UNIVERSIDADE FEDERAL DE ALAGOAS - UFAL
CAMPUS DE ARAPIRACA
CIÊNCIA DA COMPUTAÇÃO - BACHARELADO

DANIEL DOS SANTOS SILVA

IMPACTO DA OTIMIZAÇÃO DE CONSULTAS SQL EM SGBDS RELACIONAIS

ARAPIRACA
2020

Daniel dos Santos Silva

Impacto da otimização de consultas SQL em SGBDS relacionais

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal de Alagoas, UFAL, Campus Arapiraca.

Orientador: Prof. Dr. Patrick Henrique da Silva Brito

Arapiraca

2020

Universidade Federal de Alagoas – UFAL
Biblioteca Campus Arapiraca - BCA
Bibliotecário Responsável: Nestor Antonio Alves Junior
CRB - 4 / 1557

S586i Silva, Daniel dos Santos
Impacto na otimização de consultas SQL em SGBDS relacionais / Daniel dos Santos Silva. – Arapiraca, 2020.

41 f. : il.

Trabalho de Conclusão de Curso (Bacharelado Ciência da Computação) - Universidade Federal de Alagoas, *Campus* Arapiraca, Arapiraca, 2020.

Orientador: Prof. Dr. Patrick Henrique da Silva Brito.

Bibliografia: p. 41.

1. Otimização de linguagem – Consulta estruturada. 2. SQL. 3. Sistema de Gerenciamento de Banco de Dados (SGBD). 4. Banco de dados. 5. Consulta SQL. I. Brito, Patrick Henrique da Silva. II. Título.

CDU 004

Daniel dos Santos Silva

IMPACTO NA OTIMIZAÇÃO DE CONSULTAS SQL EM SGBDS RELACIONAIS

Trabalho de Conclusão de Curso depositado como requisito parcial para obtenção do título de bacharel em Ciência da Computação da Universidade Federal de Alagoas – UFAL, *Campus de Arapiraca*.

Data de Aprovação: 28/02/2020

Prof. Dr. Patrick H. S. Brito
Universidade Federal de Alagoas
Campus Arapiraca
Orientador

Prof. Me.Alexandre Paes dos Santos
Universidade Federal de Alagoas
Campus Arapiraca
Examinador

Prof. Dr. Alexandre de Andrade Barbosa
Universidade Federal de Alagoas
Campus Arapiraca
Examinador

AGRADECIMENTOS

Agradeço primeiramente a Deus, pois sem a Sua graça não seria capaz de alcançar a conclusão deste trabalho. Toda a minha gratidão ao corpo docente e, em especial, ao meu orientadore por todo incentivo e apoio tão importantes. Sem sua ajuda e ensino nada disso seria possível. À minha família, amigos e minha noiva, por serem meu pilar, estarem ao meu lado e me fazer acreditar que tinha a força e as ferramentas necessárias para finalizar este trabalho.

E, por fim, agradeço todas as pessoas que, de alguma forma, foram essenciais para que alcançasse este objetivo com o qual sempre sonhei.

RESUMO

Bancos de dados tornam-se cada dia mais importantes no cotidiano da sociedade atual. Durante as atividades diárias, pessoas deparam-se interações com algum banco de dados, embora na maioria das vezes sem ao menos perceberem tal feito. Pode-se afirmar que os bancos de dados são amplamente utilizados em diversas aplicações presentes no dia-a-dia, tais como: transações bancárias, transações com cartão de crédito, reservas de linhas aéreas e informações de horários, reservas de hotel, vendas, revendedores on-line, dentre outras aplicações. Devido ao grande volume de dados que estes precisam gerenciar e a complexidade de suas aplicações é fundamental que as operações de consultas por exemplo sejam realizadas de forma eficiente. Logo, esse trabalho aborda técnica de Otimização de consultas SQL baseado em heurística com objetivo de validar uma melhor performance no processamento de consultas no banco de dados tendo em vista a redução do tempo de resposta.

Palavras-chave: Otimização de linguagem de consulta estruturada (SQL). Sistema de Gerenciamento de Banco de Dados (SGBD). Banco de dados. Consulta SQL.

ABSTRACT

Databases become more and more important in the daily life of today's society. During daily activities, people interact with some databases, although most of the time without realizing what is done. It can be registered that databases are widely used in several applications present in daily life, such as: bank transfers, credit card records, airline reservations and timetable information, hotel reservations, sales, online resellers -line, among other applications. Accept the large volume of data that they can manage and the complexity of your applications. It is essential that query operations, for example, are carried out efficiently. Therefore, this work addresses techniques for optimizing SQL queries based on heuristics in order to validate a better performance in query processing in the database in order to reduce response time.

Keywords: Structured Query Language (SQL) Optimizer. Database Management System (DBMS). Database. Query SQL

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

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

SUMÁRIO

1 INTRODUÇÃO

Atualmente banco de dados tornou-se peça fundamental nas atividades diárias das pessoas. Diante disso as organizações estão recorrendo cada vez mais à Tecnologia da Informação e Comunicação (TIC), para gerenciar e armazenar os dados que serão processados e manipulados diariamente. Com isso, o volume de dados vem aumentando exponencialmente, testemunhando a necessidade de otimização do Sistemas Gerenciador de Banco de Dados (SGBD). Sendo assim, é imprescindível que, as ox'perações realizadas no banco de dados sejam eficazes e processadas no tempo esperado para a tomada de decisão. Para garantir o desempenho e a eficiência das informações acessadas no banco, a otimização deve ser desejada pelos Administradores de Banco de Dados (DBA), em especial a efetivação de consultas que gerem informações rápidas e precisas.

Segundo Elmasri e Navathe, "Um banco de dados é projetado, construído e povoado por dados, atendendo a uma proposta específica. Possui um grupo de usuários definido e algumas aplicações preconcebidas, de acordo com o interesse desse grupo de usuários." (2011, p.4)

A complexidade e o tamanho de um banco de dados são variáveis. Diante desse fenômeno sua criação e manipulação implicou a necessidade de ser computadorizados, através de um grupo de aplicações como o chamado Sistema Gerenciador de Banco de Dados.

Segundo Silberschatz, Korth e Sudarshan (2012) o Sistema Gerenciador de Banco de Dados foi projetado para gerenciar grandes volumes de informações. O gerenciamento de dados envolve definir estruturas para armazenamento de informação e fornecer mecanismo para a manipulação de informações. Além disso, o sistema de banco de dados precisa garantir a segurança das informações armazenadas, tais como a confiabilidade, integridade e a disponibilidade dos dados armazenados.

Com a crescente concentração de volume de dados nas grandes organização se faz necessário à busca 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 et al., 2011).

Para assegurar o desempenho das consultas o SGBD faz o uso de duas técnicas diferentes de otimização, a heurística e a estimativa de custo, na qual ambas utilizam a álgebra

relacional para selecionar um plano de avaliação mais eficiente para o processamento de consulta (SILBERSCHATZ; KORTG; SURDARSHAN 2012).

Diante do exposto, é de grande relevância realizar o processo de otimização, uma vez que a seleção de uma boa estratégia faz toda diferença e, levando-se em conta o tempo de avaliação para a escolha do melhor plano de avaliação, a diferença de custo pode chegar a várias ordens de grandeza

E para efeito de demonstração dos resultados da pesquisa, será utilizado o método de pesquisa experimental, na será aplicada tecnicas de otimização de consultas baseado em heuristicas. Para tanto, o experimento terá como base de teste o banco de dados chamdado *Employees Sample Database*, na qual simulará um ambiente real para execução de alguns aperfeiçoamento de consultas SQL a fim de demonstrar as mudanças nos planos de execução dessas consultas e averiguar se os resultados obtidos é o esperado.

O restante do trabalho está estruturado com segue. O capitulo 2 apresenta a fundamentação téorica onde é mostrado uma breve introdução a Sistema Gerenciadores de Banco de Dados, assim como é feita uma breve explanação sobre a algebra relacional bem como sobre otimização de consultas apartit de regras de equivalências baseads em heuristica e custo.

No capitulo 3, apresenta definição da base de dados e ambiente de execução, bem como as técnica de otimização de consultar SQL baseado em heuristicas.

O capitulo 4, apresenta as consideraçõs finais tamadas como base os resultados alcançados através da aplicação da técnica de otimização de consultas.

2 FUNDAMENTAÇÃO TEÓRICA

Logo após, serão apresentado toda fundamentação teórica sobre os elementos-chave desta pesquisa. A princípio são apresentados os Conceitos Gerais Banco de Dados, Banco de Dados Relacionais, Structured Query Language, Processamento de Consultas, Algebra Relacional, Regras de Equivalência, Otimização Baseada em Custo e Otimização Baseado em Heuristicas.

2.1 CONCEITOS GERAIS SOBRE BANCO DE DADOS

Para (Elmasri; Navathe, 2011), um banco de dados é uma coleção de dados relacionados. Os dados são fatos que podem gravados e que possuem um significado implícito, uma das propriedades implícitas é:

- Um banco de dados representa alguns aspectos do mundo real, sendo chamado, às vezes, de **minimundo** ou de unidade verso de discurso(UoD).
- Um banco de dados é projetado, construído e povoado por dados, atendendo a uma proposta específica. Possui um grupo de usuários definido e algumas aplicações preconcebidas, de acordo com o interesse desse grupo de usuários.

Segundo Silberschatz, Korth e Sudarshan (2012, p.5), “Um modelo de dados é uma coleção de ferramentas conceituais para descrever dados, relações de dados, semântica de dados e restrições de consistência.”. Um modelo de dados permite uma melhor descrição de um projeto de um banco de dados nos seguintes níveis físico, lógico e de visão, nos quais são divididos em quatro categorias distintas de modelos diferentes:

- Modelo relacional: é um modelo baseado em tabelas, utilizado para representar os dados e as relações entre eles. Cada tabela possui diversas colunas e, cada coluna possui um nome único. É um modelo baseado em registro, pois é estruturado em registro de formato fixo de vários tipos. O modelo de dados relacional é hoje é um dos mais usados em uma grande maioria dos sistemas de banco de dados.

- Modelo de entidade/relacionamento: baseado em uma percepção do mundo real que consiste em uma coleção de objetos básicos, chamados *entidades*, e as *relações* entre esses objetos. É um modelo que facilita o mapeamento para o esquema conceitual.

- Modelo de dados baseado em objeto: uma extensão do modelo de dados relacional com noções de encapsulamento, ou seja, incluindo a orientação a objetos e tipos de coleção que possuem relações, conjuntos, multiconjuntos e arrays.

- Modelo de dados semi-estruturado: possui características opostas dos modelos de dados mencionados anteriormente, pois os itens de dados individuais do mesmo tipo podem ter diferentes conjuntos de atributos. Um exemplo desse modelo é a Extensible Markup Language (XML).

2.1.1 Banco de Dados Relacionais

Um banco de dados é uma coleção de dados ditos persistente, ou seja, "os dados diferem em espécie de outros dados mais efêmeros, utilizada por aplicações de uma determinada empresa para o armazenamento das informações e tomadas de decisões". 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, 200).

Segundo Date (2000), a inserção do modelo relacional na área de banco de dados foi um fato muito importante, o modelo relacional foi baseado em certos aspectos da matemática (principalmente na teoria dos conjuntos e na lógica de predicado), até então gerenciamento de bancos, era deficiente demais em tais qualidades, 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 como as consultas (*query)* o retorno dessa consulta é uma nova tabela gerada das tabelas anteriores.

