PUCRS
Pontifícia Universidade Católica
do Rio Grande do Sul

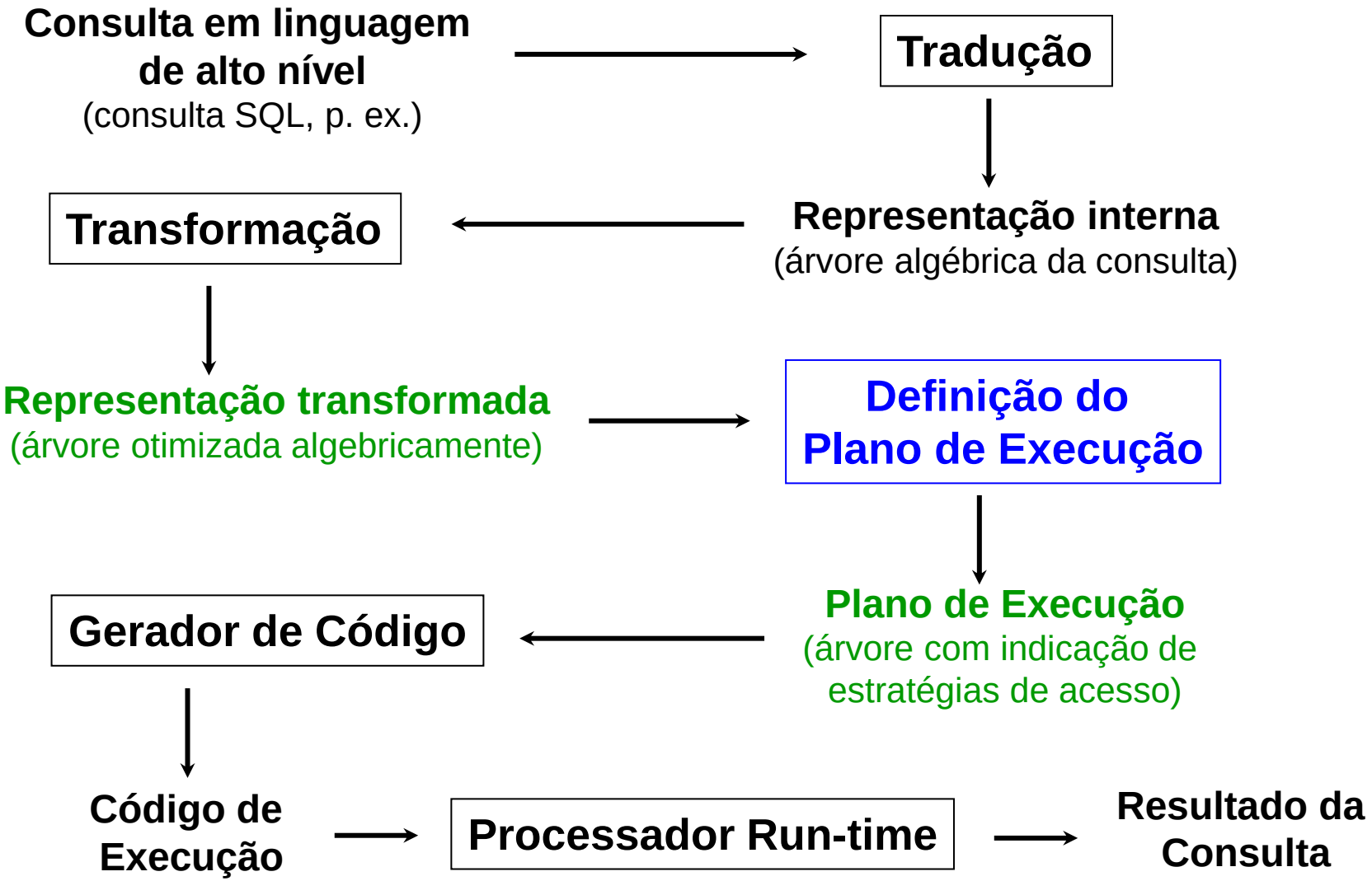
ESCOLA
POLITÉCNICA

Implementação de Bancos de Dados

98H00-04 - Infraestrutura para Gestão de Dados

Prof. Msc. Eduardo Arruda
eduardo.arruda@pucrs.br

Etapas de Processamento



Etapas de Processa

Análise de estratégias de acesso
- Existência de índices
- Estimativas sobre os dados
  (tamanho de tabelas, seletividade, ...)
- Escolha de algoritmos para implementação de operações

