UNIÃO usando categorias

Uma categoria pode ser total ou parcial:

Categoria total

- Mantém a união de todas as entidades em suas superclasses;
- Representada por uma linha dupla;

Categoria parcial

- Pode manter um subconjunto da união;
- Representada por um linha simples.

Escolhas de projeto para especialização/generalização

- O projeto conceitual de banco de dados deve ser considerado um processo de refinamento iterativo, até que o projeto mais adequado seja alcançado.

- Quais são as diretrizes e escolhas de projeto para os conceitos EER de especialização/generalização e categorias (tipos de união)?

Escolhas de projeto para especialização/generalização

Orientações que ajudam no processo de projeto para conceitos EER:

1. Representar apenas as subclasses necessárias para evitar uma aglomeração extrema do modelo conceitual;
2. Se uma subclasse possui poucos atributos específicos, ela pode ser mesclada á super-classe;
   - Os atributos específicos manteriam valores NULL para entidades não membros da subclasse;
   - Um atributo de *tipo* poderia especificar se uma entidade é um membro da subclasse.

Escolhas de projeto para especialização/generalização

Orientações que ajudam no processo de projeto para conceitos EER:

3. Os tipos de união e categorias geralmente devem ser evitados, a menos que a situação definitivamente justifique esse tipo de construção;
4. A escolha de restrições disjuntas/sobrepostas e totais/parciais sobre a especialização/generalização é controlada pelas regras no minimundo que está sendo modelado.
    - Caso não seja indicada nenhuma restrição, o padrão é sobreposição e parcial.

Seção 09

Modelo Relacional

MR: Modelo Relacional

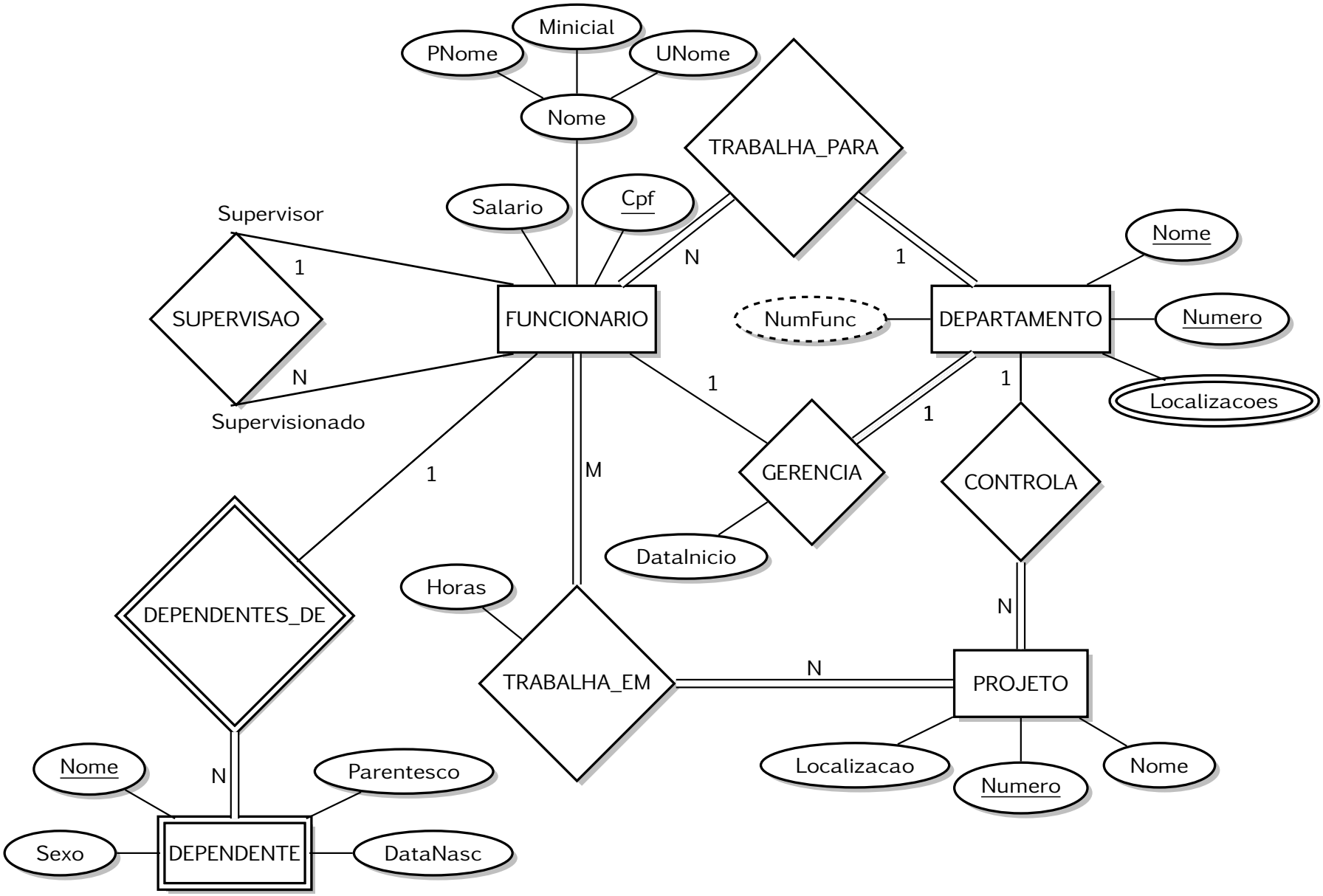
- Consiste no modelo de implementação baseado no paradigma relacional.

Mapeamento MER para MR

- Apresentação de procedimentos para criação de **esquema relacional** com base em um esquema *Entidade Relacionamento*.

- Esboçaremos um algoritmo de 7 etapas para "converter"as construções básicas do modelo ER em relações.

Mapeamento MER para MR



Mapeamento MER para MR

Etapa 1 - Mapeamento de tipos de entidade regular

- Para cada tipo de entidade regular (forte), crie uma relação que inclua todos os atributos simples.
- Inclua apenas os atributos de componente simples de um atributos composto.
- Escolha um dos atributos-chave da entidade regular como chave primária da nova relação.
- Se a chave escolhida for composta, o conjunto de atributo simples que a compõe formarão a chave primária.

Etapa 1 - Mapeamento de tipos de entidade regular

FUNCIONARIO

Cpf	Pnome	Minicial	Unome	Salario	Dnr	Cpf_supervisor

DEPARTAMENTO

Dnumero	Dnome

PROJETO

Projnumero	Projlocal	Projnome	Dnum

Etapa 2 - Mapeamento de tipos de entidade fraca

- Para cada tipo de entidade fraca, crie uma relação e inclua todos os atributos simples.
- Inclua como atributos de chave estrangeira da relação, os atributos de chave primária da relação que corresponde aos tipos de entidade proprietária.

Etapa 2 - Mapeamento de tipos de entidade fraca

FUNCIONARIO

Cpf	Pnome	Minicial	Unome	Salario	Dnr	Cpf_supervisor

DEPARTAMENTO

Dnumero	Dnome

PROJETO

Projnumero	Projlocal	Projnome	Dnum

DEPENDENTE

Fcpf	Nome_dependente	Sexo	Datanasc	Parentesco

Etapa 3 - Mapeamento dos tipos de relacionamento binário 1:1

- Para cada tipo de relacionamento, identifique as relações que correspondem aos tipos de entidades participantes.

- Existem três técnicas possíveis para solução:
  1. Técnica de chave estrangeira;
  2. Técnica de relacionamento mesclado;
  3. Técnica de relação de referência cruzada ou relacionamento.

Etapa 3 - Mapeamento dos tipos de relacionamento binário 1:1

1. Técnica de chave estrangeira:
   - Escolha uma das relações (R1) e inclua como chave estrangeira a chave primária da segunda relação (R2).
   - Inclua todos os atributos simples do tipo de relacionamento 1:1 como atributos de R1.

Etapa 3 - Mapeamento dos tipos de relacionamento binário 1:1

2 Técnica de relação mesclada:

- Mescle os dois tipos de entidade e o relacionamento em uma única relação.
- Possível somente quando ambas as relações são totais - *indica que as duas tabelas terão exatamente a mesma quantidade de tuplas.*

Etapa 3 - Mapeamento dos tipos de relacionamento binário 1:1

3 Técnica de relação de referência cruzada ou relacionamento:

- Configurar uma terceira relação R para a finalidade de referência cruzada das chaves primárias das duas relações;
- A relação R incluirá os atributos de chave primária das duas relações (origem) como chave estrangeira;
- A relação R é chamada de **relação de relacionamento** ou **tabela de pesquisa**;
- Técnica exigida para relação M:N.

Mapeamento MER para MR

Etapa 4 - Mapeamento dos tipos de relacionamento binário 1:N

- Identificar a relação S que representa o tipo de entidade participante no lado N do relacionamento;
- Inclua como chave estrangeira em S a chave primária da relação que representa o outro tipo de entidade participante.
- Uma técnica alternativa é usar a opção de **relação de relacionamento**.

Mapeamento MER para MR

Etapa 5 - Mapeamento de tipos de relacionamento binário M:N

- Para cada tipo de relacionamento M:N, crie uma nova relação S;
- Inclua como atributos de chave estrangeira em S as chaves primárias das relações que representam os tipos de entidades participantes - **sua *combinação* formará a chave primária de S**;
- Não podemos representar um tipo de relacionamento M:N por umúnico atributo de chave estrangeira em uma das relações participantes.

Etapa 5 - Mapeamento de tipos de relacionamento binário M:N

FUNCIONARIO

Cpf	Pnome	Minicial	Unome	Salario	Dnr	Cpf_supervisor

DEPARTAMENTO

Dnumero	Dnome

PROJETO

Projnumero	Projlocal	Projnome	Dnum

DEPENDENTE

Fcpf	Nome_dependente	Sexo	Datanasc	Parentesco

TRABALHA_EM

Fcpf	Pnr	Horas

Etapa 6 - Mapeamento de atributos multivalorados

- Para cada atributo multivalorado A, crie uma relação R.
- A relação R incluirá um atributo correspondente a A, mais o atributo da chave primária da relação que representa o tipo de entidade (ou relacionamento) que tem A como atributo multivalorado.

Etapa 6 - Mapeamento de atributos multivalorados

FUNCIONARIO

Cpf	Pnome	Minicial	Unome	Salario	Dnr	Cpf_supervisor

DEPARTAMENTO

Dnumero	Dnome

PROJETO

Projnumero	Projlocal	Projnome	Dnum

DEPENDENTE

Fcpf	Nome_dependente	Sexo	Datanasc	Parentesco

TRABALHA_EM

Fcpf	Pnr	Horas

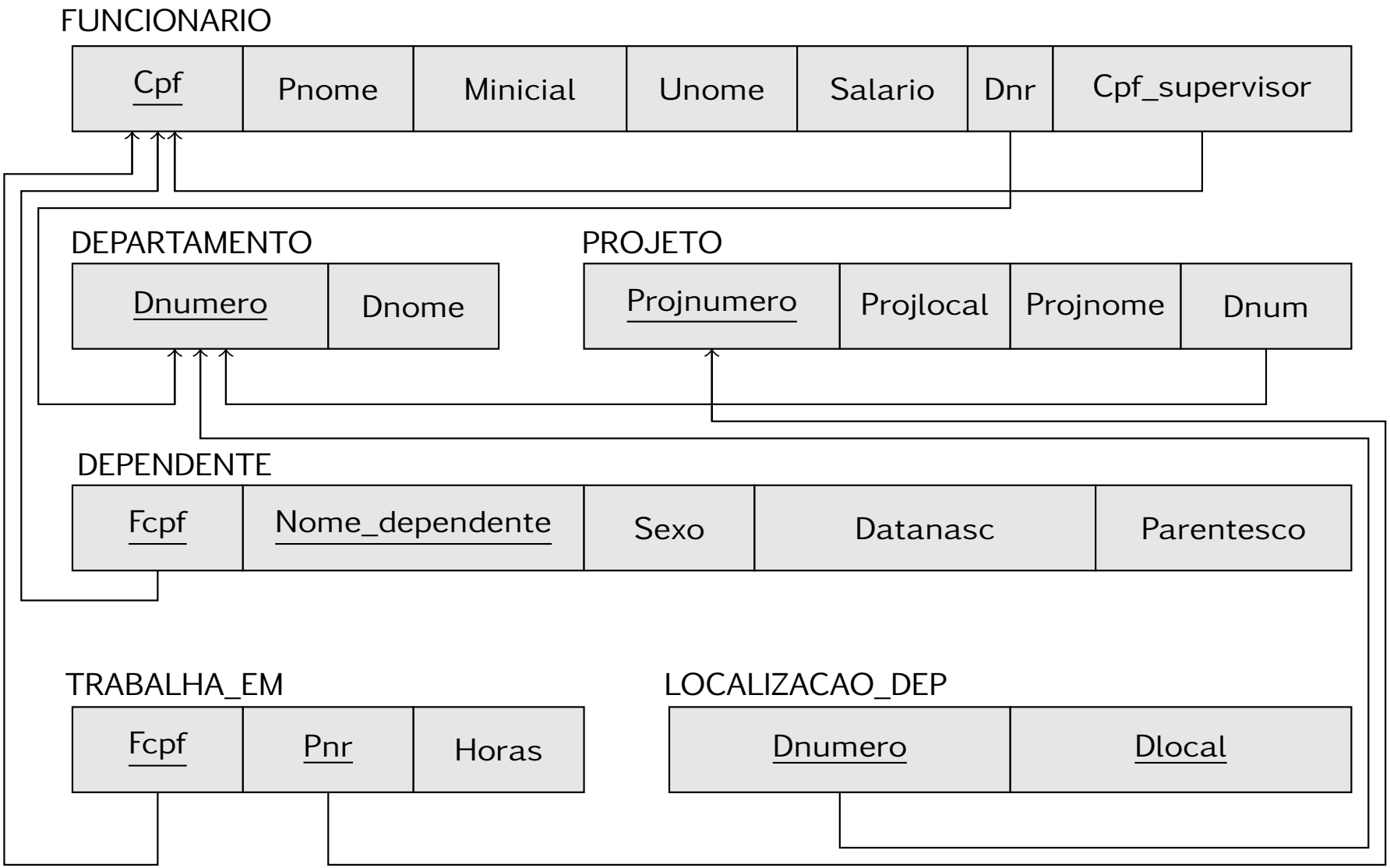
LOCALIZACAO_DEP

Dnumero	Dlocal

Etapa 7 - Mapeamento de tipos de relacionamento n-ário

- Para cada tipo de relacionamento n-ário, em que $n > 2$, crie uma relação S para representar o relacionamento.
- Inclua como atributos de chave estrangeira as chaves primárias das relações que representam os tipos de entidade participantes.
- A chave primária é a combinação de todas as chaves estrangeiras que referenciam as relações das entidades participantes.

Resultado do mapeamento



Seção 17

Armazenamento Físico

Estruturas de Arquivo

- BDs são armazenados *fisicamente* em um **meio de armazenamento** no computador
- Geralmente como **arquivos de registros** em *discos magnéticos*
- SGBD *recupera*, *atualiza* e *processa* dados de acordo com a necessidade
- Os meios (mídias) de armazenamento de um computador formam uma **hierarquia de armazenamento**, na qual os dados residem e são transportados
  - Memória primária → operada pela CPU
  - Memória secundária → não operada pela CPU
  - Memória terciária → não operada pela CPU

Estruturas de Arquivo: Memória Primária

- Operação → direto pela CPU
- Velocidade → alta
- Capacidade → baixa (KB, MB, GB)
- Volatilidade → perda de dados na falta de energia
- Custo → elevado
- Tipos → *cache* e principal (DRAM)
  - *Cache* (RAM estática) → extremamente rápida e cara, é usada pela CPU pra agilizar execução de instruções de programa com pré-busca e pipelining
  - Principal (RAM dinâmica) → de menor velocidade e mais barata que a *cahce,* é usada pela CPU pra manter instruções de programa e dados e como área de trabalho

Estruturas de Arquivo: Memória Secunária

- Operação → dados devem ser copiados para memória primária para serem operados pela CPU
- Velocidade → baixa
- Capacidade → alta (TB)
- Volatilidade → não volátil
- Custo → baixo
- Tipos → discos rígidos magnéticos (HD)

Estruturas de Arquivo: Memória Terciária

- Operação → dados devem ser copiados para memória primária para serem operados pela CPU
- Velocidade → muito baixa
- Capacidade → muito alta (TB, PB)
- Volatilidade → não volátil
- Custo → muito baixo
- Tipos → fitas magnéticas e dispositivos ópticos
  - Fitas magnéticas → leitura sequencial, sofrendo interferência eletromagnética. Muito utilizada para *backup*
  - Dispositivos ópticos → interferência eletromagnética desprezível. Muito utilizada para dados multimidia

Estruturas de Arquivo: Memória Flash

- Forma de armazenamento intermediária entre a memória principal (DRAM) e a secundária (HD)
- Operação → podem ser operadas diretamente pela CPU
- Velocidade → alta
- Capacidade → mediana (GB)
- Volatilidade → não volátil
- Custo → médio
- Tipos → EEPROM
  - Estão presentes em câmeras, MP3 players, celulares, PDAs e unidades USB e se diferenciam das DRAMs por ser necessário apagar e gravar um bloco inteiro em caso de alteração de dados

Estruturas de Arquivo: Operação em Memória

- Programas residem e são executados em memória principal
- Geralmente grandes BDs residem em armazenamento secundário
- Partes do BDs são lidos e escritos em *buffers* na memória principal conforme a necessidade

Hierarquia de memória

Banco de dados de memória principal

- São BDs que podem ser mantidos inteiros na memória principal (com uma cópia de backup no disco magnético).
- Úteis em aplicações de tempo real que exigem tempo de resposta rápido.
- *Por exemplo, em aplicações de comutação de telefone, que armazenam BDs que contêm informações de roteamento e linha.*

Memória flash

- Outra forma armazenamento entre a DRAM e o armazenamento em disco magnético;
- São de alta densidade, alto desempenho, não voláteis e usam a tecnologia EEPROM;
- A vantagem da memória flash é a velocidade de acesso rápida;
- A desvantagem é que um bloco inteiro precisa ser apagado e gravado simultaneamente;
- Estão presentes em câmeras, MP3 players, celulares, PDAs, unidades flash USB.

Discos Ópticos

- Discos de CD-ROM armazenam dados opticamente e são lidos por um laser;
- Os CD-ROMs contêm dados pré-gravados que não podem ser modificados;
- O DVD permite 4,5 a 15 GB de armazenamento por disco.

Discos Ópticos

- Os discos WORM são usados para arquivar dados;
- Permitem que os dados sejam gravados uma vez e lidos qualquer número de vezes;
- Possuem meio gigabyte de dados por disco e duram mais que os discos magnéticos.

Memórias de jukebox óptico

- Utilizam um conjunto de placas de CD-ROM, carregadas em unidades por demanda.
- Seu tempo de recuperação é mais lento do que dos discos magnéticos.
- A utilização desse armazenamento diminui devido a diminuição no custo e ao aumento nas capacidades dos discos magnéticos.

Hierarquia de memória

Fitas magnéticas

- Usadas para arquivamento e armazenamento de backup dos dados.

Jukeboxes de fita

- Possuem um banco de fitas que são catalogados e podem ser carregadas automaticamente nas unidades de fita.
- Populares como **armazenamento terciário**, para manter *terabytes* de dados.

Armazenamento de BDs

Dados persistentes

- BDs armazenam grande quantidade de dados que persistem por longos períodos de tempo;
- Dados persistentes são acessados e processados repetidamente durante esse período.

Dados transientes

- Dados transientes persistem apenas por um tempo limitado durante a execução do programa.

Armazenamento de BDs

A maioria dos BDs é armazenada de maneira permanente (ou persistente) no armazenamento secundário do disco magnético.

- BDs são muito grandes para caber inteiramente na memória principal.
- O custo de armazenamento é **menor** para o armazenamento secundário do que para o primário.
- No futuro, os bancos de dados poderão residir em diferentes níveis de hierarquia de memória.
- Contudo, os **discos magnéticos** continuarão a ser a escolha principal para BDs grandes.

Armazenamento de BDs

- **Fitas magnéticas** são usadas para o backup de banco de dados.
- O armazenamento em fita custa menos que o armazenamento em disco.
- O acesso aos dados na fita é muito lento.
- Os dados armazenados nas fitas são **off-line**, alguma intervenção para carregar uma fita é necessária antes que os dados se tornem disponíveis.
- Enquanto os discos são dispositivos **on-line**, podem ser acessados diretamente a qualquer momento.

Armazenamento de BDs

- Técnicas para armazenar grande quantidade de dados em disco são importantes para projetistas, DBA e implementadores de um SGBD.
- Em geral, o SGBD tem várias opções disponíveis para organizar os dados.
- O processo de **projeto físico de banco de dados** envolve a escolha das técnicas de organização de dados que mais se ajustam aos requisitos.

Armazenamento de BDs

- Aplicações típicas precisam apenas de uma pequena parte do BD de cada vez para processamento.
  - Essa parte é localizada no disco;
  - Copiada para memória principal para processamento;
  - E, depois, reescrita para o disco se os dados forem alterados.
- Dados armazenados no disco são organizados como **arquivos de registros**.
- Cada registro é uma **coleção de valores de dados**, interpretados como fatos sobre *entidades*, *atributos* e *relacionamentos*.
- Registros devem ser armazenados em disco de uma maneira que torne possível localizá-los de modo eficiente.

Organizações de arquivo primário

- Determinam como os registros de arquivo são *colocados fisicamente* no disco e como os *registros podem ser acessados*.
  - **Arquivo de heap** (ou **arquivo desordenado**) coloca os registros no disco sem qualquer ordem particular, acrescentando novos registros ao final do arquivo.
  - **Arquivo classificado** (ou **arquivo sequencial**) mantém os registros ordenados pelo valor de um campo em particular (*campo de classificação*).
  - **Arquivo em hashing** usa uma função de hash aplicada a um campo em particular (**chave hash**) para determinar o posicionamento de um registro no disco.

Organização secundária ou estrutura de acesso auxiliar

- Permite acesso aos registros do arquivo com base em campos alternativos (além dos que foram usados para a organização de arquivo primário).

Descrição de hardware dos dispositivos de disco

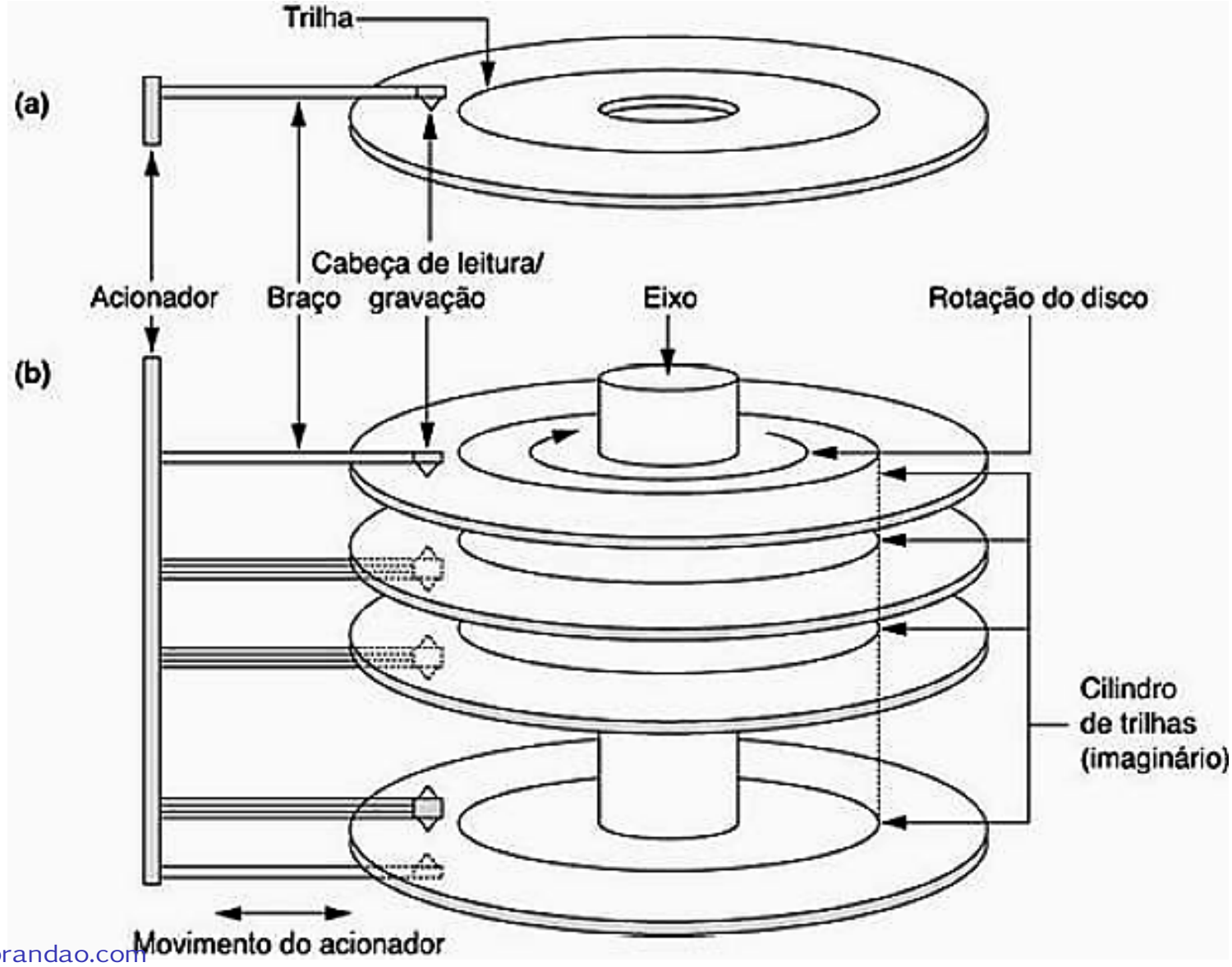
- **Discos magnéticos** são usados para armazenar grande quantidade de dados.
- A unidade de dados básica no disco é um **bit** de informação.
- Uma área do disco pode representar o valor de bit 0 (zero) ou bit 1 (um).
- Bits são agrupados em **bytes** (ou **caracteres**) para codificar a informação.
- Os tamanhos de byte variam de 4 a 8 bits.

Armazenamento secundário

Descrição de hardware dos dispositivos de disco

- A **capacidade** de um disco é o número de bytes que ele pode armazenar.
- Discos são feitos de um material magnético modelado como um disco circular fino e protegidos por uma camada de plástico ou acrílico.
- **Disco de face simples**: Armazena informações apenas em uma de suas superfícies.
- **Disco de face dupla**: As duas superfícies são usadas para armazenamento.

Descrição de hardware dos dispositivos de disco



Descrição de hardware dos dispositivos de disco

- Para aumentar a capacidade de armazenamento, discos são montados em um **disk pack**.
- Um disk pack pode incluir muitos discos (muitas superfícies).
- Informações são armazenadas na superfície do disco em círculos de *pequena largura* (diferentes diâmetros).
- Cada círculo é chamado de **trilha**.
- O número de trilhas em um disco varia de centenas até milhares.
- A capacidade de cada trilha varia de dezenas de KB até 150 KB.

Descrição de hardware dos dispositivos de disco

- Há uma melhoria na capacidade de armazenamento, nas taxas de transferência e no custo associadas aos discos.
- Trilhas com o mesmo diâmetro nas superfícies de disk packs formam um **cilindro**.
- Dados armazenados em um cilindro podem ser recuperados mais rapidamente do que distribuídos entre diferentes cilindros.

Descrição de hardware dos dispositivos de disco

- Uma trilha é dividia em **blocos** ou **setores** menores.
- A divisão é fixada na superfície do disco e não pode ser alterada.
- Várias organizações de setor são possíveis.
- No entanto, nem todos os discos têm suas trilhas dividas em setores.

Descrição de hardware dos dispositivos de disco

- A divisão de uma trilha em **blocos de disco** (**páginas**) de mesmo tamanho é definida pelo sistema operacional durante a **formatação** (**inicialização**) do disco.
- O tamanho do bloco é fixado na inicialização e não é trocado dinamicamente.
- Seu tamanho varia de 512 a 8.192 bytes.

Descrição de hardware dos dispositivos de disco

- Disco com setores fixos geralmente possuem setores subdivididos em blocos na inicialização.
- Os blocos são separados por **lacunas** de tamanho fixo.
- Lacunas entre blocos incluem **informações de controle** codificadas e gravadas durante a inicialização.
- Tais informações são utilizadas para determinar qual bloco da trilha segue cada lacuna entre blocos.

Descrição de hardware dos dispositivos de disco

- Um disco é um dispositivo endereçável por **acesso aleatório**.
- A transferência de dados entre a memória principal e o disco ocorre em *unidades de blocos de disco*.
- Um bloco de disco pode ser acessado *aleatoriamente* se especificamos seu endereço.
- O **endereço de hardware** de um bloco é fornecido ao hardware de E/S do disco.
- Em unidades de disco modernas, um número chamado LBA entre *0* e *n* é mapeado para o bloco correto pelo controlador de unidade de disco.

Descrição de hardware dos dispositivos de disco

- **Buffer** é uma área reservada contígua no armazenamento principal, que mantém um bloco de disco.
- O *endereço de um buffer* também é fornecido ao hardware de E/S do disco.
- Vários blocos contíguos podem ser transferidos como uma unidade, denominada **cluster**.

Comando de leitura

- O bloco de disco é copiado para o buffer.

Comando de gravação

- O conteúdo do buffer é copiado para o bloco de disco.

Descrição de hardware dos dispositivos de disco

- **Cabeça de leitura/gravação do disco**
  - Mecanismo de hardware que lê ou grava um bloco.
  - Faz parte de um sistema chamado **unidade de disco** (que inclui um motor que gira os discos).
  - Uma cabeça de leitura/gravação é conectada a um **braço mecânico**.

Descrição de hardware dos dispositivos de disco

- *Disk packs* com superfícies múltiplas são controlados por várias cabeças de leitura/gravação.
- Braços são conectados a um acionador conectado a outro motor elétrico;
- Responsável por mover as cabeças de leitura/gravação e as posicionar sobre o cilindro de trilhas especificado em um endereço de bloco.

Descrição de hardware dos dispositivos de disco

- Unidades de disco rígido giram o disk pack continuamente a uma velocidade constante.
  - Com a cabeça de leitura/gravação posicionada na trilha correta e o bloco especificado no endereço de bloco;
  - move-se sob a cabeça de leitura/gravação;
  - e o componente eletrônico da cabeça de leitura/gravação é ativado para transferir os dados.

Dispositivos de armazenamento secundário

Descrição de hardware dos dispositivos de disco

- **Discos de cabeça fixa**
  - Possuem cabeças de leitura/gravação fixas.
  - O número de cabeças é correspondente ao de trilhas.
  - Não ocorre movimento mecânico.
  - Uma trilha ou cilindro é selecionado eletronicamente, passando para a cabeça de leitura/gravação apropriada.
  - São mais ágeis, no entanto não são muito utilizados devido seu alto custo.
- **Discos de cabeça móvel**
  - Unidades de disco com um acionador.

Descrição de hardware dos dispositivos de disco

- **Controlador de disco**
  - Embutido na unidade de disco, é o responsável pelo seu controle.
  - Interliga a unidade de disco ao sistema de computação.
  - Aceita comandos de E/S de alto nível.
  - Posiciona o braço e prepara a ação de leitura/gravação.

Dispositivos de armazenamento secundário

Descrição de hardware dos dispositivos de disco

- **Tempo de busca**
  - A partir do endereço de um bloco de disco, o controlador de disco posiciona mecanicamente a cabeça de leitura/gravação na trilha correta.
- **Tempo exigido para a ação de transferência de um bloco de disco**
  - **Tempos de busca típicos em desktops:** 5 a 10ms
  - **Tempos de busca típicos em servidores:** 3 a 8ms

Descrição de hardware dos dispositivos de disco

- **Atraso rotacional ou latência**
  - Tempo que o bloco desejado gira até a posição sob a cabeça de leitura/gravação.
  - Relativo as *rpm* do disco.
- **Tempo de transferência de bloco**
  - Tempo adicional para transferir os dados.

Descrição de hardware dos dispositivos de disco

- **Tempo total para localizar e transferir um bloco, a partir de seu endereço**
  - É a soma do tempo de busca, do atraso rotacional e do tempo de transferência de bloco.
  - O tempo de busca e o atraso rotacional são muito maiores que o tempo de transferência do bloco.

Descrição de hardware dos dispositivos de disco

- Para tornar a transferência de múltiplos blocos mais eficiente, transfere-se vários blocos consecutivos na mesma trilha ou cilindro.
- Eliminando o tempo de busca e o atraso rotacional para todos, menos para o primeiro bloco.
- **Taxa de transferência em massa**
  - Calcula-se o tempo exigido para transferir blocos consecutivos.

Descrição de hardware dos dispositivos de disco

- O tempo necessário para localizar e transferir um bloco de disco está na ordem de milissegundos.
- É considerado alto em comparação com o tempo da memória principal.
- A localização dos dados no disco é um *gargalo principal* nas aplicações de banco de dados.
- Colocar "informações relacionadas" em blocos contíguos é o objetivo de qualquer organização de armazenamento no disco.

Dispositivos de armazenamento de fita magnética

- **Fitas magnéticas são dispositivos de acesso sequencial**
  - Para acesso do enésimo bloco na fita é necessário varrer os *n-1* blocos anteriores.
  - Bytes são armazenados consecutivamente em fitas.
  - Dados são armazenados em **bobinas de fita** magnética de alta capacidade.
  - Uma unidade de fita precisa ler os dados ou gravá-los em uma bobina de fita.

Dispositivos de armazenamento de fita magnética

- Uma cabeça de leitura/gravação é usada para ler ou gravar dados na fita.
- Os registros de dados na fita são armazenados em **blocos**.
- Tais blocos são maiores do que os dos discos, no entanto, as lacunas entre blocos são muito grandes.
- Assim, agrupa-se muitos registros em um bloco para otimização do espaço.

Dispositivos de armazenamento de fita magnética

- Em uma fita acessamos os blocos de dados em **ordem sequencial**.
- A fita é montada e depois varrida até que o bloco solicitado passe sob a cabeça de leitura/gravação.
- O acesso da fita pode ser lento, por esse motivo não são usadas para armazenar dados on-line.

Dispositivos de armazenamento de fita magnética

- Fitas possuem uma função de **backup** do banco de dados.
- Com o intuito de manter cópias de arquivos de disco no caso de falhas.
- Arquivos de disco são copiados periodicamente para a fita.
- Fitas podem ser utilizadas para armazenar arquivos de banco de dados excessivamente grandes, raramente usados e desatualizados (**arquivados** em fita).

Dispositivos de armazenamento de fita magnética

- Para aplicações críticas on-line, são usados **sistemas espelhados** para manter três conjuntos de discos idênticos.
  - Dois em operação on-line e um como backup.
  - Discos off-line tornam-se um dispositivo de backup.
  - Os três são usados de modo que possam ser trocados caso haja uma falha.

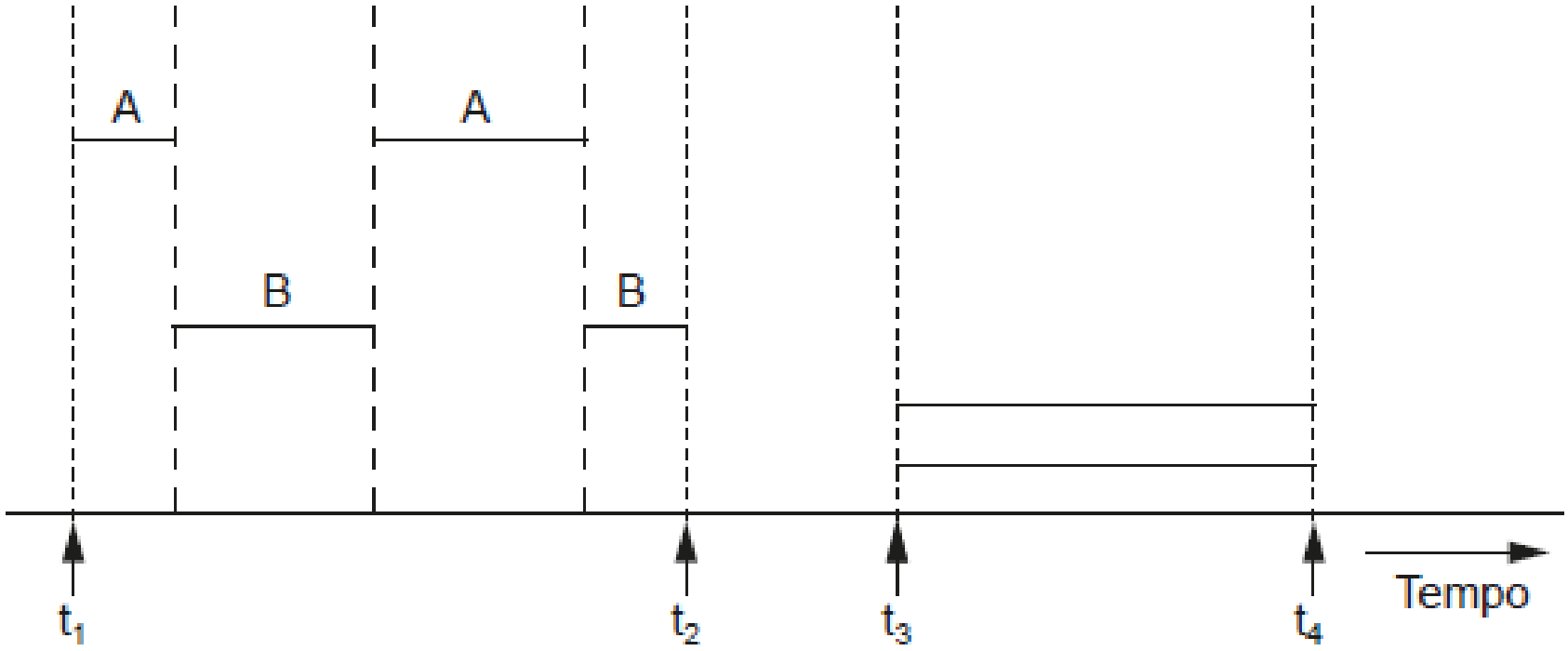
Dispositivos de armazenamento de fita magnética

- O backup de bancos de dados corporativos é uma tarefa importante. Bibliotecas de fitas com *slots* centenas de cartuchos são usados como Digital e Superdigital Linear Tapes (DLTs e SDLTs);
- Com capacidades em gigabytes;
- Registram dados em trilhas lineares.

Buffering de blocos

- Buffers podem ser reservados na memória principal para agilizar a transferência de blocos do disco para a memória principal.
- Existe um **processador (controlador) de E/S** de disco que transfere um bloco de dados entre a memória e o disco independente e em paralelo com o processamento da CPU.
- Assim, se um buffer está sendo lido ou gravado, a CPU pode processar dados em outro buffer.

Buffering de blocos



Os processos A e B estão rodando **simultaneamente** em um padrão **intervalado**.

Os processos C e D estão rodando **simultaneamente** em um padrão **paralelo**.

Buffering de blocos

- Se a CPU controla vários processos, a execução paralela não é possível.
- Contudo, processos podem ser executados simultaneamente de forma intervalada.
- O buffering é mais útil quando os processos podem ser executados simultaneamente em um padrão paralelo.
- Seja por um processador de E/S de disco separado ou por vários processadores (CPUs).

Buffering de blocos

Buffering duplo

- Uso de dois buffers para leitura do disco.
- Leitura e processamento prosseguem em paralelo.
- A CPU pode processar um bloco quando sua transferência para a memória principal termina;
- Ao mesmo tempo, o processador de E/S de disco pode estar lendo e transferindo o próximo bloco para um buffer diferente.

Buffering de blocos

Buffering duplo

- Permite a leitura ou gravação contínua de dados em blocos de disco consecutivos.
- Elimina o tempo de busca e o atraso rotacional para todas as transferências de bloco, com exceção da primeira.
- Os dados ficam prontos para processamento, reduzindo o tempo de espera nos programas.

Gravando registros de arquivo no disco

Registros

- Dados são armazenados na forma de registros.
- Cada registro contém uma coleção de valores ou itens de dados relacionados, que descrevem entidades e seus atributos.
- Cada valor é formado por bytes e corresponde a um campo em particular do registro.

Tipos de registro ou Formato de registro

- Coleção de nomes de campo e seus tipos de dados correspondentes.

Gravando registros de arquivo no disco

Tipo de dado

- Especifica os tipos de valores que um campo pode assumir, incluem:
  - **Tipos de dados numéricos**: inteiro, inteiro longo ou ponto flutuante.
  - **Cadeia de caracteres**: tamanho fixo ou variável.
  - **Booleanos**: valores 0 e 1, ou TRUE e FALSE.
  - **Data** e **tempo**, especialmente codificados.

Gravando registros de arquivo no disco

Tipo de dado

- O número de bytes para cada tipo de dado é fixo para determinado sistemas de computação.
  - **Inteiro e Número real**: 4 bytes
  - **Inteiro longo**: 8 bytes
  - **Booleano**: 1 byte
  - **Data**: 10 bytes (formato DD-MM-AAAA)
  - **String de tamanho fixo de *k* caracteres**: *k* bytes

Gravando registros de arquivo no disco

Objetos binários grandes (BLOBs)

- Grandes objetos não estruturados.
- Imagens, vídeo digitalizado ou streams de áudio, ou então texto livre.
- Um item de dados BLOB é armazenado separadamente de seu registro, em um conjunto de blocos de disco.
- Um ponteiro para o BLOB é incluído no registro.

Gravando registros de arquivo no disco

Arquivos

- Um **arquivo** é uma *sequência* de registros.
- Geralmente, todos os registros em um arquivo são do mesmo tipo.

Arquivo composto de registros de tamanho fixo

- Cada registro no arquivo tem o mesmo tamanho (em bytes).