Para Silberschatz, Korth e Sudarshan (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 através de aplicações através de consultas que são aviadas ao SGBD. A manipulação inclui algumas funções como pesquisas em banco de dados para recuperar um dado especifico, atualização do banco de dados para refletir as mudanças

no minimundo e gerar relatórios dos dados para que possam ser usados pela empresa desses dados para tomadas de decisões. O SGBD é responsável por mediar a consulta, e essas retornam a recuperação do dado solicitado ao banco de dados.

SGDB é uma colecão de programas que permite ao usúario criar e manter um banco de dados. Ele facilita os processos de definicão, onde ocorre a especificação do tipo de dados e das restrições dos dados que serão armazenados, de construcão, etapa de armazenamento dos dados. É um software que possui recursos capazes de manipular as informações do banco de dados e interagir com o usuário.

Os SGBDs oferecem as seguintes interfaces para que seus usuários possam manipular toda informação contida no banco de dados.

Figura 1 - Representação simplificada de um sistema Gerenciador de Banco de Dados.

Fonte: Disponível em: http://e-reality-database.blogspot.com/2008/09/sistema-gerenciador-de-banco- de- dados.html, Acesso em: 15 dez. 2019

A Figura 1 acima mostra de forma clara um dos principais benefícios do SGBD, que é abstrair a parte interna do banco de dados separando os interesses e facilitando a vida de programadores e DBAs.

2.1.2 Structured Query Language – SQL

Segundo Date (2000), a SQL é uma linguagem padrão para se lidar com banco de dados relacionais, e é aceita em quase todos os produtos existente no mercado, a SQL foi desenvolvida no início da década de 1970, a IBM desenvolveu a versão original da SQL, a qual foi chamada de Sequel. Após anos de evolução, passou a ser chamada de SQL (Structure Query Language), tornando-se a linguagem padrão de banco de dados relacional.

De acordo com as definições de Silberschatz, Korth e Sudarshan (2012, p.51),

> Embora a SQL seja referenciada como "linguagem de consulta", ela pode fazer muito mais do que simplesmente consultar um banco de dados, pois ela consegue definir a estrutura dos dados, modificar dados no banco de dados e especificar restrições de segurança.

Para Elmasri e Navathe (2011), é uma linguagem padrão de banco de dados que possui comandos para definição de dados, consultas e atualizações de dados, dividida em DDL e DML, (Data Definition Language) é uma linguagem para definição de Dados e (Data Manipulation Language) é uma Linguagem de Manipulação de Dados respectivamente. Além dessas instruções, ela pode definir visões sobre o banco de dados, afim de especificar segurança e autorizações, restringir a integridade dos dados e controlar as transações.

Para Ricarte (2019), SQL é uma linguagem padronizada para a definição e manipulação de bancos de dados relacionais. Tipicamente, um SGBD oferece um interpretador SQL que permite isolar a aplicação dos detalhes de armazenamento dos dados

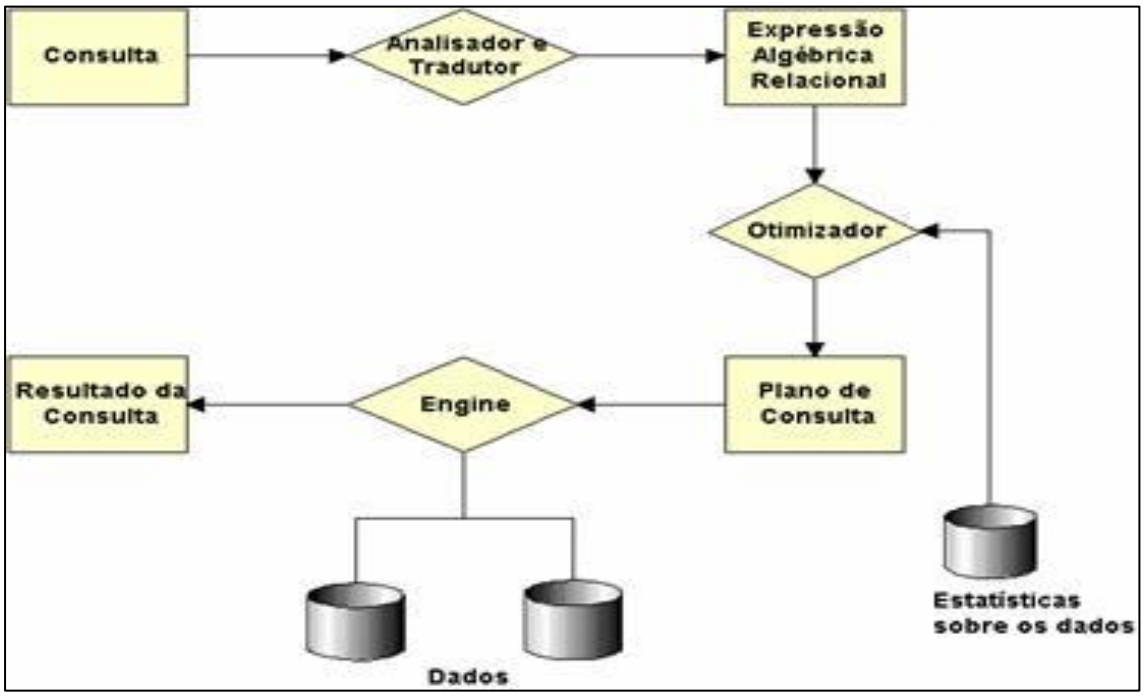
O usuário precisa definir em qual ordem as operações devem ser executadas. No entanto, a linguagem SQL proporciona uma interface declarativa de alto nível, então, o usuário somente especifica qual será o resultado, deixando com SGBD a otimização de como, de fato, executar a consulta

2.1.3 Processamento de Consultas

O processamento de uma *Query* trata do conjunto de atividades executado pelo SGBD envolvidas na extração de informações em um banco de dados a partir de uma consulta SQL. Essas atividades incluem tradução de consultas em linguagem de banco de dados de alto nível para expressões que pode ser usada no nível físico do sistema, como também uma série de transformações de otimização da consulta e avaliação real da mesma (SILBERSCHATZ; KORTG; SURDARSHAN, 2012).

Na Figura 2 é possível verificar as etapas envolvidas no processamento de uma consulta:

Figura 2 - Etapas no processamento da consulta



Fonte: Disponível em: https://www.devmedia.com.br/artigo-da-sql-magazine-28-duvidas-frequentes-sobre-bancos-de-dados/6258. Acesso em: 19 dez. 2019.

Antes que qualquer processamento de consulta possa começar, o sistema precisa traduzir a consulta para uma forma linguagem utilizável pelo banco de dados relacionais, a linguagem SQL não é uma linguagem ideal para representação interna do sistema de consulta. Não obstante, “uma representação interna mais útil é aquela baseada na álgebra relacional estendida”.

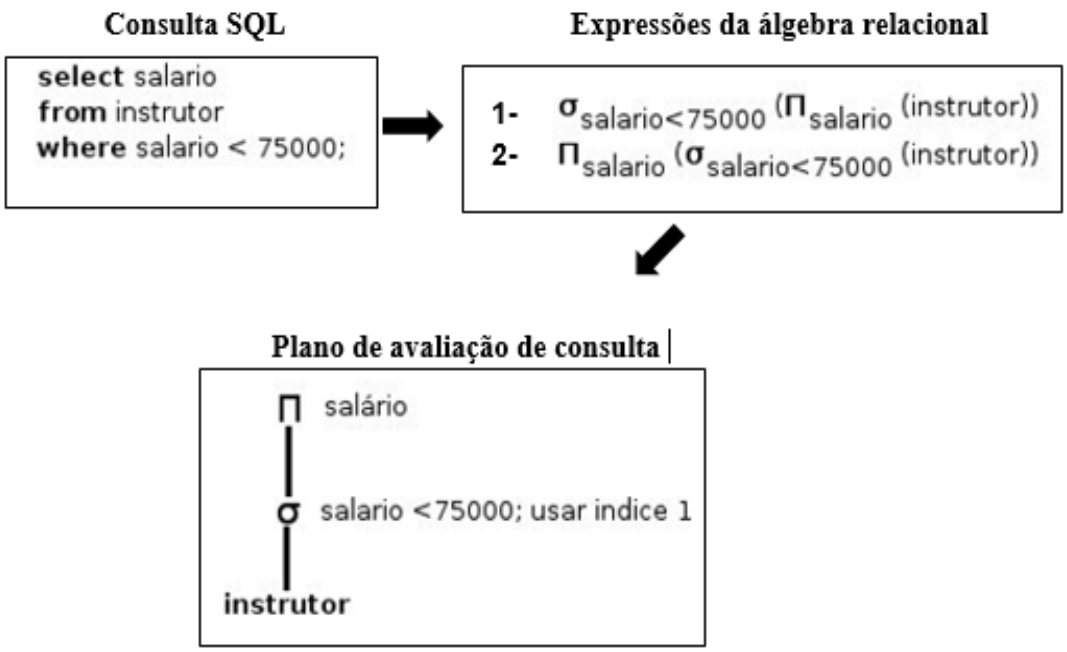
O processo é semelhante ao analisador *paser* de um compilador na qual é verificada a sintaxe da consulta do usuário etc. O sistema constrói uma representação de árvore de análise de consulta, que pode traduzir para uma expressão da álgebra relacional.

Para especificar totalmente como avaliar uma consulta, precisamos não apenas oferecer a expressão da álgebra relacional como único método de avaliar, mas também anotá-la com instruções especificando como avaliar cada operação, como por exemplo indicar o algoritmo a ser usado. Silberschatz , Korth e Sudarshan (2012, p. 358) afirma que “a representação da álgebra relacional de uma consulta especifica apenas parcialmente como avaliar uma consulta”.

Existem várias maneiras de avaliar as expressões da álgebra relacional. Os diferentes planos de avaliação como também são chamados, para determinada consulta podem ter diferentes custos.

Na Figura 3 ilustra três formas de representar uma consulta, chamada de plano de execução de consulta, ou plano de avaliação de consulta:

Figura 3 – Um plano de avaliação de consulta.



Fonte: Silberschatz, Korth e Sudarshan (2012).

2.1.4 Algebra Relacional

A Álgebra Relacional é uma linguagem de consulta formal, porém procedimental, — ou seja, o usuário dá as instruções ao sistema para que o mesmo realize uma seqüência de operações na base de dados para calcular o resultado desejado — , 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; NAVATHE 2011).

Segundo as afirmações de Date (2003, p.150) a algebra relacional é a peça fundamental para maninpulação do modelo realcional, pois possui um coleção de operadores, que apartir da relação entre eles é possivel retornar uma outra relação com seus resultados.

A álgebra relacional é muito importante por diversas razões. Primeira, porque provê um fundamento formal para operações. Segunda porque é usada como uma base para implementar e otimizar as consultas em sistema gerenciadores de banco de dados relacional (SGBDRs) e alguns de seus conceitos são incorporados na linguagem de consulta-padrão SQL para os SGBDRs.

O Quadro 1 a seguir temos um resumo das operações da álgebra relacional:

Quadro 1 - Operações de álgebra relacional

Símbolo	*Operação*	*Sintaxe*	*Tipo*
σ	Seleção / Restrição	$\sigma_{\text{condição}}$ (Relação)	Primitiva
π	Projeção	$\pi_{\text{expressões}}$ (Relação)	Primitiva
∪	União	Relação1 ∪ Relação2	Primitiva
∩	Intersecção	Relação1 ∩ Relação2	Adicional
-	Diferença de conjuntos	Relação1 - Relação2	Primitiva
x	Produto cartesiano	Relação1 x Relação2	Primitiva
|x|	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: 25 dez. 2019.