ão

ntação interna
algébrica da consulta)

Representação transformada
(árvore otimizada algebricamente)

Definição do Plano de Execução

Plano de Execução
(árvore com indicação de estratégias de acesso)

Gerador de Código

Código de Execução

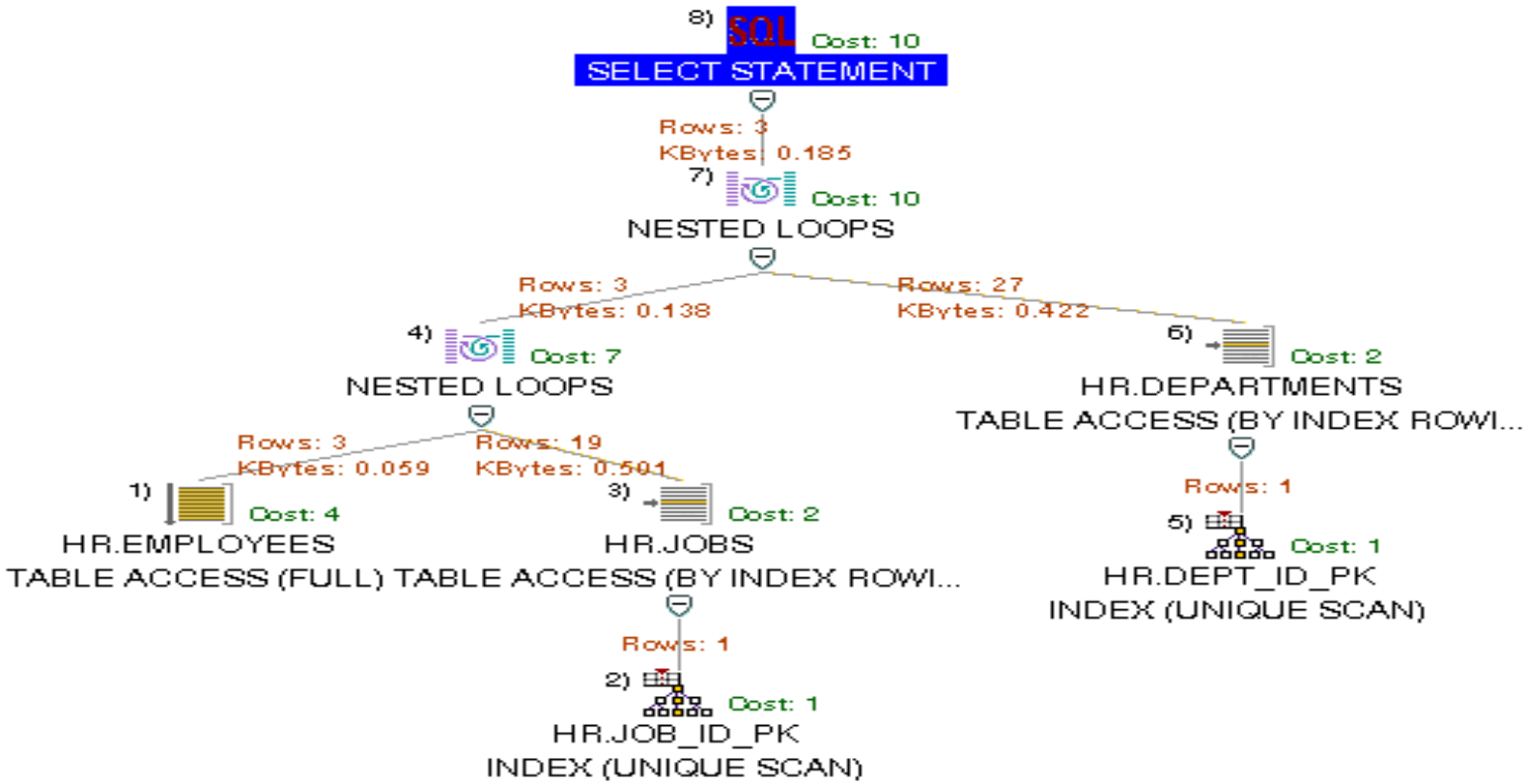
Processador Run-time

Resultado da Consulta

Planos de Execução

- Um comando SQL é executado em muitos passos; a combinação desses passos é chamada um **plano de execução**
- Inclui um método de acesso para cada tabela acessada pelo comando e a indicação do método a ser empregado nas operação binárias, bem como a ordem em que elas são feitas (chamado também de *join order*)
- Métodos de acesso compreendem diferentes algoritmos, como table access, index access, *hash* clusters, nested loops join, sort merge joins, etc.

```
SELECT e.employee_id, j.job_title, e.salary, d.department_name
FROM employees e inner join jobs j on (AND e.job_id = j.job_id )
        inner join departments d on (e.department_id = d.department_id)
WHERE e.employee_id < 103
```



Planos de Execução

- Cada passo retorna um conjunto de linhas – *resultset*
- O *resultset* gerado pelo último passo é retornado ao usuário (resposta da consulta)
- No caso do Oracle, a numeração dos passos é conforme o retornado na representação gráfica dos planos de execução (árvores) ou no comando EXPLAIN PLAN
  - A numeração do EXPLAIN PLAN geralmente não corresponde à ordem da execução dos passos

Algoritmos para a operação de seleção

- Exemplo:
  - (OP1): $\sigma_{SSN='123456789'}$ (EMPLOYEE)
  - (OP2): $\sigma_{DNUMBER>5}$ (DEPARTMENT)
  - (OP3): $\sigma_{DNO=5}$ (EMPLOYEE)
  - (OP4): $\sigma_{DNO=5\ AND\ SALARY>30000\ AND\ SEX='F'}$ (EMPLOYEE)
  - (OP5): $\sigma_{ESSN=123456789\ AND\ PNO=10}$ (WORKS_ON)

Algoritmos para a operação de seleção

S1. Busca linear (força bruta): recupera cada registro do arquivo e testa se seus valores de atributos satisfazem a condição de seleção
