Sistemas de Banco de Dados

Fundamentos em Bancos de Dados Relacionais

Wladmir Cardoso Brandão
www.wladmirbrandao.com

INTRODUÇÃO

Sistemas de Banco de Dados (SBD)

"Sistemas de banco de dados referem-se ao conjunto de dados relacionados e sua respectiva forma de acesso e organização... São compostos por uma ***coleção de dados organizados****, uma* ***estrutura lógica*** *determinando a forma como os dados são armazenados, organizados e manipulados, e um* ***software*** *que provê acesso aos dados a usuários e aplicações."*

Elmasri & Navathe, 2016

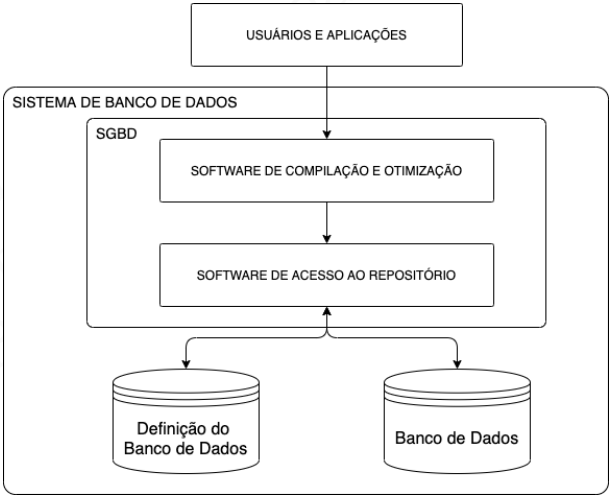
- Coleção de Dados → banco de dados
- Estrutura Lógica → modelo de dados
- Software → sistema gerenciador de banco de dados

Sistemas de Banco de Dados (SBD)

Usuários e aplicações interagem com o sistema submetendo **CONSULTAS**

CONSULTAS são interpretadas pelo sistema, que realiza otimizações necessárias para sua correta execução

O próprio sistema decide quais dados são necessários para responder uma **CONSULTA** e se encarrega de recuperá-los a partir dos repositórios sob seu controle



Banco de Dados (BD)

Coleção de dados organizados

- Dados → símbolos, sinais, códigos
- Atende necessidades específicas de usuários
- Presente em diferentes ambientes de negócio
  - Reserva de hotel
  - Reserva de livros em biblioteca
  - Visualização de catálogos de filmes
  - Compra de produtos em supermercado
  - Saque e depósito de dinheiro em caixa bancário

Bilhões de produtos em catálogo

Dezenas de milhões de transações diárias

Atualização frequente de estoque e pedidos

BD: Propriedades

BDs possuem características que os diferenciam de outros tipos de coleções

- **Finalidade** → construídos com um propósito específico
- **Realidade** → representam o *"mundo real"*
  - Mundo Real → minimundo, universo de discurso
- **Coerência** → mantêm a coerência lógica da coleção
- **Compartilhamento** → provêm compartilhamento de dados

BD: Taxonomia

BDs podem ser categorizados quanto à forma de utilização

- Manual → criado e mantido sem o uso de computadores
  - Exemplo → lista telefônica (páginas amarelas)
- Computadorizado → criado e mantido com o uso de computadores
  - Exemplo → The Human Genome Database (GDB)

BD: Taxonomia

BDs também podem ser categorizados quanto à sua aplicação

- Tradicional → texto, incluindo números e registros temporais
- Multimídia → imagens, áudios e vídeos
- Geográfico → mapas, imagens de satélite e registros climáticos
- Data Warehouse → armazém de dados utilizado no processamento analítico *online* (OLAP) para auxílio à tomada de decisão
- Ativo (Tempo Real) → utilizado em aplicações com rigorosos requisito de desempenho, como em processos industriais de manufatura

BD: Abordagens

Diferentes abordagens de implementação

- Processamento em Arquivo
  - Usuário define arquivos necessários para uma aplicação específica como parte da programação da aplicação
- Sistema Gerenciador de Banco de Dados (SGBD)
  - Repositório único
  - Abstração de dados
  - Natureza autodescritiva
  - Compartilhamento de dados
  - Isolamento entre programas e dados
  - Suporte a múltiplas visões sobre dados
  - Processamento de transação multiusuário

BD: Projeto

Construção de modelos para implementação

- Modelo → representação de entes e eventos reais
- Etapas de implementação
  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
- Revisado continuamente para que o BD reflita o estado do minimundo

BD: Atores

Ator → papel desempenhado pelos que interagem com o BD

- Administrador (DBA) → responsável pela operação e pelo cumprimento dos requisitos, atuando em todas as etapas da implementação
- Projetista → responsável pelo projeto, atuando em todas as etapas da implementação
- Analista → mais presente nas etapas de projeto conceitual e lógico
- Programador → atua preponderantemente no projeto lógico
- Usuário → demandante, conhecedor do minimundo e mais presente na especificação e análise de requisitos

Modelo de Dados

Estrutura lógica que determina a forma como os dados são armazenados, organizados e manipulados

- Coleção de conceitos que descrevem a estrutura do BD
- Incorpora operações para especificar atualização e recuperação de dados
  - Exemplo → inserir, remover, modificar ou recuperar
- Define o comportamento de uma determinada aplicação

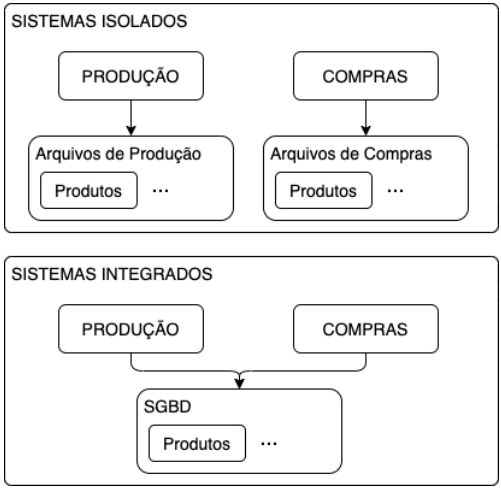
Sistema Gerenciador de Banco de Dados (SGBD)

Coleção de programas (software) que permitem aos usuários criar e manter BDs

- Definir → especificar tipos, estruturas e restrições armazenadas sob forma de metadados no catálogo (dicionário) do sistema
- Construir → armazenar dados em meio controlado pelo SGBD
- Manipular → inserir, remover, modificar e recuperar dados
- Compartilhar → prover acesso simultâneo a múltiplos usuários

SGBD: Propriedades

Controle de Redundância



- Flexibilidade
- Múltiplas interfaces
- Economia de escala
- Garantia de padrões
- Restrições de acesso
- *Backup* e recuperação
- Disponibilidade elevada
- Restrições de integridade
- Tempo de desenvolvimento
- Relacionamentos complexos

SGBD: Limitações

Uso de SGBDs pode ser inadequado em algumas situações

- Monousuário → acesso por múltiplos usuários não requerido
- Baixa complexidade → aplicações muito simples e bem definidas
- Requisitos rigorosos → aplicações de tempo real, de alta escalabilidade e sistemas embarcados com capacidade de armazenamento limitada
- Alta especialização → aplicações que demandam recursos que a generalidade oferecida pelo SGBD para definição e processamento de dados não suporta
  - Exemplo → funções de segurança sofisticadas
- Custo proibitivo → impossibilidade de investimento inicial significativo em hardware, software e treinamento

MODELO DE DADOS

Estrutura de BD

Composta por tipos, relacionamentos e restrições aplicados aos dados

- Abstração de Dados
  - Estrutura do BD percebida de maneira diferente por usuários de acordo com diferentes níveis de detalhamento
  - Abordagem de BD deve ocultar 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

Modelo de Dados

Estrutura lógica que determina a forma como os dados são armazenados, organizados e manipulados

- Coleção de conceitos que descrevem a estrutura do BD
- Incorpora operações para especificar atualização e recuperação de dados
  - Exemplo → inserir, remover, modificar ou recuperar
- Define o comportamento de uma determinada aplicação

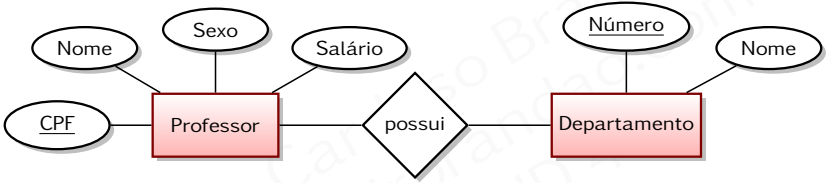
Modelo de Dados

Oferece diferentes níveis de abstração:

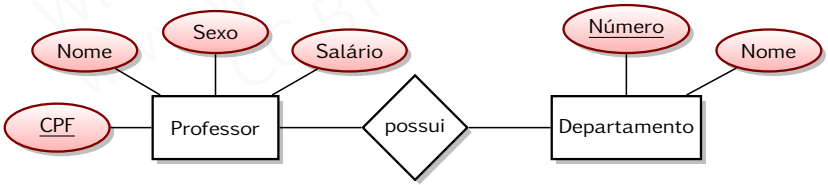
- Conceitual → alto nível de abstração
  - *Representa a estrutura como os usuários a percebem*
  - Conceitos → entidade, atributo e relacionamento
- Representativo → nível intermediário de abstração
  - Também conhecido como modelo de implementação
  - Representa a estrutura detalhando aspectos de implementação
  - Oculta detalhes de armazenamento físico
  - Conceitos → objeto, relação, tupla e coluna
- Físico → baixo nível de abstração
  - Representa a estrutura detalhando aspectos de armazenamento físico
  - Conceitos → arquivo, registro, campo, índice

Modelo de Dados: Conceitual

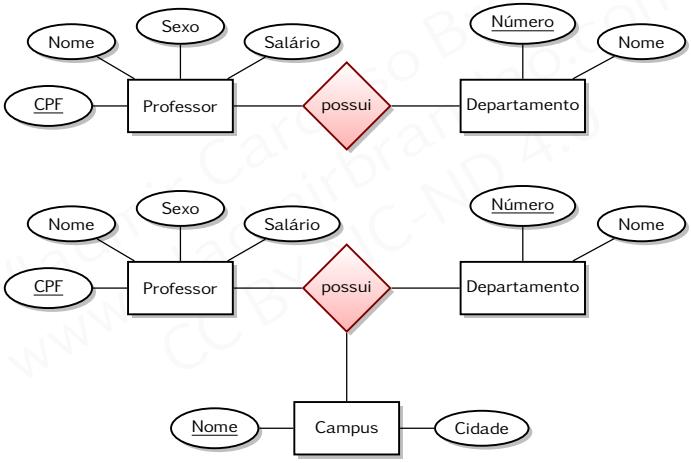
Entidade → ente (objeto) do universo de discurso



Atributo → propriedade que caracteriza uma entidade



Relacionamento → associação entre duas ou mais entidades



Modelo de Dados: Representativo

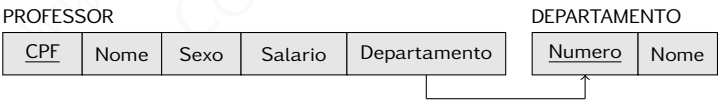
Existem diferentes modelos representativos

- 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
  - Permite associações *N:N*
- Objeto
  - BD → coleção de objetos
  - Registro → objeto
  - Associação entre registros → ligação
  - Próximo ao modelo de dados conceitual

Modelo de Dados: Representativo

Existem diferentes modelos representativos

- Relacional
  - BD → coleção de relações (elementos tabulares)
  - Registro → tupla (linha)
  - Associação entre registros → relacionamentos
  - Embasamento em lógica de predicados e na teoria dos conjuntos
  - Amplamente adotado em SGBDs comerciais baseados em transações
  - Consolidado, com alto desempenho na execução de operações básicas



Modelo de Dados: Físico

Descreve detalhes de armazenamento de dados em memória

- Formatos e ordenação de registros em arquivos
- Organização dos dados em arquivos em memória secundária
- Caminhos de acesso alternativos para recuperação rápida de registros

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento	#
12345678900	Roberto Machado	M	1200.00	1	00
21345678900	Carlos A. Martins	M	3200.00	1	10
32145678900	Ana Maria Freitas	F	7500.00	2	20
12345678901	Manuela Costa	F	2700.00	3	30
52345678902	Luiz A. Barbosa	M	5300.00	3	40
97345671200	Rebeca Lins Rêgo	F	6800.00	3	50
…	…	…	…	…	…
68345618900	Amanda Ramirez	F	1700.00	N	FF

ÍNDICE

Ponteiro	Departamento
#00	1
#20	2
#30	3
…	…
#FF	N

ESQUEMA E LINGUAGEM

Esquema de BD

Descrição do banco de dados (metadados)

- Especificado no projeto e não muda com frequência
- Existem convenções para se representar esquemas usando diagramas
- Diagrama de Esquema → representação de um esquema
  - Capta aspectos como restrições, tipos de registros e de itens de dados

- Construtor de Esquema → elemento que compõe o esquema, como por exemplo professor

Esquema de BD

O diagrama de esquema apresenta a estrutura de cada tipo de elemento, mas **NÃO** apresenta as instâncias dos elementos

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento
12345678900	Roberto Machado	M	1200.00	1
12345678901	Manuela Costa	F	2700.00	3
21345678900	Carlos A. Martins	M	3200.00	1
32145678900	Ana Maria Freitas	F	7500.00	2

Instância de BD

Conjunto de dados armazenados em determinado momento

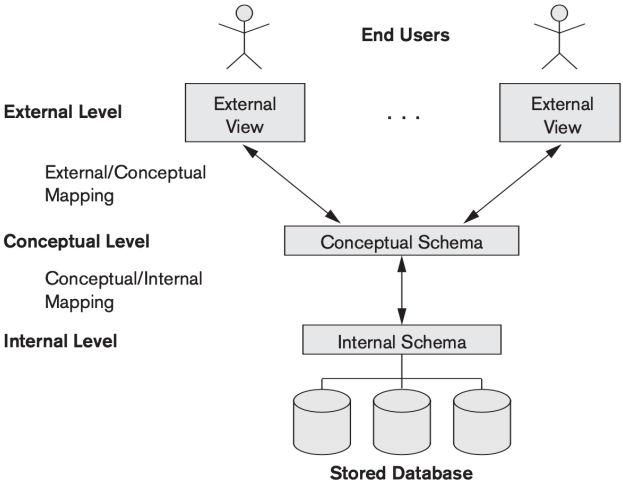
- Estado Vazio → esquema especificado, mas nenhum dado armazenado
- Estado Inicial → BD carregado (populado) com dados inicias
- Estado se altera ao se **inserir**, **remover** ou **modificar** o valor de um item

Arquitetura de Três Esquemas

Abordagem que permite visualização do esquema em diferentes níveis

- Autodescrição → metadados descritivos em diferentes níveis de abstração, de acordo com características estruturais
- Suporte a Múltiplas Visões → usuários e aplicações têm acesso restrito a porções do BD suficientes para atender suas necessidades
- Independência de Aplicação → estrutura do BD armazenada separadamente de aplicações, garantindo que alterações na estrutura não necessariamente levem a mudanças em aplicações

Arquitetura de Três Esquemas



Arquitetura de Três Esquemas

- Nível Externo
  - Esquema Externo → visões de usuário
  - Cada visão descreve a parte do BD em que um grupo de usuários está interessado, ocultando o restante
  - Implementado com modelo de dados representativo
- Nível Conceitual
  - Esquema Conceitual → estrutura do BD
  - Descrição de tipos de dados, entidades, relacionamentos, restrições e operações do usuário
  - Oculta detalhes de armazenamento físico
- Nível Interno
  - Esquema Físico → estrutura do armazenamento físico do BD
  - Descrição de detalhes de armazenamento de dados e de caminhos de acesso

Arquitetura de Três Esquemas

Níveis apresentam descritores para dados que estão efetivamente armazenados em meio físico

- **Mapeamento** → transformação de requisições e resultados entre níveis
- SGBD transforma uma solicitação especificada por usuários em uma solicitação no esquema conceitual e, em seguida, em uma solicitação no esquema interno para que o processamento de dados possa ser realizado

Independência de Dados

Capacidade de se alterar o esquema em um nível sem precisar alterar o esquema no nível adjacente superior

- Lógica → capacidade de alterar o esquema conceitual sem precisar alterar o esquema externo
  - Exemplo → ao acrescentar ou remover um tipo de dado somente o mapeamento entre os níveis e a definição da visão são alterados
- Física → capacidade de alterar o esquema interno sem precisar alterar o esquema conceitual
  - Exemplo → ao organizar arquivos físicos criando estruturas de acesso adicionais somente o mapeamento entre os níveis é alterado

Independência de Dados

A arquitetura de três esquemas facilita a independência de dados

- Independência lógica é mais difícil de ser alcançada, uma vez que é mais difícil realizar alterações estruturais e de restrição sem afetar as aplicações
- Poucos SGBDs comerciais implementam a arquitetura de três esquemas completamente por haver uma sobrecarga com os mapeamentos, levando a ineficiência

Linguagens

Abordagem de BD precisa oferecer linguagens e interfaces apropriadas para cada tipo de usuário

- VDL → linguagem de definição de visão que especifica o esquema externo, as visões de usuário e seus mapeamentos ao esquema conceitual
- DDL → linguagem de definição de dados que especifica o esquema conceitual
- SDL → linguagem de definição de armazenamento que especifica o esquema interno
- DML → linguagem de manipulação de dados utilizada para especificação de operações de inserção, exclusão, modificação e recuperação de dados

Linguagens

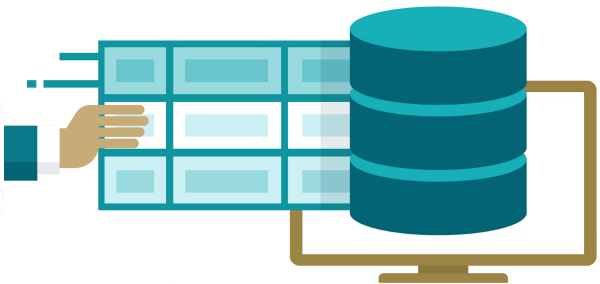
Se diferenciam quanto à forma como as operações são especificadas

- Alto Nível → não procedural
  - Especifica operações complexas de forma concisa
  - Pode recuperar muitos registros em uma única instrução
  - Declarativa → especifica quais dados recuperar e não como
  - Denominada **linguagem de consulta** por ser usada de maneira interativa
- Baixo Nível → procedural
  - Embutida em linguagem de programação de uso geral (linguagem hospedeira), sendo assim denominada sublinguagem de dados
  - Recupera objetos ou registros individuais e os processa separadamente

Linguagens

SGBDs tipicamente não consideram as diferentes linguagens como distintas

- SQL → linguagem de consulta estruturada que combina VDL, DDL, SDL e DML, bem como instruções para especificação de restrição, evolução de esquema e outros recursos

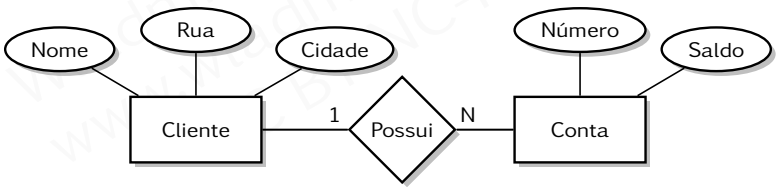


MODELO ENTIDADE-RELACIONAMENTO

Modelo Entidade-Relacionamento (ER)

Modelo conceitual elaborado a partir da especificação do minimundo

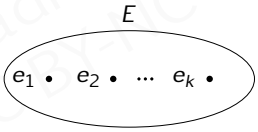
- Minimundo → tipicamente especificado de forma textual, estabelecendo os requisitos de dados
- Diagrama Entidade-Relacionamento → representação gráfica de entidades, atributos, relacionamentos e restrições do modelo ER



ER: Entidade

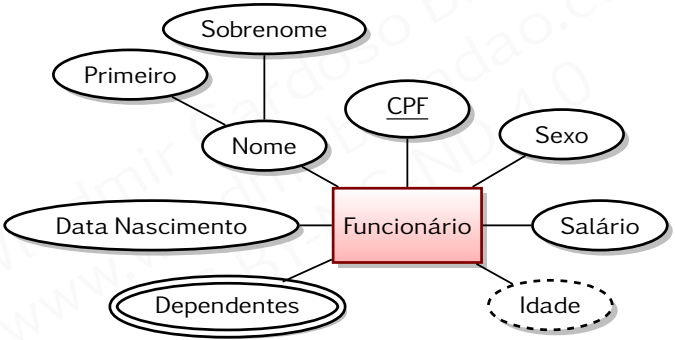
Ente com existência real no minimundo especificado

- Seja $E = \{e_1, e_2, \cdots, e_k\}$ um conjunto de k entidades de mesmo tipo
  - Tipo de Entidade (E) $\rightarrow$ conjunto de instâncias de entidades do mesmo tipo
  - Instância de Entidade (e_i) $\rightarrow$ ente específico de um tipo de entidade E, tal que $e_i \in E$

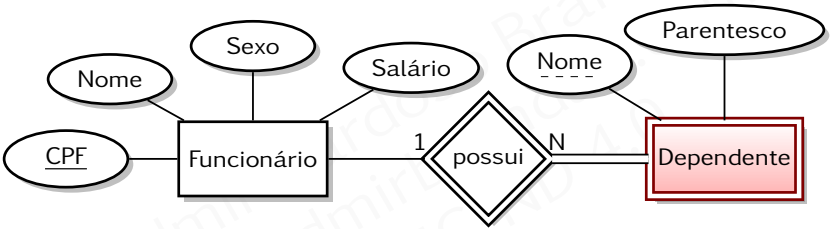


ER: Entidade

No diagrama ER representa-se um tipo de entidade, ou simplesmente entidade, como um retângulo rotulado



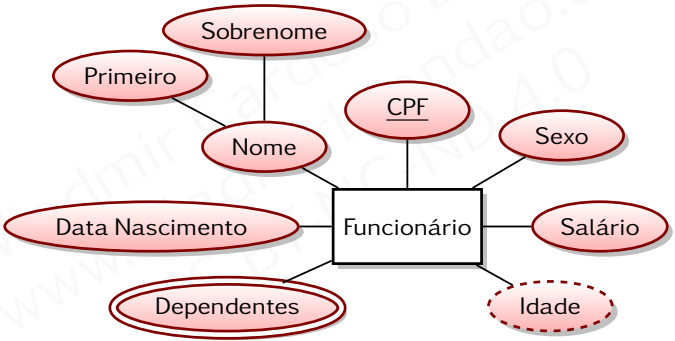
Entidade Fraca → entidade que existência depende da existência de outra



ER: Atributo

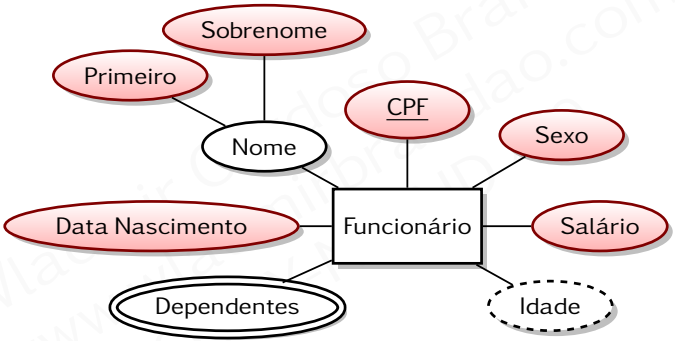
Propriedade que descreve uma característica específica de uma entidade

Representa-se como uma elipse rotulada e ligada à entidade que ele caracteriza



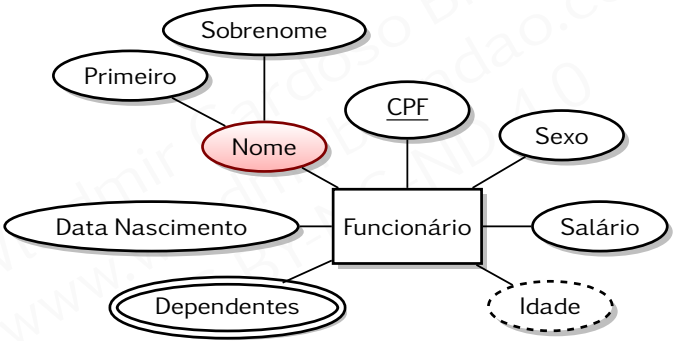
ER: Atributo

Simples → indivisível, representado por uma elipse simples rotulada



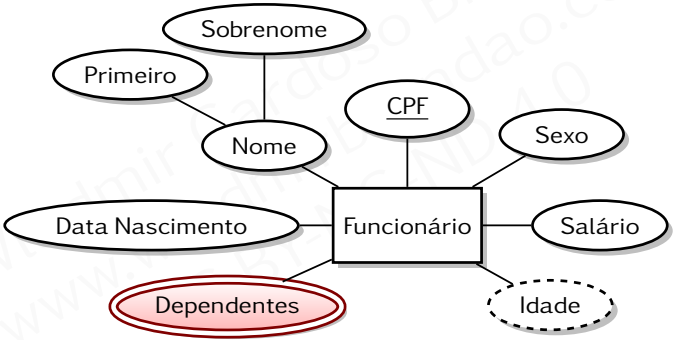
ER: Atributo

Composto → desmembra-se em outros atributos, representado por uma elipse simples rotulada com outros atributos ligados a ele



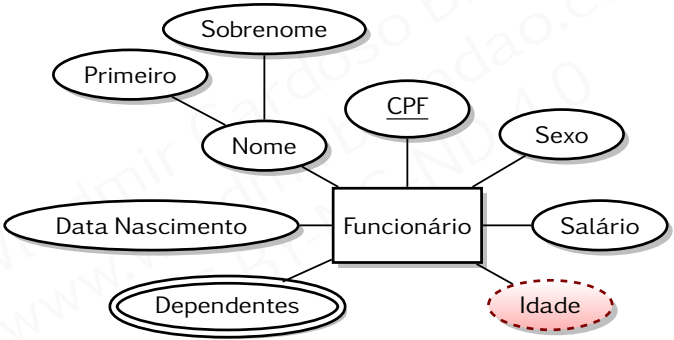
ER: Atributo

MULTIVALORADO → conteúdo formado por mais de um valor, representado por uma elipse rotulada com borda dupla



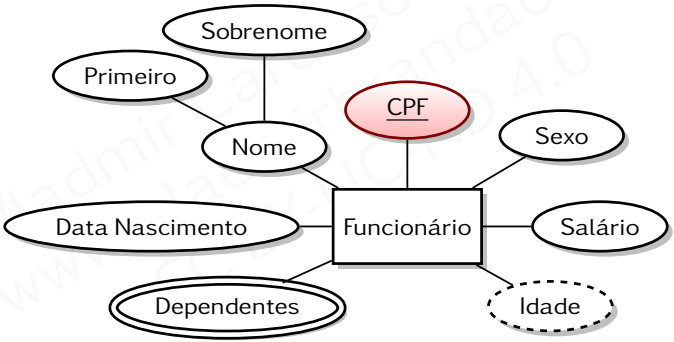
ER: Atributo

Derivado → valor obtido a partir de valores de outros atributos ou relacionamentos, representado por uma elipse rotulada com borda tracejada



ER: Atributo

Chave → atributo ou conjunto de atributos que juntos identificam cada instância de entidade de maneira exclusiva, representado por uma elipse simples rotulada com rótulo sublinhado

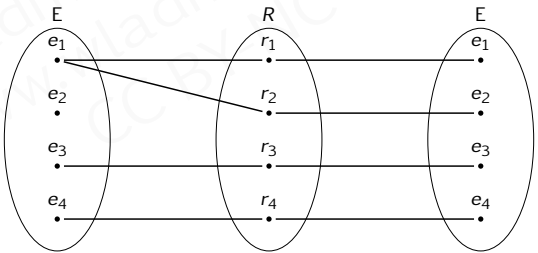


ER: Atributo

Chave Parcial → ou **discriminador**, atributo ou conjunto de atributos que juntos potencialmente identificam cada instância de entidade fraca de maneira exclusiva, representado por uma elipse simples rotulada com rótulo sublinhado de maneira tracejada

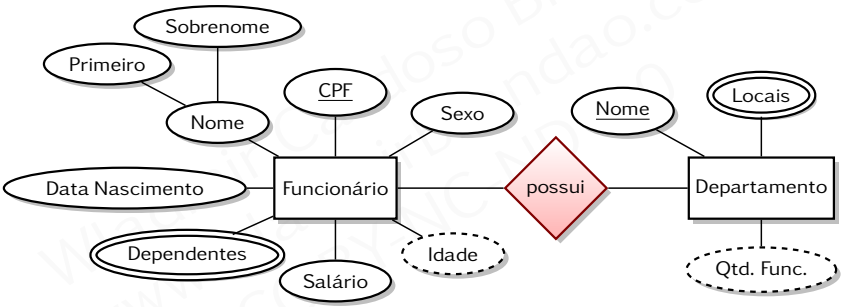
Associação entre entidades

- Seja $R = \{r_1, r_2, \cdots, r_k\}$ um conjunto de k associações entre entidades
  - Tipo de Relacionamento (R) → conjunto de instâncias de associações do mesmo tipo
  - Instância de Relacionamento (r_i) → associação específica entre instâncias de entidades, tal que $r_i \in R$