As principais operações ditas como fundamentais da álgebra relacional são:

- **Seleção:** É uma operação que para um conjunto inicial fornecido como argumento, produz um subconjunto estruturalmente idêntico, mas apenas com os elementos do conjunto original que atendem a uma determinada condição (chamada de predicado);
- **Projeção:** Produz um conjunto onde há um elemento para cada elemento do conjunto de entrada, sendo que a estrutura dos membros do conjunto resultante é definida nos argumentos da operação;
- **União:** Retorna a união das tuplas de duas relações R1 e R2 com eliminação automática de duplicatas;

- **Diferença de conjuntos:** É uma operação que requer como operandos duas relações união-compatíveis, ou seja, estruturalmente idênticas. O resultado é uma relação que possui todas as linhas que existem na primeira relação e não existem na segunda.
- **Produto Cartesiano:** O resultado do produto cartesiano de duas relações é uma terceira relação contendo todas as combinações possíveis entre os elementos das relações originais;
- **Renomeação:** Esta operação unária primitiva redefine o nome de uma tabela em um determinado contexto. É útil para auto relacionamentos, onde precisamos fazer a junção de uma tabela com ela mesma, e nesse caso cada versão da tabela precisa receber um nome diferente da outra;

2.1.5 Regras de Equivalência

Antes de tudo, uma consulta SQL é traduzida para uma expressão equivalente na álgebra relacional, representada por uma estrutura de dados *árvore de consulta*, que após é otimizada. Uma regra de equivalência diz que expressões de duas formas são equivalentes. É através delas que o otimizador transforma expressões em outras logicamente equivalentes.

Silberschatz, Korth e Sudarshan (2012) conceitua que duas expressões são equivalente quando uma expressão pode substitui uma nova expressão desde que elas resultem as mesmas saida em qualquer banco de dados. Para isso 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.

De acordo com Elmasri e Navathe (2011, p.356),

> As consultas SQL são decompostas em blocos de consultas, que formam as unidades básicas que podem ser traduzidas em operadores algébricos e otimizadas. Um bloco de consulta contém uma única expressão SELECT FROM-WHERE, bem como cláusulas GROUP BY e HAVING, se elas forem partes do bloco.

Segundo as afirmações de Date (2003, p. 472) 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

Segundo a definição de Silberschatz, Korth e Sudarshan otimização da consulta “É o processo de selecionar o plano de avaliação de consulta mais eficiente dentre as muitas

estratégias normalmente possíveis para o processamento de determinada consulta, especialmente se esta for complexa, pois uma consulta simples já possui seu estado de processamento otimizado". (2012, p.383).

Os usuários não têm obrigação de saber escrever a consulta para que a mesma seja processada de forma eficiente, mas o sistema deve construir um plano de avaliação para que o custo da avaliação da consulta seja reduzido.

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 caminh os 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 indentificada no processo de execução no banco de dados, ela é quebrada em blocos. Após 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).

O otimizador de consulta tem como tarefa gerar um plano de avaliação de consulta que terá como base uma expressão em álgebra relacional e a transformará numa expressão com menor custo. A geração de planos de avaliação envolve três etapas:

1. Definir expressões que sejam logicamente equivalentes a expressão dada;
2. Fazer uma estimativa do custo do plano de avaliação;
3. Realizar de forma alternativa anotações das expressões resultantes para gerar planos de avaliação de consulta alternativos.

Primeiramente, as etapas 1 e 3 são realizadas, sendo as expressões geradas e anotadas aos poucos. E em segundo plano a etapa 2 é feita com a coleta de informações, estatísticas sobre as relações para garantir uma boa estimativa do custo de um plano.

Além do processso da busca realizada pelo sistema de uma maneira mais eficiente para executar uma consulta, através de uma expressão equivalente na álgebra relacional, outros aspectos também são observados tais como: algoritmos para execução de operações e também índices específicos que serão utilizados.

Existem diversas maneiras de uma consulta ser expressa, com diferentes custos de avaliação. É bom ressaltar que, a ordem das tabelas é irrelevante, desde que seja equivalente o conjunto de tabelas.

2.1.7 Otimização Baseada em Custo

A técnica de otimização de consulta baseado em custo tem como objetivo estimar e comparar os custos da execução de uma consulta usando diferentes estratégias de execuções e escolher uma com a menor estimativa de custo. A escolha do plano de execução é baseado no dicionário de dados, estatística, parametros que não são influenciados pela sintaxe da consulta SQL (ELMASRI; NAVATHE 2011).

"O otimizador baseado em custo gera uma série de planos de avaliação de consulta a partir de determinada consulta, usando as regras de equivalencias, e escolhe a única com o menor custo (SILBERSCHATZ; KORTH. SUDARSHAN, 2012).

De acordo com Elmasri e Navathe (20011, p.377),

> Para estimar custos de várias estratégias de execução, devemos manter atualizadas quaisquer informações que sejam necessárias para as funções de custo. Essas informações podem ser armazenadas no catálogo do SGBD, onde são acessadas pelo otimizador de consulta.

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

A otimização baseada no custo, busca soluções para um problema e também uma solução que minimize uma função de custo, porém essas funções são apenas estimativas, pois o otimizador pode escolher uma estratégia de solução que não seja a estratégia ótima.

2.1.8 Utilização de Heurísticas na Otimização de Consultas

As regras de heurística para otimizar as consultas tem 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, pois o número de planos de avaliação diferentes pode ser muito grande, tendo como desvantagem o prório custa da otimização. 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; KORTH. SUDARSHAN, 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; KORTH. SUDARSHAN, 2012).

De acordo com Elmasri e Navathe (2011, p. 368), definem que:

> Uma árvore de consulta é uma estrutura de dados de árvore que corresponde a uma expressão da álgebra relacional. Ela representa as relações de entrada de uma consulta como nós folhas da árvore e representa as operações da álgebra relacional como nós internos.

Na Figura 4 temos como exemplo a consulta SQL “Q1:

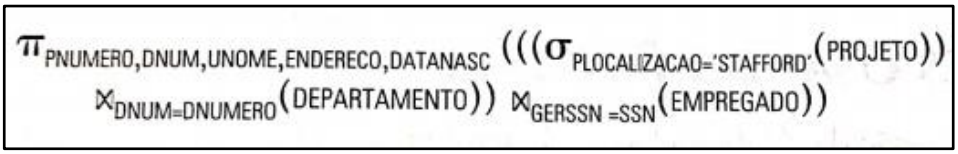
Figura 4 – Consulta SQL: “Q1”

```
SELECT P.PNUMERO, P.DNUM, E.UNOME, E.ENDERECO, E.DATANASC
FROM   PROJETO AS P, DEPARTAMENTO AS D, EMPREGADO AS E
WHERE  P.DNUM=D.DNUMERO AND D.GERSSN=E.SSN AND
       P.PLOCALIZACAO='Stafford';
```

Fonte: Elmasri e Navathe (2011, p. 368).

A Figura 5 corresponde a expressão da álgebra relacional da consulta SQL Figura 4:

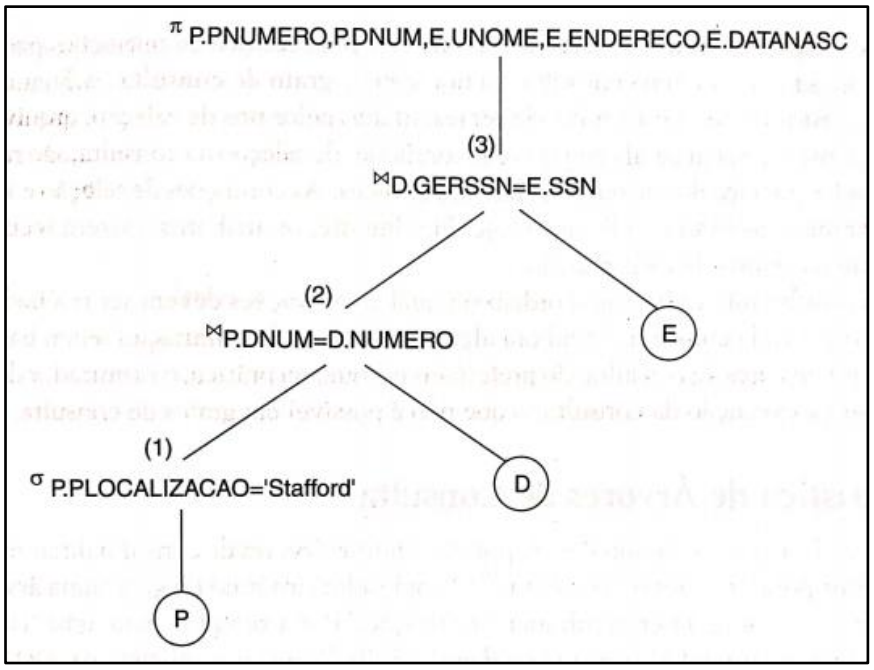
Figura 5 – Expressão álgera relacional.



Fonte: Elmasri e Navathe (2011, p. 368).

A Figura 6 é uma árvore de consulta correspondende â expressão da álgebra relacional da Figura 5:

Figura 6 – Árvore de consulta



Fonte: Elmasri e Navathe (2011, p. 368).

A Figura 6 mostra uma árvore de consulta para a consulta SQL “Q1” da Figura 4, que diz o seguinte: para cada projeto localizado em ‘Stafford’, recupere o número do projeto, o número do departamento responsável e o último nome, o endereço e a data de nascimento do gerente do departamento.

Segundo as definições de Elmasri e Navathe (2011) atestam que as regras de heurística são aplicadas para modificar consultas na sua forma de árvore ou grafos de consulta a fim de melhorar seu desempenho esperado. Antes de ser feito o processo 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 no de otimização aplicando as regras de heurística.

Para Elmasri e Navathe(2011) e Silberschatz, North e Sudarshan (2012), 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 constituem a seleção de heurística e a geração de planos de avaliação de acesso alternativo foram adotados em diversos sistemas[…]. A busca pelo plano ideal, —— pois nem sempre a busca é pela ótima — termina quando o orçamento do custo de otimização é ultrapassado e o melhor plano encontrado até esse ponto é retornado." (SILBERSCHATZ; KORTH. SUDARSHAN, 2012, p. 379).

O processo de otimação será executado passo a passo aplicando as princinpais regras de equivalência entre as expressões da álgebra relacional, transformando as a árvore de consulta inicial em uma árvore de consulta final otimizada que seja mais eficiente.(GRAVA, MARIA, 2010)

Para Elmasri e Navathe (2011) as principais regras de transformação de expressões da algebra relacional são:

1. Cascata de σ: Uma condição de seleção conjuntiva poder ser quebrada em uma cascata (ou seja, uma sequência) de operações σ individuais: **σ**c1 and c2 and ... and cn (R) ≡ **σ**c1 (**σ**c2 (… (**σ**cn (R)) ...))

