

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

**INDEXAÇÃO AUTÔNOMA DE
DADOS UTILIZANDO
ALGORITMOS GENÉTICOS**

**PRISCILLA DE FÁTIMA TRINDADE
MIEHE**

Dissertação apresentada como requisito
parcial à obtenção do grau de Mestre
em Ciência da Computação na Pontifícia
Universidade Católica do Rio Grande do
Sul.

Orientador: Prof. Duncan Dubugras Alcoba Ruiz
Co-Orientador: Prof. Felipe Meneguzzi

**Porto Alegre
2019**

**SUBSTITUA ESTA PÁGINA
PELA FICHA
CATALOGRÁFICA**

**SUBSTITUA ESTA PÁGINA
PELO TERMO DE
APRESENTAÇÃO**

DEDICATÓRIA

Dedico este trabalho aos meus filhos Ana Júlia e Alexander.

AGRADECIMENTOS

Primeiramente gostaria de agradecer minha mãe Palmira, e meus padrinhos Levi e Maria, responsáveis pela minha educação e princípios morais e éticos.

Agradeço meus familiares, tias, tios, primas e primos que sempre estiveram em oração pela minha trajetória.

Agradeço ao pai dos meus filhos Décio, que sempre me incentivou a correr atrás desse título e me amparou para que isso fosse possível. Agradeço também seus pais, à sua mãe Eli em memória, que sempre me deram força e estiveram na torcida para que tudo desse certo.

Agradeço aos meus irmãos de criação Luís Henrique, Leonardo, Liliane e Lucas, que sempre me apoiaram e me fizeram acreditar na minha capacidade.

Agradeço a congregação da Igreja Metodista Wesley, da qual faço parte, por estarem sempre em oração por mim a cada etapa realizada dessa trajetória.

Agradeço minha comadre Karla, e seus pais, por sempre se mostrarem solícitos comigo e meus filhos.

Agradeço minhas amigas Andreia, Neida, Pra. Maria Eugênia, Ludmila e Letícia, pelas orações e amparo emocional.

Agradeço meu orientador Duncan pela oportunidade dada, pelos ensinamentos, por acreditar em mim e por sempre dar o suporte necessário. Agradeço também meu co-orientador Felipe Meneguzzi que sempre me auxiliou, dando dicas essenciais de programação e escrita de artigos.

Aos meus colegas de laboratório - Julia, Silvia, Renata, Eulanda, Jônatas, Henry, Pedro, Maurício, Róger, Juarez e João - pelos ótimos momentos e ensinamentos durante minha trajetória.

Finalmente, gostaria de agradecer a SAP por ter financiado meus estudos.

INDEXAÇÃO AUTÔNOMA DE DADOS UTILIZANDO ALGORITMOS GENÉTICOS

RESUMO

Uma preocupação comum ao acessar dados em grandes bases de armazenamento é ter um desempenho satisfatório na execução das consultas. Para lidar com esse problema, várias técnicas são discutidas na literatura sobre estratégias de otimização de bancos de dados relacionais. Uma das técnicas mais utilizadas para acelerar o processo de acesso a dados é a indexação, onde estruturas são usadas para organizar registros de dados com o intuito de otimizar determinados tipos de operações de recuperação. Neste trabalho, propõe-se utilizar algoritmos genéticos para indexar dados de forma autônoma com a finalidade de otimizar o desempenho de conjuntos de consultas que são aplicadas com mais frequência. Como caso de estudo, foi utilizado um *benchmark* que fornece um banco de dados sintético, um esquema predefinido e um conjunto de consultas. Resultados da análise experimental mostram que a solução proposta obteve melhor desempenho, ao compará-la com dois otimizadores tradicionais de bancos de dados.

Palavras-Chave: Desempenho de Banco de Dados, Algoritmo Genético, Otimização de Consultas, Indexação de Dados.

AUTOMATIC INDEXING DATA USING GENETIC ALGORITHMS

ABSTRACT

The performance of databases is still a big concerning. One of the most common techniques used to speed up the data access process is indexing, which structures organize data records to optimize operations of retrieval. To deal with this problem, several techniques are discussed in the literature to optimize the relational databases. In this paper, we propose to use genetic algorithms to index data autonomously in order to optimize the performance of query operations. As a case of study, we used a benchmark that provides a synthetic database, a predefined schema, and a query set. Results from the experimental analysis show that the proposed solution performed better when compared to two traditional database optimizers.

Keywords: database tuning, genetic algorithm, query optimization, indexing data.

LISTA DE FIGURAS

Figura 2.1 – Exemplo de Índices estruturados como árvores (RG00).	36
Figura 2.2 – Comparativo do plano de execução de uma consulta submetida no MySQL utilizando o esquema do banco de dados do TPC-H com e sem índices.	38
Figura 2.3 – Comando SQL referente <i>query #1</i> retirada do conjunto de consultas utilizado no plano experimental. Os campos destacados em negrito representam aqueles que tenta-se otimizar.	39
Figura 3.1 – Fluxo de um Algoritmo Genético.	42
Figura 3.2 – Exemplo de critério de dominância para uma função de otimização multiobjetiva, baseado em (?).	45
Figura 3.3 – Método de Seleção da roleta viciada.	47
Figura 3.4 – Método de Seleção por Torneio.	48
Figura 4.1 – Arquitetura da estratégia utilizada para implementação do algoritmo genético para ambientes de SGBDs.	51
Figura 4.2 – Exemplo de codificação de um cromossomo baseado no esquema do TPC-H.	54
Figura 4.3 – Exemplo da aplicação do operador de cruzamento baseado no esquema do TPC-H.	57
Figura 4.4 – Exemplo de aplicação do operador de mutação baseado no esquema TPC-H.	58
Figura 5.1 – Esquema do TPC-H no estado inicial de configuração de índices (apenas PK's e FK's).	62
Figura 5.2 – Protocolo de execução do teste de performance do TPC-H (TCG12).	66
Figura 5.3 – Resultados obtidos sobre o tempo de execução e tamanho dos índices em Mb.	68
Figura A.1 – Consulta referente <i>Query #1</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que são foram desconsideradas.	85

Figura A.2 – Consulta referente <i>Query #2</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que ão foram desconsideradas.	86
Figura A.3 – Consulta referente <i>Query #3</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição.	87
Figura A.4 – Consulta referente <i>Query #4</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição.	88
Figura A.5 – Consulta referente <i>Query #5</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que ão foram desconsideradas.	89
Figura A.6 – Consulta referente <i>Query #6</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição.	89
Figura A.7 – Consulta referente <i>Query #7</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que ão foram desconsideradas.	90
Figura A.8 – Consulta referente <i>Query #8</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que ão foram desconsideradas.	91
Figura A.9 – Consulta referente <i>Query #9</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que ão foram desconsideradas.	92

Figura A.10 – Consulta referente <i>Query #10</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que ão foram desconsideradas.	93
Figura A.11 – Consulta referente <i>Query #11</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que ão foram desconsideradas.	94
Figura A.12 – Consulta referente <i>Query #12</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição.	95
Figura A.13 – Consulta referente <i>Query #13</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição.	96
Figura A.14 – Consulta referente <i>Query #14</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição.	96
Figura A.15 – Consulta referente <i>Query #15</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que ão foram desconsideradas.	97
Figura A.16 – Consulta referente <i>Query #16</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que ão foram desconsideradas.	98
Figura A.17 – Consulta referente <i>Query #17</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição.	99

Figura A.18 – Consulta referente <i>Query #18</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição.....	100
Figura A.19 – Consulta referente <i>Query #19</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição.....	101
Figura A.20 – Consulta referente <i>Query #20</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que ão foram desconsideradas.	102
Figura A.21 – Consulta referente <i>Query #21</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que ão foram desconsideradas.	103
Figura A.22 – Consulta referente <i>Query #22</i> . Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição.....	104
Figura B.1 – Plano de execução da consulta 1 no estado inicial do esquema (sómente com PK e FK).	105
Figura B.2 – Plano de execução da consulta 1 com o melhor indivíduo.	105
Figura B.3 – Plano de execução da consulta 2 no estado inicial do esquema (sómente com PK e FK).	106
Figura B.4 – Plano de execução da consulta 2 com o melhor indivíduo.	106
Figura B.5 – Plano de execução da consulta 3 no estado inicial do esquema (sómente com PK e FK).	107
Figura B.6 – Plano de execução da consulta 3 com o melhor indivíduo.	107
Figura B.7 – Plano de execução da consulta 4 no estado inicial do esquema (sómente com PK e FK).	108
Figura B.8 – Plano de execução da consulta 4 com o melhor indivíduo.	108
Figura B.9 – Plano de execução da consulta 5 no estado inicial do esquema (sómente com PK e FK).	109
Figura B.10 – Plano de execução da consulta 5 com o melhor indivíduo.	109

Figura B.11 – Plano de execução da consulta 6 no estado inicial do esquema (sómente com PK e FK).	109
Figura B.12 – Plano de execução da consulta 6 com o melhor indivíduo.	110
Figura B.13 – Plano de execução da consulta 7 no estado inicial do esquema (sómente com PK e FK).	110
Figura B.14 – Plano de execução da consulta 7 com o melhor indivíduo.	110
Figura B.15 – Plano de execução da consulta 8 no estado inicial do esquema (sómente com PK e FK).	111
Figura B.16 – Plano de execução da consulta 8 com o melhor indivíduo.	111
Figura B.17 – Plano de execução da consulta 9 no estado inicial do esquema (sómente com PK e FK).	111
Figura B.18 – Plano de execução da consulta 9 com o melhor indivíduo.	112
Figura B.19 – Plano de execução da consulta 10 no estado inicial do esquema (sómente com PK e FK).	112
Figura B.20 – Plano de execução da consulta 10 com o melhor indivíduo.	112
Figura B.21 – Plano de execução da consulta 11 no estado inicial do esquema (sómente com PK e FK).	113
Figura B.22 – Plano de execução da consulta 11 com o melhor indivíduo.	113
Figura B.23 – Plano de execução da consulta 12 no estado inicial do esquema (sómente com PK e FK).	114
Figura B.24 – Plano de execução da consulta 12 com o melhor indivíduo.	114
Figura B.25 – Plano de execução da consulta 13 no estado inicial do esquema (sómente com PK e FK).	115
Figura B.26 – Plano de execução da consulta 13 com o melhor indivíduo.	115
Figura B.27 – Plano de execução da consulta 14 no estado inicial do esquema (sómente com PK e FK).	116
Figura B.28 – Plano de execução da consulta 14 com o melhor indivíduo.	116
Figura B.29 – Plano de execução da consulta 15 no estado inicial do esquema (sómente com PK e FK).	117
Figura B.30 – Plano de execução da consulta 15 com o melhor indivíduo.	117
Figura B.31 – Plano de execução da consulta 16 no estado inicial do esquema (sómente com PK e FK).	118
Figura B.32 – Plano de execução da consulta 16 com o melhor indivíduo.	118
Figura B.33 – Plano de execução da consulta 17 no estado inicial do esquema (sómente com PK e FK).	119
Figura B.34 – Plano de execução da consulta 17 com o melhor indivíduo.	119

Figura B.35 – Plano de execução da consulta 18 no estado inicial do esquema (somente com PK e FK).	120
Figura B.36 – Plano de execução da consulta 18 com o melhor indivíduo.	120
Figura B.37 – Plano de execução da consulta 19 no estado inicial do esquema (somente com PK e FK).	121
Figura B.38 – Plano de execução da consulta 19 com o melhor indivíduo.	121
Figura B.39 – Plano de execução da consulta 20 no estado inicial do esquema (somente com PK e FK).	122
Figura B.40 – Plano de execução da consulta 20 com o melhor indivíduo.	122
Figura B.41 – Plano de execução da consulta 21 no estado inicial do esquema (somente com PK e FK).	123
Figura B.42 – Plano de execução da consulta 21 com o melhor indivíduo.	123
Figura B.43 – Plano de execução da consulta 22 no estado inicial do esquema (somente com PK e FK).	124
Figura B.44 – Plano de execução da consulta 22 com o melhor indivíduo.	124

LISTA DE TABELAS

Tabela 4.1 – Mapeamento das definições de algoritmos genéticos no contexto de banco de dados relacional	53
Tabela 5.1 – Informações das tabelas do esquema do TPC-H	63
Tabela 5.2 – Métodos de comparação utilizados.	64
Tabela 5.3 – Parâmetros usados no experimento.....	65
Tabela 5.4 – Resultados obtidos ao executar o teste de performance do TPC-H benchmark por cromossomo sugerido para cada método.	67
Tabela 5.5 – Resultado do plano de execução de cada consulta.....	69
Tabela 5.6 – Consultas mais beneficiadas.	70
Tabela 5.7 – Uso dos índices em cada método.	70

LISTA DE ALGORITMOS

Algoritmo 3.1 – Pseudocódigo genérico para AGs.	42
Algoritmo 5.1 – Algoritmo Geneético para indexar dados de forma dinâmica para SGBDs.	64

LISTA DE SIGLAS

AG – Algoritmo Genético

AM – Aprendizado de Máquina

AR – Aprendizado por Reforço

DBA – Database Administrator

EDB – EnterpriseDB

IA – Inteligência Artificial

ISP – Index Selection Problem

ITLCS – Index Tuning Learning Classifier System

MOP – Multi-objective Problem

OLAP – *Online Analytical Processing*

OLTP – *Online Transaction Processing*

NSGA-II – Non-dominated Sorting Genetic Algorithm

POWA – PostgreSQL Workload Analyzer

SF – Scale Factor

SGBD – Sistema de Gerenciamento de Banco de Dados

SQL – Structured Query Language

TPCH – Transaction Processing Performance Council Ad-hoc

LISTA DE SÍMBOLOS

n_{pop} – Número de indivíduos da população	32
n_{gen} – Número de gerações	32
n_{mutpb} – Probabilidade de mutação	32
n_{mutrt} – Taxa de mutação por bits	32
n_{cxbp} – Probabilidade de cruzamento	32
n_{elite} – Número de indivíduos no elitismo	32

SUMÁRIO

1	INTRODUÇÃO	31
1.1	CARACTERIZAÇÃO DO PROBLEMA	32
1.2	OBJETIVOS	33
1.3	ORGANIZAÇÃO	33
2	INDEXAÇÃO DE DADOS EM BANCOS RELACIONAIS	35
2.1	TIPOS DE ÍNDICES	35
2.1.1	ÍNDICES DE ÁRVORES B+	36
2.1.2	ÍNDICES MAPA DE BITS	37
2.2	OTIMIZADORES DE CONSULTAS	37
2.3	PLANO DE EXECUÇÃO DAS CONSULTAS	38
2.4	CONSIDERAÇÕES DO CAPÍTULO	39
3	ALGORITMOS GENÉTICOS	41
3.1	REPRESENTAÇÃO DO INDIVÍDUO	42
3.2	POPULAÇÃO	43
3.3	FUNÇÃO DE AVALIAÇÃO	44
3.3.1	NON-DOMINATED SORTING GENETIC ALGORITHM (NSGA-II)	45
3.4	CRITÉRIO DE PARADA	46
3.5	SELEÇÃO	46
3.5.1	ROLETA VICIADA	47
3.5.2	RANQUEAMENTO	48
3.5.3	TORNEIO	48
3.6	CRUZAMENTO E MUTAÇÃO	49
3.7	CONSIDERAÇÕES DO CAPÍTULO	49
4	UMA ESTRATÉGIA BASEADA EM ALGORITMOS GENÉTICOS COM INDEXAÇÃO DINÂMICA DE DADOS PARA OTIMIZAÇÃO DE BANCOS DE DADOS RELACIONAIS	51
4.1	MOTIVAÇÃO	51
4.2	VISÃO GERAL DA ESTRATÉGIA	52
4.3	GERAÇÃO DA POPULAÇÃO INICIAL	53
4.4	FUNÇÃO DE AVALIAÇÃO	54

4.5	OPERADORES GENÉTICOS	56
4.5.1	SELEÇÃO	56
4.5.2	CRUZAMENTO	57
4.5.3	MUTAÇÃO	58
4.6	CRITÉRIO DE PARADA.....	58
4.7	CONSIDERAÇÕES DO CAPÍTULO	59
5	ANÁLISE EXPERIMENTAL.....	61
5.1	TPC-H BENCHMARK	61
5.1.1	ESQUEMA DE DADOS DO TPC-H	62
5.2	PLANO EXPERIMENTAL	63
5.3	AVALIAÇÃO DO MÉTODO	65
5.3.1	TESTE DE PERFORMANCE DO TPC-H	65
5.3.2	AVALIAÇÃO DOS RESULTADOS	67
5.4	CONSIDERAÇÕES DO CAPÍTULO	71
6	TRABALHO RELACIONADOS	73
6.1	GENETIC ALGORITHM FOR DATABASE INDEXING	73
6.2	ITLCS ON HYBRID STORAGE ENVIRONMENTS	73
6.3	RELATIONAL DATABASE INDEX SELECTION ALGORITHM	74
6.4	OTIMIZAÇÃO DE INDEXAÇÃO DE DADOS COM IA	74
6.5	CONSIDERAÇÕES DO CAPÍTULO	75
7	CONCLUSÃO	77
7.1	CONTRIBUIÇÕES	78
7.2	LIMITAÇÕES	78
7.3	TRABALHOS FUTUROS	79
	REFERÊNCIAS	81
	APÊNDICE A – Comando SQL de todas as consultas utilizadas na execução do algoritmo genético	85
	APÊNDICE B – Plano de execução de todas as consultas no estado inicial do esquema e com o melhor indivíduo encontrado configurado no esquema.....	105

1. INTRODUÇÃO

O crescimento do volume de dados tem sido um fator de relevância, exigindo adaptação dos sistemas de armazenamento em relação à disposição de espaço e agilidade na recuperação da informação. À medida que o volume de dados cresce, aumenta também a necessidade de abordagens mais eficientes com a finalidade de aprimorar a usabilidade dos sistemas (Sim13). Acessos lentos a bancos de dados envolvem não apenas o desperdício de tempo para uma empresa, mas também pode indicar um alto custo computacional. Uma das alternativas para melhorar o desempenho de consultas em sistemas de banco de dados é utilizar estruturas de índices (RG00). Em geral, costuma-se adicionar índices para melhorar o desempenho de consultas. Tarefa que demanda análise, pois não é uma boa prática criar índices em todas as colunas de uma tabela.

