CENTRO UNIVERSITÁRIO DE ANÁPOLIS – UNIEVANGÉLICA
CURSO DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

GUILHERME DE OLIVEIRA FERREIRA

APLICANDO TÉCNICAS DE HEURÍSTICA PARA OTIMIZAÇÃO DE CONSULTAS SQL

ANÁPOLIS-GO
2016

GUILHERME DE OLIVEIRA FERREIRA

APLICANDO TÉCNICAS DE HEURÍSTICA PARA OTIMIZAÇÃO DE CONSULTAS SQL

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UNIEVANGÉLICA como requisito parcial à aprovação na disciplina Trabalho de Conclusão de Curso sob orientação da Prof. Esp. Aline Dayany de Lemos.

ANÁPOLIS-GO
2016

GUILHERME DE OLIVEIRA FERREIRA

APLICANDO TÉCNICAS DE HEURÍSTICA PARA OTIMIZAÇÃO DE CONSULTAS SQL

Projeto de Pesquisa apresentado ao Curso Superior de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UNIEVANGÉLICA como requisito parcial à aprovação na disciplina Trabalho de Conclusão de Curso sob orientação da Prof. Esp. Aline Dayany de Lemos.

Banca Examinadora

……..……..……..
Prof. Esp. Aline Dayany de Lemos

……..……..……..
Prof. (…….) ……
Convidado

……..……..……..
Prof. (…….) ……
Convidado

Nota: …….

Anápolis, ……. de …….. de 2016.

Dedico este trabalho a Deus, aos meus pais, meus amigos, meu irmão, minha namorada e todas as pessoas que amo e sempre estiveram ao meu lado me incentivaram a estudar.

AGRADECIMENTO

Agradeço a Deus por ter me dado saúde e força para superar as dificuldades, a minha professora orientadora que teve paciência e me ajudou bastante a concluir este trabalho, e a minha família que sempre esteve ao meu lado.

RESUMO

Este trabalho aborda técnica de Otimização de consultas baseado em heurística, a fim de demonstrar um desempenho eficiente no processamento das consultas no banco de dados, visando a redução no tempo de resposta. A pesquisa busca aplicar a técnica de heurística com auxílio de ferramentas para esta finalidade e analisar os resultados e apresentar o desempenho obtidos durante a execução em um ambiente de teste.

Palavras-chave: dados, otimização, consulta, banco.

ABSTRACT

This paper deals with optimization technique based on heuristic consultation in order to demonstrate an effective performance in the processing of queries in the database. It aims to reduce the response time spending. The research search to apply a technique in a given query with the aid of tools for this purpose, analyzing the results in order to present the performance gains when running in an test application.

Keywords: data, optimization, query, database.

ÍNDICE DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

DDL	*Data Definition Language*
DML	*Data Manipulation Language*
SGBD	Sistema Gerenciador de Banco de Dados
SQL	*Structured Query Language*

SUMÁRIO

1. **INTRODUÇÃO**

O banco de dados é uma ferramenta indispensável na tarefa de armazenar informações de forma rápida e segura, ao longo do tempo teve um papel fundamental em quase todas as áreas em que os computadores estão inseridos, o banco de dados é basicamente uma sistema de "coleção de dados relacionados" que permite que usuários busquem e atualizem essas informações quando solicitado (ELMASRI, 2011).

Segundo Silberschatz (2012) o Sistema Gerenciador de Banco de Dados foi desenvolvido para gerenciar grandes volumes de dados em uma estrutura de armazenamento em que deve garantir a segurança da informação, tal como a confiabilidade, integridade e a disponibilidade dos dados armazenados.

A crescente demanda por sistemas mais eficientes e mais robustos exigem que os bancos de dados tenham a capacidade de gerenciar volumes cada vez maiores de dados e com desempenho elevado, já que o desempenho do SGBD (Sistema Gerenciador de Banco de Dados) é medido a partir de sua eficiência na realização de consultas(CARNEIRO, 2011).

No entanto, o desempenho de uma aplicação está relacionado com a qualidade de software. Para o usuário final, um bom software é aquele que apresenta um tempo de resposta aceitável, mesmo quando submetido a grandes volumes de processamento (MEIER, 2007).

Para garantir o desempenho das consultas o SGBD utiliza duas técnicas diferentes de otimização, a heurística e a estimativa de custo, mas ambas utilizam a álgebra relacional para selecionar um plano de avaliação mais eficiente para o processamento de consulta (SILBERSCHATZ, 2012).

Segundo Silberschatz (2012), a otimização baseado em estimativa de custo seleciona o plano de avaliação utilizando as estimativas que estão armazenados nos catálogos do sistema de banco de dados. No entanto, as estimativas não são muito precisas, já que a maioria dos sistemas não atualizam as estatísticas dos catálogos e a atualização dos catálogos submete a uma sobrecarga muito grande, pois um pequeno erro percentual pode resultar em um plano dispendioso. Agora a otimização baseado em Heurística, possui um custo reduzido porque utilizam regras bem definidas que em todos os casos gera um plano ideal sem muito esforço.

Baseado nesses autores, o trabalho é movido pela questão: Utilizando técnica de heurística é possível melhorar o resultado de uma consulta ou deixá-la mais eficiente?

Essa pergunta será respondida no decorrer deste trabalho baseado nos estudos firmados por alguns estudiosos, levantando a questão de como pode obter resultado significativos com a otimização de consultas SQL utilizando a heurística.

A finalidade dessa pesquisa é aplicar regras de otimização de consulta baseado em heurística para ajustar ou modificar uma consulta SQL garantindo um melhor desempenho.

Para isso será necessário:

- Identificar as regras de heurística existentes para otimizar consulta SQL;
- Elencar as regras de Otimização de Consultas baseadas em regras de heurísticas;
- Medir o desempenho de uma consulta preexistente;
- Fazer o comparativo entre os desempenhos;
- Apresentar resultados após a otimização;

A performance do sistema é um dos requisitos mais lembrados durante o levantamento de requisitos não funcionais, que é definido pelo tempo de resposta para realizar uma determinada tarefa, no entanto, uma das causas dessa demora esta ligada com as consultas SQL que estão sendo executadas no banco de dados. Este problema tornasse cada vez mais claro quando o volume de uma base de dados cresce, com isso, os resultados de uma consulta tendem a crescer e se tornarem cada vez mais lentas (MULLINS, 1998).