Gravando registros de arquivo no disco

Arquivo composto de registros de tamanho variável

- Diferentes registros no arquivo possuem diversos tamanhos.
  - Os registros do arquivo são do mesmo tipo, mas possuem **campos de tamanho variável**.
  - *Por exemplo*: O campo **Nome** de **FUNCIONARIO** possui tamanho variável.

Gravando registros de arquivo no disco

Arquivo composto de registros de tamanho variável

- Diferentes registros no arquivo possuem diversos tamanhos.
  - Os registros do arquivo são do mesmo tipo, mas alguns campos possuem múltiplos valores para registros individuais (**campo repetitivo**).
  - Um grupo de valores para o campo é chamado de **grupo repetitivo**.

Gravando registros de arquivo no disco

Arquivo composto de registros de tamanho variável

- Diferentes registros no arquivo possuem diversos tamanhos.
  - Os registros do arquivo são do mesmo tipo, mas possuem **campos opcionais** (podem ter valores para alguns, mas não para todos os registros).

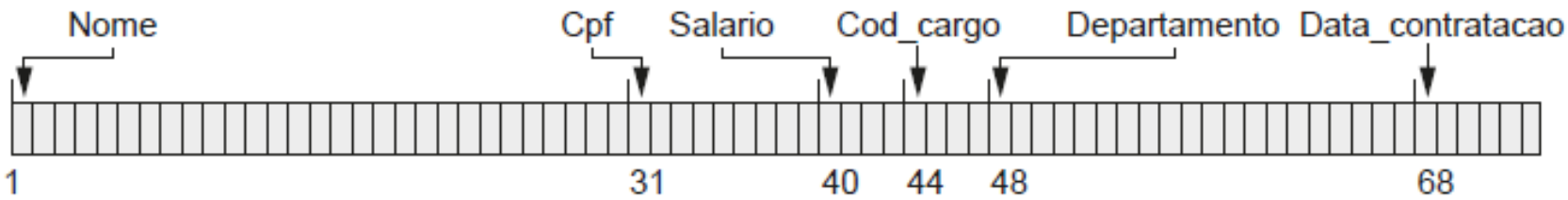
Gravando registros de arquivo no disco

Arquivo composto de registros de tamanho variável

- Diferentes registros no arquivo possuem diversos tamanhos.
  - O arquivo contém registros de *tipos de registro diferentes*, portanto de tamanho variável (arquivo misto).
  - Registros relacionados de diferentes tipos *agrupados* em blocos de disco.
  - *Exemplo*: Registros de **HISTORICO_ESCOLAR** de determinado aluno podem ser colocados após o registro desse **ALUNO**.

Gravando registros de arquivo no disco

Formatação de registros de um arquivo de tamanho fixo

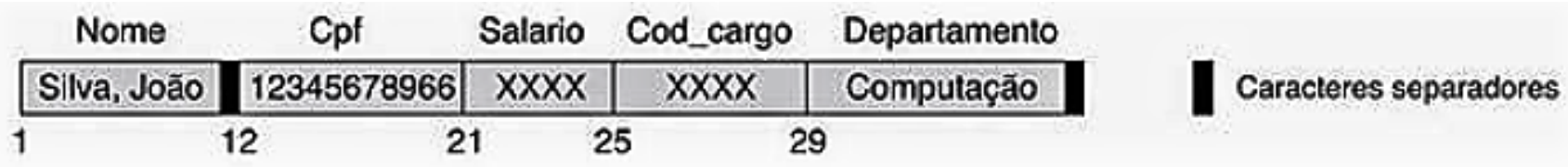


Registro de tamanho fixo com seis campos e tamanho de 71 bytes

- É possível representar um arquivo que deveria ter registros de tamanho variável como um arquivo de registros de tamanho fixo.
- Poderíamos incluir em cada registro de arquivo os campos opcionais, mas armazenar o valor NULL se necessário.
- Para um campo repetitivo, poderíamos alocar espaços em cada registro do número máximo possível de ocorrências do campo.

Gravando registros de arquivo no disco

Formatação de registros de um arquivo de tamanho variável



Registro com dois campos de tamanho variável e três campos de tamanho fixo

- Usa-se caracteres **separadores** especiais (? ou % ou $), para determinar os bytes que representa cada campo;
- Para terminar os campos de tamanho variável;
- Ou para armazenar o tamanho do campo em bytes no próprio registro (antes do valor do campo).

Gravando registros de arquivo no disco

Formatação de registros de um arquivo com campos opcionais

- Se o número de campos for pequeno, podemos incluir em cada registro **<nome-campo, valor-campo>**.
- Usa-se o mesmo caractere separador para o nome do campo do valor e para um campo do campo seguinte.
- Atribui-se um **tipo de campo** a cada campo, **<tipo-campo, valor-campo>**.
- Um campo repetitivo precisa de um caractere separador para afastar valores repetidos e para indicar o término.

Gravando registros de arquivo no disco

Formatação de um arquivo com registros de diferentes tipos

- Cada registro é precedido por um indicador de tipo de registro.

Gravando registros de arquivo no disco

Blocagem de registros espalhados *versus* não espalhados

- Bloco é a *unidade de transferência de dados* entre o disco e a memória.
- Registros de um arquivo precisam ser alocados a blocos de disco.

Gravando registros de arquivo no disco

Blocagem de registros espalhados *versus* não espalhados

- *Exemplo:*
  - Bloco com tamanho B bytes
  - Arquivo de registros de tamanho fixo, R bytes
  - Sendo $B > R$, estabelecemos **bfr** = [B/R] registros por bloco
  - Onde [(x)] (**função floor**) **arredonda para baixo** o número x para um inteiro
  - O valor **bfr** é chamado de **fator de blocagem** para o arquivo
  - Em geral, R pode não dividir B exatamente, de modo que temos algum espaço não usado igual a **B - (bfr * R) bytes**

Gravando registros de arquivo no disco

Blocagem de registros espalhados *versus* não espalhados

- **Organização espalhada**
  - Armazenar parte de um registro em um bloco e o restante em outro.
  - Um **ponteiro** no final do primeiro bloco aponta para o bloco que contém o restante do registro.

Blocagem de registros espalhados *versus* não espalhados

- **Organização não espalhada**
  - Registros que não podem atravessar os limites de bloco.
  - Usado para registros de tamanho fixo tendo B > R.
  - Para registros de tamanho variável, pode-se usar uma organização espalhada ou não espalhada.
  - Se o registro médio for grande, é vantajoso usar o espalhamento para reduzir o espaço perdido em cada bloco.

Gravando registros de arquivo no disco

Blocagem de registros espalhados versus não espalhados

A - Organização de registro não espalhada.

B - Organização de registro espalhada.

Gravando registros de arquivo no disco

Blocagem de registros espalhados *versus* não espalhados

- Para registros de tamanho variável com organização espalhada, cada bloco pode armazenar um número diferente de registros.
- O fator de bloco *bfr* representa o número *médio* de registros por bloco para o arquivo.
- Podemos usar *bfr* para calcular o número de blocos *b* necessários para um arquivo de *r* registros:
  - $b = [\ (r/bfr)]$ blocos
  - $[(x)^*]$ (**função ceiling**) arredonda para cima o valor de x até o próximo inteiro.

Alocando blocos de arquivo no disco

Técnicas-padrão para alocar os blocos de um arquivo no disco.

- **Alocação contígua**
  - Blocos de arquivo são alocados a blocos de disco consecutivos.
  - Leitura do arquivo torna-se rápida usando o buffering duplo.
  - A expansão do arquivo é dificultada.

Gravando registros de arquivo no disco

Alocando blocos de arquivo no disco

Técnicas-padrão para alocar os blocos de um arquivo no disco.

- **Alocação ligada**
  - Cada bloco de arquivo contém um ponteiro para o próximo bloco.
  - Facilita a expansão do arquivo.
  - Torna a leitura do arquivo mais lenta.

Gravando registros de arquivo no disco

Alocando blocos de arquivo no disco

Técnicas-padrão para alocar os blocos de um arquivo no disco.

- **Clusters (segmentos ou extensões de arquivo)**
  - Uma combinação das duas técnicas aloca clusters de blocos de disco consecutivos, e os clusters são ligados.
- **Alocação indexada**
  - Um ou mais blocos de índice contêm ponteiros para blocos de arquivos reais.
  - Também é comum usar combinações dessas técnicas.

Cabeçalhos de arquivo

Cabeçalho de arquivo ou descritor de arquivo

- Contém informações sobre um arquivo (exigidas pelos programas que acessam os registros do arquivo).
- O cabeçalho inclui informações para:
  - Determinar endereços de disco dos blocos de arquivo;
  - Registrar descrições de formato (tamanhos de campo e a ordem dos campos em um registro);
  - Para registros não espalhados de tamanho fixo;
  - E para ódigos de tipo de campo, caracteres separadores e códigos de tipo de registro (para registros de tamanho variável).

Gravando registros de arquivo no disco

Pesquisa linear

- Realizada pelos programas de pesquisa, nos blocos de arquivo, quando o endereço do bloco que contém o registro desejado não for conhecido.
- Cada bloco de arquivo é copiado para um buffer e pesquisado até que o registro seja localizado utilizando informação do cabeçalho (e todos os blocos pesquisados).
- Tal processo pode ser demorado para um arquivo grande.
- Uma boa organização de arquivo localiza um bloco desejado com um número mínimo de transferências de bloco.

Organização de arquivo

- Organização dos dados de um arquivo em registros, blocos e estruturas de acesso.
- Inclui o modo como registros e blocos são colocados no meio de armazenamento e interligados.
- Uma organização de arquivo bem-sucedida realiza de maneira eficiente as operações que esperamos *aplicar frequentemente* ao arquivo.

Operações em arquivos

Método de acesso

- Oferece um grupo de operações que podem ser aplicadas a um arquivo.
- É possível aplicar vários métodos de acesso a uma organização de arquivo.
- Alguns métodos de acesso só podem ser aplicados a arquivos organizados de certas maneiras.

Operações em arquivos

Agrupadas em **operações de recuperação** e **operações de atualização**.

Operações de recuperação

- Não altera dados no arquivo.
- Localiza certos registros de modo que seus valores de campo possam ser examinados e processados.

Operações de atualização

- Altera o arquivo pela inserção, exclusão de registros ou modificação dos valores de campo.

Condição de seleção (ou condição de filtragem)

- Especifica critérios que o registro deve satisfazer.
- No momento de **selecionar** um ou mais registros para recuperação, exclusão ou modificação.
- **Condição de seleção simples** envolve uma comparação de igualdade em algum valor de campo. Exemplos:
  - `Cpf = '1234567899'`
  - `Salario > 30.000`
- O caso geral é ter uma expressão booleana nos campos do arquivo como condição de seleção.

Operações em arquivos

Operações de pesquisa

- Geralmente baseadas em condições de seleção simples.
- Utilizadas para localizar os registros no disco.
- Condições complexas são decompostas (pelo SGBD ou pelo programador).
- Cada registro localizado é verificado para determinar se satisfaz a condição de seleção inteira.

Operações de pesquisa

- Pode-se extrair a condição simples

  ```
  (Departamento = 'Pesquisa')
  ```

- Da condição complexa

  ```
  ((Salario > 30.000) AND (Departamento = 'Pesquisa'));
  ```

Operações de pesquisa

- Se vários registros de arquivo satisfazem uma condição de pesquisa, o *primeiro* registro localizado é designado como **registro atual**.
- Operações de busca subsequentes começam desse registro e localizam o *próximo* registro no arquivo que satisfaz a condição.

Operações em arquivos

Conjunto de operações representativas

- Programas de software de SGBD acessam registros utilizando operações representativas.
- Estas (exceto por *Open* e *Close*) são operações de **um registro por vez**, pois cada uma se aplica a um único registro.

Operações em arquivos

Conjunto de operações representativas

Open

- Prepara o arquivo para a leitura ou gravação.
- Aloca buffers apropriados (pelo menos dois) para manter blocos de arquivo do disco.
- Recupera o cabeçalho do arquivo.
- Define o ponteiro de arquivo para o início do arquivo.

Reset

- Define o ponteiro do arquivo aberto para o início do arquivo.

Conjunto de operações representativas

Find (ou Locate)

- Procura o primeiro registro que satisfaça uma condição.
- Transfere o bloco que contém esse registro para o buffer da memória principal.
- O ponteiro de arquivo aponta para o registro no buffer (registro atual).
- Diferentes verbos indicam se o registro localizado deve ser lido ou atualizado.

Conjunto de operações representativas

Read (ou Get)

- Copia o registro atual do buffer para uma variável de programa no programa do usuário.
- Avança o ponteiro do registro atual para o próximo registro no arquivo.

Conjunto de operações representativas

FindNext

- Procura o próximo registro no arquivo que satisfaz a condição de pesquisa.
- Em seguida, transfere o bloco que contém esse registro para um buffer na memória principal (registro atual).
- Várias formas de FindNext estão disponíveis em SGBDs legados com base nos modelos hierárquicos e de rede.

Operações em arquivos

Conjunto de operações representativas

Delete

- Exclui o registro atual e atualiza o arquivo no disco para refletir a exclusão.

Modify

- Modifica alguns valores de campo para o registro atual e atualiza o arquivo no disco para refletir a modificação.

Close

- Completa o acesso ao arquivo liberando os buffers e realizando outras operações de limpeza.

Conjunto de operações representativas

Insert

- Insere um novo registro no arquivo ao localizar o bloco onde o registro deve ser inserido.
- Transfere tal bloco para um buffer da memória principal, gravando o registro no buffer e gravando o buffer em disco para refletir a inserção.

Conjunto de operações representativas

É possível resumir as operações *Find*, *FindNext* e *Read* na operação **Scan**.

Scan

- Se o arquivo já tiver sido aberto ou reiniciado, Scan retorna o primeiro registro.
- Caso contrário, ele retorna o próximo registro.
- Se uma condição for especificada com a operação, o registro retornado é o primeiro ou o próximo registro que satisfaz a condição.

Operações em arquivos

Conjunto de operações representativas

Operações adicionais de nível mais alto, de **um conjunto de cada vez**, podem ser aplicadas a um arquivo. Exemplos:

- FindAll
- Find (ou Locate)
- FindOrdered
- Reorganize

Operações em arquivos

Conjunto de operações representativas

FindAll

- Localiza todos os registros no arquivo que satisfazem uma condição de pesquisa.

Find (ou Locate) n

- Procura o primeiro registro que satisfaz uma condição e depois continua a localizar os próximos *n-1* registros que satisfazem a mesma condição.
- Transfere os blocos que contém os *n* registros para o buffer da memória principal.

Operações em arquivos

Conjunto de operações representativas

FindOrdered

- Recupera todos os registros no arquivo em alguma ordem especificada.

Reorganize

- Inicia o processo de reorganização.

Operações em arquivos

Arquivos estáticos

- Operações de atualização raramente são realizadas.

Arquivos dinâmicos

- Podem mudar com frequência, de modo que as operações de atualizações são constantemente aplicadas.

Operações em arquivos

- O agrupamento de registros em blocos e organização de blocos em cilindros é de acordo com a situação.
- Em muitos casos, uma única organização não permite que todas as operações necessárias sejam implementadas eficientemente, requer que seja escolhido um meio-termo.

Arquivos de heap

Organização arquivo de heap ou pilha

- Tipo de organização mais simples e mais básica.
- Normalmente, usada com caminhos de acesso adicionais.
- Também é usada para coletar e armazenar registros de dados para uso futuro.
- Registros são arquivados na ordem em que são inseridos (novos registros são inseridos ao final do arquivo).

Arquivos de heap

- Inserção de um novo registro é feita de maneira eficiente.
- O último bloco de disco do arquivo é copiado para um buffer, o novo registro é acrescentado e o bloco é regravado de volta.
- O endereço do último bloco de arquivo é mantido no cabeçalho do arquivo.

Arquivos de heap

Pesquisa linear

- Envolve procurar um registro usando uma condição de pesquisa, pelo bloco de arquivo por bloco.
- Se apenas um registro satisfazer a condição de pesquisa, um programa lerá a memória e pesquisará metade dos blocos de arquivo antes de encontrar (em média).
- Para um arquivo de b blocos, isso exige pesquisar $(b/2)$ blocos, na média.
- Se nenhum registro satisfazer a condição de pesquisa, o programa deve ler e pesquisar todos os b blocos no arquivo.

Arquivos de heap

Exclusão de um registro

- Um programa encontra o bloco, o copia para o buffer, exclui o registro do buffer e **regrava** o bloco de volta ao disco.
- A exclusão de registros resulta em vários espaços desperdiçados.

Arquivos de heap

Exclusão com marcador de exclusão

- Marcador de exclusão é um byte ou bit extra, armazenado em cada registro.
- Um registro é excluído ao se definir o marcador de exclusão com determinado valor.
- Um valor diferente para o marcador indica um registro válido (não excluído).
- Programas de pesquisa consideram apenas registros válidos.

Arquivos de heap

- As duas técnicas citadas exigem **reorganização** periódica do arquivo para retomar o espaço não usado.
- Na reorganização, os blocos de arquivo são acessados de maneira consecutiva e os registros são compactados pela remoção de registros excluídos.
- Após a organização, os blocos são preenchidos até a capacidade.
- Outra possibilidade é usar o espaço dos registros excluídos ao inserir novos registros.

Arquivos de heap

- Usamos a organização espalhada ou não espalhada para um arquivo desordenado, com registros de tamanho fixo ou variável.
- A modificação de um registro de tamanho variável exige a exclusão do registro antigo e a inserção de um registro modificado.
- Para ler todos os registros na ordem dos valores de algum campo, criamos uma cópia classificada do arquivo.
- Por ser uma operação de custo alto, técnicas especiais para **classificação externa** são utilizadas.

Arquivos de heap

Para um arquivo de *registros de tamanho fixo* desordenados usando *blocos não espalhados e alocação contígua*, é simples acessar qualquer registro por sua **posição** no arquivo.

Arquivo relativo ou direto

- Registros podem ser facilmente acessados por suas posições relativas.
- Não auxilia a localizar um registro com base em uma condição de busca.
- Facilita a construção de caminhos de acesso no arquivo, como os índices.

Arquivos de registros ordenados

Os arquivos ordenados estão em blocos armazenados em cilindros contíguos para minimizar o tempo de busca.

Realizada a partir de uma pesquisa binária nos blocos dos arquivos de disco.

- Acesso de $\log_2(b)$ blocos;
- Melhoria se comparada a pesquisa linear, onde são acessados:
  - $(b/2)$ blocos, quando o registro é encontrado;
  - b blocos, quando o registro não é encontrado.

Arquivos de registros ordenados

Tempos de acesso médios para um arquivo de **b** blocos:

Tipo de organização	Método de acesso/pesquisa	Média de blocos para acesso
Heap (não ordenado)	Varredura sequencial	$b/2$
Ordenado	Varredura sequencial	$b/2$
Ordenado	Pesquisa binária	$\log_2 b$

- Um critério de pesquisa envolvendo as condições >, <, >= e <= no campo de ordenação é muito eficiente;
  - A ordenação física dos registros indica que todos que satisfazem a condição são contíguos no arquivo.
- Grande dificuldade na prática de **inserção, exclusão** e **alteração**.

Arquivos de registros ordenados

Dificuldade na prática de inserção

1. Para inserir, deve-se encontrar sua posição correta no arquivo, com base no campo de ordenação;
2. Criar um espaço no arquivo para inserir registro na posição definida acima;
3. "Deslocar"os registros sucessivos para as posições consecutivas.

Para um arquivo grande, é um processo demorado, uma vez que metade dos blocos do arquivo deve ser lida e regravada após o deslocamento dos registros.

Alternativas para eficiência da inserção

1. Manter algum espaço não usado em cada bloco para novos registros;
    - O problema original reaparece ao ser totalmente utilizado.

Alternativas para eficiência da inserção

2. Criar um arquivo desordenado temporário, chamado de **overflow** ou **transação**.
   - Arquivo ordenado real chamado de **principal** ou **mestre**;
   - Funcionamento:
     - Os novos registros que deveriam ser inseridos ao mestre, são inseridos no final do overflow;
     - De tempos em tempos, o arquivo de overflow é classificado e mesclado ao mestre.

Alternativas para eficiência da inserção

2. Criar um arquivo desordenado temporário, chamado de **overflow** ou **transação**.
   - Maior complexidade no algoritmo de pesquisa;
     - O overflow precisa ser consultado (pesquisa linear) quando, após uma pesquisa binária, o registro não tiver sido encontrado no mestre.
   - O overflow pode ser ignorado em aplicações que não exigem informações mais atualizadas.

Arquivos de registros ordenados

A complexidade da **alteração** de um registro, depende de:

1. Condição de pesquisa para localizar o registro
2. O campo a ser modificado

[FATOR 1] A condição de pesquisa envolve o campo chave de ordenação?

- Sim: Utiliza-se a pesquisa binária.
- Não: Utiliza-se a pesquisa linear.

Arquivos de registros ordenados

A complexidade da **alteração** de um registro, depende de:

1. Condição de pesquisa para localizar o registro
2. O campo a ser modificado

[FATOR 2] O campo a ser alterado é o campo ordenado do arquivo ?

- Sim: Dependendo do novo valor, este campo pode ter sua posição física alterada no arquivo.
  - Requer a exclusão do registro antigo e a inserção do registro modificado.
- Não: A modificação deste campo é realizada e este é regravado no mesmo local físico do disco.

Arquivos de registros ordenados

Aplicações de banco de dados, normalmente, utilizam arquivos ordenados com caminho de acesso adicional (**índice primário**), chamados de:

Arquivo sequencial-indexado

- Otimiza o tempo de acesso aleatório ao campo-chave de ordenação;
- Se o atributo de ordenação não for uma chave, o arquivo é chamado de **arquivo agrupado**.

Técnicas de hashing

- Outro tipo de organização do arquivo principal;
- Acesso muito eficiênte aos registros sob determinadas condições de pesquisa.

Arquivo de Hash

- A condição de pesquisa precisa ser uma condição de igualdade em um único campo: **campo de hash**
  - Na maior parte dos casos também é um campo-chave, chamado de **chave hash**.

Arquivo de Hash - Funcionamento

- Oferece uma função de hash **h**, onde se aplica o valor do campo de hash de um registro e gera o endereço do bloco de disco em que o registro está armazenado;
- A pesquisa pode ser realizada pelo buffer da memória principal.

Uma boa função de hashing distribui os registros de maneira uniforme pelo espaço de endereços de modo a **minimizar as colisões enquanto não deixam muitos locais não usados**.

Hash Interno

- Implementado através de uma **tabela de hash** por um array de registros;
- Exemplo de estrutura de dados de hashing interno - array de M posições para uso no hashing:

	Nome	Cpf	Cargo	Salario
0				
1				
2				
3				
	...			
M-2				
M-1				

Hash Interno - Exemplo

- Uma função hash comum seria:

$$h(K) = K \bmod M$$

Retorna o resto de um valor de campo de hash inteiro K após a divisão por M: esse valor é usado para o endereço do registro.

Hash Interno - Colisão

- Ocorre quando, na inserção, a função de um campo de hash resulta em um endereço já utilizado.
- É necessário localizar outra posição disponível:
  - **Resolução de colisão**
- Métodos para resolução de colisão:
  1. Endereçamento Aberto
  2. Encadeamento
  3. Hashing múltiplo

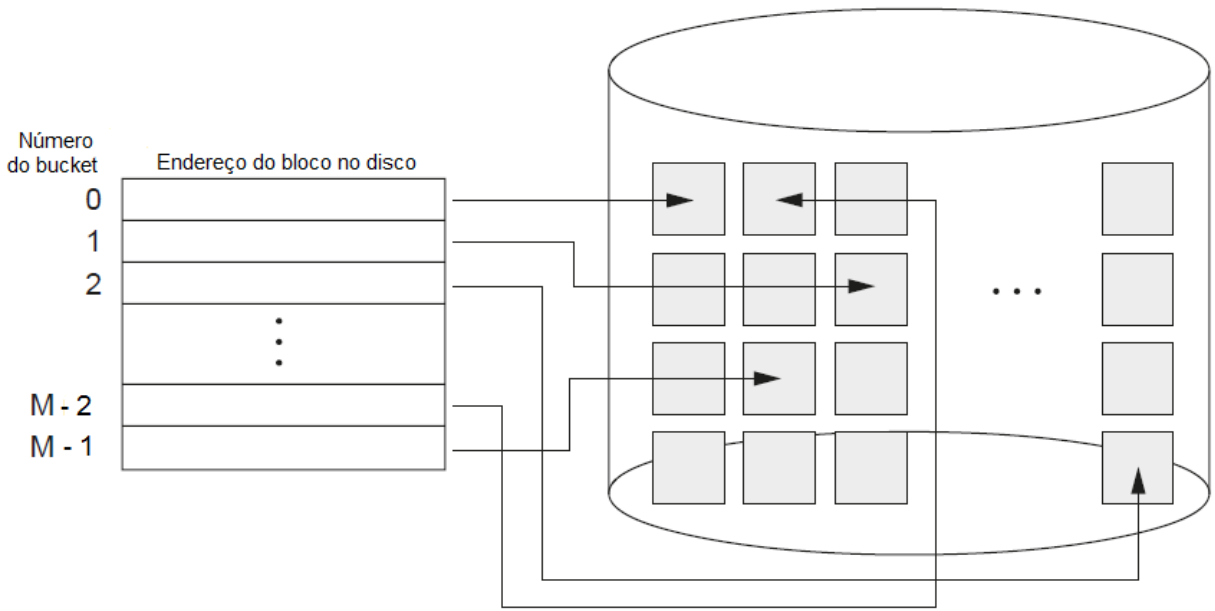
    - Cada método requer os próprios algoritmos para inserção, recuperação e exclusão de registros.

Hash Interno - Colisão (Métodos de resolução)

1. Endereçamento Aberto
   - Verifica as posições subsequentes, a partir da posição ocupada, até que uma posição vazia seja localizada.
2. Encadeamento
   - É mantida uma lista composta por registros de overflow para cada endereço de hash.
3. Hashing múltiplo
   - Aplica-se uma segunda função hash se a primeira resultar em uma colisão.

Hash Externo

- Destinado a arquivos de disco;
- O espaço de endereços de destino é feito em *buckets*, cada um mantendo vários registros.
  - **Bucket:** bloco de disco ou cluster de blocos de discos contíguos.



Hash Externo - Funcionamento

1. A função de hashing mapeia uma chave em um número de bucket;

2. Uma tabela mantida no cabeçalho do arquivo converte o número do bucket para o endereço do bloco de disco correspondente.

Hash Externo - Colisão

- Menos complexa;
  - Independentemente de quantos registros estejam em um bucket, eles podem ser definidos por hashing ao mesmo bucket sem causar problemas.

- É previsto o caso em que um bucket está cheio e um novo registro a ser inserido tem um hash para esse bucket.
  - Utiliza-se a técnica semelhante a de **Encadeamento**:
  - Os ponteiros na lista devem ser "ponteiros de registro", que incluem um endereço de bloco e uma posição de registro relativa no bloco.

Hash Externo - Estático

- Funções definidas, partindo do princípio que há um número fixo de buckets **M** alocado.
- Desvantagem para arquivos dinâmicos - *Exemplo*:
  1. Alocamos M buckets para o espaço de endereços e deixamos o número máximo m de registros que podem caber em um bucket. Ou seja, caberão m * M registros no espaço alocado. Caso o número de registros for muito menor ou muito maior do que m * M, é necessário alterar o número de blocos M alocados e depois usar uma nova função de hashing para redistribuir os registros.
     - Reorganização muito demorada para arquivos grandes.

Hash Externo - Busca de registro

- Se for buscado um registro em um campo diferente do campo de hash, o comportamento da pesquisa será semelhante a um arquivo desordenado.

Hash Externo - Exclusão de registro

- É realizada a partir da remoção do próprio registro em seu bucket.
  - Se o bucket tiver uma cadeia de overflow:
    1. Mover um dos registros de overflow para os excluídos;
    2. Substituir o registro excluído.
  - Se o registro a ser excluído já tiver no overflow:
    1. Remover o registro da lista.

Hash Externo - Alteração de registro

- Depende de:
  1. Condição de pesquisa para localizar o registro
  2. O campo a ser modificado

[FATOR 1] É uma comparação de igualdade no campo de hash?

- Sim: Utiliza-se a função de hash.
- Não: Utiliza-se a pesquisa linear.

Hash Externo - Alteração de registro

- Depende de:
  1. Condição de pesquisa para localizar o registro
  2. O campo a ser modificado

[FATOR 2] O campo a ser alterado é o campo de hash?

- Sim: Dependendo do novo valor, este campo pode se mover para outro bucket.
  - Requer a exclusão do registro antigo e a inserção do registro modificado.
- Não: O campo é modificado e gravado no mesmo bucket.

Técnicas de hashing

Técnicas para expansão dinâmica

- Técnicas de hashing que permitem a expansão dinâmica do arquivo:
  1. Hashing extensível
  2. Hashing dinâmico
  3. Hashing linear

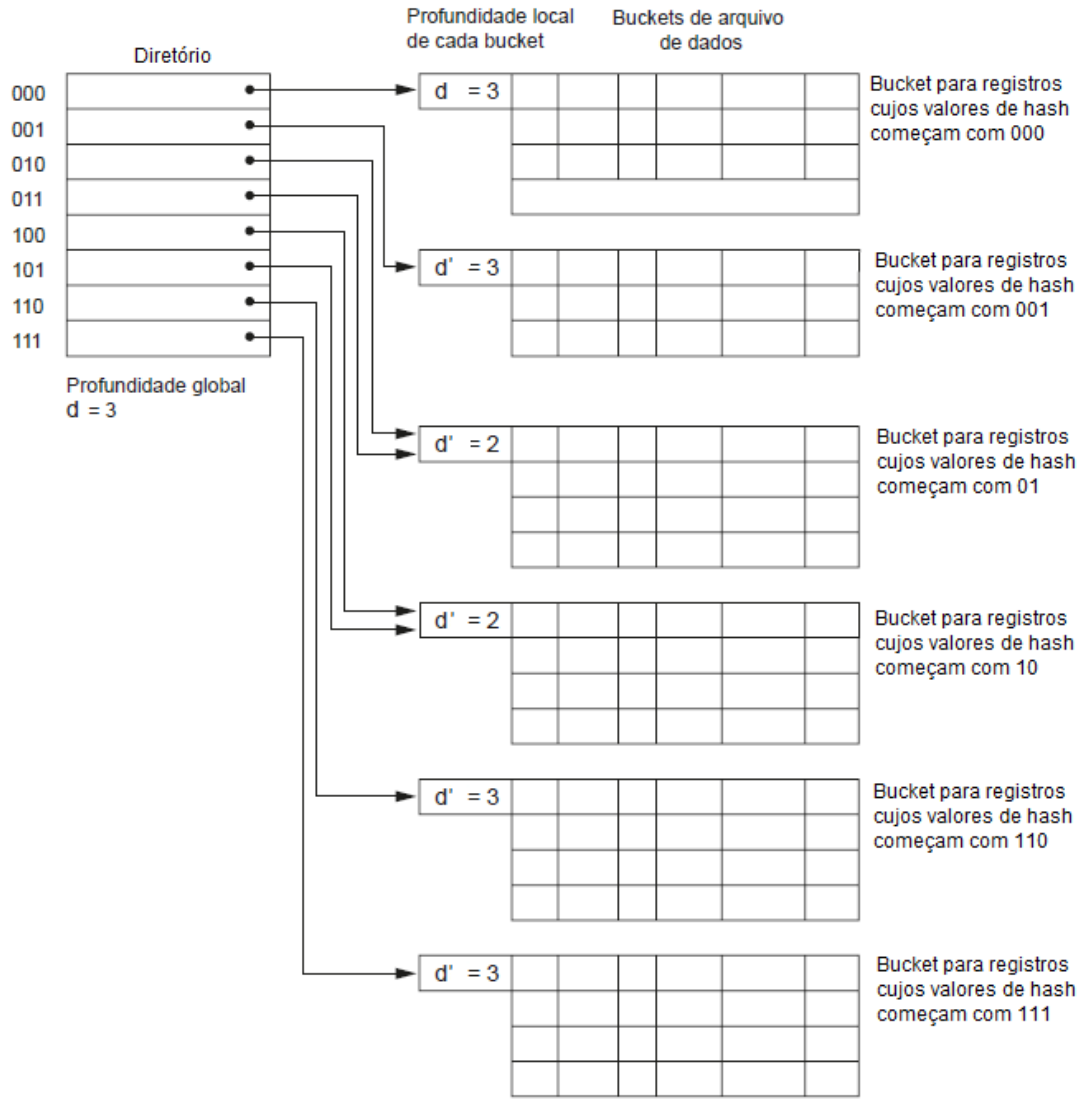
- O resultado da aplicação de uma função de hashing **é um inteiro não negativo** e **pode ser representado como um número binário**;

- Estrutura de acesso feita em "representação binária"do resultado da função de hashing (string de "bits").

Técnicas para expansão dinâmica - Hashing extensível

- Um array de 2^d endereços de bucket é mantido, chamado de **diretório**.
  - **d** é chamado de profundidade global do diretório.
- O valor inteiro correspondente aos primeiros d' bits de um valor de hash é utilizado como um índice para o array determinar uma entrada de diretório e o endereço nessa entrada determina o bucket em que os registros correspondentes estão armazenados;
- Uma profundidade local d' especifica o número de bits em que os conteúdos dos buckets são baseados.

Técnicas para expansão dinâmica - Hashing extensível



Técnicas para expansão dinâmica - Hashing extensível

- O valor de *d* pode ser aumentado ou diminuído, dobrando ou reduzindo á metade do número de entradas no array do diretório;
- A maioria das recuperações de registro exige dois acessos de bloco - um para o diretório e outro para o bucket;
- Desvantagem:
  - O diretório precisa ser pesquisado antes do acesso aos próprios buckets, resultando em dois acessos ao bloco.

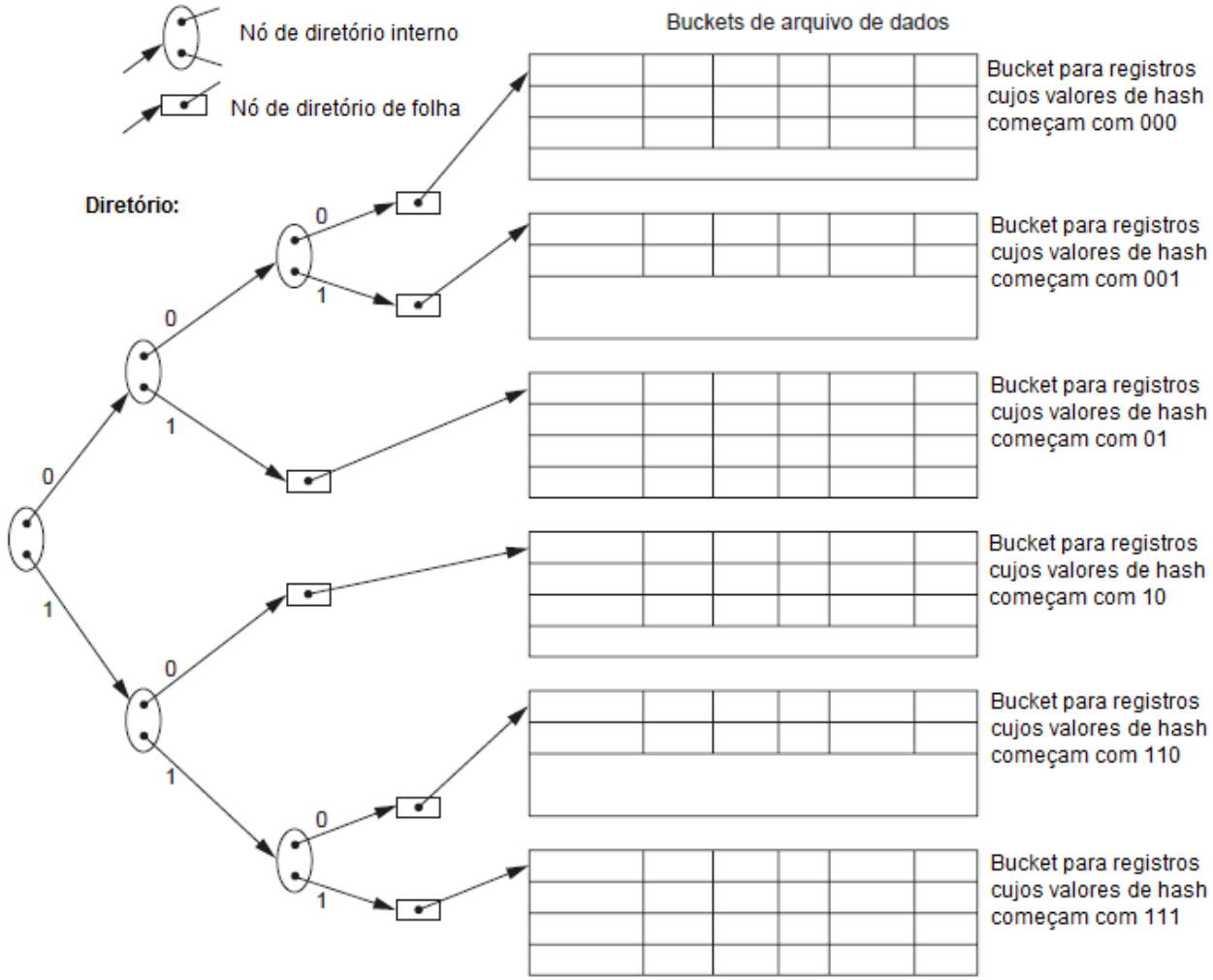
Técnicas para expansão dinâmica - Hashing extensível

- Vantagens:
  - O desempenho da busca não degrada enquanto o arquivo cresce;
  - Nenhum espaço é alocado no hashing extensível para crescimento futuro
    - Buckets adicionais podem ser alocados de maneira dinâmica conforme a necessidade;
  - O tamanho máximo do diretório é 2^k, onde **k** é o número de bits no valor de hash.

Técnicas para expansão dinâmica - Hashing dinâmico

- Armazenamento de registros em buckets semelhante ao hashing extensível.
  - Se difere na organização do diretório: ao invés de profundidade global, utiliza-se um diretório estruturado em árvore com dois tipos de nós:
    1. **Nós internos com dois ponteiros:** O ponteiro esquerdo representa o bit 0 e o ponteiro direito representa o bit 1.
    2. **Nós folha:** Mantém um ponteiro para o bucket real com registros.

Técnicas para expansão dinâmica - Hashing dinâmico



Técnicas para expansão dinâmica - Hashing linear

- Permite que um arquivo de hash expanda e encolha seu número de buckets dinamicamente sem precisar de um diretório.
- **Funcionamento:**
  1. O arquivo começa com **M** buckets (0, 1, ..., M-1) e usa a função **h(K) = K mod M** - função de hash inicial (h_i);
  2. Quando a colisão leva a um registro de estouro em qualquer bucket de arquivo, o primeiro bucket (0) é dividido em dois: o bucket original (0) e um novo bucket (**M**) ao final do arquivo.

Técnicas para expansão dinâmica - Hashing linear

- **Funcionamento:**
  3. Os registros do bucket 0 são distribuídos entre dois buckets em uma função diferente $h_{i+1}(K) = K \bmod 2^M$.
     - *A medida que mais colisões levem registros a overflow, buckets adicionais são divididos na ordem linear 1, 2, 3...*
  4. Os registros do overflow são redistribuídos em buckets regulares, usando a função h_{i+1}, por meio de uma *divisão adiada* dos buckets.
     - *Sempre que ocorrer uma divisão, um inteiro* **n** *é incrementado por 1, para ao final, determinar a quantidade de buckets divididos.*

Técnicas para expansão dinâmica - Hashing linear

- **Passos -** Recuperar um valor de chave hash K:
  1. Aplicar a função h_i;
  2. Se $h_i(K) < n$, aplicar a função h_{i+1} (porque o bucket já está dividido).

Quando $n = M$, significa que todos os buckets originais foram divididos e a função hash h_i+1 já não é mais suficiente. Deste modo, cria-se a $h_{i+2}(K) = K \bmod 4^M$.

Técnicas para expansão dinâmica - Hashing linear

- **Passos -** Recuperar um valor de chave hash K:
  - A divisão pode ser controlada ao monitorar o fator de carga do arquivo (**l**) ao invés de dividir sempre que ocorrer um overflow.
    - Definido como **l = r/(*bfr* * N)**, onde

      r = número atual de registros do arquivo;
      brf = número máximo de registros que cabem no bucket;
      N = número atual de buckets do arquivo.

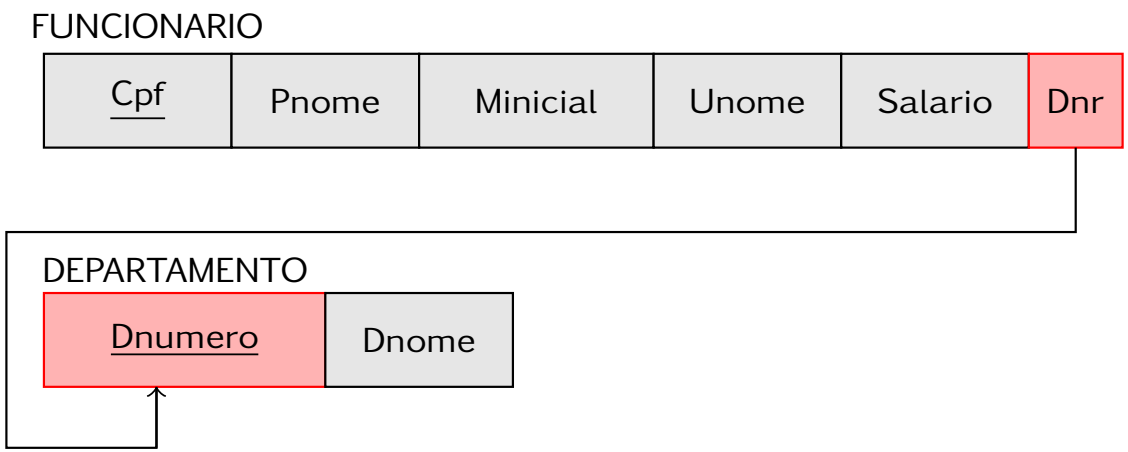
    - Vantagem:

      Mantém o fator de carga razoavelmente constante enquanto o arquivo aumenta e diminui e não requer um diretório.