   Usualmente recupera os registros *datablock* a *datablock*
S2. Busca binária: se a condição de seleção envolver uma comparação de igualdade em um atributo chave para o qual o arquivo está ordenado
   Mais eficiente que busca linear
   Ex: OP1 $\sigma_{SSN='123456789'}$ (EMPLOYEE)
S3. Utilização de um índice primário: se a condição de seleção envolver uma comparação de igualdade em um atributo chave com um índice primário
   Ex: $_{SSN='123456789'}$ na operação OP1

Algoritmos para a operação de seleção

S4. Utilização de um índice primário para recuperar múltiplos registros: Se a condição de comparação for >, ≥, <, or ≤ em um campo chave com índice primário

Ex: $_{DNUMBER>5}$ na operação OP2

Recupere todos os registros seguintes no arquivo ordenado

S5. Utilização de um índice cluster para recuperar múltiplos registros: Se a condição de seleção envolver uma comparação de igualdade em um atributo não chave com um índice clustering

Ex: $_{DNO=5}$ na OP3

Algoritmos para a operação de seleção

S6. Utilização de um índice secundário em uma comparação de igualdade: recuperar um único registro se o campo de indexação for uma chave ou recuperar múltiplos registros se o campo de indexação não for chave

Também pode ser usado para comparações envolvendo >, >=, < ou <=

Algoritmos para a operação de seleção

- Métodos de busca para seleção complexa (conjuntiva - AND):
  - Ex: OP4

    $\sigma_{\text{DNO=5 AND SALARY>30000 AND SEX='F'}}(\text{EMPLOYEE})$

S7. Seleção conjuntiva utilizando um índice individual:
  - Em caso de existência de mais de método de acesso, será escolhido o método de acesso que tiver maior seletividade
  - Para cada atributo envolvido será avaliada a possibilidade de emprego de um dos métodos de acesso de S2 a S6
    - usa aquela condição para recuperar os registros
    - depois verifica se cada registro recuperado satisfaz as condições simples restantes da conjunção

