Processamento de Consultas

- Processar consultas envolve:
  - Traduzir consultas expressas em linguagens de alto nível (como SQL) em expressões que podem ser implementadas no nível físico do sistema de banco de dados (nível de tabelas);
  - Otimizar a expressão destas consultas;
  - Avaliar a base de dados de acordo com as diretrizes da consulta, para fornecer o resultado.

Processamento de Consultas

- Consulta SQL
  - É adequada para uso humano;
  - Não adequada ao processamento pelo SGBD:
    - Não descreve uma seqüência de passos (procedimento) a ser seguida;
    - Não descreve uma estratégia eficiente para a implementação de cada passo no que tange o acesso em nível físico (arquivos do BD).
- Cabe ao SGBD deve se preocupar com este processamento → módulo Processador de Consultas.

Módulo Processador de Consultas

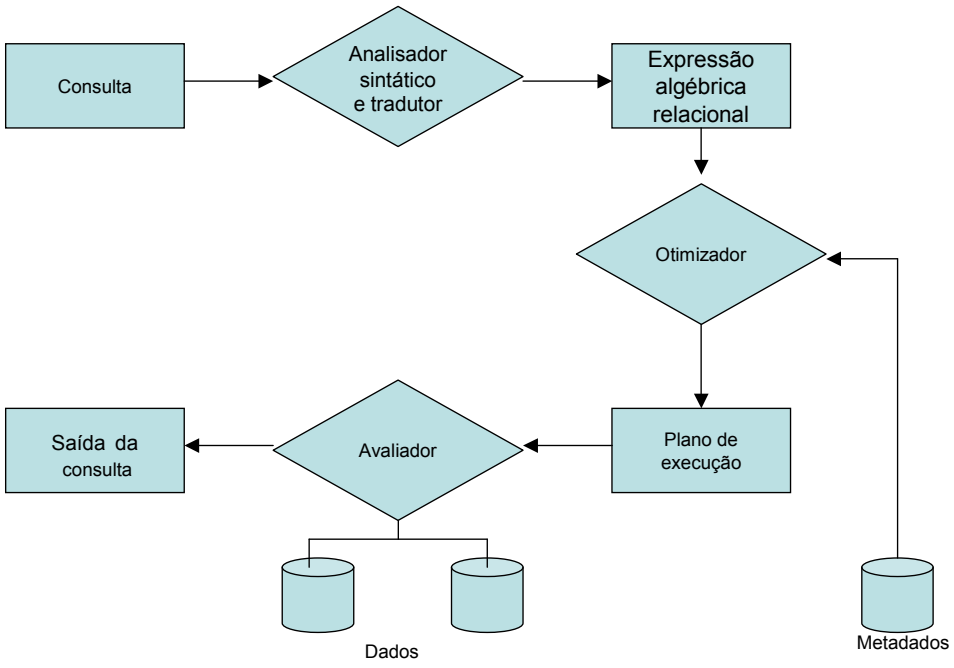
- Objetivo: Otimização do processamento de uma consulta
  - Tradução, transformação e geração de uma estratégia (plano) de execução;
  - Estratégia de acesso:
    - Considera algoritmos predefinidos para implementação de passos do processamento e estimativas sobre os dados.
- O esforço é valido, pois quase sempre