Outras organizações de arquivo primárias

Arquivos de registros mistos

- Na maioria das aplicações de banco de dados, existem arquivos com registros de diferentes tipos;
- Os relacionamentos entre registros em vários arquivos podem ser representados por **campos de conexão**;
- Exemplo:



Outras organizações de arquivo primárias

Arquivos de registros mistos

- Se for necessário recuperar valores de campo de dois registros relacionados:
  1. Recuperar um dos registros primeiro;
  2. Utiliza-se o valor do item 1 para recuperar o registro relacionado no outro arquivo;

Os relacionamentos são implementados por referências de campo lógicas entre os registros de arquivos distintos.

Outras organizações de arquivo primárias

Arquivos de registros mistos

- Para os SGBDs **orientado a objeto, hierárquicos** e de **rede**:
  - Implementam relacionamentos entre registros como relacionamentos físicos realizados pela continuidade física de registros relacionados ou por ponteiros físicos;
  - Registros de diferentes tipos podem ser **fisicamente agrupados** no disco.

A implementação física do relacionamento (utilizado com frequência) pode aumentar a eficiência do sistema na recuperação dos registros.

Outras organizações de arquivo primárias

Arquivos de registros mistos

- Presença do campo **tipo de registro**:
  - Diferencia os registros em um arquivo misto;
  - Posicionado antes do valor do campo;
  - Utilizado pelo software para determinar o tipo de registro que está prestes a ser processado.

Usando a informação do catálogo, o SGDB, determina os campos desse tipo de registro e seus tamanhos, a modo de interpretar seus valores.

Paralelizando o acesso de disco

RAID

- *Redundant Array of Inexpensive Disks*;
- Nivelar as diferentes taxas de melhoria de desempenho dos discos com as taxas da memória e dos microprocessadores:
  - As capacidades RAM têm se quadruplicado a cada dois ou três anos;
  - Os tempos de acesso do disco estão melhorando 10% ao ano;
  - As taxas de transferência do disco estão melhorando 20% ao ano.

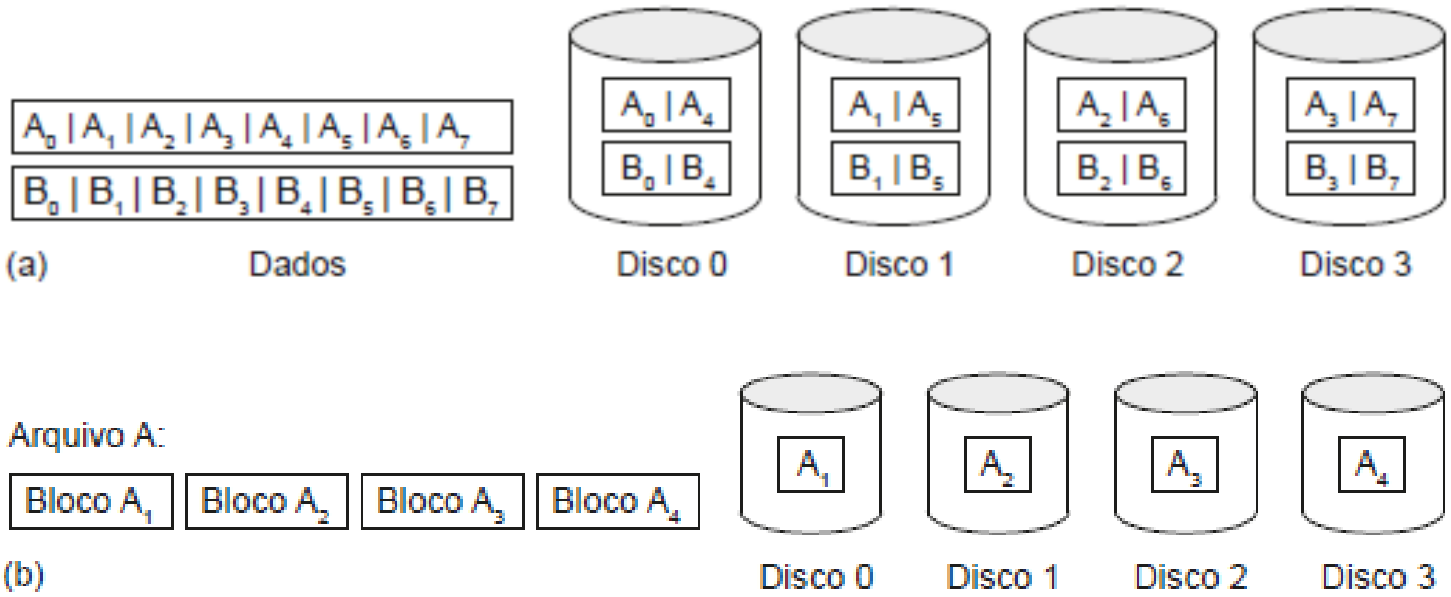
Paralelizando o acesso de disco

RAID

- Criação de um grande array de pequenos discos independentes;
- **Striping de dados**
  - Emprega o paralelismo (melhora o desempenho do disco);
  - Distribui os dados transparentemente por vários discos;
  - Melhora o desempenho de E/S, oferecendo assim altas taxas de transferência gerais;
  - Balanceia a carga entre os discos.

Paralelizando o acesso de disco

RAID



Striping de dados em vários discos. **(a)** Striping em nível de bit em quatro discos. **(b)** Striping em nível de bloco em quatro discos.

RAID - Melhorando a confiabilidade

- Para um array de **n** discos, a probabilidade de falha é de **n** vezes.

Emprega-se a **redundância de dados**:

- Desvantagens:
  - Operações de E/S adicionais para gravação;
  - Processamento extra para manter a redundância e realizar recuperação de erros.

RAID - Melhorando a confiabilidade

- Para um array de **n** discos, a probabilidade de falha é de **n** vezes.

Emprega-se a **redundância de dados**:

- **Sombreamento** ou **Espelhamento**
  - Técnica para introdução de redundância;
  - Dobra a taxa em que as solicitaçoes de leitura são tratadas;
  - A taxa de transferência de cada leitura permanece igual à taxa para um único disco.

RAID - Melhorando a confiabilidade

- Para um array de **n** discos, a probabilidade de falha é de **n** vezes.

- Emprega-se a **redundância de dados**:
  - **Sombreamento** ou **Espelhamento**
    1. Os dados são gravados em dois discos físicos idênticos;
    2. Na leitura, buscar a partir de um disco com menor atraso de fila, busca e rotacionais;
    3. Caso algum disco falhe, utilizar outro disco, até que o primeiro seja reparado.

RAID - Melhorando a confiabilidade

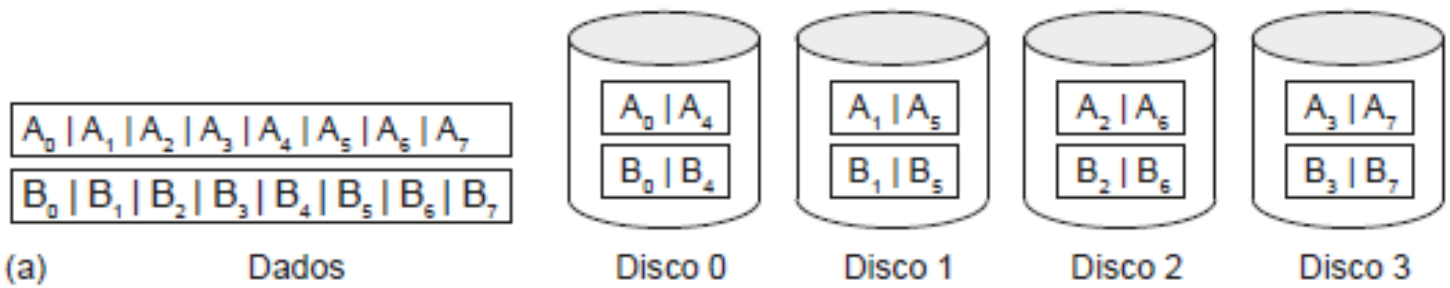
- Para um array de **n** discos, a probabilidade de falha é de **n** vezes.

- Emprega-se a **redundância de dados**:

- Necessário selecionar uma técnica para calcular a informação redundante.
  - Utilizar códigos de correção de erros (bits de paridade).
- Necessário selecionar um método de distribuição de informação redundante pelo array de disco.
  - Armazenar a informação redundante em um pequeno número de discos.

RAID - Melhorando o desempenho

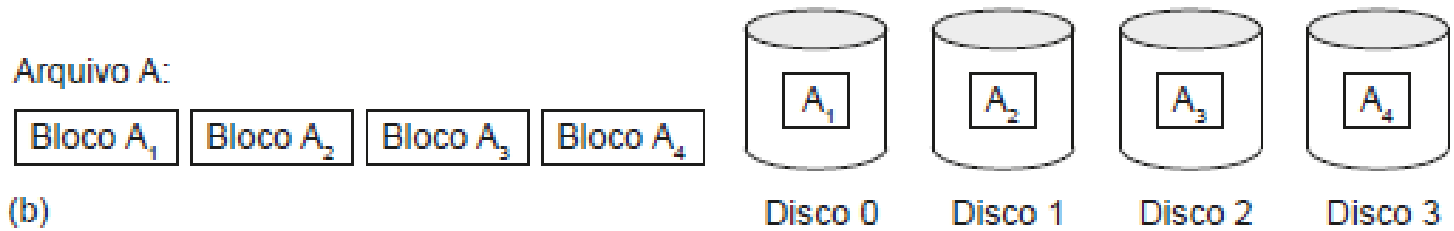
1. Striping de dados em nível de bit
   - Dividir um byte de dados e gravar o bit **j** no **j-ésimo** disco;
   - Com bytes de 8 bits, oito discos físicos podem ser considerados um disco lógico, com um aumento de oito vezes na taxa de transferência de dados.
   - Exemplo:



RAID - Melhorando o desempenho

2 Striping de dados em nível de bloco

- Blocos de um arquivo são espalhados pelos discos;
- Exemplo:



- Solicitações independentes que acessam blocos isolados podem ser atendidas em paralelo por discos separados, diminuindo o tempo de enfileiramento das solicitações de E/S.

Redes de área de armazenamento

- SANs - *Storage Area Networks*;
- Periféricos de armazenamento online configurados como nós em uma rede de alta velocidade;
- Alternativas de arquitetura:
  - Conexões ponto a ponto;
  - Uso de canal de fibra;
  - Uso de hubs e switches de canal de fibra.
- Vantagens:
  - Conectividade flexível;
  - Melhores capacidades de isolamento.

Armazenamento conectado à rede

- NAS - *Network-Attached Storage*;
- Somente permitem o acréscimo de armazenamento para compartilhamento de arquivos;
- Substituição dos servidores de arquivos tradicionais;
- Alto grau de escalabilidade, confiabilidade, flexibilidade e desempenho;
- Operação confiável e administração fácil;
- Residem em qualquer local de uma rede local (LAN).

Sistemas de armazenamento iSCSI

- Os clientes enviam comandos SCSI para dispositivos de armazenamento SCSI em canais remotos;
- Não exige cabeamento especial;
- Simplicidade e baixo custo;
- Funcionamento para o SGBD acessar os dados:
  1. O SO gera comandos SCSI apropriados e, dependendo do caso, encapsula e criptografa;
  2. Um cabeçalho do pacote é acrescentado antes que os pacotes IP resultantes sejam transmitidos por uma conexão Ethernet;

Sistemas de armazenamento iSCSI

- Os clientes enviam comandos SCSI para dispositivos de armazenamento SCSI em canais remotos;
- Não exige cabeamento especial;
- Simplicidade e baixo custo;
- Funcionamento para o SGBD acessar os dados:
  3. Quando um pacote é recebido, ele é descriptografado;
  4. Os comandos SCSI seguem por meio do controlador SCSI para o dispositivo de armazenamento SCSI.

Seção 18
Indexação de Arquivos

ESTRUTURAS DE INDEXAÇÃO PARA ARQUIVOS:

Tipos de índices ordenados de único nível

Tipos de índices ordenados de único nível

Índice

- Índices são utilizados para **agilizar a recuperação de registros** em resposta a certas condições de pesquisa.
- As estruturas de índice são arquivos adicionais no disco que oferecem caminhos alternativos para acesso a registros.
- Para um arquivo com determinados campos (atributos), uma estrutura de acesso a índice normalmente é definida em um único campo.
- Chamado **campo de índice** (ou **atributo de indexação**).

Tipos de índices ordenados de único nível

Índice

- O índice armazena cada valor do **campo de índice** junto com uma **lista de ponteiros** para todos os blocos de disco que contêm registros com esse valor de campo.
- Para encontrar um registro, o índice é pesquisado, o que leva a ponteiros para um ou mais blocos de disco onde os registros exigidos estão localizados.

Tipos de índices ordenados de único nível

Índices de único nível

- Baseados em arquivos ordenados.
- Os valores no índice são *ordenados* de modo que possamos realizar uma *pesquisa binária* no índice.
- Tipos de índices ordenados:
  - Índice primário;
  - Índice de agrupamento (clustering);
  - Índice secundário.

Tipos de índices ordenados de único nível

Índice primário

- Atua como uma estrutura de acesso para procurar e acessar os registros em um arquivo.
- É um **arquivo ordenado** cujos registros são de tamanho fixo com dois campos:
  - o primeiro campo é do mesmo tipo do **campo de chave de ordenação (chave primária)** do arquivo;
  - e o segundo campo é um ponteiro para um bloco de disco (um endereço de bloco).

Tipos de índices ordenados de único nível

Índice primário

- O índice primário é especificado no *campo de chave de ordenação*.
- Um **campo de chave de ordenação** é usado para *ordenar* fisicamente os registros de arquivo no disco.
- Cada registro tem um valor *único* para esse campo.

Tipos de índices ordenados de único nível

Índice primário

- No arquivo de índice há uma **entrada de índice (registro de índice)** para cada *bloco no arquivo de dados*.
- Na entrada de índice há o valor da chave primária para o *primeiro* registro em um bloco e um ponteiro para esse bloco com seus dois valores de campo.
- Valores de campo da entrada de índice i:

```
<K(i), P(i)>
```

Tipos de índices ordenados de único nível

Exemplo: *Criar um índice primário no arquivo ordenado abaixo.*

Bloco 1

Nome	Cpf	Data_nascimento	Cargo	Salario	Sexo
Aaron, Eduardo					
Abílio, Diana					
•••					
Acosta, Marcos					

Bloco 2

Adams, João					
Adams, Roberto					
•••					
Akers, Janete					

Bloco 3

Alexandre, Eduardo					
Alfredo, Roberto					
•••					
Allen, Samuel					

Alguns blocos de um arquivo ordenado (sequencial) de registros de **FUNCIONARIO** com **Nome** como campo-chave de ordenação (campo exclusivo).

Tipos de índices ordenados de único nível

Exemplo: *Criar um índice primário no arquivo ordenado abaixo.*

- Sendo **Nome** exclusivo, usamos esse campo como chave primária (chave de ordenação do arquivo).
- Cada entrada no índice tem um valor de *Nome* e um ponteiro.

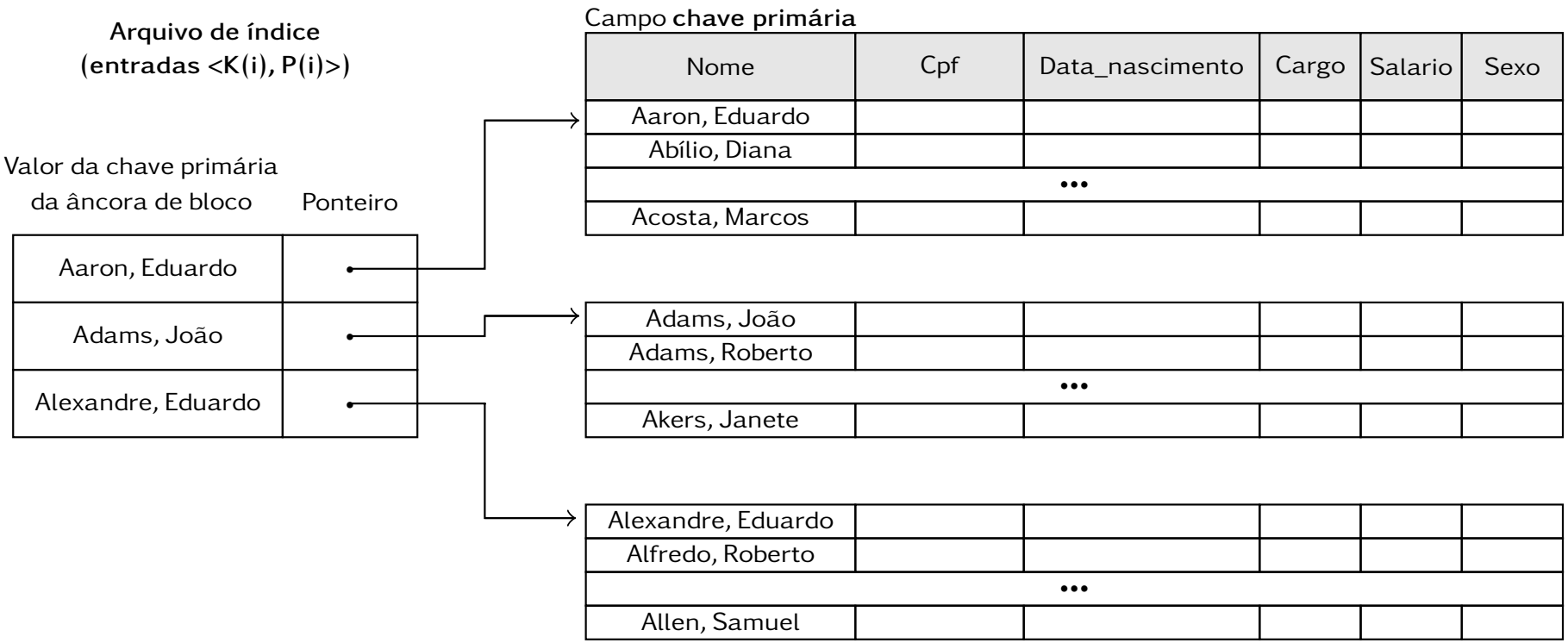
<K(1) = (Aaron, Eduardo), P(1) = endereco de bloco 1>

<K(2) = (Adams, Joao), P(2) = endereco de bloco 2>

<K(3) = (Alexadre, Eduardo), P(3) = endereco de bloco 3>

Tipos de índices ordenados de único nível

Exemplo: *Criar um índice primário no arquivo ordenado abaixo.*



Índice primário no campo de chave de ordenação.

Tipos de índices ordenados de único nível

Índice primário

- O número de entradas no índice é igual ao *número de blocos de disco* no arquivo ordenado.
- O primeiro registro em cada bloco é **registro de âncora** do bloco (**âncora do bloco**).

Tipos de índices ordenados de único nível

Índice primário

- Os índices também podem ser caracterizados como **densos** ou **esparsos**.
  - **Índice denso** tem uma entrada de índice para *cada valor de chave de pesquisa* (cada registro) no arquivo.
  - **Índice esparso** (ou **não denso**) tem entradas de índice para somente alguns dos valores de pesquisa.
- Um índice primário é um índice esparso, pois inclui uma entrada para cada bloco de disco do arquivo e as chaves de seu registro de âncora (ao invés de cada registro).

Tipos de índices ordenados de único nível

Índice primário

- O arquivo de índice (para um índice primário) ocupa um espaço menor do que o arquivo de dados.
  - existem *menos entradas de índice* do que registros;
  - cada entrada de índice normalmente é *menor em tamanho* do que um registro de dados.
- Em consequência, mais entradas de índice do que registros podem caber em um bloco.

Tipos de índices ordenados de único nível

Índice primário

- Portanto, uma pesquisa binária no arquivo de índice requer menos acessos de bloco do que uma pesquisa binária no arquivo de dados.
- Uma pesquisa binária requer aproximadamente $\mathbf{log_2 b_i}$ acessos de bloco para um índice com b_i blocos.
- Se o arquivo de índice primário possuir b_i blocos, localizar um registro com um valor de chave de pesquisa exige uma pesquisa binária desse índice e o acesso ao bloco que contém esse registro: $\log_2 b_i + 1$ acessos.

Tipos de índices ordenados de único nível

Índice primário

- Um registro cujo valor da chave primária é K se encontra no bloco de endereço $P(i)$, onde $K(i) <= K < K(i + 1)$.
- O i-ésimo bloco no arquivo de dados contém todos os registros.
- Para recuperar um registro, realiza-se uma pesquisa binária no arquivo de índice para encontrar a entrada de índice apropriada i.

  Depois recupera-se o bloco do arquivo de dados cujo endereço é $P(i)$.

Tipos de índices ordenados de único nível

Índice primário

EXEMPLO 1:

Suponha um arquivo ordenado com r = 30.000 registros em um disco com tamanho de bloco B = 1.024 bytes. Os registros são de tamanho fixo com tamanho R = 100 bytes.

- O fator de bloco para o arquivo seria
  $bfr = [(B/R)] = [(1.024/100)] = 10$ registros por bloco.
- O número de blocos necessários para o arquivo é
  $b = [(r/bfr)] = [(30.000/10)] = 3.000$ blocos.
- Uma pesquisa binária no arquivo de dados precisaria de
  $[\log_2 b] = [(\log_2 3.000)] = 12$ acessos de bloco.

Tipos de índices ordenados de único nível

Índice primário

EXEMPLO 1:

Suponha que o campo de chave de ordenação do arquivo seja V = 9 bytes de extensão, ponteiro de bloco seja P = 6 bytes de extensão e tenhamos construído um índice primária para o arquivo.

- O tamanho de cada entrada de índice é
  $R_i = (9 + 6) = 15$ bytes.
- O fator de bloco para o índice é
  $bfr_i = [(B/R_i)] = [(1.024/15)] = 68$ entradas por bloco.

Tipos de índices ordenados de único nível

Índice primário

EXEMPLO 1:

- O número total de entradas de índice r_i é 3.000 (igual ao número de blocos no arquivo de dados).
- O número de blocos de índice é

$$b_i = [(r_i/bfr_i)] = [(3.000/68)] = 45 \text{ blocos.}$$

- Para uma pesquisa binária no arquivo de índice seriam necessários $[(\log_2 b_i)] = [(\log_2 45)] = 6$ acessos de bloco.
- Para procurar um registro usando o índice, precisa-se de um acesso de bloco adicional ao arquivo de dados,

  $6 + 1 = 7$ acessos a bloco de disco (melhoria em relação a pesquisa binária no arquivo de dados).

Tipos de índices ordenados de único nível

Índice primário: Inserção e exclusão de registros

- Em um índice primário temos de mover registros e mudar algumas entradas de índice para inserção e exclusão,

  pois a movimentação de registros mudará os *registros de âncora* de alguns blocos.

- Possibilidades:
  - Usar um arquivo de overflow desordenado.
  - Usar uma lista ligada de registros de overflow para cada bloco.

Tipos de índices ordenados de único nível

Índice primário: Inserção e exclusão de registros

- Os registros em cada bloco e sua lista ligada de overflow podem ser classificados para melhorar o tempo de recuperação.
- A exclusão de registro é tratada com marcadores de exclusão.

Tipos de índices ordenados de único nível

Índice de agrupamento (clustering)

- Se registros no arquivo puderem ter o mesmo valor para o campo de ordenação (campo de ordenação não é um campo de chave).
- O arquivo de dados é chamado de **arquivo agrupado**.
- Um arquivo pode ter no máximo um índice primário ou de agrupamento (máximo um campo de ordenação), *mas não ambos*.

Tipos de índices ordenados de único nível

Índice de agrupamento (clustering)

- Se os registros forem fisicamente ordenados em um campo não chave (que *não* tem um valor distinto para cada registro) esse campo é um **campo de agrupamento**.
- O arquivo de dados é chamado de **arquivo agrupado**.
- O **índice de agrupamento** agiliza a recuperação de todos os registros que têm o mesmo valor para o campo de agrupamento.

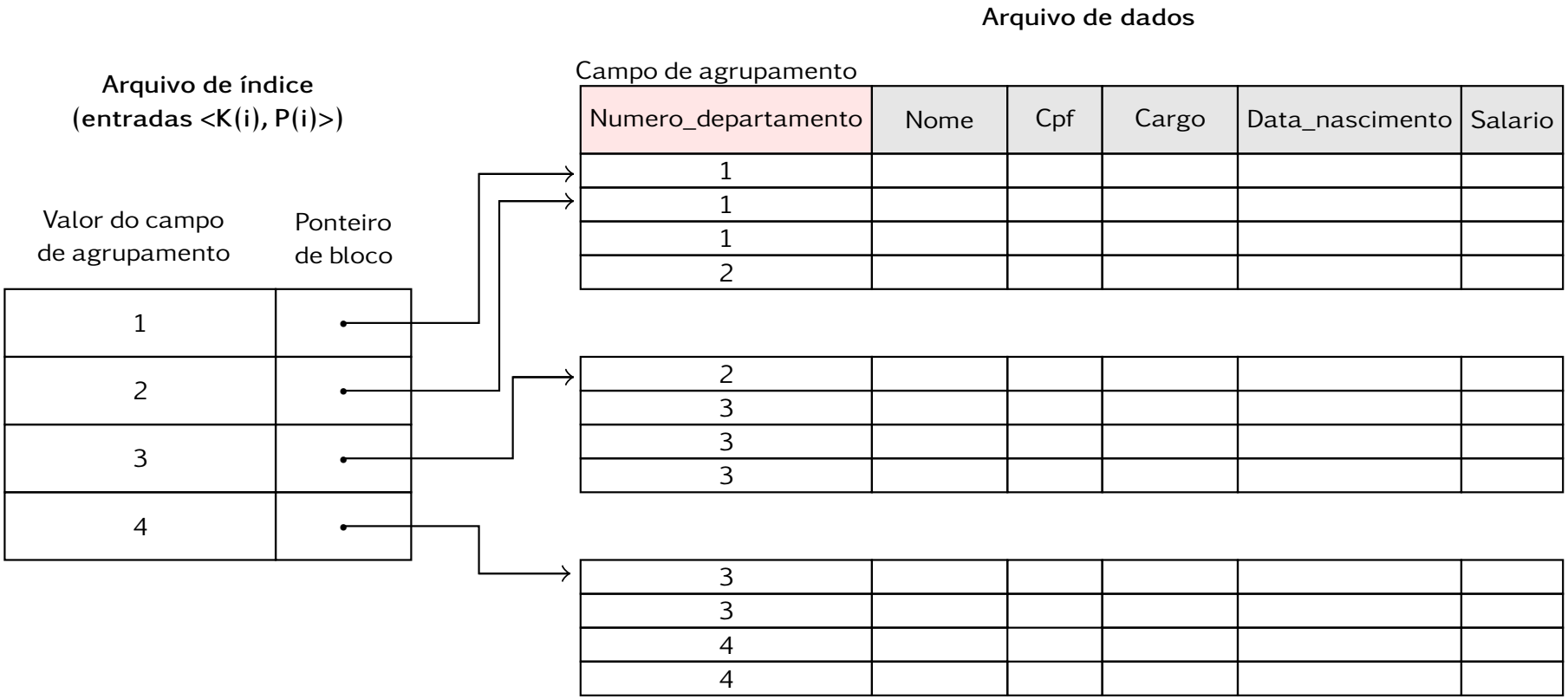
Tipos de índices ordenados de único nível

Índice de agrupamento (clustering)

- Um índice de agrupamento é um arquivo ordenado com dois campos:
  - o primeiro campo é do mesmo tipo do campo de agrupamento;
  - e o segundo campo é um ponteiro para o ***primeiro bloco*** no arquivo que tem um registro com esse valor para seu campo de agrupamento.
- Há uma entrada no índice para cada ***valor distinto*** do campo de agrupamento.

Tipos de índices ordenados de único nível

Índice de agrupamento (clustering)



Índice de agrupamento no campo não chave de ordenação **Numero_departamento** de um arquivo **FUNCIONARIO**.

Tipos de índices ordenados de único nível

Índice de agrupamento (clustering): Inserção e exclusão de registro

- A movimentação de registros ainda causa problemas, pois os registros estão fisicamente ordenados.
- Possibilidade:
  - Reservar um bloco inteiro (ou um cluster de blocos contínuos) para ***cada valor*** do campo de agrupamento;
  - todos os registros com esse valor são colocados no bloco (ou cluster de bloco).

Tipos de índices ordenados de único nível

Índice de agrupamento (clustering)

- Um índice de agrupamento é um índice ***não denso***, pois ele tem uma entrada para cada ***valor distinto*** do campo de índice.

Tipos de índices ordenados de único nível

Índice secundário

- Especificado em qualquer campo *não* ordenado de um arquivo.
- Um arquivo de dados pode ter vários índices secundários além de seu método de acesso primário.

Tipos de índices ordenados de único nível

Índices secundários

- Oferecem um meio secundário para acessar um arquivo de dados para o qual algum acesso primário já existe.
- Registros do arquivo podem ser ordenados, desordenados ou hashed.
- Consiste em um arquivo ordenado com dois campos:
  - o primeiro campo é do mesmo tipo de algum *campo não ordenado* do arquivo que seja um **campo de índice**.
  - o segundo campo é um ponteiro de *bloco* ou um ponteiro de *registro*.

Tipos de índices ordenados de único nível

Índices secundários

- O índice secundário é criado em um campo que é uma chave candidata e tem um valor único em cada registro;
- Ou em um campo não chave com valores duplicados.
- *Muitos* índices secundários (e, portanto, campos de indexação) podem ser criados para o mesmo arquivo.
- Cada um representa um meio adicional de acessar esse arquivo com base em algum campo específico.

Tipos de índices ordenados de único nível

Índice secundário em um campo chave (único)

- Esse campo que tem um *valor distinto* para cada registro, é chamado de chave **secundária**.
- Correspondente ao atributo de chave UNIQUE (Modelo Relacional) ou ao atributo de chave primária de uma tabela.
- Existe uma entrada de índice para *cada registro* no arquivo que contém:
  - o valor do campo para o registro;
  - e um ponteiro para o bloco em que o registro está armazenado ou para o próprio registro (índice **denso**).

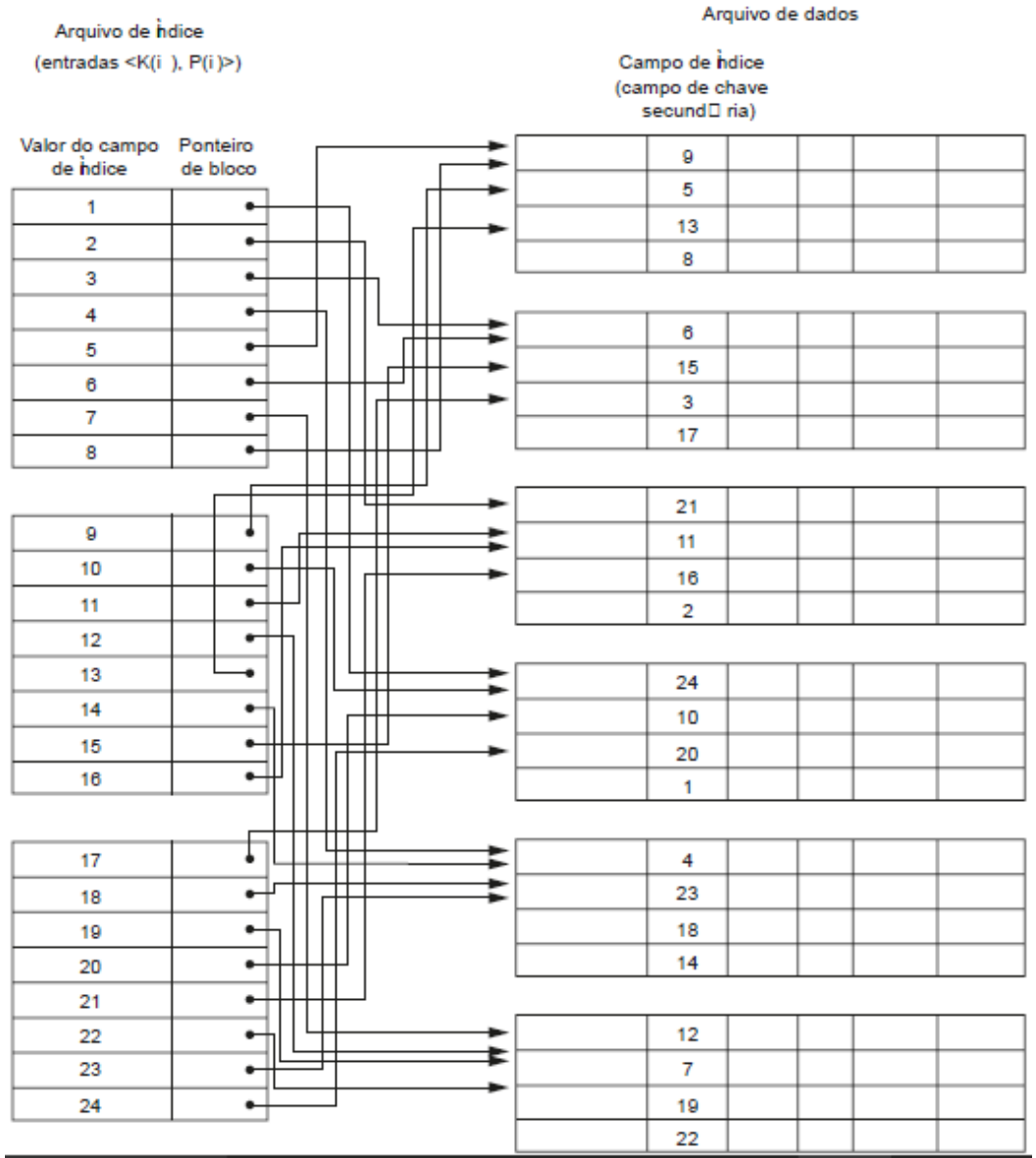
Tipos de índices ordenados de único nível

Índice secundário em um campo chave (único)

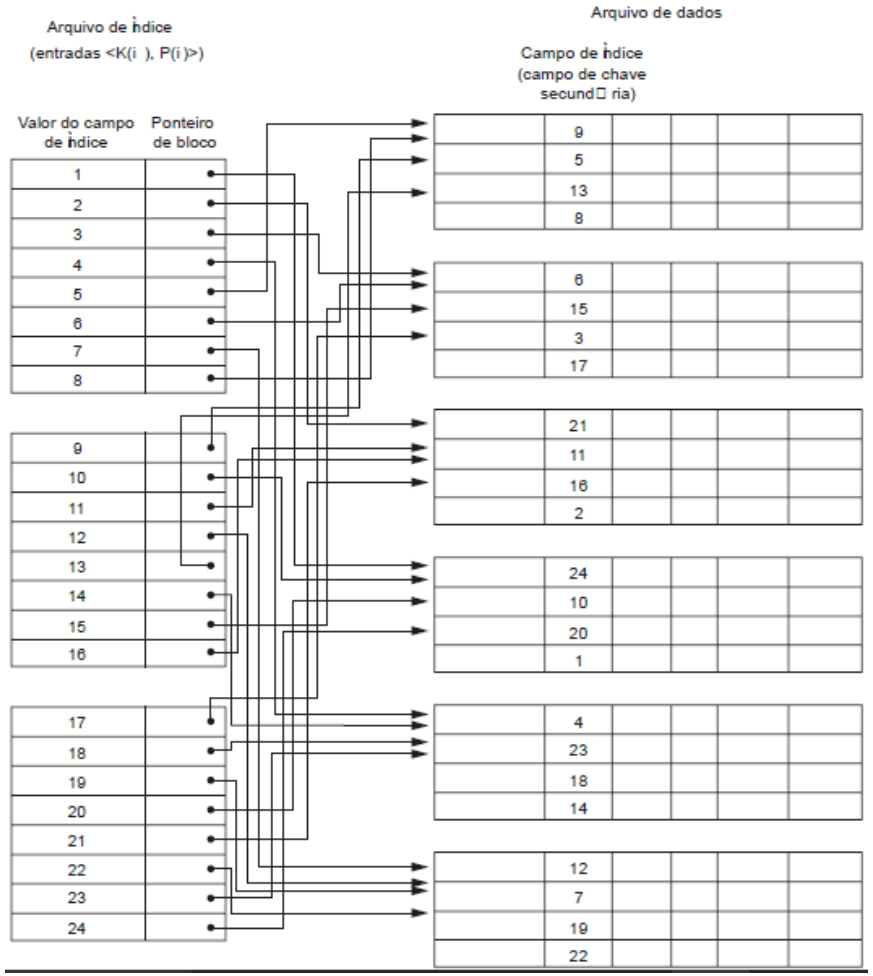
- Valores de campo da entrada de índice *i*: *<K(i), P(i)>*
- As entradas são **ordenadas** pelo valor de *K(i)* (é possível realizar uma pesquisa binária).
- Como os registros do arquivo *não são* fisicamente ordenados pelos valores do campo de chave secundário, *não podemos* usar âncoras de bloco.

Tipos de índices ordenados de único nível

Índice secundário em um campo chave (único)



Tipos de índices ordenados de único nível



Quando o bloco de disco apropriado é transferido para um buffer da memória principal, uma pesquisa pelo registro desejado no bloco pode ser executada.

Tipos de índices ordenados de único nível

Índices secundários

- Índice secundário em um campo chave (único)
- Índice secundário, em geral, precisa de mais espaço de armazenamento e tempo de busca maior do que um índice primário, devido a seu maior número de entradas.
- Porém, a *melhoria* no tempo de pesquisa para um registro é muito maior para um índice secundário.
- Visto que teríamos de fazer uma *pesquisa linear* no arquivo.

Tipos de índices ordenados de único nível

Índices secundários

Índice secundário em um campo não chave (não ordenado)

- Nesse caso, diversos registros no arquivo podem ter o mesmo valor para o campo de índice. Opções para implementar tal índice:

OPÇÃO 1

- Incluir entradas de índice duplicadas com o mesmo K(i) - um para cada registro (índice denso).

Tipos de índices ordenados de único nível

Índices secundários

Índice secundário em um campo não chave (não ordenado)

OPÇÃO 2

- Registros de tamanho variável para as entradas de índice, com um campo repetitivo para o ponteiro.
- Mantemos um ponteiro para cada bloco que contém um registro cujo valor do campo de índice é igual a $K(i)$.

Na opção 1 ou 2, o algoritmo de pesquisa binária deve ser modificado para considerar um número variável de entradas de índice por valor de chave de índice.

Tipos de índices ordenados de único nível

Índices secundários

Índice secundário em um campo não chave (não ordenado)

OPÇÃO 3

- Esquema não denso e mais utilizado.
- Mantem as próprias entradas de índice em um tamanho fixo e tem uma única entrada para cada *valor de campo de índice*.
- Mas cria *um nível de indireção extra* para lidar com os múltiplos ponteiros.

Tipos de índices ordenados de único nível

Índices secundários

Índice secundário em um campo não chave (não ordenado)

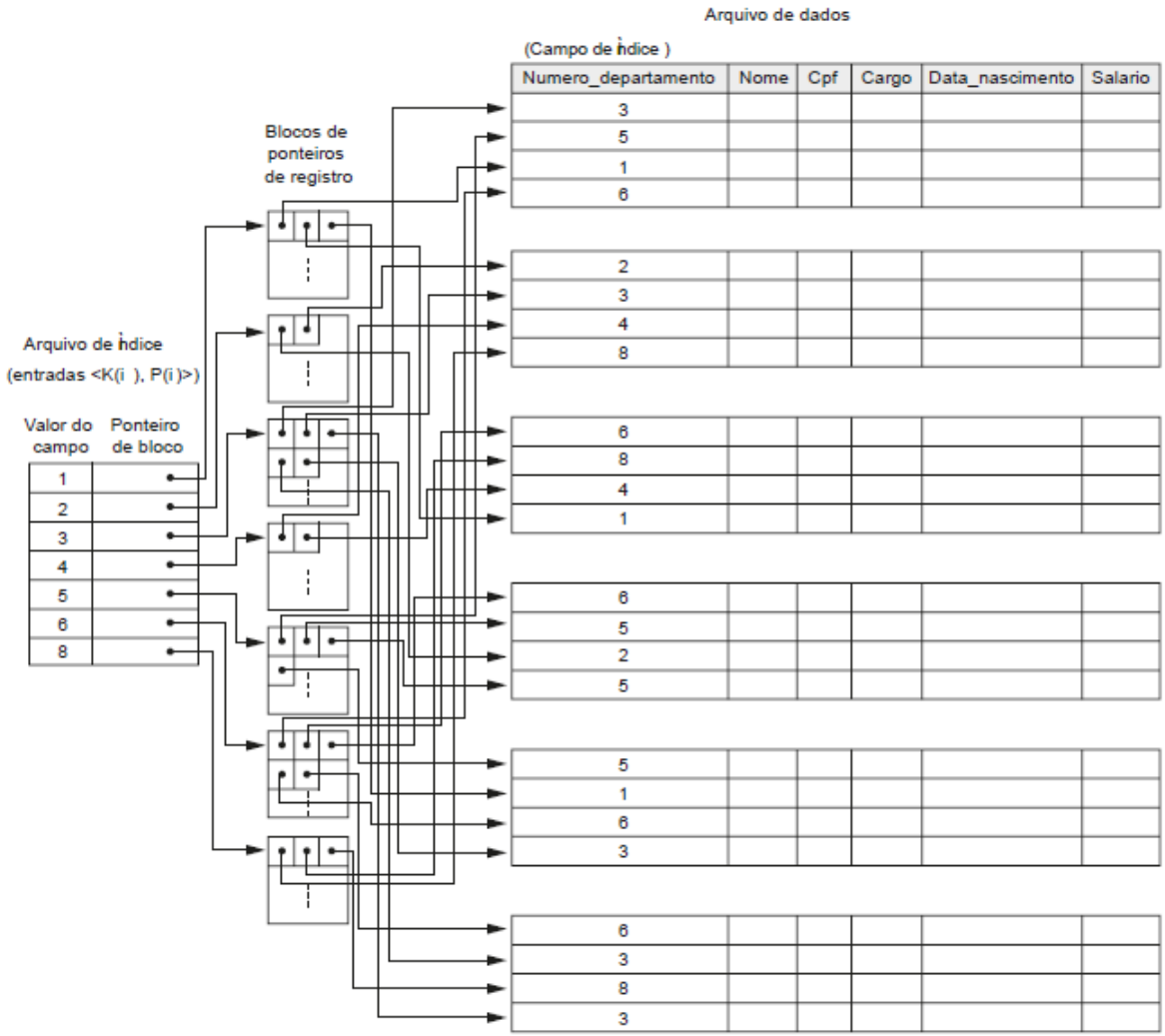
OPÇÃO 3

- O ponteiro *P(i)* na entrada de índice <*K(i)*, *P(i)*> aponta para um bloco de disco, que contém um *conjunto de ponteiros de registro*.
- Cada ponteiro de registro aponta para um dos registros de arquivo com valor *K(i)* para o campo de índice.
- Se algum valor *K(i)* ocorrer em muitos registros, um cluster ou lista ligada de blocos é utilizada.