Algoritmos para a operação de seleção

- Métodos de busca para seleção complexa (conjuntiva - AND):
  - Ex: OP4

    $\sigma_{\text{DNO=5 AND SALARY>30000 AND SEX='F'}}(\text{EMPLOYEE})$

S8. Seleção conjuntiva utilizando um índice composto

- se dois ou mais atributos estiverem envolvidos em condições de igualdade na condição conjuntiva e houver um índice composto para a combinação dos campos então usa-se o índice diretamente
- Seja um índice IDX(A, B, C) sobre as colunas A, B e C de uma tabela, e uma consulta com os seguintes critérios
  - where A=? and B=? and C=? → usa o índice
  - where A=? and B=? → usa o índice
  - where A=? → usa o índice
  - where B=? and C=? → não usa o índice
  - where C=? → não usa o índice

Algoritmos para a operação de seleção

S9. Seleção conjuntiva por meio da interseção de registros:

- se índices secundários (ou outros caminhos de acesso) estiverem disponíveis para mais de um dos campos envolvidos nas condições simples de uma condição conjuntiva
  - cada índice poderá ser usado para recuperar o conjunto de ponteiros de registros que satisfaça a condição individual
  - a interseção destes conjuntos de ponteiros de registros resulta nos ponteiros de registros que satisfazem a condição conjuntiva e que são usados depois para recuperar diretamente aqueles registros
- se apenas algumas das condições possuírem índices secundários, cada registro recuperado será posteriormente testado para determinar se ele satisfaz as condições restantes

Algoritmos para a operação de seleção

S10. Seleção disjuntiva (OR):

- Ex: OP4

  $\sigma_{\text{DNO=5 OR SALARY>30000}}(\text{EMPLOYEE})$
- Se ambas as operações possuírem índices nos campos envolvidos na operação, pode dividir em duas consultas e depois combinar o resultado
- Senão, provavelmente será necessário acesso sequencial à tabela
- Também poderá optar por acesso sequencial se a seletividade do critério da consulta for baixa

Algoritmos para a operação de seleção

- Resumindo…
  - Sempre que uma condição individual especifica a seleção, podemos verificar se existe um caminho de acesso no atributo envolvido naquela condição
    - Se houver: o método correspondente àquele caminho será utilizado
    - Caso contrário: a abordagem da força bruta (S1) será utilizada
  - Em condições de seleção conjuntivas, sempre que mais de um dos atributos envolvidos nas condições tiver um caminho de acesso
    - O otimizador deve escolher o caminho que recupera o menor número de registros de forma mais eficiente (critério de maior seletividade)
      - Por meio de estimativas de custos

Algoritmos para a operação de projeção

- π <attribute list> (R)
  - Se <lista de atributos> contém a chave de R
    - São extraídos os atributos em <lista de atributos> para todas tuplas
  - Caso contrário
    - Tuplas duplicadas devem ser removidas ser for usado o *distinct*
    - Métodos para remover duplicatas
      - Ordenação do resultado
        - Eliminar tuplas idênticas consecutivas
      - *Hashing*
        - Compara pelo *hash*: se for endereçado o mesmo *bucket*, não insere no *resultset*

Algoritmos para a operação de junção

- Implementação do operador JOIN: alto custo!

- Exemplos de operações:
  (OP6): EMPLOYEE $\bowtie_{\text{DNO=DNUMBER}}$ DEPARTMENT
  (OP7): DEPARTMENT $\bowtie_{\text{MGRSSN=SSN}}$ EMPLOYEE

Algoritmos para a operação de junção

J1. Junção de laço aninhado (*nested loops join*)
- Para cada registro t em R (laço externo)
  - Recupere cada registro s em S
  - Teste se os dois registros satisfazem à condição de junção
- Para cada bloco b_r em R (laço externo)
  - Recupere cada bloco b_s em S

J2. Junção de laço único (*single loop join*)
- Usando uma estrutura de acesso para recuperar os registros correspondentes à junção
  - Se existir um índice para um dos dois atributos de junção (por exemplo, b de S)
    - recupere cada registro t em R (um por vez)
      - Use a estrutura de acesso para recuperar os registros em S que satisfaçam a condição de junção

Algoritmos para a operação de junção

J3. Junção ordenação-fusão (*sort merge join*)