Em geral, os índices são criados por um Database Adminstrator (DBA) durante a implementação do esquema ou de forma reativa, quando se considera o tempo de resposta das consultas mais executadas. Embora os otimizadores de Sistemas de Gerenciamento de Banco de Dados (SGBD) possam auxiliar na decisão de quando e onde criar um índice, a escolha final geralmente é tomada pelo DBA. Em um cenário ideal, consultas que envolvam as mesmas colunas de forma frequente, deveriam utilizar índices, pois resolveria o problema de desempenho dessas operações. Entretanto, a presença de índices implica em custos com relação ao armazenamento e recursos computacionais, sempre que inserções ou atualizações ocorram nas tabelas com colunas indexadas (RG00). A escolha de um conjunto ideal dessas estruturas, para fins de consulta, não é suficiente para garantir o desempenho que se deseja. É necessário também promover um equilíbrio com a quantidade do espaço ocupado.

1.1 Caracterização do Problema

Algumas questões devem ser observadas quando se utiliza índices como a quantidade de memória ocupada por essas estruturas, as colunas requisitadas nas consultas, quantidade de registros em uma tabela e seletividade de uma consulta, por exemplo. Utilizar índices em todas as colunas de uma tabela não é uma boa prática por consumir recursos do sistema de forma desnecessária, além de impactar em operações de atualização de dados. O tempo gasto para atualizar os índices pode afetar a escalabilidade do banco. Em alguns casos a execução da consulta pode ser prejudicada, pois o espaço ocupado poderá prejudicar o *cache* dos dados ou índices que realmente são importantes. Além disso, acessar primeiro o índice para depois acessar o dado tem um custo, e às vezes esse custo pode ser maior do que acessar os dados diretamente. E nem sempre o otimizador do sistema escolhe de corretamente de que forma irá executar o acesso aos dados.

Outros pontos também levam a um desempenho ruim como índices que não são utilizados, ou colunas que devem ser indexadas, mas não são. O espaço ocupado por índices que não são utilizados indica o gerenciamento ineficiente do espaço de memória disponível no SGBD, e índices que deveriam ser criados em colunas, mas não são, afetam o processo de acesso a dados, haja visto que, o sistema precisará realizar uma varredura completa sobre as tabelas. Atingir a melhor configuração de índices não é uma tarefa trivial, e como o desempenho de consultas é um recurso para otimizar é preciso revisar com frequência, o que pode ser bom em um momento pode não ser mais no futuro. O problema de seleção de índices é conhecido como *Index Selection Problem* (ISP) e considerado NP-Difícil (?). Estratégias evolucionárias e Algoritmos Genéticos (AG) são reconhecidos como técnicas factíveis para resolver o problema ISP (?).

Diferentes abordagens foram utilizadas em pesquisas recentes para otimizar o desempenho de um SGBD (BLC⁺16, DPP⁺18, PIM15). Com isso, observou-se a oportunidade de desenvolver uma estratégia para indexar dados automaticamente com a finalidade de otimizar operações de consulta realizadas em ambiente de bancos de dados relacionais.

1.2 Objetivos

Considerando a relevância do problema descrito, propõe-se neste trabalho uma estratégia para otimizar o desempenho de consultas submetidas sobre bancos de dados relacionais. Para isso, através de algoritmos genéticos tenta-se explorar as possíveis soluções de indexação de dados, aplicando ao processo do ciclo evolutivo do algoritmo as principais consultas utilizadas em um dado esquema de SGBD. O principal objetivo do presente trabalho é obter uma configuração de índices próxima da ótima que otimize o tempo de respostas de conjuntos de consultas submetidas de forma frequente por um determinado período de tempo, considerando a quantidade de memória alocada. Adicionalmente, espera-se obter uma boa relação de custo-benefício entre a configuração de índices gerada e a sua utilização pelo otimizador do sistema no plano de execução das consultas.

Para análise da solução proposta, a presente estratégia foi aplicada em um esquema de banco de dados sintético que simula o processamento de requisições *Online Analytical Processing* (OLAP) de forma intensiva. Para comparação, foram escolhidos dois otimizadores de SGBDs com a finalidade de sugerir uma configuração de índices de acordo com as estatísticas provenientes de seus respectivos sistemas, e um trabalho ITLCS (PNR18), que está descrito de forma detalhada no Capítulo 6. Com o objetivo de avaliar a significância dos resultados obtidos, foi executado um *benchmark*, que realiza operações de inclusão e exclusão de dados, além de fornecer um conjunto de consultas. Ao final, compara-se o desempenho da solução encontrada na presente estratégia com otimizadores de consulta de dois SGBDs. O objetivo é superar os resultados obtidos nessas ferramentas, e ainda, os resultados do ITLCS, trabalho escolhido para comparação.

1.3 Organização

Este trabalho está organizado da seguinte forma: nos Capítulos 2 e 3 são apresentados a fundamentação teórica, abordando os temas de indexação de dados em bancos relacionais, plano de execução de consultas e algoritmos genéticos. No Capítulo 4, uma solução para o problema de criação de índices é descrita em detalhes. No Capítulo 5, a análise experimental, realizada em razão de avaliar a qualidade da solução proposta é descrita. No Capítulo 6 os trabalhos relacionados ao tema de pesquisa são apresentados. Por fim, este documento é concluído com a apresentação das considerações finais e trabalhos futuros no Capítulo 7.

2. INDEXAÇÃO DE DADOS EM BANCOS RELACIONAIS

Um SGBD é responsável por controlar o armazenamento, recuperação, exclusão, segurança e integridade dos dados (SKS10). Em geral, um banco de dados relacional armazena seus dados em tabelas, organizadas por colunas. As tabelas possuem tipicamente chaves em uma ou mais colunas que unicamente identificam uma linha na tabela. Além das chaves primárias, índices adicionais podem ser criados em determinado conjunto de registro de dados, cada um com uma chave de pesquisa diferente, para acelerar para acelerar operações de pesquisa não suportadas de maneira eficiente pela organização do arquivo utilizada para armazenar os dados (RG00). O uso de índices pode trazer grandes avanços para o desempenho do banco de dados, principalmente quando filtram poucas linhas de um conjunto, conceito conhecido como seletividade (WLKC16).

Entretanto, o uso de índices requer algumas análises, antes de aplicá-los. Em operações de atualização de dados, como `INSERT` e `DELETE`, por exemplo, fazem com que o SGBD desloque recursos internamente para manter esses índices atualizados e associados (?). A manutenção de índices também é um fator relevante por demandar tempo e memória para armazenamento dessas estruturas, tornando ineficaz manter índices que não são utilizados. Ou seja, criar um número excessivo de índices ou utilizar índices em tabelas que são frequentemente atualizadas tornará o desempenho do banco insatisfatório. Nas seções seguintes são descritos com maiores detalhes dois tipos de índices, mais comumente utilizados, conceitos de otimizadores de consultas tradicionais e planos de execução de consultas.

2.1 Tipos de Índices

Índices são estruturas que organizam os registros com o intuito de otimizar determinadas operações de acesso a dados (RG00). A estrutura mais simples para armazenamento de dados é um arquivo não ordenado, conhecido como arquivo *heap*, onde os dados são adicionados de forma aleatória. Outra forma de organizar entradas de dados é criando uma estrutura em árvore, onde as entradas são organizadas de forma ordenada pelo valor da chave de pesquisa. Outra estrutura de armazenamento conhecida são os índices mapa de bits, onde as entradas de dados são mapeadas através de uma cadeia de bits que representam a presença de uma dada informação. As subseções seguintes descrevem de forma mais detalhada os índices baseados em árvores e índices mapa de bits.

2.1.1 Índices de Árvores B+

Índices de árvore B+ são estruturas de dados organizadas de forma ordenada pelo valor da chave de pesquisa. Os nós internos da árvore direcionam a pesquisa e os nós folhas armazenam as entradas de dados. Essa estrutura permite localizar de forma eficiente todas as entradas de dados com valores de chaves de pesquisa por um intervalo desejado (RC02). Em bancos de dados relacionais esse é o tipo mais comum de índice utilizado. Índices baseados em árvores são bastante usados para trabalhar em operações de consulta com grandes volumes de dados, já que proporcionam rápido acesso, tanto sequencialmente quanto para acesso único (SKS10). A árvore B+ é uma estrutura de índice que garante que todos os caminhos da raiz até uma folha em determinada árvore tenham o mesmo comprimento, ou seja, a árvore está sempre balanceada em altura (CLRS01). A inserção e a remoção são mais custosas, devido à necessidade de novos ajustes na estrutura da árvore, em situações de divisão de nós, por exemplo. A Figura 2.1 ilustra um exemplo de índices baseados em árvores.

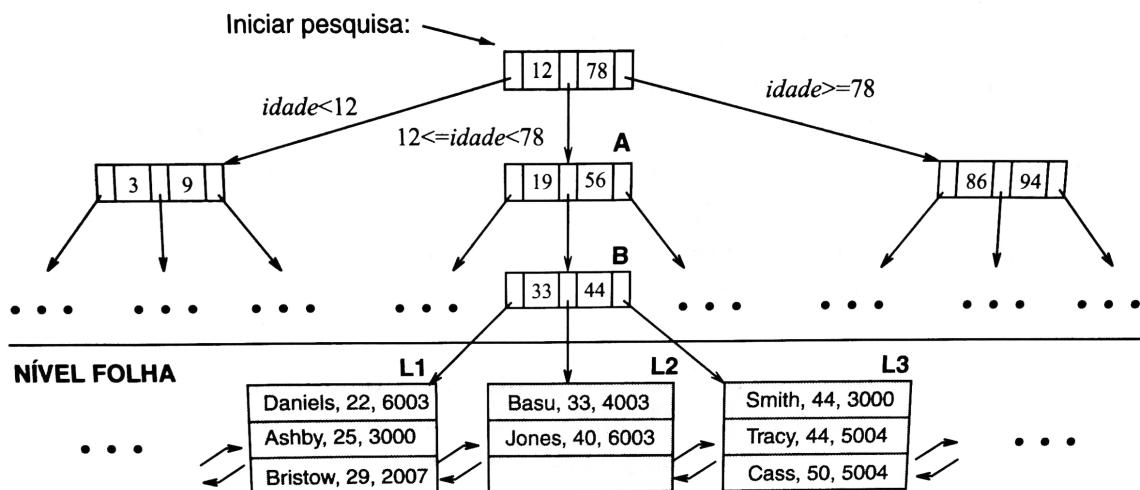


Figura 2.1 – Exemplo de Índices estruturados como árvores (RG00).

2.1.2 Índices Mapa de Bits

Os índices Mapa de Bits são estruturas projetadas para permitir consultas eficientes em colunas com múltiplas chaves (SKS10). Essa estrutura é comumente utilizada em sistemas que possuem colunas de baixa cardinalidade, ou seja, com pouca variação de valores nas linhas de uma tabela. Esses índices usam uma matriz de bits para representar a existência de um valor ou condição. Ao criar um índice mapa de bits em uma coluna, o SGBD monta um mapa de bits para todas as linhas da tabela, contendo todos os valores possíveis para a coluna. O SGBD grava um bit 1 onde o valor existe em uma determinada linha ou um bit 0 para os valores que não existem.

2.2 Otimizadores de Consultas

Os SGBDs tradicionais implementam em sua arquitetura uma funcionalidade conhecida como otimizador de consultas. Um otimizador de consultas relacional típico tem como principal tarefa encontrar um bom plano de execução de consultas. Tarefa que consiste em encontrar caminhos alternativos para as consultas submetidas ao banco de dados, com o menor custo (métrica calculada pelo próprio SGBD) possível (RG00, Cap. 15). Uma das formas utilizadas para realizar essa tarefa é através de informações estatísticas armazenadas no sistema, que nem sempre estão atualizadas (SZT12).

Para isso, muitas vezes é necessária a intervenção de um DBA, para decidir quais medidas devem ser consideradas de forma a beneficiar o desempenho do sistema (RC02). Expressões de consulta em *Structured Query Language* (SQL) geram várias possibilidades de caminhos. O otimizador inicialmente enumera os planos alternativos para avaliar a expressão, e em seguida estima o custo de cada um (RG00, Cap. 15). Estratégias como estimar o tamanho do resultado de uma consulta é uma das etapas realizadas para avaliar o custo de um plano de execução de consulta. As estimativas utilizadas em um SGBD para tamanhos e custos de resultado são, em geral, aproximações dos valores concretos (SKS10). O comportamento desejado, ao utilizar otimizadores de consultas, é encontrar um bom plano de execução e evitar os piores planos.

2.3 Plano de Execução das Consultas

A principal tarefa de um otimizador de SGBDs é encontrar um plano ideal para executar expressões SQL. Diversas pesquisas utilizaram estratégias diferentes para melhorar o desempenho de SGBDs, dentre elas, a técnica mais utilizada é a indexação de dados (SSS19). Executar consultas sem utilizar índices pode fazer com que o sistema realize uma varredura completa nas tabelas, operações que demandam bastante tempo e alocação de recursos, especialmente quando se tem um número alto de registros. Com o intuito de mostrar o grau de complexidade na estruturação de índices de forma a beneficiar o desempenho de consultas sobre um sistema, foram realizados alguns testes em um servidor MySQL. O teste consiste em submeter ao banco de dados uma consulta em SQL com uma cláusula WHERE bem simples, utilizando junto um operador de comparação. A comando de consulta em SQL utilizado nos testes pode ser visto na Figura 2.3.

Dois cenários foram utilizados na execução do comando de consulta: com e sem índices no esquema. A Figura 2.2 mostra o plano de execução adotado pelo otimizador do SGBD. Pode-se observar que, para essa consulta, e havendo a presença de índice sobre a coluna requisitada na cláusula WHERE, o otimizador irá fazer uso da estrutura de índice no plano de execução. Sobre o aspecto da otimização no tempo de resposta, ao comparar os valores obtidos com as duas abordagens, pode-se notar que o valor do custo reduziu em 35%. Sobre o número de registros retornados, pode-se afirmar que a consulta examinou a quantidade necessária de linhas para retornar os critérios desejados da operação de consulta, com base no número de *hints* retornado no plano de execução (RG00).

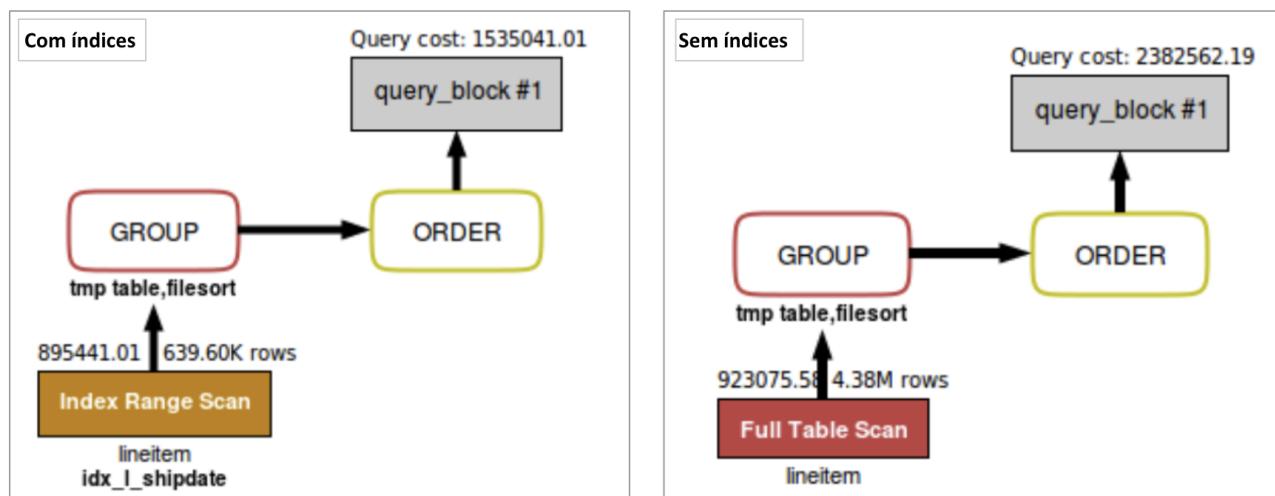


Figura 2.2 – Comparativo do plano de execução de uma consulta submetida no MySQL utilizando o esquema do banco de dados do TPC-H com e sem índices.

```

SELECT l_returnflag, l_linenstatus,
       SUM(l_quantity) AS sum_qty,
       SUM(l_extendedprice) AS sum_base_price,
       SUM(l_extendedprice * (1 - l_discount)) AS sum_disc_price,
       SUM(l_extendedprice * (1 - l_discount) * (1 + l_tax)) AS sum_charge,
               AVG(l_quantity) AS avg_qty, AVG(l_extendedprice) AS avg_price,
               AVG(l_discount) AS avg_disc, COUNT(*) AS count_order
  FROM
    lineitem
 WHERE
   l_shipdate <= DATE '1994-7-17' - INTERVAL '108' DAY
 GROUP BY l_returnflag, l_linenstatus
 ORDER BY l_returnflag, l_linenstatus;

```

Figura 2.3 – Comando SQL referente *query #1* retirada do conjunto de consultas utilizado no plano experimental. Os campos destacados em negrito representam aqueles que tenta-se otimizar.