2. Comutatividade de **σ:** A operação **σ** é comutativa: **σ**c1 (**σ**c2 (R)) ≡ **σ**c2 (**σ**c1 (R))

3. Cascata de **π:** Em uma sequência de operações **π**, todas, com exceção da última, podem ser ignoradas.

4. Comutatividade de **σ** e **π** Se a condição de seleção c envolve somente os atributos A1, ..., An na lista de projeção, as duas operações podem ser comutadas. **π**A1A2, …, An (**σ**c (R)) ≡ **σ**c (**π**A1A2, …, An (R))

5. Comutatividade de |X| (e ×): A operação |X| é comutativa assim como a operação × também é comutativa.

R |X|c S ≡ S |X|c R

$R \times S \equiv S \times R$

6. Comutando σ com |X| (ou ×): Se todos os atributos na condição de seleção c envolvem apenas os atributos de uma das relações participantes da junção - digamos R -, as duas operações podem ser comutadas como segue:

σc (R |X| S) ≡ (**σ**c (R)) |X| S

7. Comutatividade de π e |X| com (ou ×): Suponha que a lista de projeção seja L = {A1, ..., An, B1, ..., Bm}, onde A1, ..., An são atributos da relação R e B1, ..., Bm são atributos da relação S. Se a condição junção c envolve apenas atributos que estão em L, então as duas operações podem ser comutadas como segue:

πL (R |X|c S) ≡ (**π**A1, …, An (R)) |X| c (**π**B1, …, Bm (S))

Se a condição de junção c envolve atributos adicionais que não estejam em L, estes devem ser adicionados à lista de projeção e uma projeção adicional é necessária. Por exemplo, se os atributos An+1, ..., An+k de R e Bm+1, ..., Bm+p de S estão envolvidos na condição de junção c, mas não estão na lista de projeção L, então π e |X| (ou ×) podem ser comutadas da seguinte forma:

πL (R |X|c S) ≡ πL ((πA1, …, An, An+1, …, An+k (R)) |X|c (πB1, …, Bm, Bm+1, …, Bm+p (S)))

8. Associatividade de , ×, ∪ e ∩ Essas quatro operações são individualmente associativas, ou seja, se θ for aplicado para alguma destas operações, então:

(R θ S) θ T ≡ R θ (S θ T)

9. Convertendo uma sequência (σ, ×) em Se a condição c de uma σ que segue uma × corresponde à condição de junção, então (σ, ×) pode ser convertida em |X| da seguinte forma:

σc (R × S) ≡ (R |X|c S)

3 DESENVOLVIMENTO

Buscando agilidade nas consultas desses dados, com base nesses autores, o intuito dessa pesquisa é aplicar regras de otimização de consulta baseado em heurística para ajustar ou modificar uma consulta SQL, sendo de notável grande importância a escolha da solução mais adequada para agilizar no processo de consultas, garantindo um melhor desempenho.

Para isso deve-se:

- Identificar as regras de heurística existentes para otimizar consulta SQL;
- Listar as regras de Otimização de Consultas baseadas em regras de heurísticas;
- Medir o desempenho de uma consulta preexistente;
- Identificar e aplicar as regras de equivalências possíveis;
- Comparar os desempenhos entre as duas consultas;
- Apresentar resultados após aplicação das heurísticas;.

3.1 DEFINIÇÃO DA BASE DE DADOS E AMBIENTE DE EXECUÇÃO

A base de dados ultilzada como teste será o banco de dados de amostra Employees disponibilizado no site oficial da Mysql, foi desenvolvido por Patrick Crews e Giuseppe Maxia e fornece uma combinação de uma grande base de dados (aproximadamente 160 MB) distribuídos em seis tabelas separadas e consistindo em 4 milhões de registros no total.

Para a realização dos experimentos foi utilizado um computador com as seguintes especificações e aplicações:

- Notebook Asus S550C
- Processador: Intel(R) Core(TM) i5 – 3317U CPU 1.70GHz.
- Memória RAM: 8GB
- Armazenamento: 500GB
- Sistema Operacional Windows 10 Version 1809 64bits.
- Mysql Workbench CE Version 18.0.12

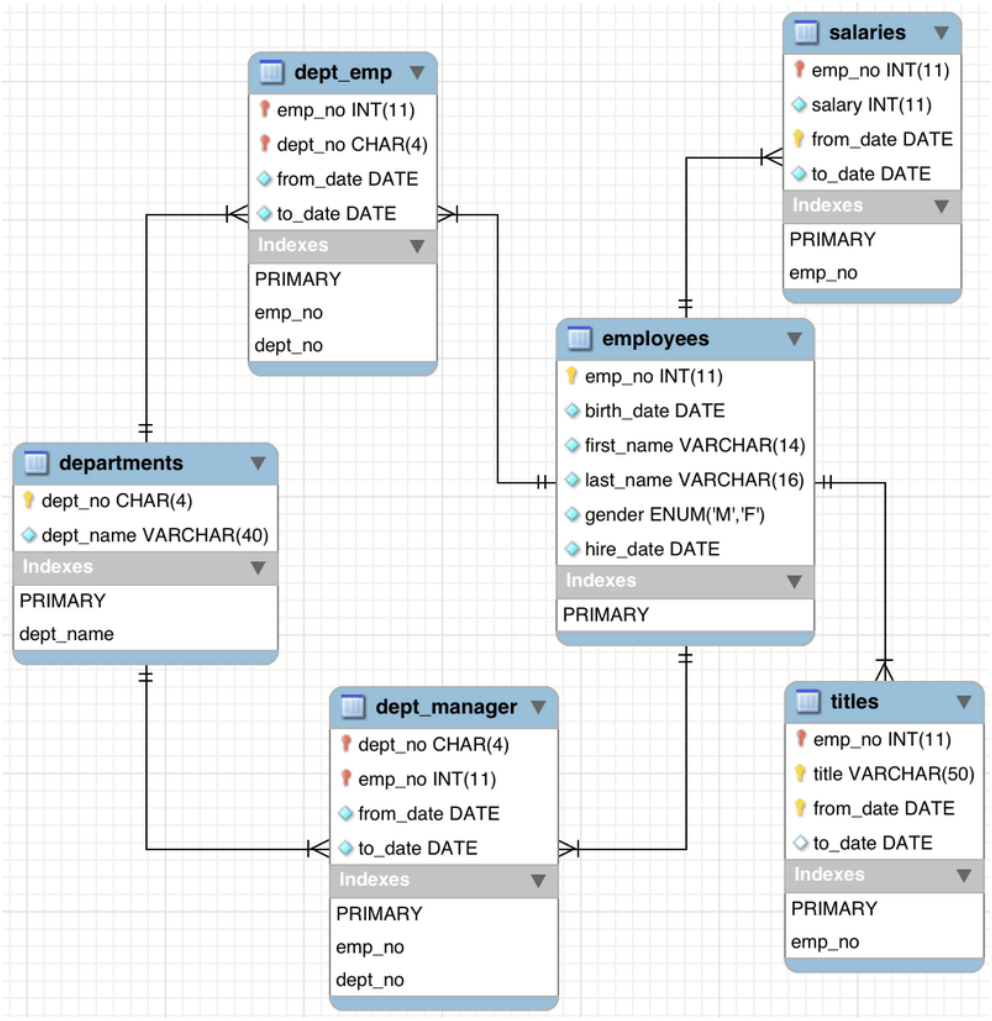
O Mysql Workbench foi a ferramenta escolhida para realizar a analise de desempenho das consultas SQL , é uma ferramenta gráfica para trabalhar com servidores e banco de dados Mysql. Nele é possivel criar e gerenciar conexões com servidores de banco de dados, possui

a capacidade de executar consultas SQL usando o Editor SQL embutido, assim como gerar resultados de desempenho de consulta .

O MySql tem por padrão um otimizador que utiliza heurística configurado para construir um plano ideal para a execução das consultas SQL. Para não houver interferência nas consultas pelo otimizador, foi atribuído no arquivo de configuração do Mysql (my.ini) a variável *optimizer_prune_level* = 0, dessa forma é desativada a execução do otimizador do SGBD e a *query_cache_type* = 0, desativando o cache das consultas, por padrão essas variavéis recebem valor 1.

Na Figura 7 mostra o Modelo Relacional do banco de dados Employees Sample Database.

Figura 7 – Modelo Relacional do Employees Sample Database



Fonte: Disponível em: https://launchpad.net/test-db/.Acessado em: 21 dez. 2019.

3.2 ANALISE DE DESEMPENHO DAS CONSULTAS

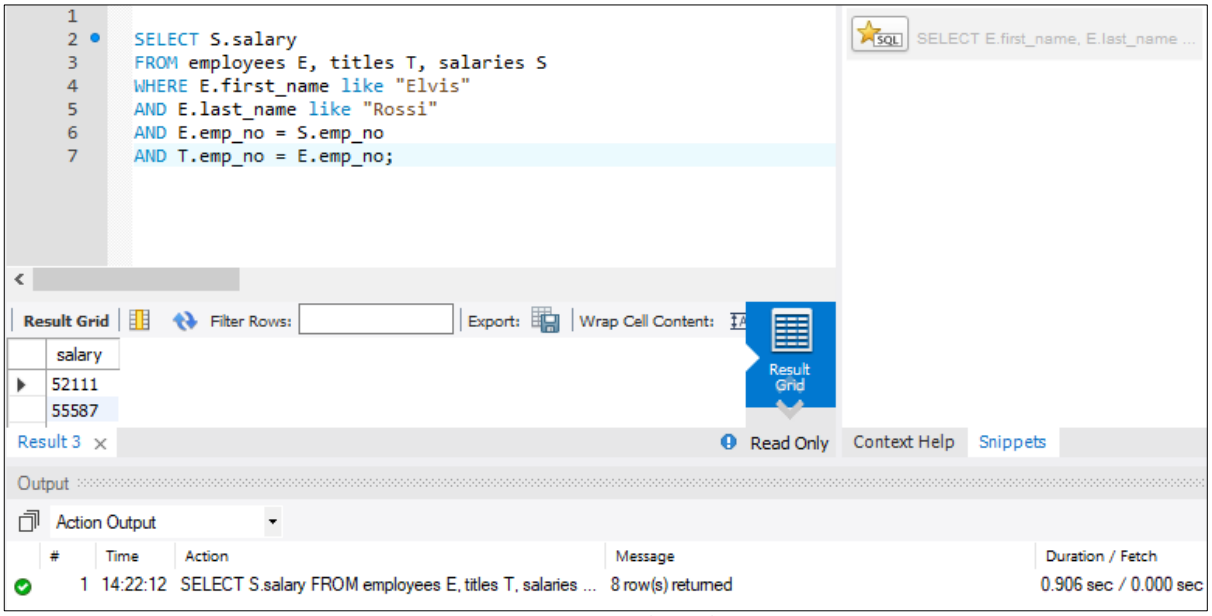
Nesta seção será apresentada o processo de otimização da consulta na qual é aplicada as regras de otimização para validar ou não a eficiência dessa técnica. Sendo assim, será medido o desempenho de consultas antes e após a aplicação da técnica de heurística. Com o auxilio da ferramenta Mysql Workbench na execuçao das consultas, foi criado três situações de otimização listadas nos topicos abaixo, nelas foi criado duas consultas, uma otimizada e a outra não otimizada mas ambas com resultados equivalentes.

As etapas do processo de otimização de consultas, partirá de uma consulta SQL ao passo que essa consulta será convertidada em expressões da algebra relacional equivalentes representada em forma de árvore de consulta, aplicando as regras de heuristicas.

3.2.1 Otimização Situação 01

Na atual consulta é feito uma projeção do atributo “salary” tendo como alvo a junção de 3 relação “employees”, “tittles” e “salaries” nomeada pelas inicias E, T e S faz uma seleção de filtro pelos atribistos “*first_name = Elvis”* e “*last_name = Rossi*” da relação “employees”.

Figura 8 – Consulta não otimizada Situação 01



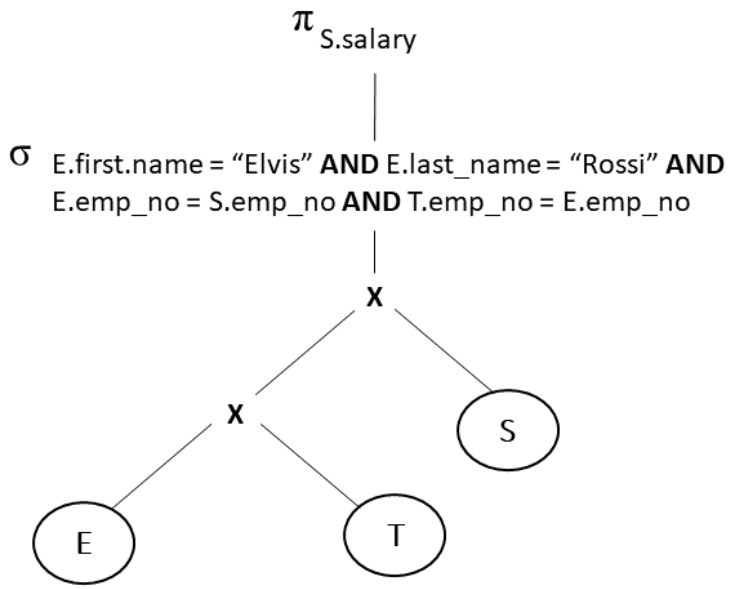
Fonte: O autor (2020).