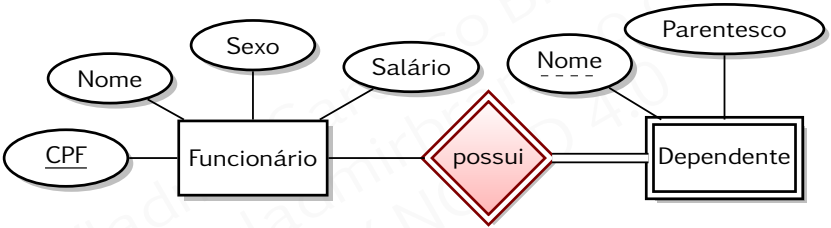
ER: Relacionamento

No diagrama ER representa-se um tipo de relacionamento, ou simplesmente relacionamento, como um losango rotulado



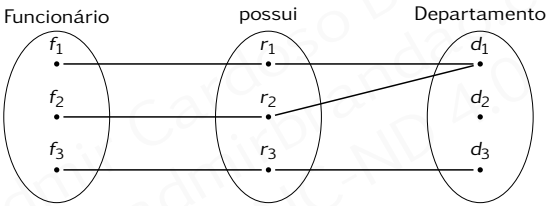
ER: Relacionamento

Relacionamento Fraco → ou **de dependência**, associação envolvendo ao menos uma entidade fraca, representado por um losango rotulado com borda dupla



ER: Relacionamento

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



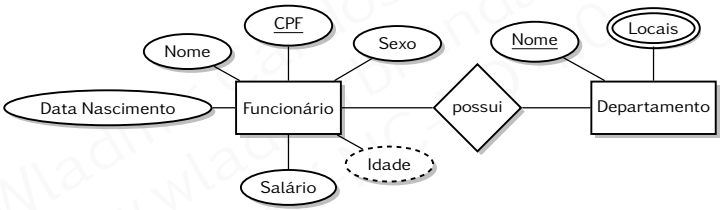
Grau do Relacionamento → # entidades participantes no relacionamento

- Binário → grau 2
- Ternário → grau 3

ER: Relacionamento

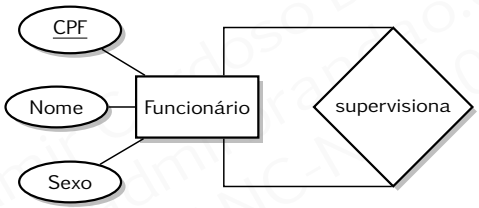
Nome de Função → rotula o relacionamento e representa a função que uma entidade desempenha em cada relacionamento

- Enriquece a semântica do relacionamento



No relacionamento *possui*, Funcionário desempenha a função *possuído*, enquanto Departamento desempenha a função *possuidor*

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

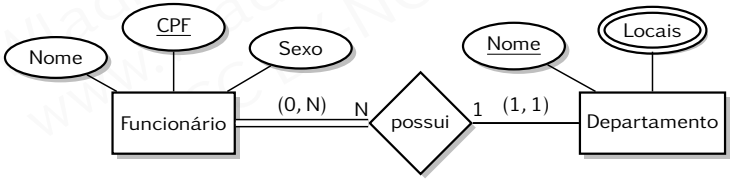


No exemplo, Funcionário participa com as funções de *supervisor* e *supervisionado* no relacionamento *supervisiona*

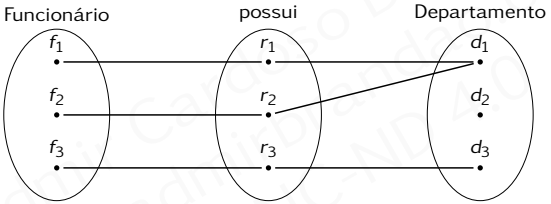
ER: Restrição

Característica limitadora da possibilidade de associação entre entidades nos relacionamentos

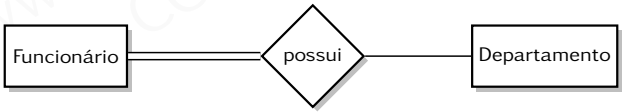
- Razão de Participação → especifica se a participação de uma entidade no relacionamento é parcial ou total
- Razão de Cardinalidade → especifica o número máximo de relacionamentos em que uma entidade pode participar, opcionalmente indica limites mínimos



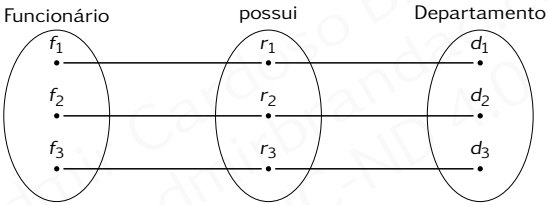
Participação Total → todas as instâncias da entidade devem obrigatoriamente participar de relacionamentos



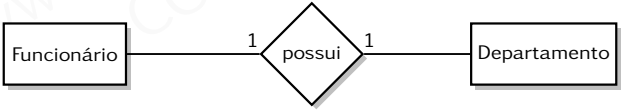
Representa-se por uma linha dupla entre a entidade e o relacionamento



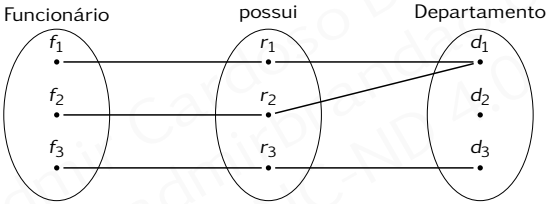
CARDINALIDADE *1:1* → uma instância de cada entidade só pode participar de um único relacionamento



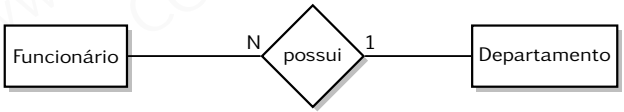
Representa-se por rótulos *1* nas duas extremidades do relacionamento



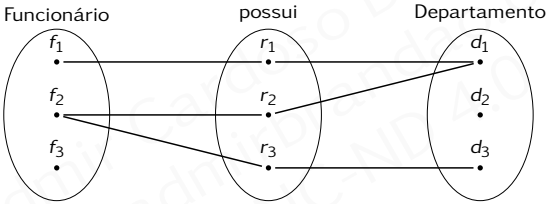
Cardinalidade *1:N* → uma instância de uma entidade só pode participar de um relacionamento, enquanto uma instância da outra pode participar de múltiplos



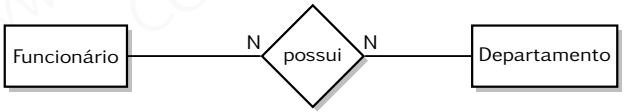
Representa-se por rótulos *1* em uma extremidade e *N* na outra



Cardinalidade *N:N* → uma instância de qualquer entidade pode participar de múltiplos relacionamentos



Representa-se por rótulos *N* nas duas extremidades do relacionamento



ER: Restrição

Mínimos e Máximos → opcionalmente pode-se definir limites mínimos e máximos de cardinalidade para os relacionamentos

Representa-se por rótulos *(min, max)* nas duas extremidades do relacionamento

ER: Convenção de Nomes

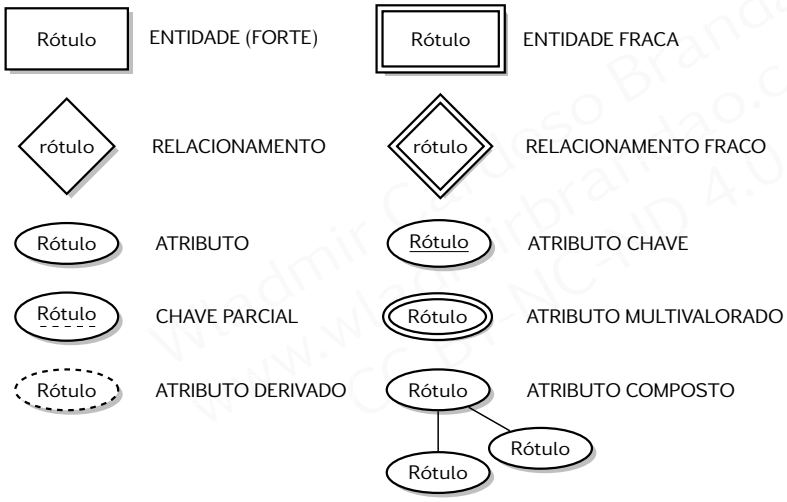
Elementos do modelo ER referenciam elementos textuais em uma especificação textual de minimundo

- Substantivo → pode indicar entidade ou atributo
- Verbo → pode indicar relacionamento

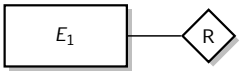
Para a construção do diagrama ER deve-se adotar uma convenção, por exemplo:

- Entidade → nome no singular com letra inicial em maiúscula
- Relacionamento → nome no singular com todas as letras minúsculas
- Atributo → nome com as letras iniciais em maiúscula e atributo multivalorado com nome no plural

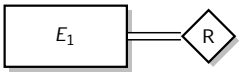
ER: Resumo de Elementos



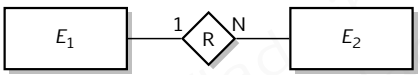
ER: Resumo de Elementos



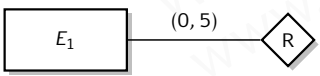
PARTICIPAÇÃO PARCIAL DE E_1 EM R



PARTICIPAÇÃO TOTAL DE E_1 EM R



CARDINALIDADE 1:N DE E_1 PARA E_2



LIMITES DE CARDINALIDADE DE E_1 EM R

MODELO ER ESTENDIDO

Modelo ER Estendido (EER)

Modelo entidade-relacionamento (ER) aprimorado, incorporando conceitos adicionais de modelagem semântica de dados

- Acrônimo do inglês para **Enhanced Entity-Relationship**
- Apresenta requisitos mais complexos e precisos
  - Herança
  - Supertipo e subtipo
  - Restrições complexas
  - Generalização e especialização

EER: Supertipos e Subtipos

Entidades podem possuir SUBTIPOS que precisam de representação explícita

SUBTIPO ou **SUBCLASSE** são subagrupamentos de uma entidade denominada **SUPERTIPO** ou **SUPERCLASSE**

- O relacionamento entre essas entidades é denominado relacionamento de *supertipo/subtipo, superclasse/subclasse* ou *classe/subclasse*

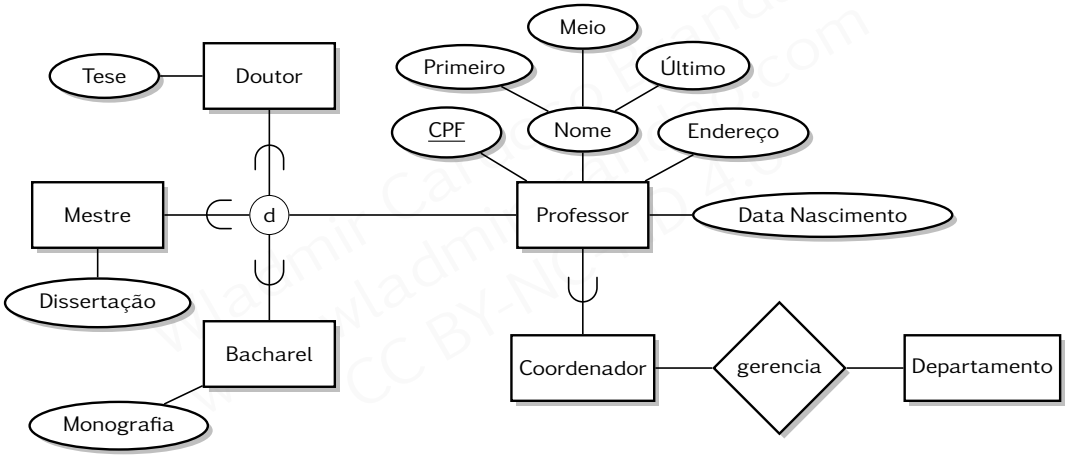
Professor

- Doutor
- Mestre
- Bacharel
- Mensalista
- Horista

Professor *(supertipo)*

- Doutor *(subtipo)*
- Mestre *(subtipo)*
- Bacharel *(subtipo)*
- Mensalista *(subtipo)*
- Horista *(subtipo)*

EER: Supertipos e Subtipos



EER: Herança de Tipo

Ocorre em situações em que a entidade de uma subclasse **herda** todos os atributos e relacionamentos da classe

- Uma entidade na subclasse possui atributos específicos, assim como valores de atributos da classe
  - Exemplo → subclasse DOUTOR possui atributo próprio *Tese* e vários outros atributos herdados da superclasse PROFESSOR, como *CPF*

EER: Especialização

Definição de um conjunto de subclasses de uma entidade com base em alguma característica específica

Caso 1 → Titulação

Professor

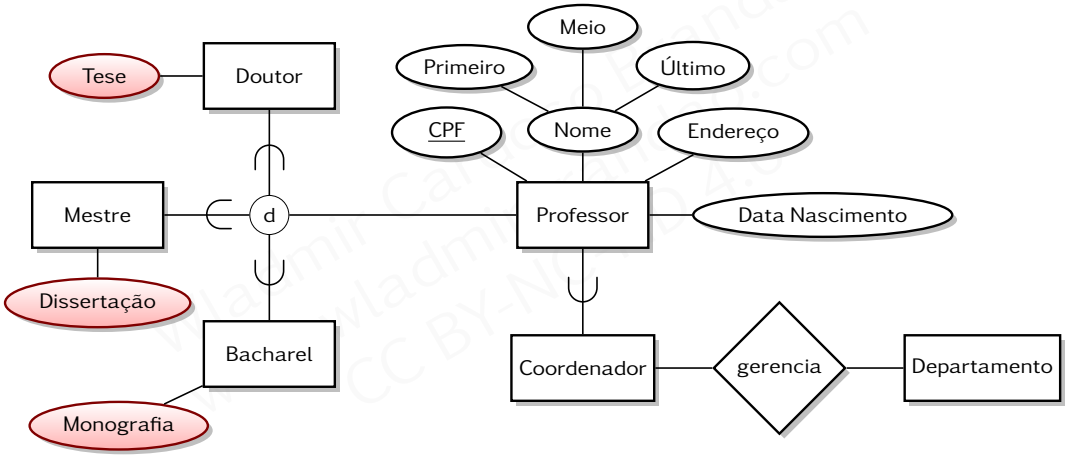
- Doutor
- Mestre
- Bacharel

Caso 2 → Forma de remuneração

Professor

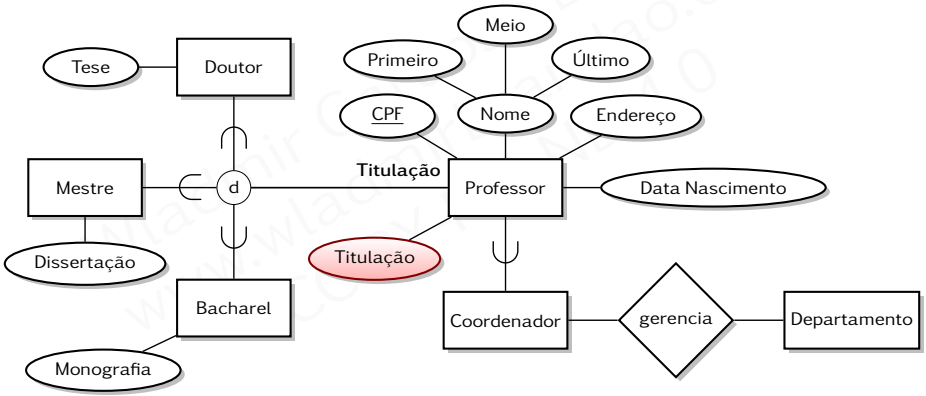
- Mensalista
- Horista

EER: Especialização



EER: Especialização

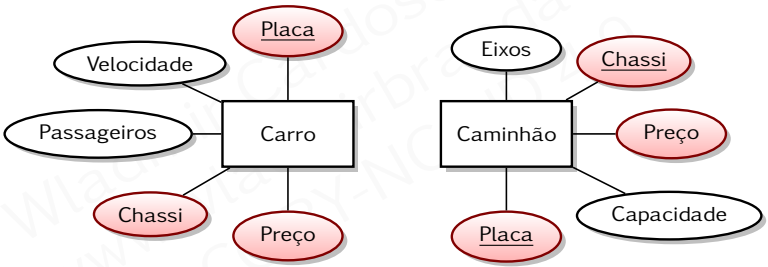
Definição por Condição → entidades que serão subclasse são definidas a partir de uma condição aplicada a um atributo



EER: Generalização

Definição de um tipo de entidade geral com base em entidades específicas

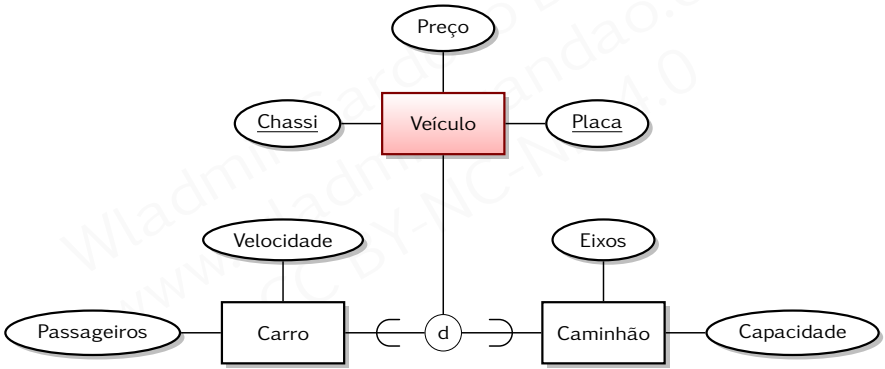
- Identifica-se características comuns e generaliza-se em uma classe



Carro e Caminhão possuem vários atributos comuns

EER: Generalização

Carro e Caminhão podem ser generalizadas, passando a ser subclasses da classe generalizada Veículo

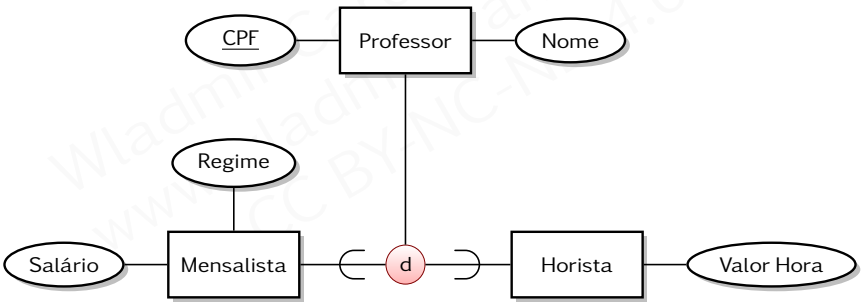


EER: Restrição

Característica limitadora da participação de entidades em subclasses

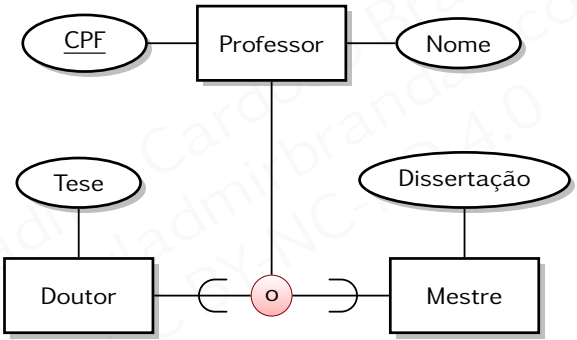
Disjunção → uma entidade pode ser membro de no máximo uma das subclasses

- Uma especialização definida por um atributo de valor único implica em uma restrição de disjunção



EER: Restrição

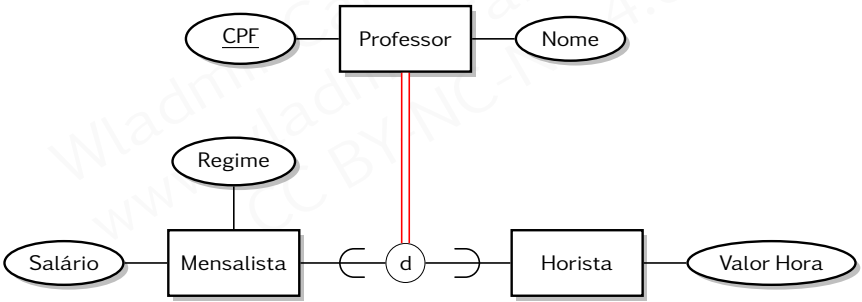
Sobreposição → uma entidade pode ser membro de mais de uma subclasse



EER: Restrição

Participação → determina a participação de uma entidade em subclasses

- Especialização Parcial → entidade não precisa ser membro de subclasses
- Especialização Total → toda entidade precisa ser membro de pelo menos uma subclasse na especialização



EER: Reticulado e Herança

Uma subclasse pode ser superclasse de outras subclasses, formando um **reticulado** de especializações

- Hierarquia Estrita → cada subclasse tem apenas uma superclasse, resultando em uma *estrutura de árvore*
- Reticulado → cada subclasse pode pertencer a diferentes superclasses, resultando em uma *estrutura emaranhada complexa*

Subclasses podem herdar atributos e relacionamentos de múltiplas classes

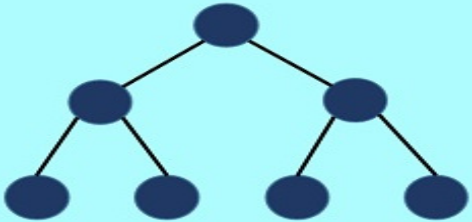
- Herança Simples → herança de uma única classe
- Herança Múltipla → herança de múltiplas classes

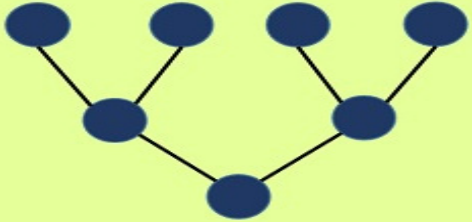
EER: Especialização x Generalização

Especialização → processo de refinamento conceitual de cima para baixo (***top-down***) em que se inicia com uma entidade e depois são definidas subclasses pela especialização sucessiva

Generalização → processo de refinamento conceitual de baixo para cima (***bottom-up***) em que pela síntese conceitual é possível se chegar a mesma hierarquia ou reticulado da alcançada pela outra direção

Estruturalmente o resultado de ambos os processos são idênticos





EER: União

Subclasse representa uma coleção de entidades, um subconjunto da união de entidades distintas

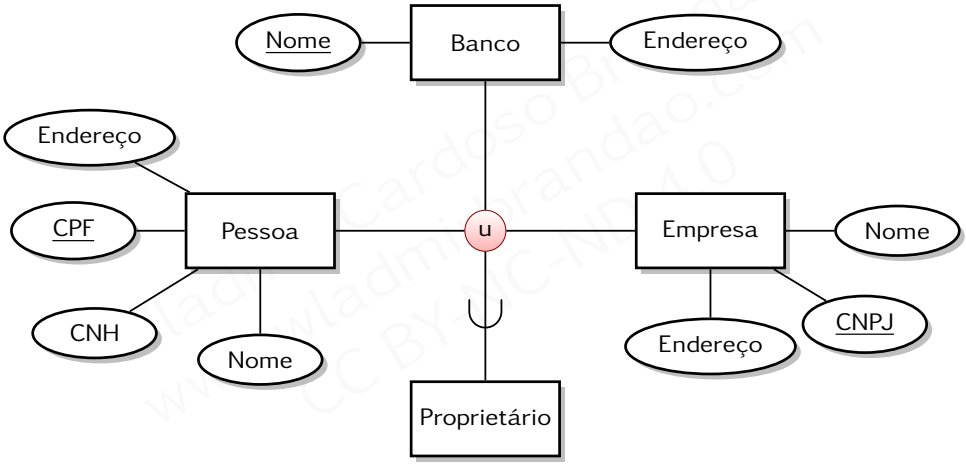
Exemplo → Proprietário pode ser um Banco, uma Empresa ou uma Pessoa

Proprietário (união)

- Banco (entidade)
- Pessoa (entidade)
- Empresa (entidade)

Proprietário herda atributos e relacionamentos de Banco, Empresa e Pessoa

EER: União


Nome
Banco
Endereço
Endereço
CPF
Pessoa
u
Empresa
Nome
CNH
Nome
Endereço
CNPJ
Proprietário

EER: Projeto Conceitual

Em um projeto conceitual há um processo de refinamento iterativo até que o projeto mais adequado seja alcançado

Existem diretrizes para direcionar escolhas em projetos:

1. Representar apenas subclasses necessárias para evitar aglomeração do modelo conceitual
2. Subclasse com poucos atributos são candidatas à mesclagem com a superclasse
   - Atributos específicos da subclasse teriam valores NULL para entidades não membros da subclasse
   - Atributo de *tipo* pode especificar se uma entidade é um membro da subclasse

3 União deve ser evitada, a menos que a situação definitivamente justifique esse tipo de construção

4 A escolha de restrições de disjunção, sobreposição e totalidade sobre a especialização deve ser regida pelas regras do minimundo

- Se na especificação do minimundo não é explicitamente citada nenhuma restrição, o padrão menos restritivo de sobreposição parcial deve ser adotado

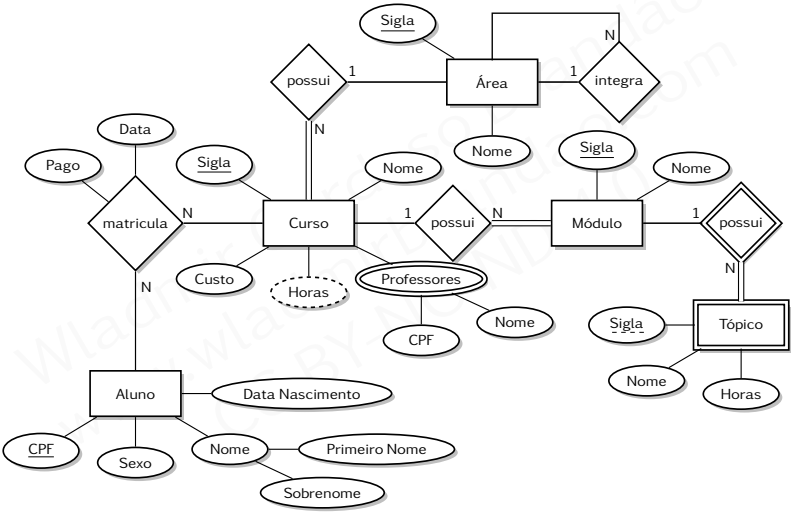
MODELO RELACIONAL

Modelo Relacional

Modelo de implementação (representativo) baseado no paradigma relacional

- Dados são organizados de maneira tabular:
  - Entidade → Relação
  - Relacionamento → Relacionamento
  - Atributo → Atributo
  - Registro → Tupla
- Pode ser criado a partir do EER utilizando um procedimento em 7 etapas

Diagrama EER: Sistema de Matrícula



Etapa 1 → Entidades Fortes

- Crie uma relação para cada entidade forte e inclua todos os atributos simples
- Inclua apenas atributos simples de um atributo composto
- Escolha um dos atributos chave da entidade forte como chave primária da nova relação
- Se a chave escolhida for composta, o conjunto de atributos simples que a compõe formarão a chave primária

Etapa 1 → Entidades Fortes

CURSO

Sigla	Nome	Horas	Custo

AREA

Sigla	Nome

MODULO

Sigla	Nome

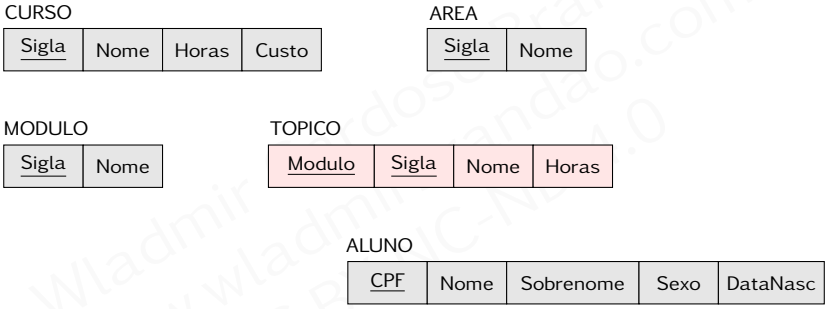
ALUNO

CPF	Nome	Sobrenome	Sexo	DataNasc

Etapa 2 → Entidades Fracas

- Crie uma relação para cada entidade fraca e inclua todos os atributos simples
- Inclua como atributos de chave estrangeira da relação, os atributos de chave primária da relação que corresponde à entidade proprietária
- Escolha a chave estrangeira e um atributo chave parcial como chave primária da nova relação