2.4 Considerações do Capítulo

Nesse capítulo foi apresentada uma visão geral sobre Bancos de Dados Relacionais voltada para estruturas de índices e demais funcionalidades desses sistemas, como otimizadores de consulta e planos de execução. O problema de otimização de banco de dados é um fator de relevância, tanto em grandes corporações quanto na comunidade científica. O *trade-off* entre tempo de resposta na submissão de consultas e quantidade de memória alocada para armazenamento de índices é um assunto amplamente discutido, desde o significativo crescimento no volume de dados armazenados. Várias abordagens são propostas com a finalidade de encontrar soluções ideais para o problema citado, de forma a beneficiar ambientes tradicionais de armazenamento. Entretanto, independente da estratégia utilizada, não existe um consenso na literatura sobre uma solução universal que resolva da melhor maneira o problema de indexação de dados para aprimorar o desempenho de SGBDs, o que motiva a busca por novas soluções.

3. ALGORITMOS GENÉTICOS

A busca pela melhor solução dentro de um conjunto de soluções possíveis é um objetivo comum no desenvolvimento de algoritmos. Esse conjunto de soluções possíveis é conhecido por espaço de busca (Mit96). Para um espaço de busca pequeno, todas as soluções podem ser examinadas em um tempo razoável. No entanto, uma busca exaustiva rapidamente se torna inviável quando esse espaço torna-se grande. Nesses casos, o problema é definido como sendo difícil ou intratável devido o seu tempo de execução ser considerado da ordem de uma função exponencial, ou factorial. Técnicas de exploração são úteis na busca de soluções para esses tipos de problemas.

AEs (Algoritmos Evolutivos) são uma coleção de técnicas de otimização inspiradas no processo de evolução Darwiniana (Mit96). Essas técnicas têm se mostrado eficientes para varrer o espaço de busca e encontrar uma aproximação da solução ótima. AEs são considerados uma heurística por não serem capazes de garantir uma solução ótima, mas sim, uma aproximação. AGs (Algoritmos Genéticos) são um das técnicas contidas em AEs. Holland desenvolveu o conceito de Algoritmos Genéticos inspirado na teoria evolucionista que explica a origem das espécies (?), simulando os princípios da evolução biológica.

O princípio de funcionamento de um AG ocorre a partir de uma população de indivíduos¹ e de acordo com a avaliação de quão apto cada um indivíduo da população é para ser um possível candidato à solução do problema em questão. Deseja-se que os indivíduos mais aptos tenham maior chance de serem selecionados para reprodução, e de acordo com a teoria de C. Darwin, o princípio de seleção privilegia aqueles que são mais aptos com, maior longevidade. O processo de seleção escolhe os pais para recombiná-los entre si suas características (cruzamento), gerando assim novos descendentes. Os descendentes estarão sujeitos a alterações em suas características (mutação), exatamente como ocorre no processo biológico natural. Essa sequência de eventos é repetida, e cada população nova originada é denominada geração. As gerações vão sendo criadas até que um determinado critério de parada seja alcançado. Para melhor compreensão, o fluxo de funcionamento de um AG é ilustrado na Figura 3.1

AGs são bastante aplicados em problemas complexos de otimização, onde existem diversos parâmetros ou características que precisam ser combinadas para alcançar a melhor solução, com muitas restrições ou condições que não podem ser representadas matematicamente, e com grande espaço de busca. Essa técnica tenta resolver problemas de otimização, encontrando a melhor solução possível em modelos de decisão não-lineares que frequentemente têm várias soluções sub-ótimas. O pseudocódigo de um AG está resumido sucintamente no Algoritmo 3.1.

¹O termo cromossomo também é referenciado como indivíduo.

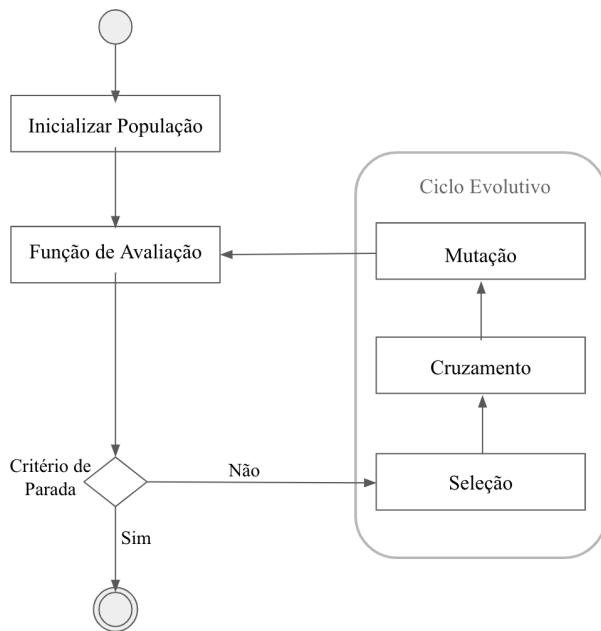


Figura 3.1 – Fluxo de um Algoritmo Genético.

- 1: Gerar população inicial
- 2: Executar função de avaliação
- 3: **for** Cada indivíduo da população atual **do**
- 4: Selecionar os indivíduos aptos
- 5: Aplicar operadores genéticos nos indivíduos selecionados
- 6: Criar novos indivíduos
- 7: Calcular função de avaliação de cada indivíduo
- 8: Gerar nova população e assumir como atual
- 9: Verificar critério de parada
- 10: **end for**

Algoritmo 3.1 – Pseudocódigo genérico para AGs.

3.1 Representação do Indivíduo

AGs são usualmente representados por um *array* de caracteres. Essa representação foi adotada inicialmente na forma binária por Holland (?), sendo atualmente utilizada devido sua facilidade de implementação. Além disso, é bastante simples atualizar dados binários através de operadores genéticos como mutação, cruzamento e similares (Ric08). Na terminologia de AGs, um *array* de bits é considerado como um indivíduo ou cromossomo. Os cromossomos são compostos por unidades de bits chamados genes. Os genes são quaisquer características do ambiente que deseja-se otimizar. Eles devem representar informações importantes sobre o problema em questão.

É interessante que as informações utilizadas para representar o indivíduo sejam concisas, uma vez que elas irão orquestrar o processo de otimização. Grandes indivíduos geram espaços de busca maiores, o que pode contribuir para a ineficiência do Algoritmo Genético na busca das melhores soluções. Portanto, é aconselhável que o indivíduo seja representado de maneira objetiva para codificar as informações não redundantes mais importantes relativas ao problema que se almeja resolver.

3.2 População

O AG trabalha com uma coleção de indivíduos, denominada população. Primeiramente, é necessário definir o tamanho da população, que é uma grande discussão na área da ciência da computação. Alguns pesquisadores defendem que uma população com poucos indivíduos pode ocasionar em uma procura ineficiente (DGH07), já que o espaço de busca não será adequadamente explorado. Por outro lado, uma população muito grande levará muito tempo para uma convergência da solução. A inicialização da população geralmente é aleatória. Utilizar a técnica de aleatoriedade na geração da população inicial proporciona diversidade, garantindo que o espaço de busca seja adequadamente explorado, especialmente nas fases iniciais do processo de otimização. Uma baixa diversidade na inicialização da população gera indivíduos com características muito semelhantes, podendo resultar na rápida convergência do algoritmo em uma solução ótima local.

Uma vez gerada a população inicial, é necessário realizar uma avaliação de cada indivíduo, com o intuito de saber o quanto apto é para uma possível solução. A geração da próxima população é conduzida pelos operadores genéticos e através dos valores obtidos com a função de avaliação. A motivação para isso é atingir soluções melhores a cada nova população gerada. As soluções selecionadas para formar novos indivíduos, são chamados de descendentes. Quanto mais apto um indivíduo é, mais chances ele tem de ser utilizado para posterior reprodução. À medida que o algoritmo evolui, a população obtém soluções mais aptas e, eventualmente, converge. É possível ainda transferir um número de indivíduos da população atual que alcançaram o melhor valor na função de avaliação para a próxima geração da população, com o intuito de manter os melhores indivíduos nas próximas gerações. Técnica conhecida como Elitismo (?).

3.3 Função de Avaliação

A função de avaliação é uma forma de validar a qualidade de cada indivíduo na tentativa de resolver um problema. A implementação dessa função deve ser capaz de diferenciar soluções boas e ruins, assim como, aquelas sub-ótimas, tendo ciência de quais estão mais próximas da solução desejada (Ric08). Em geral, uma função de avaliação tenta maximizar ou minimizar um determinado objetivo de forma numérica. Para o problema de otimização de um SGBD, por exemplo, alguém poderia querer encontrar a melhor solução (ou uma aproximação dessa) para o tempo de respostas em consultas aplicadas rotineiramente por períodos de tempo. Nesse caso, a função de avaliação é mono-objetiva, pois, usa um único critério para guiar a busca de soluções ótimas. Em contrapartida, poderia se estabelecer também uma relação mais complexa de avaliação, analisando o custo-benefício entre desempenho no tempo de resposta e alocação de recursos do sistema. Para estes casos, onde se deseja otimizar mais de um critério, a função de avaliação é denominada multi-objetiva.

Uma solução multi-objetivo perfeita é quase impossível, já que é inevitável o prejuízo de um dos objetivos enquanto tenta-se otimizar outros. Para esses casos, os objetivos são definidos para balancear algum *tradeoff* causado pela otimização de um único objetivo. Com isso, os conceitos de optimalidade são significativamente alterados, pois, uma única solução não pode ser ótima para todos os objetivos simultaneamente. Para esses casos, o processo de otimização resulta em uma seleção de respostas, e não mais uma única solução, conhecida como solução ótima ou ótimo global. O conjunto de respostas que representam as melhores soluções, considerando todos os objetivos que se deseja otimizar, é denominado Fronteiras de Pareto (CPL04).

Em uma otimização multiobjetivo, a qualidade da resposta é verificada a partir do critério de dominância, que pode ser definido da seguinte forma: para um problema com duas soluções viáveis P e Q , onde o valor em um dos objetivos de P é melhor que o mesmo objetivo em Q e ainda, o restante dos valores dos objetivos de P não podem ser piores que o restante dos mesmos valores nos objetivos em Q pode-se dizer então que a solução P domina a solução Q . Ou seja, a solução P não pode possuir nenhum objetivo com menos qualidade do que a solução Q e a solução P é melhor que a solução Q em ao menos um objetivo (DPAM02). A Figura 3.2 ilustra as regiões de dominância de uma determinada solução P , para um problema de otimização multiobjetivo onde qualquer solução presente em algumas dessas regiões estará presente no quadrante definido como Dominada P , Dominada por P ou em condição de igualdade com relação a P , estando nesse caso, presente no quadrante Equivalente a P .

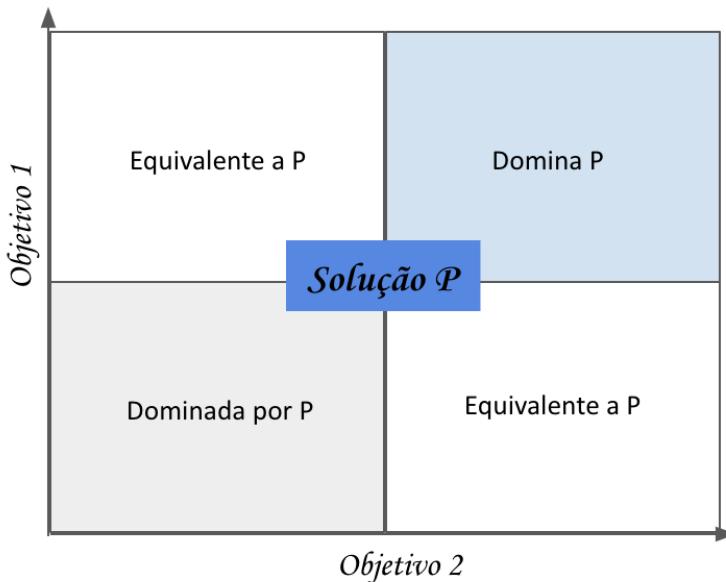


Figura 3.2 – Exemplo de critério de dominância para uma função de otimização multiobjetiva, baseado em (?).

Existem várias técnicas de otimização que tentam encontrar um conjunto que apresenta as melhores soluções para um dado problema multiobjetivo (LLHL14). Um deles, considerado referência, é conhecido como *Non-dominated Sorting Genetic Algorithm-II* (NSGA-II) (DPAM02), que está descrito na subseção seguinte.

3.3.1 Non-dominated Sorting Genetic Algorithm (NSGA-II)

A técnica NSGA-II aplica o critério de dominância sobre os indivíduos de uma população classificando-os em fronteiras. Quando um indivíduo da população não possui outro que o domine, é dito que ele está na primeira fronteira. Já aqueles dominados por apenas um indivíduo são classificados como pertencentes a segunda fronteira, seguindo assim até que se classifique toda a população, ou seja, até que a última fronteira seja formada. A cada geração, os indivíduos classificados na primeira fronteira são os melhores, ou dominantes, enquanto os piores estão classificados nas últimas fronteiras. Após os indivíduos estarem classificados em fronteiras, eles são ordenados também através do algoritmo *Crowding Distance* (DPAM02). Esse algoritmo realiza uma classificação através dos valores das funções objetivo de cada solução viável. Cada indivíduo é ordenado em relação aos demais integrantes de uma mesma fronteira, por intermédio de um índice que relaciona a distância dos vizinhos imediatos com a distância entre as soluções extremas dessa mesma fronteira.

3.4 Critério de Parada

Considera-se como critério de parada em um AG as seguintes abordagens: (I) Número máximo de gerações, ocorre quando o algoritmo atinge um número específico de gerações; (II) Tempo decorrido, ocorre quando o algoritmo atinge um determinado tempo de processamento; (III) Nenhuma mudança de resultados, ocorre quando não se atinge valores melhores na função de avaliação por um número específico de gerações; (IV) Tempo de melhora, ocorre caso não haja nenhuma mudança nos valores obtidos com a função de avaliação durante um determinado período de tempo; e (V) Solução satisfatória, nesse caso se assume que o melhor indivíduo tenha sido encontrado.

3.5 Seleção

O operador de seleção escolhe indivíduos dentro da população atual para reprodução de acordo com os valores obtidos na função de avaliação. De acordo com a teoria da evolução de Darwin, os melhores indivíduos devem sobreviver para gerar descendentes (?). A questão é escolher uma forma de selecionar os melhores indivíduos preservando a diversidade da população. A longo prazo, o indivíduo que carrega a combinação correta em seus genes podem se tornar dominante na população. Nesse tipo de método, usualmente indivíduos com melhor valor de avaliação são privilegiados, sem desprezar totalmente aqueles com menor valor. Dessa forma, os indivíduos com função de avaliação extremamente baixa também podem ter características genéticas que sejam favoráveis à criação de um indivíduo que seja a melhor solução para o problema em questão (Ric08).

Os operadores de seleção são caracterizados por um parâmetro conhecido como pressão seletiva, que está diretamente relacionado ao tempo de dominância do operador. O tempo de dominância, é a velocidade que a melhor solução da população inicial leva para dominar toda o resto da população através da aplicação isolada do operador de seleção (?). Se o tempo de dominância de um operador é grande, então a pressão seletiva é fraca, e vice-versa. Portanto, a pressão seletiva mostra o quanto “guloso” é o operador de seleção no que se refere à dominância de um indivíduo da população. Se o operador de seleção apresenta uma forte pressão seletiva, então a população perde diversidade rapidamente.

Existem dois tipos de seleção (SD08): proporcional e baseada em ordem. A seleção proporcional escolhe os indivíduos com base no valor de suas funções de avaliação em relação a outros indivíduos da população. Para esse caso, as chances de um indivíduo ser selecionado são proporcionais ao seu valor de avaliação. A seleção baseada em ordem escolhe os indivíduos não apenas se baseando no valor bruto da função de avaliação, mas também a partir do ranqueamento do indivíduo dentro da população. Ou seja, a diferença entre as posições no ranqueamento de indivíduos é mais relevante do que a margem da diferença dos valores brutos da avaliação. As próximas subseções, apresentam três tradicionais métodos de seleção que envolvem seleção proporcional e seleção baseada em ordem.