Tipos de índices ordenados de único nível

Índices secundários

Índice secundário em um campo não chave (não ordenado)



Tipos de índices ordenados de único nível

Índices secundários

Índice secundário em um campo não chave (não ordenado)

OPÇÃO 3

- A recuperação por meio do índice requer um ou mais acessos de bloco adicionais.
- Os algoritmos para pesquisar o índice e para inserir novos registros no arquivo são simples.
- Recuperações em condições de seleção complexas podem ser tratadas referindo-se aos ponteiros de registro.

Tipos de índices ordenados de único nível

Índices secundários

Índice secundário em um campo não chave (não ordenado)

- Um índice secundário oferece uma **ordenação lógica** nos registros pelo campo de índice.
- Os índices primário e de agrupamento assumem que o campo utilizado para a **ordenação física** dos registros no arquivo é o mesmo que o campo de índice.

Tipos de índices ordenados de único nível

Tipos de índices baseados nas propriedades do campo de índice:

	Campo de índice p/ ordenação física do arquivo	Campo de índice não usado p/ ordenação física do arquivo
Campo de índice é chave	Índice primário	Índice secundário (chave)
Campo de índice é não chave	Índice de agrupamento	Índice secundário (não chave)

Tipos de índices ordenados de único nível

Índices baseados nas propriedades do campo de índice:

Tipo de índice	Número de entradas de índice (primeiro nível)	Denso ou não denso (esparso)	Ancoragem de bloco no arquivo de dados
Primário	Número de blocos no arquivo de dados	Não denso	Sim
Agrupamento	Número de valores de campo de índice distintos	Não denso	Sim/não
Secundário (chave)	Número de registros no arquivo de dados	Denso	Não
Secundário (não chave)	Número de registros ou de valores de campo de índice distintos	Denso ou não denso	Não

ESTRUTURAS DE INDEXAÇÃO PARA ARQUIVOS:

Índices multiníveis

Índices multiníveis

- Implementam uma extensão da ideia de pesquisa binária com uma técnica mais eficiente.
- O objetivo é reduzir mais rapidamente o espaço de pesquisa.
- Para isso, se reduz a parte do índice que continuamos a pesquisar por fator de bloco para o índice (bfr_i), que é maior que 2.

Índices multiníveis

- O valor bfr_i é chamado de **fan-out** do índice multinível (*fo*).
- O *espaço de pesquisa de registro* é dividido *n* vezes (onde *n* = o fan-out) a cada passo de pesquisa, usando o índice multinível.
- A pesquisa em um índice multinível requer logfobi acessos de bloco.

Índices multiníveis

Primeiro nível (ou base) do índice multinível

- Arquivo de índice como um ***arquivo ordenado*** com um ***valor distinto*** para cada *K(i)*.
- Ou seja, consiste em um arquivo de dados classificado.

Índices multiníveis

Segundo nível do índice multinível

- Índice primário para o primeiro nível.
- Pode-se usar âncoras de bloco de modo que o segundo nível tenha uma entrada para ***cada bloco*** do primeiro nível.
- Todas as entradas de índice são do mesmo tamanho (valor de campo e um endereço de bloco).
- O fator de bloco bfr_i para o segundo nível (e para os subsequentes) é o mesmo daquele para o índice do primeiro nível.

Índices multiníveis

Segundo nível do índice multinível

- Se o primeiro nível tiver r_1 entradas e o fator de bloco (fan-out) para o índice for $bfr_i = fo$;
- então o primeiro nível precisará de $[(r_1/fo)]$ blocos.
- Portanto, o número de entradas r_2 necessárias no segundo nível do índice.

Terceiro nível do índice multinível

- Índice primário para o segundo nível.
- Possui uma entrada para cada bloco do segundo nível, de modo que o número de entradas do terceiro nível é $r_3 = [(r_2/fo)]$.

Índices multiníveis

- Exige-se um segundo nível se o primeiro nível precisar de mais de um bloco de armazenamento de disco.
- Exige-se um terceiro nível somente se o segundo nível precisar de mais de um bloco.

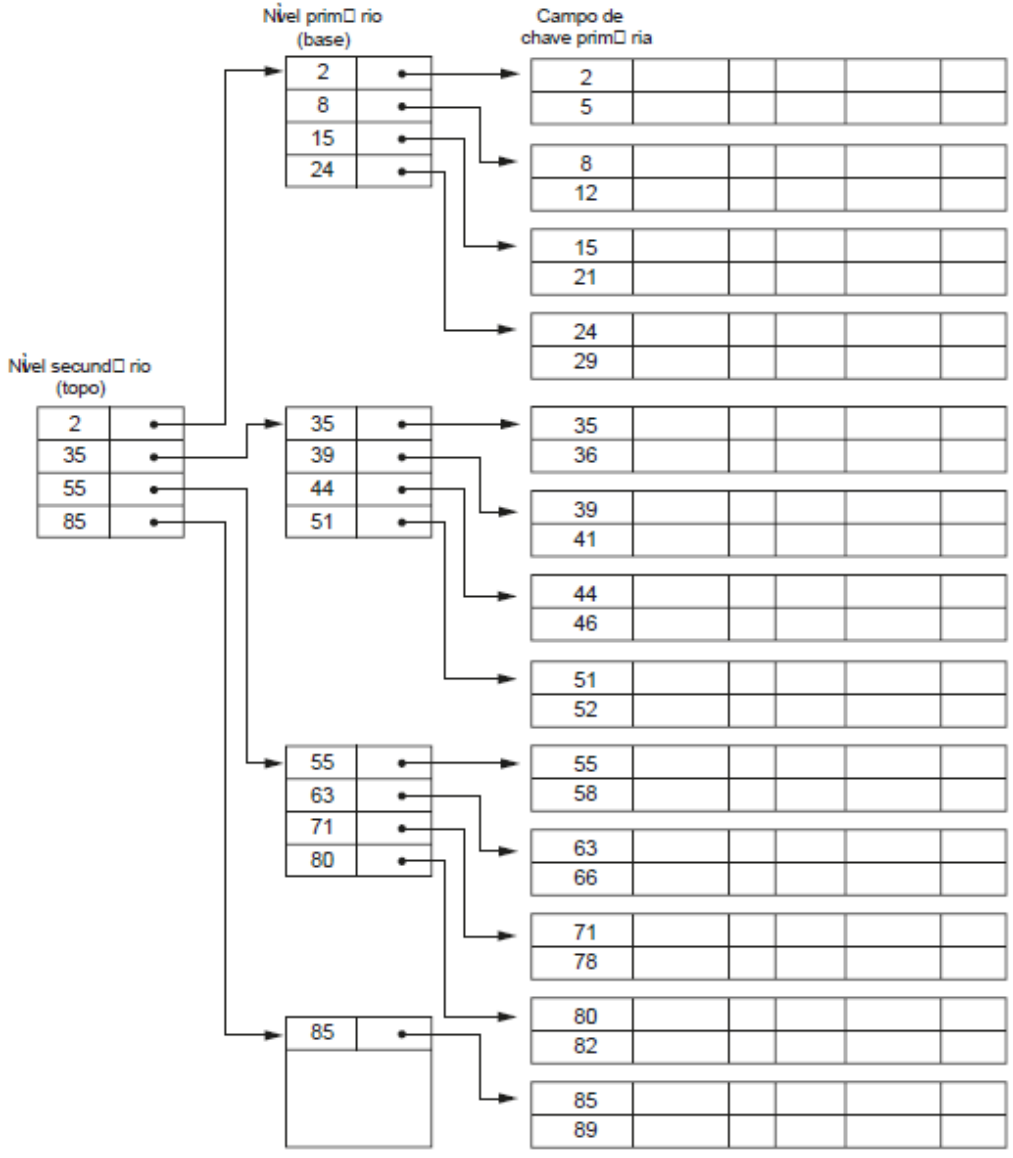
Índices multiníveis

- Bloco no t-ésimo nível.
- Cada nível reduz o número de entradas nos níveis anteriores por um fator de *fo*.
- Pode-se usar a fórmula 1 <= (r_1/((fo)t)) para calcular *t*.
- Portanto, um índice multinível com r_1 entradas de primeiro nível terá aproximadamente *t* níveis, onde $t = [(\log_{fo}(r_1))]$.

Índices multiníveis

- Ao pesquisar o índice, um único bloco de disco é recuperado em cada nível.
- Logo, t blocos de disco são acessados para uma pesquisa de índice, onde i é o número de níveis de índice.
- Desde que o índice de primeiro nível tenha ***valores distintos para K(i) e entradas de tamanho fixo***, o esquema multinível pode ser usado em qualquer tipo de índice (primário, de agrupamento, ou secundário).

Índices multiníveis



Índices multiníveis

- Pode-se também ter um índice primário multinível não denso.
- Nesse caso, *temos* de acessar o bloco de dados do arquivo antes de podermos determinar se o registro sendo pesquisado está no arquivo.
- Para um índice denso, isso pode ser determinado acessando o primeiro nível de índice (sem ter de acessar um bloco de dados).
- Já que existe uma entrada índice para *cada* registro no arquivo.

Arquivo sequencial indexado

- Organização de arquivo comum usada no processamento de dados comercial.
- É um arquivo ordenado com um índice primário multinível em seu campo de chave de ordenação.

Índices multiníveis

- Um índice multinível reduz o número de blocos acessados quando se pesquisa um registro, dado seu valor de campo de indexação.
- Como todos os níveis de índice são *arquivos ordenados* fisicamente ainda ocorre problemas em inserções e exclusões de índice.

Índices multiníveis

Índice multinível dinâmico

- Deixa espaço em cada um de seus blocos para inserir novas entradas.
- Usa algoritmos apropriados de inserção/exclusão para criar e excluir novos blocos de índice.
- Normalmente implementado ao usar estruturas chamadas B-trees e B+-trees.

Índices multiníveis dinâmicos usando B-trees e B+-trees

Árvore

- Uma **árvore** é formada de **nós**.
- Cada nó (exceto pela **raiz**) tem um nó **pai**.
- O nó raiz não tem pai.
- Cada nó tem zero ou mais nós **filhos**.
- Um nó que não possui filhos é denominado **folha**.
- Um nó não folha é chamado de nó **interno**.
- O **nível** de um nó é sempre um a mais que o nível de seu pai (nível do nó raiz é *zero*).

ESTRUTURAS DE INDEXAÇÃO PARA ARQUIVOS:

Índices de multiníveis dinâmicos usando B-trees e B^+-trees

Subárvore

- Uma subárvore de um nó consiste nesse nó e todos os seus nós **descendentes**.
- Definição recursiva: Uma subárvore consiste em um nó n e as subárvores de todos os nós filhos de n.

Árvores de pesquisa

- Tipo especial de árvore utilizada para orientar a pesquisa por um registro, a partir do valor de um dos seus campos.
- Ligeiramente diferente de um índice multinível.

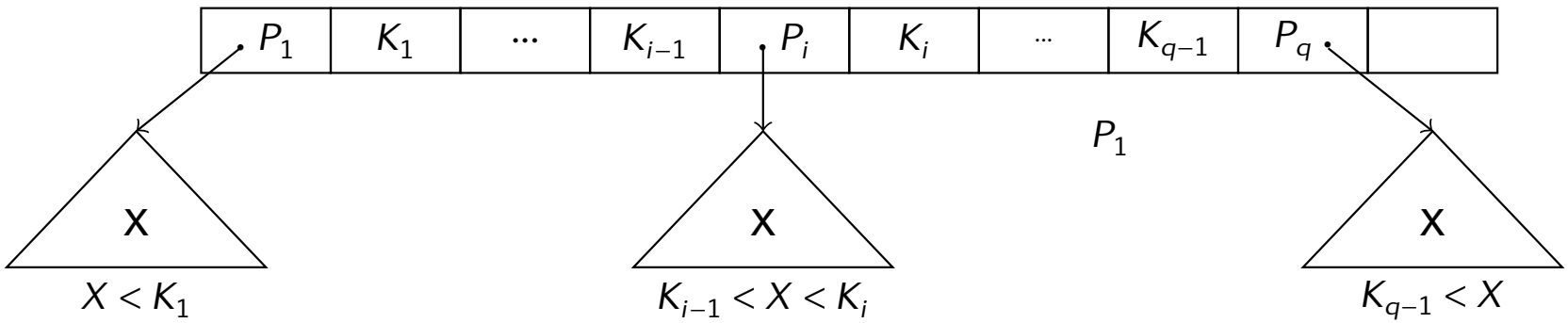
Árvores de pesquisa

- Em uma **árvore de pesquisa de ordem *p*** cada nó contém no ***máximo p - 1 valores*** de pesquisa;
- E ***p*** ponteiros na ordem <P_1, K_1, P_2, K_2, ..., P_{q-1}, K_{q-1}, P>, onde q <= p.
- Cada P_i é um ponteiro para um nó filho (ou um ponteiro NULL);
- Cada K_i é um valor de pesquisa de algum conjunto ordenado de valores.

Árvores de pesquisa

Restrições:

- Em cada nó, $K_1 < K_2 < ... < K_{q-1}$



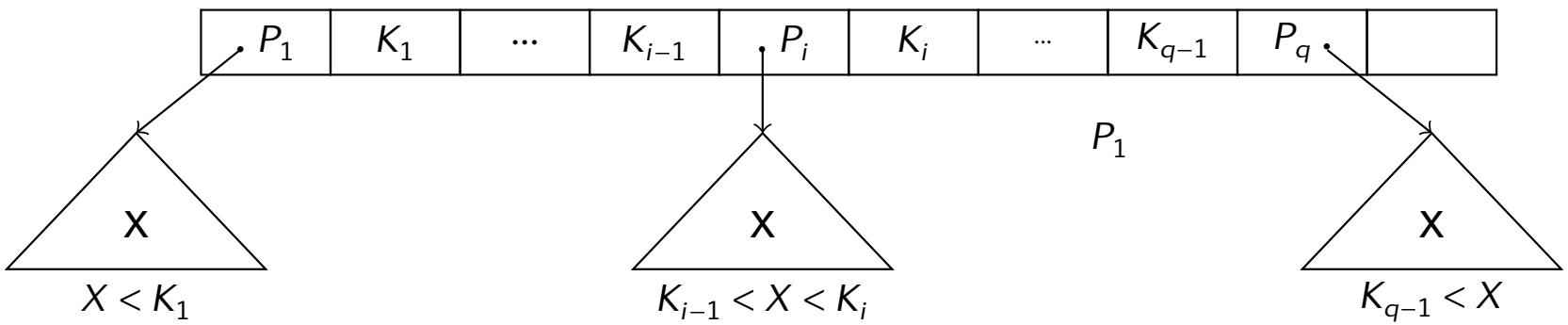
Um nó em um árvore de pesquisa com ponteiros para subárvores abaixo dela.

Árvores de pesquisa

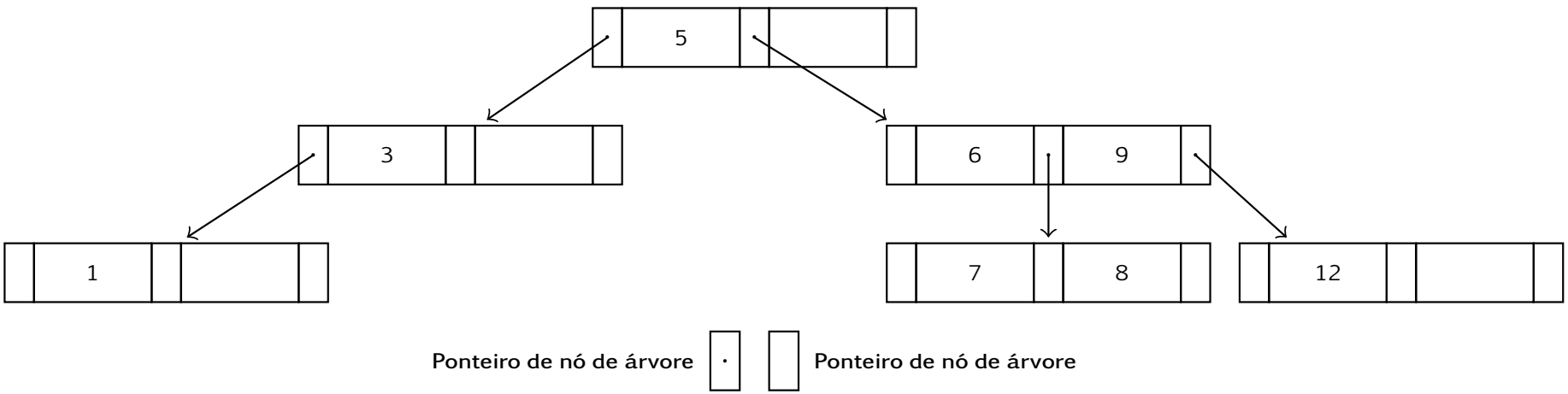
Restrições:

- Para os valores X na subárvore, temos $K_{i-1} < X < K_i$ para $1 < i < q$;
- $X < K_i$ para $i = 1$;
- e $K_{i-1} < X$ para $i = q$.

Ao procurar um valor X, segue-se o ponteiro P_i apropriado, de acordo com as condições acima.



Árvores de pesquisa



Uma árvore de pesquisa de ordem $p = 3$ e valores de pesquisa inteiros (alguns dos ponteiros P_i em nós podem ser NULL).

Árvores de pesquisa

- Podemos usar uma árvore de pesquisa para procurar registros armazenados em um arquivo de disco.
- Os valores na árvore podem ser os valores de um dos **campos de pesquisa** do arquivo.
- Cada valor de chave na árvore é associado a um ponteiro para o registro no arquivo (ou bloco de disco) que tem esse valor.
- A própria árvore de pesquisa pode ser armazenada no disco ao atribuir cada nó a um bloco de disco.

Árvores de pesquisa

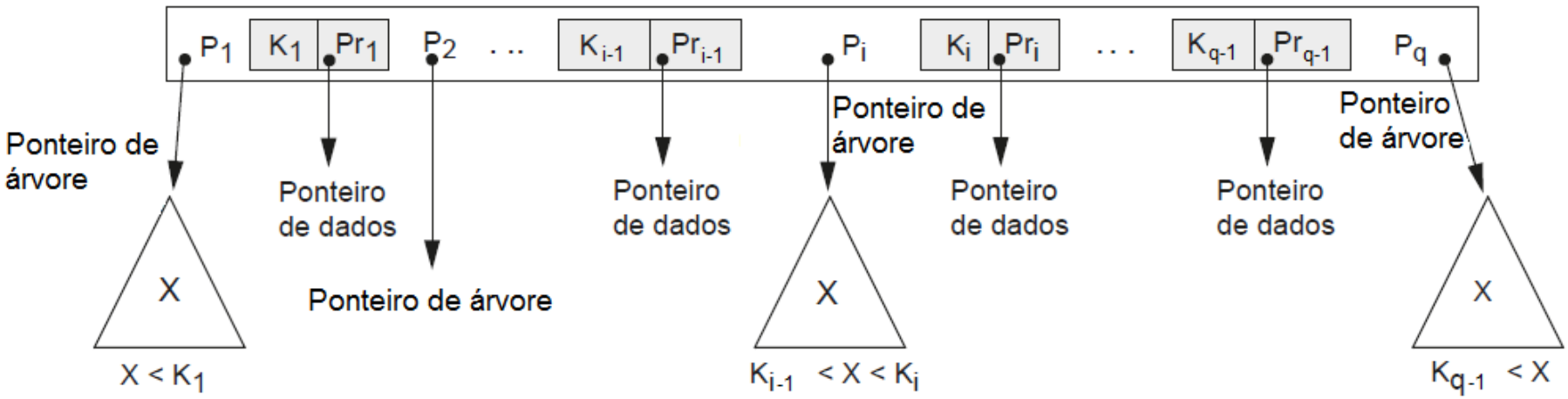
- São necessários algoritmos para inserir e excluir valores de pesquisa na árvore (para se manter as restrições).
- Em geral, esses algoritmos não garantem que uma árvore de pesquisa seja **balanceada**.
- Os objetivos para balancear uma árvore de pesquisa são:
  - Garantir que os nós sejam distribuídos por igual, de modo que a profundidade da árvore seja minimizada e que a árvore não fique distorcida.
  - Tornar a velocidade de pesquisa uniforme.

Índices de multiníveis dinâmicos

B-trees

- Restrições adicionais:
  - A árvore deve estar sempre balanceada;
  - O espaço desperdiçado pela exclusão nunca é excessivo.
- Estrutura de acesso multinível;
  - Estrutura de árvore balanceada em que cada nó está cheio pelo menos até a metade.
- Estrutura básica para B-tree de ordem **p** (estrutura de acesso de um campo chave).

B-trees


P1
K1
Pr1
P2
. . .
Ki-1
Pri-1
Pi
Ki
Pri
. . .
Kq-1
Prq-1
Pq
Ponteiro de árvore
Ponteiro de dados
Ponteiro de árvore
Ponteiro de dados
Ponteiro de árvore
Ponteiro de dados
Ponteiro de árvore
Ponteiro de dados
Ponteiro de árvore
X
X
X
X < K1
Ki-1 < X < Ki
Kq-1 < X

B-trees

- Cada nó interno tem a forma:

  $<P_1, <K_1, Pr_1>, P_2, <K_2, Pr_2>, ..., <K_{q-1}, Pr_{q-1}<, P_q>$

  onde:

  - $q <= p$
  - P_i: ponteiro de árvore (para outro nó);
  - Pr_i: ponteiro de dados (para o registro cujo valor do campo chave é igual a K_i);

  - Máximo de p ponteiros;
  - Máximo de $p-1$ ponteiros de dados;
  - Máximo de $p-1$ valores de campo de chave;

  - $K_1 < K_2 < ... < K_{q-1}$

B-trees

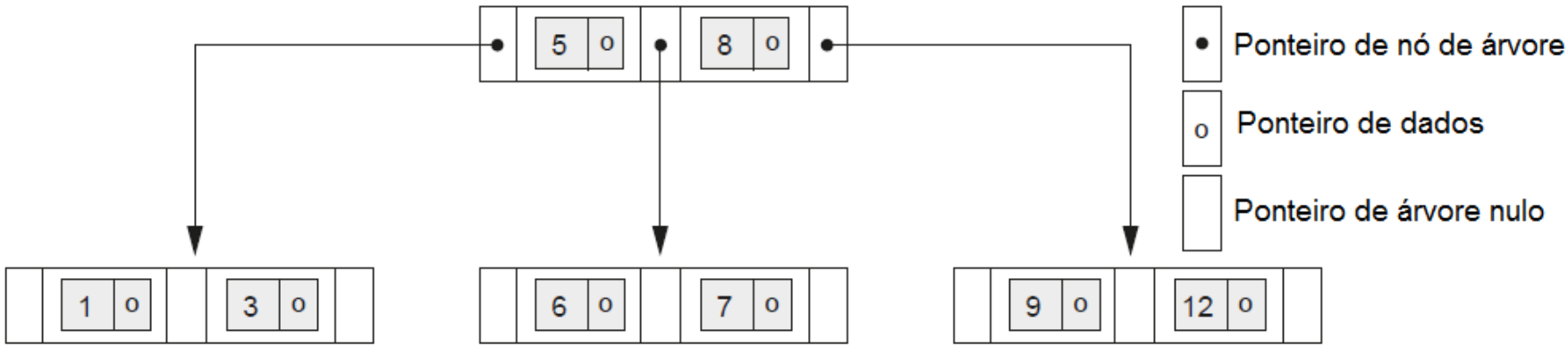
- Para todos os valores de campo da chave de pesquisa X, na subárvore apontada por P_i, temos:

  $K_{i-1} < X < K_i$ para $1 < i < q$
  $X < K_i$ para $i = 1$
  $K_{i-1} < X$ para $i = q$

  onde:

  - Cada nó tem no máximo **p** ponteiros de árvore e, ao menos **p/2** ponteiros de árvore (exceto os nós raiz e folha);
  - Um nó com **q** ponteiros de árvore (sendo **q <= p**) tem **q-1** valores de campo chave de pesquisa.

B-trees - Exemplo



- Observe:
  - p = 3
  - Todos os valores de pesquisa K são únicos.

B-trees

- Procedimentos para inserção:
  1. Começa com um único nó raiz, no nível 0;
  2. SE o nó raiz está cheio com **p-1** valores de chave de pesquisa:
     - Se divide em dois de nível 1;
     - Somente o valor do meio é mantido no nó raíz.
  3. SENÃO:
     - Se divide em dois nós no mesmo nível;
     - A entrada do meio é movida para o nó pai junto com os ponteiros aos divididos;
     - Se o nó pai estiver cheio, este também é dividido;
       A divisão se propaga até o nó raiz.

B-trees

- Se a exclusão de um valor fizer que um nó fique com menos da metade cheio, este é combinado com seus nós vizinhos;
  - Pode reduzir o número de níveis da rede.
- Os nós ficam aproximadamente 69% cheios quando o número de valores na árvore se estabiliza;
  - A divisão e a combinação de nós ocorrerão raramente, de modo que a inserção e a exclusão se tornam muito eficientes.

B-trees - Ex (Cálculo do número de blocos e níveis):

- A pesquisa será por um campo de chave não ordenado, em uma B-tree com **p = 23**. Cada nó está 69% cheio, portanto, cada nó terá, em média, **p * 0,69 = 23 * 0,69**, ou aproximadamente **16 ponteiros**, ou seja, 15 valores de campo chave de pesquisa.
    - Começamos na raíz, verificando quantos ponteiros podem existir, na média, em cada nível subsequente.

B-trees - Ex (Cálculo do número de blocos e níveis):

Raíz	1 nó	15 entradas	16 ponteiros
Nível 1	16 nós	240 entradas	256 ponteiros
Nível 2	256 nós	3.840 entradas	4.096 ponteiros
Nível 3	4.096 nós	61.440 entradas	

Uma B-tree de três níveis mantém 65.536 entradas de chave na média.

Índices de multiníveis dinâmicos

B^+-trees

- Variação da estrutura de dados da B-tree;
- Utilizado na maioria das implementações de um índice multinível dinâmico;
- Os ponteiros de dados são armazenados apenas nos **nós folha** da árvore.

B^+-TREES

- Nós folha:
  - Armazenam todos os ponteiros de dados;
  - Uma entrada para cada valor do campo de pesquisa;
  - SE o campo de pesquisa for chave
    Também contém o ponteiro de dados para o registro.
    SENÃO
    Também contém o ponteiro que aponta para o bloco com os ponteiros para os registros do arquivo de dados.
  - São ligados para oferecer acesso ordenado no campo de pesquisa;
  - Alguns valores do campo de pesquisa são repetidos para os nós internos para guiar a pesquisa.

B^+-trees

- Nós folha - forma:

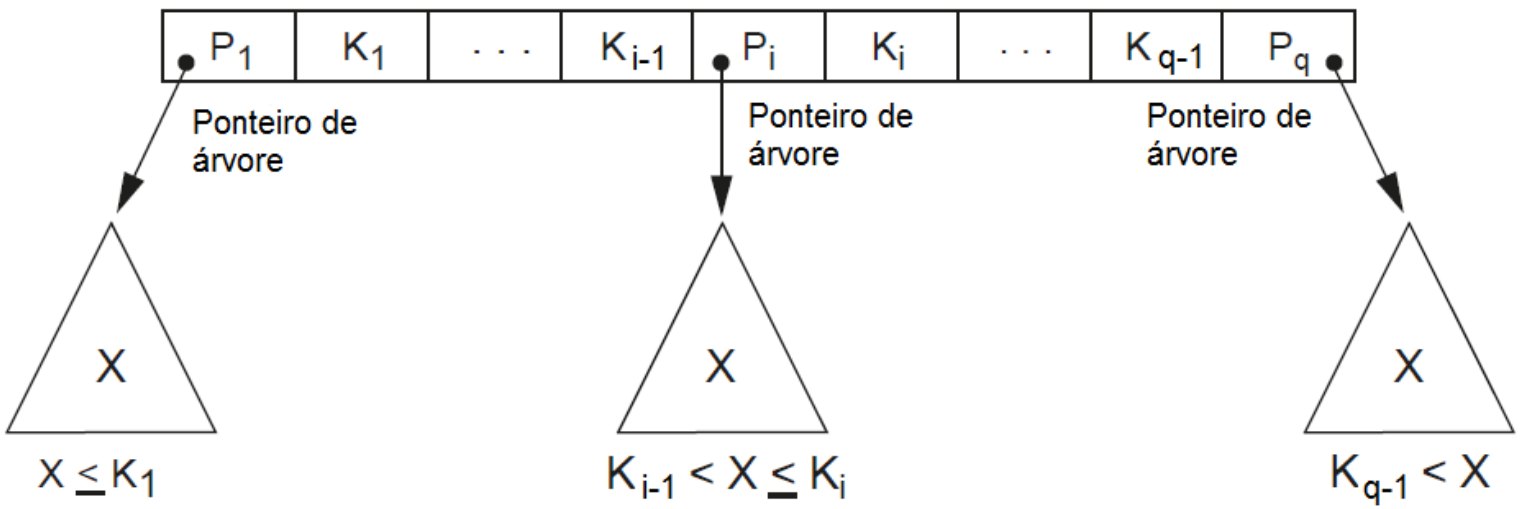
  $< <K_1, Pr_1>, <K_1, Pr_2>, ..., <K_{q-1}, Pr_{q-1}>, P_{proximo}>$

  onde $q <= p$, cada Pr_i é um ponteiro de dados e $P_{proximo}$ aponta para o próximo nó folha da B^+-tree.

  - $K_1 <= K_2 ... , K_{q-1}$ em que $q <= p$;
  - **Pri**: ponteiro de dados - aponta para o registro cujo valor do campo pesquisa é K_i ou para um bloco de arquivo que contém registro;
  - Tem, ao menos, $p/2$ valores;
  - Todos os nós folha estão no mesmo nível.

B$^+$-TREES

- Nós internos:
  - Correspondem aos outros níveis de índice multinível.
  - Estrutura básica para B$^+$-tree de ordem **p**:



B^+-TREES

- Nós internos - forma:

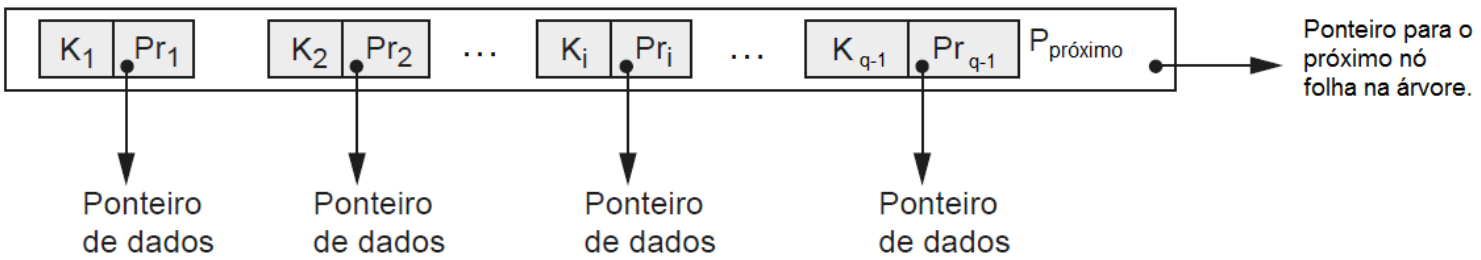
$$<P_1, K_1, P_2, K_2, ..., P_{q-1}, K_{q-2}, P_q>$$

- onde $q<=p$ e cada P_i é um ponteiro de árvore.
  - $K1 < K2 < ... < Kq-1$.
  - Para todos os valores de campo de pesquisa X na subárvore apontada por Pi, temos:

    $K_{i-1} < X <= K$, para $1 < i < q$
    $X < K_i$ para $i = 1$
    $K_{i-1} < X$ para $i = q$

B$^+$-TREES

- Nós internos - forma:
  - Tem, no máximo, **p** ponteiros de árvore;
  - Exceto a raíz, tem pelo menos **p/2** ponteiros de árvore;
  - Um nó interno com **q** ponteiros (**q <= p**) tem **q - 1** valores de campo de pesquisa.
- Observe:



B^+-TREES

- Ao começar a leitura no nó folha mais Ãă esquerda, é possível atravessar os nós folha como uma lista, usando os ponteiros $P_{proximo}$;
  - Acesso ordenado aos registros de dados no campo de índice.
- Pode ser incluído um ponteiro $P_{anterior}$ (adicional);
- Comparações entre **B^+-tree** com **B-tree**:
  - Mais entradas podem ser compactadas em um nó interno de uma B^+-tree do que para uma B-tree semelhante;

B^+-TREES

- Comparações entre **B^+-tree** com **B-tree**:
  - Para o mesmo tamanho de bloco, a ordem **p** será maior para a B^+-tree do que para a B-tree;
  - Proporciona menos níveis a B^+-tree, otimizando seu tempo de pesquisa.
- A seguir, **p** será usado para indicar a ordem dos nós internos e $\mathbf{p_{folha}}$ para indicar a ordem para os nós folha.

B^+-trees - Ex (Cálculo de ordem de B^+-tree):

- Comparações entre B^+-tree com B-tree:
  - O campo chave da pesquisa é **V = 9 bytes**, o tamanho do bloco é **B = 512 bytes**, o ponteiro de registro seja **Pr = 7 bytes** e o ponteiro de bloco seja de **6 bytes**.
  - Um nó interno da B^+-tree pode ter até **p** ponteiros de árvore e **p-1** valores de campo de pesquisa - estes precisam caber em um único bloco.

    Temos:

    (p * P) + ((p-1) * V) <= B
    (P * 6) + ((p-1) * 9) <= 512
    (15 * p) <= 512
    p <= 34

B^+-trees - Ex (Cálculo de ordem de B^+-tree):

- Comparações entre B^+-tree com B-tree:
  - O valor encontrado em **p** é maior do que o valor 23 para a B-tree, resultando em um *fun-out* maior e mais entradas em cada nó interno de uma B^+-tree do que na B-tree correspondente.
  - Os nós folha da B^+-tree terão o mesmo número de valores e ponteiros, exceto que os ponteiros são de dados e de ponteiro próximo.

B^+-trees - Ex (Cálculo de ordem de B^+-tree):

- Comparações entre B^+-tree com B-tree:
  - A ordem p_{folha} pode ser calculada:

$(p_{folha} * (Pr * V)) + P <= B$
$(pfolha * (7+9) + 6 <= 512$
$(16 * p_{folha}) <= 506$
$p_{folha} <= 31$

Conclui-se que cada nó folha poderá manter até $\mathbf{p_{folha} = 31}$ combinações de ponteiro de valor/dados de chave.

B$^+$-TREES

- Para implementar algoritmos de inserção e exclusão, será necessário inserir *informações adicionais* em cada nó:
  - Tipo de nó (interno ou folha);
  - Número de entradas atuais q no nó;
  - Ponteiros para os nós pai e irmão.
- Antes de proceder com os cálculos para $\mathbf{p}$ e $\mathbf{p_{folha}}$, se faz necessário reduzir o tamanho do bloco pela quantidade de espaço necessária para toda a informação.

B^+-trees - Ex (Cálculo número de entradas):

- No cálculo do número aproximado de entradas, considera-se que cada nó esteja 69% cheio. Em média, cada nó interno terá **34 * 0,69** ou, aproximadamente, 23 ponteiros e 22 valores.

- Cada nó folha manterá **0,36 * p_{folha} = 0,69 * 31 = 21** ponteiros de registro de dados.

B^+-trees - Ex (Cálculo número de entradas):

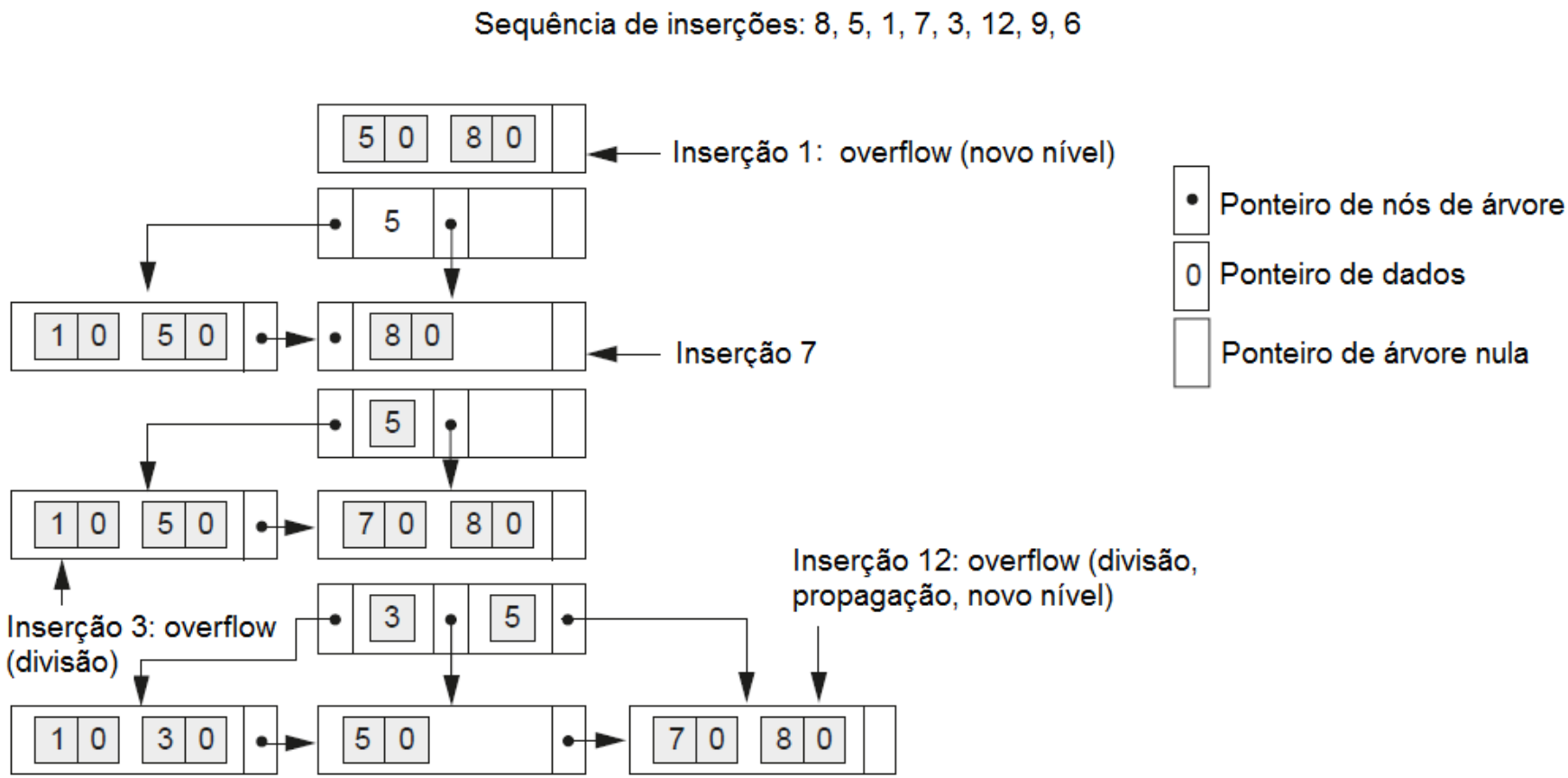
- Número médio de entradas para cada nível:

Raíz	1 nó	22 entradas	23 ponteiros
Nível 1	23 nós	506 entradas	529 ponteiros
Nível 2	529 nós	11.638 entradas	12.167 ponteiros
Nível folha	12.167 nós	255.507 entradas	

- Conclusão:
  - Para o tamanho do bloco, do ponteiro e do campo de pesquisa, uma B^+-tree de três níveis mantém até 255.507 ponteiros de registro, com a média de 69% de ocupação de nós.