Para efeito de demonstração segue abaixo as modificações expressa em forma de *arvore de consulta* equivalentes, aplicado na atual consulta norteada atravé das regras de

heurística, ao aplicar todas regras cabivés essa consulta será converta em SQL e executada no MySQL Workbench afim de obter o resultado esperado.

Na Figura 8 mostra a arvore de consulta inicial (canônica) correspondente à expressão da álgebra relacional para a consulta SQL Situação 01.

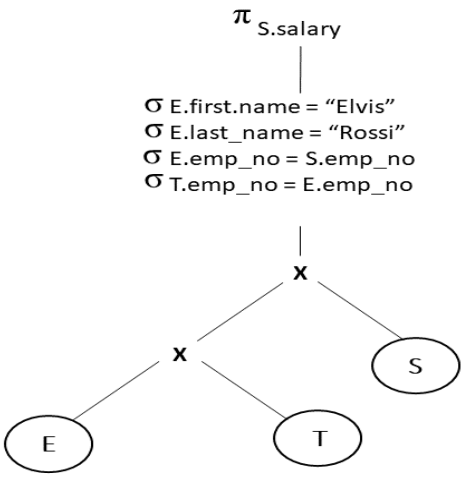
Figura 9 – Arvore inicial (canônica)



Fonte: O autor (2020).

1. Separar as seleções conjuntivas em um cascateamento de seleções, com objetivo de facilitar a movimentação das seleções.

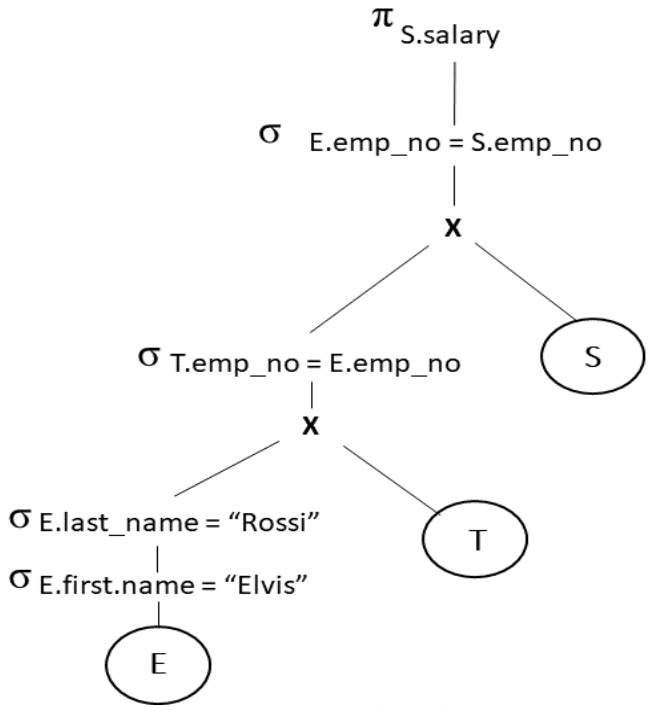
Figura 10 – Otimização Situação 01 aplicando cascateamento de seleção



Fonte: O autor (2020).

2. Movimente as seleções o mais para baixo possivel na árvore de consulta, com isso tenhe-se sua execução antecipada, de forma à diminuir o tamanho das relações no produto catesiano.

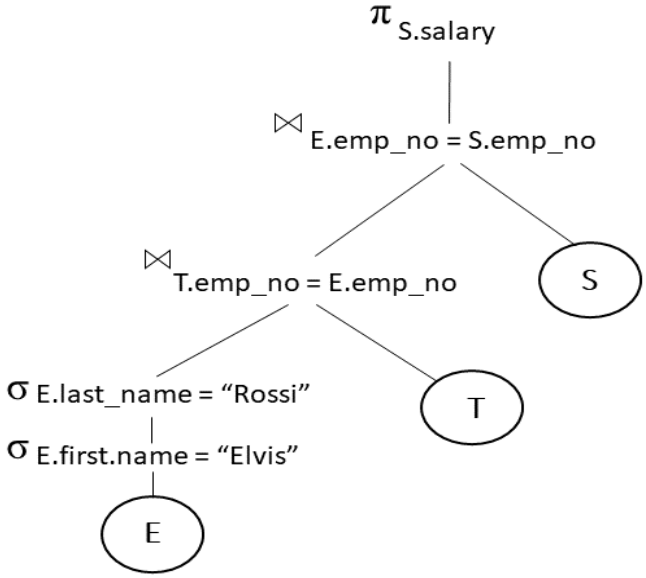
Figura 11 – Otimização Situação 01 aplicando regra 1 heuristica.



Fonte: O autor (2020).

3. Substituir produto cartesiano seguido de seleção por junção.

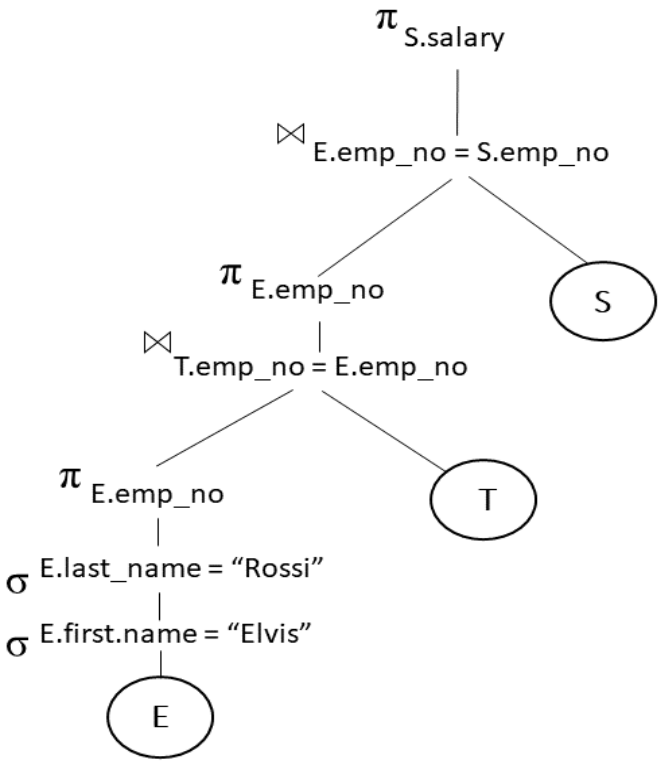
Figura 12 – Otimização Situação 01 aplicando regra 2 heuristica.



Fonte: O autor (2020).

4. Movimente as projeções o mais para baixo possivel na árvore de consulta.

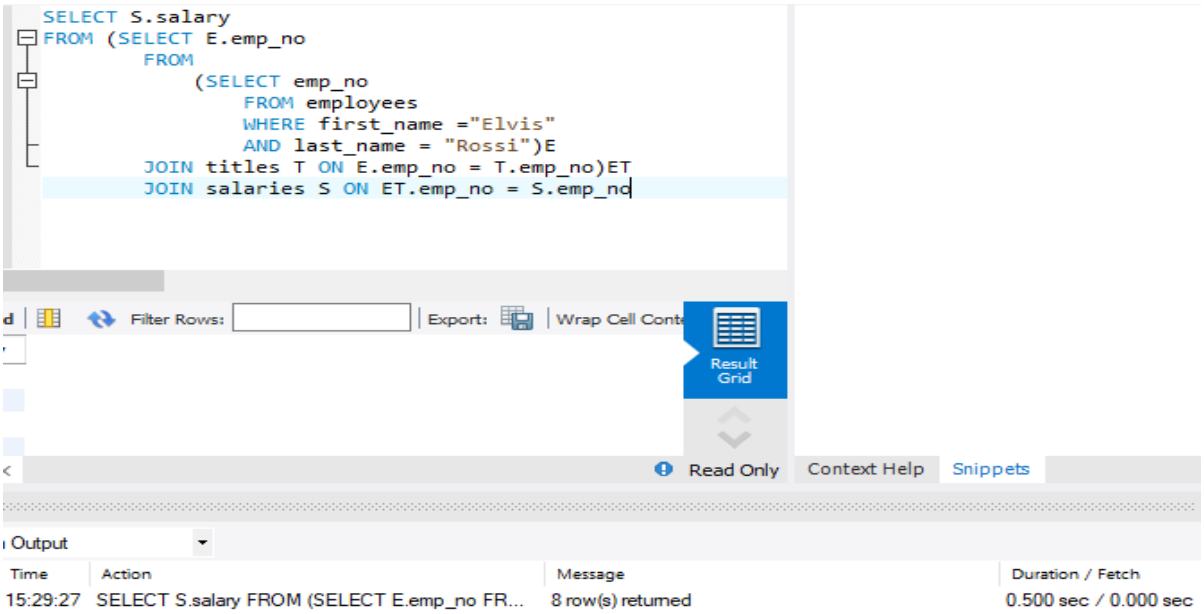
Figura 13 - Otimização Situação 01 aplicando regra 1 heuristica.



Fonte: O autor (2020).

A Figura 13 mostra a consulta otimizada em SQL proviniente da árvore de consulta Figura 12.

Figura 14 – Consulta otimizada Situação 01



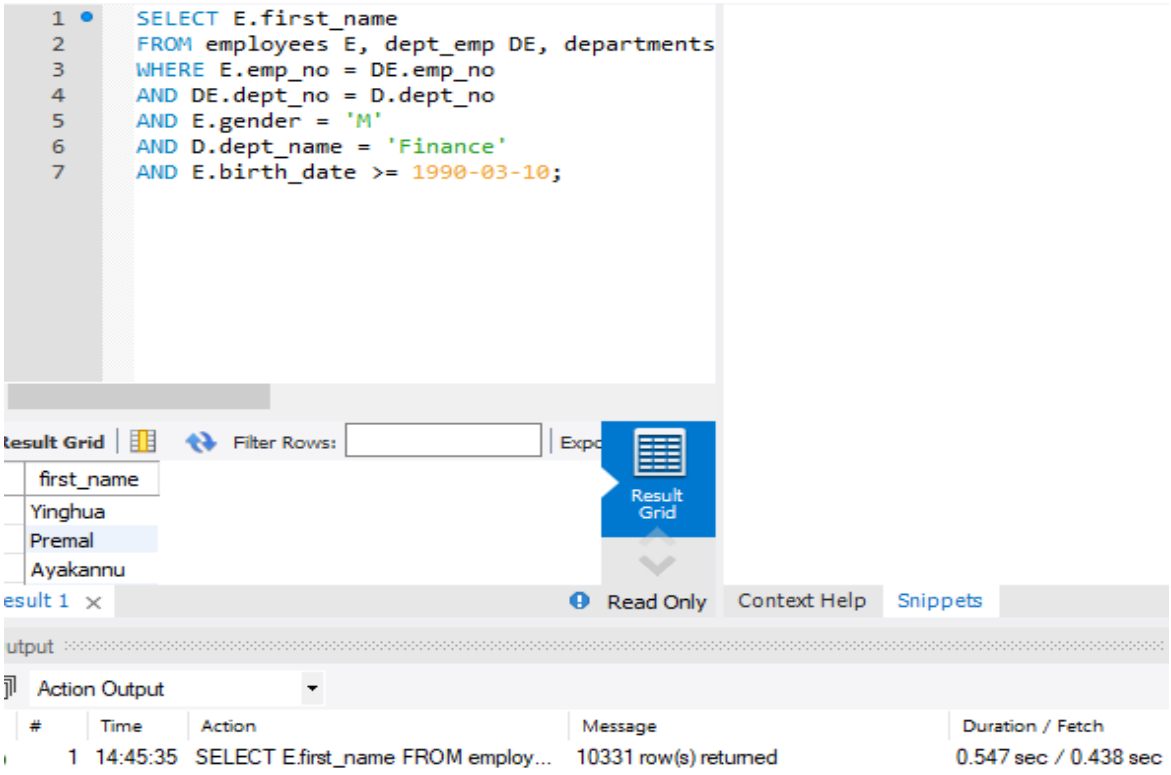
Fonte: Fonte: O autor (2019).