Os problemas de performance de um sistema – ligado a um banco de dados – não está relacionado somente ao hardware ou sistema operacional, mas também ao SQL no código da aplicação. Segundo Gervazoni (2016), 80% dos problemas de desempenho dos bancos de dados são causados por códigos SQL mal elaborado.

Hayes (2009) afirma que otimizar um banco de dados pode resultar em vários benefícios tanto para a organização quanto para a aplicação, produzindo um efeito em cadeia: menor consumo de CPU, hardware, licenciamento de software e tempo de resposta da aplicação, consequentemente o software se torna mais confiáveis e mais previsíveis.

Existem duas técnicas para otimizar uma consulta, a baseado em custos e em heurística. A otimização baseado em custos utiliza informações de estatística que são armazenados nos SGBDs, porem como este método é baseado em estatística ele pode gerar um resultado que talvez não seja o melhor. Isto acontece devido o SGBD utilizar informações armazenadas que talvez estejam desatualizadas, diferente do método baseado em heurística que utiliza apenas regras de otimização (SILBERSCHATZ, 2012).

Assim, a motivação inicial deste trabalho foi contribuir na construção de consultas SQL ou na sua otimização para melhorar a performance.

2. **REFERENCIAL TEÓRICO**

A seguir, serão abordados o posicionamento teórico sobre os elementos-chave desta pesquisa. Inicialmente são apresentados os conceitos Banco de Dados Relacionais, Consulta SQL, Processamento de Consultas e, logo após, Heurística para Otimização de Consultas.

2.1.1. **Banco de Dados Relacionais**

Um banco de dados é uma coleção de dados organizado que representa qualquer aspecto do mundo real para resolver algum problema, utilizada por aplicações para armazenar dados, usualmente ditos dados persistentes, ou seja, “os dados diferem em espécie de outros dados mais efêmeros”. Nas organizações os bancos de dados relacionais são atualmente os mais utilizados e possuem a compatibilidade em qualquer tipo de plataforma de hardware ou software (DATE, 2000).

Segundo Date (2000), a introdução do modelo relacional foi o evento mais importante na área de banco de dados, o termo relacional foi baseado nos estudos de lógica e matemática e que acabou sendo aplicado aos bancos de dados, assim, o termo relacional consiste em representar para o usuário os dados em forma de tabelas (relação) compostas por tuplas (linhas) e atributos (colunas), e as operações que o usuário realiza também são gerados novas tabela de tabelas antigas.

Para Silberschatz (2012, p. 25):

> “O modelo relacional é hoje o principal modelo de dados para aplicações comerciais de processamento de dados. Ele conquistou sua posição de destaque devido à sua simplicidade, que facilita o trabalho do programador, comparando com os modelos de dados anteriores, como o modelo de rede ou o modelo hierárquico.”

Os bancos de dados são acessados pelas aplicações através de consultas (ou query) que são enviadas para o SGBD. A consulta é como uma pergunta realizada pela aplicação para o banco de dados e o SGBD é o responsável por intermediar a pergunta, geralmente as consultas resultam na recuperação de alguns dados (ELMASRI, 2011).

O SGBD é um conjunto genérico de software complexo de uso geral que funciona como um facilitador para gerenciar as operações de um banco de dados. Estes softwares permitem automatizar os processos de definição, construção, manipulação e compartilhamento dos dados e

também garante a proteção contra defeitos, falhas de hardware ou software e acessos não autorizados (ELMASRI, 2011).

2.1.2. **Consulta SQL**

Segundo Silberschatz (2012), o SQL é uma linguagem padrão utilizada pelos bancos de dados, apesar de ser chamada de linguagem de consulta ela pode fazer muito mais do que simplesmente realizar uma consulta no banco de dados. “Ela pode definir a estrutura dos dados, modificar dados no banco de dados e especificar restrições de segurança”.

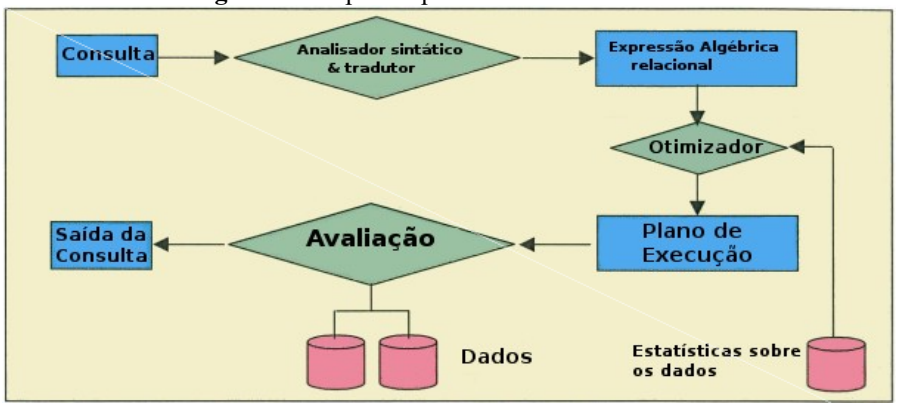
Para Elmasri (2011), é uma linguagem padrão de banco de dados que inclui instruções para definição de dados, consultas e atualizações de dados, dividida em DDL[1] e DML[2]. Além dessas instruções, ela pode definir visões sobre o banco de dados, especificar segurança e autorizações, restringir a integridade dos dados e controlar as transações.

2.1.3. **Processamento de Consultas**

O processamento de consulta refere-se no conjunto de atividades executado pelo SGBD para extração de dados a partir de uma consulta SQL, ou seja, é o processo de traduzir a consulta de alto nível para uma consulta equivalente de baixo nível, de forma a permitir a avaliação e a otimização das consultas (SILBERSCHATZ, 2012).

Na Figura 1 é possível verificar as etapas envolvidas no processamento de uma consulta.