$$T_x << T_y$$

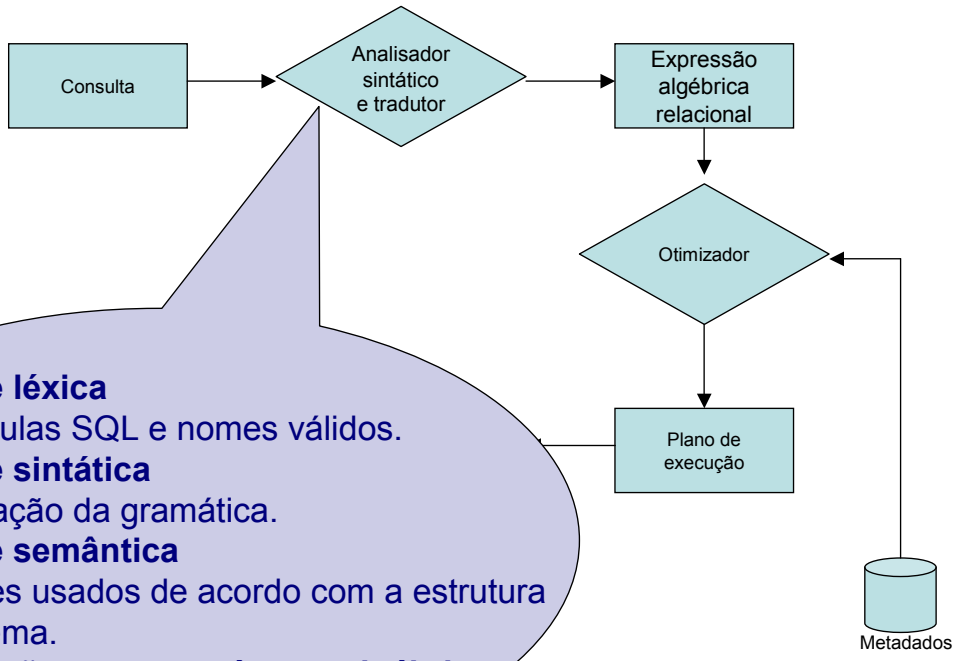
T_x = Tempo para definir e executar uma estratégia otimizada de processamento;

T_y = Tempo para executar uma estratégia não-otimizada de processamento.

Passos no Processamento de Consultas

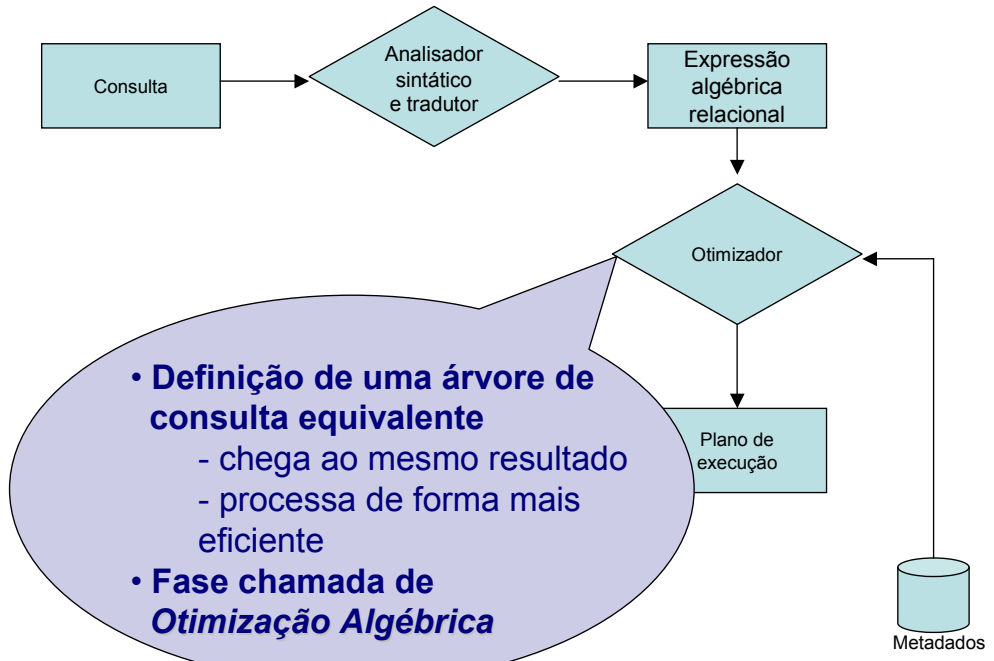


Passos no Processamento de Consultas



- **Análise léxica**
  - cláusulas SQL e nomes válidos.
- **Análise sintática**
  - validação da gramática.
- **Análise semântica**
  - nomes usados de acordo com a estrutura do esquema.
- **Conversão para uma árvore algébrica da consulta**