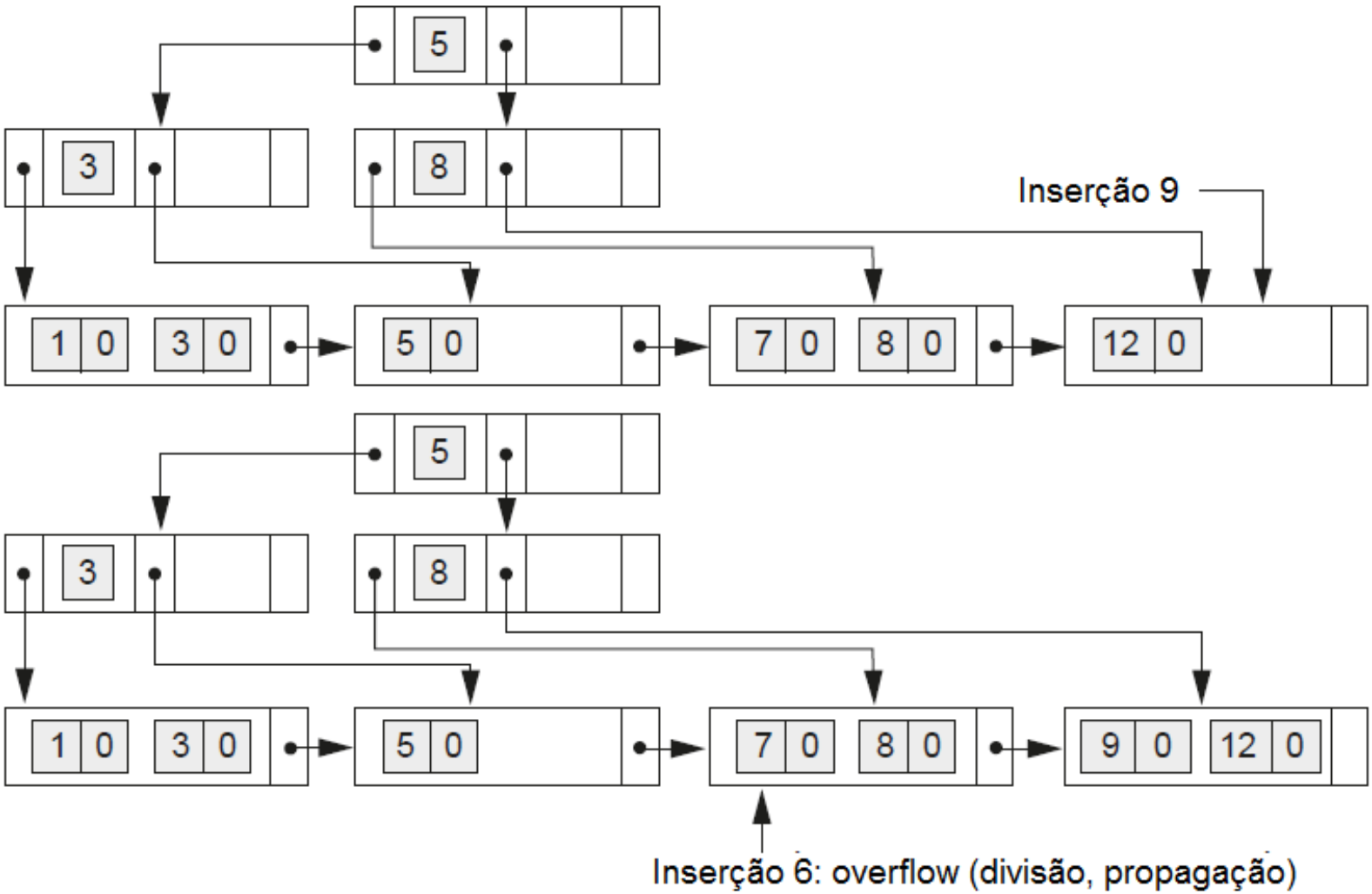
B$^+$-TREES - INSERÇÃO DE REGISTROS:

Ordem p=3 e p_{folha}=2



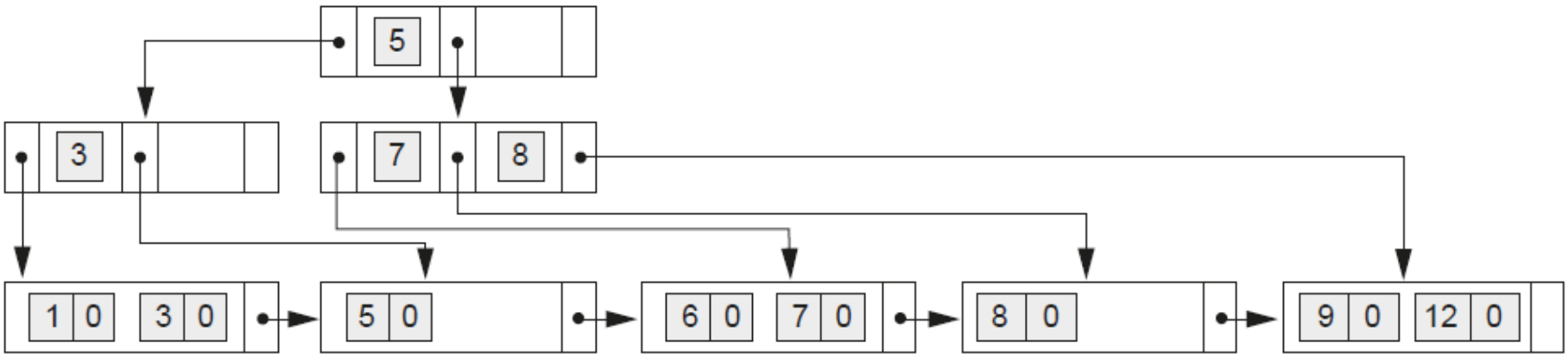
B$^+$-TREES - INSERÇÃO DE REGISTROS:

Ordem p=3 e p_{folha}=2



B$^+$-TREES - INSERÇÃO DE REGISTROS:

Ordem p=3 e p_{folha}=2



B^+-TREES - INSERÇÃO DE REGISTROS:

Ordem $p=3$ e $p_{folha}=2$

- Observe:
  - A raíz é o único nó da árvore que também é um nó folha;
  - Assim que mais de um nível é criado, a árvore é dividida em nós internos e nós folha;
  - Cada valor de chave precisa existir no nível de folha, por conta dos ponteiros;
  - Somente alguns valores existem nos nós internos para guiar a pesquisa;

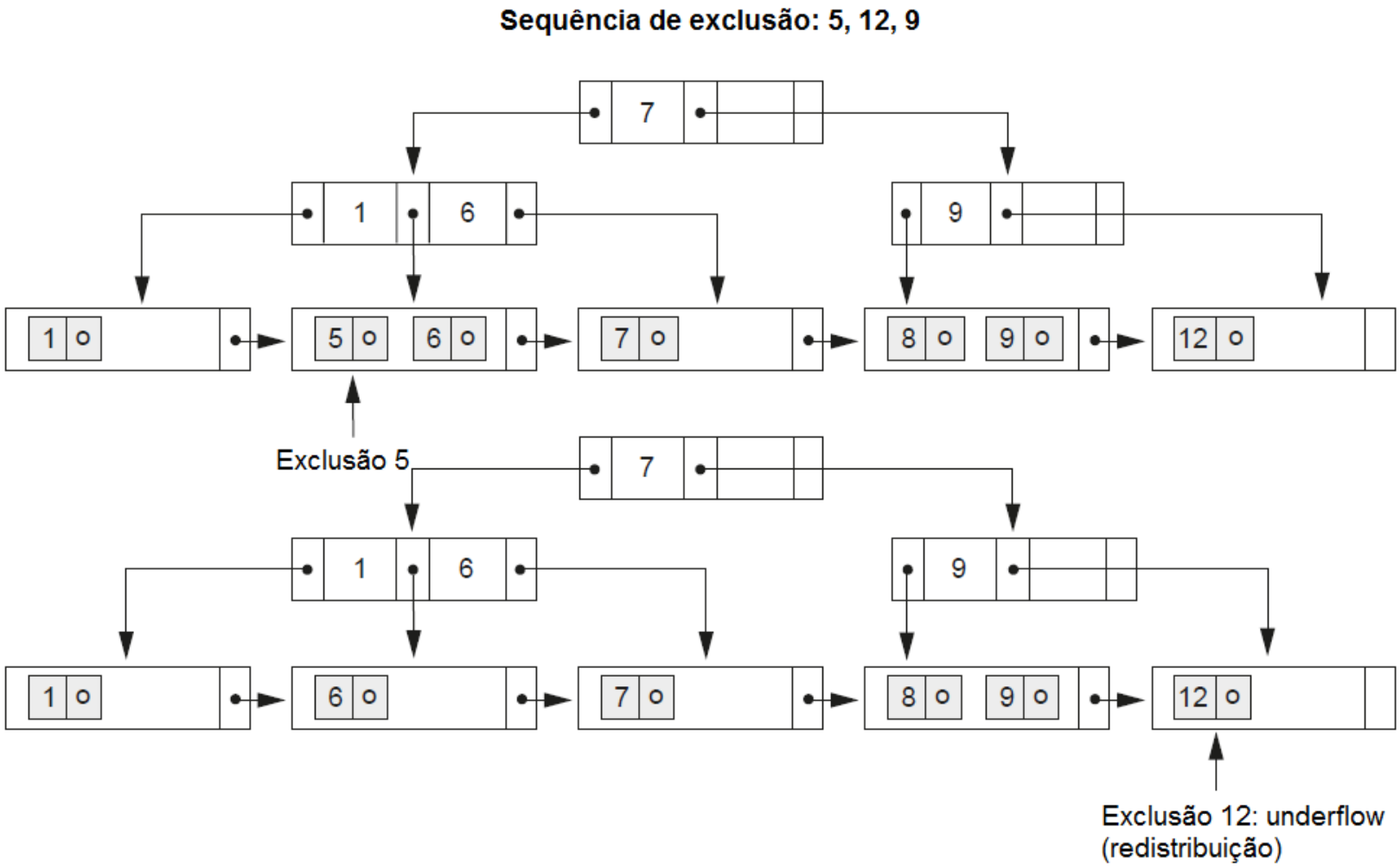
B^+-TREES - INSERÇÃO DE REGISTROS:

Ordem p=3 e p_{folha}=2

- Observe:
  - Necessário dividir o nó folha quando é inserida uma nova entrada e este estiver cheio;
  - As primeiras entradas $i = [((p_{folha} + 1)/2)]$ no nó original são mantidas e as entradas restantes são movidas para um novo nó folha;
  - O *i*-ésimo valor de pesquisa é replicado no nó interno do pai, e um ponteiro para este é criado;
  - Um novo nó interno manterá as entradas de P_{i+1} para o final das entradas no nó.

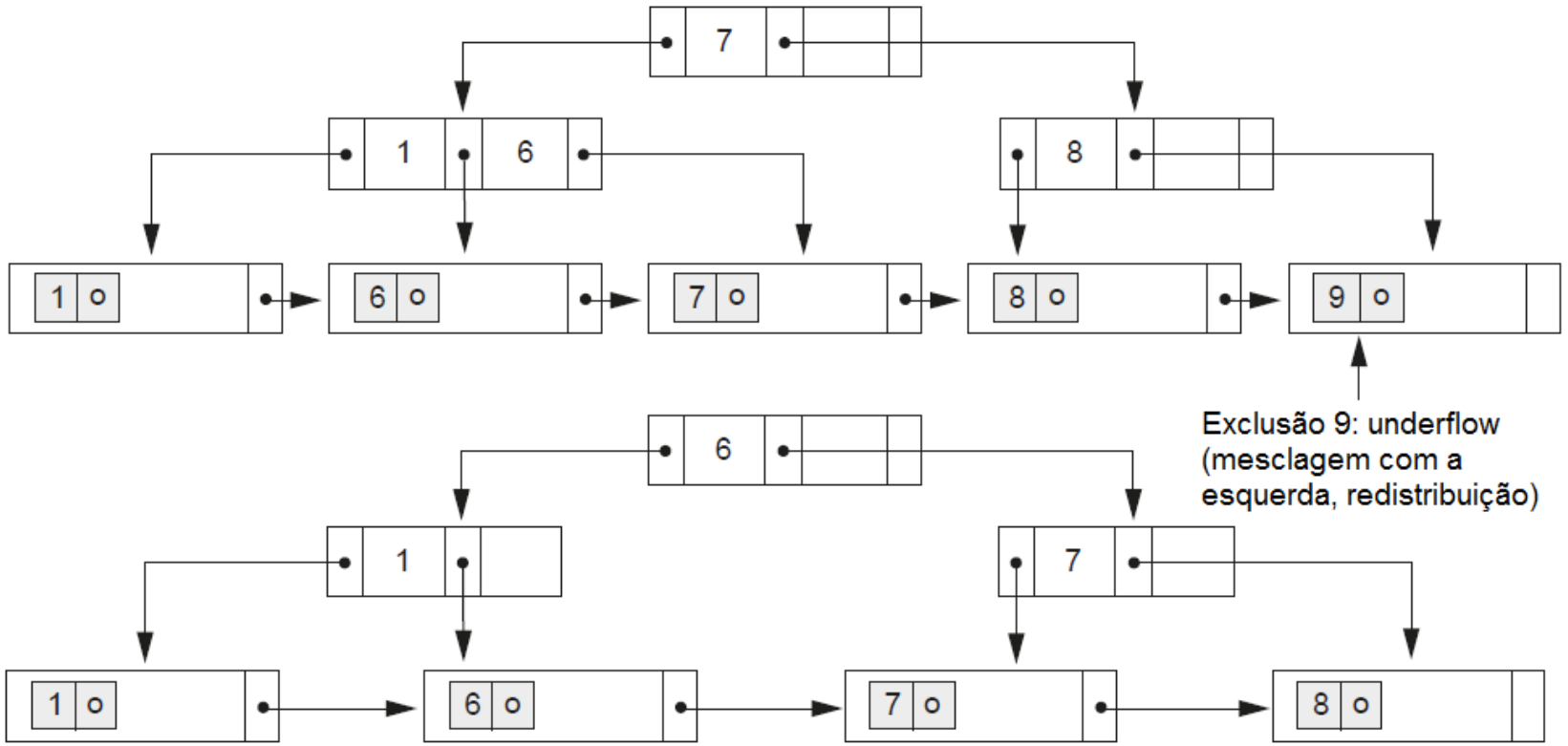
B$^+$-TREES - EXCLUSÃO DE REGISTROS:

Ordem p=3 e p_{folha}=2



B$^+$-TREES - EXCLUSÃO DE REGISTROS:

Ordem p=3 e p_{folha}=2



B^+-TREES - EXCLUSÃO DE REGISTROS:

Ordem p=3 e p_{folha}=2

- Observe:
  - Quando uma entrada é excluída, ele é sempre removida do nível folha;
  - A exclusão pode gerar *overflow*, reduzindo o número de entradas no nó folha abaixo do mínimo exigido;
  - Tenta-se encontrar um nó folha "irmão"e redistribuir suas entradas, os deixando cheios até a metade;
  - Caso contrário, os nós são mesclados é o número de folhas reduzido.

ESTRUTURAS DE INDEXAÇÃO PARA ARQUIVOS:

ÍNDICES EM CHAVES MÚLTIPLAS

Índices em chaves múltiplas

- Solicitações de atualização e recuperação frequente com a mesma combinação de atributos.
  - Configura-se uma estrutura de acesso para oferecer acesso eficaz.
- Exemplo:

FUNCIONARIO

Cpf	Pnome	Minicial	Unome	Salario	Dnr	Idade

Índices em chaves múltiplas

- Considere: Listar os funcionários no **departamento número 4**, cuja **idade é 59**.
  - **Dnr** nem **Idade** são atributos não chave, ou seja, poderá apontar para vários registros.
  - Estratégias:
    1. Se **Dnr** for indexado, mas **Idade** não: acessar os registros **Dnr = 4**, usando o índice, e depois selecionar os registros com **Idade = 59**;
    2. Se **Idade** for indexada, mas **Dnr** não: acessar os registros **Idade = 59**, usando o índice, e depois selecionar os registros com **Dnr = 4**;

Índices em chaves múltiplas

- Considere: Listar os funcionários no **departamento número 4**, cuja **idade é 59**.
  - Estratégias:
    3. Se os índices forem criados sobre **Dnr** e **Idade**: cada registro contém um conjunto de registros ou ponteiros. Utiliza-se uma interseção entre esses conjuntos para obter o resultado.
  - Se o volume de retorno for grande, nenhuma das técnicas será eficiênte.
  - Necessário utilizar as estratégias para tratar a combinação como chave de pesquisa (chaves compostas).

Índices em chaves múltiplas

ÍNDICE ORDENADO EM MÚLTIPLOS ATRIBUTOS

- Se um índice for criado nos atributos $<A_1, A_2, ..., A_n>$, os valores de chave de pesquisa são tuplas com **n** valores: $<v_1, v_2, ..., v_n>$;
- Uma ordenação lexicográfica desses valores estabelece uma ordem na chave de pesquisa composta.

Índices em chaves múltiplas

Hashing particionado

- Extensão do hashing externo estático - permite o acesso a múltiplas chaves;
- Adequado a comparações de igualdade;
- Nenhuma estrutura de acesso separada precisa ser mantida para os atributos individuais;
- Para uma chave que consiste **n** componentes, a função de hash produz um resultado com **n** endereços separados;
- O endereço do bucket é uma concatenação desses **n** endereços.

Hashing particionado - Exemplo

- Chave composta <**Dnr, Idade**>
  - **Dnr**: hash para endereço de 3 bits
  - **Idade**: hash para endereço de 5 bits
  - Bucket de 8 bits (3 bits + 5 bits)

- Se **Dnr = 4** tiver um endereço hash ′100′ e **Idade = 59** tiver um endereço hash ′10101′, na pesquisa dos valores, temos que ir ao endereço:

100 10101

Índices em chaves múltiplas

Arquivos de grade

- É construído um vetor de grade com uma escala (ou dimensão) linear para cada um dos atributos de pesquisa;
- Objetivo de alcançar uma distribuição uniforme dos atributos;
- Cada célula aponta para o endereço de bucket onde os registros correspondentes são armazenados;

Índices em chaves múltiplas

Arquivos de grade

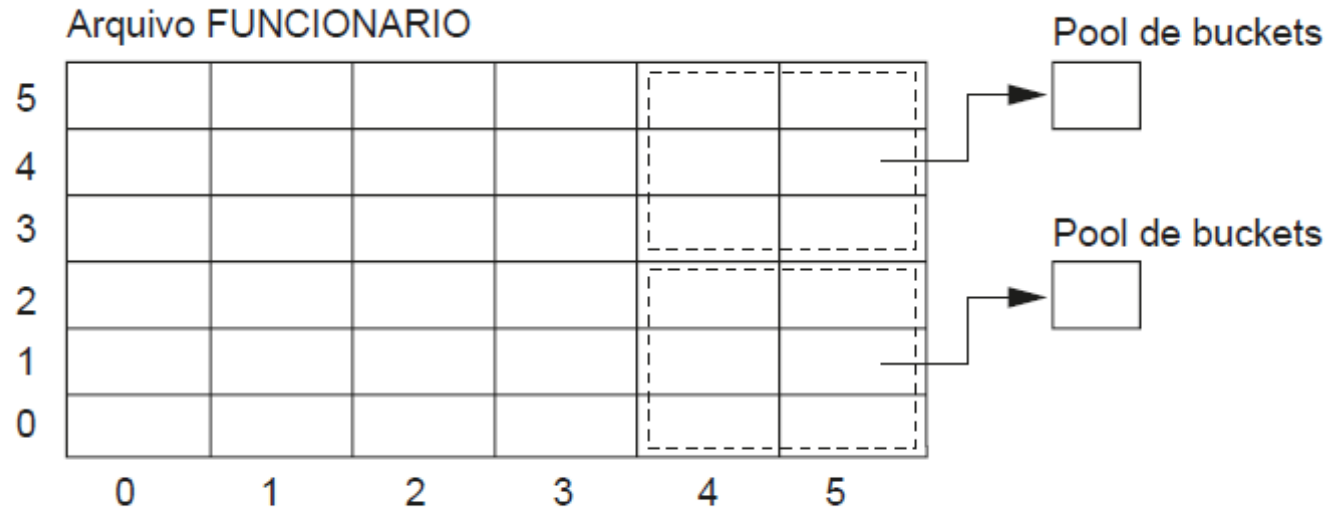
- Adequado a consultas de intervalo (mapeadas para um conjunto de células - grupo de valores ao longo de escalas lineares);
- Aplicado a qualquer quantidade de chaves de pesquisa;
- Apresentam *overhead* de espaço - uma reorganização frequênte do arquivo (aumentando o custo de manutenção).

Arquivos de grade - Exemplo

Dnr

0	1, 2
1	3, 4
2	5
3	6, 7
4	8
5	9, 10

Escala linear para Dnr



Escala linear para Idade

0	1	2	3	4	5
< 20	21-25	26-30	31-40	41-50	> 50

Arquivos de grade - Exemplo

- Vetor de grade para o arquivo FUNCIONARIO com uma escala linear para **Dnr** e outra para o atributo **Idade**;
- A solicitação de **Dnr = 4** e **Idade = 59** é mapeada para a célula **(1,5)**. Os registros para esta combinação serão encontrados no bucket correspondente.

ESTRUTURAS DE INDEXAÇÃO PARA ARQUIVOS:

Outros tipos de índices

Outros tipos de índices

Índices de hash

- Estrutura secundária para acessar o arquivo usando hashing em uma chave de pesquisa diferente da que é usada para a organização do arquivo de dados primário;
- Índice físico;
- Entradas de índice:

  tipo <K, Pr> ou <K, P>

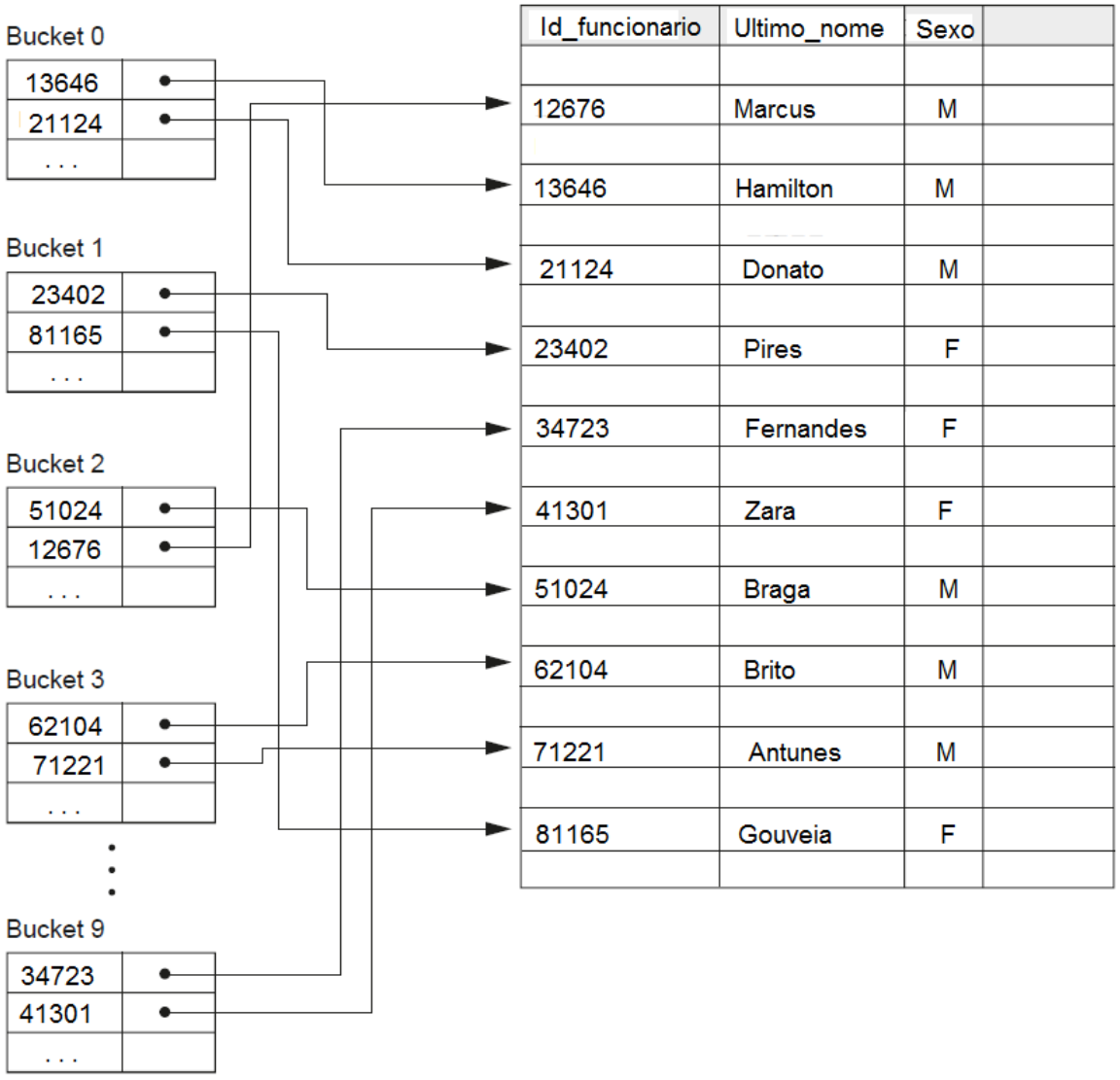
  onde
  - **Pr**: ponteiro para o registro que contém a chave;
  - **P**: ponteiro para o bloco que contém o registro para a chave.

Outros tipos de índices

Índices de hash

- O arquivo de índice pode ser organizado como um arquivo de hash dinamicamente expansível;
- Pesquisa por uma entrada:
  - Algoritmo de pesquisa de hash em K;
  - Quando uma entrada é localizada, o ponteiro Pr é utilizado para localizar o registro no arquivo de dados.

Índices de hash - Exemplo



Outros tipos de índices

Índices de hash - Exemplo

- O **Func_id** passa pelo hashing para obter um número de butcket usando a função de hashing: a soma dos dígitos de **Func_id módulo 10**.
- Encontrando o **Func_id = 51024**:
  1. 5 + 1 + 0 + 2 + 4 = 12
  2. 12 módulo 10 = 2
- Neste caso, o bucket 2 é acessado primeiro. Este contém a entrada de índice **<51024, Pr>**;
- O ponteiro **Pr** nos leva ao registro real no arquivo.

Outros tipos de índices

Índices bitmap

- Estrutura que facilita a consulta em múltiplas chaves;
- Utilizada em relações que contém um grande número de linhas;
- Cria um índice para uma ou mais colunas e cada valor (ou intervalo de valores) é indexado;
- Eficiêntes em relação ao espaço de armazenamento;
- Para criação do índice em um conjunto de registros, estes devem estar devidamente numerados.

Outros tipos de índices

Índices bitmap

- O índice é criado em um valor específico de um campo (a coluna em uma relação) e é apenas um vetor de bits;
- Considere um índice bitmap para a coluna **C** e um valor **V** para esta coluna:
  - Para relação com **n** linhas, contém **n** bits;
  - O *i*-ésimo bit é definido como 1 se a linha **i** tiver o valor **V** para a coluna **C**;
  - Se **C** contiver o conjunto de valores $<v_1, v_2, ..., v_m>$, então **m** índices bitmap seriam criados para a coluna.

Outros tipos de índices

Índices bitmap - Exemplo

- A seguir, a relação FUNCIONARIO com colunas **Func_id, Unome, Sexo, Cep** e **Faixa_salarial** e um índice bitmap para as colunas **Sexo** e **Cep**:

Linha_id	Func_id	Unome	Sexo	Cep	Faixa_salarial
0	51024	Braga	M	09404011	-
1	23402	Pires	F	03002211	-
2	62104	Beto	M	01904611	-
3	34723	Fernandes	F	03002211	-
4	81165	Gouveia	F	01904611	-
5	13646	Hamilton	M	01904611	-
6	12676	Marcus	M	03002211	-
7	41301	Zara	F	09404011	-

Índice bitmap para Sexo:

M: 101001100 - **F:** 01011001

Outros tipos de índices

Índices bitmap - Exemplo

Índice bitmap para Cep:

Cep 01904655: 00101100 - **Cep 03002211:** 01010010 - **Cep 09409433:** 100000001

- Para encontrar funcionários com **Sexo = F** e **Cep = 30022512345**, realiza-se a inserção dos bitmaps ′01011001′ e ′01010010′, resultando nos Linha_id 1 e 3.
- Os funcionários que não moram no **Cep = 09404066** são obtidos pelo complemento do vetor de bits ′10000001′, produzindo Linha_id de 1 a 6.

Outros tipos de índices

Índices bitmap (de existência)

- Evita o gasto de renumeração de linhas e o deslocamento de bits, na exclusão de registros;
- Tem-se um bit 0 para as linhas que foram excluídas mas ainda estão presentes;
- Tem-se um bit 1 para as linhas que realmente existem;
- Nas inserções, uma entrada deve ser criada em todos os bitmaps de todas as colunas que têm índice bitmap.

Outros tipos de índices

ÍNDICES BITMAP (PARA NÓS FOLHA DE B^+-TREE)

- Bitmaps podem ser usados:
  - Em nós folha dos índices de B^+-tree;
  - Para apontar o conjunto de registros que contém cada valor específico do campo indexado no nó folha.
- Quando a B^+-tree está embutida em um campo de pesquisa não chave:
  - O registro folha contém uma lista de ponteiros de registro ao longo de cada valor do atributo indexado;
  - Para os valores frequêntes, um índice bitmap pode ser armazenado em vez de ponteiros.

Outros tipos de índices

Indexação baseada em função

- Cria um índice tal que o valor que resulta da aplicação de alguma função em um campo (ou coleção de campos) torna-se chave para o índice;

- Garantem que o sistema Oracle Database, por exemplo, usará o índice em vez de realizar uma varredura completa na tabela.

Outros tipos de índices

Indexação baseada em função - Exemplo

- Criação de índice na tabela FUNCIONARIO com base na coluna **Unome** (representada em maiúsculo):

```
CREATE INDEX idx_maiusc
ON Funcionario (UPPER(Unome))
```

Cria um índice com base na função UPPER(Unome), que retorna o sobrenome em letras maiúsculas.

UPPER('Silva') retornará '**Silva**'

Outros tipos de índices

Indexação baseada em função - Exemplo

- A consulta abaixo usará o índice:

```
SELECT Primeiro_nome, Unome
FROM Funcionario
WHERE UPPER(Unome) = 'Silva'
```

- Sem o índice baseado em função, um Oracle database poderia realizar uma varredura completa da tabela, pois um índice de B^+-tree só é pesquisado pelo uso direto do valor da coluna.

ESTRUTURAS DE INDEXAÇÃO PARA ARQUIVOS:

QUESTÕES GERAIS SOBRE INDEXAÇÃO

Questões gerais sobre indexação

Índices Lógicos X Físico

- Éndice Físico
  - O ponteiro precisa ser mudado se o registro for movimentado para outro local no disco;
  - Ex: Se a organização de arquivo primária for baseada em hashing linear ou extensível, toda vez que um bucket for dividido, alguns registros serão alocados para novos buckets e terão novos endereços físicos;
  - Se houvesse um índice secundário no arquivo, os ponteiros para esses registros devem ser localizados e atualizados.

Questões gerais sobre indexação

Índices Lógicos X Físico

- índice Lógico
  - As entradas de índice têm a forma **<K, Kp>**;
  - Cada entrada tem um valor **K** para algum valor de campo de índice secundário combinado com o valor de **Kp** (do campo usado para a organização do arquivo primário);
  - Ao pesquisar o índice secundário no valor de **K**, um programa pode localizar o valor correspondente e acessar o registro pela organização do arquivo primário.

Questões gerais sobre indexação

Índices Lógicos X Físico

- Índice Lógico
  - Introduzem um nível adicional de indireção entre a estrutura de acesso e os dados;
  - São usados quando se espera que os endereços de registro físicos mudem com frequência.

Estrutura de acesso

- O índice não faz parte integral do arquivo de dados;
- Pode ser criado e descartado dinamicamente.

Questões gerais sobre indexação

Índice Secundário

- Evitar a ordenação física dos registros no arquivo de dados em disco;
- Podem ser criados junto com qualquer organização de registro primária;
- Criação em B^+-tree:
  - Percorre todos os registros no arquivo para criar as entradas no nível de folha;
  - As entradas são classificadas e preenchidas de acordo com o fator de preenchimento especificado;
  - Outros níveis de índice são criados, simultâneamente.

Questões gerais sobre indexação

Índice para restrição de chave em atributo

- Enquanto se pesquisa o índice para inserir um novo registro, é fácil verificar ao mesmo tempo se outro registro no arquivo tem o mesmo valor de atributo de chave que o novo registro;
- Se o índice for criado em um campo não chave, ocorrem duplicatas.

Questões gerais sobre indexação

Armazenamento de relações baseado em coluna

- Alternativo ao modo tradicional de armazenar relações linha por linha;
- Permite a liberdade adicional na criaçao de índices;
- Vantagens para aplicações que exigem somente consultas de leitura;
- A mesma coluna pode estar presente em várias projeções de um tabela e os índices podem ser criados em cada projeção.

Questões gerais sobre indexação

Armazenamento de relações baseado em coluna

- Vantagens de desempenho para as seguintes áreas:
  - Particionamento vertical da tabela coluna por coluna - somente as colunas necessárias são acessadas;
  - Uso de índices por colunas e índices de junção em várias tabelas para responder as consultas sem acesso as tabelas de dados;
  - Uso de visões materializadas para suporte a consultas em múltiplas colunas.

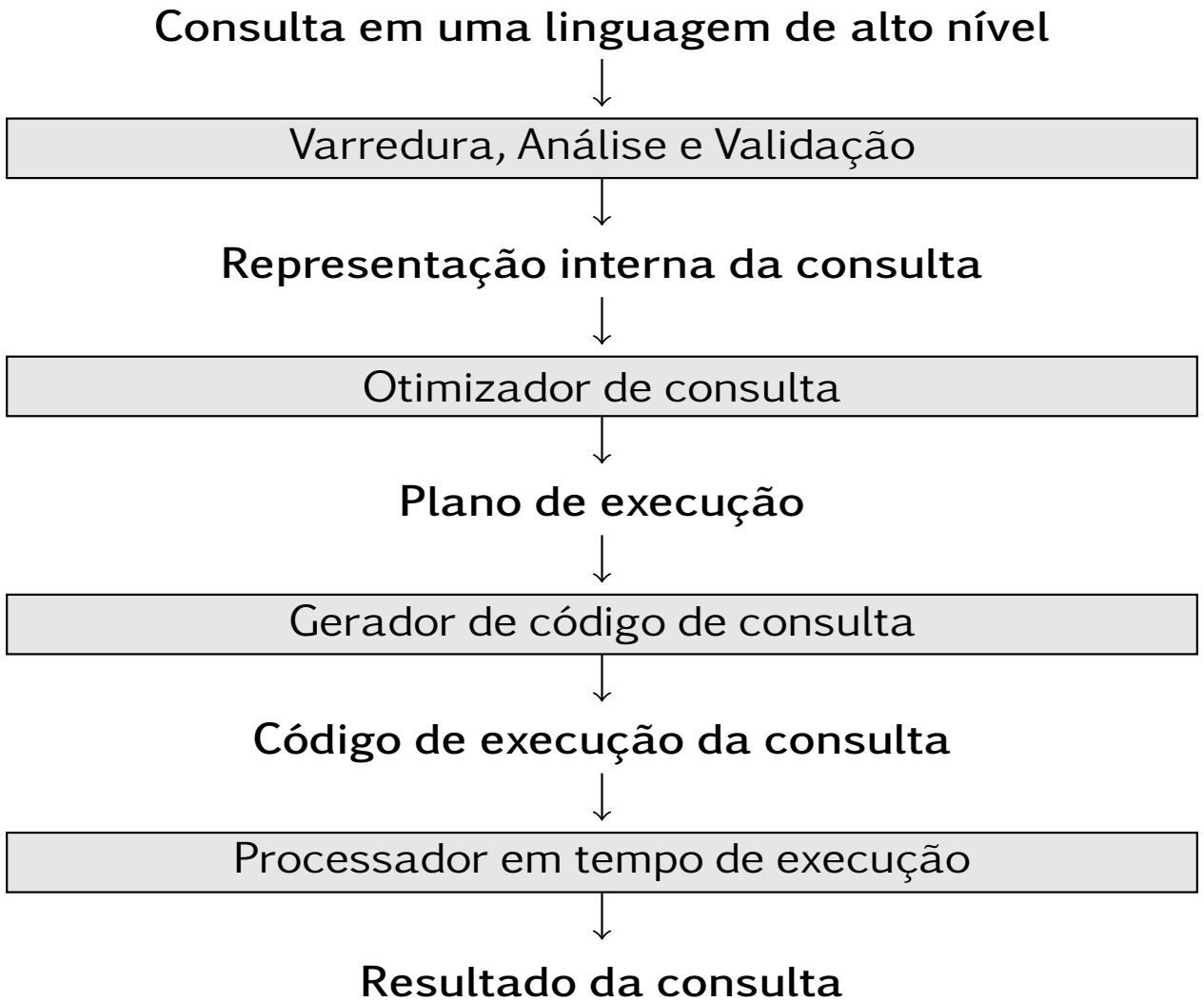
Seção 19

Processamento e Otimização de Consultas

Processamento de Consultas

- SGBDs utilizam técnicas para **processar**, **otimizar** e **executar** consultas
- Consultas precisam ser **lidas**, **analisadas** e **validadas**
  1. Leitura (Varredura) → identificação de *tokens* de consulta, como palavras-chave SQL e nomes de relações e atributos
  2. Análise → verificação sintática, determinando se a consulta segue as regras gramaticais da SQL
  3. Validação → verificação de validade de nomes de atributos e relações no esquema do BD
- Para cada consulta, é criada uma representação interna em memória, geralmente na forma de **árvore** ou **grafo**
- Plano de execução → estratégia de execução da consulta idealizada pelo SGBD para obtenção de resultados
- Otimização → processo (interno) de escolha de uma estratégia eficiente de execução

Processamento de Consultas



Processamento de Consultas

- O gerador de código de consulta produz o código, compilado ou interpretado, que será executado pelo processador em tempo de execução para produção do resultado final
- A descoberta da estratégia ótima para execução da consulta pode demandar elevado custo computacional
- Os otimizadores de consulta dos SGBDs comumente não garantem a produção do melhor plano de execução possível

Tradução de Consultas

- Para se otimizar uma consulta, ela primeiramente precisa ser traduzida para uma expressão equivalente em álgebra relacional e representada como uma árvore (ou grafo) de consulta
- A consulta precisa ser decomposta em **blocos**:
  - Unidades básicas a serem traduzidas em operadores algébricos para otimização
  - Contém uma única expressão SELECT-FROM-WHERE e, caso exista, GROUP BY e HAVING.
- Como exemplo, consideremos a consulta:

```
SELECT Unome, Pnome
FROM   FUNCIONARIO
WHERE  Salario > (SELECT MAX(Salario)
                  FROM   FUNCIONARIO
                  WHERE  Dnr=5);
```

Tradução de Consultas

- Para esse caso, temos dois blocos de consulta. O bloco de consulta externo:

```
SELECT Unome, Pnome
FROM   FUNCIONARIO
WHERE  Salario > X;
```

- Que pode ser traduzido para a seguinte expressão da álgebra relacional:

$$\pi_{Unome,Pnome}(\sigma_{Salario>X}(FUNCIONARIO))$$

Tradução de Consultas

- E o bloco de consulta interno:

```
SELECT MAX(Salario)
FROM   FUNCIONARIO
WHERE  Dnr = 5;
```

- Que pode ser traduzido para a seguinte expressão da álgebra relacional:

$$\Im_{MAX(Salario)}(\sigma_{Dnr=5}(FUNCIONARIO))$$

- Apresenta uma consulta aninhada - só precisa ser avaliado uma vez para produzir o resultado.

ALGORITMOS PARA PROCESSAMENTO E OTIMIZAÇÃO DE CONSULTA:

Algoritmos para ordenação externa

Algoritmos para ordenação externa

- Ordenação dos resultados da consulta quando a cláusula ORDER BY for especificada.

Sorting (Intercalação):

- Ordenação dos resultados da consulta quando houver as cláusulas: ORDER BY, Junção, Projeção (para eliminação de duplicata - DISTINCT);
- Pode ser evitada se um índice apropriado houver no atributo de arquivo desejado.

Sorting - Intercalação Externa:

- Adequado a grandes arquivos de registros armazenados no disco (que não cabem totalmente na memória principal);
- Estratégia ordenação-intercalação (sort-merge);
- Exige buffer na memória principal, onde se realiza a classificação e mesclagem;
- Dividido em buffers individuais.
  - Mantém o conteúdo de exatamente um bloco de disco.

SORTING - INTERCALAÇÃO EXTERNA:

1. Fase de ordenação (classificação)
    - Classificar pequenos "subarquivos";
    - Os pedaços do arquivo são lidos para memória principal, ordenados e regravados no disco (como subarquivos temporários);
    - Tamanho de cada pedaço ($\mathbf{n_R}$) definido pelo número de blocos de arquivo ($\mathbf{b}$);
    - Número de blocos de arquivo ($\mathbf{n}$) definido pelo espaço de buffer disponível ($\mathbf{n_B}$).

SORTING - INTERCALAÇÃO EXTERNA:

1. Fase de ordenação (classificação) - Exemplo:

   - Se o número de buffers disponíveis na memória principal n_B = **5 blocos** e o tamanho do arquivo **b = 1.024 blocos**, então $n_r = (b/n_B)$ = 205 pedaços iniciais.

   - Após a ordenação, 205 pedaços ordenados são armazenados como subarquivos temporários no disco.

Sorting - Intercalação Externa:

2 Fase de intecalação (mesclagem):

- Pedaços ordenados mesclado em passos de intercalação;
- Cada passo de intercalação pode ter uma ou mais etapas;
- **Grau de intercalação (d_M)** = número de subarquivos ordenados que podem ser mesclados em cada etapa, que contém:
  - **Bloco de buffer** - mantém um bloco de disco em cada subarquivo;
  - **Bloco de buffer adicional** - mantém um bloco de disco do resultado da intercalação.

Sorting - Intercalação Externa:

2 Fase de intecalação (mesclagem):

- $\mathbf{d_M}$ é o menor de $(\mathbf{n_B}\text{-}1)$ e $\mathbf{n_R}$.
- **Número de passos da intercalação** = $(\log_{d,M}(n_R))$
- Exemplo:
  - $\mathbf{n_B} = 5$ e $\mathbf{d_M} = 4$, de modo que os 205 pedaços iniciais ordenados seriam intercalados 4 de cada vez em cada etapa em 52 subarquivos, ao final do primeiro passo de intercalação;
  - Os 52 arquivos ordenados serão intercalados quatro de cada vez em 13 arquivos ordenados, que depois intercalados em 4 arquivos ordenados e, por fim, para um arquivo totalmente ordenado.