Figura 1 - Etapas no processamento da consulta



Fonte: Silberschatz (2012, p. 337)

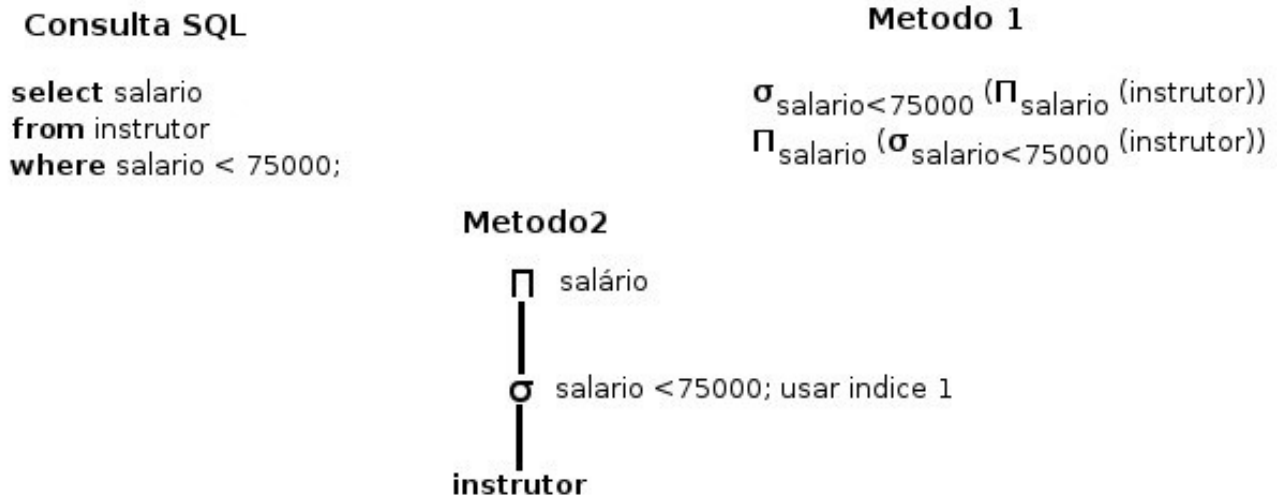
1 Sigla para *Data Definition Language* é uma linguagem para definição de Dados.
2 Sigla para *Data Manipulation Language* é uma Linguagem de Manipulação de Dados.

Antes de realizar qualquer processamento de consulta o banco de dados precisa traduzir a consulta para uma linguagem que são utilizadas pelos bancos de dados relacionais, a linguagem SQL não é uma linguagem ideal para representação interna do sistema de consulta. No entanto, “uma representação interna mais útil é aquela baseada na álgebra relacional estendida” (SILBERSCHATZ, 2012).

Para avaliar uma consulta é preciso expressá-la não só na forma de álgebra relacional, mas também anotá-la com instruções que especificam como avaliar cada operação. Silberschatz (2012, p. 337) afirma que “a representação da álgebra relacional de uma consulta especifica apenas parcialmente como avaliar uma consulta”.

Na Figura 2, ilustra três forma de representar uma consulta, também chama de plano de execução de consulta, ou plano de avaliação de consulta:

Figura 2 - Um plano de avaliação de consulta



Fonte:Silberschatz (2012, p. 338)

2.1.4. **Álgebra Relacional**

A álgebra relacional é uma linguagem formal de consulta procedural que a partir de seus conceitos foi criado a linguagem de consulta SQL. Ela possui um conjunto de operações para extração de dados de um banco de dados, que a partir dessas operações e recursos de entradas podem produzir uma nova relação (consulta) (ELMASRI, 2011).

Date (2000, p. 132) afirma que a álgebra relacional é o principal componente para a manipulação do modelo relacional. Possui um conjunto de operadores que a partir de relações com seus operadores é possível retornar uma outra relação com seus resultados.

A álgebra relacional é um instrumento muito importante para os bancos de dados relacionais, pois ela possui operações que servem para manipular os dados como a linguagem de consulta SQL, utilizadas como base para a implementação do processamento de consultas utilizadas pelos SGBDs na otimização de consultas (ELMASRI, 2011).

A Tabela 1 lista as operações básicas da álgebra relacional:

Tabela 1 - Operações de álgebra relacional

Símbolo	***Operação***	***Sintaxe***	***Tipo***
σ	Seleção/Restrição	$\sigma_{\text{condição}}$ (Relação)	Primitiva
π	Projeção	$\pi_{\text{expressão}}$ (Relação)	Primitiva
U	União	Relação1 Relação2	Primitiva
∩	Intersecção	Relação1 ∩ Relação2	Adicional
-	Diferença de conjuntos	Relação1 - Relação2	Primitiva
x	Produto cartesiano	Relação1 x Relação2	Primitiva
⋈	Junção	Relação1 |x| Relação2	Adicional
÷	Divisão	Relação1 ÷ Relação2	Adicional
ρ	Renomeação	ρ_{nome} (Relação)	Primitiva
←	Atribuição	Variável ← Relação	Adicional

Fonte: disponível em: <http://www.macoratti.net/13/06/sql_arcb.htm>. Acesso em 07 Outubro de 2015

2.1.5. **Regras de Equivalência**

Duas expressões são equivalente quando uma expressão pode substitui uma nova expressão desde que elas produzam os mesmos resultados em qualquer banco de dados. São definidas regras para gerar uma expressão equivalente, essas regras são utilizadas pelos otimizadores para transformar expressões em uma nova expressão para selecionar a mais eficiente (SILBERSCHATZ, 2012).

Date (2000, p. 472) afirma que “dada uma expressão particular a ser transformada, a aplicação de uma regra pode gerar uma expressão passível de transformação de acordo com alguma outra regra”.

2.1.6. **Otimização**

Uma mesma consulta complexa de SQL pode ser processada de diversas formas com diferentes custos de avaliação. Então, o processo de otimização de consulta consiste em selecionar uma desses planos de execução que consiga ser o mais eficiente, pois o sistema não exige que o usuário consiga escrever uma consulta que possa ser processada de forma eficiente. Para realizar a otimização de consulta é necessário que a consulta SQL seja complexa, pois uma consulta simples já possui seu estado de processamento otimizado (SILBERSCHATZ, 2012).