3.5.1 Roleta Viciada

O AG original desenvolvido por Holland aplicou a seleção proporcional, de forma que o valor esperado de um indivíduo é dividido pelo valor médio de avaliação da população. Esse método é conhecido como Roleta Viciada. Basicamente, essa técnica funciona da seguinte forma: cada indivíduo recebe uma fatia de uma roda circular virtual, proporcional ao valor de avaliação obtido por cada indivíduo. Quanto melhor o valor de avaliação do indivíduo, mais chances ele terá de ser selecionado. Para selecionar k indivíduos, a roleta é girada k vezes (onde k é o número de indivíduos definido previamente para aplicar o método de seleção). A cada giro da roleta, um número x é obtido aleatoriamente. Esse número x , deve estar contido no intervalo de números possíveis para cada fatia da roleta. Na roleta ilustrada pela Figura 3.3, por exemplo, há 40 números possíveis na fatia que representa o melhor indivíduo. Isto, em termos probabilísticos, significa cerca de 90% de chances de ser selecionado (44 / 90).

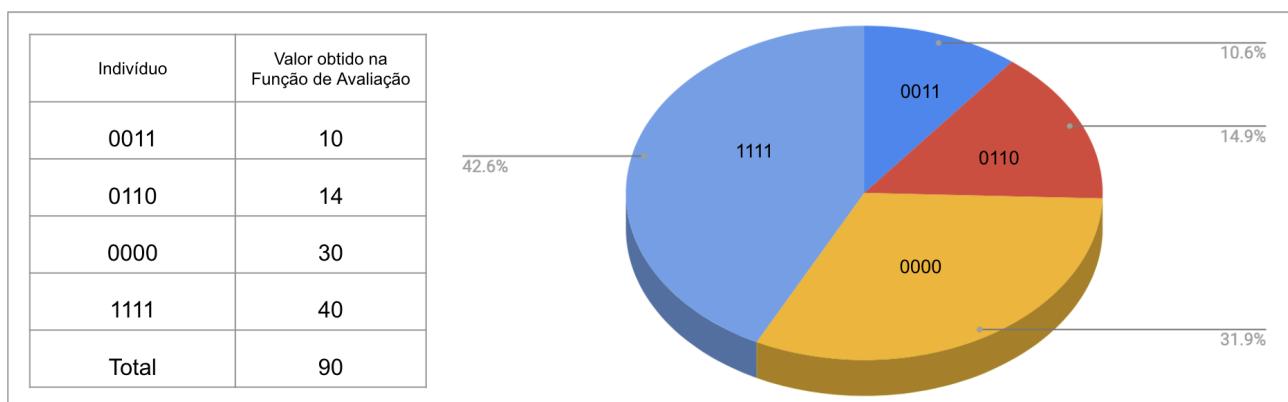


Figura 3.3 – Método de Seleção da roleta viciada.

3.5.2 Ranqueamento

O método de seleção por ranqueamento é baseado em ordem. Os indivíduos da população primeiramente são ordenados de acordo com o seu respectivo valor de avaliação. Assim, as chances de seleção de cada indivíduo dependem mais de sua posição no ranqueamento do que o seu valor absoluto obtido na função de avaliação. Em seguida, determina-se a probabilidade de seleção de cada indivíduo. O desempenho desse método de seleção depende da função de mapeamento de distribuição de probabilidades.

3.5.3 Torneio

A técnica de torneio seleciona de forma aleatória um conjunto de indivíduos da população atual e em seguida realiza uma competição entre os indivíduos contidos no conjunto selecionado. Para tanto, é necessário definir um valor de k , que representa o tamanho do torneio, indicando quantos indivíduos participarão do processo. Vence aquele que apresentar o maior valor obtido na função de avaliação, comparado ao de seus oponentes. Após o término da rodada, todos os indivíduos, inclusive o vencedor, retornam à população, podendo ser selecionados novamente para um novo torneio. Este processo se repete até que o número definido como tamanho do torneio seja atingido. A técnica de Torneio é ilustrada na Figura 3.4. Essa técnica de seleção permite ajustar a pressão seletiva.

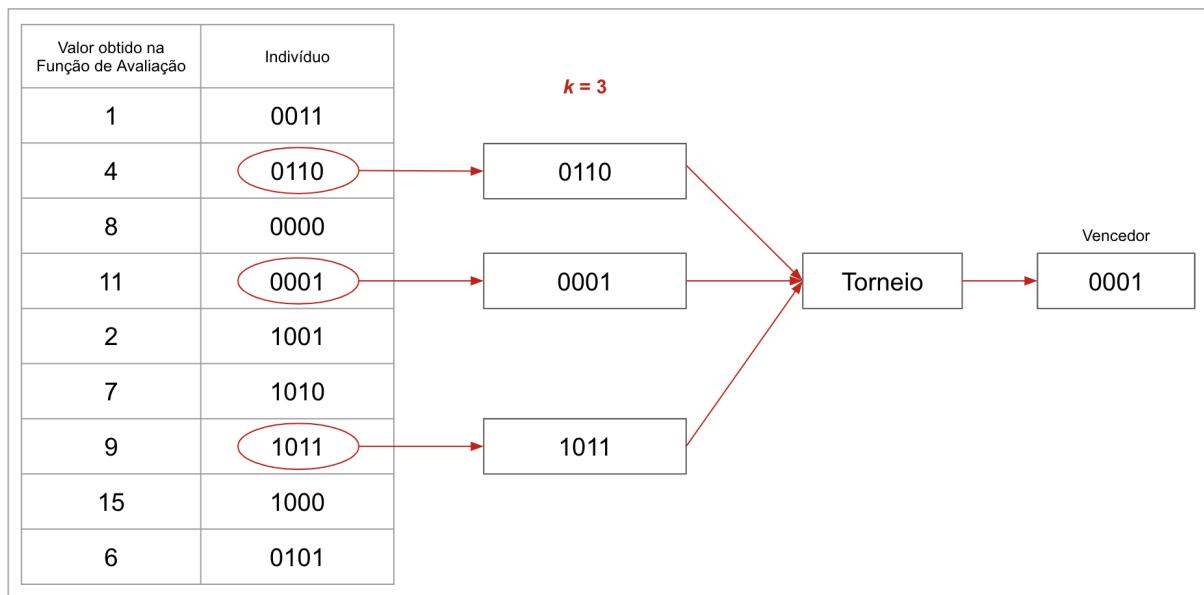


Figura 3.4 – Método de Seleção por Torneio.

3.6 Cruzamento e Mutação

O operador genético de cruzamento é responsável pela troca de informações entre dois indivíduos, chamados de pais. O principal objetivo desse operador é possibilitar que os genes de dois indivíduos possam gerar um indivíduo que obtenha um valor melhor na função de avaliação. Tanto o cruzamento quanto a mutação são aplicados sob alguma probabilidade previamente definida. Esse valor de probabilidade controla a frequência da aplicação do operador de cruzamento nos indivíduos.

Valores de cruzamento mais altos levam à exploração em profundidade dos indivíduos de uma população, mas restringem a exploração do espaço de busca. Existem três tipos principais de operadores de cruzamento (SDJ91): *one-point-crossover*, *multi-point-crossover* e *uniform-crossover*. Na abordagem *one-point-crossover* um ponto de cruzamento aleatório é selecionado e indivíduos trocam as partes selecionadas entre sim, formando um novo indivíduo. O *multi-point-crossover* é uma generalização da abordagem anterior, sendo o número de pontos de cruzamento a única diferença entre essas duas técnicas. Na abordagem *uniform-crossover* cada gene do indivíduo é tratado separadamente. Funciona basicamente escolhendo através de probabilidades se o gene escolhido será ou não utilizado para formar o novo indivíduo.

Entretanto, algumas vezes os pais escolhidos para cruzamento podem ter o mesmo bit em seus genes, desfavorecendo a diversidade na geração de novos indivíduos. O operador de mutação é projetado para contornar esse problema. Sendo assim, o próximo passo é executar o operador de mutação. Nesta fase, é preciso definir a probabilidade de aplicar esse operador em cada indivíduo. Além de um valor de probabilidade é necessário definir a taxa de mutação, que se refere ao número de genes que serão alterados. Usualmente, a taxa de mutação é bem pequena e depende do número de genes definidos para comporem o tamanho de cada indivíduo.

3.7 Considerações do Capítulo

A computação evolutiva é uma das sub-áreas da computação aplicada em problemas de busca e otimização. Seus princípios bioinspirados tem sido bastante utilizados nos últimos anos. Um dos motivos para o crescimento de sua aplicação deve-se principalmente a um vasto conjunto de técnicas que propiciam uma intensiva exploração de soluções para um dado problema altamente complexo. Nesse capítulo, apresentamos uma breve revisão sobre algoritmos genéticos, abordando métodos que são essenciais para compreensão da estratégia de exploração de soluções proposta neste trabalho.

4. UMA ESTRATÉGIA BASEADA EM ALGORITMOS GENÉTICOS COM INDEXAÇÃO DINÂMICA DE DADOS PARA OTIMIZAÇÃO DE BANCOS DE DADOS RELACIONAIS

Neste capítulo será apresentada a estratégia proposta para indexação dinâmica de dados com o intuito de otimizar padrões de consultas aplicadas a bancos de dados relacionais. Cada seção do presente capítulo segue o fluxo previamente retratado na Figura 4.1. As próximas seções apresentam maiores detalhes da estratégia proposta.

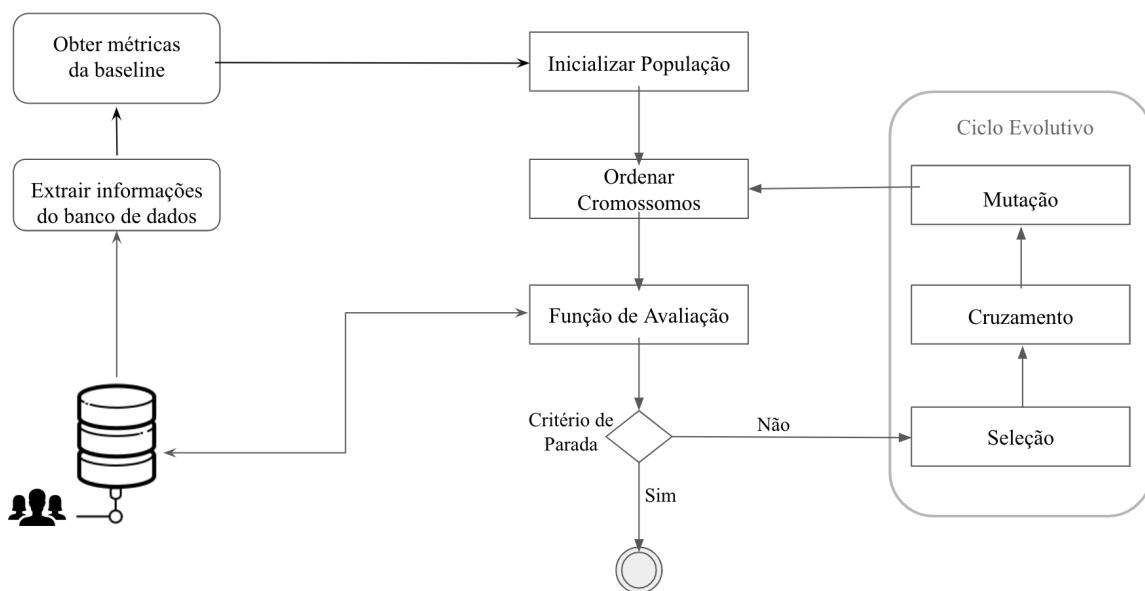


Figura 4.1 – Arquitetura da estratégia utilizada para implementação do algoritmo genético para ambientes de SGBDs.

4.1 Motivação

Conforme pode-se observar na Seção 1.1, o problema de desempenho de bancos de dados é um fator de relevância no quesito otimização. Embora os SGBDs mais clássicos sejam um pouco mais rápidos na recuperação de informações de um conjunto de dados estruturados, não são eficientes o bastante em relação ao desempenho no tempo de resposta, quando se trata de um grande volume de informações armazenadas. Para lidar com esse problema, diversas pesquisas têm focado em diferentes abordagens visando otimizar o desempenho dos processos em ambiente de bancos de dados, tanto em processamentos OLAP (BLC⁺16, DPP⁺18, PIM15), como em processamentos OLTP (*Online Transaction Processing*) (KIM⁺11, KD13, KNR13, LZZ13).

Com essa variedade de abordagens propostas para otimizar o desempenho de bancos de dados relacionais, observou-se a oportunidade de utilizar estruturas de índices com o intuito de buscar um desempenho satisfatório na realização de consultas sobre sistemas tradicionais de armazenamento. Um sistema ideal deveria ser capaz de utilizar a arquitetura de índices mais benéfica para um determinado conjunto de consultas que é aplicado de forma rotineira em um dado período de tempo. De forma paralela, também deve ser observado se um índice está muito tempo em desuso, para que o mesmo seja então deletado, liberando espaço para outros índices. Com base no que foi exposto, propõe-se uma estratégia baseada em AGs para explorar as possíveis combinações de índices com o intuito de auxiliar a tarefa de encontrar uma configuração ótima ou sub-ótima de índices. Com a finalidade de prover equilíbrio na busca pela melhor solução, considerou-se dois aspectos na otimização: tempo de resposta, e quantidade de memória utilizada para armazenamento dos índices. A seguir, são apresentados maiores detalhes sobre a solução proposta.

4.2 Visão Geral da Estratégia

Ao criar ou excluir índices, é necessário verificar qual combinação é a mais eficaz, dado um esquema de banco de dados, um conjunto de consultas e o padrão de execução das mesmas. A questão é encontrar a combinação que satisfaça ambos critérios, minimizar o tempo de resposta para um conjunto variado de consultas e prover equilíbrio com o espaço ocupado em memória. Utilizando o exemplo do esquema do TPC-H que contém 45 colunas, ditas como indexáveis, irá gerar 2^{45} combinações possíveis de índices, resultando em um grande espaço de busca. No entanto, com uma estratégia de exploração eficiente, é possível verificar uma quantidade de combinação de índices satisfatória, ao passo que essas possíveis soluções vão sendo testadas. A presente estratégia mapeia o cenário de banco de dados relacional que utiliza tabelas como estruturas de armazenamento sobre um AG, para explorar as possibilidades de combinação de configuração de índices com base em um conjunto de consultas. As definições da presente estratégia foram mapeadas para o contexto de banco de dados, conforme Tabela 4.1.

Definições de GA	Mapeamento para SGBDs
Gene	0 ausência de índice; 1 presença de índice
Cromossomo	conjunto de colunas do esquema codificados pelos genes
População	coleção composta pelo conjunto de colunas do esquema do SGBD
Pais	duas configurações de índices selecionados para troca de material genético
<i>Mating pool</i>	coleção de pais previamente selecionados
Função de Avaliação	uma função que diga quão bom é um determinado indivíduo da população
Mutação	operador genético que altera a ausência ou a presença de índices
Cruzamento	operador utilizado para troca de material genético entre dois vetores de bits
Elitismo	técnica utilizada para manter as melhorias configurações de índices

Tabela 4.1 – Mapeamento das definições de algoritmos genéticos no contexto de banco de dados relacional

4.3 Geração da População Inicial

Uma vez que o esquema do SGBD está previamente conhecido, é definido então a estrutura dos indivíduos. Conforme visto na Seção 3.1, utilizar uma estrutura binária como representação do indivíduo torna a implementação do algoritmo mais acessível e facilita a aplicação dos operadores genéticos. Por isso, utilizou-se esse tipo de estrutura para representar os indivíduos dentro da estratégia proposta. Os indivíduos são definidos então como um vetor binário composto pelo número de colunas que podem ser indexadas. Nessa estratégia, cada posição dentro do vetor indica se uma coluna deve ser indexada ou não.

Formalmente, o indivíduo é representado como um vetor binário \mathbf{x} , onde $|\mathbf{x}| = C$ e C é o número de todas as colunas em todo o esquema do banco de dados. Portanto, x_i denota duas possíveis ações para a coluna i^{th} : CREATE INDEX, se ainda não existir índice para a respectiva coluna, ou DROP INDEX, se essa ação for possível para aquela coluna. Para tanto, algumas restrições foram aplicadas na presente estratégia com o intuito de eliminar ações desnecessárias que gerariam sobrecarga de operações no sistema. Conforme visto no Capítulo 2, em alguns casos não é interessante utilizar índices, como em tabelas com poucos registros, por exemplo.

As estratégias utilizadas para elaboração das restrições foram: (i) considerar apenas as tabelas do esquema que contenham maior número de registros e (ii) desconsiderar todas as chaves primárias, uma vez que essas colunas já estão indexadas por padrão. Ao final, dentro do plano experimental, onde se utilizou o esquema de dados do TPC-H¹ (Transaction Processing Performance Council), obteve-se o total de 23 colunas disponíveis para indexar, sendo esse o tamanho de cada indivíduo. Consequentemente, tanto as chaves primárias quanto as estrangeiras não são afetadas por nenhum operador genético como cruzamento, mutação ou ação adicional.

¹TPC: <http://www.tpc.org/>

Dessa forma, cada indivíduo é composto por $C = 23$ genes mutáveis, gerando um espaço de busca que contém $2^{22} = 4,194,304$ como combinação possível de índices. A Figura 4.2 mostra um exemplo da representação de um indivíduo, dentro do plano experimental utilizado que se baseou no esquema do TPC-H.