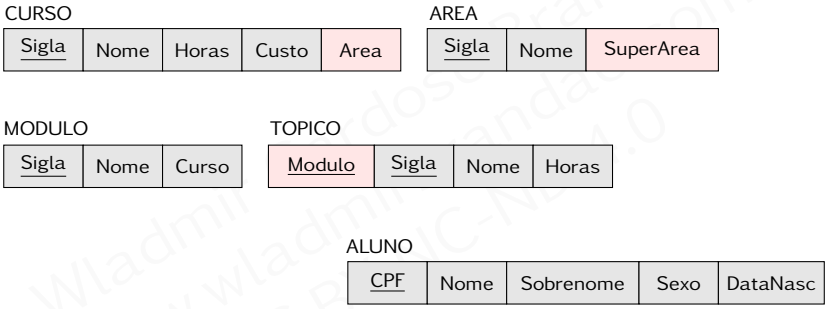
Etapa 2 → Entidades Fracas



Etapa 3 → Relacionamentos Binários 1:N

- Identifique a relação $R1$ que representa a entidade participante no lado N do relacionamento
- Inclua como chave estrangeira em $R1$ a chave primária de $R2$, que representa a outra entidade participante do relacionamento

Etapa 3 → Relacionamentos Binários 1:N



Etapa 4 → Relacionamentos Binários N:N

- Crie uma nova relação $R3$ para cada relacionamento N:N
- Inclua como chave estrangeira em $R3$ as chaves primárias das relações $R1$ e $R2$, que representam as entidades participantes no relacionamento
- A chave primária de $R3$ será formada pela combinação das chaves estrangeiras em $R3$

Etapa 4 → Relacionamentos Binários N:N

CURSO

Sigla	Nome	Horas	Custo	Area

AREA

Sigla	Nome	SuperArea

MODULO

Sigla	Nome	Curso

TOPICO

Modulo	Sigla	Nome	Horas

MATRICULA

Curso	Aluno	Data	Pago

ALUNO

CPF	Nome	Sobrenome	Sexo	DataNasc

Etapa 5 → Relacionamentos Binários 1:1

- Identifique as relações que correspondem às entidades participantes
- Existem três estratégias:

1. Mesclagem → consiste em mesclar as entidades e o relacionamento em uma única relação
2. Chave Estrangeira → consiste em mapear o relacionamento 1:1 como um relacionamento 1:N
3. Referência Cruzada → consiste em mapear o relacionamento 1:1 como um relacionamento N:N

Etapa 6 → Atributos Multivalorados

- Crie uma nova relação para cada atributo multivalorado A
- A nova relação incluirá um atributo de A, mais o atributo da chave primária da relação que representa a entidade ou relacionamento que tenha A como atributo multivalorado

Etapa 6 → Atributos Multivalorados

CURSO

Sigla	Nome	Horas	Custo	Area

AREA

Sigla	Nome	SuperArea

MODULO

Sigla	Nome	Curso

TOPICO

Modulo	Sigla	Nome	Horas

MATRICULA

Curso	Aluno	Data	Pago

ALUNO

CPF	Nome	Sobrenome	Sexo	DataNasc

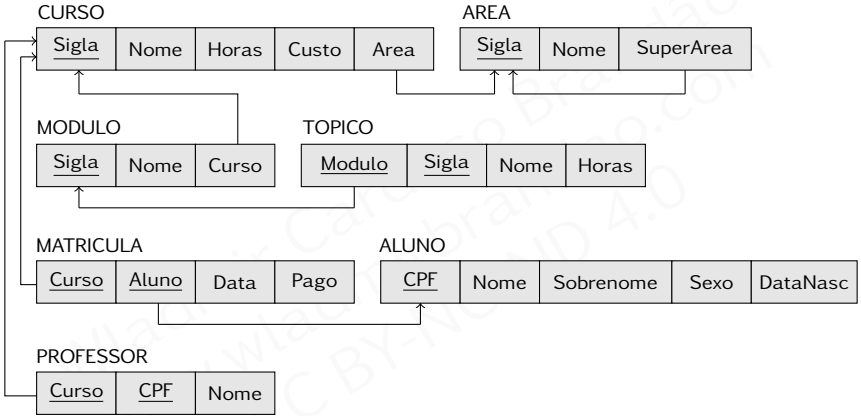
PROFESSOR

Curso	CPF	Nome

Etapa 7 → Relacionamentos de Alto Grau

- Crie uma relação $R3$ para cada relacionamento n-ário, em que $n > 2$
- Inclua como chave estrangeira em $R3$ as chaves primárias das relações que representam as entidades participantes
- A chave primária de $R3$ é a combinação de todas as chaves estrangeiras que referenciam as relações das entidades participantes

Modelo Relacional: Sistema de Matrícula



ÁLGEBRA RELACIONAL I

Álgebra Relacional

Conjunto de operações para manipulação de BD relacionais

- Linguagem formal para o modelo relacional
- Consulta → solicitação de recuperação de dados especificada por uma sequência de operações algébricas
  - Resultado de uma consulta é uma nova relação
- SQL incorpora seus principais conceitos
  - SQL → linguagem de consulta prática para BD relacionais
- Multiplicidade de operações:
  - Unárias → aplicadas sobre uma relação
  - Binárias → aplicadas sobre duas relações
  - Agregação → resumem dados de relações

Operação SELEÇÃO

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

- Operador → selecionar (σ *sigma*)
- Função → **filtrar tuplas** de uma relação R que satisfaçam a uma condição
- Tuplas que não satisfazem a condição são descartadas do resultado
- Relação resultante tem os mesmos atributos de R
- Número de tuplas na relação resultante é menor ou igual ao número de tuplas em R

Operação SELEÇÃO

Condição de seleção é uma **expressão booleana**

<atributo> <operador> <valor> | <atributo> <operador> <atributo>

- *<atributo>* → nome de um atributo de R
- *<operador>* → operador de comparação

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

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

Conectada por operadores booleanos ($\wedge$ $\vee$ $\neg$) formam um único bloco condicional

Operação SELEÇÃO

Exemplo → selecionar tuplas de professores do sexo feminino

$$\sigma_{Sexo = 'F'}(PROFESSOR)$$

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento
12345678900	Roberto Machado	M	1200.00	1
12345678901	Manuela Costa	F	2700.00	3
21345678900	Carlos A. Martins	M	3200.00	1
32145678900	Ana Maria Freitas	F	7500.00	2

Operação SELEÇÃO

Exemplo → selecionar tuplas de professores do sexo masculino que recebem salário maior que 3000,00

$$\sigma_{(Sexo = 'M' \wedge Salario > 3000.00)}(PROFESSOR)$$

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento
12345678900	Roberto Machado	M	1200.00	1
12345678901	Manuela Costa	F	2700.00	3
21345678900	Carlos A. Martins	M	3200.00	1
32145678900	Ana Maria Freitas	F	7500.00	2

Operação SELEÇÃO

- Condições de seleção são aplicadas a cada tupla individualmente e não se aplicam a mais de uma tupla
- Operação **unária** e **comutativa**

$$\sigma_{cond_1}(\sigma_{cond_2}(R)) = \sigma_{cond_2}(\sigma_{cond_1}(R))$$

- Pode-se combinar uma sequência de operações em uma única operação com operadores conjuntivos

$$\sigma_{cond_1}(\sigma_{cond_2}(\ldots(\sigma_{cond_n}(R))\ldots)) = \sigma_{cond_1 \ \wedge \ cond_2 \ \wedge \ \ldots \ \wedge \ cond_n}(R)$$

Operação PROJEÇÃO

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

- Operador → projetar (π *pi*)
- Função → **filtrar atributos** de uma relação R
- Atributos não especificados são descartados do resultado
- Relação resultante possui um subconjunto de atributos de R explicitamente especificados e na mesma ordem
- Número de tuplas na relação resultante é menor ou igual ao número de tuplas em R
  - Menor se houverem tuplas duplicadas, pois duplicatas são eliminadas

Operação PROJEÇÃO

Exemplo → projetar nome e salário de professores

$$\pi_{Nome,\ Salario}\ (PROFESSOR)$$

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento
12345678900	Roberto Machado	M	1200.00	1
12345678901	Manuela Costa	F	2700.00	3
21345678900	Carlos A. Martins	M	3200.00	1
32145678900	Ana Maria Freitas	F	7500.00	2

Operação PROJEÇÃO

Exemplo → projetar número do departamento e sexo de professores

$$\pi_{Departamento,\ Sexo}\ (PROFESSOR)$$

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento
12345678900	Roberto Machado	M	1200.00	1
12345678901	Manuela Costa	F	2700.00	3
21345678900	Carlos A. Martins	M	3200.00	1
32145678900	Ana Maria Freitas	F	7500.00	2

Duas tuplas repetidas ($< 1, M >$) no resultado, uma delas será eliminada

Operação PROJEÇÃO

- Se a lista de atributos para projeção inclui a chave da relação R, a relação resultante terá o mesmo número de tuplas de R
- Operação **unária**, mas **não comutativa**

$$\pi_{atr_1}(\pi_{atr_2}(R)) \neq \pi_{atr_2}(\pi_{atr_1}(R))$$

- Aninhamento de sequência de operações válidas equivale à operação externa do aninhamento

$$\pi_{atr_1}(\pi_{atr_2}(R)) = \pi_{atr_1}(R)$$

Sequência de Operações

Consultas combinam sequências de operações algébricas

- Expressão em Linha → aninha-se múltiplas operações, gerando uma única expressão algébrica

$$\pi_{\ CPF,\ Nome,\ Salario}\ (\ \sigma_{Sexo\ =\ 'F'}\ (PROFESSOR)\ )$$

- Relacões Intermediárias → aplica-se uma operação de cada vez, criando relações com resultados intermediários reutilizáveis

$$A \longleftarrow \sigma_{Sexo\ =\ 'F'}\ (PROFESSOR)$$
$$B \longleftarrow \pi_{\ CPF,\ Nome,\ Salario}\ (A)$$

Operação RENOMEAR

$$\rho_{S(atributos)}(R)$$

- Operador → renomear (ρ *rho*)
- Função → **renomear** relações e atributos
- S → nome da relação resultante
- *atributos* → lista dos novos nomes dos atributos de R na relação resultante, ordem na lista deve ser compatível com ordem dos atributos de R
- Variações:
  - $\rho_{atributos}(R)$ → renomeia apenas atributos
  - $\rho_S(R)$ → renomeia apenas relação

Operação RENOMEAR

Exemplo → renomear a relação professor e seus respectivos atributos

$$\rho_{TEACHER(CPF,Name,Gender,Salary,DNum)}(PROFESSOR)$$

TEACHER

CPF	Name	Gender	Salary	DNum
12345678900	Roberto Machado	M	1200.00	1
12345678901	Manuela Costa	F	2700.00	3
21345678900	Carlos A. Martins	M	3200.00	1
32145678900	Ana Maria Freitas	F	7500.00	2

Operação RENOMEAR

Alternativamente, pode-se renomear relações e atributos utilizando **relações intermediárias**

$$TEACHER_{(CPF,\ Name,\ Gender,\ Salary,\ DNum)} \longleftarrow PROFESSOR$$

$$TEACHER_{(Name,\ Salary)} \longleftarrow \pi_{\ Nome,\ Salario}\ (PROFESSOR)$$

$$MAN_{(CPF,\ Nom,\ Sex,\ Sal,\ DNum)} \longleftarrow \sigma_{\ Sexo\ =\ 'M'}\ (PROFESSOR)$$

Pode-se renomear qualquer subconjunto de atributos de R

Operação PRODUTO

$$R_1 \times R_2$$

- Operador $\rightarrow$ multiplicar ($\times$)
- Função $\rightarrow$ **combinar** tuplas de duas relações
- Relação resultante possui os atributos de $R1$ e $R2$, incluindo como tuplas todas as combinações possíveis entre as tuplas de $R1$ e $R2$
- Número de tuplas da relação resultante é o produto cartesiano entre o número de tuplas de $R1$ e $R2$

Operação PRODUTO

Exemplo → *PROFESSOR* × *DEPARTAMENTO*

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento
12345678900	Roberto	M	1200.00	1
12345678901	Manuela	F	2700.00	3

DEPARTAMENTO

Numero	Nome
1	Administrativo
2	Comercial
3	Tecnologia

Resultado:

CPF	Nome	Sexo	Salario	Departamento	Numero	Nome
12345678900	Roberto	M	1200.00	1	1	Administrativo
12345678900	Roberto	M	1200.00	1	2	Comercial
12345678900	Roberto	M	1200.00	1	3	Tecnologia
12345678901	Manuela	F	2700.00	3	1	Administrativo
12345678901	Manuela	F	2700.00	3	2	Comercial
12345678901	Manuela	F	2700.00	3	3	Tecnologia

Operação PRODUTO

Associada à SELEÇÃO opera como uma JUNÇÃO

$$\sigma_{Departamento = Numero}(PROFESSOR \times DEPARTAMENTO)$$

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento
12345678900	Roberto	M	1200.00	1
12345678901	Manuela	F	2700.00	3

DEPARTAMENTO

Numero	Nome
1	Administrativo
2	Comercial
3	Tecnologia

Resultado:

CPF	Nome	Sexo	Salario	Departamento	Numero	Nome
12345678900	Roberto	M	1200.00	1	1	Administrativo
12345678901	Manuela	F	2700.00	3	3	Tecnologia

Operação JUNÇÃO

$$R_1 \bowtie_{condicional} R_2$$

- Operador → juntar ($\bowtie$)
- Função → **combinar** tuplas de duas relações a partir de uma condição
- Relação resultante possui atributos de $R1$ e $R2$, incluindo como tuplas todas as combinações entre as tuplas de $R1$ e $R2$ que respeitam condição
- Tuplas que não respeitam condição de junção ou que valores dos atributos usados na condição sejam NULL são descartadas do resultado

Operação JUNÇÃO

Exemplo → $PROFESSOR \bowtie_{Departamento = Numero} DEPARTAMENTO$

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento
12345678900	Roberto	M	1200.00	1
12345678901	Manuela	F	2700.00	3

DEPARTAMENTO

Numero	Nome
1	Administrativo
2	Comercial
3	Tecnologia

Resultado:

CPF	Nome	Sexo	Salario	Departamento	Numero	Nome
12345678900	Roberto	M	1200.00	1	1	Administrativo
12345678901	Manuela	F	2700.00	3	3	Tecnologia

Operação JUNÇÃO

Equijunção → condicionais com operadores de igualdade

$$PROFESSOR \bowtie_{Departamento = Numero} DEPARTAMENTO$$

Junção Natural → equijunção automática (natural) com atributos que possuem o mesmo nome nas duas relações removendo-se duplicatas

$$PROFESSOR * DEPARTAMENTO$$

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento
12345678900	Roberto	M	1200.00	1
12345678901	Manuela	F	2700.00	3

DEPARTAMENTO

Numero	Nome
1	Administrativo
2	Comercial
3	Tecnologia

Operação JUNÇÃO

Frequentemente relações diferentes possuem atributos com mesmo nome

- Operação **RENOMEAR** deve ser utilizada antes da junção para evitar problemas de ambiguidade

Exemplo:

$$A_{(Numero,\ DNome)} \longleftarrow \pi_{\ Numero,\ Nome}\ (DEPARTAMENTO)$$

$$RESULTADO \longleftarrow PROFESSOR \bowtie_{\ Departamento\ =\ Numero} A$$

RESULTADO

CPF	Nome	Sexo	Salario	Departamento	Numero	DNome
12345678900	Roberto	M	1200.00	1	1	Administrativo
12345678901	Manuela	F	2700.00	3	3	Tecnologia

Operação AGREGAÇÃO

$$atributos\ \gamma_{funcoes}(R)$$

- Operador → agregar (γ *gamma*)
- Função → **agregar** tuplas de uma relação a partir de uma lista de atributos (**atributos de agregação**), aplicando **funções de agregação** em atributos remanescentes
- Múltiplas notações → $\mathscr{F}$, G
- Relação resultante possui atributos de agregação e um atributo para cada função de agregação
- Tuplas com valores NULL nos atributos usados na função de agregação são descartadas da agregação

Operação AGREGAÇÃO

Função de Agregação → função matemática aplicada em tuplas agrupadas

- COUNT → conta o número de tuplas agrupadas
- SUM → soma valores do atributo utilizado na função
- AVG → calcula média dos valores do atributo utilizado na função
- MIN → captura valor mínimo dentre valores do atributo utilizado na função
- MAX → captura valor máximo dentre valores do atributo utilizado na função

Se função de agregação não for renomeada, nome do atributo resultante será concatenação do nome da função e do nome do atributo usado por ela

Operação AGREGAÇÃO

Exemplo → apresentar o número de professores e o total em salários

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento
12345678900	Roberto Machado	M	1200.00	1
12345678901	Manuela Costa	F	2700.00	3
21345678900	Carlos A. Martins	M	3200.00	1
32145678900	Ana Maria Freitas	F	7500.00	2

$$\gamma_{\,COUNT(CPF),\,SUM(Salario)}\,(PROFESSOR)$$

Resultado:

COUNT_CPF	SUM_SALARIO
4	14600.00

Operação AGREGAÇÃO

Exemplo → apresentar o total e a média salarial por sexo

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento
12345678900	Roberto Machado	M	1200.00	1
12345678901	Manuela Costa	F	2700.00	3
21345678900	Carlos A. Martins	M	3200.00	1
32145678900	Ana Maria Freitas	F	7500.00	2

$$Sexo\ \gamma\ SUM(Salario),\ AVG(Salario)\ (PROFESSOR)$$

Resultado:

Sexo	SUM_SALARIO	AVG_SALARIO
F	10200.00	5100.00
M	4400.00	2200.00

ÁLGEBRA RELACIONAL II

Operação PROJEÇÃO

Projeção Generalizada → estende operação de projeção permitindo que funções sejam incluídas na lista de atributos para projeção

$$\pi_{funcoes}(R)$$

Funções envolvem operações aritméticas e valores constantes

Exemplo:

$$A \leftarrow \pi_{CPF,\ Nome\ +\ '\ '\ +\ Sobrenome,\ Salario\ *\ 1.1}(PROFESSOR)$$
$$B \leftarrow \rho_{CPF,\ NomeCompleto,\ Bonus}(A)$$

Operação JUNÇÃO

Junção Interna (INNER JOIN) → operação de junção convencional (JOIN)

Exemplo → $PROFESSOR \bowtie_{Departamento = Numero} DEPARTAMENTO$

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento
12345678900	Roberto	M	1200.00	1
12345678901	Manuela	F	2700.00	3

DEPARTAMENTO

Numero	Nome
1	Administrativo
2	Comercial
3	Tecnologia

Resultado:

CPF	Nome	Sexo	Salario	Departamento	Numero	Nome
12345678900	Roberto	M	1200.00	1	1	Administrativo
12345678901	Manuela	F	2700.00	3	3	Tecnologia

Operação JUNÇÃO

Junção Externa (OUTER JOIN) → a *"relação externa"* participa com tuplas não correspondentes da junção interna

Exemplo → $PROFESSOR \ltimes_{Departamento = Numero} DEPARTAMENTO$

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento
12345678900	Roberto	M	1200.00	1
12345678901	Manuela	F	2700.00	3

DEPARTAMENTO

Numero	Nome
1	Administrativo
2	Comercial
3	Tecnologia

Resultado:

CPF	Nome	Sexo	Salario	Departamento	Numero	Nome
12345678900	Roberto	M	1200.00	1	1	Administrativo
12345678901	Manuela	F	2700.00	3	3	Tecnologia
					2	Comercial

Operação JUNÇÃO

Junção Externa à Esquerda (LEFT OUTER JOIN) → junção externa em que a *"relação externa"* é a da esquerda

$$PROFESSOR \ltimes_{Departamento = Numero} DEPARTAMENTO$$

Junção Externa à Direita (RIGHT OUTER JOIN) → junção externa em que a *"relação externa"* é a da direita

$$PROFESSOR \rtimes_{Departamento = Numero} DEPARTAMENTO$$

Junção Externa Completa (FULL OUTER JOIN) → junção externa em que ambas as relações são *"externas"*

$$PROFESSOR \ltimes\!\rtimes_{Departamento = Numero} DEPARTAMENTO$$

Operações de Conjunto

Operações da teoria dos conjuntos usadas para **mesclar elementos de dois conjuntos**, através de operações binárias

- **União** → adiciona todas as tuplas de ambas as relações
- **Interseção** → adiciona as tuplas comuns entre as relações
- **Diferença** → adiciona as tuplas da primeira relação que não pertencem à segunda relação

Relações devem ser compatíveis, possuindo o mesmo número de atributos alinhados de acordo com o domínio de dados

Tuplas duplicadas são eliminadas da relação resultante

Nomes dos atributos da primeira relação são mantidos na relação resultante

Operações UNIÃO

$$R_1 \cup R_2$$

- Operador → unir ($\cup$)
- Função → **unir** tuplas de duas relações
- Operação **comutativa**

$$R_1 \cup R_2 = R_2 \cup R_1$$

Exemplo → *PROFESSOR* $\cup$ *ALUNO*

PROFESSOR

Nome	Depto
Roberto Machado	1
Manuela Costa	3

ALUNO

Nome	Departamento
Roberto Machado	2
Manuela Costa	3

UNIÃO

Nome	Depto
Roberto Machado	1
Manuela Costa	3
Roberto Machado	2

Operações INTERSEÇÃO

$$R_1 \cap R_2$$

- Operador → intersecionar ($\cap$)
- Função → selecionar tuplas comuns nas duas relações
- Operação **comutativa**

$$R_1 \cap R_2 = R_2 \cap R_1$$

Exemplo → *PROFESSOR* $\cap$ *ALUNO*

PROFESSOR

Nome	Depto
Roberto Machado	1
Manuela Costa	3

ALUNO

Nome	Departamento
Roberto Machado	2
Manuela Costa	3

INTERSEÇÃO

Nome	Depto
Manuela Costa	3

Operações DIFERENÇA

$$R_1 - R_2$$

- Operador → menos ($-$)
- Função → selecionar tuplas da primeira relação não contidas na segunda
- Operação **não comutativa**

$$R_1 - R_2 \neq R_2 - R_1$$

Exemplo → *PROFESSOR – ALUNO*

PROFESSOR

Nome	Depto
Roberto Machado	1
Manuela Costa	3

ALUNO

Nome	Departamento
Roberto Machado	2
Manuela Costa	3

DIFERENÇA

Nome	Depto
Roberto Machado	1

Operações DIVISÃO

$$R_1 \div R_2$$

- Operador → dividir ($\div$)
- Função → **extrair** subconjunto de tuplas de $R1$ presente em $R2$
- Relação resultante possui os atributos de $R1$ não presentes em $R2$
- Relação resultante possui as tuplas de $R1$ que contêm as tuplas de $R2$
- Operação **não comutativa**

$$R_1 \div R_2 \neq R_2 \div R_1$$

Operação DIVISÃO

Exemplo → *CARGA* ÷ *EXATAS*

Encontrar os professores que possuem cargas horárias em todos os cursos da área de ciências exatas

CARGA

Professor	Curso
Felipe	Administração
Max	Administração
Felipe	Computação
Max	Computação
Teldo	Computação
Luiz	Engenharia
Max	Engenharia

EXATAS

Curso
Computação
Engenharia

DIVISÃO

Professor
Max

SQL: DEFINIÇÃO DE DADOS

Structured Query Language

- Linguagem de consulta estruturada
- Linguagem declarativa de alto nível
- Usuário especifica o que deseja, deixando decisões sobre como executar a consulta para o SGBD
- Contém instruções para definição e manipulação de dados
- Padrão em SGBDs relacionais comerciais
- Mantém equivalência com o modelo relacional
  - Relação → Tabela
  - Tupla → Linha
  - Atributo → Coluna

SQL: Recursos

SQL oferece múltiplos recursos

- Definição de visões sobre dados
- Definição de restrições sobre os dados
- Especificação de controles de transações
- Especificação de autorizações e segurança

SQL: Definição de Dados

Existem diferentes instruções (comandos) para definição de dados

- **CREATE** → cria elementos no catálogo, como **esquemas**, **tabelas** e **domínios**
- **ALTER** → modifica elementos no catálogo
- **DROP** → remove elementos do catálogo

SQL: Definição de Dados

Esquema → elemento que agrupa outros elementos que pertencem à mesma aplicação de BD

- **Nome** → identificador do esquema
- **Proprietário** → usuário com autoridade sobre o esquema

Exemplo → criar esquema UNIVERSIDADE pertencente ao usuário 'Pedro'

```
CREATE SCHEMA UNIVERSIDADE AUTHORIZATION 'Pedro';
```

SQL: Definição de Dados

Catálogo → coleção nomeada de esquemas

- Contém o esquema padrão que oferece informação sobre todos os esquemas no catálogo, bem como sobre os descritores dos elementos
- Esquema padrão → *INFORMATION_SCHEMA*

Tabela Base → declarada por meio da instrução CREATE TABLE

- Tabela realmente criada e armazenada como um arquivo pelo SGBD

Tabela Virtual → declarada por meio da instrução CREATE VIEW

- Tabela pode ser criada e armazenada como um arquivo pelo SGBD
- Geralmente não são realmente armazenados em arquivo

CREATE TABLE

- Cria uma nova tabela, dando-lhe um nome e especificando suas colunas e restrições iniciais
- Restrições de tipo são geralmente especificadas
- Uma vez que colunas e suas respectivas restrições de tipo são criadas, podem ser redefinidas a partir da instrução **ALTER TABLE**

Exemplo:

```
CREATE TABLE PROFESSOR;
```

Esquema em que as tabelas são criadas é especificado implicitamente no ambiente em que as instruções CREATE TABLE são executadas

SQL: Restrições

Restrições podem ser especificadas em SQL como parte da criação de tabela

- Tipo → domínio de valores válidos para a coluna
- Nulidade → possibilidade de valor NULL em coluna
- Valor → faixa de valores válidas para uma coluna
- Valor Padrão → valor atribuído a uma coluna caso nenhum valor seja especificado
- Chave → coluna(s) identificadora(s) de uma instância
- Unicidade → coluna(s) candidata(s) a identificadora(s) de uma instância
- Integridade Referencial → regras para atualização de linhas correlacionadas em diferentes tabelas

SQL: Restrição de Tipo

Numérico → incluem números inteiros de vários tamanhos (INTEGER e SMALLINT) e números de ponto flutuante (reais) de várias posições (FLOAT, REAL e DOUBLE PRECISION)

Cadeias de Caracteres → incluem cadeias de caracteres de diferentes tipos

- CHAR(n) → cadeias de tamanho fixo, onde n é a quantidade exata de caracteres a ser armazenada
- VARCHAR(n) → cadeias de tamanho variável, onde n é a quantidade máxima de caracteres armazenados
- Valor literal da cadeia de caracteres deve ser especificado entre aspas simples, com maiúsculas diferenciadas de minúsculas (*case sensitive*)

SQL: Restrição de Tipo

Cadeias de Bits → incluem cadeias binárias de diferentes tipos

- BIT(n) → cadeias de tamanho fixo, onde n é a quantidade exata de bits a ser armazenada
- BIT VARYING(n) → cadeias de tamanho variável, onde n é a quantidade máxima de bits armazenados
- O valor literal da cadeia de bits deve ser especificado entre apóstrofos, precedidos por um B para distingui-los das cadeias de caracteres
  - Exemplo → B'10101'

Booleano → valores binários verdadeiro (1) e falso (0)

SQL: Restrição de Tipo

DATE & TIME → valores de data e hora

- **DATE** → dez posições compostas de dia, mês e ano na forma **DD-MM-YYYY**
- **TIME** → oito posições compostas de hora, minuto e segundo na forma **HH:MM:SS**

TIMESTAMP → valores temporais de alta precisão

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

SQL: Restrição de Nulidade

Pode ser especificada se valor NULL não for permitido para determinada coluna

- Implícito para colunas que fazem parte da chave primária

Exemplo:

```
CREATE TABLE PROFESSOR (
    CPF CHAR(11) NOT NULL,
    Nome VARCHAR(80) NOT NULL,
    Departamento INT
);
```

SQL: Restrição de Valor Padrão

Define valor padrão para uma coluna

- Valor padrão será incluído em qualquer nova linha se um valor explícito não for fornecido para essa coluna
- Se essa restrição não for especificada valor padrão será NULL para colunas que não possuem a restrição NOT NUL.