Passos no Processamento de Consultas

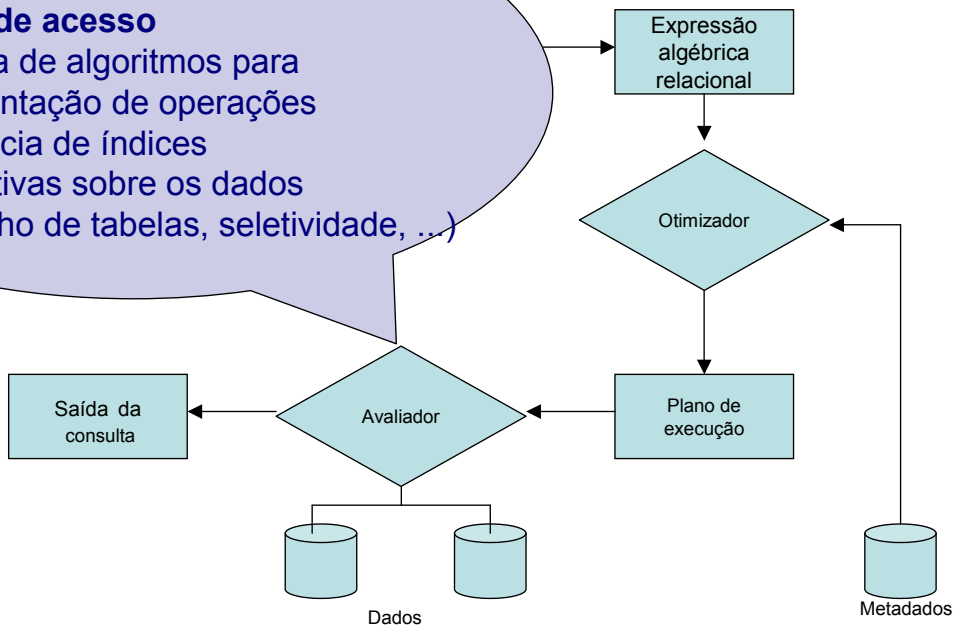


- **Definição de uma árvore de consulta equivalente**
  - chega ao mesmo resultado
  - processa de forma mais eficiente
- **Fase chamada de *Otimização Algébrica***