Nota-se que, a consulta não otimizada teve um tempo de execução de 0,906 segundos, enquanto a consulta otimizada teve um tempo de execução reduzido para 0,500 segundos. Esse ganho de desempenho foi oriundo da aplicação das operações de seleção e projeção antes de aplicar a junção regra 1 e 2, ou outras operações binárias, visto que operações binárias tem alto custo, pois o tamando de uma consulta proviniente desssa operação assim como as junções, é uma operação multiplicativa do tamanho das consultas de entrada. Com isso houve redução no número de tuplas de entrada no "*joins*", realizado nas relações nomeadas "E", "T" e "S".

3.2.2 Otimização Situação 02

A seguinte consulta, faz uma projeção do atributo "*first_name*" tendo como como alvo as junção de três tabelas "employyes", "*dept_emp*" e "*departaments*", fazendo uma a seleção de filtro pelos atributos "*birth_date = 1990-03-10*" e "*gender = M"*.

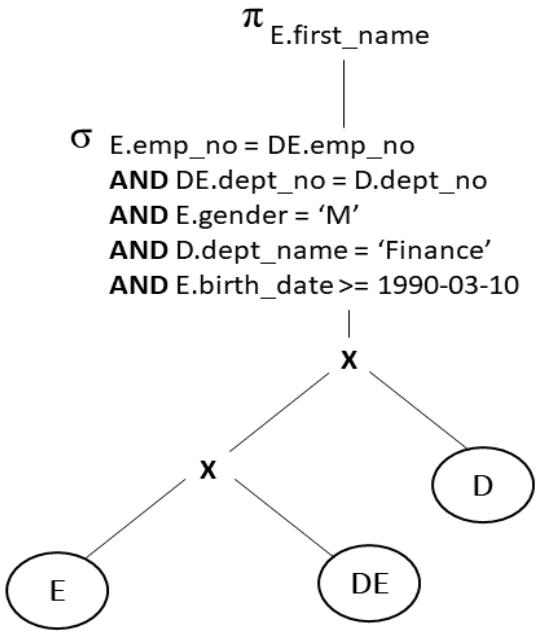
Figura 15 – Consulta não otimizada Situação 02



Fonte: O autor (2020).

Na Figura 15 temos a construção da arvore inicial (canônica) da consulta SQL Figura 14.

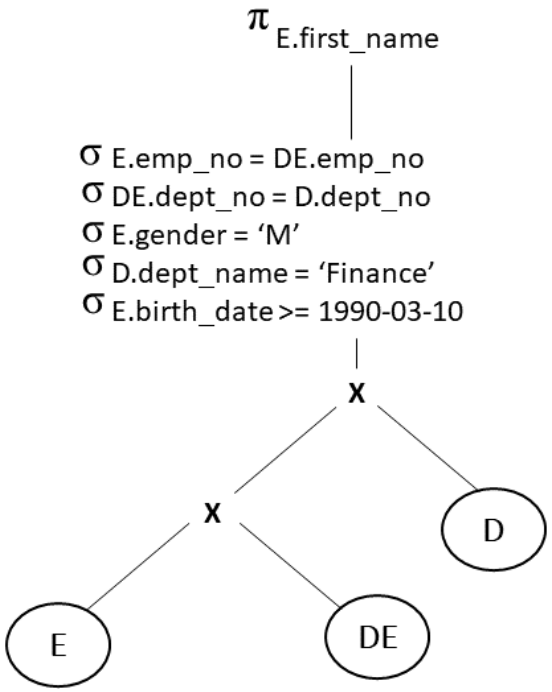
Figura 16 – Arvore inicial (canônica)



Fonte: O autor (2020).

1. Separar as seleções conjuntivas em um cascateamento de seleções, com objetivo de facilitar a movimentação das seleções.

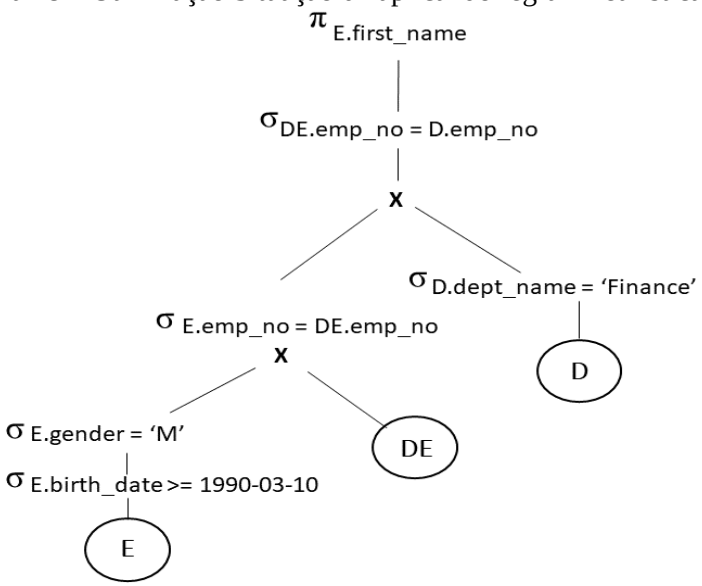
Figura 17 – Otimização Situação 02 aplicando cascateamento de seleção



Fonte: O autor (2020).

2. Movimente as seleções o mais para baixo possivel na árvore de consulta, com isso tenhe-se sua execução antecipada de forma à diminuir o tamanho das relações no produto catesiano.

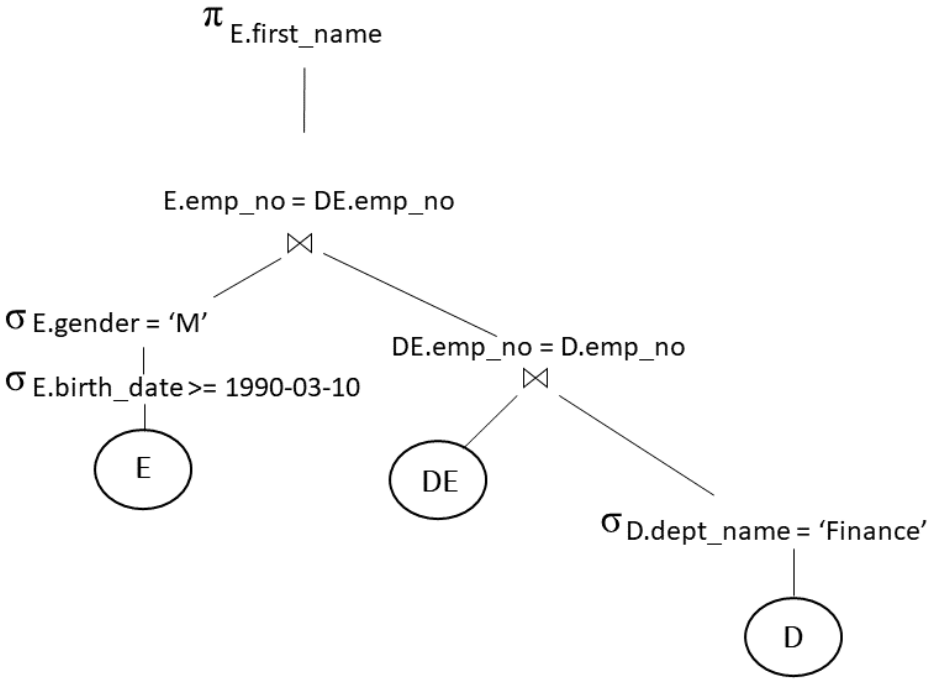
Figura 18 – Otimização Situação 02 aplicando regra 1 heuristica.



Fonte: O autor (2020).

3. Rearranje os nós folhas posicionando como folhas as relações com operações mais restritivas. Substitua o produto cartesiano seguido de seleção por junção.

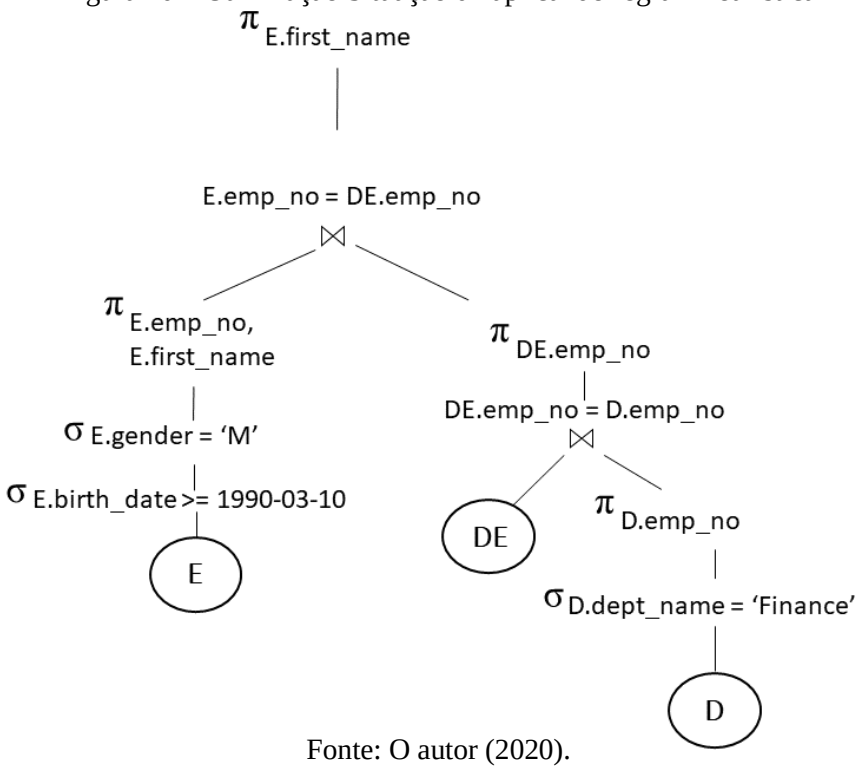
Figura 19 – Otimização Situação 02 aplicando regra 2



Fonte: O autor (2020).