Exemplo:

```
CREATE TABLE PROFESSOR (
    CPF CHAR(11) NOT NULL,
    Nome VARCHAR(80) NOT NULL,
    Departamento INT DEFAULT 1
);
```

SQL: Restrição de Valor

Limita valores possíveis para coluna

Exemplo → supondo que números de departamento sejam restritos a inteiros entre 1 e 20, podemos modificar a tabela PROFESSOR, adicionando uma restrição para a coluna *Departamento*:

```
ALTER TABLE PROFESSOR ADD CHECK
(Departamento > 0 AND Departamento < 21);
```

SQL: Restrição de Chave

Especifica uma ou mais colunas que compõem a chave primária de uma tabela

- Se a chave primária for composta por apenas uma coluna, a cláusula PRIMARY KEY pode acompanhar a coluna diretamente

Exemplo:

```
CREATE TABLE PROFESSOR (
    CPF CHAR(11) NOT NULL,
    Nome VARCHAR(80) NOT NULL,
    Departamento INT,
    PRIMARY KEY (CPF)
);
```

SQL: Restrição de Unicidade

Especifica chaves secundárias alternativas

- Pode ser especificada diretamente para chave secundária em coluna única

Exemplo:

```
CREATE TABLE PROFESSOR (
    CPF CHAR(11) NOT NULL,
    Nome VARCHAR(80) NOT NULL,
    Departamento INT,
    PRIMARY KEY (CPF),
    UNIQUE (Nome)
);
```

SQL: Restrição de Integridade Referencial

Estabelece regras para restrição de atualização de linhas correlacionadas em diferentes tabelas através de referência à chave primária por chave estrangeira

Exemplo:

```
CREATE TABLE PROFESSOR (
    CPF CHAR(11) NOT NULL,
    Nome VARCHAR(80) NOT NULL,
    Departamento INT,
    PRIMARY KEY (CPF),
    FOREIGN KEY (Departamento)
    REFERENCES DEPARTAMENTO(Numero)
);
```

SQL: Restrição de Integridade Referencial

- A integridade referencial entre tabelas pode ser violada quando linhas são manipuladas ou o valor de uma chave primária é modificado
- Ação de disparo referencial especifica uma ação alternativa para os casos de violação de integridade:
  - RESTRICT → a linha da chave primária não pode ser modificada se houver linhas contendo chaves estrangeiras associadas a ela
  - CASCADE → a linha da chave primária, bem como as linhas contendo chaves estrangeiras são modificadas
  - SET NULL → a linha da chave primária é modificada, desde que se consiga atualizar para NULL as chaves estrangeiras associadas a ela
  - SET DEFAULT → a linha da chave primária é modificada, desde que as chaves estrangeiras associadas a ela possuam valor padrão que possa ser usado

SQL: Restrição de Integridade Referencial

Ações de disparo devem ser escolhidas em caso de remoção (**ON DELETE**) ou atualização (**ON UPDATE**)

Exemplo:

```
CREATE TABLE PROFESSOR (
    CPF CHAR(11) NOT NULL,
    Nome VARCHAR(80) NOT NULL,
    Departamento INT DEFAULT 1,
    PRIMARY KEY (CPF),
    FOREIGN KEY (Departamento)
    REFERENCES DEPARTAMENTO(Numero)
    ON DELETE SET DEFAULT ON UPDATE CASCADE
);
```

SQL: Restrições Nomeadas

Restrição pode ser rotulada utilizando o descritor **CONSTRAINT**

- Nomes de todas as restrições de um esquema precisam ser exclusivos

Exemplos:

```
CONSTRAINT PK_PROFESSOR PRIMARY KEY (CPF);

CONSTRAINT FK_DEPARTAMENTO_PROFESSOR
    FOREIGN KEY (Departamento)
    REFERENCES DEPARTAMENTO(Numero)
    ON DELETE SET DEFAULT ON UPDATE CASCADE;
```

SQL: MANIPULAÇÃO DE DADOS

Structured Query Language

- Linguagem de consulta estruturada
- Linguagem declarativa de alto nível
- Usuário especifica o que deseja, deixando decisões sobre como executar a consulta para o SGBD
- Contém instruções para definição e manipulação de dados
- Padrão em SGBDs relacionais comerciais
- Mantém equivalência com o modelo relacional
  - Relação → Tabela
  - Tupla → Linha
  - Atributo → Coluna

SQL: Manipulação de Dados

Existem diferentes instruções (comandos) para manipulação de dados

- **INSERT** → inserir linhas em tabelas
- **DELETE** → remover linhas de tabelas
- **UPDATE** → atualizar valores de colunas em linhas de tabelas
- **SELECT** → recuperar dados em tabelas

Instrução INSERT

Acrescenta uma linha em uma tabela

- Necessário especificar o nome da tabela e uma lista de valores para a linha
- Valores devem ser listados na mesma ordem em que as colunas correspondentes foram definidas na tabela

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento

```
INSERT INTO PROFESSOR VALUES
('12345678900', 'Ricardo Marini', 'M', 3000.00, 1);
```

Instrução INSERT

É possível especificar nomes de colunas correspondentes a valores fornecidos

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento

```
INSERT INTO PROFESSOR (CPF, Sexo, Nome, Departamento)
VALUES ('12345678900', 'M', 'Ricardo Marini', 1);
```

Coluna não especificada tem seu valor definido como DEFAULT ou NULL, sendo que valores e colunas devem ser listadas na mesma ordem

Se alguma restrição for violada a operação é rejeitada

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento
12345678900	Roberto Machado	M	1200.00	1
12345678901	Manuela Costa	F	2700.00	3
21345678900	Carlos A. Martins	M	3200.00	1
32145678900	Ana Maria Freitas	F	7500.00	2

```
INSERT INTO PROFESSOR (CPF, Nome, Sexo, Departamento)
VALUES ('68345618900', 'Amanda Ramirez', 'F', 4);
```

Caso não exista tupla na tabela DEPARTAMENTO com chave primária *Numero = 4* para manter integridade referencial com a coluna *Departamento* da tabela PROFESSOR a operação será rejeitada

Se alguma restrição for violada a operação é rejeitada

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento
12345678900	Roberto Machado	M	1200.00	1
12345678901	Manuela Costa	F	2700.00	3
21345678900	Carlos A. Martins	M	3200.00	1
32145678900	Ana Maria Freitas	F	7500.00	2

```
INSERT INTO PROFESSOR (Nome, Sexo, Departamento)
VALUES ('Amanda Ramirez', 'F', 1);
```

Valor da chave primária *CPF* não foi fornecido, o que viola a restrição de chave, logo a operação será rejeitada

Instrução INSERT

É possível inserir múltiplas linhas na tabela usando a instrução INSERT combinada com a instrução SELECT

```
INSERT INTO PROFESSOR (CPF, Nome, Sexo, Departamento)
SELECT CPF, Nome, Sexo, 1
FROM ALUNO;
```

Nesse caso, todas as linhas da tabela ALUNO serão inseridas na tabela PROFESSOR, sendo que para todas as linhas inseridas a coluna *Departamento* terá valor *1*

Instrução DELETE

Remove linhas de uma tabela

- Linhas são excluídas de apenas uma tabela
  - Exceção → exclusão pode se propagar para linhas em outras tabelas, de acordo com restrições de integridade referencial
- Condição (cláusula WHERE) inexistente especifica que todas as linhas na tabela serão excluídas
  - Tabela permanece no BD como uma tabela vazia

Instrução DELETE

Exemplo:

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento
12345678900	Roberto Machado	M	1200.00	1
12345678901	Manuela Costa	F	2700.00	3
21345678900	Carlos A. Martins	M	3200.00	1
32145678900	Ana Maria Freitas	F	7500.00	2

```
DELETE FROM PROFESSOR
WHERE Salario < 1000,00;
```

Instrução não removerá nenhuma linha da tabela

Instrução DELETE

Exemplo:

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento
12345678900	Roberto Machado	M	1200.00	1
12345678901	Manuela Costa	F	2700.00	3
21345678900	Carlos A. Martins	M	3200.00	1
32145678900	Ana Maria Freitas	F	7500.00	2

```
DELETE FROM PROFESSOR
WHERE Sexo = 'M';
```

Instrução removerá duas linha da tabela

Instrução DELETE

Exemplo:

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento
12345678900	Roberto Machado	M	1200.00	1
12345678901	Manuela Costa	F	2700.00	3
21345678900	Carlos A. Martins	M	3200.00	1
32145678900	Ana Maria Freitas	F	7500.00	2

```
DELETE FROM PROFESSOR;
```

Instrução removerá todas as linha da tabela

Instrução UPDATE

Modifica valores em colunas de uma ou mais linhas

- Cada instrução afeta apenas uma tabela
  - Exceção → atualização de uma chave primária pode ser propagada para os valores de chave estrangeira das linhas em outras tabelas de acordo com restrições de integridade referencial
- Cláusula SET especifica colunas a serem modificadas e seus novos valores

Instrução UPDATE

Exemplos:

```
UPDATE  PROFESSOR
SET     Salario = 2500,00, Departamento = 2
WHERE   CPF = '12345678900';
```

Altera o salário e o número do departamento do professor de determinado CPF

```
UPDATE  PROFESSOR
SET     Salario = Salario * 1.1;
```

Aumenta em 10% o salário de todos os professores

Instrução SELECT

Recupera linhas em múltiplas tabelas

- MAPEAMENTO → forma básica da instrução SELECT, também chamado de BLOCO SELECT-FROM-WHERE

  SELECT <lista de colunas>
  FROM <lista de tabelas>
  WHERE <condição>;

- **<lista de colunas>** → lista de nomes de colunas que valores devem ser recuperados pela consulta
- **<lista de tabelas>** → lista dos nomes de tabelas necessárias para processar a consulta
- **<condição>** → expressão condicional que identifica linhas que devem ser recuperadas pela consulta

Instrução SELECT

Exemplo → recuperar o nome e o salário de todos os professores do sexo masculino do departamento de número 1

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento
12345678900	Roberto Machado	M	1200.00	1
12345678901	Manuela Costa	F	2700.00	3
21345678900	Carlos A. Martins	M	3200.00	1
32145678900	Ana Maria Freitas	F	7500.00	2

```
SELECT  Nome, Salario
  FROM  PROFESSOR
 WHERE  Departamento = 1
   AND  Sexo = 'M';
```

Exemplo → recuperar o CPF e o nome dos professores do sexo masculino que também são alunos

```
SELECT  A.CPF, A.Nome
  FROM  PROFESSOR A, ALUNO B
 WHERE  A.CPF = B.CPF
   AND  A.Sexo = 'M';
```

Resultado:

CPF	Nome
12345678900	Roberto Machado
21345678900	Carlos A. Martins

Instrução SELECT

Exemplo → recuperar o nome do departamento e do professor para todos os professores que são alunos e que trabalham no departamento de nome *Pesquisa*

```
SELECT  A.Nome AS Departamento, B.Nome AS Professor
  FROM  DEPARTAMENTO A, PROFESSOR B, ALUNO C
 WHERE  A.Numero = B.Departamento
   AND  B.CPF = C.CPF
   AND  A.Nome = 'Pesquisa';
```

Resultado:

Departamento	Professor
Pesquisa	Roberto Machado
Pesquisa	Carlos A. Martins

Instrução SELECT

Junções podem ser especificadas tanto na cláusula **WHERE** quanto na cláusula **FROM** com o uso do operador **JOIN**

```
SELECT  A.CPF, A.Nome
  FROM  PROFESSOR A, ALUNO B
 WHERE  A.CPF = B.CPF
   AND  A.Sexo = 'M';

SELECT  A.CPF, A.Nome
  FROM  PROFESSOR A JOIN ALUNO B ON A.CPF = B.CPF
 WHERE  A.Sexo = 'M';
```

Variações do operador de junção podem ser especificados, como **INNER JOIN**, **LEFT OUTER JOIN** e **FULL JOIN**

Ambiguidade

Mesmo nome pode ser usado em mais de uma coluna, desde que as colunas pertençam a tabelas diferentes e estejam devidamente prefixadas para evitar ambiguidade

```
SELECT  PROFESSOR.Nome, ALUNO.Nome
  FROM  PROFESSOR, ALUNO
 WHERE  PROFESSOR.CPF = ALUNO.CPF
   AND  PROFESSOR.Sexo = 'M';

SELECT  A.Nome, B.Nome
  FROM  PROFESSOR A, ALUNO B
 WHERE  A.CPF = B.CPF
   AND  A.Sexo = 'M';
```

Ausência de Cláusula WHERE

Inexistência de condições para seleção e junção de linhas traz impactos diferentes no resultado das consultas

- Tabela Única → todas as linhas da única tabela especificada na cláusula FROM são retornadas

```
SELECT  CPF
  FROM  PROFESSOR;
```

- Múltiplas Tabelas → todas as combinações possíveis entre linhas das tabelas especificadas na cláusula FROM são retornadas, equivalendo à operação **Produto Cartesiano** da álgebra relacional

```
SELECT  A.CPF
  FROM  PROFESSOR A, DEPARTAMENTO B;
```

Duplicatas

Uma tabela constitui um multiconjunto e linhas duplicadas podem aparecer no resultado de uma consulta

- DISTINCT → elimina linhas duplicadas no resultado

```
SELECT  Departamento
  FROM  PROFESSOR
 WHERE  Salario < 5000.00;
```

Departamento
1
3
1

```
SELECT  DISTINCT Departamento
  FROM  PROFESSOR
 WHERE  Salario < 5000.00;
```

Departamento
1
3

Operadores Especiais

Asterisco (*) → recupera todas as colunas das linhas selecionadas sem a necessidade de listar seus nomes explicitamente

```
SELECT  *
  FROM  PROFESSOR
 WHERE  Departamento = 1
```

Nesse caso, recupera todas as colunas de professores que trabalham no departamento de número 1

Operadores Especiais

LIKE → comparação sobre subcadeias de caracteres

- Subcadeias são especificadas usando dois caracteres especiais
  - % substitui zero ou mais caracteres
  - _ substitui um único caracter

```
SELECT  CPF, Nome
  FROM  PROFESSOR
 WHERE  Endereco LIKE '%Belo Horizonte%';
```

Recupera o CPF e nome de todos os professores em que seu endereço contenha a subcadeia de caracteres *Belo Horizonte*

Operadores Especiais

BETWEEN → comparação com intervalos

- Valores para colunas comparadas devem estar entre um intervalo de valores

```
SELECT  *
  FROM  PROFESSOR
 WHERE  Salario BETWEEN 2000,00 AND 5000,00;
```

Recupera todas as colunas de professores com salários entre 2 e 5 mil

Ordenação de Resultados

ORDER BY → ordena linhas do resultado de uma consulta

- ASC → operador **padrão** para ordenação crescente
- DESC → operador para ordenação decrescente

```
SELECT      A.Nome, B.Nome
  FROM      DEPARTAMENTO A, PROFESSOR B
 WHERE      A.Numero = B.Departamento
ORDER BY    B.Nome, A.nome DESC;
```

Recupera o nome do departamento e do professor para todos os professores que trabalham em um departamento, ordenando o resultado de maneira crescente pelo nome do professor e decrescente pelo nome do departamento

Valores NULL

IS NULL (IS NOT NULL) → verifica se valor de coluna é NULL

- NULL tem semântica imprecisa
  - Valor desconhecido?
  - Valor indisponível?
  - Valor não aplicável?

```
SELECT      CPF, Nome
  FROM      PROFESSOR
 WHERE      Departamento IS NULL;
```

Recupera o CPF e o nome dos professores que não trabalham em algum departamento

Consulta Aninhada

Bloco SELECT completo na cláusula WHERE de outra consulta, denominada **consulta externa**

- IN (NOT IN) → verifica se um conjunto de valores pertence a um multiconjunto de valores

```
SELECT  A.Nome, A.Salario
  FROM  PROFESSOR A
 WHERE  (A.CPF, A.Nome) IN
        (SELECT B.CPF, B.Nome
           FROM ALUNO B
          WHERE A.Sexo = B.Sexo);
```

Recupera nome e salário dos professores que possuem mesmo CPF e nome de algum aluno, desde que tenham o mesmo sexo

Consulta Aninhada

Bloco SELECT completo na cláusula WHERE de outra consulta, denominada **consulta externa**

- EXISTS (NOT EXISTS) → verifica se o resultado da consulta interna é conjunto vazio

```
SELECT  A.CPF, A.Nome
  FROM  PROFESSOR A
 WHERE  NOT EXISTS
        (SELECT  *
           FROM  DEPARTAMENTO B
          WHERE  A.Departamento = B.Numero
            AND  B.Nome = 'Pesquisa');
```

Retorna CPF e nome dos professores que não trabalham no departamento de nome *Pesquisa*

Agregação

GROUP BY → agrupa múltiplas linhas em uma utilizando **função de agregação**

- **COUNT** → conta o número de linhas agrupadas
- **SUM** → soma o valor na coluna de linhas agrupadas
- **MAX** → retorna o valor máximo na coluna de linhas agrupadas
- **MIN** → retorna o valor mínimo na coluna de linhas agrupadas
- **AVG** → retorna a média dos valores na coluna de linhas agrupadas

Funções de agregação não têm efeito em linhas com colunas participantes da função com valor NULL

Agregação

Exemplos:

```
SELECT  COUNT(*), SUM(Salario)
  FROM  PROFESSOR
 WHERE  Sexo = 'F';
```

Retorna o número de professores do sexo feminino e o salário pago a elas

```
SELECT      Departamento, COUNT(*), SUM(Salario),
            MAX(Salario), MIN(Salario), AVG(Salario)
  FROM      PROFESSOR
GROUP BY    Departamento;
```

Para cada departamento retorna seu número, a quantidade de professores, a soma de salários, o salário máximo e mínimo e a média salarial

Agregação

HAVING → remove linhas do resultado agregado de acordo com condição imposta sobre as funções de agregação

```
SELECT      Departamento, COUNT(*), SUM(Salario)
  FROM      PROFESSOR
GROUP BY    Departamento
HAVING      AVG(Salario) > 8000.00;
```

Para cada departamento retorna seu número, a quantidade de professores e a soma de salários, desde que a média salarial do departamento seja maior que 8000,00