SORTING - INTERCALAÇÃO EXTERNA:

2 Fase de intecalação (mesclagem):

- O desempenho pode ser medido a partir do número de leituras e gravações de bloco de disco, antes que a ordenação seja concluída. Custo expresso na fórmula:

$$(2 * b) + (2 * b * (\log_{d.M} n_R)), \text{ onde}$$

- $(2 * b)$ = Número de acessos de blocos para a fase de ordenação.
- $(2 * b *$ = Número de acessos de blocos para a fase de intercalação.
- $(\log_{d.M} n_R))$ = Número de passos de intercalação.

ALGORITMOS PARA PROCESSAMENTO E OTIMIZAÇÃO DE CONSULTA:

Algoritmos para operação de Seleção

Algoritmos para operação de Seleção

Exemplos para as próximas discussões

- Considere o seguinte banco de dados relacional (parte 1):

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Endereco	Sexo	Salario	Cpf_supervisor	Dnr

DEPARTAMENTO

Dnome	Dnumero	Cpf_gerente	Inicio_gerente

LOCALIZACAO_DEP

Dnumero	Dlocal

Algoritmos para operação de Seleção

Exemplos para as próximas discussões

- Considere o seguinte banco de dados relacional (parte 2):

PROJETO

Projnome	Projnumero	Projlocal	Dnum

TRABALHA_EM

Fcpf	Pnr	Horas

DEPENDENTE

Fcpf	Nome_dependente	Sexo	Datanasc	Parentesco

Algoritmos para operação de Seleção

Exemplos para as próximas discussões

- Considere as seguintes operações aplicadas ao banco de dados apresentado no slide anterior:

OP1: $\sigma_{Cpf='12345678966'}(FUNCIONARIO)$

OP2: $\sigma_{Dnumero>5}(DEPARTAMENTO)$

OP3: $\sigma_{Dnr=5}(FUNCIONARIO)$

OP4: $\sigma_{Dnr=5\ AND\ Salario=30000\ AND\ Sexo='F'}(FUNCIONARIO)$

OP5: $\sigma_{Fcpf='12345678966'\ AND\ Pnr=10}(TRABALHA\_EM)$

- A seguir, serão apresentados os métodos para varredura de arquivos para seleção simples.

Algoritmos para operação de Seleção

1. Pesquisa Linear
   - Recupera cada registro do arquivo e testa se seus valores de atributo satisfazem a condição de seleção;
   - Diferente dos demais métodos (que dependem de um caminho de acesso apropriado no atributo usado na condição de seleção), se aplica a qualquer arquivo.
2. Pesquisa Binária
   - O mais eficiente se a condição envolver uma comparação de igualdade em um atributo chave, no qual o arquivo é ordenado;
   - Exige o arquivo seja ordenado no atributo de pesquisa.

Algoritmos para operação de Seleção

3 Usando um índice primário

- Utilizar se a condição de seleção envolver uma comparação de igualdade entre este atributo.

4 Usando uma chave hash

- Utilizar se a condição de seleção envolver uma comparação de igualdade entre esta chave.

Algoritmos para operação de Seleção

5 Usando um índice primário para recuperar vários registros

- Nos casos de condição de comparação >, >=, < ou <= em um campo chave com um índice primário (ex: Dnumero > 5):
  1. Utilizar o índice para encontrar o registro que satisfaz a igualdade (ex: Dnumero = 5);
  2. Recuperar todos os registros subsequentes no arquivo ordenado;
- Aplicáveis a consultas de intervalos.

Algoritmos para operação de Seleção

6 Usando um índice de agrupamento para recuperar vários registros

- Se a condição de seleção envolver uma comparação de igualdade em um atributo não chave com índice de agrupamento, utilizar o índice para recuperar todos os registros que satisfazem a seleção.

Algoritmos para operação de Seleção

7 Usando um índice secundário (B^+-tree em uma comparação de igualdade

- Utilizado para recuperar um único registro se o campo de índice for uma chave ou para recuperar múltiplos registros se o campo de índice não for uma chave;
- Aplicáveis a consultas de intervalos.

- A seguir, serão apresentados os métodos de pesquisa para selação complexa (ex: presença de conectivo lógico AND).

Algoritmos para operação de Seleção

8 Seleção conjuntiva usando um índice individual

- Se o atributo tiver caminho de acesso que permita os métodos (2) a (7):
  1. Utilizar esta condição para recuperação;
  2. Verificar se cada registro satisfaz as condições simples restantes da seleção.

9 Seleção conjuntiva usando índice composto

- Se dois ou mais atributos estiverem envolvidos nas condições de igualdade na seleção conjuntiva e um índice composto existe nos campos combinados, pode-se usar o índice diretamente.

Algoritmos para operação de Seleção

10 Seleção conjuntiva por untersecção de ponteiros de registro

- Os índices secundários devem estar disponíveis em mais de um dos campos envolvidos na seleção simples e devem incluir ponteiros de registros;
- A intersecção desses conjuntos de ponteiros gera os ponteiros de registro que satisfazem a condição de seleção conjuntiva;
- Se apenas uma das condições tiverem índices secundários, cada registro recuperado é testado mais de uma vez para verificar se o mesmo satisfaz a condição.

Algoritmos para operação de Seleção

- Uma condição disjuntiva (conectivo lógico OR) é complexo para se otimizar e processar. Considere a OP4′:

$$\sigma_{Dnr=5\ AND\ Salario=30000\ AND\ Sexo='F'}(FUNCIONARIO)$$

  - Neste caso, pouca otimização pode ser feita, pois os registros que satisfazem a condição são a união dos registros que satisfazem as condições individuais;
  - Se qualquer uma das condições não tiver um caminho de acesso, deve-se utilizar a pesquisa linear.

ALGORITMOS PARA PROCESSAMENTO E OTIMIZAÇÃO DE CONSULTA:

Algoritmos para operação de Junção

Algoritmos para operação de Junção

- Junções multivias: envolvem mais de dois arquivos.
  - O número de vias para executar junções multivias cresce muito rapidamente.
  - A seguir, serão apresentadas técnicas somente para junções de duas vias, a partir da junção no seguinte formato:

$$R \bowtie_{A=B} (S)$$

Onde A e B são atributos de junção, que devem ser atributos compatíveis por domínio de R e S, respectivamente.

Algoritmos para operação de Junção

- As operações a seguir serão utilizadas para as demais discussões:

$$\text{FUNCIONARIO} \overset{\underline{\text{OP6:}}}{\bowtie}_{Dnr=Dnumero} (DEPARTAMENTO)$$

$$\text{DEPARTAMENTO} \overset{\underline{\text{OP7:}}}{\bowtie}_{Cpf\_ger=Cpf} (FUNCIONARIO)$$

Algoritmos para operação de Junção

J1 Junção de loop aninhado

- Algoritmo padrão - não existe quaisquer caminhos de acesso especiais.

1. Para cada registro **t** em **R**, recupere cada registro **s** de **S**;
2. Testar se os dois registros satisfazem a relação **t**[A] = **s**[B].

J2 Junção de único loop

- Utiliza-se estrutura de acesso para recuperar os registros correspondentes;
- Somente se houver índice para um dos dois atributos.

1. Para um dos dois atributos, recupere cada registro **t** em **R**;
2. Usar a estrutura de acesso para recuperar diretamente os registros correspondentes **s** de **S** que satisfazem **s**[B] = **t**[A].

Algoritmos para operação de Junção

J3 Junção ordenação-intercalação

- Desejável que os registros **R** e **S** estejam fisicamente ordenados por valor dos atributos de junção A e B;
- Os dois arquivos são varridos simultâneamente, na ordem dos atributos de junção, combinando os registros;
- Pares de blocos de arquivo são copiados para buffers de memória na ordem e os registros de cada arquivo são varridos apenas uma vez cada.

Algoritmos para operação de Junção

J4 Junção de partição-hash

- Os registros de **R** e **S** são particionados em arquivos menores;
- O particionamento é feito usando a mesma função hashing **h** no atributo de junção **A** de **R** (para particionamento do arquivo **R**) e **B** de **S** (para particionamento do arquivo **S**);
- **Fase de particionamento** e **Fase de investigação**.

Algoritmos para operação de Junção

J4 Junção de partição-hash

1. Fase de particionamento
   - Cria-se hashes dos registros do menor arquivo para as diversas partições de **R**;
   - A coleção de registros com o mesmo valor de **h(A)** é colocada na mesma partição, bucket de hash em uma tabela hash na memória principal.

2. Fase de investigação
   - Cria-se o hash de cada registro do arquivo **S** usando a mesma função de hash **h(B)** para investigar o bucket apropriado;
   - O registro é então combinado com todos os correspondentes de **R** nesse bucket.

ALGORITMOS PARA PROCESSAMENTO E OTIMIZAÇÃO DE CONSULTA:

Algoritmos para operação de Projeção

Algoritmos para operação de Projeção

$$\text{pi}_{\text{<lista atributos>}}(R)$$

- **Se <lista atributos> incluir uma chave da relação R**
  - É simples de se implementar, pois nesse caso o resutlado da operação terá o mesmo número de tuplas que R.
- **Se <lista atributos> não incluir uma chave da relação R**
  - As tuplas duplicadas devem ser eliminadas.

Algoritmos para operação de Projeção

A duplicidade pode ser eliminada...

- Ao ordenar o resultado e depois eliminar as tuplas duplicadas (que aparecem consecutivamente após a ordenação).
- A partir do hashing:
  - A medida que cada registro gera um hash e é inserido em um bucket do arquivo hash, ele é comparado com os demais que já estão no bucket.

ALGORITMOS PARA PROCESSAMENTO E OTIMIZAÇÃO DE CONSULTA:

Algoritmos para operações de conjunto

Algoritmos para operações de conjunto

Produto Cartesiano

- Seu resultado (R x S) inclui um registro para cada combinação de registros de R e S;
- Importante evitar esta operação e substituí-la por outras operações, como a junção, durante a otimização da consulta.

União, Intersecção e Diferença de Conjunto

- Relações compatíveis no tipo;

União, Intersecção e Diferença de Conjunto

- Utilizar várias técnicas de ordenação-intercalação:
  1. As duas relações são ordenadas nos mesmos atributos;
  2. Uma única varredura por cada relação é suficiente para produzir o resultado.

- Também pode-se utilizar o hashing:
  1. Uma tabela é varrida e particionada em uma tabela hash na memória com buckets;
  2. Os registros na outra tabela são varridos um de cada vez e usados para investigar a partição apropriada.

ALGORITMOS PARA PROCESSAMENTO E OTIMIZAÇÃO DE CONSULTA:

Implementando operações de agregação

Implementando operações de agregação

- Operações de agregação (**MIN, MAX, COUNT, AVERAGE, SUM**) quando aplicadas a uma tabela inteira, podem ser calculados por uma varredura de tabela ou usando um índice apropriado.
- Exemplo:

```
SELECT MAX(Salario)
FROM   FUNCIONARIO
```

- Se houver índice de B^+-tree (crescente) em **Salario** para a relação **FUNCIONARIO**, então o otimizador pode decidir sobre o uso do índice Salario para procurar.

Implementando operações de agregação

- No uso do **GROUP BY**, o operador de agregação deve ser aplicado separadamente a cada grupo de tuplas. Logo, a tabela precisa primeiro ser particionada em subconjuntos de tuplas.
  - Cada partição com o mesmo valor para os atributos de agrupamento.
- Exemplo:

```
SELECT Dnr, AVG(Salario)
FROM FUNCIONARIO
GROUP BY Dnr;
```

Implementando operações de agregação

1. Utilizar a ordenação ou o hashing nos atributos de agrupamento para particionar os arquivos;
2. Calcular a função de agregação para as tuplas de cada grupo obtido em (1). Ou seja, no conjunto de tuplas de **FUNCIONARIO**, cada número de departamento seria agrupado em uma partição e o salário médio calculado para cada grupo.

- Se houver um índice de agrupamento nos atributos de agrupamento, os registros já estarão particionados.

ALGORITMOS PARA PROCESSAMENTO E OTIMIZAÇÃO DE CONSULTA:

Implementando operações de junção externa

Implementando operações de junção externa

- Exemplo:

```
SELECT Unome, Pnome, Dnome
FROM (FUNCIONARIO LEFT OUTER JOIN DEPARTAMENTO
      ON Dnr=Dnumero);
```

- O resultado desta consulta é uma tabela de nomes de funcionários e seus departamentos associados;
- Pode ser calculada modificando-se um dos algoritmos de junção;
- Também pode ser calculada ao executar uma combinação de operadores da álgebra relacional.

Implementando operações de junção externa

- Sequência de operações relacionadas a consulta do exemplo:

1. Calcule a JUNÇÃO das tabelas **FUNCIONARIO** e **DEPARTAMENTO**:

$$A \longleftarrow FUNCIONARIO \bowtie_{Dnr=Dnumero} (DEPARTAMENTO)$$
$$B \longleftarrow \pi_{Unome,\ Pnome,\ Dnome}(A)$$

2 Ache as tuplas de **FUNCIONARIO** que não aparecem no resultado da JUNÇÃO:

$$C \longleftarrow \pi_{Unome,\ Pnome}(FUNCIONARIO)$$
$$D \longleftarrow \pi_{Unome,\ Pnome}(B)$$
$$E \longleftarrow C - D$$

Implementando operações de junção externa

- Sequência de operações relacionadas a consulta do exemplo:

3 Preencha cada tupla em **E** com um campo Dnome NULL:

$E \longleftarrow E \ x \ NULL$

4 Aplique a operação UNION a **B** e **E** para produzir o resultado JUNÇÃO EXTERNA A ESQUERDA:

$RESULT \longleftarrow B \cup E$

- O custo, portanto, seria a soma dos custos das etapas associadas.

ALGORITMOS PARA PROCESSAMENTO E OTIMIZAÇÃO DE CONSULTA:

Combinando operações com pipelining

Combinando operações com pipelining

- Uma consulta em SQL é traduzida para uma expressão da álgebra relacional (sequência de operações relacionais);
- Se cada operação for executada uma de cada vez, é necessário gerar arquivos temporários no disco para manter os resultados;
  - Gera um overhead excessivo.
- Para reduzir a quantidade desses arquivos, gera-se um código de execução de consulta que corresponde a algoritmos para combinações de operações em uma consulta.

Combinando operações com pipelining

- A partir deste código de execução da consulta é gerado apenas um arquivo de resultado.
  - Processo chamado de **pipelining** ou **processamento baseado em fluxo**.
- Normalmente, cria-se um código de execução de consultas de maneira dinâmica para implementar múltiplas operações.
  - Combina vários algoritmos que correspondem Ãăs operações individuais.

ALGORITMOS PARA PROCESSAMENTO E OTIMIZAÇÃO DE CONSULTA:

Usando heurísticas na otimização da consulta

Usando heurísticas na otimização da consulta

Árvore de consulta

- Estrutrura de dados de árvore que corresponde a uma expressão da álgebra relacional;
- Representa as **relações de entrada** da consulta como **nós folha**;
- Representa as **operações** da álgebra relacional como **nós internos**.

Usando heurísticas na otimização da consulta

Árvore de consulta

- Sua execução consiste na execução de uma operação de nó interno sempre que seus operandos estão disponíveis e depois na substituição desse nó interno pela relação que resulta da execução da operação;

- Termina quando a operação do nó raíz é executada e produz a relação de resultado para a consulta.

Árvore de consulta - Exemplo

- A seguir, um exemplo de consulta a partir do esquema relacional EMPRESA, em álgebra relacional, SQL e em árvore de consulta.

- **Para cada projeto localizado em 'Mauá', recuperar o número do projeto, o número do departamento de controle, o sobrenome, o endereço e a data de nascimento do gerente do departamento.**

Árvore de consulta - Exemplo

- Consulta representada em Álgebra Relacional:

$$A \longleftarrow \sigma_{Projlocal='Mau'}(PROJETO)$$
$$B \longleftarrow A \bowtie_{Dnum=Dnumero} (DEPARTAMENTO)$$
$$C \longleftarrow B \bowtie_{Cpf\_ger = Cpf} (FUNCIONARIO)$$
$$D \longleftarrow \pi_{Projnumero,\ Dnum,\ Unome,\ Endereco,\ Data\_nasc}(C)$$

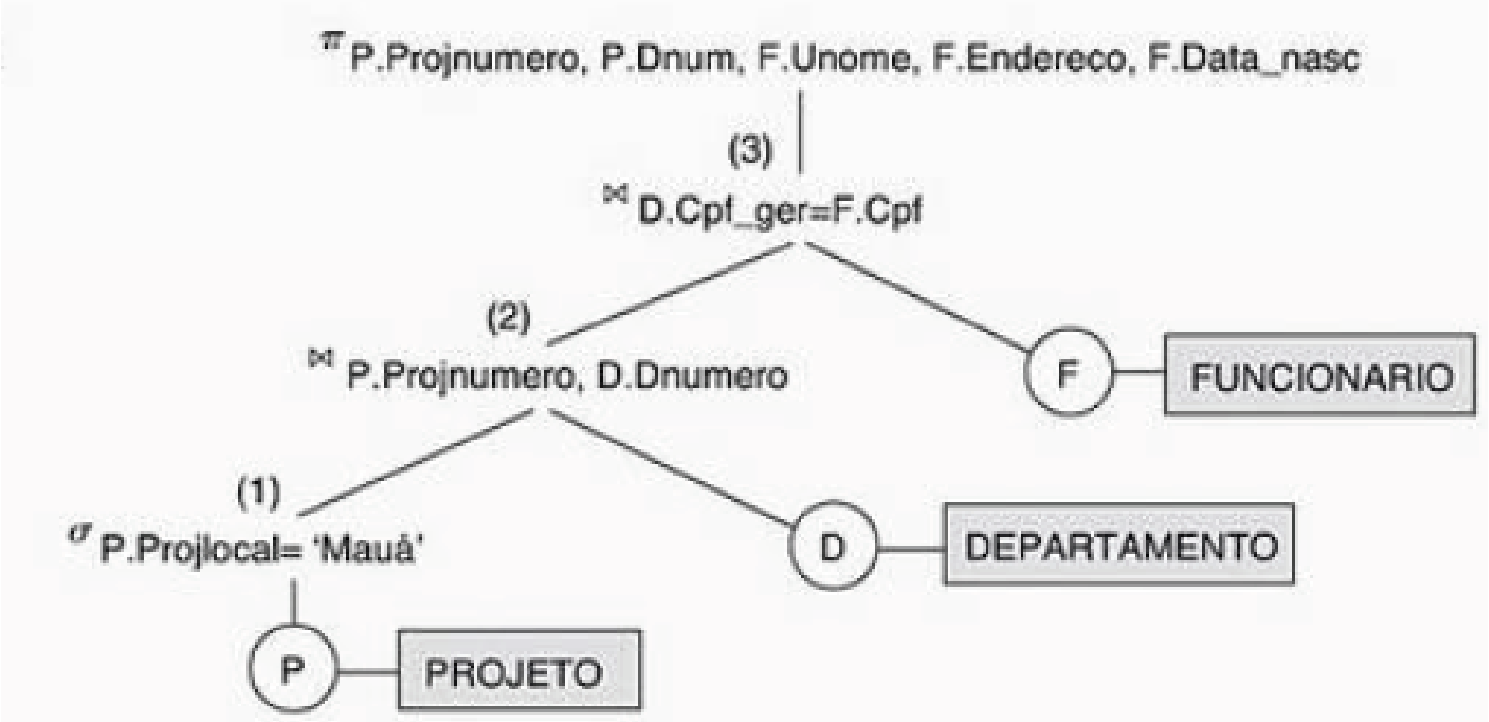
Árvore de consulta - Exemplo

- Consulta representada em SQL:

```
SELECT P.Projnumero,
       P.Dnum,
       F.Unome,
       F.Endereco,
       F.Data_nasc
FROM   PROJETO      AS P,
       DEPARTAMENTO AS D,
       FUNCIONARIO  AS F
WHERE P.Dnum      = D.Dnumero
  AND D.Cpf_ger   = F.Cpf
  AND P.Projlocal = 'Maua';
```

Árvore de consulta - Exemplo

- Consulta representada em árvore:



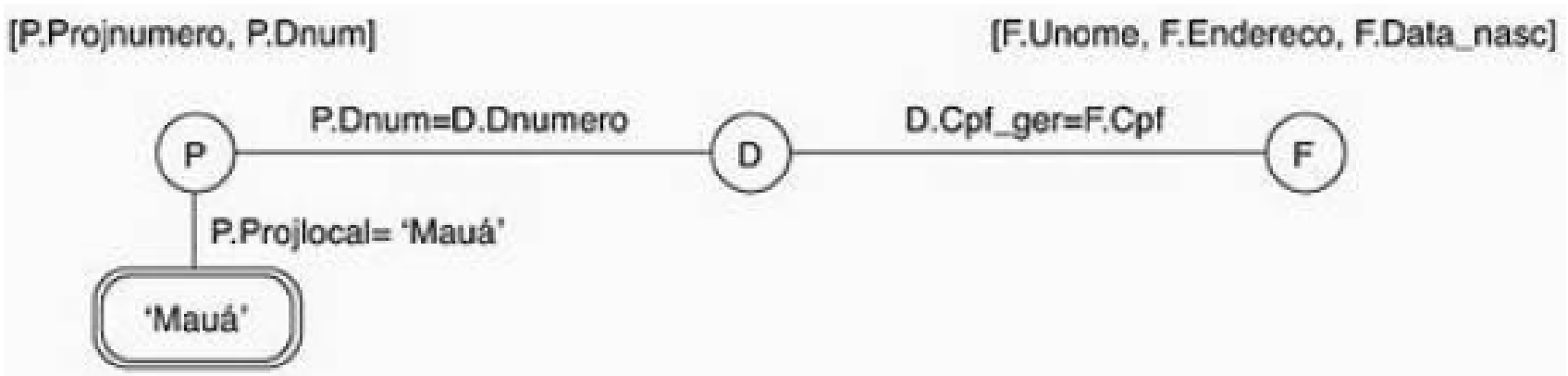
Usando heurísticas na otimização da consulta

Grafo de consulta

- As **relações de consulta** são representadas por **nós de relação**;
- Os **valores constantes** são representados por **nós de constantes**;
- As **condições de seleção** são representadas pelas **arestas**;
- Os **atributos** a serem recuperados são exibidos entre colchetes, acima das relações correspondentes;
- Não indica a ordem sobre quais operações realizar primeiro.

Grafo de consulta - Exemplo

- Consulta representada em grafo:



Usando heurísticas na otimização da consulta

- Processo de uma consulta (incluindo passos para otimização de desempenho):

  1. A varredura e o analisador gera primeiro uma estrutura de dados - **representação inicial de consulta**;

  2. A representação inicial é otimizada de acordo com regras heurísticas - **representação da consulta otimizada**;

  3. É gerado um plano de execução para executar grupos de operações com base nos caminhos de acesso disponíveis nos arquivos envolvidos.

Usando heurísticas na otimização da consulta

- A seguir, algumas técnicas de otimização que aplicam regras heurísticas para modificar a representação interna de uma consulta para melhorar seu desempenho.
  - **Principal: aplicar operações SELEÇÃO e PROJEÇÃO antes da JUNÇÃO ou outras operações binárias.**
- O otimizador de consulta heurística transforma a árvore de consulta inicial em uma árvore de consulta final equivalente, eficiente para execução.
  - Deve incluir regras para equivalência entre expressões da álgebra relacional.

Usando heurísticas na otimização da consulta

- A seguir, **regras de transformações gerais para operações de álgebra relacional**:

1. Cascata de σ
   - Uma condição de seleção conjuntiva pode ser desmembrada em uma cascata (sequência) de σ como operações individuais.

2. Comutatividade de σ
   - A operação σ é comutativa:

     $\sigma_{c1}(\sigma_{c2}(R))$ equivale a $\sigma_{c2}(\sigma_{c1}(R))$

Usando heurísticas na otimização da consulta

- A seguir, **regras de transformações gerais para operações de álgebra relacional**:

3 Cascata de π
  - Em uma cascata (sequência) de operações π, todas podem ser ignoradas, menos á última:

    $\pi_{Lista_1}(\pi_{Lista_2}(...(\pi_{Lista_n}(R))...))$ equivale a $\pi_{Lista_1}(R)$

4 Comutação de σ com π
  - Se a condição de seleção c envolve apenas os atributos A_1, ..., A_n na lista de projeção, as duas operações podem ser comutadas:

    $\pi_{A_1 . A_2 . ..., A_n}(\sigma_c(R))$ equivale a $\sigma_c(\pi_{A_1 . A_2 . ..., A_n}(R))$

Usando heurísticas na otimização da consulta

- A seguir, **regras de transformações gerais para operações de álgebra relacional:**

5 Comutatividade de $\bowtie$ (e **x**)

- A operação de junção é comutativa, assim como a operação X:

  $R \bowtie_c S$ *equivalea* $S \bowtie_c R$

  R x S equivale a S x R

6 Comutação de σ com $\bowtie$

- Se todos os atributos na condição de seleção **c** envolvem apenas os atributos de uma das relações sendo juntadas, as duas operações podem ser comutadas da seguinte forma:

  $\sigma_c(R \bowtie S)$ *equivale a* $(\sigma_c(R)) \bowtie S$

Usando heurísticas na otimização da consulta

- A seguir, **regras de transformações gerais para operações de álgebra relacional**:

7 Comutação de π com $\bowtie$ (ou x)

- Suponha que a lista de projeção seja $L = A_1, ..., A_n, B_1, ..., B_m$, onde $A_1, ..., A_n$, são os atributos de R e $B_1, ..., B_m$ são atributos de S.

- Se a condição de junção **c** envolver apenas atributos de L, as duas operações podem ser comutadas conforme:

$$\pi_L(R \bowtie_c S) \underline{\textit{equivale a}} (\pi_{A_1, ..., A_n}(R)) \bowtie_c (\pi_{B_1, ..., B_M}(S))$$

Usando heurísticas na otimização da consulta

- A seguir, **regras de transformações gerais para operações de álgebra relacional**:

8 Comutatividade das operações de conjunto

- As operações de conjunto $\cap$ e $\cup$ são comutativas, mas - (diferença) não é.

9 Associatividade de $\bowtie$, x, $\cup$ e $\cap$

- Essas quatro operações são associativas individualmente.

Usando heurísticas na otimização da consulta

- A seguir, **regras de transformações gerais para operações de álgebra relacional**:

10 Comutatividade de σ com operações de conjunto

- A operação σ comuta com $\cup$, $\cap$ e -. Se θ indicar qualquer uma dessas três operações, temos:

  $\sigma_c(R\ \theta\ S)$ equivale a $(\sigma_c(R))\ \theta\ (\sigma_c(S))$

11 A operação π comuta com $\cup$

- $\pi_L(R\ \cup\ S)$ equivale a $(\pi_L(R))\ \cup\ (\pi_L(S))$

Usando heurísticas na otimização da consulta

- A seguir, **regras de transformações gerais para operações de álgebra relacional**:

12 Convertendo uma sequência (σ, x) em $\bowtie$

- Se a condição c de um σ que segue um x corresponde a uma condição de junção, converta a sequência (σ, x) em $\bowtie$, da seguinte forma:

  $(\sigma_c(R \times S))$ equivale a $(R \bowtie_c S)$

Usando heurísticas na otimização da consulta

Esboço de algoritmo de otimização algébrica heurística

1. Usando a Regra 1, quebre quaisquer operações de SELEÇÃO com condições conjuntivas em uma cascata de operações;

2. Usando as Regras 2, 4, 6 e 10, mova cada operação SELEÇÃO o mais baixo possível na árvore de consulta (que for permitido pelos atributos na condição de seleção);

Usando heurísticas na otimização da consulta

Esboço de algoritmo de otimização algébrica heurística

3 Usando as Regras 5 e 9, reorganize os nós folha a partir dos critérios:

- Posicione as relações do nó folha com as operações SELEÇÃO mais restritivas;
- Garanta que a ordenação dos nós folha não cause operações de PRODUTO CARTESIANO.

4 Usando a Regra 12, combine uma operação PRODUTO CARTESIANO com uma operação SELEÇÃO subsequente na árvore para uma operação JUNÇÃO, se for o caso;

Usando heurísticas na otimização da consulta

Esboço de algoritmo de otimização algébrica heurística

5 Usando as Regras 3, 4, 7 e 11, desmembre e mova as listas de atributos de projeção para baixo da árvore no máximo possível, criando novas operações de PROJEÇÃO;

6 Identifique subárvores que representam grupos de operações que podem ser executados por um único algoritmo.

Usando heurísticas na otimização da consulta

Esboço de algoritmo de otimização algébrica heurística - Resumo

- Aplicar primeiro as operações que reduzem o tamanho dos resultados intermediários;
- As operações SELEÇÃO e JUNÇÃO devem ser executadas antes de outras operações semelhantes;
- Reordenação dos nós folha da árvore, enquanto evita produtos cartesianos, e do ajuste do restante da árvore.

Usando heurísticas na otimização da consulta

- O **plano de execução** inclui:
  - Informações sobre os métodos de acesso disponíveis em cada relação;
  - Algoritmos a serem usados no processamento dos operadores relacionais representados em árvore.

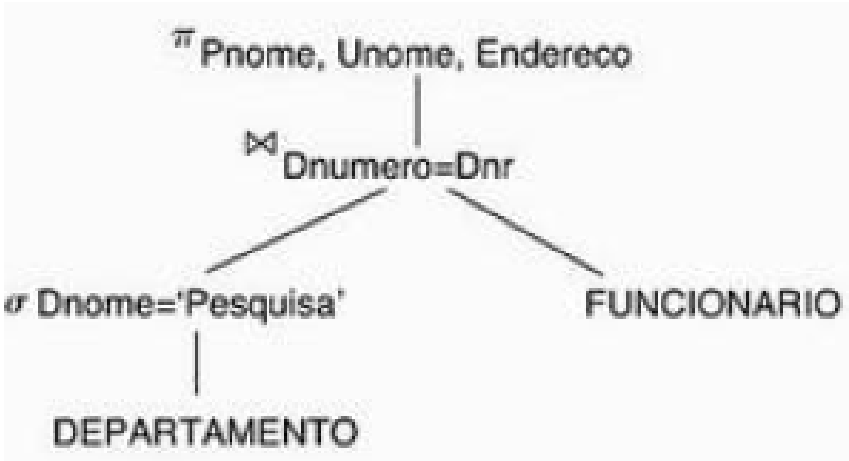
Usando heurísticas na otimização da consulta

Conversão: árvores de consulta em planos de execução

- Considere a consulta abaixo:

$$A \longleftarrow \sigma_{Dnome = 'Pesquisa'}(DEPARTAMENTO)$$
$$B \longleftarrow A \bowtie_{Dnumero = Dnr} (FUNCIONARIO)$$
$$C \longleftarrow \pi_{Pnome,\ Unome,\ Endereco} (B)$$

- Árvore de consulta correspondente:



Usando heurísticas na otimização da consulta

Conversão: árvores de consulta em planos de execução

- O otimizador escolhe:

  1. Uma busca de índice para a operação SELEÇÃO em DEPARTAMENTO;

  2. Um algoritmo de junção de único loop que percorra todos os registros no resultado da operação SELEÇÃO em DEPARTAMENTO para a operação de junção;

  3. Uma varredura do resultado de JUNÇÃO para a entrada do operador PROJEÇÃO.

Usando heurísticas na otimização da consulta

Conversão: árvores de consulta em planos de execução

- Pode especificar uma **avaliação materializada**
  - O resultado é armazenado como uma relação temporária, ou seja, fisicamente materializado.
- **Avaliação em pipeline**
  - A medida que as tuplas resultantes de uma operação são produzidas, são encaminhadas para a próxima operação na sequência da consulta;
  - Economia de custo por não ter de gravar os resultados imediatos em disco;
  - Não tem de ler os registros de volta para a próxima operação.

Otimização de consulta baseada em custo

- O otimizador limita o número de estratégias de execução a serem consideradas; caso contrário, muito tempo será gasto.
- Essa técnica é mais adequada para **consultas compiladas**, nas quais a otimização é feita na compilação e o código da estratégia de execução resultante é executado em tempo de execução).
- Para **consultas interpretadas**, uma otimização em escala total pode atrasar o tempo de resposta.

Componentes de custo para execução de consulta

1. **Custo de acesso ao armazenamento secundário:** Também conhecido como ***custo de E/S (entrada/saída) de disco***. O custo de procurar registros em um arquivo de disco depende do tipo de estruturas de acesso nesse arquivo. Além disso, fatores como se os blocos de arquivo estão alocados consecutivamente ou espalhados afetam o custo de acesso. Ênfase principal para banco de dados grandes.

2. **Custo de armazenamento em disco:** Custo de armazenar no disco quaisquer arquivos intermediários que sejam gerados por uma estratégia de execução para a consulta.

Componentes de custo para execução de consulta

3. **Custo de computação:** Custo de realizar operações na memória em registros dentro dos buffers de dados durante a execução da consulta. Também é conhecido como ***custo de CPU*** (unidade central de processamento). Ênfase de banco de dados menores e distribuídos.

4. **Custo de uso da memória:** Custo que diz respeito ao número de buffers da memória principal necessários durante a execução da consulta.

5. **Custo de comunicação:** Custo de envio da consulta e seus resultados do local do BD até o local ou terminal onde a consulta foi originada.

Informações de catálogo usada nas funções de custo

A informação necessária para as funções de custo pode ser armazenada no catálogo do SGBD, onde é acessada pelo otimizador de consulta.

- Tamanho de cada arquivo
- Para um arquivo cujos registros são do mesmo tipo:
  - **Número de registros (tuplas)** *(r)*
  - **Tamanho do registro (médio)** *(R)*
  - **Número de blocos do arquivo** *(b)*
- **Fator de bloco** *(bfr)*
- A ***organização de arquivo primária*** para cada arquivo

Informações de catálogo usada nas funções de custo

A informação necessária para as funções de custo pode ser armazenada no catálogo do SGBD, onde é acessada pelo otimizador de consulta.

- O **número de níveis** (x) de cada índice multinível (primário, secundário ou de agrupamento) para funções de custo que estimam o número de acessos de bloco.
- **Número de blocos de índice de primeiro nível** ($bL1$)
- **Número de valores distintos** (d) de um atributo
- **Seletividade** (sl), fração de registros que satisfazem uma condição de igualdade no atributo.

Algoritmos para processamento e otimização de consulta

Informações de catálogo usada nas funções de custo

A informação necessária para as funções de custo pode ser armazenada no catálogo do SGBD, onde é acessada pelo otimizador de consulta.

- Estimativa da **cardinalidade de seleção** ***(s = sl * r)*** de um atributo, número *médio* de registros que satisfarão uma condição de seleção de igualdade nesse atributo.
  - Para um ***atributo chave, d = r, sl = 1/r e s = 1***
  - Para um **atributo não chave**, ***d*** valores distintos distribuídos uniformemente entre os registros, *sl = (1/d), s = (r/d)*
- O otimizador de consulta precisará de valores próximos, mas não necessariamente atualizados até o último minuto desses parâmetros para uso na estimativa do custo.

Seção 20

Projeto Físico e Sintonia de SGBDRs

Projeto Físico em SGBDRs

- Criação de estrutura apropriada para armazenamento dos dados, de forma a garantir um bom desempenho
- Combinação de tarefas:
  - Consultas, transações e aplicações que deverão usar o banco de dados;
  - Analisadas minuciosamente pelos administradores/ projetistas.
- Deve-se ter uma boa ideia da intenção de uso do BD

Projeto Físico em SGBDRs

- Fatores que influenciam decisões sobre o projeto físico:
  - Consultas e transações
  - Frequência esperada de chamada
  - Restrições de tempo de execução
  - Frequência esperada de atualização
  - Restrições de exclusividade em atributos

Consultas e transações

- Consultas de recuperação:
  1. Quais arquivos serão acessados pela consulta?
  2. Quais atributos estão especificados na consulta?*
  3. A condição de seleção é uma condição de igualdade, desigualdade ou de intervalo?
  4. Quais são os atributos em que são especificadas condições de junção?*
  5. Quais são os atributos cujos valores serão recuperados?

* Candidatos para definição de estruturas de acesso (índices, chaves de hash ou ordenação do arquivo).

Consultas e transações

- Consultas de atualização:
  1. Quais arquivos serão atualizados?
  2. Que operações serão feitas em cada arquivo?
  3. Quais atributos estão submetidos a uma condição?*
  4. Quais atributos em que são especificadas condições de junção?*
  5. Quais atributos terão valores alterados (por atualização)?**

* Candidatos para definição de estruturas de acesso.

** Candidatos para evitar uma estrutura de acesso (ao modificá-los, exigirá atualização das estruturas de acesso).

Frequência esperada de chamada

- Consulta de recuperação
  - Quantas vezes espera-se executar a consulta em um determinado período de tempo?
  - Expressa como: frequência de uso esperada de cada atributo nos arquivos como atributo de seleção ou junção.
- Regra dos 80-20 (para um volume maior de dados):

  Cerca de 80% do processamento é atribuído a apenas 20% das consultas e transações.

Restrições de tempo de execução

- Ex: Transações que devem ser concluídas em no máximo 5 segundos em 95% das vezes
- Análise criteriosa na definição de estruturas de acesso primária para os arquivos.
  - Priorizar atributos de seleção (para consultas e transações) com restrições de tempo.

Frequências esperada de atualização

- Arquivos frequentemente utilizados devem ter um menor número de caminhos de acesso;
- O *overhead* para atualização de arquivos (não ordenados) pode atrasar as operações de inserção.

Restrições de exclusividade em atributos

- Caminhos de acesso especificados em todos atributos de chave candidata (chave primária ou atributos únicos);
- Um índice é suficiente na verificação de exclusividade
  - Os valores do atributo existirão nos nós folha do índice.

Projeto Físico em SGBDRs

- Cada relação da base é representada como um arquivo de banco de dados;
- As opções do caminho de acesso incluem:
  - Especificação do tipo de organização de arquivo primário, a cada relação, e dos atributos índices.
- A seguir, serão propostas decisões de projeto para definição da indexação adequada.

Decisões de projeto sobre a indexação

- Atributos cujos valores são normalmente exigidos nas condições de igualdade ou intervalo;
  - Exigem caminhos de acesso (índice)
- O desempenho das consultas depende de quais índices existem para agilizar o processamento de seleções e junções;
- Na inserção, exclusão ou atualização, a existência de índices aumenta o *overhead*.
  - Desvantagem justificada pelo ganho de eficiência ao agilizar consultas e transações.

Decisões de projeto sobre a indexação

- Decisões de projeto físico para indexação:
  1. Se um atributo deve ser indexado:
     - Para criação do índice em um atributo, este deve ser chave (único) ou deve ser usado em uma condição de seleção/junção.
  2. Que atributo ou atributos indexar?
     - Índice construído em um único atributo ou em um conjunto de atributos (se for composto).
     - Índice multiatributo se vários atributos de uma relação estiverem envolvidos em várias consultas.

Decisões de projeto sobre a indexação

- Decisões de projeto físico para indexação:
  3. Se um índice agrupado deve ser montado:
     - Índice primário ou de agrupamento - um por tabela (máximo);
     - Se exigir vários índices, a decisão depende da necessidade de manter a tabela ordenada neste atributo;
     - Não utilizar o agrupamento se o resultado da consulta tiver de ser respondida somente a partir de um índice.
  - As consultas de intervalo se beneficiam com o agrupamento.

Decisões de projeto sobre a indexação

- Decisões de projeto físico para indexação:
  4. Se um índice de hash deve ser usado em um índice de árvore:
     - Funcionam bem com condições de igualdade;
     - Não admitem consultas de intervalo.
  5. Se o hashing dinâmico deve ser usado para o arquivo:
     - Vantagem de utilização nos casos de arquivos voláteis (aumentam e diminuem com frequência).

Como criar um índice

- Comando básico (genérico) para criação de índice:

```
CREATE [ UNIQUE ] INDEX <nome indice>
ON <nome tabela> ( <nome coluna> [ <ordem> ]
{ , <nome coluna> [ <ordem> ] } ) [ CLUSTER ];
```

  - **CLUSTER**: Quando o índice deve classificar os registros do arquivo de dados pelo atributo de indexação.
  - **ORDEM**: *ASC* (ascendente - padrão) ou *DESC* (descentente) - método de ordenação do arquivo de dados a partir do atributo de indexação.

    * UNIQUE e CLUSTER são comandos opcionais.

Como criar um índice

- Exemplo:
  - Criação de índice de agrupamento (crescente) no atributo não chave Dnr do arquivo FUNCIONARIO:

```
CREATE INDEX Idx_Dnr
ON FUNCIONARIO (Dnr)
CLUSTER;
```

Desnormalização para agilizar o processamento de consultas

- **Desnormalização**: armazena o projeto lógico em uma forma normal fraca (1FN ou 2FN);
- Otimiza a performance nas consultas e transações mais frequentes;
- Ex: Inclusão de atributos da tabela S em outra tabela R - reduz a junção de R com S para consultas frequentes;
  - Evita a operação de junção de R com S.
- Introduz a redundância nas tabelas da base (e todos seus problemas decorrêntes).