Para Date (2000, p. 470):

> "[…]o otimizador executa uma série de otimizações que "são garantidamente boas", quaisquer que sejam os valores de dados reais e os caminhos de acesso físico que existam no banco de dados armazenados. O fato é que as linguagens relacionais em geral permitem que todas as consultas, exceto as mais simples, sejam expressas de vários modos distintos, pelo menos superficialmente."

Quando uma consulta complexa é executada no banco de dados, ela é quebrada em blocos. Depois de serem quebradas, o processo de otimização e feito individualmente, pois quando o bloco é aninhado o sistema trata a consulta como uma única chamada com sub-rotinas que são executadas uma a uma pela tupla mais externa (DATE, 2000).

2.1.7. **Estimativa de Custo na Otimização**

A técnica de otimização de consulta baseado em custo consiste em estimar sistematicamente o custo de estratégias de execuções diferentes e escolher a que tenha o custo de execução mais baixo. A escolha do plano de execução é baseado no dicionário de dados, estatística, parametros que não são influenciados pela sintaxe da consulta SQL (ELMASRI, 2011).

"O otimizador baseado em custo explora o espaço de todos os planos de avaliação de consulta que são equivalentes a determinada consulta e escolhe aquele com o menor custo estimado". (SILBERSCHATZ, 2012, p. 375)

Caso a estatística esteja ausente ou desatualizada, o otimizador não terá informações vitais necessárias ao processo de otimização da consulta. Por isso, essa técnica é mais indicada para consultas compiladas, onde o processo de otimização é feito na hora da compilação e a estrategia selecionada e armazenada e executado diretamente em tempo de execução (ELMASRI, 2011).

As principais estatísticas coletadas são:

- Estatísticas de tabela: número de blocos, número de blocos vazios, número de chained ou migrated rows, tamanho médio da linha, data da coleta e tamanho da amostra.
- Estatísticas de índices b-tree: altura, número de blocos folha, número de chaves distintas, número médio de blocos folha por chave, número médio de blocos por chave, número de entradas (index entries), clustering factor, data da coleta e tamanho da amostra.
- Estatísticas de Colunas: número de valores distintos, menor valor, maior valor, data da coleta e tamanho da amostra;

2.1.8. **Heurística na Otimização**

As regras de heurística para otimizar as consultas leva uma grande vantagem sobre o método baseado em custo, pois o método baseado em custo precisa selecionar o plano de avaliação utilizando as estimativas que estão armazenados nos catálogos do sistema de banco de dados e esse processo requer muito esforço. Já a heurística trabalha diretamente na álgebra relacional, aplicando regras definidas gerando um plano de execução mais eficiente sem submeter a qualquer tipo de análise de estatística (SILBERSCHATZ, 2012).

A otimização baseado em heurística é também conhecida como otimização algébrica, é uma técnica que aplica as regras de otimização na forma algébrica de uma consulta, podendo a partir de uma forma algébrica gerada pela consulta SQL gerar uma nova forma algébrica a partir das regras de equivalência da álgebra relacional. No entanto, é necessário sempre assegurar que as transformações feitas leve a uma expressão equivalente da consulta (SILBERSCHATZ, 2012).

Elmasri (2011) afirma que as regras de heurística são aplicadas para modificar consultas normalmente na sua forma de árvore ou grafos de consulta para melhorar seu desempenho esperado. Antes de aplicar as regras de otimização a varredura e o analisador de uma consulta SQL gera primeiro uma representação inicial da consulta, após isso o otimizador entra em ação aplicando as regras de heurística.

De acordo com Elmasri e Silberschatz (2011), uma das principais regras de heurísticas são:

- Executar operações de seleção e projeção primeiramente;
- A junção só deve ser realizada depois da seleção e projeção;

- Somente os atributos solicitados para o resultado da consulta e os que realmente são necessários em consultas subsequentes é que devem ser projetados;
- Evitar geração de múltiplas tabelas intermediárias;
- Pesquisar as subexpressões comuns e processá-las somente uma vez;

"As técnicas de otimização de consulta que integram a seleção de heurística e a geração de planos de acesso alternativo foram adotados em diversos sistemas[…]. A busca pelo plano ideal termina quando o orçamento do custo de otimização é ultrapassado e o melhor plano encontrado até esse ponto é retornado." (SILBERSCHATZ, 2012, p. 379)

3. **DESENVOLVIMENTO**

Para a execução desse trabalho o método aplicado será a pesquisa bibliográfica que fundamenta-se em fontes bibliográficas, ou seja, utilização de registro disponível, decorrente de pesquisas anteriores, em documentos impressos, como livros, artigos, teses e etc (SEVERINO, 2007).

Para Lakatos e Marconi (2001, p. 183), a pesquisa bibliográfica,

> "[...] abrange toda bibliografia já tornada pública em relação ao tema estudado, desde publicações avulsas, boletins, jornais, revistas, livros, pesquisas, monografias, teses, materiais cartográficos, etc. [...] e sua finalidade é colocar o pesquisador em contato direto com tudo o que foi escrito, dito ou filmado sobre determinado assunto [...]".

E para demonstrar os resultados da pesquisa será utilizado o método de pesquisa experimental, onde o autor "[…] toma o próprio objetivo em sua concretude como fonte e o coloca em condições técnicas de observação e manipulação experimental[…]. Para tanto, o pesquisador seleciona determinadas variáveis e testa suas relações funcionais, utilizando formas de controle." (SEVERINO, 2007, p. 123).

Para a realização da pesquisa experimental, o autor da pesquisa aplicará a técnica de otimização de consultas baseado em heurística. Para isso, o experimento será executado em um banco de dados de teste chamado *Employees Sample Database*, onde simulará um ambiente real para a execução de alguns refinamentos de consultas SQL para demonstrar as mudanças nos planos de execução dessas consultas e verificar se os resultados obtidos serão realmente alcançados.