Resumo de Instruções

```
INSERT INTO <tabela>
[(<lista de atributos>)]
VALUES (<lista de valores>);

UPDATE <tabela>
SET <lista de atribuicoes>
[WHERE <condicao>];

DELETE FROM <tabela>
[WHERE <condicao>];

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

ARMAZENAMENTO EM MEMÓRIA

Armazenamento em Memória

BDs são armazenados fisicamente em meios (mídias) de armazenamento computacional

- Meios de armazenamento formam uma **hierarquia**, em que dados residem e transitam, sendo que a hierarquia reflete a *distância* do meio à CPU
  - Memória primária → *próxima* e operada diretamente pela CPU
  - Memória secundária → *distante* e não operada pela CPU
  - Memória terciária → *muito distante* e não operada pela CPU
- Programas residem e são executados em memória primária
- BDs são geralmente grandes e persistem em memória secundária
- SGBD transfere partes do BD entre memórias de acordo com a necessidade

Armazenamento em Memória

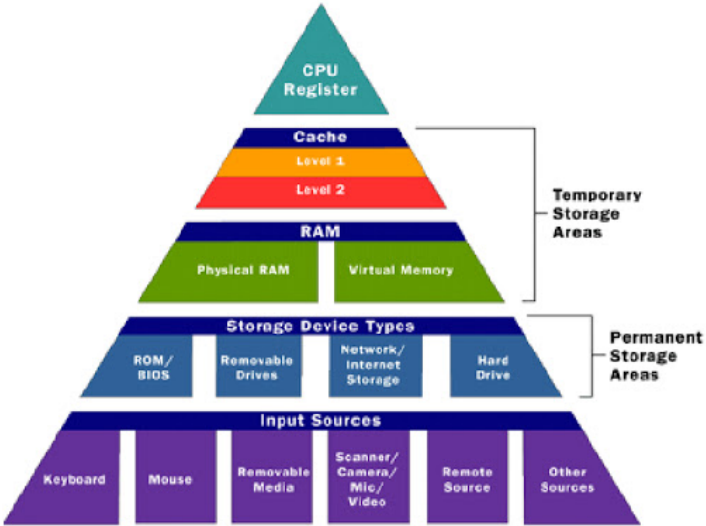
Existe uma correlação entre capacidade de armazenamento, velocidade de transferência e custo em meios de armazenamento

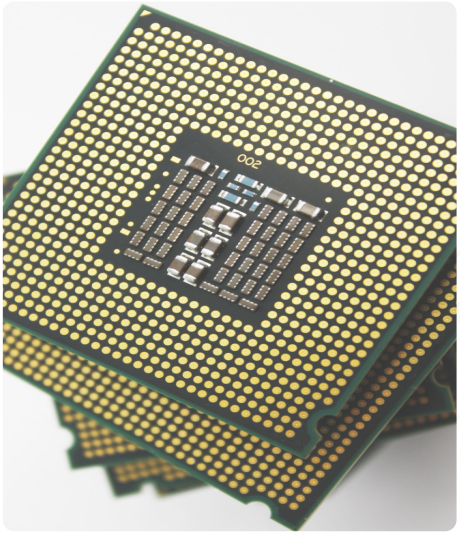
- Capacidade de armazenamento → quantidade de dados (bytes) que podem ser armazenados na memória
- Velocidade de transferência → quantidade de dados (bits) que podem ser transferidos de ou para a memória por unidade de tempo (segundo)
- Custo → unidade monetária ($) por quantidade de dados (bytes) que podem ser armazenados na memória

Correlação:

- > capacidade ⇒ < velocidade
- > velocidade ⇒ > custo

Hierarquia de Memória





Registrador

Memória eletrônica

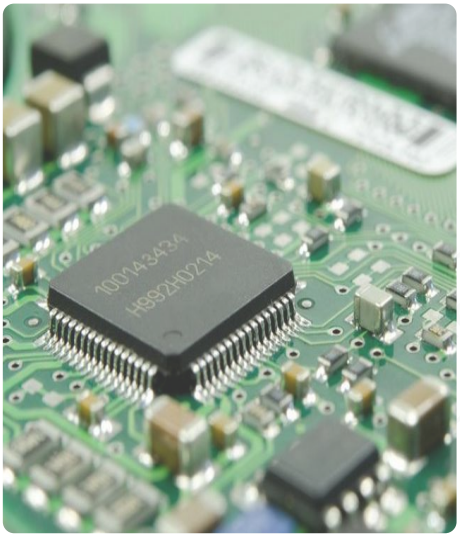
Interna da CPU

Rápida $\rightarrow \approx$ 60 Tbps

Pequena $\rightarrow$ centenas de bytes

Cara $\rightarrow >$ 500 R$/MB

Utilizada para execução de instruções de programa



Cache

Memória eletrônica

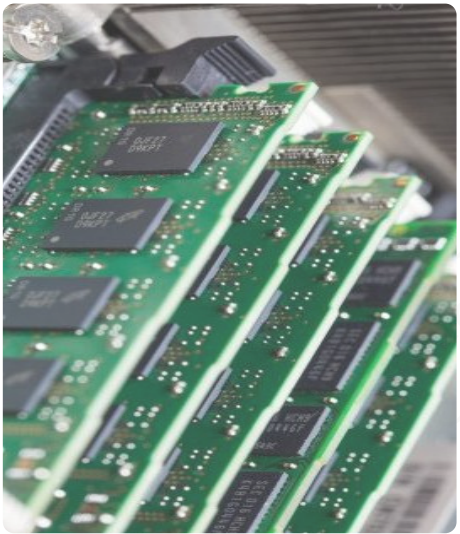
Vários níveis → L0 a L4

Rápida → L1 ≈ 6 Tbps

Pequena → L4 ≈ 128 MB

Cara → L0 > 100 R$/MB

Acelera a execução de instruções de programa (pré-busca e *pipelining*)



RAM

Memória eletrônica

Acesso aleatório

Rápida $\rightarrow \approx$ 80 Gbps

Pequena $\rightarrow$ dezenas de GB

Cara $\rightarrow \approx$ 0,05 R$/MB

Utilizada para manter instruções de programa e dados temporários

Flash

Memória eletrônica

Resistente e durável

Rápida $\rightarrow \approx$ 5 Gbps

Média $\rightarrow$ alguns TB

Barata $\rightarrow \approx$ 0,0007 R$/MB

Utilizada para manter dados de maneira persistente

HD

Memória magnética

Discos em alta rotação

Lenta $\rightarrow \approx$ 100 Mbps

Grande $\rightarrow$ dezenas de TB

Barata $\rightarrow \approx$ 0,0002 R$/MB

Utilizada para manter dados de maneira persistente

Fita

Memória magnética removível

Acesso sequencial

Lenta $\rightarrow \approx$ 2 Mbps

Grande $\rightarrow$ PB (jukebox)

Barata $\rightarrow \approx$ 0,00003 R$/MB

Utilizada para manter dados pouco mutáveis e acessados de maneira persistente, como *backups*

Óptica

Memória removível

Discos ópticos

Lenta $\rightarrow \approx$ 20 Mbps

Grande $\rightarrow$ PB (jukebox)

Barata $\rightarrow \approx$ 0,0001 R$/MB

Utilizada para manter dados pouco mutáveis e de acesso sequencial de maneira persistente, como multimídia

Comparativo entre diferentes tipos de memória:

Tipo	Nome	Velocidade (bps)	Capacidade	Custo (R$/MB)	Volátil
CPU	Registrador	60T	KB	500	sim
Primária	Cache	6T	MB	100	sim
Primária	RAM	80G	GB	0,05	sim
Secundária	Flash	5G	TB	0,0007	não
Secundária	HD	100M	TB	0,0002	não
Terciária	Óptico	20M	PB	0,0001	não
Terciária	Fita	2M	PB	0,00003	não

Os valores de velocidade, capacidade e custo são estimativas, a fim de fornecer uma ordem de grandeza. Estimativas foram baseadas em memórias disponíveis atualmente, podendo variar de acordo com a tecnologia e o fabricante

Armazenamento em Memória

Em sistemas de banco de dados, os dados são efetivamente armazenados em diferentes tipos de memória de acordo com sua natureza

- TRANSIENTES → persistem em memória por um período limitado de tempo, apenas durante a execução do programa
- PERSISTENTES → permanecem em memória por longos períodos de tempo, sendo acessados e processados repetidamente durante esse período

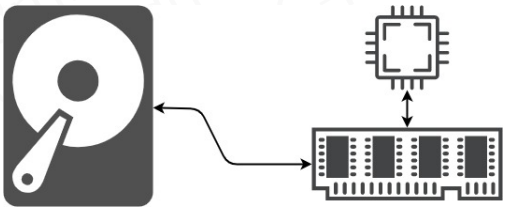
SGBDs devem ser capazes de gerenciar eficientemente a transferência de dados transientes e permanentes entre memórias

- No **PROJETO FÍSICO**, DBAs e projetistas devem escolher as melhores técnicas de organização de dados para garantir equilíbrio entre custo e desempenho, atendendo aos requisitos funcionais e operacionais do BD

Armazenamento em Memória

Aplicações tipicamente necessitam de apenas uma pequena parte do BD de cada vez para processamento, sendo responsabilidade do SGBD garantir que:

1. A parte seja transferida da memória secundária para a primária
2. A CPU processe os dados em memória primária adequadamente
3. Os dados processados sejam transferidos de volta à memória secundária

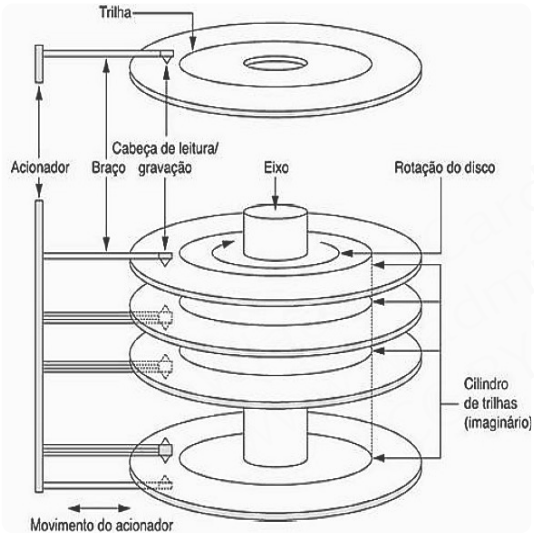


Armazenamento em Memória

Tipicamente BDs são armazenados de maneira permanente em discos magnéticos

- BDs são muito grandes para caberem inteiramente em memória primária, com capacidade limitada de armazenamento
- Custo de armazenamento em memória primária é muito alto
- Memórias terciárias tem grande capacidade de armazenamento e baixo custo, mas são muito lentas e frequentemente demandam intervenção manual (*off-line*)
- Discos magnéticos apresentam excelente relação custo-benefício, ainda mais vantajosa que outros tipos de memória secundária

Disco Magnético (HD)



Acesso aleatório

Múltiplas superfícies

Armazenamento em TRILHAS

Trilhas divididas em BLOCOS

Tamanho do bloco é fixado na formatação do HD e não pode ser trocado dinamicamente

Transferências entre memória primária e HD ocorrem em unidades de bloco

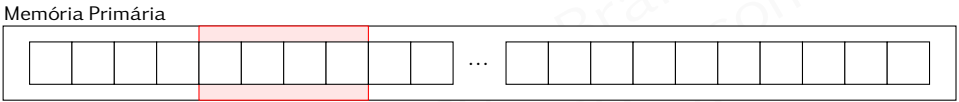
Disco Magnético (HD)

Bloco (página) → unidade mínima de transferência de dados entre disco e memória primária

- Tamanho fixado na formatação, geralmente entre 512B a 8KB, que não pode ser alterado dinamicamente
- Separados nas trilhas por **lacunas** de tamanho fixo que incluem dados de controle, como ponteiro para o bloco subsequente
- Pode ser acessado aleatoriamente pelo seu endereço de hardware, denominado **endereço de bloco**
- Hardware controladores de disco usam o endereço do bloco para transferir o bloco do disco para um *buffer* em memória primária

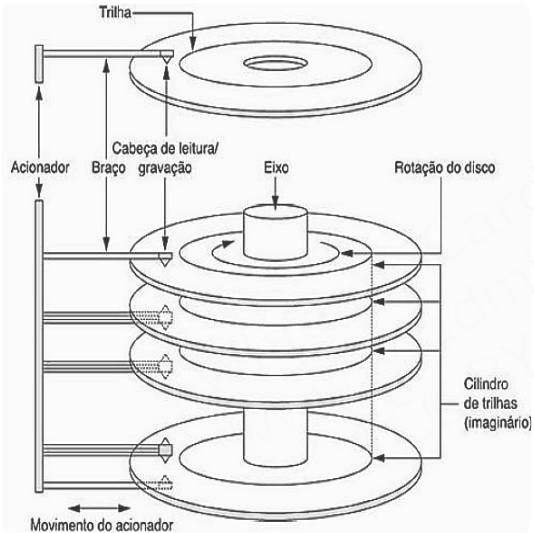
Disco Magnético (HD)

Buffer → área reservada contígua em memória primária



- Controladores de disco usam o endereço de bloco e de *buffer* para realizar a transferência do bloco de disco para a memória primária
  - Leitura (Input) → bloco é copiado para *buffer*
  - Escrita (Output) → *buffer* é copiado para bloco

HD: Leitura e Escrita (I/O)



1) Controlador recebe endereços de bloco e *buffer*
2) Controlador comanda acionador a movimentar braço para posicionar cabeça na trilha do endereço de bloco
3) Discos giram até o ponto de leitura e escrita
4) Dados são copiados de ou para *buffer*

HD: Leitura e Escrita (I/O)

Tempo de Transferência → tempo necessário para transferir um bloco entre disco e memória primária

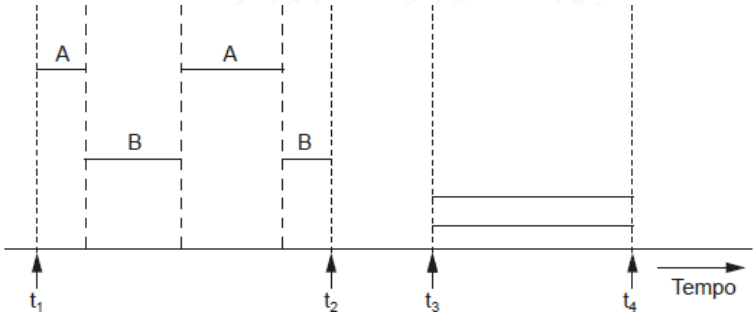
- Tempo de Busca → tempo necessário para posicionar a cabeça de leitura e escrita na trilha do endereço de bloco
- Tempo de Latência → ou atraso rotacional é o tempo necessário para o disco girar até o ponto de leitura e escrita
- Tempo de Transferência de Bloco → tempo necessário para os dados serem copiados de ou para o *buffer* em memória primária

Transferência de Bloco ≪ Busca + Latência

- Transferir múltiplos blocos consecutivos na mesma trilha ou cilindro elimina tempos de busca e latência acumulados, tornando a transferência mais eficiente

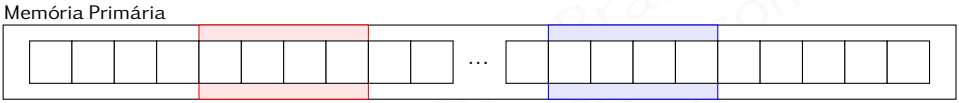
Buffering de Blocos → técnica que reserva vários *buffers* em memória primária para agilizar a transferência de blocos do disco

- Controladores de disco e CPUs podem operar de forma independente e paralela usando *buffers* diferentes



HD: Leitura e Escrita (I/O)

Duplo Buffering → uso de dois *buffers* para leitura ou gravação em disco



- Enquanto o controlador de disco transfere dados de ou para um *buffer*, a CPU processa dados no outro *buffer*
- Permite leitura ou gravação contínua em blocos consecutivos
- Elimina tempos de busca e latência para todas as transferências de bloco, com exceção da primeira
- Dados ficam prontos para processamento mais rapidamente, reduzindo ociosidade da CPU e, consequentemente o tempo de espera das aplicações

HD: Alocação de Blocos

A forma como os blocos são alocados em disco impacta o desempenho de I/O

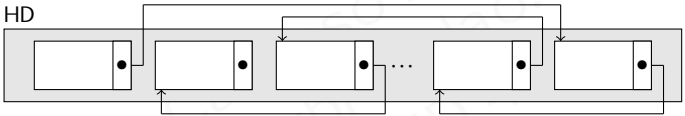
- Alocação Contígua → blocos consecutivos em disco

  - Rápido I/O com *duplo buffering*
  - Difícil expansão, podendo resultar em múltiplas realocações em caso de alteração dos dados

HD: Alocação de Blocos

A forma como os blocos são alocados em disco impacta o desempenho de I/O

- Alocação por Ligação → cada bloco contém um ponteiro para o próximo

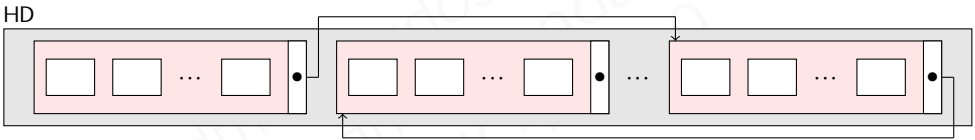


  - Facilita expansão
  - I/O mais lento pela impossibilidade de uso de *duplo buffering*

HD: Alocação de Blocos

A forma como os blocos são alocados em disco impacta o desempenho de I/O

- Alocação por Segmento → agrupa blocos consecutivos em segmentos (*clusters*) e cada segmento contém um ponteiro para o próximo segmento

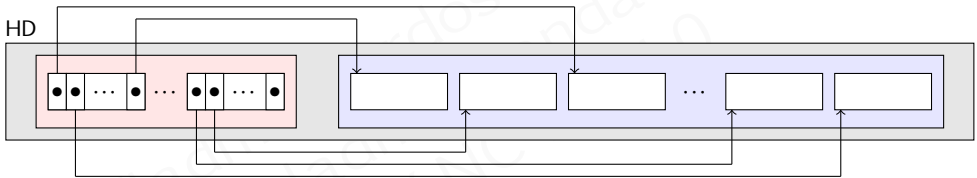


  - Combinação de alocação contígua e por ligação
  - Torna *duplo buffering* viável em um segmento, agilizando I/O
  - Facilita expansão, reduzindo o número de realocações em caso de alteração dos dados

HD: Alocação de Blocos

A forma como os blocos são alocados em disco impacta o desempenho de I/O

- Alocação Indexada → blocos especiais de índice são criados contendo ponteiros para blocos de dados



  - Rápido I/O com busca sendo efetuada em blocos de índice, que podem ter alocação contígua ou por segmento (*duplo buffering*)
  - Fácil expansão, com realocações ocorrendo em blocos de índice

ORGANIZAÇÃO DE DADOS

Organização de Dados

A forma como dados são dispostos em memória secundária impacta o desempenho do SGBD para recuperação e manipulação desses dados

- Tipicamente dados são organizados como **ARQUIVOS DE REGISTROS**

REGISTRO → coleção de valores relacionados a fatos sobre o minimundo, tais como atributos, instâncias de entidades e relacionamentos

12345678900	Roberto Machado	M	1200.00	1

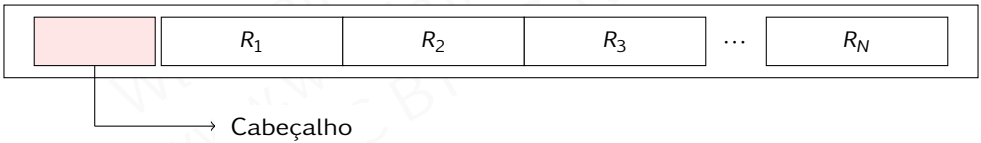
ARQUIVO → coleção de registros relacionados

12345678900	Roberto Machado	M	1200.00	1	⋯	32145678900	Ana Maria Freitas	F	7500.00	2

Registros devem ser organizados de forma a serem rapidamente localizados

Um ARQUIVO possui um **cabeçalho (descritor)** contendo metadados úteis aos programas que acessam seus registros

- Ordem, tipo e tamanho de campos dos registros
- Endereços dos blocos que armazenam registros do arquivo
- Códigos de caracteres especiais, como separadores de campos



Arquivos de Registros

Cada valor de um **REGISTRO** está restrito a um **tipo de dado**, sendo que o número de bytes para cada tipo é fixo, dependendo do sistema computacional

- BOOLEANO: 1 byte
- INTEIRO: 4 bytes
- NÚMERO REAL: 4 bytes
- INTEIRO LONGO: 8 bytes
- DATA: 10 bytes (formato DD-MM-AAAA)
- STRING: n bytes, onde n é o número de caracteres
- BLOB: $p + n$ bytes, onde p é o tamanho do ponteiro no registro para o endereço de bloco onde o objeto binário de tamanho n está armazenado

Arquivos de Registros

Arquivos podem ser compostos por registros de **tamanho**:

- Fixo → cada registro no arquivo tem o mesmo tamanho
- Variável → registros no arquivo possuem tamanhos diferentes
  - Campos de tamanho variável → VARCHAR, TEXT
  - Campos opcionais → NULL
  - Campos multivalorados
  - Arquivos mistos com registros de instâncias de entidades diferentes

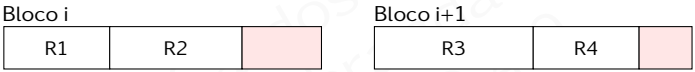
Tipicamente, todos os registros em um arquivo referem-se às instâncias de uma mesma entidade

- Arquivo Professor → Entidade Professor

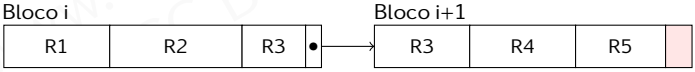
Arquivos de Registros

Um **ARQUIVO** é alocado em diferentes blocos de disco, sendo que seus **REGISTROS** podem estar alocados em um ou vários blocos

- NÃO ESPALHADO → registro não pode atravessar o limite de um bloco



- ESPALHADO → registro é armazenado em múltiplos blocos
  - Ponteiro no fim de cada bloco aponta para o bloco de continuidade do registro



Blocagem, ou **Fator de Bloco**, ou **Fator de Blocagem** de um arquivo é a quantidade de registros desse arquivo que cabem em um bloco de disco

Considere:

- Blocos com t bytes
- Registros de r bytes, sendo $r \leq t$
- Fator de Bloco $\rightarrow F = \left\lfloor \frac{t}{r} \right\rfloor$

Em arquivo com registros de tamanho fixo, r é o tamanho do registro, enquanto em arquivo com registros de tamanho variável, considera-se r o tamanho médio de registros

Se r é suficientemente grande, tal que $r > t/2$, o espaço não usado em disco pode ser grande e o espalhamento de registros passa a ser vantajoso para reduzir esse *"espaço perdido"*

- Espaço não usado $\rightarrow U = t - (F \times r)$

O **número de blocos** (B) necessários para armazenar um arquivo é:

- $B = \left\lceil \frac{n}{F} \right\rceil$, onde n é o número de registros do arquivo

Por exemplo, considere um arquivo de Professor armazenado em um disco com blocos de $t = 4KB$, onde:

- $r = 185B$
- $n = 10.000$

Nesse caso, teremos:

- $F = \left\lfloor \frac{4KB}{185B} \right\rfloor = \left\lfloor \frac{4 \times 1.024B}{185B} \right\rfloor \approx \lfloor 22,14 \rfloor = 22$
- $U = 4KB - (22 \times 185B) = 4.096B - 4.070B = 26B$
- $B = \left\lceil \frac{10.000}{22} \right\rceil \approx \lceil 454,54 \rceil = 455$
- Consumo de espaço $\rightarrow 455 \times 4KB = 1.820KB \approx 1,77MB$

Métodos de Acesso

Grupo de operações que podem ser aplicadas a um arquivo

- Recuperação → localização de registros em arquivo para que valores de campos possam ser lidos e processados, sem que haja alteração nos dados
- Atualização → alteração de arquivo pela inserção ou exclusão de registros, ou pela modificação de valores de campos de registros

A frequência da mudança em arquivos determina a frequência de execução de operações de atualização

- Arquivos Estáticos → operações de atualização são raramente executadas
- Arquivos Dinâmicos → mudam com frequência de forma que operações de atualização são constantemente executadas

Métodos de Acesso

Muitas operações aplicadas a arquivos envolvem **PESQUISA**

- Especifica critérios que registros devem satisfazer
- Tipicamente, critérios envolvem expressões booleanas
- Expressões podem apresentar diferentes graus de complexidade
  - SIMPLES → expressões booleanas simples
    - Exemplo: (Sexo = 'M')
  - COMPLEXAS → expressões booleanas complexas
    - Exemplo: ((Sexo = 'M') ∧ ((Salario > 3.000) ∨ (¬ TemDependente)))

Métodos de Acesso

SGBDs acessam registros utilizando **operações representativas**, em que tipicamente um registro é processado por vez

- Open → prepara arquivo para leitura ou escrita
  - Aloca *buffers* para blocos
  - Recupera o cabeçalho do arquivo
  - Posiciona o ponteiro de arquivo no início do arquivo
- Reset → posiciona o ponteiro do arquivo aberto para o início do arquivo
- Close → libera *buffers* alocados e realiza operações de limpeza de memória

Operações representativas

- Find (Locate) → procura o primeiro registro que satisfaça uma condição
  - Transfere o bloco que contém o registro para um *buffer* alocado
  - Posiciona o ponteiro de arquivo no registro, tornando-o o registro atual
- FindNext → procura o próximo registro que satisfaça uma condição
  - Transfere o bloco que contém o registro para um *buffer* alocado
  - Posiciona o ponteiro de arquivo no registro, tornando-o o registro atual
- Read (Get) → copia o registro do *buffer* para uma variável de programa
  - Posiciona o ponteiro no próximo registro, tornando-o o registro atual

Operações representativas

- Delete → remove o registro atual
  - Transfere o *buffer* de volta ao bloco no disco
- Modify → modifica valores de campos do registro atual
  - Transfere o *buffer* de volta ao bloco no disco
- Insert → insere um novo registro no arquivo
  - Localiza o bloco onde o registro deve ser inserido
  - Transfere o bloco para um *buffer*
  - Escreve o registro no *buffer*
  - Transfere o *buffer* de volta ao bloco no disco

Operações representativas

- Scan → combinação das operações Find, FindNext e Read
  - Se uma condição é especificada
    - Se o ponteiro de arquivo estiver posicionado no início do arquivo, reposiciona-o no primeiro registro que satisfaça a condição
    - Se o ponteiro de arquivo estiver posicionado em algum registro, reposiciona-o no próximo registro que satisfaça a condição
  - Caso contrário
    - Se o ponteiro de arquivo estiver posicionado no início do arquivo, reposiciona-o no primeiro registro
    - Se o ponteiro de arquivo estiver posicionado em algum registro, reposiciona-o no próximo registro

Métodos de Acesso

Existem **operações representativas** de nível mais alto que podem ser aplicadas a conjuntos de registros

- FindAll → procura o conjunto de registros que satisfaça uma condição
- FindOrdered → procura, em uma ordem específica, o conjunto de registros que satisfaça uma condição
- FindN → procura os N primeiros registros que satisfaçam uma condição
- Reorganize → reorganiza os blocos e registros de um arquivo

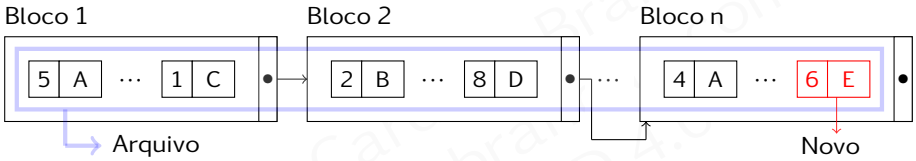
Tipos de Arquivo

Métodos de acesso operam de maneira diferente dependendo da forma como arquivos são organizados, especialmente de como os registros encontram-se dispostos dentro dos arquivos

- **Arquivo Heap (Pilha)** → registros posicionados sem ordem, com novos registros acrescentados ao final do arquivo
- **Arquivo Sequencial** → registros posicionados ordenadamente por um ou mais campos, denominados *campos de ordenação*
- **Arquivo Hash** → registros posicionados a partir da aplicação de uma *função hash* sobre um ou mais campos, denominados *campos hash*

Tipos de Arquivo: Heap

Arquivo organizado de forma que os registros são dispostos desordenadamente



- Pesquisa → linear, varrendo todos os registros do arquivo no pior caso
  - Endereço do primeiro bloco do arquivo é recuperado do cabeçalho
  - Começando do primeiro, cada bloco é copiado para um *buffer*, onde deve-se verificar se cada registro do bloco satisfaz os critérios de pesquisa
  - Complexidade → $O(n)$, onde n é o número de blocos do arquivo

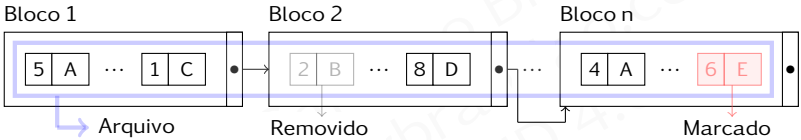
Tipos de Arquivo: Heap

Inserção eficiente, mas operações de alteração demandam pesquisa para encontrar o registro a ser alterado

- Inserção → registro arquivado na ordem em que é inserido
  - Endereço do último bloco do arquivo é recuperado do cabeçalho
  - Bloco copiado para um *buffer*, onde o novo registro é acrescentado, e o *buffer* é copiado de volta ao bloco
- Alteração → pode resultar em exclusão seguida de inclusão, caso o registro aumente de tamanho
  - Endereço do bloco do arquivo é recuperado via pesquisa
  - Bloco copiado para um *buffer*, onde o registro é modificado, e o *buffer* é copiado de volta ao bloco

Tipos de Arquivo: Heap

Operações de exclusão resultam em desperdício de espaço no bloco, demandando reorganização periódica do arquivo



- Exclusão → efetuada diretamente ou por marcação
  - Endereço do bloco do arquivo é recuperado via pesquisa
  - Direta → bloco copiado para um *buffer*, onde o registro é removido, e o *buffer* é copiado de volta ao bloco
  - Marcação → cada registro possui um byte extra, denominado marcador de exclusão. Assim, o bloco é copiado para um *buffer*, onde o marcador de exclusão do registro é modificado, e o *buffer* é copiado de volta ao bloco

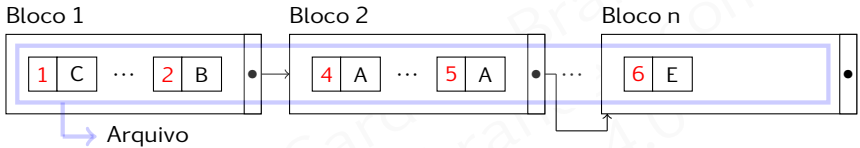
Tipos de Arquivo: Heap

Arquivo Direto (Relativo) → arquivo heap com registros de tamanho fixo, não espalhados, com blocos em alocação contígua

- Acesso simples a qualquer registro pela posição relativa no arquivo
- Não ajuda na pesquisa baseada em critérios
- Facilita a construção de índices no arquivo

Tipos de Arquivo: Sequencial

Arquivo organizado de forma que os registros são dispostos ordenadamente

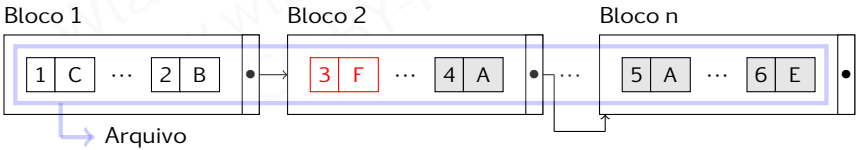


- Blocos em cilindros contíguos, minimizando tempo de busca
- Pesquisa → binária, varrendo pequena quantidade de registros se a pesquisa for feita com operadores $< \leq = > \geq$ sobre os campos de ordenação
  - Blocos intermediários são recuperados e segmentos são descartados até se encontrar registros que satisfaçam os critérios de pesquisa
  - Complexidade → $O(\log_2 n)$, onde n é o número de blocos do arquivo

Tipos de Arquivo: Sequencial

Operações de alteração são onerosas, pois podem demandar reorganização dos registros para preservação de ordem

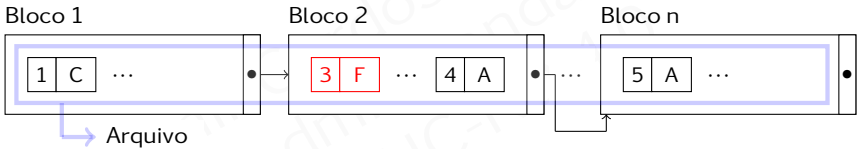
- Inserção → registro deve ser inserido na posição correta
  - Endereço do bloco onde registro deve ser inserido é recuperado via pesquisa
  - Deslocam-se registros para posições subsequentes, abrindo-se espaço para o registro a ser inserido
  - Blocos modificados nos deslocamentos são gravados de volta no disco



Tipos de Arquivo: Sequencial

Existem alternativas para desonerar a inclusão

- Espaços Vazios → diminui deslocamentos, mas problema reaparece com espaços vazios totalmente preenchidos



- Arquivo Temporário (Overflow) → arquivo heap onde novos registros são inseridos a um baixo custo
  - Periodicamente arquivo temporário é mesclado ao arquivo principal
  - Maior complexidade de pesquisa com necessidade de pesquisa linear no arquivo temporário caso o registro não seja encontrado no arquivo principal

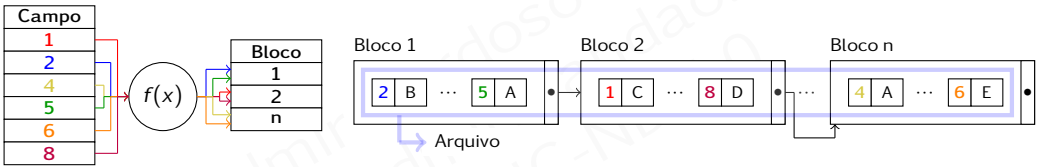
Tipos de Arquivo: Sequencial

Alteração no registro pode demandar seu reposicionamento

- Alteração → dependente da condição de pesquisa e do campo a ser alterado
  - Endereço do bloco do arquivo é recuperado via pesquisa
    - Pesquisa binária → critério envolver os campos de ordenação
    - Pesquisa linear → caso contrário
  - Bloco copiado para um *buffer*
    - Campo de ordenação não modificado → modifica-se o registro no *buffer*, copiando o *buffer* de volta ao bloco
    - Campo de ordenação modificado → deslocam-se registros gravando todos os blocos modificados
- Exclusão → igualmente dependente da condição de pesquisa

Tipos de Arquivo: Hash

Arquivo organizado de forma que os registros são distribuídos em blocos de acordo com uma *função hash*



- Pesquisa → tempo constante, localizando diretamente o bloco do registro se a pesquisa for feita com operador = sobre o campo *hash*
  - Função aplicada sobre o campo *hash* calcula o endereço do bloco do registro
  - Complexidade → $O(1)$

Tipos de Arquivo: Hash

Operações de alteração de registros eficientes

- Inclusão → pode gerar colisão e necessidade de expansão de arquivo
  - Aplica-se a *função hash* sobre o campo *hash* para calcular o endereço do bloco
  - Bloco copiado para um *buffer*, onde o novo registro é acrescentado, e o *buffer* é copiado de volta ao bloco
- Alteração → dependente da condição de pesquisa e do campo alterado
  - Endereço do bloco do arquivo é recuperado via pesquisa em tempo constante
  - Bloco copiado para um *buffer*
    - Campo *hash* não modificado → modifica-se o registro no *buffer*, copiando o *buffer* de volta ao bloco
    - Campo *hash* modificado → desloca-se o registro para outro bloco gravando os dois blocos modificados
- Exclusão → igualmente dependente da condição de pesquisa

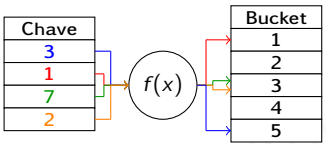
Tipos de Arquivo: Hash

Funções hash podem ser implementadas de formas diferentes

- Idealmente devem ser mantidas em memória primária, tornando muito eficiente o mapeamento de valores
- Implementações robustas distribuem valores de maneira uniforme, consumindo pouca memória primária