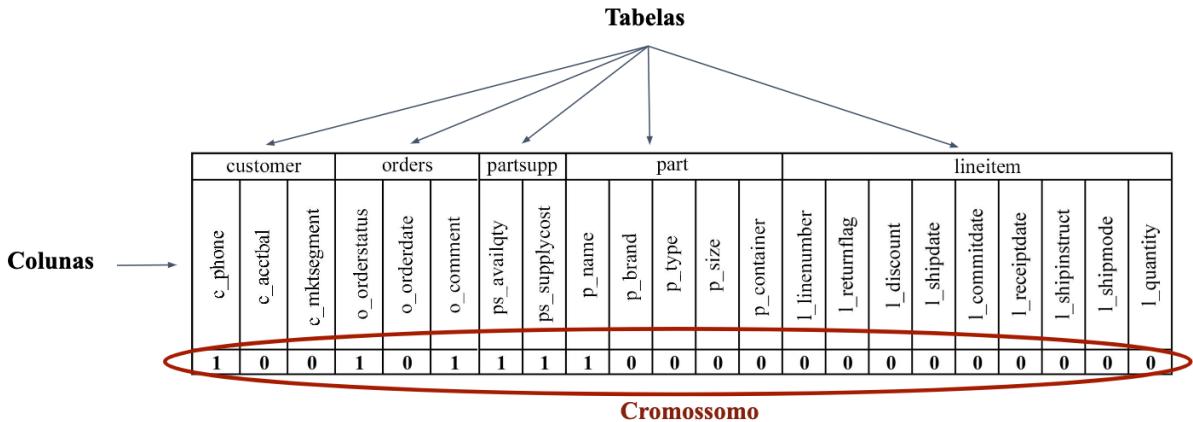


Figura 4.2 – Exemplo de codificação de um cromossomo baseado no esquema do TPC-H.

Para criação da população inicial foi utilizada a técnica de geração aleatória. A inicialização da população de forma aleatória visa intensificar a exploração das soluções disponíveis. Utilizar a técnica de geração aleatória da população inicial tende a conduzir a solução à otimização desejada (?). Adicionalmente, utilizar alguma heurística para gerar a população inicial pode fazer com que a população tenha soluções semelhantes e pouca diversidade, contribuindo para uma exploração menor do espaço de busca. Também foi aplicado um método para classificação dos cromossomos de acordo com o número de *bits* menos significativos. Ou seja, quanto mais *bits* 0 um indivíduo possui, mais prioridade ele tem de ser avaliado, haja visto que, haverá mais colunas sem índices do que o contrário. Este método foi aplicado com o objetivo de otimizar o tempo de requisições realizadas no banco de dados para os comandos de criação e exclusão de índices.

4.4 Função de Avaliação

Após criar a população inicial, o algoritmo implementa uma função multi-objetiva para avaliar a qualidade dos indivíduos, considerando dois aspectos: o tempo de resposta

sobre um conjunto de consultas e o espaço ocupado em disco pelos índices. Para isso, foram criadas duas funções para calcular a função de avaliação de cada objetivo. O objetivo 1 tenta minimizar o tempo de resposta de um determinado conjunto de consultas. Inicialmente se obtém o tempo de respostas do conjunto de consultas através do somatório dos tempos, conforme mostrado na Fórmula 4.2.

O valor retornado representa o tempo de resposta do conjunto de consultas submetido ao sistema quando o indivíduo em avaliação é aplicado ao SGBD. Tendo os tempos de resposta, aplica-se então a Fórmula 4.1, implementada para comparar o tempo de resposta do indivíduo atual com o que foi utilizado como base de comparação. Quanto maior a razão entre o indivíduo utilizado como base de cálculo e o indivíduo atual, mais ganho na otimização do tempo de resposta será alcançado. Para isso, é necessário definir qual configuração de índices utilizar como base de comparação de forma que o algoritmo consiga encontrar soluções melhores. Como o melhor tempo de resposta conhecido se refere ao indivíduo que codifica todas as colunas indexadas, já que não se tem ciência da melhor configuração de índices até então, foi utilizado esse indivíduo como numerador na função objetivo 1.

$$\text{Objetivo 1} = \frac{\text{tempo_individuo_tudo_indexado}}{\text{tempo_individuo_atual}} \quad (4.1)$$

$$\text{tempo_consultas} = \sum_{q=1}^N t_q \quad (4.2)$$

O objetivo 2 visa minimizar o espaço ocupado em disco pelos índices. Dentro do atual cenário de otimização, o objetivo mais importante que se deseja aprimorar é o tempo de resposta. A Fórmula 4.4 foi elaborada com o intuito de ponderar o objetivo 2, para que os valores referentes ao tamanho dos índices não fossem tratados de forma direta. Essa decisão tem como finalidade fazer com que o algoritmo priorize a minimização do tempo de resposta em relação ao espaço ocupado. Optou-se por usar \log_{10} , devido os índices terem um percentual de amplitude grande na oscilação dos seus tamanhos. Foi adicionado 1 no tamanho dos índices para garantir que o resultado não seja indefinido, para \log_{10} . O próximo passo é aplicar a Fórmula 4.3, que segue a mesma lógica do objetivo 1. Todavia, para esse caso, considera-se como melhor indivíduo aquele que codifica a configuração de índices no estado inicial do banco, apenas chaves primárias como índices.

$$\text{Objetivo 2} = \frac{\text{tamanho_estado_inicial}}{\text{tamanho_individuo_atual}} \quad (4.3)$$

$$\text{escala_tamanho} = \frac{1}{\log_{10} * (\text{tamanho_indices} + 1)} \quad (4.4)$$

Em problemas mono-objetivo, o algoritmo genético deve considerar apenas um valor de avaliação para cada solução na população. Esse valor indica como cada solução satisfaz o objetivo único. No entanto, em problemas multiobjetivos, cada solução possui valores de acordo com o número de objetivos. Isso pode ser um problema na hora de julgar quais os melhores indivíduos. Conforme visto na Seção 3.3.1 existem algumas metodologias comumente utilizadas para resolver essa questão da seleção de melhores indivíduos em problemas multiobjetivos. Na presente estratégia foi utilizado o NSGA-II, metodologia considerada referência no âmbito de pesquisas (SD94).

4.5 Operadores Genéticos

O próximo passo do algoritmo é selecionar indivíduos para serem pais e, em seguida, trocar material genético para produzir a nova população através do operador de cruzamento. Para os casos dos quais o referido operador não for aplicado, realiza-se uma cópia de ambos indivíduos. Uma vez tendo a população a ser definida para a próxima geração já preenchida, desconsiderando-se o(s) indivíduos do elitismo, a estrutura é percorrida, considerando cada um de seus indivíduos para a possível aplicação do operador de mutação. A mutação é aplicada em alguns indivíduos escolhidos através de uma probabilidade definida inicialmente. Observe que os indivíduos selecionados com o elitismo não são modificados com os operadores do GA. No estágio final do ciclo evolutivo, a nova população substituirá a mais antiga. As próximas subseções descrevem detalhadamente os operadores de seleção, cruzamento e mutação.

4.5.1 Seleção

O método de seleção da Roleta Viciada tem um problema quando o valor da função de avaliação possui uma diferença muito grande de um indivíduo para outro. Por exemplo, se um indivíduo ocupa 90% da roleta virtual, então os demais indivíduos terão poucas chances de serem selecionados. Diferentemente da Roleta Viciada, o método de seleção por ranqueamento evita dar probabilidade maior de seleção para um grupo pequeno de indivíduos com alta avaliação, reduzindo, assim, a pressão da seleção quando a variação do valor da avaliação é muito elevada.

A seleção utilizada foi a tradicional técnica de Torneio. Nessa abordagem, k indivíduos são escolhidos aleatoriamente para disputar uma vaga no repositório de pais, conhecido como *mating pool*. Para execução do experimento foram utilizados 5 indivíduos como valor de k para disputarem entre si a vaga de pais (Mit96). As informações sobre os pais são armazenadas até que a população seja preenchida. O algoritmo também implementa o Elitismo para preservar as melhores soluções na próxima geração. Como não se tem um consenso sobre o melhor valor para esse operador, definiu-se 1% de n_e , com base em algumas pesquisas que buscam a otimização de hiper-parâmetros no elitismo (IM96, KCS06, GCJ17).

4.5.2 cruzamento

O operador de *cruzamento* da presente estratégia consiste em alterar partes dos indivíduos, escolhidas de forma aleatória. Neste operador, foi utilizada a técnica *two-point-crossover*, onde dois pontos aleatórios de cruzamento são selecionados e em seguida, trocados entre os pais para produzir os descendentes para a próxima geração, mantendo o tamanho original de cada cromossomo. O processo desse operador está ilustrado na Figura 4.3. Tipicamente, em AG são usados números moderadamente altos para probabilidade de cruzamento, variando entre $50\% \leq n_{cxpb} \leq 80\%$ (Mit96). Portanto, o valor de probabilidade estabelecido para n_{cxpb} foi de 65% (DPAM02).



Figura 4.3 – Exemplo da aplicação do operador de cruzamento baseado no esquema do TPC-H.

4.5.3 Mutação

A mutação é responsável por mudar aleatoriamente cada novo descendente. Em indivíduos codificados de forma binária, basta alterar os bits escolhidos de 0 para 1, ou vice-versa. O operador de mutação requer dois parâmetros: (I) a probabilidade de aplicar a mutação; e (II) a proporção de genes que serão alterados. Optou-se por definir o valor de n_{mutpb} em 20% de probabilidade de aplicar o operador genético ou não. Para n_{mutrt} foi considerado 0.001% por gene, conforme (Mit96). Tendo em vista que os indivíduos são compostos por 23 genes, foi definido 2.3% para essa probabilidade. A técnica utilizada foi a multi-flip bit, que seleciona aleatoriamente dois ou mais bits e em seguida verifica através da probabilidade se irá invertê-los ou não. Essa técnica é mais comumente usado para AGs codificados em vetores binários (SGK05). A aplicação do operador de mutação está ilustrada na Figura 4.4. Por fim, a população atual é substituída pelos novos indivíduos em conjunto com aqueles selecionados pelo elitismo. Esse procedimento é repetido de acordo com o número de gerações previamente definidas.

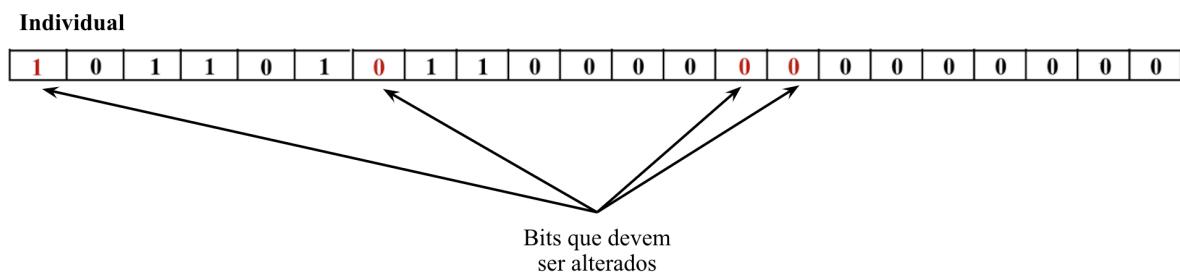


Figura 4.4 – Exemplo de aplicação do operador de mutação baseado no esquema TPC-H.

4.6 Critério de Parada

O critério de parada tem comportamentos diferentes em diversos AEs, não sendo possível formular uma regra geral que proporcione um resultado ótimo (JPW01). Entretanto, optou-se por utilizar alguma abordagem para o critério de parada para os casos em que o algoritmo atinja um ótimo local deixando de explorar novos espaços de busca, e tendo esse ótimo local como solução final para o problema de otimização. Como critério de parada optou-se pela abordagem k -iterações (BMP12, LW01), onde o algoritmo finaliza de forma precoce sua execução, caso não haja melhoria nos valores obtidos com a função de avaliação após k iterações consecutivas. Como valor de k foi definido 10% de n_{ger} por acreditar ser um bom valor, uma vez que não está claro qual o número de gerações que deve-se considerar como padrão na não melhoria dos resultados.

4.7 Considerações do Capítulo

Nesse capítulo foi apresentada uma estratégia de otimização de consultas para bancos de dados relacionais através da implementação de algoritmos genéticos. A presente estratégia pode ser facilmente empregada em qualquer ambiente de banco de dados relacional. O SGBD a ser utilizado será explorado para a obtenção das informações necessárias do esquema com a finalidade de aplicar as devidas restrições com base nas chaves primárias das tabelas e também desprezando tabelas com poucos registros. Além disso, qualquer conjunto de consultas pode ser aplicado na execução do modelo. Optou-se por utilizar AG devido ser uma das técnicas mais utilizadas em problemas categorizados como NP-Difícil, onde necessita-se explorar o espaço de busca de forma intensiva, e realizar essa tarefa de forma exaustiva é computacionalmente oneroso.

Conforme visto nesse capítulo, a função de avaliação em AG irá comandar o comportamento da exploração, tendo em vista que, através dela que se verifica a qualidade de um indivíduo. Foi escolhido utilizar a relação de razão entre dois indivíduos na função de avaliação dos objetivos por entender que esse método de comparação é bem simples e claro. E ainda, como a estratégia proposta trabalha sobre ambientes que, a princípio, não teria como se predizer o desempenho de índices ainda não aplicados ao banco de dados, julgou-se a relação de razão mais comprehensível para a função de avaliação. Na presente estratégia, priorizou-se o tempo de resposta, contudo, o algoritmo pode ser calibrado de forma a ponderar os pesos de cada objetivo, dando mais importância para memória ocupada, se assim for necessário. É possível também utilizar outros modelos de função de avaliação, ficando em aberto o critério da fórmula a ser utilizada para o(s) objetivo(s) que se tem necessidade de otimizar.

5. ANÁLISE EXPERIMENTAL

Nesta seção, apresenta-se a análise experimental conduzida para avaliação do desempenho da estratégia proposta. Informações inerentes ao *benchmark* utilizado nos testes, assim como o esquema, serão expostas com maiores detalhes. Em seguida, descreve-se o método de avaliação utilizado para validar os resultados e o plano de execução dos testes. Ao final desta seção, será exposto o comparativo realizado entre os resultados obtidos e outros métodos que também propuseram otimização de desempenho de SGBDs utilizando estruturas de índices.

5.1 TPC-H Benchmark

O TPC-H *benchmark* simula um ambiente com características OLAP. Consiste em um esquema de banco de dados, um conjunto de tabelas orientadas a um modelo comercial de negócios, além de modificações simultâneas de dados. Essas modificações simulam sistemas de suporte à decisão que usam um conjunto de dados massivo. As consultas provenientes desse *benchmark* são consideradas de alto grau de complexidade e fornecem vários aspectos da capacidade do sistema de processar consultas. Os testes fornecidos pelo *benchmark* foram formulados para medir aspectos da capacidade de desempenho do sistema. Além disso, a complexidade do processamento de consultas simula ambientes único e multi-usuários (SSSF09).

Optou-se por utilizar o TPC-H *benchmark* por ser largamente empregado sobre sistemas com a finalidade de fundamentar sua agilidade. Outrossim, o TPC-H está sendo bastante aplicado em pesquisas mais recentes, por prover todo o ferramental necessário na execução dos testes, que são totalmente preparados para medir o desempenho de sistemas com grande volume de dados. Outro aspecto importante, é a disponibilidade de variados tamanhos de esquema. As tabelas podem ter tamanhos diferentes e são preenchidas através de um gerador de dados fornecido pelo pacote TPC-H. As consultas acesam uma grande porcentagem dos dados, proporcionando intensa atividade de disco e de *Central Process Unit* (CPU). Adicionalmente, as consultas submetidas para testes são consideradas *ad-hoc* realistas (TCG12), voltadas para sistemas que não envolvem decisões recorrentes. Na subseção seguinte encontra-se o esquema do TPC-H *benchmark*.

5.1.1 Esquema de dados do TPC-H

Para a realização da análise empírica, foi verificado o desempenho do algoritmo usando o esquema de dados fornecido pelo TPC-H. O esquema representa um armazém de dados simples que lida com vendas, clientes e fornecedores. É composto por 8 tabelas e 65 colunas. Para o presente experimento foi utilizado o esquema de dados com 1Gb de tamanho. O esquema TPC-H está ilustrado na Figura 5.1.