3.1. **Definição da base de dados ferramenta utilizada**

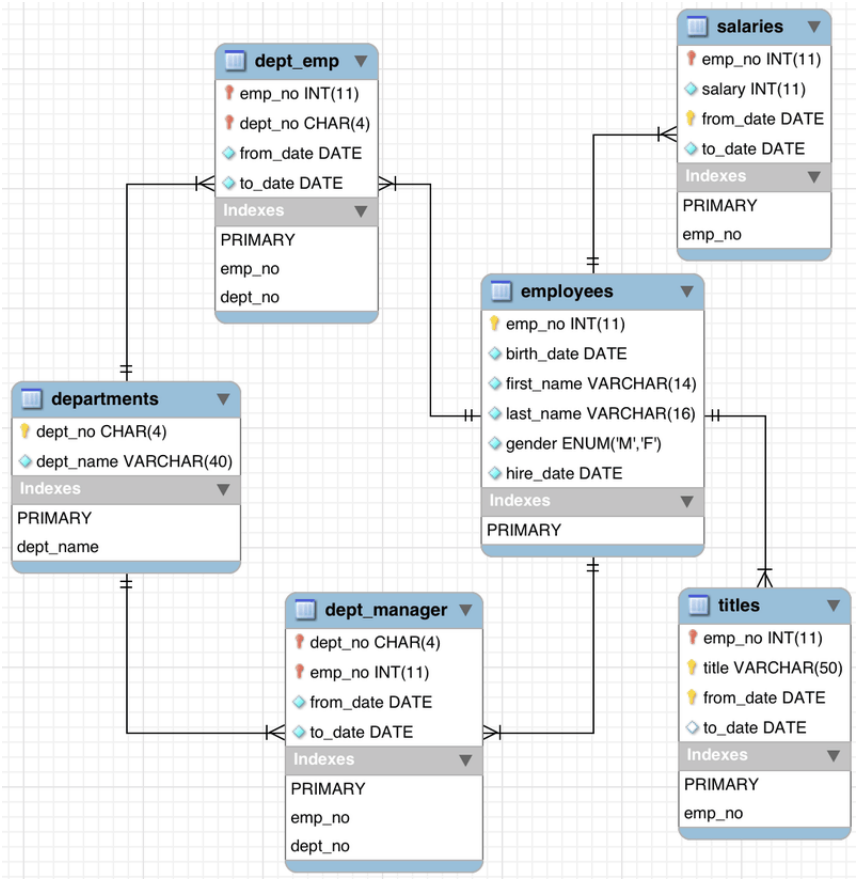
Para aplicar a técnica de otimização será utilizado um banco de dados *Employees Sample Database*[3], é um banco de dado funcional de amostra desenvolvido pela Crews Patrick e Giuseppe Maxia e fornece uma combinação de uma grande base de dados (aproximadamente 160MB) distribuído por tabelas separadas e consiste de 4 milhões de registros no total. Esse banco de dados está disponibilizado no site oficial da Mysql para qualquer pessoa utilizar como teste.

3 https://launchpad.net/test-db/

A ferramenta para diagnóstico de desempenho selecionado foi a Mysql Workbench, é uma ferramenta gráfica para trabalhar com servidores e banco de dados Mysql. Nele é possível criar e gerenciar conexões com servidores de banco de dados, permite configurar os parâmetros de conexão, tem a capacidade de executar consultas SQL usando o built-in Editor SQL, permite criar modelar os esquemas de banco de dados, gerar resultados de desempenho de consulta e etc.

O banco de dados MySql fornece por padrão um otimizador heurística configurado por padrão para construir um plano ideal para a execução das consultas SQL. Para as consultas não terem interferência pelo otimizador foi atribuído no arquivo de configuração do Mysql a variável *optimizer_prune_level* com o valor 0, desativando a execução do otimizador do SGBD e também a *query_cache_type* com o valor 0, desativando o cache das consultas.

Figura 3- Modelo Relacional *Employees Sample Database*.



Fonte : Autor da pesquisa

3.2. **Medindo o desempenho das consultas**

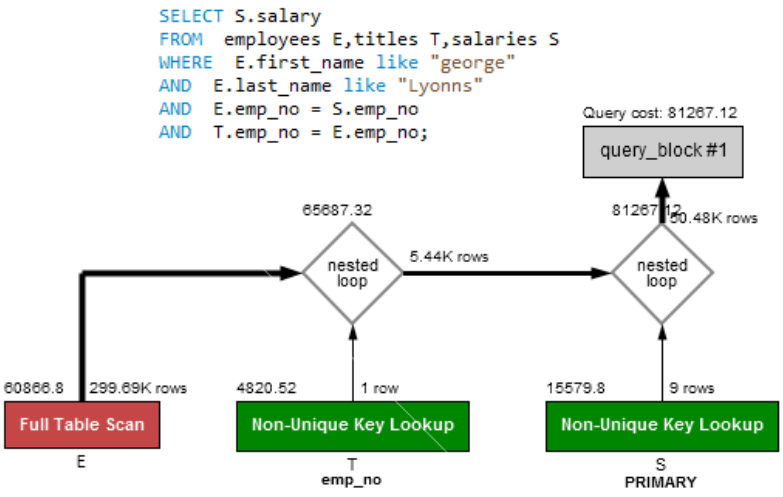
Nesse capítulo demonstra o processo de otimização da consulta em que as regras de otimização serão aplicadas para comprovar ou não a eficiência dessa técnica. Para isso será medido o desempenho de consultas antes e após a aplicação da técnica de heurística para otimizar as consultas.

Utilizando a ferramenta Workbench para auxilio na execuçao das consultas e dos gráfico, foi criado três casos de otimização presente nos topicos abaixo, neles foi criado duas consultas, uma otimizada e a outra não otimizada mas ambas produzem os mesmos resultados.

3.2.1. **Otimização Caso 1**

Na seguinte consulta, que faz uma projeção do atributo "*salary*" pela junção de 3 tabelas "*employees*", "*tittles*" e "*salaries*" nomeada pelas inicias E, T e S fazendo uma seleção pelos atributos "*first_name*" e "*last_name*" da tabela "*employees*". Foi otimizada utilizando 3 regras de heurística que estão listadas abaixo.