- Existem dois problemas muito comuns em implementações *hash*
  - Colisão → diferentes valores são mapeados para o mesmo endereço, que já pode estar ocupado
  - Expansão → não há mais endereços disponíveis para armazenamento de registros e o espaço de endereçamento precisa ser expandido

Tipos de Arquivo: Hash

Hashing Universal → mapeia conjunto de chaves de **tamanho variável** para espaço de tamanho m, tal que a probabilidade de colisão é $1/m$

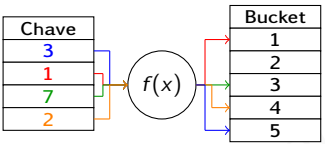


- Colisão → problema frequente quando $n \approx m$ ou $n \geq m$, onde n é o número de chaves
  - Endereçamento Aberto
  - Lista Encadeada
  - Hash Múltiplo
- Fator de Carga → n/m (max 0.75)

- Expansão → fundamental para evitar degradação da estrutura, proveniente do grande número de colisões
  - Múltiplas soluções, da reconstrução até o uso de múltiplas funções *hash*

Tipos de Arquivo: Hash

Hashing Perfeito → mapeia **conjunto fixo de chaves** para espaço de tamanho m, tal que não haja colisão



- Funções ocupam mais espaço em memória, com complexidade linear $O(n)$
- **Mínimo** → $n = m$, onde n é o número de chaves
  - Ordem preservada → $\Omega(n \log n)$
  - Ordem não preservada → $1.44n$

- **Expansão** → expandir significa reconstruir, já que conjunto de chaves é fixo
  - Hashing perfeito dinâmico pode ser a solução, mas complexa e de difícil implementação

TECNOLOGIAS DE ARMAZENAMENTO

Desempenho e confiabilidade em sistemas de banco de dados estão intimamente relacionados à **organização dos dados** nos meios de armazenamento e à **tecnologia de armazenamento** de dados empregada

- Minimizar o número de transferências de blocos e *buffers*
- Agilizar o tempo de cada transferência

Organização adequada de dados ajuda a minimizar número de transferências

- Tipo de arquivo empregado pode tornar a busca linear, logarítmica ou em tempo constante

Tecnologia de armazenamento adequada ajuda a reduzir custo de transferência

- Tipo de memória secundária empregada, bem como sua configuração, podem acelerar o acesso aos dados em ordens de magnitute

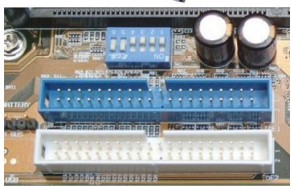
Padrão de interligação de periféricos determina a forma como dispositivos, como memórias secundárias, são interligados ao hardware computacional

- ATA (*Advanced Technology Attachment*) → ou IDE (*Integrated Drive Eletronics*) é um padrão que oferece baixos custo e desempenho, suportando velocidade de transferência até ≈ 0,15Gbps
- SATA (*Serial ATA*) → padrão flexível, oferecendo uma gama de opções com custos variados, suportando velocidade de transferência até ≈ 6Gbps
- SCSI (*Small Computer System Interface*) → custo e desempenho elevados, suportando velocidade de transferência até ≈ 6Gbps

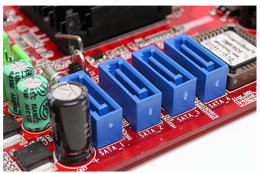
Padrões diferem entre si não só quanto à velocidade de transferência, mas também quanto aos recursos suportados, em especial a possibilidade de criação de conjuntos de discos magnéticos (RAID) com *hot-swap*

Hardware de Armazenamento

ATA



SATA



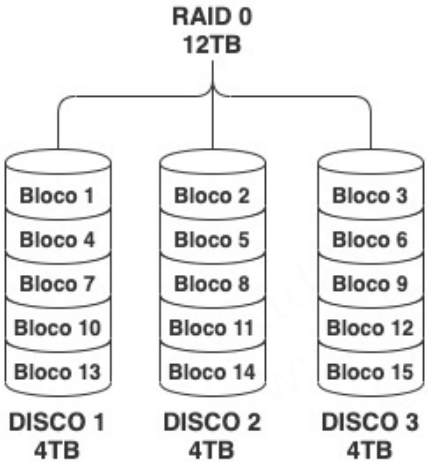
SCSI

Conjuntos de Discos

RAID (*Redundant Array of Independent Disks*) é uma tecnologia de virtualização de armazenamento que combina discos em uma unidade lógica

- Desempenho → possibilidade de ampliação da capacidade de armazenamento e da velocidade de transferência de dados
  - Distribuição de dados em vários discos, com balanceamento de carga
  - I/O paralelo, provendo alta taxa de transferência
- Redundância → possibilidade de ampliação da disponibilidade e da confiabilidade por ser tolerante a falhas
  - Distribuição de cópias de dados em vários discos

Suporta diferentes esquemas de configuração, provendo diferentes níveis de desempenho e redundância



RAID 0

Distribuição sem cópia

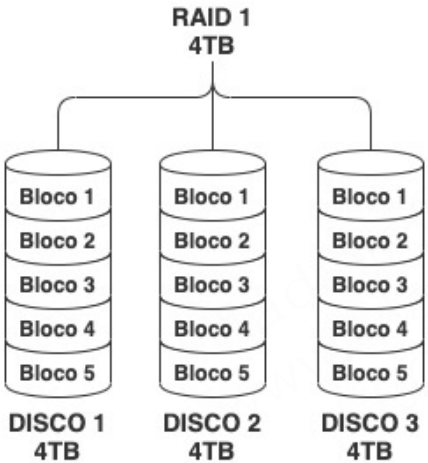
Requisito $\rightarrow n > 1$ discos

Velocidade $\rightarrow \propto n$

Capacidade $\rightarrow \equiv n$

Não tolerante a falhas

Alto desempenho, com taxa de falha maior que em discos sem RAID



RAID 1

Espelhamento sem distribuição

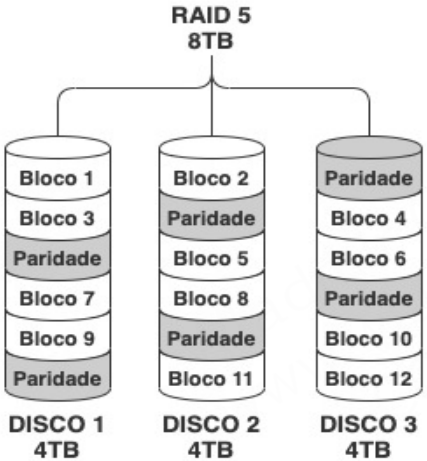
Requisito $\rightarrow n > 1$ discos

Velocidade $\rightarrow \propto 1$

Capacidade $\rightarrow \equiv 1$

Tolerante a falhas

Desempenho equivalente a discos sem RAID, mas com taxa de falha menor



RAID 5

Distribuído com cópia

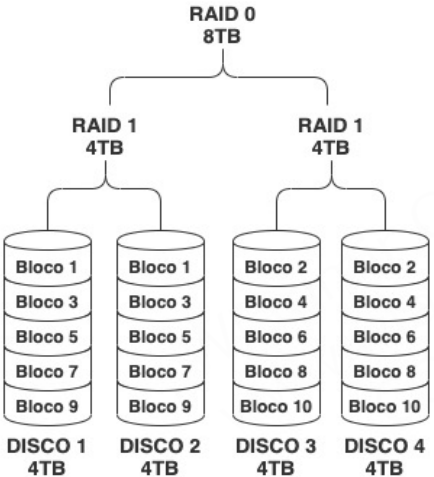
Requisito $\rightarrow n > 2$ discos

Velocidade $\rightarrow \propto n$

Capacidade $\rightarrow \equiv n - 1$

Tolerante a falhas

Desempenho e capacidade próximos ao RAID 0, com taxa de falha menor que discos sem RAID



RAID 10

Espelhado com distribuição

Requisito $\rightarrow n \propto m \wedge n > m$

Velocidade $\rightarrow \propto n/m$

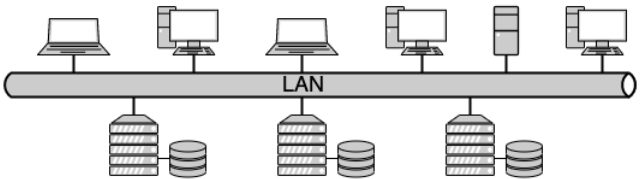
Capacidade $\rightarrow \equiv n/m$

Tolerante a falhas

Combinação RAID 1 (espelho tamanho m) e 0, suportando múltiplas falhas, enquanto houver cópia espelhada

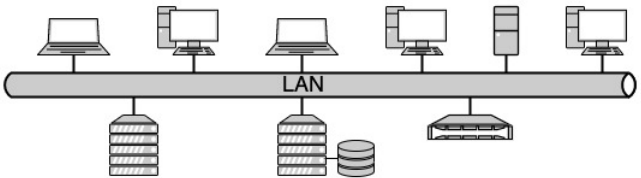
Abordagens de Armazenamento

DAS (DIRECT-ATTACHED STORAGE) → discos magnéticos contendo os arquivos dos bancos de dados integrados ao hardware do sistema de banco de dados



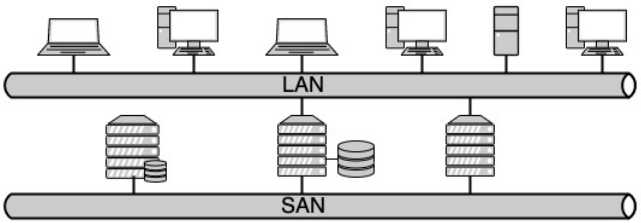
- Integração por meio de padrões de interligação, como ATA, SATA e SCSI
- Discos acessíveis diretamente apenas pelo hardware do sistema
- Abordagem simples e barata, mas menos robusta e escalável

NAS (NETWORK-ATTACHED STORAGE) → arquivos de bancos de dados parcial ou totalmente contidos em hardware especializado de compartilhamento de arquivo



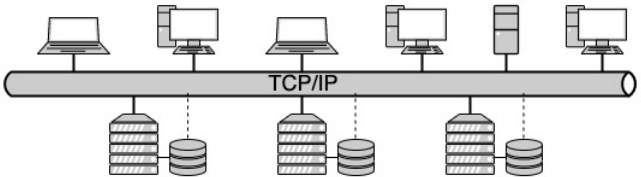
- Integração por protocolos de compartilhamento, como NFS, SMB e AFP
- Hardware de sistema de banco de dados *"enxerga"* hardware especialista como servidor de arquivo

SAN (Storage Area Network) → rede dedicada, geralmente em fibra óptica, de hardwares de armazenamento e de sistemas de banco de dados



- Suporta hardware dedicado de armazenamento interligado ao hardware de sistema através da rede
- Hardware de sistema *"enxerga"* hardware dedicado como disco

iSCSI (Internet SCSI) → arquivos de bancos de dados parcial ou totalmente contidos em discos espalhados e acessíveis diretamente pela rede



- Integração por protocolos de camada de transporte TCP/IP
- Hardware de sistema *"enxerga"* e acessa diretamente discos distribuídos pela rede como se estivessem fisicamente conectados
- Abordagem flexível, barata e escalável

INDEXAÇÃO DE NÍVEL ÚNICO

Indexação

Construção de índices para acelerar recuperação de registros de arquivos

Índice → Caminho alternativo de acesso a registros de um arquivo

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento	
12345678900	Roberto Machado	M	1200.00	1	#00
21345678900	Carlos A. Martins	M	3200.00	1	
32145678900	Ana Maria Freitas	F	7500.00	2	#10
...	...	...	...	...	
52345678902	Luiz A. Barbosa	M	5300.00	N	#FF

ÍNDICE

Ponteiro	Departamento
#00	1
#10	2
...	...
#FF	N

- Arquivo de índice → arquivo adicional ao arquivo de dados (indexado)
- Contém campos de indexação provenientes do arquivo indexado
- Registro → campos de indexação + ponteiro para bloco que armazena o registro do arquivo indexado referenciado pelos campos de indexação

Índice de nível único → arquivo ordenado pelo campo de indexação

- Denso → um registro de índice para cada registro no arquivo indexado

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento	
12345678900	Roberto Machado	M	1200.00	1	#00
21345678900	Carlos A. Martins	M	3200.00	1	
32145678900	Ana Maria Freitas	F	7500.00	2	#10

ÍNDICE

Ponteiro	Departamento
#00	1
#00	1
#10	2

- Esparso → registros de índice para alguns registros no arquivo indexado

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento	
12345678900	Roberto Machado	M	1200.00	1	#00
21345678900	Carlos A. Martins	M	3200.00	1	
32145678900	Ana Maria Freitas	F	7500.00	2	#10

ÍNDICE

Ponteiro	Departamento
#00	1
#10	2

Índice Primário

Índice esparso com registros de tamanho fixo

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento	
12345678900	Roberto Machado	M	1200.00	1	#00
12345678901	Manuela Costa	F	2700.00	3	
21345678900	Carlos A. Martins	M	3200.00	1	#10
32145678900	Ana Maria Freitas	F	7500.00	2	
52345678902	Luiz A. Barbosa	M	5300.00	3	#20

ÍNDICE

Ponteiro	CPF
#00	12345678900
#10	21345678900
#20	52345678902

- Campo de indexação → referencia chave primária do arquivo indexado
- Demanda arquivo indexado ordenado pela chave primária
- Um registro de índice para cada bloco do arquivo indexado
  - Primeiro registro de cada bloco do arquivo indexado (âncora do bloco) encontra-se referenciado por um registro no arquivo de índice

Índice Primário

Para um arquivo indexado de Professor, com 10.000 registros de tamanho fixo de 185B, ordenado pela chave primária CPF e armazenado em um disco com blocos de 4KB, teremos:

- Fator de Bloco $\rightarrow F = \left\lfloor \frac{4KB}{185B} \right\rfloor = \left\lfloor \frac{4 \times 1.024B}{185B} \right\rfloor \approx \lfloor 22,14 \rfloor = 22$
- # Blocos $\rightarrow B = \left\lceil \frac{10.000}{22} \right\rceil \approx \lceil 454,54 \rceil = 455$
- Espaço $\rightarrow S = 455 \times 4KB = 1.820KB \approx 1,77MB$

Pesquisas nesse arquivo demandarão acessos a blocos de disco:

- Pela chave primária $\rightarrow A = \lceil \log_2 455 \rceil \approx \lceil 8,83 \rceil = 9$
- Por outro campo $\rightarrow A = 455$

Índice Primário

Para um índice primário criado sobre a chave primária CPF de 11B, onde o ponteiro de bloco ocupa 16B, teremos:

- Fator de Bloco $\rightarrow F_P = \left\lfloor \frac{4KB}{11B+16B} \right\rfloor = \left\lfloor \frac{4\times1.024B}{27B} \right\rfloor \approx \lfloor 151,70 \rfloor = 151$
- # Blocos $\rightarrow B_P = \left\lceil \frac{455}{151} \right\rceil \approx \lceil 3,01 \rceil = 4$
- Espaço $\rightarrow S_P = 4 \times 4KB = 16KB$

Pesquisas nesse arquivo só podem ser realizadas pelo campo de indexação e demandarão acessos a blocos de disco:

- $A_P = \lceil \log_2 4 \rceil = 2$
- $+1$ acesso para recuperar o registro no arquivo indexado

Índice Primário

Arquivo de índice é significativamente menor que arquivo indexado, ocupando menos blocos em disco

- Esparsidade → menos registros no arquivo de índice
- Tamanho → registros de índice menores que registros do arquivo indexado
- Ordenação → arquivo ordenado com pesquisa de complexidade logarítmica
  - Pesquisa binária no arquivo de índice para encontrar a chave procurada
  - Acesso direto ao registro de dados através do endereço de bloco recuperado a partir do índice

Operações de atualização no arquivo indexado podem envolver reordenação de registros do próprio arquivo indexado e do arquivo de índice

Índice de Agrupamento

Ou **ÍNDICE CLUSTERING**, é um ÍNDICE ESPARSO com registros de tamanho fixo

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento	
12345678900	Roberto Machado	M	1200.00	1	#00
21345678900	Carlos A. Martins	M	3200.00	1	
32145678900	Ana Maria Freitas	F	7500.00	2	#10
12345678901	Manuela Costa	F	2700.00	3	
52345678902	Luiz A. Barbosa	M	5300.00	3	#20

ÍNDICE

Ponteiro	Departamento
#00	1
#10	2
#10	3

- Campo de indexação → referencia campo de agrupamento (chave não exclusiva) do arquivo indexado
- Demanda arquivo indexado ordenado pelo campo de agrupamento
- Um registro de índice para cada valor distinto no campo de agrupamento
  - Bloco de primeira ocorrência de um valor distinto no campo de agrupamento encontra-se referenciado por um registro no arquivo de índice

Índice de Agrupamento

No arquivo indexado de Professor, para um índice de agrupamento criado sobre o campo Departamento de 8B, contendo 200 valores distintos, onde o ponteiro de bloco ocupa 16B, teremos:

- Fator de Bloco $\rightarrow F_C = \left\lfloor \frac{4KB}{8B+16B} \right\rfloor = \left\lfloor \frac{4\times1.024B}{24B} \right\rfloor \approx \lfloor 170,66 \rfloor = 170$
- # Blocos $\rightarrow B_C = \left\lceil \frac{200}{170} \right\rceil \approx \lceil 1,17 \rceil = 2$
- Espaço $\rightarrow S_C = 2 \times 4KB = 8KB$

Pesquisas nesse arquivo só podem ser realizadas pelo campo de indexação e demandarão acessos a blocos de disco:

- $A_C = \lceil \log_2 2 \rceil = 1$
- $+1$ acesso para recuperar o registro no arquivo indexado

Índice Secundário

Índice denso com registros de tamanho fixo

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento	
12345678900	Roberto Machado	M	1200.00	1	#00
12345678901	Manuela Costa	F	2700.00	3	
21345678900	Carlos A. Martins	M	3200.00	1	#10
32145678900	Ana Maria Freitas	F	7500.00	2	
52345678902	Luiz A. Barbosa	M	5300.00	3	#20

ÍNDICE

Ponteiro	Departamento
#00	1
#10	1
#10	2
#00	3
#20	3

- Campo de indexação → referencia campo não ordenado do arquivo indexado
- Não demanda ordenação no arquivo indexado
- Um registro de índice para cada registro do arquivo indexado
  - Bloco de cada registro do arquivo indexado encontra-se referenciado por um registro no arquivo de índice

Índice Secundário

No arquivo indexado de Professor, para um índice secundário criado sobre o campo Departamento de 8B, onde o ponteiro de bloco ocupa 16B, teremos:

- Fator de Bloco $\rightarrow F_S = \left\lfloor \frac{4KB}{8B+16B} \right\rfloor = \left\lfloor \frac{4\times1.024B}{24B} \right\rfloor \approx \lfloor 170,66 \rfloor = 170$
- # Blocos $\rightarrow B_S = \left\lceil \frac{10.000}{170} \right\rceil \approx \lceil 58,82 \rceil = 59$
- Espaço $\rightarrow S_S = 59 \times 4KB = 236KB$

Pesquisas nesse arquivo só podem ser realizadas pelo campo de indexação e demandarão acessos a blocos de disco:

- $A_S = \lceil \log_2 59 \rceil \approx \lceil 5,88 \rceil = 6$
- $+1$ acesso para recuperar o registro no arquivo indexado

Índice Secundário

Arquivo de índice tipicamente menor que arquivo indexado

- Tamanho → registros de índice menores que registros do arquivo indexado
- Ordenação → arquivo ordenado com pesquisa binária
- Flexibilidade → múltiplos índices sobre um mesmo arquivo
  - Não demandam arquivo indexado ordenado
  - Somente um índice primário ou de agrupamento por arquivo indexado, por demandarem ordenação
- Desempenho → proporcionalmente, maior ganho em tempo de pesquisa
  - Para índices primário ou de agrupamento, tem-se a opção de pesquisa binária tanto no arquivo indexado, quanto no arquivo índice

Comparativo entre diferentes tipos de índices de nível único

Tipo	Esparsidade	Arquivo Indexado	Quantidade Registros
Primário	Esparso	Ordenado	Número de blocos do arquivo indexado
Agrupamento	Esparso	Ordenado	Número de valores distintos no campo de agrupamento
Secundário	Denso	Qualquer	Número de registros do arquivo indexado

INDEXAÇÃO MULTINÍVEL

Indexação

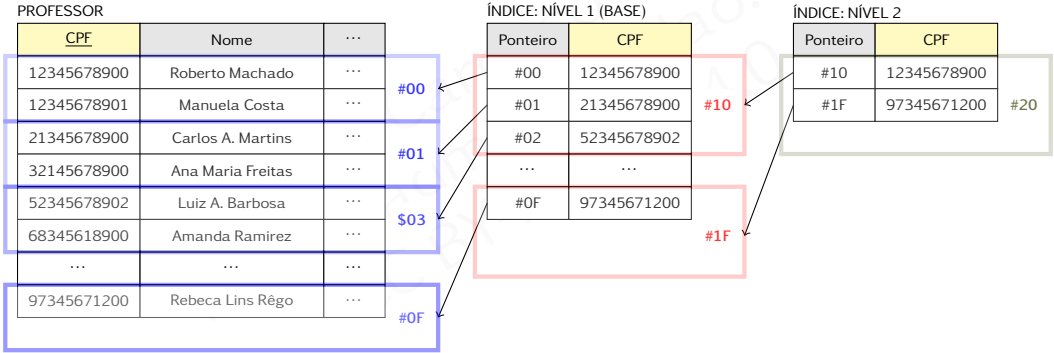
Índice multinível → diferentes níveis de índices são construídos, reduzindo o espaço de pesquisa

- **Estático** → compacto, sem espaço extra em blocos de índice para acomodação de novos registros
  - Tipicamente implementado como arquivo de índice, em que se contrói índices sobre índices
- **Dinâmico** → flexível, com espaço para alocação dinâmica de registros, tornando operações de alteração de dados mais eficientes
  - Tipicamente implementado como árvore B (B Tree) ou B+ (B+ Tree), estruturas baseadas em árvores de pesquisa de múltiplos caminhos, com restrições

Índice Multinível Estático

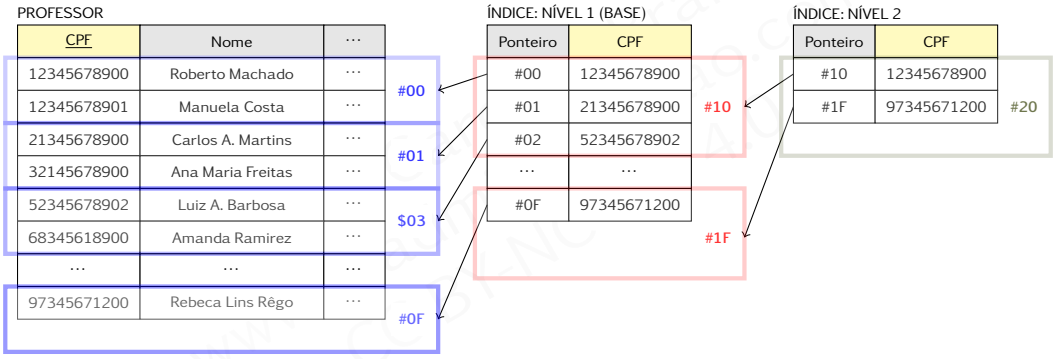
Nível Base → índice com valores distintos no campo de ordenação

Nível Subsequente → índice primário sobre nível adjacente anterior



Índice Multinível Estático

Derivam-se níves até que o *n*-ésimo nível seja armazenado em apenas um bloco



Índice Multinível Estático

Pesquisa → complexidade logarítmica com base > 2, apenas um bloco em cada nível precisa ser acessado

- Base logarítmica → fator de bloco do índice, ou **fan-out** (fo)
- Níveis → $h \approx \lceil log_{fo} r_1 \rceil$, onde r_1 é o número de registros no nível base
- Acessos a blocos → $A = h$

Alteração → níveis ordenados, custo alto de operações de alteração do campo de indexação

Arquivo **ISAM** (Indexed Sequential Access Method) → arquivo sequencial indexado com índice multinível na chave primária

Índice Multinível Estático

Para um arquivo indexado de Professor, com 200.000 registros de tamanho fixo de 185B, ordenado pela chave primária CPF e armazenado em um disco com blocos de 4KB, teremos:

- Fator de Bloco $\rightarrow F = \left\lfloor \frac{4KB}{185B} \right\rfloor = \left\lfloor \frac{4 \times 1.024B}{185B} \right\rfloor \approx \lfloor 22,14 \rfloor = 22$
- # Blocos $\rightarrow B = \left\lceil \frac{200.000}{22} \right\rceil \approx \lceil 9.090,91 \rceil = 9.091$
- Espaço $\rightarrow S = 9.091 \times 4KB = 36.364KB \approx 35,51MB$

Pesquisas nesse arquivo demandarão acessos a blocos de disco:

- Pela chave primária $\rightarrow A = \lceil \log_2 9.091 \rceil \approx \lceil 13,15 \rceil = 14$
- Por outro campo $\rightarrow A = 9.091$

Índice Multinível Estático

Para Professor sendo arquivo ISAM, com CPF de 11B, e ponteiro de bloco ocupando 16B, teremos:

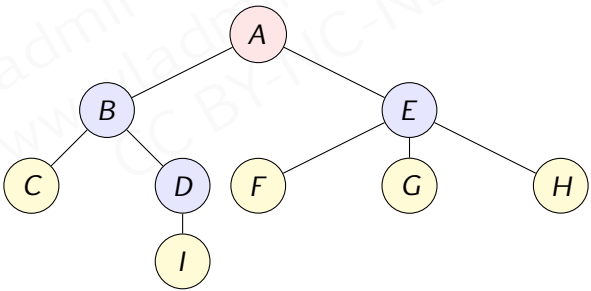
- Fator de Bloco $\rightarrow F_M = \left\lfloor \frac{4KB}{11B+16B} \right\rfloor = \left\lfloor \frac{4 \times 1.024B}{27B} \right\rfloor \approx \lfloor 151,70 \rfloor = 151$
- # Blocos $\rightarrow B_{M1} = \left\lceil \frac{9.091}{151} \right\rceil \approx \lceil 60,20 \rceil = 61$, $B_{M2} = \left\lceil \frac{61}{151} \right\rceil \approx \lceil 0,40 \rceil = 1$
- Espaço $\rightarrow S_M = (61+1) \times 4KB = 248KB$

Pesquisas nesse arquivo só podem ser realizadas pelo campo de indexação e demandarão acessos a blocos de disco:

- $A_M = h \approx \lceil \log_{151} 9.091 \rceil \approx \lceil 1,81 \rceil = 2$
- $+1$ acesso para recuperar o registro no arquivo indexado

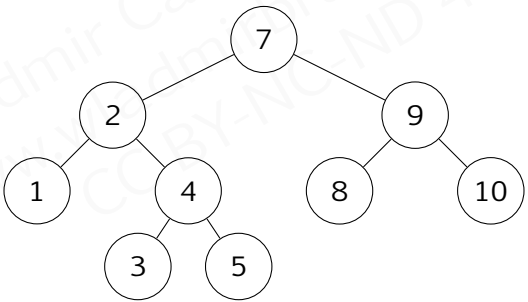
Árvore (*Tree*) → estrutura hierárquica de nós (elementos) conectados

- Raiz → nó sem pai, de nível zero
- Folha → nó sem filhos
- Interno → nó não folha e não raiz
- O nível de um nó na árvore é o nível do seu pai mais um
- Subárvore → árvore formada por um nó e todos os seus descendentes



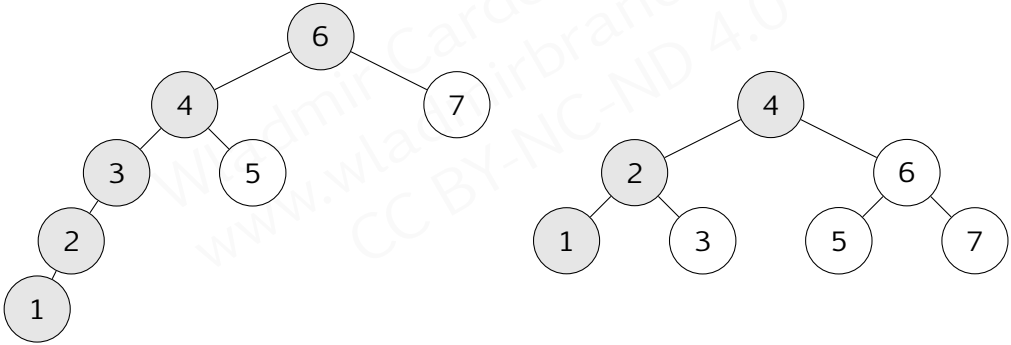
Árvore de Busca (*Search Tree*) → nós com restrições para eficiência em busca

- Binária → nó tem no máximo dois filhos
  - Chave não pode ser menor que qualquer outra em subárvores da esquerda
  - Chave não pode ser maior que qualquer outra em subárvores da direita



Árvore de busca binária (BST) → balanceamento fundamental para eficiência

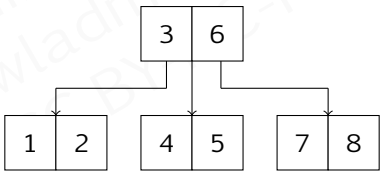
- Balanceamento → altura de árvore $\approx \lceil \log_2 n \rceil$, onde n = número de nós
  - Subárvores esquerda e direita com ≈ mesmo número de nós



Índice Multinível Dinâmico

Árvore de busca de múltiplos caminhos (*m-way*) → generalização de BST

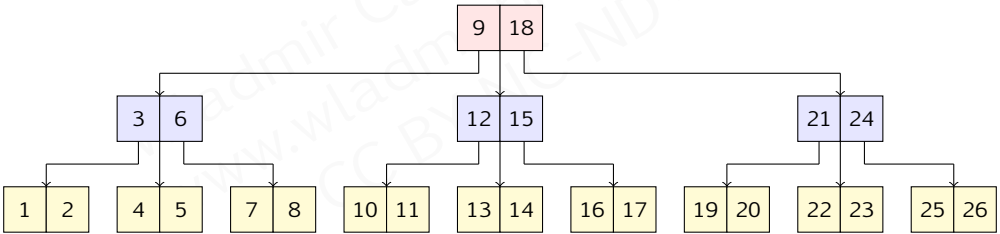
- Cada nó tem m filhos
- Multiplicidade → cada nó contém $m - 1$ elementos
  - $h \leq n \leq m^h - 1$, onde $h =$ altura e $n =$ número de nós
- Balanceamento → $h \approx \lceil \log_m n \rceil$



Índice Multinível Dinâmico

B Tree → *m-way* com restrições que tornam busca e atualização muito eficientes

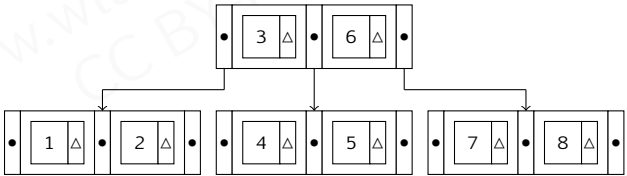
- Nó raiz, não folha, tem ao menos dois filhos
- Nó interno tem ao menos $\lceil m/2 \rceil$ filhos
- Nós folha estão no mesmo nível



Índice Multinível Dinâmico

B Tree → nós ao menos meio cheios, ≈ 69% quando árvore estabiliza

- Pesquisa → eficiente, poucos níveis e perfeitamente balanceada
- Alteração → eficiente, espaço para acomodar novos registros
- Ideal para armazenamento em memória secundária
  - Tipicamente configurada para que um nó ocupe um bloco em disco
  - Elemento → campo de indexação + ponteiro de bloco (△)
  - Nó com m ponteiros de nó (●), um para cada filho



Índice Multinível Dinâmico

Arquivo de PROFESSOR, com índice B TREE em DEPARTAMENTO de 8B, com um nó ocupando um bloco de disco, ponteiro de nó de 12B, ponteiro de bloco de 16B

- Tamanho do elemento $\rightarrow (8B + 16B) = 24B$
- Nó ocupa 1 bloco $\rightarrow 4KB \geq ((m-1) \times (24B + 12B)) + 12B$
  - # Elementos por nó $\rightarrow (m-1) = \left\lfloor \frac{4KB - 12B}{24B + 12B} \right\rfloor = \left\lfloor \frac{4.084B}{36B} \right\rfloor \approx \lfloor 113,44 \rfloor = 113$
  - Ordem da árvore $\rightarrow m = 113 + 1 = 114$
- Altura da árvore $\rightarrow h \approx \lceil \log_{114} 200.000 \rceil \approx \lceil 2,57 \rceil = 3$

Nível	# Nós	# Registros	# Ponteiros de Nó
0	1	113	114
1	114	12.882	12.996
2	12.996	1.468.584	-

Índice Multinível Dinâmico

Considerando uma ocupação de nós em 69%:

- Fator de Bloco $\rightarrow F_B = \lceil 113 \times 0.69 \rceil \approx \lceil 77,97 \rceil = 78$
- # Blocos $\rightarrow B_B = \left\lceil \frac{200.000}{78} \right\rceil \approx \lceil 2.564,10 \rceil = 2.565$
- Espaço $\rightarrow S_B = 2.565 \times 4KB = 10.260KB \approx 10,01MB$

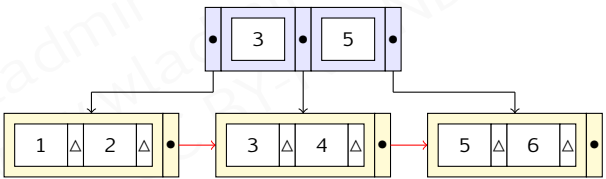
Pesquisas nesse índice só podem ser realizadas pelo campo de indexação e demandarão acessos a blocos de disco:

- $A_B = h = 3$
- $+1$ acesso para recuperar o registro no arquivo indexado

Índice Multinível Dinâmico

B+ Tree → extensão B Tree com restrições que tornam ainda mais eficientes a busca e a remoção

- Nó Índice → raiz ou interno que armazena exclusivamente chave
- Nó Registro → folha que armazena registro de índice
- Nós folha em lista encadeada ordenada



Índice Multinível Dinâmico

Arquivo de Professor, com índice B+ Tree em Departamento de 8B, com um nó ocupando um bloco de disco, ponteiro de nó de 12B, ponteiro de bloco de 16B

- Nó Índice
  - Tamanho do elemento $\rightarrow 8B$
  - Nó ocupa 1 bloco $\rightarrow 4KB \geq ((m-1) \times (8B + 12B)) + 12B$
    - Elementos por nó $\rightarrow (m-1) = \left\lfloor \frac{4KB - 12B}{8B + 12B} \right\rfloor = \left\lfloor \frac{4.084B}{20B} \right\rfloor \approx \lfloor 204,2 \rfloor = 204$
    - Ordem da árvore $\rightarrow m = 204 + 1 = 205$
  - Altura da árvore $\rightarrow h \approx \lceil log_{205} 200.000 \rceil \approx \lceil 2,29 \rceil = 3$
  - Altura de nós índice $\rightarrow h_i = h - 1 = 2$

Índice Multinível Dinâmico

Nível	# Nós	# Registros	# Ponteiros de Nó
0	1	204	205
1	205	41.820	42.025
2	42.025	8.573.100	-

- Nó Registro
  - Tamanho do elemento $\rightarrow (8B + 16B) = 24B$
  - Nó ocupa 1 bloco $\rightarrow 4KB \geq ((m-1) \times (24B)) + 12B$
    - Elementos por nó $\rightarrow (m-1) = \left\lfloor \frac{4KB-12B}{24B} \right\rfloor = \left\lfloor \frac{4.084B}{24B} \right\rfloor \approx \lfloor 170,16 \rfloor = 170$

Índice Multinível Dinâmico

Considerando uma ocupação de nós em 69%:

- Fator de Bloco $\rightarrow F_{B+} = \lceil 170 \times 0.69 \rceil \approx \lceil 117,30 \rceil = 118$
- # Blocos $\rightarrow B_{B+} = \left\lceil \frac{200.000}{118} \right\rceil \approx \lceil 1.694,91 \rceil = 1.695$
- Nível 0 a $h_i - 1 \rightarrow 1.695$ ponteiros
  - # Nós = # Blocos = $\rightarrow \left\lceil \frac{1.695}{\lceil (204 \times 0.69) \rceil + h_i} \right\rceil \approx \left\lceil \frac{1.695}{143} \right\rceil \approx \lceil 11,85 \rceil = 12$
- Espaço $\rightarrow S_{B+} = (1.695 + 12) \times 4KB = 6.828KB \approx 6,67MB$

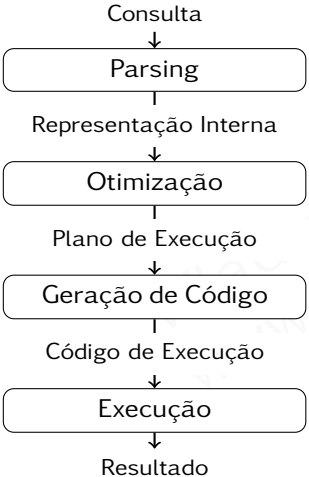
Pesquisas pelo campo de indexação demandam acessos a blocos:

- $A_{B+} = h = 3$
- $+1$ acesso para recuperar o registro no arquivo indexado

PROCESSAMENTO DE CONSULTA

Processamento de Consulta

SGBDs **PROCESSAM, OTIMIZAM** e **EXECUTAM** consultas



- PARSING → análise sintática
  - VARREDURA → tokenização
  - ANÁLISE → regras gramaticais SQL
  - VALIDAÇÃO → metadados ∈ esquema
- OTIMIZAÇÃO → escolha de estratégia eficiente para execução da consulta
  - ÁRVORE DE CONSULTA → ou grafo de consulta é uma representação interna da consulta
  - PLANO DE EXECUÇÃO → estratégia de execução
- GERAÇÃO DE CÓDIGO → código compilado ou interpretado para execução

Processamento de Consulta: Tradução

Parsing → consulta é traduzida para álgebra relacional

- **Tradução** → decomposição em blocos de expressões SELECT-FROM-WHERE