Desnormalização para agilizar o processamento de consultas

TAREFA (Func_id, Proj_id, Nome_func, Cargo_func, Porc_atribuida, Nome_proj, Id_ger_proj, Nome_ger_proj)

- Corresponde a lista de tarefas de um funcionário.
- Se encontra somente na 1FN:

Proj_id $\longrightarrow$ Nome_proj , Id_ger_proj
Id_ger_proj $\longrightarrow$ Nome_ger_proj
Func_id $\longrightarrow$ Nome_func , Cargo_func

Desnormalização para agilizar o processamento de consultas

- Para este caso, pode ser preferida ao projeto da 2FN:

FUNC (Func_id, Nome_func, Cargo_func)
PROJ (Proj_id, Nome_proj, Id_ger_proj)
FUNC_PROJ (Func_id, Proj_id, Porc_atribuida)

- Na extração da lista de tarefas do funcionário, utiliza-se:

$A \longleftarrow FUNC\_PROJ \times FUNC$
$B \longleftarrow A \times PROJ$
$C \longleftarrow B \bowtie_{PROJ.Id\_ger\_proj=FUNC.Id\_func} FUNC$

Desnormalização para agilizar o processamento de consultas

- É possível, também, criar uma visão para a tabela TAREFA:
  - O usuário evita especificar as junções;
  - Se for materializada:
    As junções são evitadas.
  - Senão (visão virtual sem criação de arquivo materializado):
    Os cálculos das junções continuam sendo necessários.

Sintonia em BDs

- Atividade que monitora e revisa o projeto físico do BD.
- É um ajuste contínuo do projeto físico.
- As mesmas decisões de projeto são revisadas durante o ajuste do BD.

Objetivos da sintonia em BDs

- Fazer as aplicações rodarem mais rapidamente.
- Melhorar (reduzir) o tempo de resposta de consultas e transações.
- Melhorar o desempenho geral das transações.

Estatísticas coletadas internamente pelos SGBDRs

- Tamanhos de tabelas
- Número de valores distintos em uma coluna
- Número de vezes que uma consulta ou transação é submetida e executada
- Tempos exigidos para diferentes fases do processamento de consulta e transação

Estatísticas obtidas pelo monitoramento:

- **Estatísticas de armazenamento**
  - Dados sobre alocação de armazenamento em espaços de tabela (tablespaces), espaços de índices e pools de buffer.
- **Estatísticas de desempenho de E/S e dispositivo**
  - Atividade total de leitura/gravação (paginação) em extensões de disco e "hot spots" do disco.
- **Estatísticas de processamento de consulta/transação**
  - Tempos de execução e tempos de otimização.

Estatísticas obtidas pelo monitoramento:

- **Estatísticas relacionadas a bloqueio/logging**
  - Taxas de emissão de diferentes tipos de bloqueios, taxas de vazão da transação e atividade de registros de log
- **Estatísticas de índice**
  - Número de níveis em um índice, número de páginas folha não contíguas

Sintonia em SGBDRs

A sintonia de um BD envolve os seguintes problemas:

- Evitar disputa excessiva por bloqueio, aumentando a concorrência entre transações.
- Minimizar o *overhead* do logging e o dumping desnecessário de dados.
- Otimizar o tamanho do buffer e o escalonamento de processos.
- Alocar discos, RAM e processos para que a utilização seja mais eficiente.

Os DBAs são treinados para lidar com esses problemas de sintonia para o SGBD específico.

Sintonia de Índices

- Motivos:
  - Consultas podem demorar para serem executadas, por falta de um índice.
  - Índices podem nem ser utilizados.
  - Índices podem sofrer muita atualização.

Sintonia de Índices

- Com base na análise da sintonia:
  - Alguns índices podem ser removidos e novos podem ser criados.
  - Um índice não agrupado pode ser alterado para um índice agrupado e vice-versa.
  - Recriação do índice.
- Objetivo da sintonia:
  - Avaliar dinamicamente os requisitos.
  - Reorganizar índices e organizações de arquivo.
  - Gerar melhor desempenho geral.

Sintonia de Índices

- A remoção e criação de novos índices é um *overhead*.
- A atualização de uma tabela em geral é suspensa enquanto um índice é descartado ou criado.

Sintonia de Índices

- Recriação do índice
  - Pode melhorar o desempenho.
  - A maioria dos SGBDRs utiliza *B+-trees* para um índice.
  - Muitas exclusões na chave de índice → páginas de índice com espaço desperdiçado.
  - Muitas inserções → causam estouros em um índice agrupado.
  - **A recriação de um índice agrupado significa reorganizar a tabela ordenada nessa chave.**

Sintonia de Índices

- As opções disponíveis para indexação varia de um sistema para outro.
- Um **índice esparso**, como um índice primário, terá um ponteiro de índice para cada página (bloco de disco) no arquivo de dados.
- Um **índice denso**, como um índice secundário, terá um ponteiro de índice para cada registro.

Sintonia de Índices

- Sybase
  - Oferece índices de agrupamento como índices esparsos na forma de *B+-trees*.
- INGRES
  - Oferece índices de agrupamento esparsos como arquivos ISAM.
  - E índices de agrupamento densos como *B+-trees*.
- Oracle e DB2
  - A opção de configurar um índice de agrupamento é limitada a um índice denso (com muito mais entradas de índice).

Sintonia de Projeto

- Se um projeto físico de BD não atender aos objetivos, o DBA pode reverter para o projeto lógico e mapeá-lo para um novo conjunto de tabelas físicas e índices.
- O projeto do BD precisa ser controlado pelos requisitos de processamento e pelos requisitos de dados.
- Se os requisitos de processamento estiverem mudando, o projeto realiza mudanças no esquema conceitual, para refletir no esquema lógico e projeto físico.
- Ajustes são projetados para atender ao grande volume de consultas ou transações, com ou sem sacrificar as formas normais.

Sintonia de Projeto

- Natureza das mudanças:
  - Para determinado conjunto de tabelas, pode haver escolhas de projeto alternativas, todas alcançando a 3FN ou FNBC.
  - Um projeto normalizado pode ser substituído por outro.

Sintonia de Projeto

- Natureza das mudanças:
  - Uma relação **R(Ch, A, B, C, D, ...)**, com **Ch** como um conjunto de atributos de chave (que está na FNBC).
  - Pode ser armazenada em várias tabelas que também estão na FNBC, ao replicar a chave **Ch**: **R1(Ch, A, B)**, **R2(Ch, C, D)**, **R3(Ch, ...)**
  - Tal processo é conhecido como **particionamento vertical**.
  - Cada tabela agrupa conjuntos de atributos que são acessados juntos.
  - O(s) atributo(s) de uma tabela pode(m) ser repetido(s) em outra, embora isso crie redundância e uma anomalia em potencial.

Sintonia de Projeto

- Natureza das mudanças:
  - O **particionamento horizontal** pega fatias horizontais de uma tabela e as armazena como tabelas distintas.
  - Cada tabela tem o mesmo conjunto de colunas (atributos), mas contém um conjunto distinto de produtos (tuplas).
  - Se uma consulta/transação se aplica a todos os dados do produto, ela pode ter de ser executada novamente contra todas as tabelas e os resultados podem ter de ser combinados.

Sintonia de Consultas

- Natureza das mudanças:
  - Indicações que sugerem que o ajuste da consulta seja necessário:

1. Uma consulta emite muitos acessos ao disco.
2. O plano de consulta mostra que índices relevantes não estão sendo usados.

Sintonia de Consultas

- Natureza das mudanças:
  - Casos típicos de situações que precisam de ajuste de consulta:

1. Muitos otimizadores de consulta não usam índices na presença de expressões numéricas, comparações numéricas de diferentes tipos, comparações NULL e comparações de substring.

Sintonia de Consultas

- Indicações que sugerem que o ajuste da consulta seja necessário:
  2. Índices não costumam ser usados para consultas aninhadas usando **IN**.

```
SELECT Cpf
FROM FUNCIONARIO
WHERE Dnr IN (
SELECT Dnumero FROM DEPARTAMENTO
WHERE Cpf_ger = '33344555587');
```

Esta consulta pode não usar o índice em **Dnr** em **FUNCIONARIO**, enquanto o uso de **Dnr** = **Dnumero** pode fazer que o índice seja utilizado.

Sintonia de Consultas

- Natureza das mudanças:
  - Indicações que sugerem que o ajuste da consulta seja necessário:

3 **DISTINCTs** podem ser redundantes e ser evitados sem alterar o resultado.

   Um **DISTINCT** normalmente causa uma operação de ordenação e deve ser evitado ao máximo possível.

Sintonia de Consultas

- Natureza das mudanças:
  - Indicações que sugerem que o ajuste da consulta seja necessário:

4 O uso desnecessário de tabelas de resultado temporárias pode ser evitado ao se reduzir consultas múltiplas em uma única consulta.

A menos que a relação temporária seja necessária para algum processamento intermediário.

Sintonia de Consultas

- Natureza das mudanças:
  - Indicações que sugerem que o ajuste da consulta seja necessário:

5 Em algumas situações com consultas correlacionadas, os temporários são úteis.

```
SELECT Cpf
FROM FUNCIONARIO
WHERE Dnr IN (
SELECT Dnumero FROM DEPARTAMENTO
WHERE Cpf_ger = '33344555587');
```

Essa consulta de pesquisar toda a tabela **FUNCIONARIO** interna **M** em busca de cada tupla da tabela **FUNCIONARIO** externa **F**.

Sintonia de Consultas

- Natureza das mudanças:
  - Indicações que sugerem que o ajuste da consulta seja necessário:

5 Para tornar a execução mais eficiente, o processo pode ser desmembrando em duas consultas.

```
SELECT MAX (Salario) AS Salario_alto, Dnr INTO TEMP
FROM FUNCIONARIO
GROUP BY Dnr;
SELECT Cpf.FUNCIONARIO
FROM FUNCIONARIO, TEMP
WHERE FUNCIONARIO.Salario = TEMP.Salario_alto
AND FUNCIONARIO.Dnr = TEMPO.Dnr;
```

Sintonia de Consultas

- Natureza das mudanças:
  - Indicações que sugerem que o ajuste da consulta seja necessário:

6 Escolha uma condição de junção que utilize um índice de agrupamento e evite as com comparações de string.

Sintonia de Consultas

- Natureza das mudanças:
  - Indicações que sugerem que o ajuste da consulta seja necessário:

7 A ordem das tabelas na cláusula FROM pode afetar o processamento da junção.

Pode ser preciso trocar a ordem para que a menor das duas relações seja varrida e a relação maior seja usada com um índice apropriado.

Sintonia de Consultas

- Natureza das mudanças:
  - Indicações que sugerem que o ajuste da consulta seja necessário:

  8 Alguns otimizadores de consulta funcionam pior em consultas aninhadas em comparação com não aninhadas. Tipos de consultas aninhadas:

    - Subconsultas não correlacionadas com agregações em uma consulta interna.
    - Subconsultas não correlacionadas sem agregações.
    - Subconsultas correlacionadas com agregações em uma consulta interna.
    - Subconsultas correlacionadas sem agregações.

Sintonia de Consultas

- Natureza das mudanças:
  - Indicações que sugerem que o ajuste da consulta seja necessário:

9 Muitas aplicações são baseadas em visões que definem os dados de interesse. As vezes, essas visões se tornam exageradas porque uma consulta pode ser proposta contra uma tabela da base, em vez de passar por uma visão que é definida por uma JUNÇÃO.

Orientações adicionais de sintonia de consulta

1. Uma consulta com múltiplas condições de seleção que são conectadas por OR pode ser dividida e expressa como uma união de consultas.

```
SELECT Pnome, Unome, Salario, Idade
FROM FUNCIONARIO
WHERE Idade > 45 OR Salario < 50.000;
```

   - Dividindo-o desta maneira, pode utilizar índices em **Idade** e em **Salario**.

```
SELECT Pnome, Unome, Salario, Idade
FROM FUNCIONARIO
WHERE Idade > 45
UNION
SELECT Pnome, Unome, Salario, Idade
FROM FUNCIONARIO
WHERE Salario < 50.000;
```

Orientações adicionais de sintonia de consulta

2 **Para agilizar a consulta:**

- A condição NOT pode ser transformada em uma expressão positiva.
- Blocos SELECT embutidos usando IN, = ALL, = ANY e = SOME podem ser substituídos por junções.
- Se uma junção de igualdade for usada entre duas tabelas, a condição de seleção no atributo de junção usado em uma tabela pode ser repetida para a outra tabela.

Orientações adicionais de sintonia de consulta

2 **Para agilizar a consulta:**

- Condições WHERE podem ser reescritas para utilizar os índices em múltiplas colunas. Exemplo:
- Nessa consulta pode-se usar um índice apenas em **Num_regiao** e pesquisar todas as páginas folha do índice para uma combinação em **Tipo_prod**.

```
SELECT Num_regiao, Tipo_prod, Mes, Vendas
FROM ESTATISTICAS_VENDAS
WHERE Num_regiao = 3 AND ((Tipo_prod BETWEEN 1 AND 3)
OR (Tipo_prod BETWEEN 8 AND 10));
```

Orientações adicionais de sintonia de consulta

2 **Para agilizar a consulta:**

```
SELECT Num_regiao, Tipo_prod, Mes, Vendas
FROM ESTATISTICAS_VENDAS
WHERE (Num_regiao = 3 AND (Tipo_prod BETWEEN 1 AND 3)
OR (Num_regiao = 3 AND (Tipo_prod BETWEEN 8 AND 10));
```

Seção 21

Processamento de Transações

Sistemas de monousuário *versus* multiusuário

- SGDB **monousuário**: um usuário de cada vez pode utilizar o sistema. Restrito a sistemas de computador pessoal.
- SGBD **multiusuário**: muitos usuários podem utilizar o sistema **simultaneamente**.

  São exemplos, banco de dados usados em *bancos, agências de seguros, mercado de ações, supermercados,* etc.

Processamento de transações

Multiprogramação: permite que um sistema operacional execute vários programas (ou **processos**) ao mesmo tempo.

Sistemas operacionais de multiprogramação, para uma CPU:

- Executam alguns comandos de um processo, depois o suspendem e executam alguns comandos do processo seguinte, e assim por diante.
- A execução simultânea do processo é **intercalada**.
- E a intercalação impede que a CPU permaneça ociosa durante o tempo de E/S, e que um processo longo atrase os demais processos.

Se o sistema possuir múltiplos processadores e hardware (CPUs), o **processamento paralelo** é possível.

Sistemas de processamento de transação

- Consistem em sistemas com grandes bancos de dados e centenas de usuários simultâneos executando transações.
- Exigem alta disponibilidade e tempo de resposta rápido para centenas de usuários simultâneos.

Transação

- Programa em execução que forma uma unidade lógica de processamento de banco de dados.
- Inclui uma ou mais operações de acesso ao banco de dados.
- Embutidas em um programa de aplicação ou especificadas por uma linguagem de consulta (SQL).

Processamento de transações

Transação somente de leitura: suas operações não atualizam o banco de dados, apenas recuperam dados.

Transação leitura-gravação: suas operações atualizam o banco de dados.

Transação confirmada (*commited*): suas operações foram concluídas com sucesso e seu efeito é registrado permanentemente no banco de dados.

Transação abortada: suas operações não possuem qualquer efeito no banco de dados ou em quaisquer outras transações.

Processamento de transações

Begin transaction e end transaction

- Especificam os limites de uma transação em um programa de aplicação.
- Um programa de aplicação pode conter mais de uma transação se tiver vários limites.

Conjunto de leitura

- Conjunto de todos os itens que a transação lê.

Conjunto de gravação

- Conjunto de todos os itens que a transação grava.

Itens de banco de dados

- **Coleção de itens de dados nomeados:** representação de um banco de dados.
- **Item de dados:** registro de banco de dados, bloco de disco inteiro ou valor de campo (atributo) individual de algum registro. Cada item possui um nome único.
- **Granularidade:** tamanho de um item de dados.

Operações básicas de uma transação

- ***read_item(x)*:** Lê um item do BD para uma variável do programa.
  1. Ache o endereço do bloco de disco que contém o item X.
  2. Copie esse bloco para um buffer na memória principal.
  3. Copie o item X do buffer para a variável de programa X.

Operações básicas de uma transação

- ***write_item(x)*:** Grava o valor da variável de programa no item de banco de dados.
  1. Ache o endereço do bloco de disco que contém o item *X*.
  2. Copie esse bloco para um buffer na memória principal.
  3. Copie o item *X* da variável de programa *X* para o local correto no buffer.
  4. Armazene o bloco atualizado do buffer de volta no disco.

Controle de concorrência

Problemas encontrados se duas transações forem executadas simultaneamente:

- **O problema da atualização perdida:**

  Ocorre quando duas transações que acessam os mesmos itens de BD têm suas operações intercaladas de modo que isso torna o valor de alguns itens da base de dados incorreto.

- **O problema da atualização temporária (leitura suja):**

  Uma transação atualiza um item do BD e depois a transação falha por algum motivo. Nesse meio tempo, o item atualizado é acessado (lido) por outra transação, antes de ser alterado de volta para seu valor original.

Controle de concorrência

Problemas encontrados se duas transações forem executadas simultaneamente:

- **O problema do resumo incorreto:**

  Se uma transação está calculando uma função de resumo de agregação em itens do BD, enquanto outras transações estão atualizando alguns desses itens, a função de agregação pode calcular valores antes que eles sejam atualizados e outros, depois que eles forem atualizados.

- **O problema da leitura não repetitiva:**

  Uma transação T lê o mesmo item duas e o item é alterado por uma outra transação T' entre as duas leituras. Logo, T recebe valores diferentes para as duas leituras do mesmo item.

Processamento de transações

As falhas são classificadas como falhas de transação, sistema e mídia.

Tipos de falhas:

1. **Falha do computador (falha do sistema):**

   Um erro de hardware, software ou rede no sistema de computação durante a execução da transação.

2. **Erro de transação ou do sistema:**

   Alguma operação na transação pode fazer que esta falhe ou valores de parâmetro errôneos ou erro lógico de programação.

   Além disso, o usuário também pode interromper a transação durante sua execução.

Tipos de falhas:

3. **Erros locais ou condições de exceção detectadas pela transação:**

   Durante a execução da transação, podem ocorrer condições que necessitam de cancelamento. Essa exceção poderia ser programada na própria transação e não seria considerada uma falha.

4. **Imposição de controle de concorrência:**

   O método de controle de concorrência pode abortar uma transação porque ela viola a serialização ou pode abortar uma ou mais transações para resolver um deadlock.

Tipos de falhas:

5. **Falha de disco:**

   Blocos de disco podem perder seus dados devido a um defeito de leitura, gravação ou por causa de uma falha da cabeça de leitura/gravação.

6. **Problemas físicos e catástrofes**:

   Falha de energia, incêndio, roubo, regravação de discos ou fitas por engano e montagem da fita errada pelo operador.

Operações adicionais

O sistema precisa registrar quando cada transação *começa*, *termina* e *confirma* ou *aborta*. O gerenciador de recuperação do SGBD acompanha as seguintes operações:

- ***BEGIN_TRANSACTION*:** Início da execução da transação.
- ***READ* ou *WRITE*:** Operações de leitura ou gravação nos itens de banco de dados de uma transação.
- ***END_TRANSACTION*:** Final da execução da transação. Nesse ponto é necessário verificar se a transação será confirmada ou abortada.

Operações adicionais

O sistema precisa registrar quando cada transação *começa*, *termina* e *confirma* ou *aborta*. O gerenciador de recuperação do SGBD acompanha as seguintes operações:

- ***COMMIT_TRANSACTION*:** Atualizações executadas pela transação podem ser confirmadas (*commited*) ao banco de dados e não serão desfeitas.
- ***ROLLBACK* (ou *ABORT*):** Mudanças ou efeitos que a transação possa ter aplicado ao banco de dados precisam ser desfeitos.

Estados de transação

- **Estado ativo:** após iniciar a execução, onde pode executar suas operações *READ* e *WRITE*.
- **Estado parcialmente confirmado:** transação terminada.
- **Estado confirmado:** alguns protocolos de recuperação precisam garantir que não há falhas.

  Caso não ocorra falhas, a transação alcançou seu ponto de confirmação e todas as suas mudanças precisam ser gravadas permanentemente no BD.

Estados de transação

- **Estado de falha:** se uma das verificações falhar ou se a transação for abortada. A transação pode ter de ser cancelada para desfazer o efeito de suas operações. As transações com falha ou abortadas podem ser reiniciadas depois.
- **Estado terminado:** corresponde a transação que sai do sistema.

Log do sistema

- Arquivo sequencial, apenas para inserção, que é mantido no disco.
- Registra todas as operações de transação que afetam os valores dos itens de banco de dados, bem como outras informações que podem ser necessárias para permitir a recuperação de falhas.
- A noção de recuperação de uma falha de transação equivale a desfazer ou refazer operações de transação individualmente com base no *log*.

Conceitos de transação e sistema

Registros de log:

1. [***start_transaction, T***]**.** Indica que a transação *T* iniciou sua execução.

2. [***write_item, T, X, valor_antigo, valor_novo***]**.** Indica que a transação *T* mudou o valor do item do banco de dados *X* de *valor_antigo* para *valor_novo*.

3. [***read_item, T, X***]**.** Indica que a transação *T* leu o valor do item de banco de dados *X*.

4. [***commit, T***]**.** Indica que a transação *T* foi concluída com sucesso, e afirma que seu efeito pode ser confirmado.

5. [***abort, T***]**.** Indica que a transação *T* foi abortada.

Ponto de confirmação

- Uma transação alcança seu **ponto de confirmação** quando todas as suas operações tiverem sido executadas com sucesso e registradas no *log*.
- É comum manter um ou mais blocos do arquivo de *log* nos buffers da memória principal (*buffer de log*), até que eles sejam preenchidos com entradas de *log* e, depois, gravá-los de volta ao disco apenas uma vez.
- Antes que uma transação alcance seu ponto de confirmação, qualquer parte do *log* que ainda não tenha sido gravada no disco deve agora sê-lo (**gravação forçada**).

Propriedades desejáveis das transações

*Propriedades **ACID** devem ser impostas pelos métodos de controle de concorrência e recuperação do SGBD.*

- **Atomicidade.** Uma transação é uma unidade de processamento atômica; ela deve ser realizada em sua totalidade ou não ser realizada em sua totalidade ou não ser realizada de forma alguma.
- **Durabilidade ou permanência.** As mudanças aplicadas ao banco de dados pela transação confirmada precisam persistir (não devem ser perdidas).

Propriedades desejáveis das transações

*Propriedades **ACID** devem ser impostas pelos métodos de controle de concorrência e recuperação do SGBD.*

- **Preservação da consistência.** Se uma transação for completamente executada do início ao fim sem interferência deve-se levar o banco de dados de um estado consistente para outro.
  - **Estado de bando de dados**: coleção de todos os itens de dados armazenados no banco de dados em determinado período.
  - **Estado consistente:** banco de dados satisfaz as restrições especificadas no esquema.

Propriedades desejáveis das transações

*Propriedades **ACID** devem ser impostas pelos métodos de controle de concorrência e recuperação do SGBD.*

- **Isolamento.** A execução de uma transação não deve ser interferida por quaisquer outras transações que acontecem simultaneamente.
  - **Isolamento nível 0 (zero)**: transação não grava sobre as leituras sujas das transações de nível mais alto.
  - **Isolamento nível 1 (um)**: não possui atualizações perdidas.
  - **Isolamento nível 2 (dois)**: não possui atualizações perdidas ou leituras sujas.
  - **Isolamento nível 3 (três)**: além das propriedades de nível 2, leituras repetitivas.

Escalonamento de transações

Escalonamentos (históricos) de transações

- Ordem da execução das operações que estão executando simultaneamente em um padrão intercalado.
- As operações de uma transação precisam aparecer na mesma ordem em que ocorrem.
- A ordem das operações é uma ordenação total, se para duas operações quaisquer no schedule uma precisa ocorrer antes da outra.

Escalonamentos (históricos) de transações

- Notação para schedule: *begin_transaction* (***b***), *read_item* (***r***), *write_item* (***w***), *end_transaction* (***e***), *commit* (***c***) ou *abort* (***a***).
- E acrescenta como **subscrito** a *id* da transação a cada operação no schedule.

$$S: r_1(X); w_1(X); r_2(X); w_2(X); r_1(Y); a_1$$

- Note que a transação foi cancelada após sua operação *read_item(Y)*.

Escalonamento de transações

- Note que a transação foi cancelada após sua operação *read_item(Y)*.
- Duas operações em um schedule estão em conflito se satisfazerem todas as condições a seguir:
  - Pertencem a diferentes transações;
  - Acessam o mesmo item X;
  - Pelo menos uma das operações é um *write_item(X)*.
- Duas operações estão em conflito se a mudança de sua ordem puder resultar em algo diferente.
  - **Conflito de leitura-gravação**
  - **Conflito de gravação-gravação**

Escalonamento completo

Condições:

1. As operações são exatamente as das transações, incluindo uma operação de confirmação ou cancelamento como última operação em cada transação;

2. Para qualquer par de operações da mesma transação, sua ordem de aparecimento relativa na schedule é a mesma que a ordem de aparecimento na transação;

3. Para duas operações quaisquer em conflito, uma das duas precisa ocorrer antes da outra schedule.

Escalonamento de transações

Escalonamento completo

- Como cada transação é confirmada ou cancelada, um schedule completo **não terá quaisquer transações ativas** ao final do schedule;

- Difícil encontrar schedules completos em um sistema de processamento de transação.
  - Novas transações estão sendo continuamente submetidas ao sistema;
  - Convém definir a **projeção confirmada C(S)**:
    - Inclui apenas as operações em **S** que pertencem a transações confirmadas.

Escalonamento de transações

- Importante caracterizar os tipos de escalonamentos para os quais a recuperação é possível;
- Quando uma transação T é confirmada, nunca deve ser necessário cancelar T.
  - Garante a não violação a propriedade de durabilidade no SGBD.

1. Schedules **recuperáveis**;
2. Schedules **não recuperáveis**;

1. Escalonamentos recuperáveis

- Um schedule **S** é recuperável se nenhuma transação **T** em **S** for confirmada até que todas as transações **T′**, que tiverem gravado algum item **X** que **T** lê, sejam confirmadas;
- Uma transação **T** lê a transação **T′** em um schedule **S** se algum item **X** for gravado primeiro por **T′** e depois lido por **T**;
- **T′** não deve ser cancelado antes que **T** leia o item **X**.

1. Escalonamentos recuperáveis

- Considere o escalonamento S_a' a seguir:

  S_a': $r_1(X)$; $r_2(X)$; $w_1(X)$; $r_1(Y)$; $w_2(X)$; c_2; $w_1(Y)$; c_1;

- S_a' é recuperável, embora sofra do problema da atualização (tratado pela teoria da serialização).

2 Escalonamentos não recuperáveis

- Considere os escalonamentos (parciais) $\mathbf{S_c}$ e $\mathbf{S_d}$ a seguir:

  S_c: $r_1(X)$; $w_1(X)$; $r_2(X)$; $r_1(Y)$; $w_2(X)$; c_2; a_1;

  S_d: $r_1(X)$; $w_1(X)$; $r_2(X)$; $r_1(Y)$; $w_2(X)$; $w_1(Y)$; c_1; c_2;

  S_c: $r_1(X)$; $w_1(X)$; $r_2(X)$; $r_1(Y)$; $w_2(X)$; $w_1(Y)$; a_1; a_2;

- $\mathbf{S_c}$ não é recuperável porque $\mathbf{T_2}$ lê o item $\mathbf{X}$ de $\mathbf{T_1}$, mas $\mathbf{T_2}$ confirma antes que $\mathbf{T_1}$ confirme.

- O problema ocorre se $\mathbf{T_1}$ abortar depois da operação $\mathbf{c_2}$ em $\mathbf{S_c}$, então o valor de X que $\mathbf{T_2}$ lê não é mais válido e $\mathbf{T_2}$ precisa ser abortado **depois** de ser confirmado, levando a um schedule que **não é recuperável**.

Escalonamento de transações

- Em um escalonamento recuperável, nenhuma transação confirmada precisa ser cancelada.
  - A definição de transação confirmada como durável não é violada;
  - Possibilidade do **rollback em cascata**:
    - Uma transação não confirmada é cancelada porque leu um item de uma transação que falhou;
    - Exemplificado em S_c - a transação R_2 foi cancelada porque leu o item X de T_1, e T_1 então foi cancelada.

Escalonamento de transações

- **Escalonamento sem cascata** (evita rollback em cascata)
  - Cada transação nele ler apenas itens que tenham sido gravadas por transações;
  - Todos os itens lidos não serão descartados;
  - Neste caso, o comando $r_2(X)$ nos schedules S_d e S_c devem ser adiados até depois que T_1 tiver sido confirmada.

Escalonamento de transações

- **Escalonamento estrito**
  - As transações não podem ler nem gravar um item X até que a última transação que gravou X tenha sido confirmada (ou cancelada);
  - Simplificam o processo de recuperação;
  - O processo de desfazer uma operação **write_item(X)** de uma transação abortada serve apenas para restaurar a **imagem anterior** do item de dados X;
  - Pode não funcionar para schedules recuperáveis ou sem cascata.

Escalonamento de transações

- A seguir os tipos de escalonamentos serão caracterizados corretos quando transações concorrentes estão sendo executadas.
  - **Escalonamentos serializáveis.**
- Suponha que dois usuários submetam as transações no SGBD T_1 e T_2 :

T_1
read_item(X);
X := X-N;
write_item(X);
read_item(Y);
Y := Y+N;
write_item(X);

T_2
read_item(X);
X := X+M;
write_item(X);

Escalonamento de transações

- Se nenhuma intercalação de operações for permitida, os resultados possíveis são:

    1. Executar todas as operações de T_1 seguidas por todas as operações de T_2;
    2. Executar todas as operações de T_2 seguidas por todas as operações de T_1.

Escalonamento de transações

- Escalonamentos possíveis:

calonamento A:

T_1	T_2
read_item(X);	
X := X-N;	
write_item(X);	
read_item(Y);	
Y := Y+N;	
write_item(X);	
	read_item(X);
	X := X+M;
	write_item(X);

Escalonamento de transações

- Escalonamentos possíveis:

calonamento B:

T_1	T_2
	read_item(X);
	X := X+M;
	write_item(X);
read_item(X);	
X := X-N;	
write_item(X);	
read_item(Y);	
Y := Y+N;	
write_item(X);	

Escalonamento de transações

- Escalonamentos possíveis:

calonamento C:

T_1	T_2
read_item(X);	
X := X-N;	
	read_item(X);
	X := X+M;
write_item(X);	
read_item(Y);	
	write_item(X);
Y := Y+N;	
write_item(X);	

Escalonamento de transações

- Escalonamentos possíveis:

calonamento D:

T_1	T_2
read_item(X);	
X := X-N;	
write_item(X);	
	read_item(X);
	X := X+M;
	write_item(X);
read_item(Y);	
Y := Y+N;	
write_item(X);	

Escalonamento de transações

- Se a intercalação de operações for permitida, haverá muitas ordens possíveis para execução das operações individuais.
- **Serialização de escalonamentos**
  - Identifica quais escalonamentos estão corretos quando as execuções da transação tiverem intercalação de suas operações nos escalonamentos.

Escalonamento de transações

Escalonamentos seriais

- Os escalonamentos de exemplo **a** e **b** são seriais;
- As operações de cada transação são executadas consecutivamente, sem intercalações;
- Somente uma transação ativa por vez;
  - O commit (ou abort) da transação ativa inicia a execução da próxima transação.
- Desvantagem: Limitam a concorrência - desperdício de tempo de CPU.
  - Os tornam inaceitáveis na prática.

Escalonamentos não seriais

- Os escalonamentos de exemplo **c** e **d** são não seriais;
- Cada sequência intercala operações das duas transações;
- Necessário determinar quais schedules sempre produzem o resultado correto e quais podem gerar resultados errôneos:
  - Processo chamado de **serialização** de um schedule.

Escalonamento de transações

Escalonamento serializável

- Um escalonamento **S** de **n** transações é serializável se for equivalente a algum escalonamento serial das mesmas **n** transações;
- Existem **n!** escalonamentos seriais possíveis de **n** transações e muito mais escalonamentos não seriais possíveis.
- Pode-se formar dois grupos dos escalonamentos não seriais:
  1. Equivalentes a um (ou mais) dos escalonamentos seriais - são serializáveis;
  2. Não são equivalentes a qualquer escalonamento serial - não são serializáveis.

Escalonamento serializável

- Dizer que um escalonamento é serializável quer dizer que ele é correto.

Quando dois escalonamentos são considerados equivalentes?

As operações aplicadas a cada item de dados afetado pelos escalonamentos devem ser aplicadas a esse item nos dois escalonamentos, **na mesma ordem**. Assim, verificaremos se estes **produzirão o mesmo estado final do banco de dados**.

Escalonamento serializável

- Equivalência de conflito
  - Dois escalonamentos são considerados **equivalentes em conflito** se a ordem de duas operações for a mesma nos dois escalonamentos;
  - Escalonamento S é **serializável de conflito** se ele for equivalente (em conflito) a algum escalonamento serial S′;
    - Reordena as operações **não em conflito** em **S** até formar o escalonamento serial equivalente **S′**.

Escalonamento serializável

- Equivalência de conflito
  - O escalonamento **D** é equivalente ao escalonamento serial **A**;
  - O escalonamento **C** não é equivalente a qualquer um dos possíveis escalonamentos seriais **A** e **B**, e, portanto, não é serializável.

Testando a serialização de conflito

- Existe um algoritmo simples para determinar se um escalonamento é serializável de conflito ou não;
- Verifica apenas as operações **read_item** e **write_item** para construir um **grafo de precedência**;
  - Grafo direcionado $G = (N, E)$;
  - Conjunto de nós $N = T_1, T_2, ..., T_N$;
  - Conjunto de arestas direcionadas $A = a_1, a_2, ..., a_n$;
  - Um nó para cada transação T_i no escalonamento.

Testando a serialização de conflito - Algoritmo

1. Para cada transação T_i participante no schedule **S**, crie um nó rotulado T_i no grafo de precedência;

2. Para cada caso em **S** onde T_i executa um **read_item(X)** depois de T_i executar um **write_item(X)**, crie uma aresta T_i - T_j;

3. Para cada caso em **S** onde T_i executa um **write_item(X)** após T_i executar um **read_item(X)**, crie uma aresta T_i - T_j;

Testando a serialização de conflito - Algoritmo

4 Para cada caso em **S** onde T_i executa um **write_item(X)** após T_i executar um **write_item(X)**, crie uma aresta T_i - T_j;

5 O escalonamento **S** é serializável se, e somente se, o grafo de precedência não tiver ciclos.

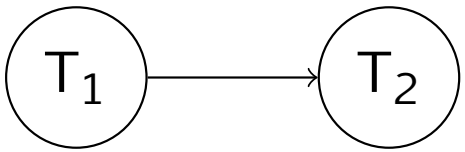
Testando a serialização de conflito - Observações

- Uma aresta de T_i para T_j significa que a transação T_i precisa vir antes da transação T_j em qualquer escalonamento serial que seja equivalente a **S**;
- Vários escalonamentos seriais podem ser equivalentes a **S** se o grafo de precedência para **S** não tiver ciclo;
- Se o grafo tiver ciclo, é fácil mostrar que não pode-se criar qualquer escalonamento serial equivalente, de modo de **S** não é serializável.

Testando a serialização de conflito

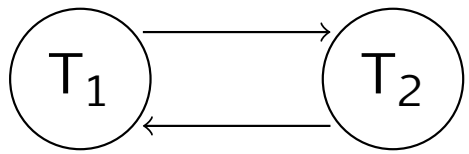
- Grafo de precedência criado para o escalonamento **A**:

- Grafo de precedência criado para o escalonamento **D**:



Testando a serialização de conflito

- Grafo de precedência criado para o escalonamento **C**:



Observe que o grafo para o escalonamento **C** tem um ciclo e, portanto, não é serializável.

Escalonamento de transações

- Pontos de atenção:
  - **Um escalonamento ser serializável é diferente de ser serial.**
  - Um escalonamento serial representa um processamento ineficiente;
  - Um escalonamento serializável oferece os benefícios da execução concorrente sem abrir mão de qualquer exatidão.
- Difícil, na prática, testar a serialização de um escalonamento.
  - Fatores definidos e iniciados pelo Sistema Operacional.

Escalonamento de transações

- Técnica para obtenção de resultado de "teste"da serialização de um escalonamento:
  - Determinação de métodos ou protocolos que garantem a serialização, sem ter de testar os próprios schedules.
- Quando as transações são submetidas, é difícil determinar quando um escalonamento começa e termina.
  - Solução: adaptação da teoria da serialização;
  - Considera-se somente a projeção confirmada de um escalonamento **S**.

Escalonamento de transações

- Adaptação da teoria da serialização:
  - Pode-se definir um um escalonamento **S** para ser serializável se sua projeção confirmada **C(S)** for equivalente a algum escalonamento serial.
    - Apenas transações confirmadas são garantidas pelo SGBD.
  - Técnica de bloqueio de duas fases:
    - Mais comum para controle de concorrência, que garantem a serialização;
    - Consiste no **bloqueio de itens de dados** para impedir que transações concorrentes infiram umas com as outras. Sempre na imposição de uma condição adicional que garanta a serialização.

Equivalência de visão

- Outra definição de equivalência de escalonamentos;
- Dois escalonamentos **S** e **S′** são considerados **equivalentes de visão** se:

1. O mesmo conjunto de transações participa em **S** e **S′**, e **S** e **S′** incluem as mesmas operações dessas transações;

2. Para qualquer operação $\mathbf{r_i(X)}$ de $\mathbf{T_i}$ em **S**, se o valor de X lido pela operação tiver sido gravado por uma operação $\mathbf{w_i(X)}$ de $\mathbf{T_i}$, a mesma condição deve ser mantida para o valor de X lido pela operação $\mathbf{r_i(X)}$ de $\mathbf{T_i}$ em **S′**;

Equivalência de visão

- Outra definição de equivalência de escalonamentos;
- Dois escalonamentos **S** e **S′** são considerados **equivalentes de visão** se:

  3 Se a operação $w_k(Y)$ de T_k for a última operação a gravar o item Y em S, então $w_k(Y)$ de T_k também deve ser a última operação a gravar o tem Y em **S′**.

Equivalência de visão

- Desde que cada operação de leitura leia o resultado da mesma operação de gravação nos dois escalonamentos, as operações de gravação de cada transação devem produzir os mesmos resultados;
- As leituras veem a mesma visão nos dois escalonamentos;
- Um escalonamento **S** é considerado **serializável de visão** se for equivalente de visão a um escalonamento serial.

Suposição de gravação restrita

- Qualquer operação de gravação $w_i(X)$ em T_i é precedida por um $r_i(X)$ em T_i e que o valor gravado por $w_i(X)$ em T_i depende apenas do valor de X lido por $r_i(X)$;
- O cálculo do novo valor de X é uma função $f(X)$ baseada no valor antigo de X lido no banco de dados;
- As definições de **serialização de conflito** e **serialização de visão** se esta condição se mantiver em todas as transações no escalonamento.

Gravação cega

- Gravação em uma transação T em um item X que não depende do valor de X, de modo que não é precedida por uma leitura de X na transação T.

- A **serialização de visão** é menos restrita do que a **serialização de conflito** sob a **suposição de gravação irrestrita**;
- O valor gravado por uma operação $w_i(X)$ em T_i pode ser independente no banco de dados;
- É possível quando as gravações cegas são permitidas.

Suporte para transação em SQL

Transação SQL

- Unidade lógica de trabalho e tem garantias de ser atômica;
- Seu início é feito implicitamente quando instruções SQL são encontradas;
- Seu fim precisa ter uma instrução explícita:
  - Comando **COMMIT** ou **ROLLBACK**.
- Características: (definidas por **SET TRANSACTION**)
  1. Modo de acesso;
  2. Tamanho da área de diagnóstico;
  3. Nível de isolamento.

Suporte para transação em SQL

Transação SQL

1. Modo de acesso
   - **READ WRITE**
     - Modo default (exceto quando o nível de isolamento é READ UNCOMMITTED);
     - Permite a execução de comandos de seleção, atualização, inserção, exclusão e criação.
   - **READ ONLY**
     - Somente para recuperação de dados.

Transação SQL

2 Tamanho da área de diagnóstico (**DIAGNOSTIC SIZE n**)

- Especifica um valor inteiro **n** que indica o número de condições que podem ser mantidas de maneira simultânea na área de diagnóstico;
- Fornecem informações de feedback (erros ou exceções) ao usuário ou programas nas **n** instruções SQL executadas.

Transação SQL

3 Nível de isolamento (**ISOLATION LEVEL**)

- READ UNCOMMITTED
- READ COMMITTED
- SERIALIZABLE (REPEATABLE READ)
  - Modo default;
  - Se uma transação é executada em um nível de isolamento inferior a SERIALIZABLE, uma ou mais violações a seguir podem ocorrer: **Leitura suja**, **Leitura não repetitiva** e **Fantasmas**.

Transação SQL

3.1 Leitura suja

- Uma transação T_1 pode ler a atualização de uma transação T_2, que ainda não foi confirmada;
- Se T_2 falhar e for abortada, T_1 teria lido um valor que não existe e é incorreto.

3.2 Leitura não repetitiva

- Uma transação T_1 pode ler determinado valor de uma tabela;
- Se outra transação T_2 mais tarde atualizar esse valor e T_1 ler o valor novamente, T_1 verá um valor diferente.

Transação SQL

3.3 Fantasmas

- Uma transação T_1 pode ler um conjunto de linhas de uma tabela, talvez com base em alguma condição especificada na cláusula SQL WHERE;
- Se uma transação T_2 inserir uma nova linha que também satisfaça a condição da cláusula WHERE usada em T_1, na tabela usada por T_1;
- Se T_1 for repetida, então T_1 verá um fantasma, uma linha que anteriormente não exista.

Transação SQL

- Violações para diferentes níveis de isolamento:

Tipo de violação			
Nível de Isolamento	Leitura suja	Leitura não repetitiva	Fantasma
READ UNCOMMITTED	Sim	Sim	Sim
READ COMMITTED	Não	Sim	Sim
REPEATABLE READ	Não	Não	Sim
SERIALIZABLE	Não	Não	Não

Seção 22

Controle de Concorrência

Técnicas de controle de concorrência

- As técnicas de controle de concorrência são usadas para garantir a não interferência ou isolamento das transações executas simultaneamente.
- Garantem a **serialização de schedules** usando protocolos de controle de concorrência.
- Tais **protocolos de bloqueio** são utilizados na maioria dos SGBDs.