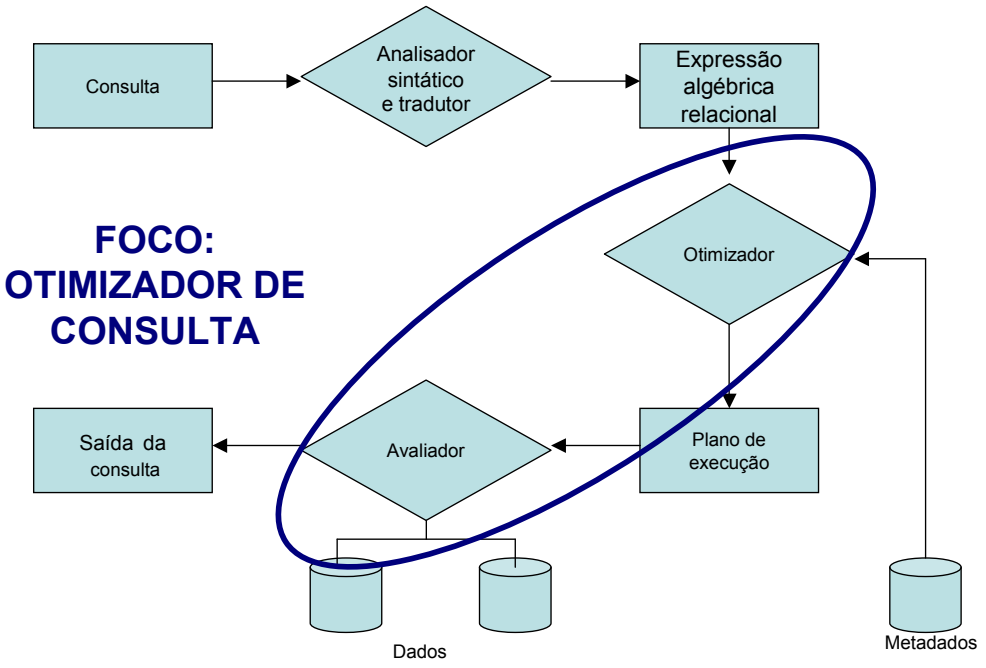
Passos no Processamento de Consultas

Análise de alternativas de definição de estratégias de acesso

- escolha de algoritmos para implementação de operações
- existência de índices
- estimativas sobre os dados (tamanho de tabelas, seletividade, ...)



Passos no Processamento de Consultas



Exemplo Introdutório

- Suponha a consulta:

  select saldo
  from conta
  where saldo < 2500;

- Esta pode ser traduzida nas duas expressões algébricas relacionais diferentes:

  σ saldo < 2500 (π saldo (conta))

  π saldo (σ saldo < 2500(conta))

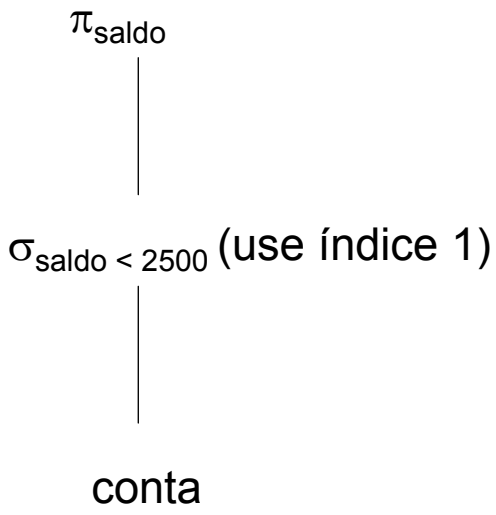
Exemplo Introdutório

- Além desta variação, é possível executar cada operação algébrica relacional usando um entre diversos algoritmos diferentes. Por exemplo:
  - Para executar a seleção, podemos procurar em todas as tuplas de *conta* a fim de encontrar as tuplas com saldo menor 2.500.
  - Se um índice árvore-B+ estiver disponível no atributo *saldo*, podemos usar o índice em vez de localizar as tuplas.
- É necessário prover as expressões algébricas de anotações que permitam especificar como serão avaliadas.

Exemplo Introdutório

- Uma operação algébrica relacional anotada com instruções sobre como ser avaliada é chamada de *avaliação primitiva*.
- Vária avaliações primitivas podem ser agrupadas em *pipeline*, e executadas em paralelo.
- Uma seqüência de operações primitivas é um plano de execução de consulta ou plano de avaliação de consulta.

Exemplo Introdutório



- Uma vez escolhido o plano de consulta, a consulta é avaliada com aquele plano e o resultado da consulta é produzido

Otimização de Consultas

- Existem 2 técnicas básicas para otimização de consultas:
  - As baseadas em heurísticas para a ordenação de acesso ao banco de dados, que participarão da estratégia de acesso;
  - e as que estimam sistematicamente o custo de estratégias de execução diferentes e escolhem o plano de execução com o menor custo estimado.

Catálogo de Informações para Estimativa de Custo

- n_r: é o número de tuplas na relação r;
- b_r: é o número de blocos que contêm tuplas da relação r;
- s_r: é o tamanho em bytes de uma tupla da relação r;
- f_r: é o fator de bloco da relação r, ou seja, o número de tuplas da relação r que cabe em um bloco;
- V(A,r): é o número de valores distintos que aparecem na relação r para o atributo A. Esse valor é igual ao tamanho (em número de tuplas) de $\pi_A(r)$. Se A é uma chave para a relação r, V(A,r) é n_r.