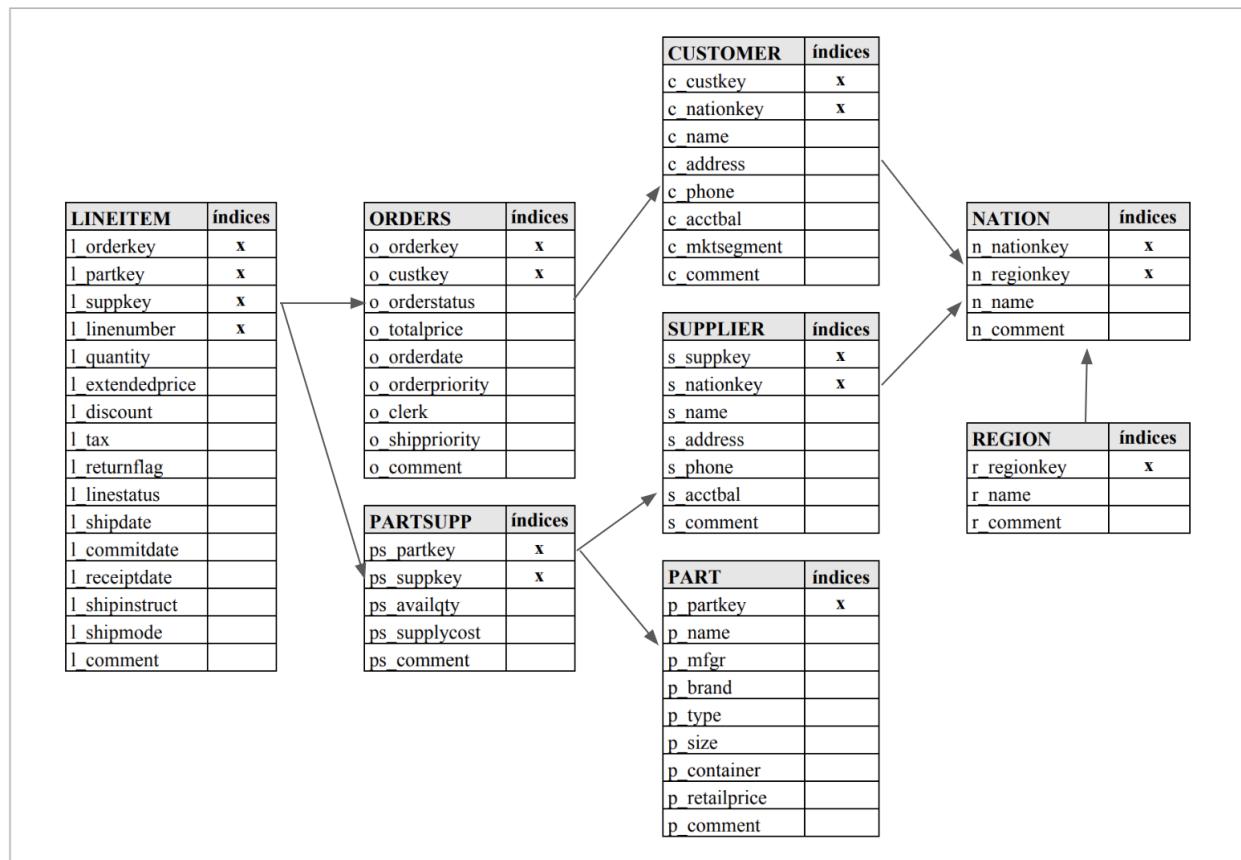


Figura 5.1 – Esquema do TPC-H no estado inicial de configuração de índices (apenas PK's e FK's).

Escolher quais índices usar não é uma tarefa simples, especialmente quando envolve um grande número de possibilidades. Por exemplo, o esquema do TPC-H contém 61 colunas e se alguma abordagem de exploração considerar todas elas então haverá 2^{61} combinações possíveis. No entanto, é comum que algumas colunas já estejam indexadas devido à presença de chaves primárias. No caso do esquema do TPC-H, 20 das 61 colunas já possuem chaves primárias em sua configuração padrão. O algoritmo desconsidera tais colunas por entender que não há a necessidade de otimizá-las, haja visto que, as chaves já possuem índices. A figura 5.1 mostra a configuração inicial do esquema do TPC-H, com as devidas relações de chaves primárias e estrangeiras.

A análise seguinte foi realizada sobre o tamanho das tabelas do esquema. Se uma tabela possui poucos registros, então a presença de índices pode não ser benéfica no tempo de resposta das submissões de consulta, pois, na maioria das vezes é mais rápido acessar os dados sequencialmente do que pelos índices. A Tabela 5.1 mostra o tamanho de cada tabela do esquema utilizado no plano experimental. Dessa forma, foram consideradas as seguintes tabelas: LINEITEM, ORDERS, PARTSUPP, PART e CUSTOMER, por serem as maiores dentro do esquema. No final, foi assumido que as colunas disponíveis para se tornarem índices são aquelas que não possuem chaves primárias e as que compõem mais de 2% do esquema. Com isso, ao final obteve-se 23 colunas indexáveis, sendo esse o tamanho de cada indivíduo.

Tabela	Número de Colunas	Tamanho (MB)	Número de Registros
Lineitem	16	1.252	4.423.659
Orders	9	265	1.500.000
Partsupp	5	167	800.000
Part	9	33.5	200.000
Customer	8	33	150.000
Supplier	7	2.8	10.000
Nation	4	< 1	25
Region	3	< 1	5

Tabela 5.1 – Informações das tabelas do esquema do TPC-H .

5.2 Plano Experimental

O objetivo do presente experimento é mostrar que a estratégia proposta é promissora e eficiente na sugestão de índices que beneficiem o desempenho de um SGBD ao realizar um determinado conjunto de consultas. Ou seja, deseja-se mostrar que através deste trabalho consegue-se chegar em uma solução sub-ótima que promova o equilíbrio entre o desempenho de um conjunto de consultas submetido a um sistema e o espaço ocupado em memória de disco pelos índices. Para isso, foi utilizado o TPC-H *benchmark* no plano experimental, haja visto que, através dele tem-se o ferramental necessário para a simulação de um ambiente real com variedades de tamanhos de esquemas, consultas predefinidas e testes de performance elaborados de forma robusta. Para comparação dos resultados obtidos, foram utilizados os critérios mostrados na Tabela 5.2. Todos os resultados obtidos estão detalhados na Subseção 5.3.2.

Métodos	Descrição
Todos os índices	Todas as colunas indexadas
Otimizador #1	Otimizador de consultas do PostgreSQL
Otimizador #2	Otimizador de consultas do EDB Server
Busca aleatória	Explorar o espaço de busca sem parâmetros
Configuração inicial	Apenas chaves primárias como índices
ITLCS (PNR18)	Pesquisa similar a presente estratégia

Tabela 5.2 – Métodos de comparação utilizados.

A visão geral da implementação utilizada na presente estratégia pode ser consultada no Algoritmo 5.1. Para a execução do experimento, foram aplicados os parâmetros mostrados na Tabela 5.3. A escolha dos valores dos parâmetros é baseada na literatura, conforme descrito no Capítulo 4.

- 1: Conectar com o Banco de Dados
- 2: Extrair informações do esquema
- 3: Avaliar o indivíduo utilizado como base na Função de Avaliação
- 4: Inicializar população aleatoriamente
- 5: **for** indivíduo na população atual **do**
- 6: Aplicar os operadores genéticos
- 7: Ordenar os indivíduos
- 8: Calcular a Função de Avaliação
- 9: *individual* \leftarrow *fitness*
- 10: **for** Valor obtido na função de avaliação de cada indivíduo **do**
- 11: Verificar critério de parada
- 12: **end for**
- 13: **end for**

Algoritmo 5.1 – Algoritmo Geneético para indexar dados de forma dinâmica para SGBDs.

Parâmetros	Valores Utilizados
n_{pop}	50
n_{gen}	100
n_{mutpb}	20%
n_{mutrt}	2,3%
n_{cxpb}	65%
n_{elite}	$k=1$
Método de seleção	Torneio, $k=5$
Tipo de Função de Avaliação	NSGA-2
Tipo de mutação	Flip-bit
Tipo de cruzamento	Two-point-crossover
Critério de Parada	k -iterações, $k=10$

Tabela 5.3 – Parâmetros usados no experimento.

5.3 Avaliação do Método

Para análise e avaliação do método foram aplicados testes que utilizaram duas abordagens: (I) fazendo uso das métricas provenientes do TPC-H *benchmark* e, (II) fazendo uso do tempo de resposta das consultas e o tamanho dos índices. A primeira abordagem foi escolhida por simular um ambiente de SGBD bastante realístico, com operações de inserção e exclusão de dados, e aplicar testes de desempenho no SGBD através de métricas que avaliam um sistema usuário único e multi-usuário. Os teste de desempenho providos pelo *benchmark* foram executados para os critérios mostrados na Tabela 5.2. Para a segunda abordagem, executou-se o conjunto de consultas composto por 22 consultas com do tipo SELECT para obter o total do tempo de resposta, bem como a quantidade de memória ocupada pelos índices gerados.

5.3.1 Teste de Performance do TPC-H

O teste de desempenho do TPC-H segue um protocolo de execução composto por três equações: I) Equação 5.1: o POWER@SIZE, II) Equação 5.2: o THROUHPUT@SIZE e III) Equação 5.3: o QPHH@SIZE. Conforme ilustrado na Figura 5.2. Inicialmente, o teste executa operações de inserção de dados, depois disso o conjunto de 22 consultas é submetido, seguido de operações de exclusão de outros dados (diferentes dos que foram inseridos). A partir deste momento é obtida a métrica POWER@SIZE.

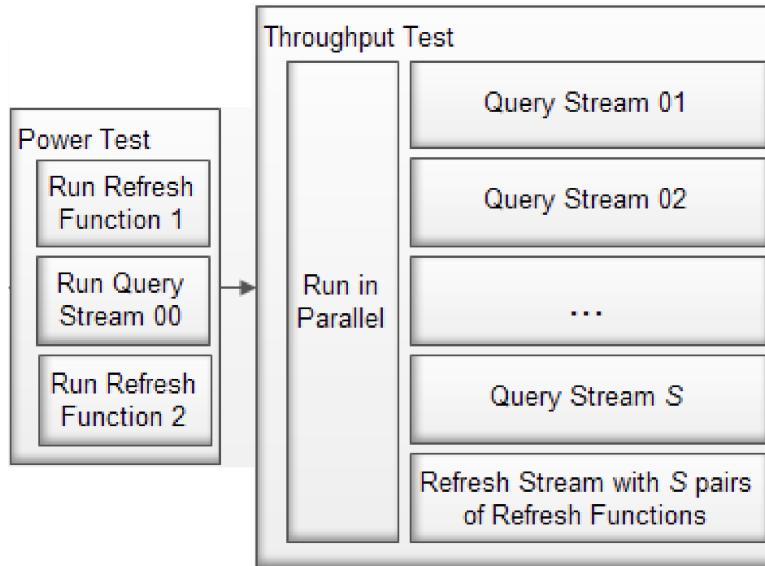


Figura 5.2 – Protocolo de execução do teste de performance do TPC-H (TCG12).

O próximo passo é executar as consultas em paralelo e em seguida novas operações de inserção de exclusão de dados. Com isso, tem-se o resultado da métrica THROUGHPUT@SIZE. A equação 5.1 comprehende a média geométrica, onde os números são multiplicados e em seguida, a raiz n^{th} do produto resultante é obtida. O numerador representa os segundos em uma hora, que em seguida é multiplicado pelo tamanho do banco de dados *Scale Factor* (SF). O conjunto de consultas é representado por $QI(i, 0)$ e as operações de inserção e exclusão por $RI(j, 0)$. A equação 5.2 mede quantas consultas foram executadas em paralelo no tempo decorrido, simulando um ambiente multi-usuário. O numerador $S \times 22$ é a quantidade de fluxos multiplicados pelo número de consultas submetidas. O denominador T_s representa o tempo total decorrido por fluxo. O resultado é multiplicado por 3600, que representa a quantidade de segundos em uma hora, e em seguida pelo tamanho do banco de dados. Finalmente, a equação 5.3, que tem por finalidade medir o desempenho das consultas por hora, é obtida através dos valores de POWER@SIZE e THROUGHPUT@SIZE, capturando o nível de desempenho geral do sistema.

$$\text{POWER@SIZE} = \frac{3600}{\sqrt[24]{\prod_{i=1}^{22} QI(i, 0) \times \prod_{j=1}^2 RI(j, 0)}} \times SF \quad (5.1)$$

$$\text{THROUGHPUT@SIZE} = \frac{S \times 22}{T_s} \times 3600 \times SF \quad (5.2)$$

$$\text{QPHH@SIZE} = \sqrt{\text{POWER@SIZE} \times \text{THROUGHPUT@SIZE}} \quad (5.3)$$

5.3.2 Avaliação dos Resultados

Nesta subseção, são relatados os resultados obtidos a partir de uma avaliação empírica envolvendo dados sintéticos provenientes do TPC-H *benchmark*. A Tabela 5.4 mostra os valores obtidos em todos os métodos escolhidos para comparação. O teste de desempenho do TPC-H foi executado 10 vezes para cada abordagem, sendo que para cálculo da média dessas execuções, foram desprezados os melhores e piores valores obtidos em cada método. Como o objetivo da presente estratégia é ser mais eficiente do que otimizadores de bancos de dados, utilizou-se como métodos de comparação duas ferramentas de otimização de SGBDs: (I) EnterpriseDB (EDB): uma ferramenta comercial de consultoria de banco de dados (EDB); e (II) PostgreSQL Workload Analyzer (POWA): uma ferramenta *open source* de otimização (POW). Além disso, optou-se por usar como base de comparação o trabalho realizado por Pedrozo et. al (PNR18), que sugere índices para uma única consulta por vez do conjunto de consultas fornecidos pelo TPC-H *benchmark*. Maiores detalhes sobre esse método estão descritos no Capítulo 6.

Métodos	QphH	Tempo de resposta (em segundos)	Memória em Disco (em Mb)
EDB	3044.05	60.59	755.18
POWA	3014.55	61.19	894.60
ITLCS	3136.11	59.35	758.83
Estado Inicial	1713.82	153.78	395.00
Tudo Indexado	3139.38	61.53	2050.00
Busca Aleatória	1690.67	150.09	1460.93
<u>AG para SGBDs</u>	3175.58	59.19	782.87

Tabela 5.4 – Resultados obtidos ao executar o teste de performance do TPC-H *benchmark* por cromossomo sugerido para cada método.

Os resultados obtidos com a busca aleatória mostram que os operadores genéticos são realmente úteis na exploração de estados não conhecidos. Os parâmetros usados no AG revelam que o algoritmo faz um caminho de exploração coerente, atingindo estados melhores à medida que novas gerações vão sendo formadas. Embora, a métrica referente o valor ocupado em memória de disco da presente estratégia não superou todos os métodos utilizados como base de comparação, conforme pode ser visto no comparativo ilustrado na Figura 5.3, o tempo de resposta assim como o valor de QPHH@SIZE mostraram-se bastante promissores, ficando apenas atrás do método *Index Tuning with Learning Classifier System*(ITLCS) (PNR18), o qual ressalta-se ter utilizado redes neurais em sua abordagem.

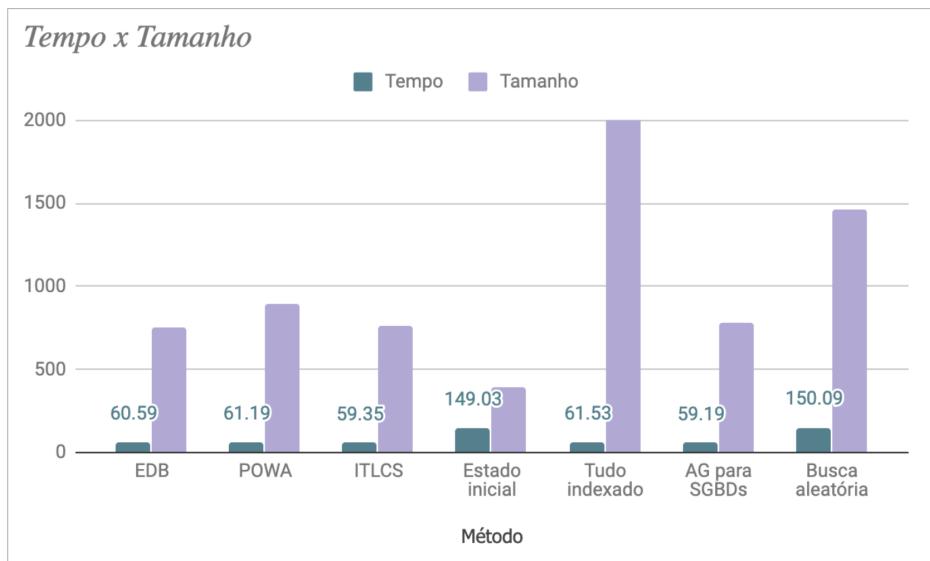


Figura 5.3 – Resultados obtidos sobre o tempo de execução e tamanho dos índices em Mb.

A Tabela 5.5 mostra os resultados do plano de execução de cada consulta submetida ao SGBD. Na abordagem com índices foi aplicado ao banco de dados o melhor indivíduo encontrado na presente estratégia. Na execução das consultas sem índices, foram consideradas apenas as chaves primárias, conforme estado inicial de configuração dos índices. O campo linhas verificadas exibe o número de linhas que cada consulta escaneou. Algumas obtiveram melhorias grandes, outras tiveram pioras significativas. Analisando o plano de execução de cada uma, constatou-se que o caminho escolhido para as consultas #5 e #8 foram diferentes, ainda que o SGBD utilizou somente os índices primários, e outras poucas vezes optou por fazer uma varredura em alguma das tabelas requisitadas. Em relação ao caminho escolhido para executar a consulta #7, observou-se que o SGBD realizou o plano de execução da consulta de forma igual para ambas abordagens (com e sem índices), mesmo estando presente a coluna **I_shipdate** na cláusula WHERE, um dos índices sugeridos pelo melhor indivíduo. A maior diferença dos valores retornados por essa consulta encontra-se no acesso à tabela **customer**, onde a chave de pesquisa é o índice primário.