4. Movimente as projeções o mais para baixo possivel na árvore de consulta.

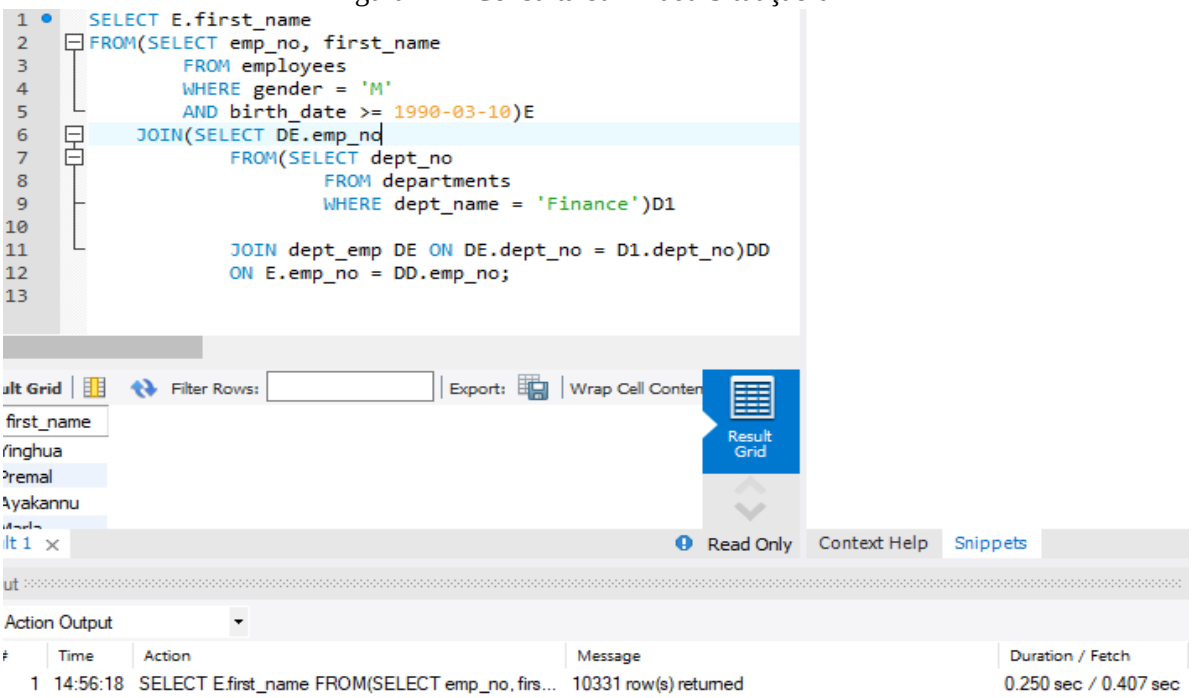
Figura 20 – Otimização Situação 02 aplicando regra 1 heuristica



Fonte: O autor (2020).

A Figura 20 mostra a consulta otimizada em SQL oriunda da árvore de consulta Figura 19.

Figura 21 – Consulta otimizada Situação 02



Fonte: O autor (2020).

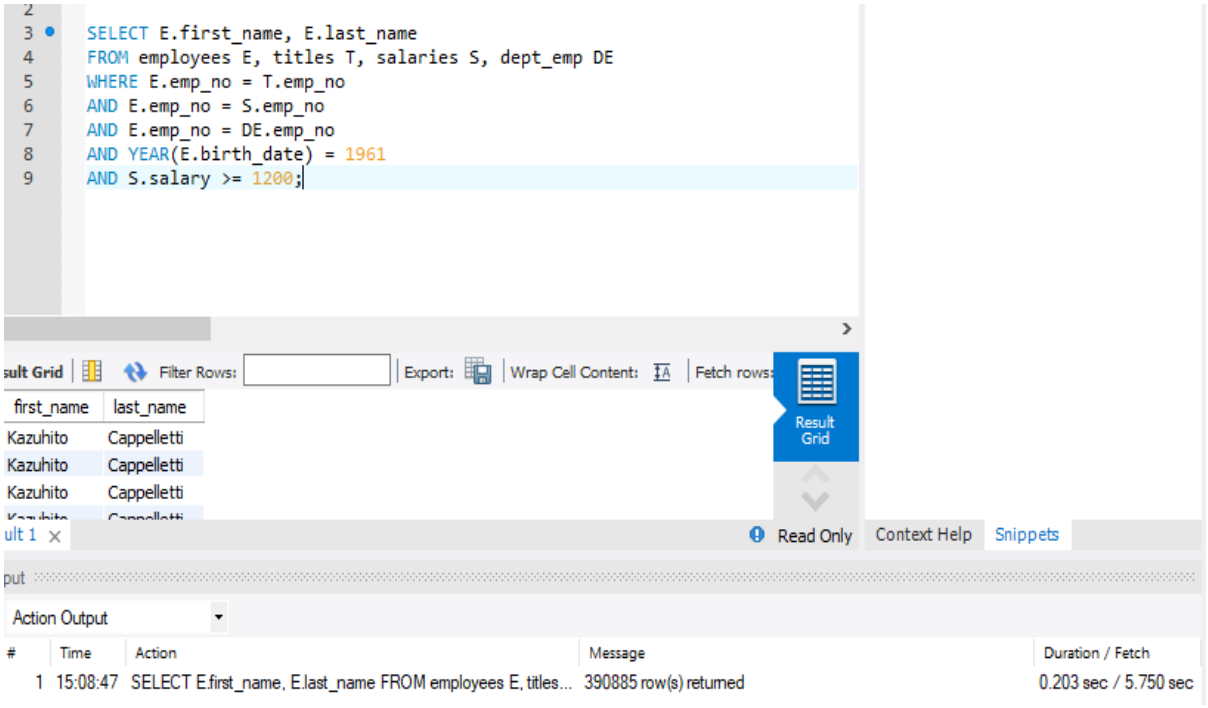
É possivel observar que, a consulta não otimizada teve um tempo de execução de 0,547 segundo e um tempo fetch (busca) de 0,438 segundos, então considerando esses dois times *Duration/Fetch* tem-se um total de 0,985 segundos, enqundo a consulta otimizada teve um tempo de execução e fetch reduzido para 0,250 e 0,407 respectivamente, somando um tempo de 0,657 segundos.

Esse ganho de performance se deve à aplicação de uma das principais regras de euristicas que tem como objetivo aplicar as operações de seleção e projeção antes de junção ou outras operações binárias reduzindo as consultas de entrada para os "joins", foi possivél também aplicar a regra que substitui as subconsultas por "*joins*" com condição de seleção mais restritiva nos niveis mais internos, visto no (passo 03 da arvore de consulta).

3.2.3 Otimização Situação 03

A consulta a seguir, faz uma projeção dos atributos '*first_name" e "last_name"* da tablea "employyes", utilizando seleção de filtro dos atributos "*birth_date =1961*" da tabela "*employyes*" e "*salary >= 1200*" da tabela "*salaries*", pela junção de 4 tabelas "*employyes", "titles", "salaries" e "dept_emp*".

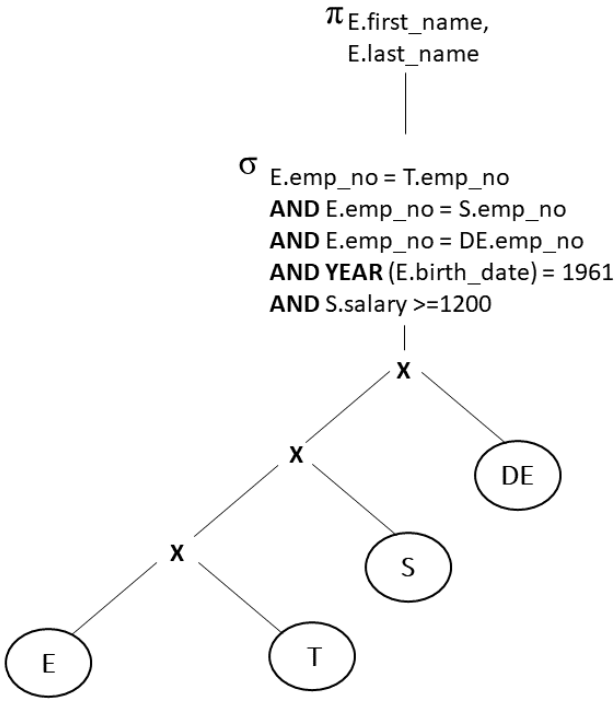
Figura 22 – Otimização não Situação 03



Fonte: O autor (2020).

Na Figura 22 temos a construção da arvore inicial (canônica) da consulta SQL Figura 21.

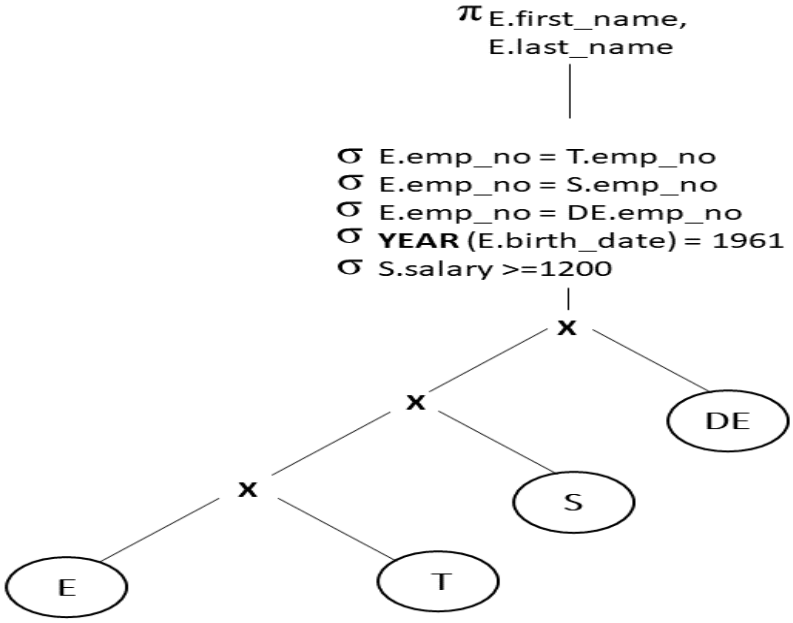
Figura 23 - Arvore inicial (canônica)



Fonte: O autor (2020).

1. Separar as seleções conjuntivas em um cascateamento de seleções, com objetivo de facilitar a movimentação das seleções

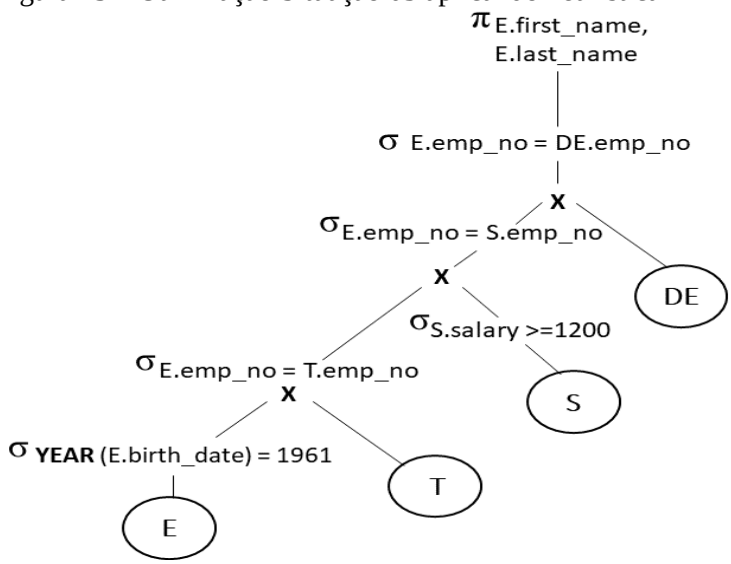
Figura 24 – Otimização Situação 03 aplicando cascateamento de seleção



Fonte: O autor (2020).

2. Movimente as seleções o mais para baixo possivel na árvore de consulta, com isso tenhe-se sua execução antecipada de forma à diminuir o tamanho das relações no produto catesiano.