Catálogo de Informações para Estimativa de Custo

- SC(A,r): é a cardinalidade de seleção (seletividade) do atributo A da relação r.
  - Dados uma relação r e um atributo A da relação, SC(A,r) é o número médio de registros que satisfazem uma condição de igualdade no atributo A, dado que pelo menos um registro satisfaz a condição de igualdade.
  - Exemplo:
    - SC(A,r) = 1 se A é um atributo-chave de r;
    - Para um atributo que não é chave, estimamos que os valores distintos de V(A,r) são distribuídos uniformemente entre as tuplas, produzindo SC(A,r) = (n_r / V(A,r))

Catálogo de Informações para Estimativa de Custo

- As duas últimas estatísticas podem ser estendidas de forma a valer para um conjunto de atributos, ao invés de valer para apenas um atributo.

- Se as tuplas da relação r estiverem armazenadas fisicamente juntas em um arquivo, a seguinte equação é válida:

$$B_r = [n_r, f_r]$$

Catálogo de Informações para Estimativa de Custo

- Informações sobre índices:
  - f_i: é o *fan-out* (número de ponteiros) médio dos nós internos do índice i para índices estruturados em árvore, como árvores B^+;
  - HT_i: é o número de níveis no índice i, ou seja, a altura do índice i.
  - LB_i: é o número de blocos de índice de nível mais baixo no índice i, ou seja, o número de blocos no nível de folha do índice (o número de blocos que contém os registros folha de um índice).

Catálogo de Informações para Estimativa de Custo

- As variáveis estatísticas são usadas para estimar o tamanho do resultado e o custo para várias operações e algoritmos.
- A estimativa de custo do algoritmo A é E_A.
- Para manter as estatísticas precisas, toda vez que uma relação for modificada tem-se que atualizar as estatísticas. Contudo, a maioria do sistema não atualiza as estatísticas em todas as modificações. Atualiza-as periodicamente.
- Quanto mais informações forem utilizadas para estimar o custo da consulta e quanto mais precisas forem essas informações, melhores serão as estimativas de custo.

Medidas do Custo de uma Consulta

- O custo de uma consulta pode ser estimado de diversas formas:
  - Por acessos a disco;
  - Por tempo de uso da CPU;
  - Pelo tempo de comunicação nos BD paralelos e/ou distribuídos;

- O tempo de execução de um plano poderia ser usado para estimar o custo da consulta, contudo em grandes sistemas de BD, utiliza-se o número de acessos a disco, porque estes estabelecem o tempo crítico de execução do plano (já que são lentos quando comparados às operações realizadas em memória).

Medidas do Custo de uma Consulta

- Para simplificar nossos cálculos assumiremos que todas as transferências de blocos (do disco para memória) têm o mesmo custo. Desconsideraremos o tempo de latência e o tempo de busca. Também desconsideramos o custo de escrever o **resultado final** de uma operação de volta para o disco.

- Os custos dos algoritmos dependem significativamente do tamanho do buffer na memória principal. No melhor caso, todos os dados podem ser lidos para o buffer e o disco não precisa ser acessado novamente. **No pior caso, supomos que o buffer pode manter apenas alguns blocos de dados – aproximadamente um bloco por relação.** Geralmente faremos a suposição do pior caso.

Operação de Seleção

- É a varredura de arquivos: o operador de mais baixo nível para se ter acesso aos dados.
- São algoritmos de procura que localizam e recuperam os registros que estão de acordo com uma condição de seleção.
- Tem-se vários algoritmos diferentes, que variam de acordo com a complexidade da seleção e o uso ou não de índices.

Operação de Seleção

- Exemplo de algoritmos usados na implementação do operador ***select***:
  - Busca Linear (ou força bruta);
  - Busca Binária;
  - Utilização de índice primário (atributo chave);
  - Utilização de índice primário para recuperar múltiplos registros (atributo chave);
  - Utilização de um índice *cluster* para recuperar múltiplos registros (atributo não chave);
  - Utilização de um índice secundário (Árvore B+) em uma comparação de igualdade;
  - Busca para seleções complexas

Operação de Seleção

- Busca para seleções complexas:
  - Se uma condição de uma instrução *select* é uma condição conjuntiva – ou seja, é formada por diversas condições simples conectadas pelo conectivo lógico AND, o SGBD pode usar os seguintes métodos:
    - Seleção conjuntiva utilizando um índice individual;
    - Seleção conjuntiva utilizando um índice composto;
    - Seleção conjuntiva por meio da interseção de registros.