Figura 4- Consulta não otimizada Caso1



Fonte : Autor da pesquisa

Para demonstração segue abaixo as modificações que ocorreu na consulta seguindo as regras de heurística para chegar no resultado final:

1. Executar operações de seleção tão cedo quanto possível com o objetivo de diminuindo os tamanhos das relações a serem utilizadas no produto cartesiano:

Figura 5 - Caso 1 aplicando regra 1

```
SELECT S.salary
FROM titles T,salaries S,
    (SELECT * FROM employees WHERE  first_name like "george" AND  last_name like "Lyonns") as E
WHERE  E.emp_no = S.emp_no
    AND  T.emp_no = E.emp_no;
```

Fonte: Autor da pesquisa

2. Substituir operações de produto cartesiano seguidas pelos respectivos critérios de seleção por operações de junção:

Figura 6 - Caso 1 aplicando regra 2

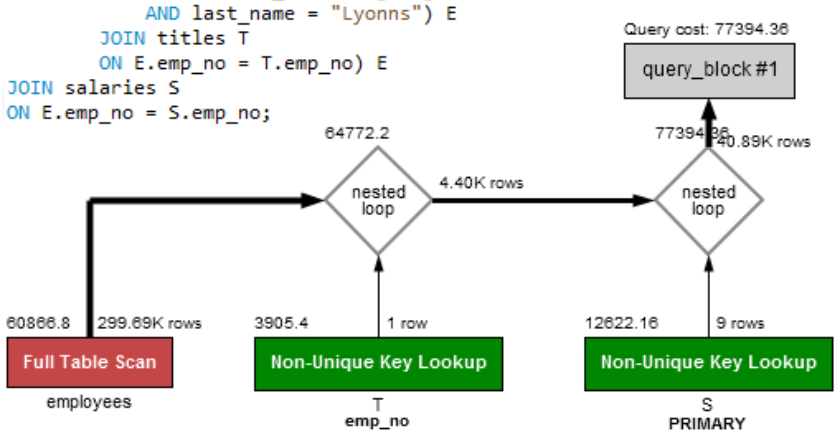
```
SELECT ST.salary
FROM (Select S.* from titles T JOIN salaries S ON T.emp_no = S.emp_no) as ST,
(SELECT * FROM employees WHERE  first_name like "george" AND  last_name like "Lyonns") as E
WHERE  ST.emp_no = E.emp_no
```

Fonte: Autor da pesquisa

3. Executar as operações de projeção tão cedo quanto possível:

Figura 7- Consulta otimizada Caso 1

```
SELECT S.salary
FROM
    (SELECT E.emp_no
        FROM
        (SELECT emp_no
            FROM employees
            WHERE first_name = "george"
            AND last_name = "Lyonns") E
        JOIN titles T
        ON E.emp_no = T.emp_no) E
JOIN salaries S
ON E.emp_no = S.emp_no;
```



Fonte : Autor da pesquisa

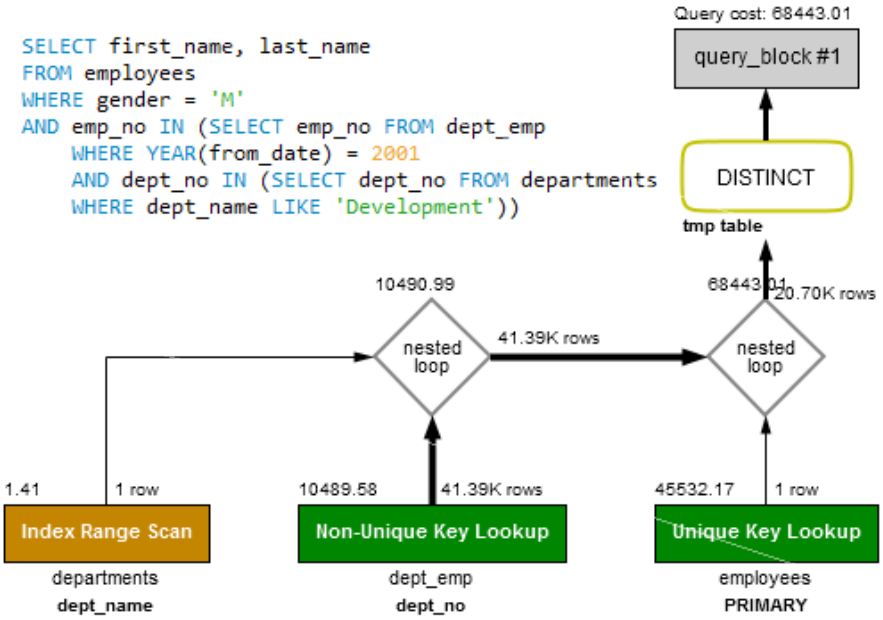
É possível observar que a consulta não otimizada teve um custo da consulta (*Query cost*) de 81267,12 onde o tempo de execução de 0,516 segundos, já a consulta otimizada obteve o custo da consulta reduzido no valor de 77394,36 e o tempo de execução também reduzido para 0,454 segundos.

Esse ganho de performance ocorreu porque foi utilizado uma das principais regras que tem o propósito de aplicar as operações de seleção e projeção antes de aplicar a junção ou outras operações binárias reduzindo as consultas de entrada para os "*joins"*, pois o tamanho da consulta resultante de uma operação binária como as junções normalmente é uma operação multiplicativa dos tamanhos das consultas de entrada, com isso, faz com que reduza os trabalhos dos *loop's* (operação realizada pelos *join's*) que foi realizado entre as tabelas "*employees*", "*emp_no*" e "salaries".

3.2.2. **Otimização Caso 2**

Na seguinte consulta, que faz uma projeção dos atributos "*first_name*" e "*last_name*" da tabela "*employees*" fazendo uma seleção pelos atributos "*gender*" da tabela "*employees*", "*from_date*" da tabela "*dept_emp*" e "*dept_name*" da tabela "*departments*".