- Se os registros de R e S estiverem ordenados pelos valores dos atributos da junção
  - Forma mais eficiente!
  - Varrer ambos arquivos simultaneamente
    - Fazendo a correspondência dos registros que possuem os mesmos valores para os atributos de junção
- Pode ordenar as tabelas envolvidas antes de fazer a junção
- Também eficiente se o resultado final for ordenado pelos atributos de junção

Algoritmos para a operação de junção

J4. Junção hash (*hash join*)

- Os arquivos R e S são particionados em um mesmo espaço de endereçamento *hash* utilizando a mesma função com os atributos A e B como atributo da função *hash*
  - Duas fases neste algoritmo:
    - Fase de separação: passa uma vez pela tabela com o menor número de registros (por exemplo R) e coloca os seus registros no (*bucket*) do arquivo *hash*
    - Fase de sondagem: passa uma vez por S e (1) procura o *bucket* apropriado, e (2) combina aquele registro com todos os registros de R que estão naquele *bucket*

Algoritmos para as operações de conjunto

- Produto cartesiano
  - Um registro para cada combinação de registros das duas tabelas
    - Linhas: n*m
    - Número de blocos lidos b_n x b_m (*nested loop*)
    - Atributos: n+m: pode gravar ainda mais blocos
  - Evitá-la!
    - Substituir por outras operações equivalentes durante a otimização da consulta

Algoritmos para as operações de conjunto

- União, interseção e diferença
  - Ordene as relações segundo o mesmo critério
  - Uma única varredura em cada relação é suficiente para produzir o resultado
- União
  - Varredura e fusão de relações ordenadas simultaneamente
    - Sempre que houver o mesmo registro em ambas as relações, se não for um UNION ALL, apenas um é mantido no *resultset*
- Interseção
  - Manter no *resultset* da interseção apenas os registros que aparecem em ambas as relações

Algoritmos de Acesso do Oracle

- Cluster Access
- Hash Access
- Sample Table Scan
- Join
  - Nested Loop Join
  - Hash Join
  - Sort-Merge Join
  - Cartesian Join
  - Outer Join
- Full Table Scan
- Rowid Scan
- Index Scan
  - Index Unique Scan
  - Index Range Scan
  - Index Skip Scan
  - Full Scan
  - Fast Full Index Scan
  - Index Join
  - Bitmap Index

Algoritmos de Acesso do Oracle

- Full Table Scan
  - Parâmetro DB_FILE_MULTIBLOCK_READ_COUNT (default 128), influencia decisão do otimizador em usar Full Table Scan.
  - Todos (ou a maioria) os *datablocks* da tabela são lidos.
  - Eventualmente, se os *datablocks* forem adjacentes, operações de I/O maiores que um *datablock* podem ser efetuadas, aumentando o *throughtput*, o que é mais vantajoso que utilizar índices em alguns casos
  - Quando é utilizado:
    - Ausência de índice
    - Quando a operação deverá ler a maioria dos *datablocks* da tabela
    - Tabelas pequenas
    - Alto grau de paralelismo

Algoritmos de Acesso do Oracle

- Rowid Scan
  - Maneira mais rápida de localizar um registro único
  - O RowId pode estar especificado na cláusula where (não recomendado) ou pela utilização de índices
  - Formato do RowId: OOOOOOFFFBBBBBBBRRR, onde:
    - OOOOOO – número do objeto de dados (identifica o segmento do BD)
    - FFF – número do *datafile*, relativo ao *tablespace*
    - BBBBBB – bloco de dados que contém a linha
    - RRR – a linha dentro do bloco de dados
  - Quando é utilizado:
    - Após recuperar o RowId por meio de um índice
    - Se o índice contém todas as colunas referidas na cláusula where sequer é necessário acessar a tabela

Algoritmos de Acesso do Oracle

- Index Scan (B+Tree)
  - Quando é utilizado:
    - Quando o índice possui um alto grau de clusterização (os registros da tabela que possuem o mesmo valor da chave de busca e estão no mesmo conjunto de *datablocks*)
    - Quando a busca é por igualdade pela chave primária ou chave alternativa (INDEX UNIQUE SCAN)
    - Quando a busca é por intervalo (INDEX RANGE SCAN)
    - Se a consulta for ordenada (ascendente X descendente) pelos valores da chave de busca, o order by já está pronto
    - Quando a busca é por sub-índice (SKIP SCAN)
    - Quando é necessário ler grande número de registros da tabela mas é necessário recuperar apenas as colunas do índice (FAST FULL SCAN) ou parte das colunas do índice (FULL SCAN)
    - Quando é realizado um hash join em que todas as colunas das tabelas referenciadas fazem parte da chave de busca dos índices (INDEX JOIN)