Operação de Seleção

- Busca para seleções complexas:
  - Se uma condição de uma instrução *select* é uma condição disjuntiva – ou seja, é formada por diversas condições simples conectadas pelo conectivo lógico OR, a otimização é mais simples.
  - Pouca otimização pode ser feita, pois os registros que satisfazem a condição disjuntiva são a união dos registros que satisfazem as condições individuais.

Operação de Seleção

- Veremos dois deles (os básicos):
  - Aquele que envolve uma Busca Linear;
  - Aquele que envolve uma Busca Binária.
- Considere uma operação de seleção em uma relação cujas tuplas são armazenadas juntas em um único arquivo.

Seleção por Busca Linear – A1

- Em uma busca linear, cada bloco de arquivo é varrido e todos os registros são testados para verificar se satisfazem a condição de seleção.
- Como todos os blocos precisam ser lidos, $E_{A1} = b_r$.
- No caso da seleção ser aplicada em um atributo-chave, podemos supor que a metade dos blocos é varrida antes de o registro ser encontrado, ponto no qual a varredura termina. A estimativa então será $E_{A1} = (b_r/2)$.

Seleção por Busca Binária – A2

- Se o arquivo é ordenado em um atributo e a condição de seleção é uma comparação de igualdade neste atributo, podemos usar uma busca binária para localizar os registros que satisfazem a seleção.

- Neste caso, a estimativa é:

$$E_{A2} = [\log_2(b_r)] + [SC(A,r)/f_r] - 1$$

  - O primeiro termo $[\log_2(b_r)]$ contabiliza o custo para localizar a primeira tupla por meio da busca binária nos blocos;
  - O número total de registros que satisfarão a seleção é SC(A,r), e esses registros ocuparão $[SC(A,r)/f_r]$ blocos, dos quais um já havia sido recuperado (por isso o -1).
  - Se a condição de igualdade estiver em um atributo-chave, então SC(A,r) = 1, e a estimativa se reduz a $E_{A2} = [\log(b_r)]$.

Cálculo do Custo da Busca Binária

- Acesso aos blocos:
  - Primeiro acesso (ao bloco central) → não encontro o registro procurado;
  - Segundo acesso (ao bloco central do lado esquerdo ou direito)
  - ....
  - Até o pior caso (nono acesso), o registro é encontrando na última divisão disponível (ou não é encontrado).

- Para 500 blocos:
  - 500 → 250 → 125 → 62,5 → 31,25 → 15,62 → 7,8 → 3,9 → 1,9 (nove divisões)

- Cálculo: $\log_2(500) = 9 \rightarrow 2^9 = 516$ (=~ 500)

Exemplo de Seleção por Busca Binária

- Suponha as seguintes informações estatísticas para uma relação *conta:*
  - f_{conta} = 20 (ou seja, 20 tuplas de *conta* cabem em um único bloco);
  - V(nome_agência, conta) = 50 (ou seja, existem 50 agências com nomes diferentes);
  - V(saldo, conta) = 500 (ou seja, existe 500 valores diferentes de saldos nesta relação);
  - n_{conta} = 10.000 (ou seja, a relação *conta* possui 10.000 tuplas).
- Considere a consulta:

$$\sigma_{\text{nome\_agência = Perryridge}} (\text{conta})$$

Exemplo de Seleção por Busca Binária

- Como a relação tem 10.000 tuplas, e cada bloco mantém 20 tuplas, o número de blocos é b_{conta} = 500 (10.000/20);
- Uma varredura de arquivo simples faria 500 acessos a blocos, supondo que o atributo da condição não fosse atributo-chave. Senão, seriam em média 250 acessos;
- Suponha que *conta* esteja ordenado por *nome_agência*.
- Como V(nome_agência, conta) = 50, esperamos que 10.000/50=200 tuplas da relação conta pertençam à agência Perryridge;
- Essas tuplas caberiam em 200/20 = 10 blocos;
- Uma busca binária para encontra o primeiro registro $[\log_2(500)] = 9$;
- Assim o custo total seria: 9 + 10 – 1 = 18 acessos a bloco.