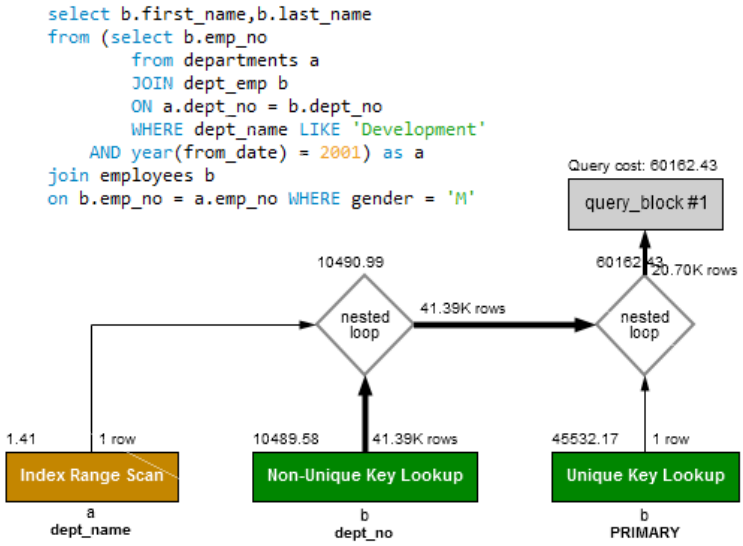
Figura 8 - Consulta não otimizada Caso2



Fonte: Autor da pesquisa

Na otimização seguinte foi possível aplicar apenas uma das regras de heurística, que utiliza a junção só após a seleção e projeção substituindo as subconsultas por *joins* e restringindo os dados nos níveis mais internos.

Figura 9 - Consulta otimizada Caso 2



Fonte : Autor da pesquisa

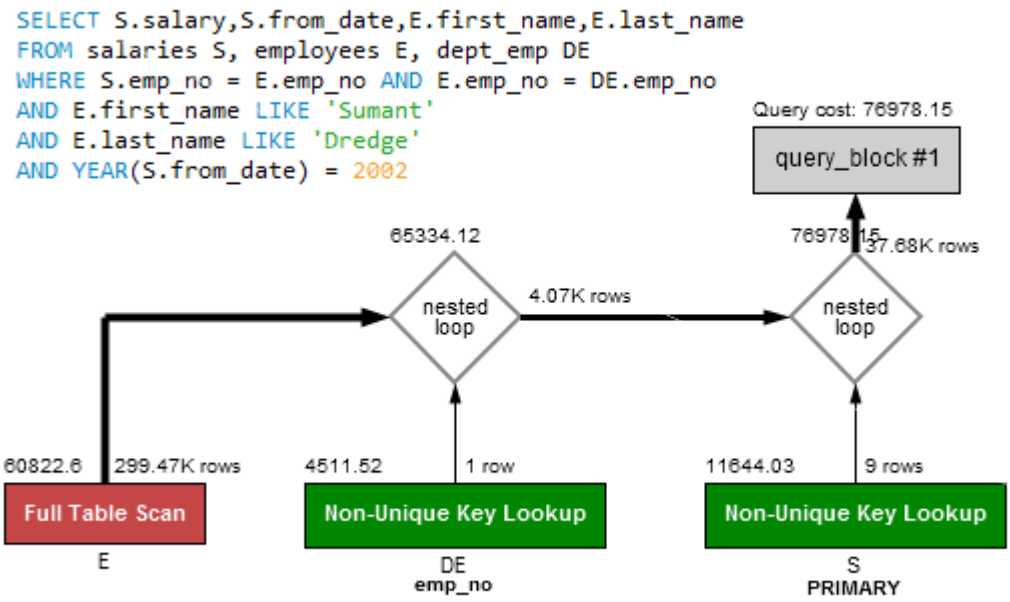
No caso acima a consulta não otimizada teve um custo da consulta (*Query cost*) de 68443,01 onde o tempo de execução de 2,844 segundos, já a consulta otimizada obteve o custo da consulta reduzido no valor de 60162,43 e o tempo de execução também reduzido para 0,688 segundos. A consulta otimizada conseguiu reduzir o custo porque ela foi modificada para aplicar as junções só após a seleção e projeção.

Nesse caso de otimizaçao, foi possivel realizar somente uma modificação pois como pode ver na figura 8 a consulta já possui a seleção e projeções nos níveis mais internos da subconsulta, no entanto foi possivel aplicar somente a regra que substitui as subconsultas por joins utilizando as restrições nos níveis mais internos, que busca reduzir o custo dos "*join",* operação que já foi discutido no Caso 1.

3.2.3. **Otimização Caso 3**

Na seguinte consulta, que faz uma projeção dos atributos "*salary*", "*from_date*", "*first_name"* e "last_name" das tabelas "*salaries*" e "employees", pela seleção dos atributos "first_name", "last_name" e "*from_date*". Foi otimizada utilizando 2 regras de heurística que estão listadas abaixo.

Figura 10- Consulta não otimizada Caso 3



Fonte: Autor da pesquisa

Para demonstração segue abaixo as modificações que ocorreu na consulta seguindo as regras de heurística para chegar no resultado final:

1. A junção só deve ser realizada depois da seleção e projeção:

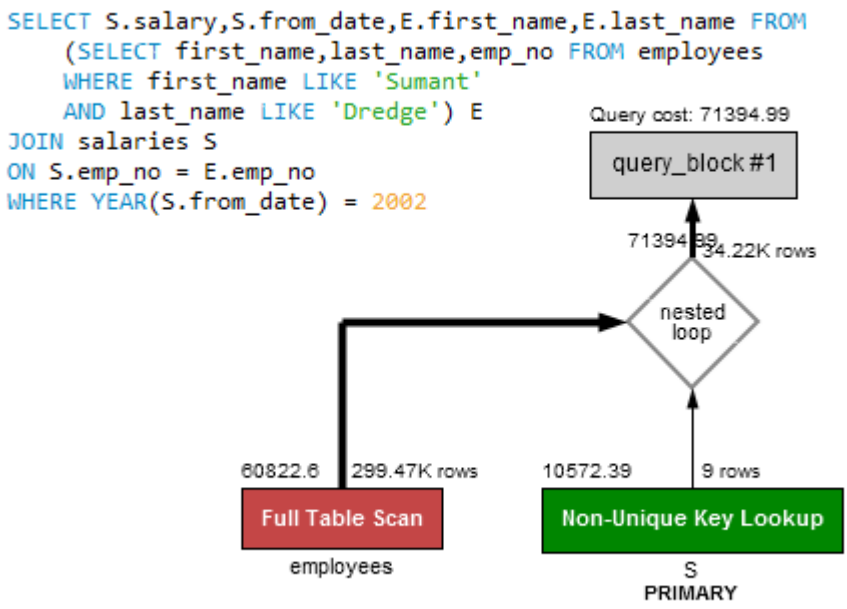
Figura 11 - Caso 1 aplicando regra 1

```
6  SELECT S.salary,S.from_date,E.first_name,E.last_name FROM
7      (SELECT * FROM employees WHERE first_name LIKE 'Sumant'
8      AND last_name LIKE 'Dredge') E
9  JOIN salaries S JOIN dept_emp DE
10 ON S.emp_no = E.emp_no AND E.emp_no = DE.emp_no
11 WHERE YEAR(S.from_date) = 2002
```