TÉCNICAS DE CONTROLE DE CONCORRÊNCIA:

TÉCNICAS DE BLOQUEIO EM DUAS FASES PARA CONTROLE DE CONCORRÊNCIA

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios binários:

- Restritivos para fins de controle de concorrência.
- Possuem dois **estados** ou **valores**: *bloqueado* e *desbloqueado*.
- Um bloqueio distinto é associado a cada item do banco de dados.
  - Se **valor do bloqueio é 1**, o item não pode ser acessado por uma operação.
  - Se **valor do bloqueio é 0**, o item pode ser acessado quanto requisitado.

    E o valor do bloqueio é mudado para 1.

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios binários:

- LOCK(X): Valor atual (ou estado) do bloqueio associado ao item X.
  - Se LOCK(X) = **1**, a transição é forçada a esperar.
  - Se LOCK(X) = **0**, a transição bloqueia o item (é configurada como **1**)

    e passa a possuir permissão para acessar o item X.

```
Nome_item_dado, LOCK, Bloqueio_de_transacao
```

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios binários:

- Impõe uma **exclusão mútua** no item de dados:
  - Se uma transação requisita acesso a um item, emite uma operação **lock_item**(X).
    - Se **LOCK**(X) = **1**, a transação deve esperar.
    - Se **LOCK**(X) = **0**, ela é configurada como 1 (o item é **bloqueado**)
      e a transação possui permissão para acessar o item.
  - Se a transação termina de usar um item, emite uma operação **unlock_item**(X),
    - que define LOCK(X) para 0 - ou seja, **desbloqueia** o item, que pode ser acessado por outras transações.

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios binários:

- Regras impostas pelo gerenciador de bloqueio do SGBD:

1. Uma transação precisa emitir a operação *lock_item(X)* antes de operações *read_item(X)* ou *write_item(X)* serem realizadas.
2. Uma transação precisa emitir a operação *unlock_item(X)* após todas as operações *read_item(X)* ou *write_item(X)* serem completadas.
3. Uma transação não emitirá uma operação *lock_item(X)* se já mantiver o bloqueio no item *X*.
4. Uma transação não emitirá uma operação *unlock_item(X)* a menos que ela já mantenha o bloqueio no item *X*.

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios binários:

- Nenhuma intercalação deve ser permitida se uma operação de bloqueio/desbloqueio é iniciada, até que a operação termine ou a transação espere.
- Entre as operações *lock_item(X)* e *unlock_item(X)* na transação, diz-se que ela mantém o **bloqueio no item *X***.
- No máximo uma transação pode manter o bloqueio de um item em particular.

  Assim, duas transações não podem acessar o mesmo item simultaneamente.

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios binários:

- **Tabela de bloqueio**: registros para os itens que o sistema mantém e que estão atualmente bloqueados.
- Os itens que não estão na tabela de bloqueio são considerados desbloqueados.
- O SGBD possui um **subsistema de gerenciador de bloqueio** para registrar e controlar o acesso aos bloqueios.

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios compartilhados/exclusivos:

- Também conhecidos como bloqueios de **modo múltiplo** ou de **leitura/gravação**.
- Utilizados em esquemas de bloqueio de BD práticos.
- Permite que várias transações acessem o mesmo item se todas acessam apenas para *fins de leitura*.

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios compartilhados/exclusivos:

- Se uma transação tiver de gravar um item, ela precisa ter acesso exclusivo.
- *LOCK(X)* possui três estados possíveis:
  - bloqueado para leitura (bloqueado p/ompartilhamento)
  - bloqueado para gravação (bloqueado exclusivo)
  - desbloqueado

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios compartilhados/exclusivos:

- Deve-se registrar o número de transações que mantêm um bloqueio compartilhado (leitura) na tabela de bloqueio.
- O sistema mantém registros de bloqueio somente para os itens bloqueados.
  - Se **LOCK(X) = bloqueado para gravação**, o valor de *Bloqueio_de_transação* é uma única transação.
  - Se **LOCK(X) = bloqueado para leitura**, o valor de *Bloqueio_de_transação* é uma lista de uma ou mais transações.

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios compartilhados/exclusivos:

- Nenhuma intercalação deve ser permitida depois que uma das operações for iniciada até que termine.

```
Nome_item_dado,LOCK,Num_de_leituras, Bloqueio_transacao
```

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios compartilhados/exclusivos:

- Regras impostas pelo gerenciador de bloqueio do SGBD:

1. Uma transação precisa emitir a operação *read_lock(X)* ou *write_lock(X)* antes que qualquer operação *read_item(X)* seja realizada.

2. Uma transação precisa emitir a operação *write_lock(X)* antes que qualquer operação *write_item(X)* seja realizada.

3. Uma transação precisa emitir a operação *unlock(X)* após todas as operações *read_item(X)* e *write_item(X)* serem completadas.

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios compartilhados/exclusivos:

- Regras impostas pelo gerenciador de bloqueio do SGBD:

4. Uma transação não emitirá uma operação *read_lock(X)* se ela já mantiver um bloqueio de leitura (compartilhado) ou um bloqueio de gravação (exclusivo) no item *X*.

5. Uma transação não emitirá uma operação *write_lock(X)* se ela já mantiver um bloqueio de leitura (compartilhado) ou um bloqueio de gravação (exclusivo) no item *X*.

6. Uma transação não emitirá uma operação *unlock(X)* a menos que mantenha um bloqueio de leitura (compartilhado) ou um bloqueio de gravação (exclusivo) no item *X*.

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueio de certificação:

- Uma transação que já mantém um bloqueio no item tem permissão, sob certas condições de **converter** o bloqueio de um estado bloqueado para outro.
  - **Upgrade** (*read-locked* para *write-locked*)
  - **Downgrade** (*write-locked* para *read-locked*)
- A tabela de bloqueios precisa incluir identificadores de transação no registro para cada bloqueio para armazenar a informação sobre quais transações mantém bloqueios no item.

Técnicas de bloqueio em duas fases

Protocolo de bloqueio em duas fases básico (2PL básico)

Bloqueio de certificação

- Todas as operações de bloqueio (*read_lock, write_lock*) precedem a primeira operação de desbloqueio na transação.
  - **Fase de expansão ou crescimento (primeira)**: novos bloqueios em itens podem ser adquiridos, mas nenhum pode ser liberado.
  - **Fase de encolhimento (segunda)**: bloqueios existentes podem ser liberados, mas nenhum novo bloqueio pode ser adquirido.

Técnicas de bloqueio em duas fases

Protocolo de bloqueio em duas fases básico (2PL básico)

Bloqueio de certificação

- Se a conversão de bloqueio for permitida:
  - O *upgrading* de bloqueios ocorre na fase de expansão.
  - O *downgrading* de bloqueios ocorre na fase de encolhimento.

Técnicas de bloqueio em duas fases

Protocolo de bloqueio em duas fases básico (2PL básico)

- Resultados de possíveis schedules seriais de T_1 e T_2:
  - Valores iniciais: X=20, Y=30
  - Schedule serial resultante T_1 seguido por T_2: X=50, Y=80
  - Schedule serial resultante T_2 seguido por T_1: X=70, Y=50

T_1	T_2

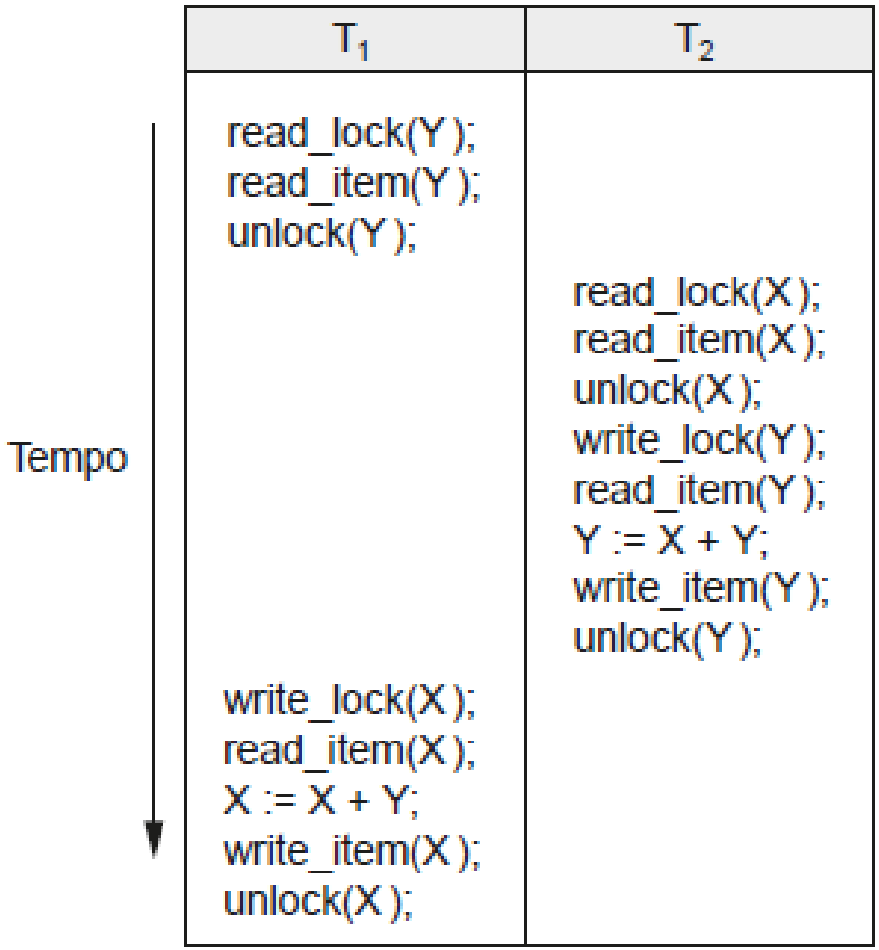
read_lock(Y);
read_item(Y);
unlock(Y);
write_lock(X);
read_item(X);
X := X + Y;
write_item(X);
unlock(X);
	read_lock(X);
	read_item(X);
	unlock(X);
	write_lock(Y);
	read_item(Y);
	Y := X + Y;
	write_item(Y);
	unlock(Y);

Transações que não obedecem ao bloqueio em duas fases.

Técnicas de bloqueio em duas fases

Protocolo de bloqueio em duas fases básico (2PL básico)

- Um schedule não serializável S que usa bloqueios.
  - Resultado de schedule S: X=50, Y=50 (não serializável)



Técnicas de bloqueio em duas fases

Protocolo de bloqueio em duas fases básico (2PL básico)

- O protocolo de bloqueio, ao impor as regras de bloqueio em duas fases, também impõe a serialização.
- Se ***cada*** transação em um schedule seguir o protocolo de bloqueio em duas fases, o schedule é ***garantidamente serializável***.
- O bloqueio em duas fases pode limitar a quantidade de concorrência passível de ocorrer em um schedule.

Técnicas de bloqueio em duas fases

Protocolo de bloqueio em duas fases básico (2PL básico)

- Embora o protocolo de bloqueio em duas fases garanta a serialização, ele não permite ***todos os schedules serializáveis*** possíveis.
- Ou seja, alguns schedules serializáveis serão proibidos pelo protocolo.

Técnicas de bloqueio em duas fases

Protocolo de bloqueio em duas fases conservador

- Variação conhecida como **2PL conservador** (ou **2PL estático**).
- Requer que uma transação bloqueie todos os itens que ela acessa *antes que a transação inicie a execução*, **pré-declarando** seu *conjunto de leitura e de gravação*.
- A transação começa em sua fase de encolhimento.
- Se um dos itens pré-declarados necessários não puder ser bloqueado, a transação espera até que os itens estejam disponíveis para bloqueio.
- É um protocolo livre de deadlock.

Técnicas de bloqueio em duas fases

Protocolo de bloqueio em duas fases estrito (2PL estrito)

- Garante schedules estritos.
- Uma transação não libera nenhum de seus bloqueios exclusivos (gravação) até ***depois*** de confirmar ou abortar.
- Levando a um schedule estrito para facilidade de recuperação.
- O 2PL estrito não é livre de deadlock.

Técnicas de bloqueio em duas fases

Protocolo de bloqueio em duas fases rigoroso (2PL rigoroso)

- Garante schedules estritos.
- Uma transação não libera nenhum de seus bloqueios (exclusivo ou compartilhado) até depois de confirmar ou abortar.
- A transação está em sua fase de expansão até que termine.
- É mais fácil de implementar do que o 2PL estrito.

Técnicas de bloqueio em duas fases

- Em muitos casos, o próprio subsistema de controle de concorrência é responsável por gerar as solicitações *read_lock* e *write_lock*.
- O uso de bloqueios pode causar dois problemas:

  deadlock e **inanição** (starvation)

Deadlock (impasse)

- Conjunto de transações esperando por algum item bloqueado por outra transação do conjunto.
- Ou seja, transações em fila de espera.
- Para lidar com deadlock:
  - Protocolos de prevenção
  - Detecção
  - *Timeouts*

Técnicas de controle de concorrência

VARIAÇÕES DE PROTOCOLOS DE PREVENÇÃO DE DEADLOCK:

1. Requer que cada transação bloqueie **todos os itens que precisar com antecedência**. Se qualquer um dos itens não puder ser obtido, a transação espera e tenta novamente.

2. **Ordena todos os itens** no BD e garante que as transações bloqueiem os itens de acordo com essa ordem.

3. **Rótulo de tempo (*timestamp*) de transação TS(T)**: O rótulo de tempo é um identificador exclusivo para cada transação. É gerado pelo sistema na mesma ordem que os tempos de início de transação.

Se a transação T_1 iniciar antes da transação T_2,
então $TS(T_1) < TS(T_2)$

Protocolos de prevenção de deadlock

- **Protocolos que incluem rótulos de tempo:**
  - **Esperar-morrer (*wait-die*):** Se $TS(T_i) < TS(T_j)$, T_i tem permissão para esperar.

    Caso contrário, aborta T_i e o reinicia posteriormente com o mesmo rótulo de tempo.
  - **Ferir-esperar (*wound-wait*):** Se $TS(T_i) < TS(T_j)$, aborta T_i e o reinicia posteriormente com o mesmo rótulo de tempo.

    Caso contrário, T_i tem permissão para esperar.

Protocolos de prevenção de deadlock

- **Protocolos que incluem algoritmos sem espera e espera cuidadosa:**
  - **Algoritmo sem espera:** Se uma transação for incapaz de obter um bloqueio, ela é abortada e reiniciada após certo atraso de tempo sem verificar se um deadlock ocorrerá ou não.
  - **Algoritmo espera cuidadosa:** Se T_j não estiver bloqueada, então T_i está bloqueada e tem permissão para esperar; caso contrário, aborte T_i.

Detecção de deadlock

Sistema verifica se um estado de deadlock realmente existe.

- *Soluções:*
  - **Grafo de espera**, modo de detectar um estado de deadlock.
  - Um nó é criado para cada transação que está sendo executada
  - Uma aresta é direcionada caso uma transação esteja esperando para bloquear um item que está atualmente bloqueado por uma outra transação.
  - Caso esta transação libere o bloqueio no item, a aresta é removida.
  - Temos um estado de deadlock, se o grafo de espera tiver um ciclo.

Detecção de deadlock

Sistema verifica se um estado de deadlock realmente existe.

Se o sistema estiver em deadlock, algumas das transações que causam este estado precisam ser abortadas.

- *Soluções:*
  - **Algoritmo seleção de vítima**, escolha de quais transações abortar.
  - Evita a seleção de transações que estiveram em execução por muito tempo e que realizam muitas atualizações.
  - Seleciona transações mais novas.

Timeouts (tempo-limite)

- Se uma transação esperar por um período maior que o *timeout,* o sistema pressupõe que a transação pode entrar em deadlock e a aborta.
- Baixo overhead e simples.

Inanição (starvation)

Acontece quando uma transação não pode prosseguir por um período indefinido, enquanto outras continuam normalmente (bloqueio).

- *Soluções:*
  - Possuir um esquema de espera justo, como o uso de uma fila **primeiro-a-chegar-primeiro-a-ser-atendido**;

    as transações bloqueiam um item na ordem em que solicitaram o bloqueio originalmente.

  - Aumentar a prioridade de uma transação quanto mais tempo ela esperar.

Inanição (starvation)

A inanição também pode ocorrer por causa da seleção de vítima se o algoritmo selecionar a mesma transação como vítima repetidamente.

- *Soluções:*
  - O algoritmo pode usar prioridades maiores para transações que tiverem sido abordadas várias vezes.
  - Os esquemas **esperar-morrer** e **ferir-esperar** evitam a inanição.

TÉCNICAS DE CONTROLE DE CONCORRÊNCIA:

Controle de concorrência por ordenação de rótulo de tempo

Controle de Concorrência - Rótulo de tempo

- O uso de bloqueios, combinado com o protocolo 2PL, garante e serialização de schedules;
- Schedules serializáveis (2PL) têm seus schedules seriais equivalentes com base na ordem em que as transações em execução bloqueiam os itens adquiridos;
- Se uma transação precisar de um item que está bloqueado.
  - É forçada a esperar até que o item seja liberado.

Controle de Concorrência - Rótulo de tempo

- Por conta de deadlocks, algumas transações podem ser abortadas e reiniciadas;
- Outra técnica de serialização:
  - Envolve o uso de rótulos de tempo de transação para ordenar a execução da transação para um schedule serial equivalente.

Controle de Concorrência - Rótulo de tempo

Rótulos de tempo (timestamp)

TS(T) - Rótulo de tempo de uma transação

- Identificador exclusivo criado pelo SGBD para **identificar uma transação**;
- Os valores são atribuídos na ordem em que as transações são submetidas ao sistema;
  - Hora de início da transação.
- As técnicas de controle de concorrência baseadas na ordenação por rótulo de tempo **não usam bloqueios**;
  - Deadlocks não podem ocorrer.

Rótulos de tempo (timestamp)

- Podem ser gerados de várias maneiras. Abaixo, duas alternativas:

  1. Utilizar um contador que é incrementado toda vez que seu valor é atribuído a uma transação;
     - Tem um valor máximo finito - o sistema precisa reiniciar o contador periodicamente.
  2. Usar o valor atual de data/hora do clock do sistema;
     - Deve-se garantir que dois valores de rótulo de tempo sejam gerados durante a mesma batida do clock.

Rótulos de tempo (timestamp) - algoritmo de ordenação

- Ordena as transações com base em seus rótulos de tempo;
- **Ordenação de rótulo de tempo (TO):**
  - Um schedule em que as transações participam é serializável e o **único schedule serial equivalete permitido** tem as transações na ordem de seus valores de rótulo de tempo;
  - O schedule é equivalente à **ordem serial em particular** correspondente à ordem dos rótulos de tempo da transação.

Rótulos de tempo (timestamp) - algoritmo de ordenação

- Para cada item acessado pelas **operações em conflito** no schedule, a ordem em que o item é acessado não deve violar a ordem do rótulo de tempo;
- O algoritmo associa a cada item X do banco de dados dois valores de rótulo de tempo (TS):
  1. **Read_TS(X)**
     - O rótulo de tempo de leitura do **item X** é o maior entre todos os rótulos de tempo das transações que leram com sucesso o **item X**;
     - **read_TS(X) = TS(T)**, onde T é a transação mais recente que leu X com sucesso.

Rótulos de tempo (timestamp) - algoritmo de ordenação

- O algoritmo associa a cada item X do banco de dados dois valores de rótulo de tempo (TS):

  2 **Write_TS(X)**

  - O rótulo de tempo de gravação do **item X** é o maior de todos os rótulos de tempo das transações que gravarem com sucesso o **item X**;
  - **write_TS(X) = TS(T)**, onde T é a transação mais recente que gravou X com sucesso.

Rótulos de tempo (timestamp) - algoritmo de ordenação

- Sempre que alguma transação T tenta emitir uma operação **read_item(X)** ou **write_item(X)**:
  - O algoritmo **TO básico** compara o rótulo de tempo de T com **read_TS(X)** e **write_TS(X)** para garantir que a ordem do rótulo não seja violada;
- Se a ordem for violada:
  - A transação T é abortada e submetida novamente com um novo rótulo de tempo.

Rótulos de tempo (timestamp) - algoritmo de ordenação

- Se T for abortada e revertida:
  - Qualquer transação T_1 que possa ter usado um valor gravado por T também precisa ser revertida.
- O efeito acima é chamado de **Propagação de cancelamento** ou **Rollback em cascata**;
  - Os schedules produzidos não têm garantias de serem recuperáveis.
- Nessário adicionar um **protocolo adicional** para garantir que os schedules sejam recuperáveis, sem cascata ou estritos.

Rótulos de tempo (timestamp) - algoritmo de ordenação

- O algoritmo de controle de concorrência deve verificar se as operações em conflito violam a ordenação em rótulo de tempo nos dois casos a seguir:

1. Sempre que uma transação T emitir uma operação **write_item(X)**, deve ser verificado:
   - Se **read_TS(X)** > **TS(T)** ou se **write_TS(X)** > **TS(T)**, aborte e reverta T e rejeite a operação;
   - Se a condição acima não ocorrer, execute a operação **write_item(X)** de T e defina **write_TS(X)** como **TS(T)**.

Rótulos de tempo (timestamp) - algoritmo de ordenação

- O algoritmo de controle de concorrência deve verificar se as operações em conflito violam a ordenação em rótulo de tempo nos dois casos a seguir:

  2. Sempre que uma transação T emitir a operação **read_item(X)**, o seguinte deve ser verificado:

     - Se **write_TS(X) > TS(T)**, então aborte e reverta T e rejeite a operação;
     - Se **write_TS(X) <= TS(T)**, então execute a operação **read_item(X)** de T e defina **read_TS(X)** como o **maior** de **TS(T)** e o **read_TS(X)** atual.

Rótulos de tempo (timestamp) - algoritmo de ordenação

- Sempre que o algoritmo de **TO básico** detectar duas operações em conflito que ocorrem na ordem incorreta:
  - Rejeita uma das duas, abortando a transação que a emitiu.
- Os schedules gerados pela **TO básica** têm garantias de serem **serializáveis por conflito**;
- Alguns schedules são possíveis sub um protocolo que não são permitidos sob o outro.
  - Nenhum protocolo permite todos os schedules serializáveis possíveis

Rótulos de tempo (timestamp) - algoritmo de ordenação

- **TO estrita**
  - Garante que os schedules sejam tanto **estritos** (maior facilidade de recuperação) quanto serializáveis (conflito);
  - Uma transação T emite um **read_item(X)** ou **write_item(X)** tal que **TS(T) > write_TS(X)** tenha sua operação adiada até que a transação T′ que gravou o valor de X (portanto, **TS(T′) = write_TS(X)**) tenha sido confirmada ou abortada;
  - Não causa deadlock, pois T espera por T′ somente se **TS(T) > TS(T′)**.

Rótulos de tempo (timestamp) - algoritmo de ordenação

- **Regras da gravação de Thomas**
  - Modificação do **TO básico**;
  - Não impõe a serialização por conflito, mas rejeita menos operações de gravação ao modificar as verificações para a operação **write_item(X)** da seguinte forma:
    a. Se **read_TS(X)** > **TS(T)**, então aborte e reverta **T**, e rejeite a operação;
    b. Se **write_TS(X)** > **TS(T)**, então não execute a operação de gravação, mas continue processando;
    c. Se nenhuma condição em **(a)** nem em **(b)** acontecer, então execute a operação **write_item(X)** de **T** e defina **write_TS(X)** para **TS(T)**.

TÉCNICAS DE CONTROLE DE CONCORRÊNCIA:

Técnicas para controle de concorrência multiversão

Controle de concorrência multiversão

- Protocolos para controle de concorrência que mantêm os valores antigos de um item de dados quando este é atualizado;
- Quando uma transação requer acesso a um item, uma versão apropriada é escolhida para manter a serialização do schedule atualmente em execução;
- Necessidade de mais armazenamento para manter várias versões dos itens no BD;
  - As versões mais antigas podem ser mantidas de qualquer forma.

Controle de concorrência multiversão

- Algumas aplicações de BD exigem que versões mais antigas sejam mantidas como histórico da evolução de valores;
- Banco de dados temporal (caso extremo)
  - Registra todas as mudanças e os momentos em que elas ocorreram;
  - Não existe penalidade de armazenamento adicional.
- Além dos esquemas de controle de concorrência, o **método de controle de concorrência de validação** também mantém múltiplas versões.

Multiversão baseada na ordenação de rótulo de tempo

- Para cada versão, o valor da versão X_i e os dois rótulos de tempo são mantidos:

1. **Read_TS(X_i)**
   - O rótulo de tempo de leitura de X_i é o maior de todos os rótulos de tempo de transações que leram a versão X_i com sucesso.

2. **Write_TS(X_i)**
   - O rótulo de tempo de gravação de X_i é o rótulo de tempo da transação que gravou o valor da versão X_i.

Controle de concorrência multiversão

Multiversão baseada na ordenação de rótulo de tempo

- Sempre que uma transação T tem permissão para executar uma operação **write_item**(X), uma nova versão X_{k+1} do item X é criada, e tanto **write_TS**(X_{k+1}) quanto **read_TS**(X_{k+1}) são definidos como TS(T);

- Regras usadas para garantir a serialização:

1) Se a transação T emitir uma operação **write_item**(X) e a versão **i** de X tiver o **write_TS**(X_i) mais alto de todas as versões de X <= TS(T) e **read_TS**(X_i) > TS(T):
    - Aborte e retroceda a transação T.

Senão Crie uma nova versão X_i de X com **read_TS**(X_i) = **write_TS**(X_i) = TS(T).

Multiversão baseada na ordenação de rótulo de tempo

- Regras usadas para garantir a serialização:

  2) Se a transação T emitir uma operação **read_item**(X):

     - Determine a versão **i** de X que tem o **write_TS**(X_i) mais alto de todas as versões de X <= **a** TS(T); depois, retorne o valor de X_i à transação T e defina o valor de **read_TS**(X_i) ao maior de TS(T) e o **read_TS**(X_i) atual.

- Para garantir a facilidade de recuperação:

  - Uma transação T não deve ter permissão para confirmar até que todas as transações que gravam alguma versão que T leu tenha sido confirmadas.

Multiversão baseada na ordenação de rótulo de tempo

- Em modo múltiplo, existem três tipos de bloqueio para um item:
  - **Leitura, gravação** e **certificação**.
- O estado de LOCK(X) para um item X pode ser um dentre bloqueado para leitura, bloqueado para gravação, bloqueado para certificação ou desbloqueado;
- No esquema de **bloqueio padrão**, com apenas bloqueios de leitura e gravação, um bloqueio de gravação é um bloqueio exclusivo.

Multiversão baseada na ordenação de rótulo de tempo

- Tabela de compatibilidade de bloqueio:

	Leitura	Gravação
Leitura	Sim	Não
Gravação	Não	Não

Tabela de compatibilidade para o esquema de bloqueio leitura/gravação.

Multiversão baseada na ordenação de rótulo de tempo

- A ideia do **2PL** multiversão é permitir que outras transações T leiam um item X enquanto uma única transação T mantém o bloqueio de gravação em X.
  - Custo: uma transação pode ter de esperar sua confirmação até que obtenha bloqueios de certificação exclusivos em todos os itens que atualizou;
  - Evita a propagação de abortos;
  - Podem ocorrer deadlocks se o upgrading de um bloqueio de leitura para um bloqueio de gravação for permitido.

Multiversão baseada na ordenação de rótulo de tempo

- Tabela de compatibilidade de bloqueio:

	Leitura	Gravação	Certificação
Leitura	Sim	Sim	Não
Gravação	Sim	Não	Não
Certificação	Não	Não	Não

Tabela de compatibilidade para o esquema de bloqueio leitura/gravação/certificação.

TÉCNICAS DE CONTROLE DE CONCORRÊNCIA:

Técnicas de controle de concorrência de validação (otimista)

Controle de concorrência de validação (otimista)

- Também conhecidas como **técnicas de validação** ou **certificação**;
- **Nenhuma verificação** é feita enquanto a transação está executando;
- Vários métodos teóricos são baseados nesta técnica de validação - o esquema será apresentado a seguir;
- As atualizações na transação **não são** aplicadas diretamente aos itens do BD até que a transação alcance seu final;
  - Cópias locais dos itens de dados

Controle de concorrência de validação (otimista)

- Ideia geral do método:
  - Realizar todas as verificações ao mesmo tempo;
  - A execução da transação prossegue com um mínimo de overhead até que a fase de validação seja alcançada.
- Utiliza rótulos de tempo;
- Requer que os **write_sets** e **reads_sets** das transações sejam mantidas pelo sistema;
- Tempos de **início** e **fim** para algumas das três fases precisam ser mantidas para cada transação.

Controle de concorrência de validação (otimista)

- Fases para este protocolo de controle de concorrência:

1. Fase de leitura:
   - Uma transação pode ler valores dos itens de dados confirmados com base no BD;
   - As atualizações são aplicadas a cópias locais.

2. Fase de validação:
   - Verificação realizada para garantir que a serialização não será violada;
   - Confere se as atualizações da transação foram aplicadas ao BD.

3. Fase de gravação:
   - Somente se a fase da validação for bem-sucedida;
   - Atualizações da transação são aplicadas ao BD.

Controle de concorrência de validação (otimista)

- A fase de validação para T_i verifica se, para **cada** transação T_j que é confirmada ou está em sua fase de validação, **uma** das seguintes condições é mantida:

  1. A transação T_j completa sua fase de gravação antes que T_i inicie sua fase de leitura;

  2. T_i inicia sua fase de gravação depois que T_j completa sua fase de gravação, e o **read_set** de T_i não tem itens em comum com o **write_set** de T_j;

  3. Tanto o **read_set** quanto o **write_set** de T_i não têm itens em comum com o **write_set** de T_j, e T_j completa sua fase de leitura antes que T_i o faça.

TÉCNICAS DE CONTROLE DE CONCORRÊNCIA:

Granularidade de itens e bloqueio de granularidade múltiplo

Granularidade de itens e bloqueio múltiplo

- Todas as técnicas de concorrência consideram que o banco de dados é formado por uma série de itens de dados nomeados;
- Um item de BD poderia ser escolhido como sendo um dos seguintes:
  - Um registro de BD;
  - Um valor de campo de um registro do BD;
  - Um bloco de disco;
  - Um arquivo inteiro;
  - Um banco de dados inteiro.
- A granularidade pode afetar o desempenho do controle de concorrência e recuperação.

Granularidade de itens e bloqueio múltiplo

Granularidade do item de dados - bloqueio

- Tamanho dos itens de dados;
- **Granularidade fina** refere-se a tamanhos de item pequenos;
- **Granularidade grossa** refere-se a tamanhos de itens grandes.

A seguir, será apresentada a discussão do tamanho do item de dados no contexto do bloqueio:

- Quanto maior o tamanho do item de dados, menor o grau de concorrência permitido, e vice-versa;

Granularidade de itens e bloqueio múltiplo

Granularidade do item de dados - bloqueio

- O sistema terá um grande número de bloqueios ativos para serem tratados pelo gerenciador de bloqueios
  - Cada transação está associada a um bloqueio.
- Quanto mais operações de bloqueio/desbloqueio, maior o overhead.
  - Também necessário mais espaço de armazenamento para a tabela de bloqueio.

Granularidade de itens e bloqueio múltiplo

Granularidade do item de dados - bloqueio

- Para os rótulos de tempo, o armazenamento é exigido para o **read_TS** e **write_TS** para cada item de dados
  - Haverá um overhead semelhante para o tratamento de um grande número de itens.

Qual o melhor tamanho de item?

Depende dos tipos de transações envolvidas.

- Se uma transação típica acessar um pequeno número de registros, é vantajoso ter uma granularidade com um registro.

Granularidade de itens e bloqueio múltiplo

Granularidade do item de dados - bloqueio

Qual o melhor tamanho de item?

Depende dos tipos de transações envolvidas.

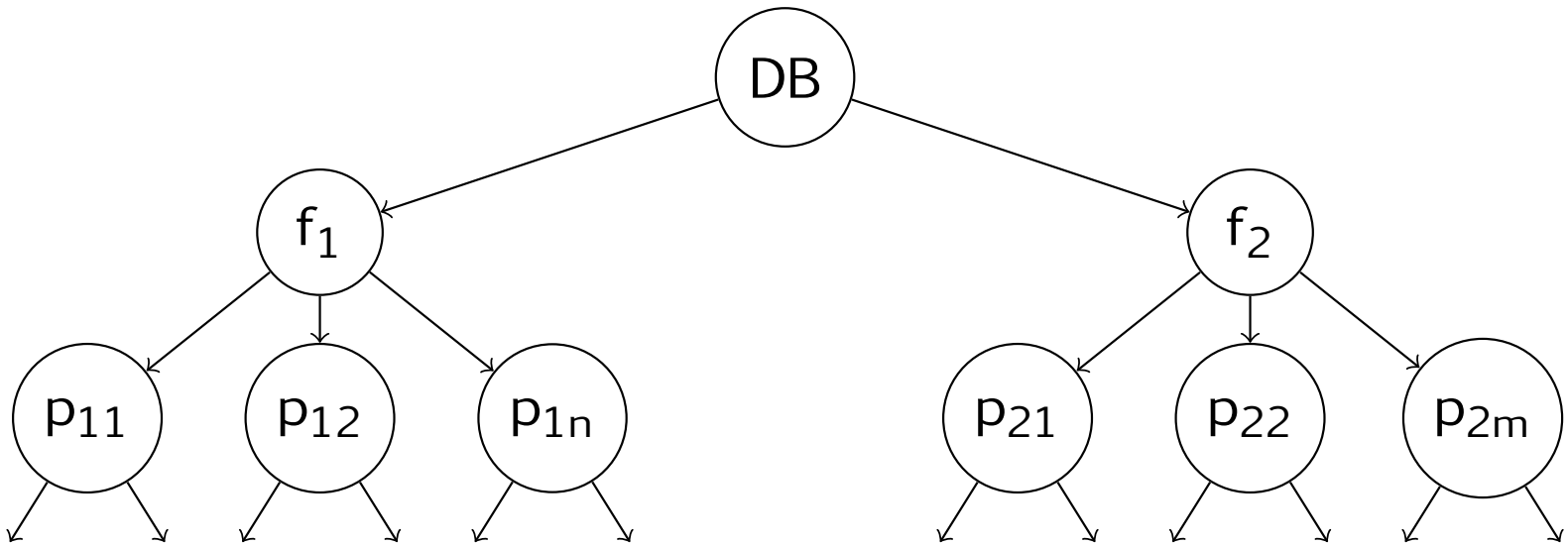
- Se uma transação acessar muitos registros em um mesmo arquivo, é vantajoso ter uma granularidade de bloco ou arquivo de modo que a transação considerará todos os registros como um (ou alguns) item(ns) de dados.

Bloqueio com nível de granularidade múltiplo

- Como o melhor tamanho de granularidade depende da transação dada, é apropriado que um sistema de BD admita múltiplos níveis de granularidade.

Bloqueio com nível de granularidade múltiplo

Hierarquia de granularidade simples com um banco de dados que contém dois arquivos - cada arquivo com várias páginas de disco, e cada página contendo vários registros:



Granularidade de itens e bloqueio múltiplo

Bloqueio com nível de granularidade múltiplo

- O esquema apresentado anteriormente ilustra o protocolo 2PL com **nível de granularidade múltiplo**.
  - O bloqueio pode ser solicitado em qualquer nível;
  - Tipos adicionais de bloqueio serão necessários para dar suporte a tal protocolo com eficiência.
- Para o tornar mais prático, são necessários os bloqueios de intenção.

Bloqueios de intenção

- Uma transação indica, junto com o caminho da raíz até o nó desejado, qual o tipo de bloqueio ele exigirá de um dos descendentes do nó;
- Três tipos de bloqueios de intenção:
  1. **Intention-shared (IS)**
     - Um ou mais bloqueios compartilhados serão solicitados em algum ou alguns nós descendentes.

Bloqueios de intenção

- Três tipos de bloqueios de intenção:

2. **Intention-exclusive (IX)**
   - Um ou mais bloqueios exclusivos serão solicitados em algum ou alguns nós descendentes.

3. **Shared-intention-exclusive (SIX)**
   - O nó atual está bloqueado de modo compartilhado, mas que um ou mais bloqueios exclusivos serão solicitados em algum ou alguns nós descendentes.

Bloqueios de intenção

Tabela de compatibilidade dos três bloqueios de intenção:

	IS	IX	S	SIX	X
IS	Sim	Sim	Sim	Sim	Não
IX	Sim	Sim	Não	Não	Não
S	Sim	Não	Sim	Não	Não
SIX	Sim	Não	Não	Não	Não
X	Não	Não	Não	Não	Não

Granularidade de itens e bloqueio múltiplo

- Protocolo de **bloqueio com granularidade múltipla**, regras:

  1. A compatibilidade de bloqueio deve ser aderida;
  2. A raíz da árvore precisa ser bloqueada primeiro, em qualquer modo;
  3. Um nó **N** pode ser bloqueado por uma transação **T** no modo **S** ou **IS** somente se o nó pai **N** já estiver bloqueado pela transação **T** no modo **IS** ou **IX**;

Granularidade de itens e bloqueio múltiplo

- Protocolo de **bloqueio com granularidade múltipla**, regras:

  4. Um no **N** só pode ser bloqueado por uma transação **T** no modo **X**, **IX** ou **SIX** se o pai do nó **N** já estiver bloqueado pela transação **T** no modo **IX** ou **SIX**;

  5. Uma transação **T** só pode bloquear um nó se não tiver desbloqueado qualquer nó;

  6. Uma transação **T** só pode desbloquear um nó, **N**, se nenhum dos filhos do nó **N** estiver atualmente bloqueado por **T**.

TÉCNICAS DE CONTROLE DE CONCORRÊNCIA:

Bloqueios para controle de concorrência de índices

Bloqueios p/ controle de concorrência de índices

- O bloqueio em duas fases também pode ser aplicado a índices:
  - Os nós de índice correspondem a páginas de disco.
- A estrutura de árvore do índice pode ser aproveitada **quando se desenvolve um esquema de controle de concorrência.**
- **Técnica conservadora para inserções:**
  - Bloquear o nó raíz no modo exclusivo e depois acessar o nó filho apropriado da raíz;
  - Se o nó filho não estiver cheio, o bloqueio no nó raíz pode ser liberado;

Bloqueios p/ controle de concorrência de índices

- **Técnica conservadora para inserções:**
  - Aplicável a toda a árvore (até a folha);
  - Embora os bloqueios exclusivos sejam mantidos, eles são logo liberados.
- **Técnica alternativa (I) - otimista:**
  - Manter **bloqueios compartilhados** nos nós que levam ao nó folha, com um **bloqueio exclusivo** na folha;
  - Se a inserção fizer que a folha seja divida, esta se propagará para os nós de maiores níveis;
  - Ao fim, os bloqueios nos nós de nível mais alto podem receber um upgrade para o modo exclusivo.

Bloqueios p/ controle de concorrência de índices

- **Técnica alternativa (II):**
  - Utiliza uma variante B^+-tree - **Árvore B-link**;
  - Os nós irmãos no mesmo nível são ligados em cada nível;
  - Permite que os bloqueios compartilhados sejam usados quando se solicita uma página e exige que o bloqueio seja liberado antes de acessar o nó filho;
  - Se a inserção fizer que a folha seja divida, esta se propagará para os nós de maiores níveis;
  - **Para inserção:** o bloqueio compartilhado em um nó receberia um upgrade para o modo exclusivo;

Bloqueios p/ controle de concorrência de índices

- **Técnica alternativa (II):**
  - **Para divisão:** o nó pai precisa se bloqueado novamente no modo exclusivo;
  - Complicação nas operações de pesquisa executadas simultâneamente com a atualização;
  - Para exclusão (em que dois ou mais nós da árvore são mesclados): faz parte do protocolo de concorrência da árvore B-link.
    - Os bloqueios nos nós a serem mesclados, e em seus "pais"são mantidos.

TÉCNICAS DE CONTROLE DE CONCORRÊNCIA:

Outras questões

Outras questões

Inserção, exclusão e registros fantasmas

- Quando um item é **inserido**, ele não pode ser acessado antes que a operação seja concluída.
  - Um bloqueio para o item pode ser criado e definido como exclusivo (gravação);
  - O bloqueio pode ser liberado ao mesmo tempo dos demais bloqueios;
  - Para um protocolo de rótulo de tempo, os rótulos de leitura e gravação são definidos como o **rótulo de tempo da transação de criação**.

Outras questões

Inserção, exclusão e registros fantasmas

- Quando um item é **excluído**:
  - Um bloqueio exclusivo deve ser obtido antes que a transação possa excluir o item;
  - Para a ordenação do rótulo do tempo, o protocolo precisa garantir que nenhuma gravação posterior tenha lido ou gravado o item antes da permissão para exclusão.

Outras questões

Inserção, exclusão e registros fantasmas

- **Problema do fantasma:**
  - Ocorre quando um novo registro que está sendo inserido por uma transação T satisfaz uma condição que um conjunto de registros acessados por outra transação T′ precisa satisfazer;
  - Se outras operações nas duas transações estiverem em conflito, o conflito do registro fantasma pode não ser reconhecido pelo protocolo.
  - **Solução:** Bloqueio de índice
  - **Solução alternativa:** Bloqueio de predicado
    - Bloqueia o acesso a todos os registros que satisfazem um **predicado** (condição) qualquer de maneira semelhante;

Outras questões

Latches

- Bloqueios mantidos por uma curta duração;
- Não seguem o protocolo de controle de concorrência normal
- Exemplo:
  - Pode ser usado para garantir a integridade física de uma página quando ela está sendo gravada do buffer para o disco;
  - Um latch seria adquirido para a página, a página seria gravada no disco e, depois, o latch seria liberado.

OBRIGADO

Wladmir Cardoso Brandão
www.wladmirbrandao.com

"Science is more than a body of knowledge. It is a way of thinking."
Carl Sagan