Operação de Seleção

- A otimização de consulta para uma operação SELECT é necessária principalmente em condições de seleção conjuntiva, sempre que **mais de um** dos atributos envolvidos nas condições possuírem um caminho de acesso.
- O otimizador deve escolher o caminho de acesso que recupera o menor número de registros (gera blocos de respostas menores), de maneira mais eficiente.
- As seleções que separam o menor número de tuplas devem ser realizadas primeiro.
- Na escolha entre múltiplas opções o otimizador considera também a **seletividade** de cada condição.

Classificação

- A ordenação é bastante importante, uma vez que o algoritmo é utilizado:
  - Na implementação do *order by*.
  - Como um componente-chave nos algoritmos de *sort-merge* usado no *join, union e intersection* e em algoritmos de eliminação de duplicatas para a operação *project*.
- A ordenação pode ser evitada se um índice apropriado existir de forma a permitir o acesso ordenado aos registros.

Classificação

- Formas de ordenação:
  - Lógica: construção de um índice na chave de classificação, o qual será usado para **ler** a relação na ordem de classificação.
    - A leitura de tuplas na ordem de classificação pode conduzir a um acesso de disco para cada tupla.
  - Física: as tuplas são gravadas de forma ordenada no disco.

Classificação

- O problema de classificação pode ser tratado sob duas condições:
  - Quando a relação cabe completamente na memória principal:
    - Técnicas padrões de classificação (**quicksort** entre outras) podem ser usadas.
  - Quando a relação é maior que a memória principal → *classificação externa*:
    - Algoritmo comum: **sort-merge externo**
      - Para entendê-lo, considere **M** o número de *frames* de páginas no *buffer* da memória principal (o número de blocos de disco cujos conteúdos podem ser colocados no *buffer* da memória principal).

Ordenação Externa

- A **ordenação externa** é adequada para manipular arquivos de registros grandes, que são armazenados em disco e que não cabem inteiramente na memória principal.

- A ordenação nesse algoritmo é feita por partes – estratégia *merge-sort.*

- Fases:
  - Fase de ordenação;
  - Fase de fusão.

Inicialização:

i ← 1;
j ← b; (tamanho do arquivo em blocos)
k ← n_0; (tamanho do buffer em blocos)
m ← ⌈ (j/k) ⌉ (maior inteiro)

Se no buffer cabem 3 blocos, e o arquivo possui 11 blocos, será preciso 4 iterações da fase de ordenação. As 3 primeiras ordenarão 9 blocos e a última ordenará 2 blocos.

Fase de ordenação

Enquanto (i <= m) **faça**
{
leia os próximos **k** blocos do arquivo para o buffer ou se houver menos do que **k** blocos restantes, leia todos os blocos restantes;
ordene os registros no buffer e grave-os como um sub-arquivo temporário (utilize um algoritmo de ordenação interna, como o bubble ou quicksort, por exemplo);
i ← i + 1;
}

Fase de fusão: fundir os subarquivos até que reste apenas 1

Inicialização

Temos 4 subarquivos ordenados (m = 4 e k = 3).

```
i ← 1;
p ← ⌈ log_{k-1} m ⌉; (p é o número de passagens da fase de fusão)
j ← m;
```

p = 2

```
enquanto (i <= p) faça
    {
        n ← 1;
        q ← ⌈ (j/(k-1)) ⌉; (número de subarquivos a gravar nesta passagem)
        enquanto (n <= q) faça
        {
                ler os próximos k-1 subarquivos ou os subarquivos
        restantes (da passagem anterior), um bloco de cada arquivo por vez;
                fundir e gravar no novo subarquivo um bloco por vez
                n ← n + 1;
        }
        j ← q;
        i ← i + 1;
    }
```

q = 2 e depois 1