```
SELECT  CPF, Nome FROM PROFESSOR WHERE Salario >
        (SELECT AVG(Salario) FROM PROFESSOR WHERE Sexo='M');
```

```
SELECT  CPF, Nome
  FROM  PROFESSOR
 WHERE  Salario > X;
```

$$\pi_{CPF,\,Nome}(\sigma_{Salario > X}(PROFESSOR))$$

```
SELECT  AVG(Salario)
  FROM  PROFESSOR
 WHERE  Sexo = 'M';
```

$$\gamma_{AVG(Salario)}(\sigma_{Sexo='M'}(PROFESSOR))$$

Processamento de Consulta: Algoritmos

Processar uma consulta envolve a escolha de algoritmos e estratégias a serem aplicados na execução de sequências de operações da álgebra relacional

- Ordenação → agregação, conjunto, junção e projeção (distinct)
- Pesquisa → junção e seleção
- Hashing → agregação, conjunto, junção, projeção e seleção

A escolha do algoritmo e estratégia adequados é feita pelo SGBD, dependendo da tecnologia de armazenamento e da organização de dados

- Memória primária livre → *hashing* em operação de junção
- Arquivo ordenado → pesquisa binária em operação de seleção

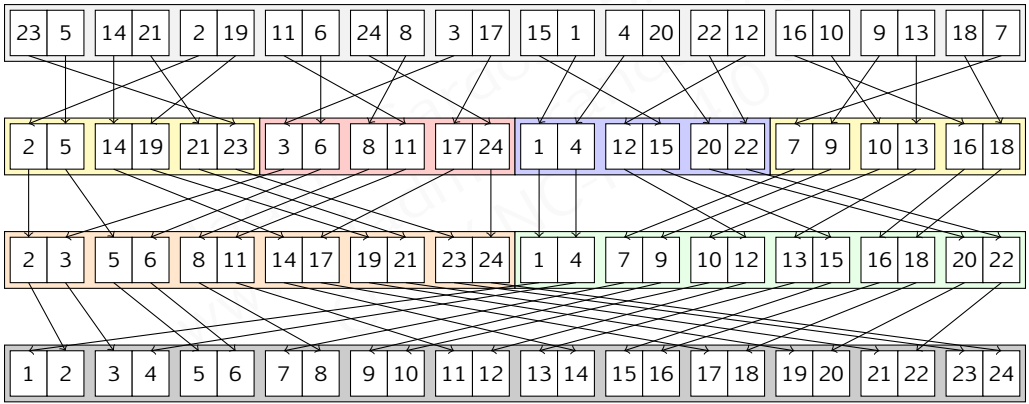
Algoritmos: Ordenação Externa

Estratégia de ordenação e intercalação (MERGE-SORT) de registros em disco

- ORDENAÇÃO → partes (*runs*) do arquivo são transferidas do disco para memória primária, ordenadas em memória primária e regravadas em disco
  - $B_M \rightarrow$ # *buffers* disponíveis em memória primária
  - $B_D \rightarrow$ # blocos do arquivo em disco
  - $R = \left\lceil \frac{B_D}{B_M} \right\rceil \rightarrow$ # *runs*
- INTERCALAÇÃO → mesclagem de *runs* ordenadas em disco
  - Grau $\rightarrow D = min((B_M - 1), R) \rightarrow$ # *runs* mescladas em cada passo
  - $S = \lceil \log_D R \rceil \rightarrow$ # passos de intercalação

Custo $O(n \log n) \rightarrow (2 \times B_D) + (2 \times B_D \times \lceil \log_D R \rceil)$

Exemplo de MERGE-SORT com 24 registros em 12 blocos e 3 *buffers*, realizando ordenação em $\lceil 12/3 \rceil = 4$ *runs* e intercalação de grau 2 em $\lceil log_2 4 \rceil = 2$ passos



Para ordenar um arquivo que ocupa 1024 blocos em disco usando 5 *buffers*:

- Ordenação → 205 *runs* ordenadas em disco
  - $R = \left\lceil \frac{1024}{5} \right\rceil = \lceil 204,8 \rceil = 205$
- Intercalação → grau 4 em 4 passos
  - $D = min((5-1), 205) = 4$
  - $S = \lceil \log_4 205 \rceil \approx \lceil 3,84 \rceil = 4$
    - Passo 1 → 205 *runs* mescladas 4 a 4
    - Passo 2 → 52 *runs* mescladas 4 a 4
    - Passo 3 → 13 *runs* mescladas 4 a 4
    - Passo 4 → 4 *runs* mescladas

Custo → $(2 \times 1024) + (2 \times 1024 \times 4) = 10.240$

Algoritmos: Projeção (π)

Estratégias envolvem eliminar registros duplicados

- Ordenação-Intercalação $\rightarrow O(n \log n)$
  - Ordena-se o resultado da projeção e varre-se sequencialmente registros removendo duplicatas em registros adjacentes
- Hashing $\rightarrow O(n)$
  - Computa-se um endereço de partição (*bucket*) a partir de uma função *hash* sobre cada registro de resultado, alocando-o no *bucket* correspondente
  - Antes da alocação verifica-se se o registro já está presente no *bucket*, somente alocando-o se não estiver presente

Algoritmos: Seleção (σ)

Inúmeras estratégias possíveis, dependendo da existência de índices e da característica e complexidade da condição de seleção

- Arquivo Não Indexado
  - Pesquisa Linear $\rightarrow O(n)$
    - Recupera-se cada registro e verifica-se se valores em campos satisfazem a condição de seleção
  - Pesquisa Binária $\rightarrow O(\log n)$
    - Condição de seleção envolve comparação de $< \leq = \geq >$ em campo de ordenação
  - Hashing $\rightarrow O(1)$
    - Condição de seleção envolve comparação de $=$ em campo *hash*

Algoritmos: Seleção (σ)

- Arquivo Indexado
  - Índice Primário, de Agrupamento, Secundário ou Multinível $\rightarrow O(\log n)$
    - Condição de seleção envolve comparação de $< \leq = \geq >$ em campo de indexação
  - Índice Hash $\rightarrow O(1)$
    - Condição de seleção envolve comparação de $=$ em campo de indexação
- Seleção Conjuntiva $\rightarrow$ operador $\wedge$ na condição de seleção
  - Índice Composto $\rightarrow O(\log n)$
    - Condição de seleção envolve um subconjunto dos campos de indexação, desde que todos os campos iniciais estejam presentes

Algoritmos: Seleção (σ)

- Seleção Conjuntiva $\rightarrow$ operador $\wedge$ na condição de seleção
  - Índice Individual $\rightarrow O(\log n)$
    - Pesquisa-se no índice e verifica-se condições remanescentes da seleção
  - Índice Múltiplo $\rightarrow O(\log n)$
    - Pesquisa-se em cada índice secundário separadamente, realiza-se a intersecção de ponteiros recuperados, e verifica-se condições remanescentes da seleção
- Seleção Disjuntiva $\rightarrow$ operador $\vee$ na condição de seleção
  - Índice Múltiplo $\rightarrow O(\log n)$
    - Pesquisa-se em cada índice secundário separadamente e realiza-se a união de ponteiros recuperados. **Demanda** índice para cada campo na condição de seleção

Algoritmos: Operações de Conjunto ($\times \cup \cap -$)

Estratégias envolvem combinação de registros

- Força Bruta $\rightarrow O(n^2)$
  - Produto Cartesiano ($\times$) $\rightarrow$ combinam-se todos os registros de cada conjunto
- Ordenação-Intercalação $\rightarrow O(n \log n)$
  - Ordenam-se os registros dos dois conjuntos, varrem-se os dois conjuntos simultâneamente e a operação de conjunto apropriada é efetuada
- Hashing $\rightarrow O(n)$
  - Computam-se endereços de *bucket* a partir de uma função *hash* para alocação de registros do menor conjunto
  - Computa-se a função *hash* para cada registro do outro conjunto
  - Aloca-se (união) ou desaloca-se (diferença ou intersecção) o registro no *bucket* correspondente de acordo com a operação de conjunto utilizada

Algoritmos: Agregação (γ)

Estratégias dependem da existência de índices e de campos de agrupamento

- Completa → campo de agrupamento (group by) não especificado
  - $O(n)$ → varre-se arquivo de dados ou de índice computando função
  - $O(\log n)$ → para índice B+ Tree no campo usado na função de agregação
- Particionada → campo de agrupamento (group by) especificado
  - Ordenação → $O(n \log n)$
    - Ordena-se arquivo pelo campo de agrupamento, varrendo-o e computando função de agregação para cada partição
  - Hashing → $O(n)$
    - Particiona-se o arquivo em *buckets* usando campo de agrupamento e computa-se função de agregação para cada *bucket*
  - Índice de Agrupamento → $O(n)$
    - Arquivo já particionado, bastando computar função de agregação

Algoritmos: Junção ($\bowtie$)

Estratégias dependem da existência de índices e da característica e complexidade da condição de junção

- Junção de *Loop* Aninhado $\rightarrow O(n^2)$
  - Força bruta
  - Para cada registro $r_i \in R$, recuperar cada registro $s_j \in S$
  - Combinar r_i e s_j se satisfazem a condição de junção
- Junção de *Loop* Único $\rightarrow O(n \log n)$
  - Aplicável caso haja índice em ao menos um arquivo
  - Para cada registro $r_i \in R$, onde R é o arquivo com maior custo de busca
  - Usar o índice em S para recuperar os registros que satisfazem a condição de junção, combinando-os com o registro r_i

Algoritmos: Junção ($\bowtie$)

- Junção Ordenação-Intercalação $\rightarrow O(n \log n)$
  - Ordena-se arquivos por campos presentes na condição de junção
  - Varrem-se simultâneamente ambos os arquivos pelo campo de ordenação, combinando os registros que satisfazem a condição de junção
  - Registros em cada arquivo são acessados apenas uma vez
- Junção Hash $\rightarrow O(n)$
  - Para condição de junção com comparação de $=$
  - Varre-se o arquivo R de menor tamanho, aplicando uma função *hash* sobre o campo presente na condição de junção para criar *buckets* em memória
  - Para cada registro em S, use a função *hash* para encontrar registros em R que satisfazem a condição de junção, e combine-os com o registro de S

Algoritmos: Junção Externa ($\bowtie \ltimes \times$)

Estratégia envolve a combinação de estratégias de junção e de conjunto

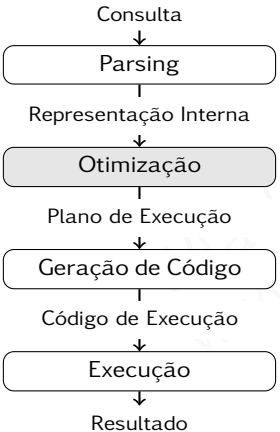
1. Junte os arquivos R e S usando a melhor estratégia de junção
2. Use a operação DIFERENÇA para encontrar os registros do arquivo R (aberto) não presentes no resultado da junção
3. Produto cartesiano dos registros não presentes de R com registro NULO de S
4. Use a operação UNIÃO para combinar os registros das etapas 1 e 3

Custo $\rightarrow (\bowtie) + (-) + (\times) + (\cup)$

OTIMIZAÇÃO DE CONSULTA

Otimização de Consulta

Escolha de algoritmos e estratégias eficientes para execução de sequências de operações de álgebra relacional



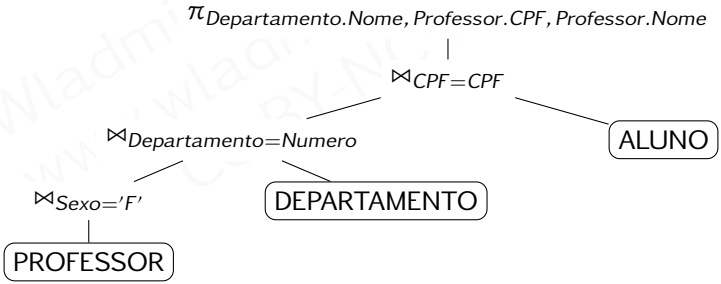
- Heurística → aplicação de regras
  - Consultas interpretadas
- Custo → estimativa de custo de execução
  - Consultas compiladas
- Semântica → compreensão e substituição
  - Consulta é substituída por outra *"melhor"*

Execução do plano tipicamente de forma encadeada (***pipeline***) sem geração de arquivos intermediários

Árvore de Consulta

Estrutura em árvore que representa uma expressão de álgebra relacional

- Folha → arquivo de entrada
- Interno → operação da álgebra relacional
- Raíz → operação de projeção final da álgebra relacional
- Ordem → operações executadas da folha esquerda para a raíz



Otimização Heurística

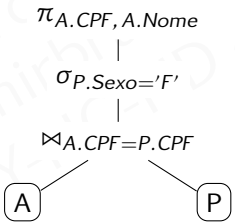
Heurística → método para modificação da representação interna de uma consulta, visando tornar seu plano de execução eficiente

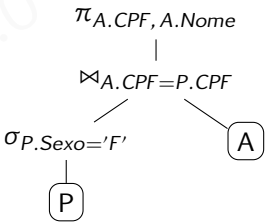
- Não garante que a melhor representação interna, aquela que gere o plano de execução mais eficiente, seja gerada
- Aplicação de um conjunto de regras sobre uma árvore de consulta inicial
  - Árvore inicial obtida a partir do *parsing* da consulta
  - Operações algébricas que reduzam o tamanho dos resultados intermediários devem ser executadas primeiro
    - Operações unárias de seleção, projeção e agregação antes de operações binárias de junção, e de conjunto
  - Operações algébricas de menor custo devem ser executadas primeiro
    - Operações binárias mais eficientes antes

Árvore Inicial → o *parsing* da consulta pode ser feito no sentido natural, da esquerda para a direita, ou reverso, da direita para a esquerda

- Arquivos e operações são inseridos na árvore na ordem de *parsing*

```
SELECT  A.CPF, A.Nome
  FROM  ALUNO A,
        PROFESSOR P
WHERE   A.CPF = P.CPF
  AND   P.Sexo = 'F';
```





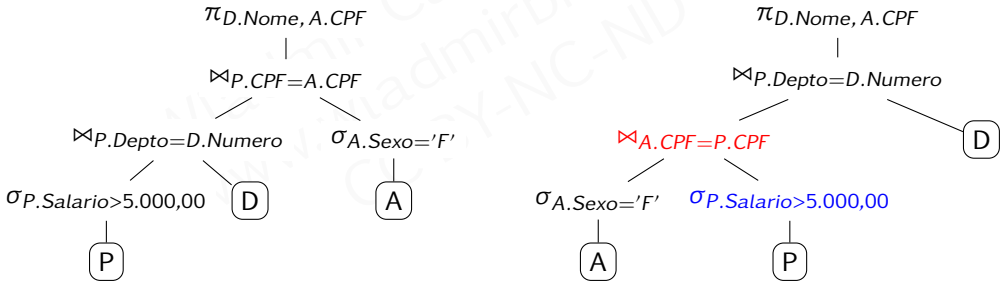
Otimização Heurística

Regras → modificam árvore de consulta inicial, baseando-se no princípio ***reduzir para combinar***

- Redução → operações unárias precedem operações binárias
  - Primeira operação em cada nó folha deve ser seleção, seguida de projeção, seguida de agregação
- Combinação → operações de menor custo precedem as de custo maior
  - Nós folhas devem ser reordenados de forma que junções mais eficientes sejam executadas primeiro, evitando-se produtos cartesianos

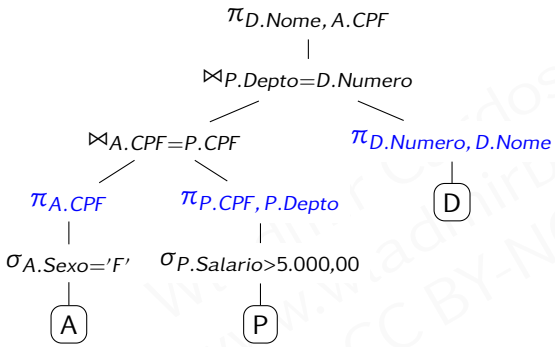
Considerando *parsing* no sentido reverso, e existência de índice multinível em *CPF* de PROFESSOR e ALUNO, e índice primário em *Numero* de DEPARTAMENTO:

```
SELECT D.Nome, A.CPF FROM ALUNO A, DEPARTAMENTO D, PROFESSOR P
 WHERE A.CPF = P.CPF AND A.Sexo = 'F'
   AND P.Depto = D.Numero AND P.Salario > 5.000,00
```



Otimização Heurística

Projeções internas reduzem ainda mais o tamanho dos resultados intermediários

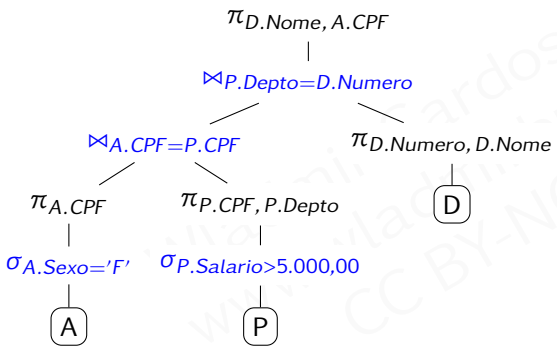



```
SELECT  D.Nome, A.CPF
  FROM  DEPARTAMENTO D,
        PROFESSOR P,
        ALUNO A
 WHERE  D.Numero = P.Depto
   AND  P.CPF = A.CPF
   AND  P.Salario > 5.000,00
   AND  A.Sexo = 'F'
```

Consulta pode ser reescrita de forma que a árvore inicial resultante do *parsing* se aproxime da árvore de consulta otimizada

Otimização Heurística

Plano de Execução → métodos de acesso, algoritmos e estratégias usados no processamento das operações na árvore de consulta otimizada



1. Pesquisa Linear → arquivo não indexado A
2. Pesquisa Linear → arquivo não indexado P
3. Junção de loop único → varre arquivo não indexado A e pesquisa arquivo de índice de P
4. Junção de loop único → varre resultado intermediário e pesquisa arquivo de índice de D

Otimização Baseada em Custo

Função de Custo → computa estimativa de custo para algumas estratégias de execução de consulta, escolhendo a estratégia de menor estimativa

- Tempo de Resposta → # estratégias de execução avaliadas é limitado
- Consultas Compiladas → otimização feita na compilação
- Evidências de Custo → combinadas pela função de custo
  - Computação → custo de processamento (CPU) de dados em memória primária
  - Memória → consumo de *buffers* de memória primária
  - Disco → consumo de blocos de disco
  - I/O → custo com operações de paginação, transferência de dados entre memória primária e secundária
  - Comunicação → custo de transferência de dados via rede

Otimização Baseada em Custo

Catálogo → armazena informação necessária para estimar custo

- Arquivo → organização, tamanho, blocagem, # blocos, tamanho de registro, # de registros
- Índice → tipo, # níveis, # blocos em 1° nível
- Registro → distribuição de valores
  - Seletividade → fração de registros que satisfazem uma condição de igualdade em um campo
    - Exemplo → 50% de registros de professor tem *Sexo* = 'M'
  - Cardinalidade → # médio de registros que satisfazem uma condição de associação por igualdade em um campo
    - Exemplo → cada registro de departamento está associado em média a 5 registros de professor

Otimização Semântica

Compreensão → SGBD precisa *compreender* o significado da consulta para reescrevê-la de uma forma *melhor*, que gere um plano de execução mais eficiente

A seguinte consulta demanda força bruta para execução:

```
SELECT CPF, Nome FROM PROFESSOR
 WHERE CPF IN (SELECT CPF FROM ALUNO)
```

Mas pode ser reescrita para:

```
SELECT A.CPF, A.Nome FROM PROFESSOR A, ALUNO B
 WHERE A.CPF = B.CPF
```

Forma que demanda junção por loop único se professor ou aluno tiverem índice em *CPF*

PROJETO FÍSICO

Projeto Físico de BD

Criação de estrutura apropriada para armazenamento de dados com foco em desempenho na execução de consultas e transações

Fator de Impacto

- **Característica da consulta**
- Frequência de execução
- Tempo de execução
- Exclusividade em campo

Decisão

- Recuperação de dados
  - Que arquivos são acessados?
  - Que campos estão especificados?
    - Junção e seleção → indexação
  - Condição em junção e seleção?
    - $=$ → multiplicidade de índices
    - $\neq$ → limita uso de índice
    - $< \leq \geq >$ → limitação índice *hash*

Criação de estrutura apropriada para armazenamento de dados com foco em desempenho na execução de consultas e transações

Fator de Impacto

- **Característica da consulta**
- Frequência de execução
- Tempo de execução
- Exclusividade em campo

Decisão

- Atualização de dados
  - Que arquivos são atualizados?
  - Que operações são realizadas?
  - Que campos estão especificados?
    - Seleção → considerar indexação
    - Modificação → evitar indexação
  - Condição em seleção?
    - $=$ → multiplicidade de índices
    - $\neq$ → limita uso de índice
    - $< \leq \geq >$ → limitação índice *hash*

Criação de estrutura apropriada para armazenamento de dados com foco em desempenho na execução de consultas e transações

Fator de Impacto

- Característica da consulta
- **Frequência de execução**
- Tempo de execução
- Exclusividade em campo

Decisão

- Execuções por unidade de tempo?
  - Frequente → considerar indexação para recuperação frequente e evitar indexação para atualização frequente
  - Pareto → 80% do processamento é consumido por 20% das consultas e transações
    - Top 20% → considerar indexação para as de recuperação e evitar indexação para as de atualização

Projeto Físico de BD

Criação de estrutura apropriada para armazenamento de dados com foco em desempenho na execução de consultas e transações

Fator de Impacto

- Característica da consulta
- Frequência de execução
- **Tempo de execução**
- Exclusividade em campo

Decisão

- Tempo médio e máximo esperado para execução?
  - Que consultas e transações tem forte restrição de tempo de execução?
  - Críticas → considerar indexação para recuperação e evitar indexação para atualização

Criação de estrutura apropriada para armazenamento de dados com foco em desempenho na execução de consultas e transações

Fator de Impacto

- Característica da consulta
- Frequência de execução
- Tempo de execução
- **Exclusividade em campo**

Decisão

- Que campos são exclusivos?
  - Campos exclusivos são usados frequentemente em JUNÇÃO e SELEÇÃO
  - Considerar indexação para campos exclusivos

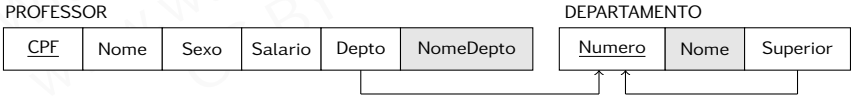
Muitas decisões de projeto físico envolvem indexação