Algoritmos de Acesso do Oracle

- Cluster Access
  - Tabelas clusterizadas acessadas via índice (B+Tree)
  - Quando é utilizado:
    - Quando é necessário acessar diversos registros com o mesmo valor da chave do cluster
- Hash Access
  - Tabelas clusterizadas acessadas via *hash*
  - Quando é utilizado:
    - Quando é necessário acessar diversos registros com o mesmo *hash value*

Algoritmos de Acesso do Oracle

- Sample Table Scan
  - Retorna uma amostra aleatória de registros, quando usada a cláusula SAMPLE na cláusula FROM de um comando SELECT
  - Exemplo de comando:

```
SELECT *
FROM employees SAMPLE BLOCK (1);
```

Algoritmos de Acesso do Oracle

- Como o otimizador do Oracle escolhe o plano de execução para Junções?
  - Primeiro, determina se na junção tem tabelas cujas linhas correspondem a no máximo 1 linha da saída; se sim, o otimizador coloca essas tabelas por primeiro na ordem de junção
  - Em *outer joins*, tabelas com operação junção externa vêm depois das outras tabelas
  - Otimizador gera um conjunto de planos de execução de acordo com possíveis ordens de junção, métodos e caminhos de acesso disponíveis

Algoritmos de Acesso do Oracle

- Como o otimizador estima o custo para Junções?
  - NESTED LOOPS: custo baseado na leitura de cada linha selecionada da tabela externa, mais as correspondentes linhas da tabela interna, em memória
  - SORT-MERGE JOIN: custo de leitura de todos os dados em memória, e classificá-los
  - HASH JOIN: custo de construir a hash table e de uma das tabelas, e de usar as linhas da outra tabela para acessar a primeira

Algoritmos de Acesso do Oracle

- Nested Loops Join
  - Otimizador determina a tabela de referência para a junção (com PKs, UNIQUEs) e a usa como tabela externa; a outra tabela é designada como tabela interna
  - Para cada linha da tabela externa, o Oracle varre todas as linhas da tabela interna
  - Usado quando a tabela interna pode ficar residente inteira em memória

Algoritmos de Acesso do Oracle

- Hash Join
  - Usado para juntar conjuntos de dados grandes, por igualdade, na chave de junção
  - Otimizador usa a menor das 2 tabelas para construir uma hash table, em memória, com a chave de junção
  - Então, varre a tabela maior, explorando a *hash table* para achar as linhas a serem juntadas
  - Método adequado quando a tabela menor cabe na memória disponível, pois o custo fica limitado a uma leitura simples em cada uma das duas tabelas

Algoritmos de Acesso do Oracle

- Sort-Merge Join
  - Usado para juntar linhas de duas fontes de dados independentes, quando Nested Loops não é aplicável (tabelas grandes)
  - Tipicamente, 2 passos:
    1. ambas as fontes de dados são classificadas pela chave de junção
    2. as listas classificadas são mescladas
  - É indicado quando:
    - As linhas de origem já estão ordenadas
    - A operação de ordenação não necessita ser feita
  - Útil, também, quando as condições de junção são do tipo <, <=, > e >=

Algoritmos de Acesso do Oracle

- Cartesian Join
  - Usado quando uma ou mais tabelas não tem qualquer condição de junção com outras tabelas, dentro do comando SQL
  - Otimizador realiza o produto cartesiano das duas tabelas

Algoritmos de Acesso do Oracle

- Outer Join
  - Retorna todas as linhas que satisfazem a condição de junção, mais as linhas de uma tabela que não tem correspondência a de outra, pela condição de junção
  - Pode ser dos tipos:
    - Nested loops outer join
    - Hash join outer join
    - Sort merge outer join
    - Full outer join

Dúvidas?