Consultas	Custo da Consulta		Linhos Verificadas	
	Sem índices	Com índice	Sem índices	Com índice
#1	2.325.972	1.535.041	4.27M	639.60K
#2	67.138	66.929	1.78K	1.76K
#3	453.754	498.776	197.47K	218.22K
#4	474.044	238.648	1.48M	106.57K
#5	903.784	1.514.915	7.62K	28.70K
#6	918.853	900.438	4.27M	4.27M
#7	937.406	1.699.990	68.20K	295.43K
#8	1.040.560	2.591.275	15.23K	41.79
#9	2.744.949	2.827.821	450.54K	464.34K
#10	743.617	448.972	60.93K	44.24K
#11	99.054	97.848	82.37K	81.36K
#12	925.054	923.600	10.54K	10.53K
#13	591.991	557.320	2.37M	2.23M
#14	1.471.444	408.663	4.27M	157.18K
#15	223.099	215.474	47.47K	46.06K
#16	185.808	185.282	321.04K	317.53K
#17	90.392	18.418	40.55K	10.41K
#18	4.676.378	4.742.879	5.49M	5.80M
#19	106.318	91.560	357	275
#20	101.932	101.591	1.01K	1k
#21	502.983	498.761	10.45K	10.36K
#22	80.807	80.906	147.98K	148.16K

Tabela 5.5 – Resultado do plano de execução de cada consulta.

A Tabela 5.7 mostra quais índices estão sendo utilizados considerando o conjunto com 22 consultas do TPC-H. Apesar de ser constatado que dos índices sugeridos pelo presente trabalho, 3 não foram usados nas consultas, ainda sim houve uma otimização de desempenho provendo balanço entre uso de memória e tempo de resposta. Se o algoritmo for calibrado com tamanho da população maior, melhores resultados serão alcançados, tanto para o espaço ocupado em memória de disco quanto para o tempo de resposta das consultas, uma vez que esse parâmetro influencia na exploração do espaço de busca. Ademais, considerou-se como fator mais importante o tempo de resposta, porém, o algoritmo pode ser facilmente alterado para ter o mesmo peso ou peso maior na ponderação dos objetivos.

Consultas	Custo da Consulta		Linhas Verificadas	
	Estado Inicial	Melhor Indivíduo	Estado Inicial	Melhor Indivíduo
#1	2.325.972	1.535.041	4.27M	639.60K
#10	743.617	448.972	60.93K	44.24K
#14	1.471.444	408.663	4.27M	157.18K
#17	90.392	18.418	40.55K	10.41K
#19	106.318	91.560	357	275

Tabela 5.6 – Consultas mais beneficiadas.

Método	Colunas Indexadas	Índice foi usado?
EDB	c_mktsegment o_orderdate l_shipdate s_name	sim sim sim sim
POWA	c_mktsegment l_returnflag l_shipdate o_orderdate p_name p_type	sim não sim sim não sim
ITLCS	l_orderkey p_type o_orderdate ps_partkey l_shipdate	sim não sim sim sim
Ag para SGBDs	c_phone l_shipdate o_orderdate p_container ps_availqty	não sim sim não não
Busca Aleatória	c_mktsegment l_discount l_returnflag l_receiptdate l_shipinstruct l_shipmode p_name p_container ps_availqty ps_supplycost	não não não não não não não não não não

Tabela 5.7 – Uso dos índices em cada método.

5.4 Considerações do Capítulo

A estratégia proposta demonstra ser bastante promissora, pois, em comparação com os demais métodos superou os valores obtidos no resultado final. Optou-se utilizar as consultas predefinidas do *benchmark* como base de treino do algoritmo proporcionarem intensa atividade de entrada e saída de dados. Entretanto, qualquer conjunto de consultas pode ser aplicado à presente estratégia, ficando a critério da necessidade do sistema quais modelos de consulta utilizar. Foi utilizado o esquema de tamanho de 1Gb devido o tempo de execução do algoritmo, tendo-se como maior peso o critério da otimização de bancos relacionais que utilizem a estruturação de tabelas para armazenamento dos dados.

6. TRABALHO RELACIONADOS

Nos últimos anos, alguns trabalhos exploraram o uso do AG para melhorar o desempenho do banco de dados. A seguir, estão descritos os trabalhos mais similares a presente estratégia, contrapondo em quais aspectos se diferenciam.

6.1 Genetic Algorithm for Database Indexing

Korytkowski et al. (KGNS04) apresenta uma maneira automática de encontrar o melhor conjunto de índices para um SGBD, usando algoritmos genéticos. O esquema, os dados e as consultas foram implementados pelos autores. Sua estratégia considera aspectos como estatísticas de uso na implementação. Um indivíduo corresponde aos índices criados em uma única tabela, e possuem tamanhos diferentes. No total foram utilizadas 3 tabelas. O gene representa o número de colunas. A função de avaliação é baseada no tempo de resposta em operações de inserção e consulta. Os experimentos foram realizados sobre um esquema com 200Mb de tamanho. O aspecto referente ao tamanho dos índices foi tratado como uma exceção dentro do código, que dispara quando os índices atingem o valor máximo permitido pelo banco de dados.

6.2 ITLCS on Hybrid Storage Environments

Na estratégia An Adaptive Approach for Index Tuning with Learning Classifier Systems, Pedrozo et al. (PNR18) propõe uma arquitetura aplicada a ambientes de armazenamento híbrido usando um Algoritmo Genético para criar índices no SGBD em dois ambientes, para armazenamento em disco rígido e armazenamento híbrido. As consultas são utilizadas para codificação do indivíduo, de forma que somente as colunas contidas nas cláusulas WHERE são consideradas. Para avaliar a abordagem propostas, os autores utilizaram o *benchmark* TPC-H. Os testes para validação da arquitetura utilizaram o tempo de resposta. Os autores buscaram otimizar o desempenho do sistema por consultas individuais, e incluíram técnicas de redes neurais em sua abordagem para diminuir o tempo de execução do algoritmo genético. O tempo de treino do algoritmo levou cerca de 2h30 em armazenamento em disco rígido e 1h20 em armazenamento híbrido.

6.3 Relational Database Index Selection Algorithm

Boronski et al. (BB14) propõe um modelo para otimizar o tempo de resposta de um conjunto de consultas, através da criação de índices em um banco de dados relacional. O ciclo evolutivo do algoritmo funciona através de 3 passos: geração da população inicial com base em 6 configurações distintas de índices, seleção dos melhores indivíduos e verificação do critério de parada. A geração da população foi realizada considerando 6 indivíduos com configurações diferentes, como, sem índices, índices criados apenas nas colunas mais comuns, índices são criados apenas na primeira metade das colunas de cada tabela, nenhum índice é criado exceto aqueles sujeitos a probabilidade de mutação, índices criados apenas nas colunas mais comuns e demais sujeitos com base na probabilidade de mutação, e por fim, índices são criados apenas na primeira metade das colunas de cada tabela e demais sujeitos de acordo com a probabilidade de mutação. Para validação do trabalho, os autores criaram 3 tabelas, com 107 registros no total, e implementaram 3 grupos de consultas. O objetivo do algoritmo é obter resultados não piores do que o otimizador do SGBD da Oracle.

6.4 Otimização de Indexação de Dados com IA

Outros trabalhos que também usam técnicas de Inteligência Artificial para criar índices automaticamente em bancos de dados relacionais foram encontrados. Basu et al. (BLC⁺16) e Sharma et al. (SSD18) usam aprendizado por reforço para criar índices. Kraska et al. (KBC⁺18) usou o AM em cenários de *Big Data* para criar estruturas de índices genéricas. Outros dois trabalhos encontrados não utilizaram técnicas de exploração nem algoritmos de aprendizado em suas abordagens, entretanto, propuseram um modelo para melhorar o desempenho do SGBD. Enquanto Petraki et al. (PIM15) explorou os recursos da CPU com o processamento de consultas, Dash et al. (DPA11) sugere índices candidatos para instâncias de problemas grandes que tentam ser melhores que um orientador de SGBD, mas não são automatizadas.

6.5 Considerações do Capítulo

O espaço de busca explorado no ITLCS. (PNR18) é representado por comandos de consulta em SQL. Basu et al. (BLC⁺16) propôs um modelo para melhorar o desempenho de sistemas com intensas operações de INSERT e DELETE. A caracterização do problema que Kraska et al. (KBC⁺18) tenta solucionar é bastante similar ao presente trabalho, porém, os autores utilizaram AR como técnica de exploração. A estratégia proposta por Korytkowski et al. (KGNS04) se diferencia por tentar otimizar o tempo de resposta em operações de inserção de dados em um SGBD, utilizando AG. Por fim, o trabalho proposto por Boronski et al. (BB14) é diferente do presente trabalho por utilizar uma validação que envolve conjunto de dados e testes modelados pelos próprios autores, os quais não são expostos. Logo, o presente trabalho utilizou técnicas diferentes na implementação do espaço de busca, método e do conjunto de testes para validação.

7. CONCLUSÃO

Com a evolução da tecnologia dos bancos de dados nas últimas décadas, grandes volumes de dados têm sido acumulados, tornando indispensável a otimização dos seus sistemas de armazenamento. Qualquer aplicação corporativa precisa utilizar informações que devem ser armazenadas em sistemas de gerenciamento de banco de dados, os quais são predominantemente relacionais. O uso de consultas para acessar dados é uma característica presente em muitas empresas, assim como em instituições públicas e privadas. Um sistema que otimize o tempo de acesso a informações é extremamente útil, pois pode propiciar agilidade nos processos realizados em ambientes corporativos. Pesquisas recentes propuseram abordagens similares para otimizar o uso de índices (BLC⁺16, KBC⁺18, KGNS04, PNR18, BB14), com variabilidade nas técnicas utilizadas. Observou-se a oportunidade de aplicar uma estratégia de otimização de desempenho de consultas em SGBDs, tendo em vista essa divergência nas implementações.

A presente proposta apresentou uma estratégia de otimização de consultas para bancos de dados relacionais utilizando algoritmos genéticos. O objetivo principal do presente trabalho que era melhorar o desempenho de um conjunto de consultas submetido de forma frequente foi alcançado, uma vez que os resultados superaram dois otimizadores de SGBDs. Para isso, algumas análises foram realizadas com a finalidade de mapear da melhor forma possível o espaço de busca que o algoritmo genético explorou. Uma das definições realizada diz respeito as chaves primárias de um esquema. Uma vez que não vale a pena criar índices nessas colunas, não faz sentido mantê-las no espaço de busca para exploração das soluções. Resultados da análise experimental apontaram que a presente estratégia se mostrou superior. Em relação aos demais modelos, observou-se uma melhoria significativa, principalmente a partir do estado inicial do esquema para a configuração sub-ótima encontrada com a presente estratégia. Para o estado onde todas as colunas estão indexadas a melhora no tempo de execução das consultas foi pouca, porém, se comparado ao tamanho dos índices criados, pode-se se dizer que o algoritmo conseguiu chegar em um bom resultado de equilíbrio entre tempo e tamanho dos índices. Por fim, a presente estratégia se diferenciou do método de busca aleatória se mostrando mais robusta em sua exploração, uma vez que para convergência do algoritmo genético uma codificação coerente é essencial.

7.1 Contribuições

A estratégia de indexação automática para bancos de dados relacionais apresentada nesse trabalho mostrou-se capaz de conseguir excelentes resultados. Pode-se afirmar que o presente trabalho representa uma contribuição significativa na literatura, uma vez que, foi aceito em uma conferência de Qualis b1. O artigo denominado *GADIS: A Genetic Algorithm for Database Index Selection* ([?](#)) foi aceito na conferência SEKE ¹. Sobre o presente trabalho, pode-se observar que o mapeamento do espaço de busca, assim como a geração da população inicial, promoveu intensidade às primeiras explorações, uma vez que, diferentes configurações de indexação de colunas foram atingidas. A função de avaliação foi implementada de forma bastante simples e se mostrou eficiente na verificação dos melhores indivíduos, tendo em vista que, valores menores foram encontrados para o objetivo 1, tempo de resposta do conjunto de consultas. Com populações maiores, é possível encontrar uma solução melhor em relação ao objetivo 2, tamanho dos índices.

7.2 Limitações

Assim como a maioria dos algoritmos genéticos, o tempo de computação da presente estratégia é elevado. O algoritmo levou cerca de duas semanas para finalizar a execução de todas as gerações, devido à necessidade de realizar requisições ao banco de dados de forma intensiva. Para as abordagens que fizeram uso do *benchmark*, o tempo de execução do treino dobrou. O que tornou o uso do teste de desempenho do *benchmark* como função de avaliação do algoritmo genético inviável. Com o intuito de reduzir o tempo de execução do algoritmo, tentou-se paralelizar os experimentos criando cópias do esquema, entretanto, como o método trabalha com requisições ao banco de dados, os resultados acabaram sendo prejudicados por consequência das operações de entrada e saída. Apesar de ser utilizado um esquema com dados sintéticos que modela um sistema de banco de dados mais realístico, outra limitação encontrada foi a impossibilidade de testar a presente estratégia em um ambiente real de SGBD. Esses dois aspectos foram os maiores desafios desse trabalho.

¹<http://ksiresearchorg.ipage.com/seke/seke19.html>

7.3 Trabalhos Futuros

Como trabalhos futuros, deseja-se paralelizar o ciclo evolutivo do treino do algoritmo, destinando indivíduos da população em avaliação para diferentes núcleos de processamento. Outros objetivos estão descritos a seguir. Aplicar o presente método em esquemas com tamanhos maiores. Utilizar abordagens como análise lexicográfica para comparar com os resultados obtidos na presente estratégica. Inicializar a população de forma não totalmente randômica, mas uma parte dela de acordo com algum critério baseado nas consultas aplicadas ao sistema. Também deseja-se utilizar um gerador de dados para operações de inclusão e exclusão no esquema durante o processo de treino do algoritmo com a finalidade de testar o quanto o método consegue se adequar com a dinamicidade dos dados. Por fim, permitir a variabilidade nas estruturas de índices, tais como a inclusão de mapa de bits, e índices compostos.

REFERÊNCIAS BIBLIOGRÁFICAS

- [] Arroyo, J. E. C.; et al.. “Heurísticas e metaheurísticas para otimização combinatória multiobjetivo”, 2002.
- [BB14] Boronski, R.; Bocewicz, G. “Relational database index selection algorithm”. In: International Conference on Computer Networks, 2014, pp. 338–347.
- [BLC⁺16] Basu, D.; Lin, Q.; Chen, W.; Vo, H. T.; Yuan, Z.; Senellart, P.; Bressan, S. “Regularized cost-model oblivious database tuning with reinforcement learning”. In: *Transactions on Large-Scale Data-and Knowledge-Centered Systems*, Springer, Berlin: Springer, 1st ed. 2016 edition (September 10, 2016), 2016, pp. 96–132.
- [BMP12] Bhandari, D.; Murthy, C.; Pal, S. K. “Variance as a stopping criterion for genetic algorithms with elitist model”, *Fundamenta Informaticae*, vol. 120–2, 2012, pp. 145–164.
- [CLRS01] Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. “Introduction to Algorithms, Second Edition”. Cambridge, MA: The MIT Press, 2001, 1184p.
- [CPL04] Coello, C. A. C.; Pulido, G. T.; Lechuga, M. S. “Handling multiple objectives with particle swarm optimization”, *IEEE Transactions on evolutionary computation*, vol. 8–3, 2004, pp. 256–279.
- [DGH07] Diaz-Gomez, P. A.; Hougen, D. F. “Initial population for genetic algorithms: A metric approach.” In: GEM 2007: Genetic and Evolutionary Methods, 2007, pp. 43–49.
- [DPA11] Dash, D.; Polyzotis, N.; Ailamaki, A. “Cophy: a scalable, portable, and interactive index advisor for large workloads”, *Proceedings of the VLDB Endowment*, vol. 4–6, 2011, pp. 362–372.
- [DPAM02] Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. “A fast and elitist multiobjective genetic algorithm: Nsga-ii”, *IEEE transactions on evolutionary computation*, vol. 6–2, 2002, pp. 182–197.
- [DPP⁺18] Durand, G. C.; Pinnecke, M.; Piriyev, R.; Mohsen, M.; Broneske, D.; Saake, G.; Sekeran, M. S.; Rodriguez, F.; Balami, L. “Gridformation: Towards self-driven online data partitioning using reinforcement learning”. In: Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, 2018, pp. 1.