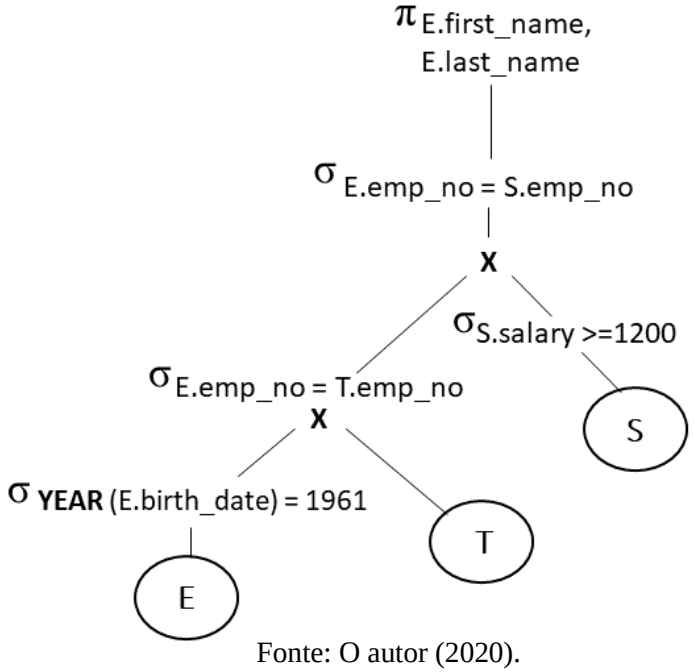
Figura 25 – Otimização Situação 03 aplicando heuristica



Fonte: O autor (2020).

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

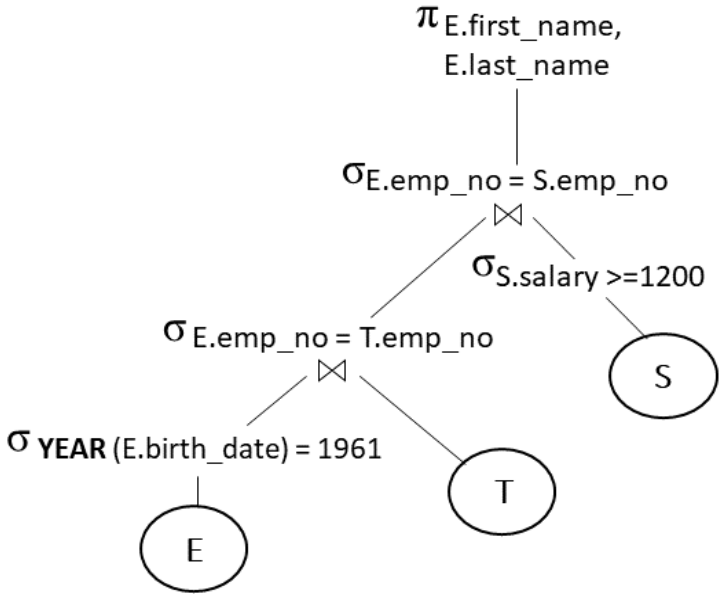
Figura 26 – Otimização Situação 03 aplicando heuristica 3



Fonte: O autor (2020).

4. Substitua o produto cartesiano seguido de seleção por junção.

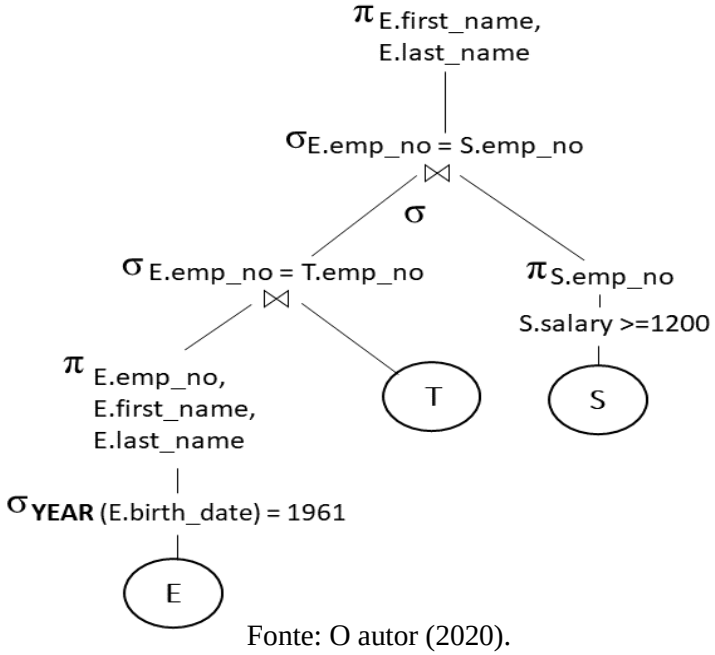
Figura 27 - Otimização Situação 03 aplicando heuristica 2



Fonte: O autor (2020).

5. Movimente as projeções o mais para baixo possivel na árvore de consulta.

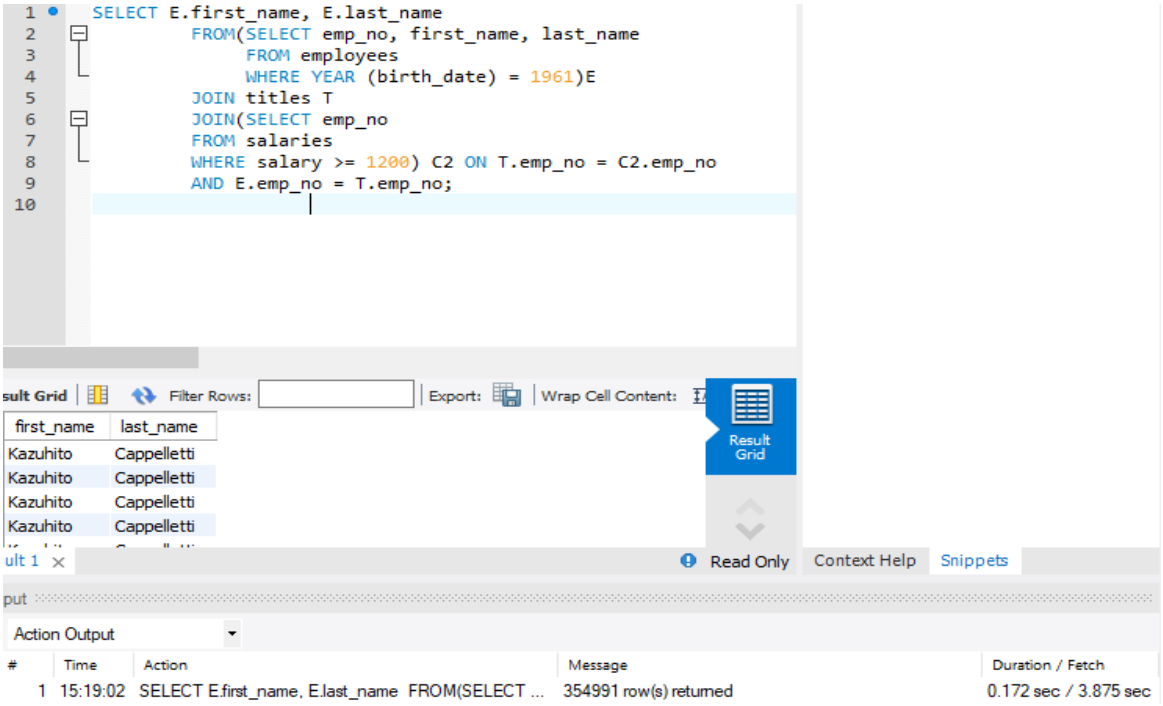
Figura 28 - Otimização Situação 03 aplicando heuristica 2



Fonte: O autor (2020).

A Figura 29 mostra a consulta otimizada em SQL oriunda da árvore de consulta Figura 28.

Figura 29 – Consulta otimizada Situação 03



Fonte: O autor (2020).

Observa-se que, a consulta não otimizada teve um tempo de execução de 0,203 segundos e um tempo fetch (busca) de 5,750 segundos considerando esses dois times *Duration/Fetch* tem-se um total de 5,953 segundos, enquanto a consulta otimizada teve um tempo de execução reduzido de 0,172 segundos e um fetch reduzido para 3,875 segundos, no geral obtém-se um tempo de 4,047 segundos.

Esse ganho de performance se deve graças a aplicação de uma das principais regras que tem como objetivo aplicar as operações de seleção e projeção antes de junção, mas também com aplicação da regra 3 de eurista eliminando o atributo "*DE.emp_no*" (passo 3 da arvore de consulta) reduzindo o número de junções.

4 CONSIDERAÇÕES FINAIS

Tomando como base os resultados alcançados nesse trabalho através do estudo e aplicação dessa técnica, a otimização de consultas baseada em heuristica apresentou resultados satisfatórios nos testes realizados conforme mostrado no quadro abaixo, pois teve o tempo de execução reduzido, melhorando assim seu desempenho.

Quadro 2 – Resultados obtidos

Descrição	**Consulta não Otimizada**	**Consulta Otimizada**
Situação 01		
Tempo de Execução	0,906 sec	0,500 sec
Situação 02		
Tempo de Execução	0,985 sec	0,657 sec
Situação 03		
Tempo de Execução	5,953 sec	4,047 sec

Fonte: O autor (2020).

Sendo assim o estudo dessa técnica pode comprovar que pequenos ajustes nas consultas SQL foi possivel melhorar o tempo de execução e que esse ganho de performance se mostrará com maior eficiencia em uma base de dados de maior proporção como: Amazon com mais de 62 milhões de clientes ativos estimando em 42TB, Facebook, ChoicePoint Google etc. Apartir disso, espera-se que estudantes, administradores de banco de dados (DBAs) e programadores se apoiem neste estudo para que diante de cada cénaria de consultas possam aplicar as regras heuristicas cabivés. Vale resaltar que alguns SGBDs possuem otimizadores em sua execução.

REFERÊNCIAS

CARNEIRO, Alessandro *et al*. **TURNIG -Técnicas de Otimização de Bancos de Dados:** um estudo comparativo: MySQL e PostegreSQL. Rio Grande: FURG, 2011.

CINTRA, João Paulo S.; CAPELLO, Renato. **Otimização de consultas de relacionais**. Campinas, SP: UNICAMP, 2005.Disponível em: http://www.ic.unicamp.br/~geovane/mo410-091/Ch15-Resumo.pdf. Acesso em: 01 jan. 2020. Trabalho de pós-graduação.

DATE, C. J. **Introdução a sistemas de bancos de dados**. 7. ed. Tradução Vandenberg Dantas de Souza, Publicare Consultoria e Serviços.Rio de Janeiro: Campus, 2000.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de banco de dados**. Tradução Daniel Vieira; Revisão técnica Enzo Seraphim e Thatyana de Faria Piola Seraphim. 6. ed. São Paulo: Pearson Addison Wesley, 2011.

GAVA, Mariana. **Otimização de consultas em SGBD relacional:** estudo de caso**.** 2010. 80f. Trabalho de Conclusão de Curso – (Graduação em Engenharia da Computação) - Universidade São Francisco, Itatiba, SP, 2010.

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. **Macoratti.net**. Disponível em: http://www.macoratti.net/13/06/sql_arcb.htm. Acesso em: 08 jan de 2020.

ORACLE CORPORATION. Employees sample database**. MySQL.** 2020. Disponível em: https://dev.mysql.com/doc/employee/en/. Acesso em: 20 dez. 2019.

RICARTE,Ivan L.M**. SQL**.
Disponível em:http://www.dca.fee.unicamp.br/cursos/PooJava/javadb/sql.html. Acesso em: 2 dez. 2019.

SILBERSCHATZ, Abraham; KORTH, Henry F. ; SUDARSHAN, S. **Sistema de banco de dados** . Tradução Daniel Vieira. Rio de Janeiro: Elsevier, 2012.

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