```
CREATE [UNIQUE] INDEX <nome>
ON <tabela> ( <coluna> [<ORDEM>] {, <coluna> [<ORDEM>]} )
[CLUSTER];
```

- UNIQUE → campo de indexação será exclusivo
- CLUSTER → índice será um arquivo ordenado pelo campo de indexação
- ORDEM → forma de ordenação do campo de indexação
  - ASC → ordenação ascendente
  - DESC → ordenação descendente

DESNORMALIZAÇÃO → modificação no projeto lógico para se obter mais eficiência no processamento de consultas e transações

- Duplicação de atributos → inclusão de atributos de uma tabela em outra
  - Evita operações de JUNÇÃO entre as tabelas
  - Introduz **redundância** em tabelas
  - Exemplo: introduzir o atributo *Nome* da tabela DEPARTAMENTO na tabela PROFESSOR evita a necessidade de JUNÇÃO entre as tabelas se a consulta por *Nome* de DEPARTAMENTO e *Nome* de PROFESSOR for frequentemente realizada



Ajuste de BD

Todo projeto precisa de ajustes ao ser executado

Sintonia (Tuning) → processo de ajuste contínuo do projeto físico

- Monitora e revisa decisões de projeto
  - Objetivo → redução do tempo de processamento de consultas e transações
  - Métricas → conjunto de medidas usadas no monitoramento
    - Processamento → tempo de otimização e execução de consultas e transações
    - Armazenamento → espaços ocupados por *pools* de *buffer*, tabelas (*tablespaces*) e índices (*indexspaces*)
    - I/O → # paginações em disco por unidade de tempo
    - Concorrência → taxa de vazão (*throughput*) de transações, de emissão de comandos de bloqueio, e de registro em *log*
    - Índice → # níveis, # nós folha não contíguos

Ajuste de BD

DBAs monitoram métricas para realizar ajustes e evitar problemas:

- Desperdício → tamanho de *buffers* inadequados
- Sobrecarga → *logging* e *dumping* desnecessários
- Aumento de Concorrência → disputa excessiva por bloqueios
- Ineficiência → alocação inadequada de discos, *buffers* e processos

Tais ajustes podem ocorrer de diferentes formas

- Sintonia de Índice → criação, remoção e reorganização de índices
- Sintonia de Projeto → alterações no projeto lógico
- Sintonia de Consulta → reescrita de consultas

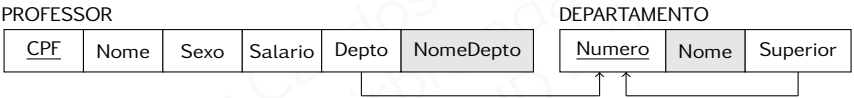
Sintonia de Índice

Avaliação dos requisitos de projeto físico para reorganizar arquivos e índices

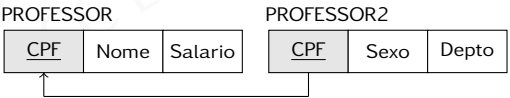
- Criação → consultas e transações podem estar demorando a serem executadas por ausência de índice
- Remoção → índices podem estar sendo pouco utilizados
- Reorganização → índices podem estar sendo muito atualizados
  - Exclusão → blocos de índice com espaço desperdiçado
  - Inclusão → *overflow* excessivo em índice agrupado
- Opções de indexação variam em soluções de SGBD comerciais
  - Sybase → índices de agrupamento esparsos em B+ Trees
  - Ingress → índices de agrupamento ISAM esparsos ou B+ Trees densos
  - Oracle → índices de agrupamento densos

Avaliação dos requisitos de projeto físico para modificação dos projetos conceitual e lógico

- Duplicação de Atributos → inclusão de atributos de uma tabela em outra



- Particionamento Vertical → divisão de atributos de uma tabela em múltiplas tabelas com mesma chave primária em relacionamento 1 : 1



Avaliação dos requisitos de projeto físico para modificação dos projetos conceitual e lógico

- Particionamento Horizontal → distribuição de tuplas de uma tabela em múltiplas tabelas com os mesmos atributos

PROFESSOR

CPF	Nome	Sexo	Salario	Departamento
12345678900	Roberto Machado	M	1200.00	1
12345678901	Manuela Costa	F	2700.00	3

PROFESSOR TOP

CPF	Nome	Sexo	Salario	Departamento
21345678900	Carlos A. Martins	M	3200.00	1
32145678900	Ana Maria Freitas	F	7500.00	2

Sintonia de Consulta

Avaliação dos requisitos de projeto físico para reescrever consultas

- Indícios → sinais de que consultas precisam ser reescritas
  - Plano de Execução → índices relevantes não estão sendo usados
  - Paginação → emissão de muitas solicitações de I/O
- Casos Típicos → situações que tipicamente demandam reescrita de consulta
  - Parsing → ordem aleatória de tabelas no FROM e operações no WHERE
  - Comparação → NULL, *substring* e campos de domínios diferentes

```
SELECT  A.CPF, B.Nome
  FROM  PROFESSOR A, DEPARTAMENTO B
 WHERE  A.Depto IS NOT NULL
   AND  B.Nome LIKE '%TI%'
   AND  A.Salario = B.Numero;
```

Sintonia de Consulta

Avaliação dos requisitos de projeto físico para reescrever consultas

- Casos Típicos → situações que tipicamente demandam reescrita de consulta
  - Consultas Aninhadas → operadores all, any, some, in e exists

```
SELECT CPF, Nome FROM PROFESSOR
 WHERE Depto IN (SELECT Numero FROM DEPARTAMENTO);
```

```
SELECT A.CPF, B.Nome FROM PROFESSOR A
 WHERE EXISTS (SELECT * FROM DEPARTAMENTO B
                 WHERE A.Depto = B.Numero);
```

  - Deduplicação → operador distinct

```
SELECT DISTINCT Nome FROM PROFESSOR;
```

Sintonia de Consulta

Avaliação dos requisitos de projeto físico para reescrever consultas

- Casos Típicos → situações que tipicamente demandam reescrita de consulta
  - Condição Disjuntiva → operador or

```
SELECT CPF, Nome FROM PROFESSOR
 WHERE Sexo = 'M' OR Salario > 2000,00;


SELECT CPF, Nome FROM PROFESSOR
 WHERE Sexo = 'M'
UNION
SELECT CPF, Nome FROM PROFESSOR
 WHERE Salario > 2000,00;
```

Avaliação dos requisitos de projeto físico para reescrever consultas

- Casos Típicos → situações que tipicamente demandam reescrita de consulta
  - Condição Complexa → operadores AND e OR

```
SELECT CPF, Nome FROM PROFESSOR
 WHERE Depto = 1
   AND ((Salario BETWEEN 1000,00 AND 2000,00) OR
        (Salario BETWEEN 5000,00 AND 7000,00));

SELECT CPF, Nome FROM PROFESSOR
 WHERE Depto = 1 AND (Salario BETWEEN 1000,00 AND 2000,00)
UNION
SELECT CPF, Nome FROM PROFESSOR
 WHERE Depto = 1 AND (Salario BETWEEN 5000,00 AND 7000,00);
```

Avaliação dos requisitos de projeto físico para reescrever consultas

- Casos Típicos → situações que tipicamente demandam reescrita de consulta
  - Consulta Complexa → subconsultas

```
SELECT A.CPF, A.Nome FROM PROFESSOR A
 WHERE A.Salario > (SELECT AVG(B.Salario) FROM PROFESSOR B
                    WHERE A.Depto = B.Depto);


  SELECT    Depto, AVG(Salario) AS Media INTO TEMP
    FROM    PROFESSOR
GROUP BY    Depto;

SELECT  A.CPF, A.Nome FROM PROFESSOR A, TEMP B
 WHERE  A.Salario > B.Media
   AND  A.Depto = B.Depto;
```

PROCESSAMENTO DE TRANSAÇÃO

Transação

Conjunto de operações de acesso ao BD, constituindo uma unidade lógica

- Exemplo → saque de dinheiro em caixa eletrônico
  - Múltiplas operações de recuperação e atualização de dados
- Todas as operações são confirmadas ou abortadas como um bloco único
  - Somente Leitura → somente operações de recuperação de dados
  - Leitura-Escrita → contém operações de atualização de dados
  - Confirmada → operações concluídas e dados registrados permanentemente
  - Abortada → operações inócuas, sem qualquer efeito

Tipicamente executada em sistema multiusuário de forma intercalada, em ambiente multiprogramação, ou paralela em hardware com múltiplas CPUs

Transação

Processa itens de dados de forma concorrente

- Item → arquivo, bloco, registro, campo
- Concorrência → múltiplas transações concorrendo pela CPU
  - Maior Concorrência → mais transações executadas concomitantemente
  - Controle de Concorrência → garantia de independência de execução
- Granularidade → tamanho do item de dados
  - Baixa (Fina) → tamanho menor, maior concorrência, mais carga sobre o sistema de controle de concorrência
  - Alta (Grossa) → tamanho maior, menor concorrência, menos carga sobre o sistema de controle de concorrência

Transação

Executa dois tipos básicos de operação sobre itens de dados

- Leitura $r(x) \rightarrow$ copia item x do disco para variável de programa
  - Encontrar endereço de bloco de disco que contém x
  - Copiar bloco para *buffer* em memória primária
  - Copiar item x do *buffer* para variável de programa x
- Escrita $w(x) \rightarrow$ copia item x da variável de programa para disco
  - Encontrar endereço de bloco de disco que contém x
  - Copiar bloco para *buffer* em memória primária
  - Copiar item x da variável de programa x para *buffer*
  - Copiar *buffer* de memória para bloco de disco

Concorrência

Execução concorrente de múltiplas transações pode resultar em problemas

- Atualização Perdida → transações intercaladas escrevem o mesmo item, tal que a atualização de uma transação sobre o item é perdida por sobrescrição do mesmo item feita por outra transação

$$\begin{array}{lcccccc} T_1 \rightarrow & r(x) & & r(y) & & x = x + y & & w(x) \\ T_2 \rightarrow & & r(x) & & r(y) & & x = x - y & & w(x) \end{array}$$

Para as transações T_1 e T_2, considerando valores iniciais $x = 10$ e $y = 5$

- Execução sequencial → valor final de $x = 10$
- Execução concorrente → valor final de $x = 5$

A operação $w(x)$ de T_1 foi sobrescrita por $w(x)$ de T_2

Concorrência

Execução concorrente de múltiplas transações pode resultar em problemas

- Leitura Suja → uma transação atualiza um item e falha posteriormente, sendo que nesse meio tempo, outra transação lê o item atualizado (sujo)

$$
\begin{array}{llllll}
T_1 \rightarrow r(x) & r(y) & x = x + y & w(x) & & a \\
T_2 \rightarrow & & & & r(x) \quad r(y) & x = x - y \quad w(x)
\end{array}
$$

Para as transações T_1 e T_2, considerando valores iniciais $x = 10$ e $y = 5$:

- Execução sequencial → valor final de $x = 5$
- Execução concorrente → valor final de $x = 10$

A operação $r(x)$ de T_2 fez uma leitura suja de $w(x)$ de T_1, que falhou

Concorrência

Execução concorrente de múltiplas transações pode resultar em problemas

- Leitura Não Repetitiva → a mesma transação lê valores diferentes para o mesmo item em momentos diferentes, uma vez que outras transações alteraram o valor do item nesse meio tempo

$$\begin{array}{lllllll} T_1 \rightarrow & r(x) & & & & & r(x) \\ T_2 \rightarrow & & r(x) & r(y) & x = x - y & w(x) & \end{array}$$

A primeira e a última operação $r(x)$ de T_1 leem valores diferentes de x não escritos por T_1, o que pode ocasionar um problema na lógica de processamento se um teste condicional $(x = x)$ for executado, por exemplo

Concorrência

Execução concorrente de múltiplas transações pode resultar em problemas

- Resumo Incorreto → uma transação calcula uma função de agregação sobre itens que estão sendo atualizados por outras transações, provendo valores de resumo incorretos

Técnicas de controle de concorrência resolvem os problemas, garantindo execução concorrente e independente de transações

- Bloqueio → técnica baseada em bloqueios de leitura e escrita em itens
  - Restringe a concorrência
  - Transações bloqueiam e desbloqueiam itens quando necessário
  - Sujeito a problemas de travamento (*deadlock*) e espera indefinida (*starvation*)

Falha

Execução concorrente de transações está sujeita a diferentes tipos de falhas

- Sistema → hardware, software ou rede
- Operação → interrupção do usuário, lógica de programação
- Concorrência → técnica de controle de concorrência
- Condição de Exceção → programada na própria transação
- Disco → blocos de dados perdidos
- Catástrofe → *blackout*, incêndio, roubo, formatação acidental de disco

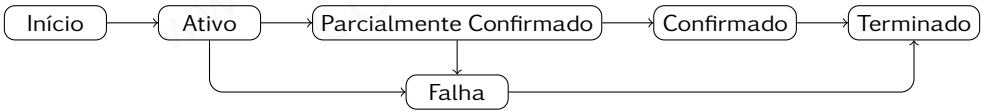
Sistemas de banco de dados estão preparados para lidar com falhas

- Logging → registro histórico de transações
- Dumping → cópia de segurança do banco de dados

Adicionalmente às operações de leitura e escrita sobre itens de dados, existem operações necessárias ao processamento de transações

- Begin Transaction (b) → marca o início da transação
- End Transaction (e) → marca o fim da transação
- Commit (c) → marca o ponto de confirmação de operações
- Abort (a) → ou *rollback*, marca o ponto de anulação de operações

Uma transação só é considerada terminada quando falhar, ou todas as suas operações tiverem sido executadas e registradas do arquivo de *log* do sistema



Processamento de Transações

Transações possuem propriedades que devem ser mantidas no processamento

- Atomicidade → unidade atômica, executada integralmente ou não executada de forma alguma
- Consistência → restrições especificadas no esquema devem continuar sendo respeitadas após processamento
- Isolamento → independência de execução, sem sofrer interferência de transações concorrentes
  - Nível 0 → sem leitura suja
  - Nível 1 → sem atualização perdida
  - Nível 2 → níveis 0 + 1
  - Nível 3 → nível 2, com leitura repetitiva
- Durabilidade → alterações confirmadas devem persistir

Escalonamento (Schedule) → intercalação de operações de transações distintas para execução simultânea

$$S_a \rightarrow b_1, r_1(x), b_2, r_2(x), r_1(y), w_2(x), w_1(y), c_2, e_2, a_1, e_1$$

- $S_a \rightarrow$ escalonamento a
- $o_i \rightarrow$ operação $o \in \{a, b, c, e, r, w\}$ realizada pela transação i
- $x, y \rightarrow$ itens de dados lidos ou escritos pelas transações
- Tipicamente b e e não são representadas, presumindo que b ocorre logo antes da primeira operação da transação, e e logo depois da última

$$S_a \rightarrow r_1(x), r_2(x), r_1(y), w_2(x), w_1(y), c_2, a_1$$

Escalonamento de Transações

Operações em um escalonamento podem estar em situação de conflito

$$S_a \rightarrow r_1(x), r_2(x), r_1(y), w_2(x), w_1(y), c_2, a_1$$

- Pertencem a transações diferentes
- Operam sobre o mesmo item
- Ao menos uma delas é w

Escalonamento de Transações

Completo → possui todas as operações de cada transação

- Operações são exatamente as mesmas das transações originais
- A ordem das operações nas transações originais é preservada
- Operações em conflito precedem ou sucedem umas as outras
- Exemplo → para as transações T_1 e T_2

$$T_1 \rightarrow r(x), r(y), w(y), a$$
$$T_2 \rightarrow r(x), w(x), c$$

o escalonamento S_a é completo

$$S_a \rightarrow r_1(x), r_2(x), r_1(y), w_2(x), w_1(y), c_2, a_1$$

Improváveis, novas transações são incorporadas a escalonamentos existentes

Recuperável → transação confirmada não será desfeita, garantindo durabilidade

- Leitura Suja → pode demandar reversão de confirmação

$$S_a \rightarrow r_1(x), r_2(x), w_1(x), r_1(y), w_2(x), r_1(x), c_1, c_2$$

- Nenhuma transação T que leia um item escrito por outra transação T' pode confirmar antes de T'

$$S_a \rightarrow r_1(x), r_2(x), w_1(x), r_1(y), w_2(x), r_1(x), c_2, c_1$$

Escalonamento de Transações

Serial → sem intercalação, com todas as operações de uma transação precedendo todas as operações de outra

$$S_a \rightarrow r_1(x), w_1(x), r_1(y), r_1(x), c_1, r_2(x), w_2(x), c_2$$

- Isolamento → transação não sofre interferência de outra
- Concorrência → limitada, gerando ociosidade de CPU

Não Serial → operações de transações intercaladas

$$S_a \rightarrow r_1(x), r_2(x), w_1(x), r_1(y), w_2(x), r_1(x), c_2, c_1$$

- Isolamento → não é garantido
- Concorrência → melhor aproveitamento de CPU

Escalonamento de Transações

Serialização → processo de determinação de escalonamentos não seriais que sejam equivalentes a seriais, garantindo isolamento

- Multiplicidade → $n!$ escalonamento seriais possíveis para n transações, e um número muito maior de escalonamentos não seriais possíveis
  - Serializável → não serial equivalente a um serial
  - Não Serializável → não equivalente a qualquer serial
- Escalonamentos serializáveis são **corretos**
- Equivalência → operações são aplicadas a itens na mesma ordem
  - Conflito → ordem de operações em conflito nos escalonamentos é a mesma
  - Visão → operações r tem a mesma visão de itens nos escalonamentos

Equivalência por Conflito → mesma ordem de operações em conflito

$$S_a \rightarrow r_1(x), w_1(x), r_1(y), w_1(y), r_2(x), w_2(x)$$

$$S_b \rightarrow r_2(x), w_2(x), r_1(x), w_1(x), r_1(y), w_1(y)$$

$$S_c \rightarrow r_1(x), r_2(x), w_1(x), r_1(y), w_2(x), w_1(y)$$

$$S_d \rightarrow r_1(x), w_1(x), r_2(x), w_2(x), r_1(y), w_1(y)$$

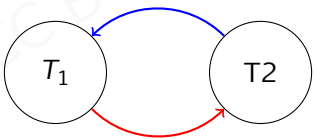
- S_d é serializável, enquanto S_c não é
  - S_c não equivalente a S_a → ordem $w_1(x)$ e $r_2(x)$ diferente
  - S_c não equivalente a S_b → ordem $w_2(x)$ e $r_1(x)$ diferente
  - S_d equivalente a S_a → mesma ordem para operações em conflito

Escalonamento de Transações

Grafo de Precedência → permite verificar se escalonamento é serializável sem precisar checar equivalência por conflito com $n!$ escalonamentos seriais

- Direcionado → ausência de ciclos indica escalonamento serializável
- Nós → transações do escalonamento
- Arestas → conflitos entre operações de transações
  - De transação sucedente para precedente, com operações em conflito

$$S_c \rightarrow r_1(x), r_2(x), w_1(x), r_1(y), w_2(x), w_1(y)$$



Processamento de Transações

SGBDs tipicamente permitem configuração de características de processamento de transações através do comando SET TRANSACTION

- ACESSO → operações permitidas em transações
  - RW → leitura e escrita (modo padrão)
  - RO → somente leitura
- ÁREA DE DIAGNÓSTICO → últimas n instruções SQL executadas
- ISOLAMENTO → nível de isolamento requerido no processamento
  - READ UNCOMMITED → $\neg$ atualização perdida
  - READ COMMITED → $\neg$ (atualização perdida $\vee$ leitura suja)
  - REPEATABLE READ → $\neg$ (atualização perdida $\vee$ leitura suja) $\wedge$ leitura repetitiva
  - SERIALIZABLE → $\neg$ (atualização perdida $\vee$ leitura suja $\vee$ resumo incorreto) $\wedge$ leitura repetitiva

CONTROLE DE CONCORRÊNCIA

Controle de Concorrência

Emprego de técnicas para garantia de isolamento entre transações

- Protocolos → garantem a serialização de escalonamentos
  - Bloqueio → aplica regras para bloqueio e desbloqueio de itens
  - Timestamp → ordena transações usando seus rótulos de tempo (*timestamp*)
  - Multiversão → armazena múltiplas versões de um item, escolhendo a mais apropriada para ser usada por cada transação em execução
  - Validação → checa *a posteriori* se execução ocorreu de forma isolada
  - Granularidade Múltipla → variação do bloqueio com ajuste de granularidade

Técnicas baseadas em protocolos de bloqueio são tipicamente adotadas por SGBDs comerciais

Protocolo de Bloqueio

Serialização por aplicação de regras para bloqueio e desbloqueio de itens

- Binário → duas operações de bloqueio
  - Lock $l(x)$ → altera estado do item para bloqueado pela transação, com outras transações sendo forçadas a esperar pelo desbloqueio
  - Unlock $u(x)$ → altera estado do item para livre, desbloqueando-o
  - Exclusividade → transações não acessam o mesmo item simultaneamente
- Ternário → três operações de bloqueio
  - Read Lock $rl(x)$ → altera estado para bloqueado para leitura, outras transações que demandarem escrita são forçadas a esperar desbloqueio
  - Write Lock $wl(x)$ → altera estado para bloqueado para escrita, outras transações são forçadas a esperar desbloqueio
  - Unlock $u(x)$ → altera estado do item para livre
  - Compartilhamento → transações podem realizar leitura simultaneamente

Protocolo de Bloqueio

Tabela de bloqueio registra emissão de solicitação de bloqueios para itens

- Regras → impostas pelo gerenciador de bloqueio
  - Transação precisa emitir bloqueio antes de operar sobre item
  - Transação precisa emitir desbloqueio após completar operações sobre o item
  - Transação não emite bloqueio se já tiver bloqueio sobre item
    - Promoção (*upgrade*) → bloqueio $rl(x)$ convertido para $wl(x)$
    - Descenso (*downgrade*) → bloqueio $wl(x)$ convertido para $rl(x)$
  - Transação não emite desbloqueio se não tiver bloqueio sobre item
  - Solicitação de bloqueio não atendida entra em fila de espera por desbloqueio

Aplicação de regras de bloqueio e desbloqueio não garante serialização

Regras adicionais para bloqueios e desbloqueios em fases são necessárias

Protocolo de Bloqueio

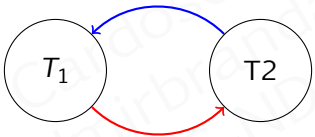
2PL → protocolo de bloqueio em duas fases (*2 Phase Locking*)

- Todas as solicitações de bloqueio de uma transação precedem sua primeira solicitação de desbloqueio
  - Espansão → bloqueios adquiridos, nenhuma liberação
    - Possibilidade de promoção de bloqueios
  - Retração → bloqueios liberados, nenhum bloqueio adquirido
    - Possibilidade de descenso de bloqueios
- Pode limitar concorrência
- Escalonamentos 2PL garantidamente serializáveis (ou seriais)

Protocolo de Bloqueio

Considere o escalonamento não serializável

$$S_a \rightarrow r_1(x), r_2(x), w_1(x), r_1(y), w_2(x), w_1(y)$$

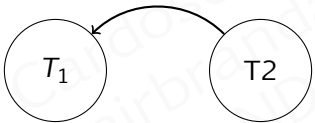


Aplicação do protocolo 2PL

$$S_a \rightarrow l_1(x), r_1(x), l_2(x), r_2(x), w_1(x), l_1(y), r_1(y), w_2(x), u_2(x), w_1(y), u_1(x), u_1(y)$$

Escalonamento $S_{a'}$ efetivamente executado a partir da aplicação 2PL sobre S_a

$$S_{a'} \rightarrow r_1(x), w_1(x), r_1(y), w_1(y), r_2(x), w_2(x)$$



Note que T_2 parou sua execução ao tentar bloquear x, já bloqueado por T_1

$$S_a \rightarrow l_1(x), r_1(x), l_2(x), r_2(x), w_1(x), l_1(y), r_1(y), w_2(x), u_2(x), w_1(y), u_1(x), u_1(y)$$

DEADLOCK → situação de travamento, impasse

$$S_a \rightarrow r_1(x), r_2(x), w_1(x), r_1(y), w_2(x), w_1(y)$$

Aplicação do protocolo 2PL com bloqueios ternários

$$S_a \rightarrow rl_1(x), r_1(x), rl_2(x), r_2(x), wl_1(x), w_1(x), rl_1(y), r_1(y), wl_2(x), w_2(x), u_2(x), wl_1(y), w_1(y), u_1(x), u_1(y)$$

Escalonamento $S_{a'}$ efetivamente executado com DEADLOCK

$$S_{a'} \rightarrow r_1(x), r_2(x), w_1(x), w_2(x)$$

Note que T_1 paralisa ao tentar $wl_1(x)$, aguardando T_2 liberar o $rl_2(x)$, e T_2 paralisa ao tentar $wl_2(x)$, aguardando T_1 liberar o $rl_1(x)$

Protocolo de Bloqueio

Resolução de DEADLOCK envolve prevenção, detecção ou presunção

- PREVENÇÃO → 2PL conservador ou estático
  - Transação emite solicitação de bloqueios que necessita antes de iniciar

$$S_a \rightarrow r_1(x), r_2(x), w_1(x), r_1(y), w_2(x), w_1(y)$$

Aplicação de protocolo 2PL estático com bloqueios ternários

$$S_a \rightarrow wl_1(x), wl_1(y), r_1(x), wl_2(x), r_2(x), w_1(x), r_1(y), w_2(x), u_2(x), w_1(y), u_1(x), u_1(y)$$

Escalonamento $S_{a'}$ efetivamente executado

$$S_{a'} \rightarrow r_1(x), w_1(x), r_1(y), w_1(y), r_2(x), w_2(x)$$

Resolução de DEADLOCK envolve prevenção, detecção ou presunção

- PREVENÇÃO → reescalonamento por *timestamp* (TS)
  - Transação mais nova, bloqueando item necessário para outra mais antiga, aborta e é reescalonada
  - Se T_i inicia antes de T_j, então $TS(T_i) < TS(T_j)$

$$S_a \rightarrow r_1(x), r_2(x), w_1(x), r_1(y), w_2(x), w_1(y)$$

$$TS(T_1) < TS(T_2)$$

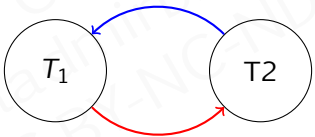
T_1 não consegue $wl_1(x)$ por $rl_2(x)$ de T_2, T_2 aborta e é reescalonada

$$S_{a'} \rightarrow r_1(x), w_1(x), r_1(y), w_1(y)$$

Protocolo de Bloqueio

Resolução de DEADLOCK envolve prevenção, detecção ou presunção

- DETECÇÃO → transações em impasse, mais nova aborta e é reescalonada
  - Transação mais nova, provavelmente com menos operações executadas
  - GRAFO DE ESPERA → nós são transações, arestas direcionadas são bloqueios em espera, ciclo indica impasse



$$S_a \rightarrow r_1(x), r_2(x), w_1(x), r_1(y), w_2(x), w_1(y)$$

Ciclo entre T_1 e T_2, T_2 aborta e é reescalonada por ser a mais nova

Protocolo de Bloqueio

Resolução de DEADLOCK envolve prevenção, detecção ou presunção

- PRESUNÇÃO → transação demorada aborta e é reescalonada
  - TEMPO LIMITE (*timeout*) → registro de tempo
  - Presume-se impasse na transação se seu tempo de execução exceder *timeout*
  - Simples, evita sobrecarga

Protocolo de Bloqueio

STARVATION → situação de espera indefinida, inanição

- Transação constantemente abortada e reescalonada
- No pior caso, nunca executada
- Solução
  - Priorização proporcional ao tempo de espera
  - Manutenção de *timestamp* original, pré-reescalonamento

Protocolo de Timestamp

Baseando-se no princípio da equivalência por conflito, ordem de acesso a um item em operações em conflito não pode violar ordem de *timestamp*

- Dois registros de *timestamp* para cada item
  - Leitura $rTS(x) \rightarrow$ *timestamp* da transação mais nova a ler x
  - Escrita $wTS(x) \rightarrow$ *timestamp* da transação mais nova a escrever x
- Regras $\rightarrow$ garantem ordenação por *timestamp*
  - T emite solicitação $r(x)$
    - Se $wTS(x) > TS(T)$, rejeite $r(x)$, aborte e reverta T
    - Senão, aceite $r(x)$, $rTS(x) = max(rTS(x), TS(T))$
  - T emite solicitação $w(x)$
    - Se $(rTS(x) > TS(T) \vee wTS(x) > TS(T))$, rejeite $w(x)$, aborte e reverta T
    - Senão, aceite $w(x)$, $wTS(x) = TS(T)$

Não gera deadlock, mas tipicamente implica em menor concorrência

Protocolo de Timestamp

Não pode haver conflito entre operação corrente de transação mais antiga e operação anterior de transação mais nova

$$S_a \rightarrow r_1(x), r_2(x), w_1(x), r_1(y), w_2(x), w_1(y)$$

$$TS(T_1) = 1 < TS(T_2) = 2$$

Aplicando-se o protocolo de *timestamp*

$$rts(x) \rightarrow 0 \;\; 1 \;\; 2 \qquad wts(x) \rightarrow 0 \;\; 2 \qquad rts(y) \rightarrow 0 \qquad wts(y) \rightarrow 0$$

Escalonamento $S_{a'}$ efetivamente executado

$$S_{a'} \rightarrow r_2(x), w_2(x)$$

Referências Bibliográficas

[1] Elmasri, Ramez; Navathe, Sham. *Fundamentals of Database Systems*. 7ed. Pearson, 2016.

[2] Silberschatz, Abraham; Korth, Henry F.; Sudarshan, S. *Database System Concepts*. 6ed. McGraw-Hill, 2011.

[3] Date, Christopher J. *An Introduction to Database Systems*. 8ed. Pearson, 2004.