Fonte: Autor da pesquisa

2. Somente os atributos solicitados para o resultado da consulta e os que realmente são necessários em consultas subsequentes é que devem ser projetados:

Figura 12 - Consulta otimizada Caso 3



Fonte: Autor da pesquisa

Nas consultas acima a consulta não otimizada teve um custo da consulta (*Query cost*) de 76978,15 onde o tempo de execução de 1,722 segundos, já a consulta otimizada obteve o custo da consulta reduzido no valor de 71394,99 e o tempo de execução também reduzido para 0,625 segundos.

Nesse caso de otimização não foi possivel utilizar a regra que executar as operações de projeção tão cedo quanto possível, pois como pode ver na Figura 11 os atributos das tabelas já estão projetando os atributos no nível mais interno da consulta, agora um diferencial dessa otimização comparando com o Caso 1 é que nele foi eliminado a junção da tabela "*dept_emp*" isso reduziu o trabalho das junções que era realizado desnecessáriamente na consulta da Figura 11.

4. **CONSIDERAÇÕES FINAIS**

Por meio do estudo dessa técnica, foi possível demonstrar que através da otimização baseado em heurística é possível reduzir o custo de uma consulta e consequentemente o seu tempo de execução, assim, melhorando o seu desempenho. O resultado obtido com essa pesquisa, conforme a tabela abaixo.

Tabela 2 - Resultados obtidos

Descrição	**Consulta não Otimizada**	**Consulta Otimizada**
Caso 1		
Custo da Consulta	81267,12	77394,36
Tempo	0,516 s	0,454 s
Resultado da Consulta	40,89 KB	40,89KB
Caso 2		
Custo da Consulta	68443,01	60162,43
Tempo	2,844 s	0,688 s
Resultado da Consulta	20,70 KB	20,70 KB
Caso 3		
Custo da Consulta	76978,15	71394,99
Tempo	1,72 s	0,625 s
Resultado da Consulta	37,68 KB	34,22 KB

Fonte: Autor da pesquisa

O objetivo geral dessa pesquisa foi utilizando as técnicas de heurística é possível melhorar o resultado de uma consulta. A conclusão geral é de que essa técnica pode sim melhorar o resultado de um consunta, quando aplicadas de maneira correta, pois existem casos que não é possível aplicar todas as regras de heuristica, por isso é necessário que o desenvolvedor faça uma análise na consulta para decidir quais regras que sejam possiveis de ser aplicadas. É importante resaltar que alguns bancos de dados mais modernos já possuem em sua execução o seu proprio algoritmo de otimização de consulta, que se mostram bem eficientes.

5. **REFERÊNCIAS BIBLIOGRÁFICAS**

CARNEIRO, Alessandro. **Técnicas de Otimização de Bancos de Dados: Um estudo Comparativo: MySQL e PostegreSQL**. Rio Grande: (FURG), 2011.

DATE, C. J., 1941 - **Introdução a sistemas de bancos de dados** / C. J. Date; tradução [da 7. ed. americana] Vandenberg Dantas de Souza, Publicare Consultoria e Serviços, Rio de Janeiro: Campus, 2000.

ELMASRI, Ramez. **Sistemas de banco de dados** / Ramez Elmasri e Shamkant B. Navathe; tradução Daniel Vieira; revisão técnica Enzo Seraphim e Thatyana de Faria Piola Seraphim. 6. ed. São Paulo: Pearson Addison Wesley, 2011.

GERVAZONI, Thiago. SQL Server: Melhorando a performance através das estatísticas. Disponível em: <http://www.linhadecodigo.com.br/Artigo.aspx?id=704> Acesso em: 16 de junho de 2016.

HAYES, Scott. **O Efeito Cadeia da Otimização de Desempenho e Custo.** Disponível em: <http://www.ibm.com/developerworks/br/data/library/dmmag/DMMag_2009_Issue2/DataManager>. Acesso em: 02 Outubro de 2015.

LAKATOS, E. M.; MARCONI, M. A. **Fundamentos de metodologia científica.** 4.ed. São Paulo: Atlas, 2001.

MACORATTI, José Carlos. **SQL - Álgebra Relacional - Operações Fundamentais - Conceitos básicos.** Disponível em: <http://www.macoratti.net/13/06/sql_arcb.htm>. Acesso em: 07 Outubro de 2015.

MEIER, J., Farre, C., Bansode, P., Barber, S., and Rea, D. (2007). **Performance testing guidance for web applications: patterns & practices:** Microsoft Press, Redmond,USA. Disponível em: <https://msdn.microsoft.com/en-us/library/bb924366.aspx>. Acesso em: 02 Outubro de 2015.

MULLINS, Craig S.. **The Most Important Thing is Performance,** 1998. Disponível em: <http://www.craigsmullins.com/cnr_perf.htm>. Acesso em: 02 Outubro de 2015.

SILBERSCHATZ, Abraham. **Sistema de banco de dados** / Avi Silberschatz, Henry F. Korth, S. Sudarshan; tradução Daniel Vieira. Rio de Janeiro: Elsevier, 2012.

SEVERINO, Antônio Joaquim, **Metodologia do trabalho científico.** 23. ed. São Paulo: Cortez, 2007.