- [EDB] “EDB: EnterpriseDB”. Accessed: 2019-07-10, Capturado em: <https://www.enterprisedb.com/>.
- [] Fotouhi, F.; Galarce, C. E. “Genetic algorithms and the search for optimal database index selection”. In: Great Lakes CS Conference on New Research Results in Computer Science, 1989, pp. 249–255.
 - [] Finkelstein, S.; Schkolnick, M.; Tiberio, P. “Physical database design for relational databases”, *ACM Transactions on Database Systems (TODS)*, vol. 13–1, 1988, pp. 91–128.
- [GCJ17] Ghoreishi, S. N.; Clausen, A.; Jørgensen, B. N. “Termination criteria in evolutionary algorithms: A survey.” In: IJCCI, 2017, pp. 373–384.
- [] Goldberg, D. E.; Holland, J. H. “Genetic algorithms and machine learning”, *Machine learning*, vol. 3–2, 1988, pp. 95–99.
 - [] Holland, J. H. “Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence”. Cambridge, MA, USA: MIT Press, 1992.
 - [] Han, J.; Pei, J.; Kamber, M. “Data mining: concepts and techniques”. Elsevier, 2011.
- [IM96] Ishibuchi, H.; Murata, T. “Multi-objective genetic local search algorithm”. In: Proceedings of IEEE international conference on evolutionary computation, 1996, pp. 119–124.
- [JPW01] Jain, B. J.; Pohlheim, H.; Wegener, J. “On termination criteria of evolutionary algorithms”. In: Proceedings of the Genetic and Evolutionary Computation Conference, 2001, pp. 768.
- [KBC⁺18] Kraska, T.; Beutel, A.; Chi, E. H.; Dean, J.; Polyzotis, N. “The case for learned index structures”. In: ICMD, 2018, pp. 489–504.
- [KCS06] Konak, A.; Coit, D. W.; Smith, A. E. “Multi-objective optimization using genetic algorithms: A tutorial”, *Reliability Engineering & System Safety*, vol. 91–9, 2006, pp. 992–1007.
- [KD13] Khurana, U.; Deshpande, A. “Hinge: enabling temporal network analytics at scale”. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, 2013, pp. 1089–1092.
- [KGNS04] Korytkowski, M.; Gabryel, M.; Nowicki, R.; Scherer, R. “Genetic algorithm for database indexing”. In: ICAISC, Rutkowski, L.; Siekmann, J. H.; Tadeusiewicz, R.; Zadeh, L. A. (Editores), 2004, pp. 1142–1147.

- [KIM⁺11] Kersten, M. L.; Idreos, S.; Manegold, S.; Liarou, E. “The researcher’s guide to the data deluge: Querying a scientific database in just a few seconds”. In: International Conference on Very Large Data Bases, 2011, pp. 1474–1477.
- [KNR13] Kumar, A.; Niu, F.; Ré, C. “Hazy: making it easier to build and maintain big-data analytics”, *Communications of the ACM*, vol. 56–3, 2013, pp. 40–49.
- [Ric08] Linden, R. “Algoritmos genéticos: uma importante ferramenta da inteligência computacional”. Brasport, 2006, 2 ed..
- [LLHL14] Luo, Y.; Liu, M.; Hao, Z.; Liu, D. “An improved nsga-ii algorithm for multi-objective traveling salesman problem”, *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 12–6, 2014, pp. 4413–4418.
- [LW01] Leung, Y.-W.; Wang, Y. “An orthogonal genetic algorithm with quantization for global numerical optimization”, *IEEE Transactions on Evolutionary computation*, vol. 5–1, 2001, pp. 41–53.
- [LZZ13] Laptev, N.; Zeng, K.; Zaniolo, C. “Very fast estimation for result and accuracy of big data analytics: The earl system”. In: 2013 IEEE 29th International Conference on Data Engineering (ICDE), 2013, pp. 1296–1299.
- [Mit96] Mitchell, M. “An Introduction to Genetic Algorithms”. Cambridge, MA, USA: MIT Press, 1996.
- [] Neuhaus, P.; Couto, J.; Wehrmann, J.; Ruiz, D. D.; Meneguzzi, F. “Gadis: A genetic algorithm for database index selection”.
- [PIM15] Petraki, E.; Idreos, S.; Manegold, S. “Holistic indexing in main-memory column-stores”. In: SIGMOD, 2015, pp. 1153–1166.
- [PNR18] Pedrozo, W. G.; Nievola, J. C.; Ribeiro, D. C. “An adaptive approach for index tuning with learning classifier systems on hybrid storage environments”. In: HAIS, 2018, pp. 716–729.
- [POW] “POWA: PostgreSQL Workload Analyzer”. Accessed: 2019-07-10, Capturado em: <https://powa.readthedocs.io/en/latest/>.
- [RC02] Rob, P.; Coronel, C. “Database Systems Design, Implementation and Management”. Boston, MA, United States: Course Technology Press, 2002, 5th ed..
- [RG00] Ramakrishnan, R.; Gehrke, J. “Database management systems”. McGraw Hill, 2000.

- [SD94] Srinivas, N.; Deb, K. “Multiobjective optimization using nondominated sorting in genetic algorithms”, *Evolutionary computation*, vol. 2–3, 1994, pp. 221–248.
- [SD08] Sivanandam, S.; Deepa, S. “Genetic algorithm optimization problems”. In: *Introduction to genetic algorithms*, Springer, 2008, pp. 165–209.
- [SDJ91] Spears, W. M.; De Jong, K. A. “An analysis of multi-point crossover”. In: *Foundations of genetic algorithms*, Elsevier, 1991, vol. 1, pp. 301–315.
- [SGK05] Sastry, K.; Goldberg, D.; Kendall, G. “Genetic algorithms”. In: *Search methodologies*, Springer, 2005, pp. 97–125.
- [Sim13] Simon, P. “Too big to ignore: the business case for big data”. John Wiley & Sons, 2013, vol. 72, 225p.
- [SKS10] Silberschatz, A.; Korth, H. F.; Sudarshan, S. “Database system concepts”. New York: McGraw-Hill, 2010, 6 ed..
- [SSD18] Sharma, A.; Schuhknecht, F. M.; Dittrich, J. “The case for automatic database administration using deep reinforcement learning”, *CoRR*, vol. abs/1801.05643, 2018, pp. 1–9.
- [SSS19] Sharma, M.; Singh, G.; Singh, R. “A review of different cost-based distributed query optimizers”, *Progress in Artificial Intelligence*, vol. 8–1, 2019, pp. 45–62.
- [SSSF09] Sint, R.; Schaffert, S.; Stroka, S.; Ferstl, R. “Combining unstructured, fully structured and semi-structured information in semantic wikis”. In: CEUR Workshop Proceedings, 2009, pp. 73–87.
- [SZT12] Schwartz, B.; Zaitsev, P.; Tkachenko, V. “High performance MySQL: optimization, backups, and replication”. "O'Reilly Media, Inc.", 2012.
- [TCG12] Thanopoulou, A.; Carreira, P.; Galhardas, H. “Benchmarking with tpc-h on off-the-shelf hardware”. In: 14th International Conference on Enterprise Information Systems, 2012, pp. 205–208.
- [WLKC16] Wang, J.; Liu, W.; Kumar, S.; Chang, S.-F. “Learning to hash for indexing big data—a survey”, *Proceedings of the IEEE*, vol. 104–1, 2016, pp. 34–57.

APÊNDICE A – COMANDO SQL DE TODAS AS CONSULTAS UTILIZADAS NA EXECUÇÃO DO ALGORITMO GENÉTICO

```

SELECT l_returnflag, l_linenstatus,
       SUM(l_quantity) AS sum_qty,
       SUM(l_extendedprice) AS sum_base_price,
       SUM(l_extendedprice * (1 - l_discount)) AS sum_disc_price,
       SUM(l_extendedprice * (1 - l_discount) * (1 + l_tax)) AS sum_charge,
               AVG(l_quantity) AS avg_qty, AVG(l_extendedprice) AS avg_price,
               AVG(l_discount) AS avg_disc, count(*) AS count_order
  FROM
    lineitem
 WHERE
   l_shipdate <= DATE '1994-7-17' - INTERVAL '108' DAY
 GROUP BY l_returnflag, l_linenstatus
 ORDER BY l_returnflag, l_linenstatus;

```

Figura A.1 – Consulta referente *Query #1*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que ão foram desconsideradas.

```

SELECT
    s_acetbal,
    s_name,
    n_name,
p_partkey,
p_mfgr,
s_address,
s_phone,
s_comment
FROM
    part,
    supplier,
partsupp,
    nation,
    region
WHERE
    p_partkey = ps_partkey
    AND s_suppkey = ps_suppkey
    AND p_size = 30
    AND p_type like '%STEEL'
    AND s_nationkey = n_nationkey
    AND n_regionkey = r_regionkey
    AND r_name = 'ASIA'
    AND ps_supplycost =
        (
            SELECT
                MIN(ps_supplycost)
            FROM
                partsupp,
                supplier,
                nation,
                region
            WHERE
                p_partkey = ps_partkey
                and s_suppkey = ps_suppkey
                and s_nationkey = n_nationkey
                and n_regionkey = r_regionkey
                and r_name = 'ASIA'
        )
    ORDER BY
        s_acetbal DESC,
        n_name,
        s_name,
        p_partkey
LIMIT 100;

```

Figura A.2 – Consulta referente *Query #2*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas des- tacadas com sublinhado tachado representam aquelas que ão foram desconsideradas.

```
SELECT
    l_orderkey,
    SUM(l_extendedprice * (1 - l_discount)) AS revenue,
    o_orderdate,
    o_shipppriority
FROM
    customer,
    orders,
    lineitem
WHERE
    c_mktsegment = 'AUTOMOBILE'
    AND c_custkey = o_custkey
    AND l_orderkey = o_orderkey
    AND o_orderdate < date '1995-03-13'
    AND l_shipdate > date '1995-03-13'
GROUP BY
    l_orderkey,
    o_orderdate,
    o_shipppriority
ORDER BY
    revenue DESC,
    o_orderdate
LIMIT 10;
```

Figura A.3 – Consulta referente *Query #3*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição.

```
SELECT
    o_orderpriority,
    COUNT(*) ASorder_count
FROM
    orders
WHERE
    o_orderdate >= DATE '1995-01-01'
    AND o_orderdate < DATE '1995-01-01' + INTERVAL '3' MONTH
    AND EXISTS (
        SELECT
            *
        FROM
            lineitem
        WHERE
            l_orderkey = o_orderkey
            AND l_commitdate < l_receiptdate
    )
GROUP BY
    o_orderpriority
ORDER BY
    o_orderpriority;
```

Figura A.4 – Consulta referente *Query #4*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição.

```

SELECT
    n_name,
    SUM(l_extendedprice * (1 - l_discount)) AS revenue
FROM
    customer,
    orders,
    lineitem,
    supplier,
    nation,
    region
WHERE
    c_custkey = o_custkey
    AND l_orderkey = o_orderkey
    AND l_suppkey = s_suppkey
    AND c_nationkey = s_nationkey
    AND s_nationkey = n_nationkey
    AND n_regionkey = r_regionkey
    AND r_name = 'MIDDLE EAST'
    AND o_orderdate >= DATE '1994-01-01'
    AND o_orderdate < DATE '1994-01-01' + INTERVAL '1' YEAR
GROUP BY
    n_name
ORDER BY
    revenue desc;

```

Figura A.5 – Consulta referente *Query #5*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que não foram desconsideradas.

```

SELECT
    SUM(l_extendedprice * l_discount) AS revenue
FROM
    lineitem
WHERE
    l_shipdate >= DATE '1993-01-01'
    AND l_shipdate < DATE '1993-01-01' + INTERVAL '1' YEAR
    AND l_discount BETWEEN 0.06 - 0.01 AND 0.06 + 0.01
    AND l_quantity < 24;

```

Figura A.6 – Consulta referente *Query #6*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição.

```

SELECT
    supp_nation,
    cust_nation,
    l_year,
    SUM(volume) AS revenue
FROM
(
    SELECT
        n1.n_name as supp_nation,
        n2.n_name as cust_nation,
        extract(year from l_shipdate) AS l_year,
        l_extendedprice * (1 - l_discount) AS volume
    FROM
        supplier,
        lineitem,
        orders,
        customer,
        nation n1,
        nation n2
    WHERE
        s_suppkey = l_suppkey
        AND o_orderkey = l_orderkey
        AND c_custkey = o_custkey
        AND s_nationkey = n1.n_nationkey
        AND c_nationkey = n2.n_nationkey
        AND (
            (n1.n_name = 'JAPAN' AND n2.n_name = 'INDIA')
            OR (n1.n_name = 'INDIA' AND n2.n_name = 'JAPAN')
        )
        AND l_shipdate BETWEEN DATE '1995-01-01' AND DATE '1996-12-31'
    ) AS shipping
GROUP BY
    supp_nation,
    cust_nation,
    l_year
ORDER BY;
    supp_nation,
    cust_nation,
    l_year;

```

Figura A.7 – Consulta referente *Query #7*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que não foram desconsideradas.

```

SELECT
    o_year,
    SUM(CASE
        WHEN nation = 'INDIA' THEN volume
        ELSE 0
    END) / SUM(volume) AS mkt_share
FROM
(
    SELECT
        EXTRACT(YEAR FROM o_orderdate) AS o_year,
        l_extendedprice * (1 - l_discount) AS volume,
        n2.n_name AS nation
    FROM
        part,
        supplier,
        lineitem,
        orders,
        customer,
        nation n1,
        nation n2,
        region
    WHERE
        p_partkey = l_partkey
        AND s_suppkey = l_suppkey
        AND l_orderkey = o_orderkey
        AND o_custkey = c_custkey
        AND c_nationkey = n1.n_nationkey
        AND n1.n_regionkey = r_regionkey
        AND r_name = 'ASIA'
        AND s_nationkey = n2.n_nationkey
        AND o_orderdate BETWEEN DATE '1995-01-01' AND DATE
        '1996-12-31'
        AND p_type = 'SMALL PLATED COPPER'
    ) AS all_nations
GROUP BY
    o_year
ORDER BY
    o_year;

```

Figura A.8 – Consulta referente *Query #8*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que são foram desconsideradas.

```

SELECT
    nation,
    o_year,
    SUM(amount) AS sum_profit
FROM
(
    SELECT
        n_name AS nation,
        EXTRACT(YEAR FROM o_orderdate) AS o_year,
        l_extendedprice * (1 - l_discount) - ps_supplycost *
l_quantity AS amount
    FROM
        part,
        supplier,
        lineitem,
        partsupp,
        orders,
        nation
    WHERE
        s_suppkey = l_suppkey
        AND ps_suppkey = l_suppkey
        AND ps_partkey = l_partkey
        AND p_partkey = l_partkey
        AND o_orderkey = l_orderkey
        AND s_nationkey = n_nationkey
        AND p_name LIKE '%dim%'
    ) AS profit
GROUP BY
    nation,
    o_year
ORDER BY
    nation,
    o_year DESC;

```

Figura A.9 – Consulta referente *Query #9*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que não foram desconsideradas.

```

SELECT
    c_custkey,
    c_name,
    SUM(l_extendedprice * (1 - l_discount)) AS revenue,
    c_acctbal,
    n_name,
    c_address,
    c_phone,
    c_comment
FROM
    customer,
    orders,
    lineitem,
    nation
WHERE
    c_custkey = o_custkey
    AND l_orderkey = o_orderkey
    AND o_orderdate >= DATE '1993-08-01'
    AND o_orderdate < DATE '1993-08-01' + INTERVAL '3' MONTH
    AND l_returnflag = 'R'
    AND c_nationkey = n_nationkey
GROUP BY
    c_custkey,
    c_name,
    c_acctbal,
    c_phone,
    n_name,
    c_address,
    c_comment
ORDER BY
    revenue DESC
LIMIT 20;

```

Figura A.10 – Consulta referente *Query #10*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que são foram desconsideradas.

```

SELECT
    ps_partkey,
    SUM(ps_supplycost * ps_availqty) AS value
FROM
    partsupp,
    supplier,
    nation
WHERE
    ps_suppkey = s_suppkey
    AND s_nationkey = n_nationkey
    AND n_name = 'MOZAMBIQUE'
GROUP BY
    ps_partkey HAVING
        SUM(ps_supplycost * ps_availqty) > (
            SELECT
                SUM(ps_supplycost * ps_availqty) * 0.0001000000
            FROM
                partsupp,
                supplier,
                nation
            WHERE
                ps_suppkey = s_suppkey
                AND s_nationkey = n_nationkey
                AND n_name = 'MOZAMBIQUE'
        )
ORDER BY
    value DESC;

```

Figura A.11 – Consulta referente *Query #11*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que são foram desconsideradas.

```

SELECT
    l_shipmode,
    SUM(CASE
        WHEN o_orderpriority = '1-URGENT'
            OR o_orderpriority = '2-HIGH'
            THEN 1
        ELSE 0
    END) AS high_line_count,
    SUM(CASE
        WHEN o_orderpriority <> '1-URGENT'
            AND o_orderpriority <> '2-HIGH'
            THEN 1
        ELSE 0
    END) AS low_line_count
FROM
    orders,
    lineitem
WHERE
    o_orderkey = l_orderkey
    AND l_shipmode IN ('RAIL', 'FOB')
    AND l_commitdate < l_receiptdate
    AND l_shipdate < l_commitdate
    AND l_receiptdate >= DATE '1997-01-01'
    AND l_receiptdate < DATE '1997-01-01' + INTERVAL '1' YEAR
GROUP BY
    l_shipmode
ORDER BY
    l_shipmode;

```

Figura A.12 – Consulta referente *Query #12*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição.

```

SELECT
    c_count,
    COUNT(*) AS custdist
FROM
(
    SELECT
        c_custkey,
        COUNT(o_orderkey) AS c_count
    FROM
        customer LEFT OUTER JOIN orders ON
            c_custkey = o_custkey
            AND o_comment NOT LIKE '%pending%deposits%'
    GROUP BY
        c_custkey
) c_orders
GROUP BY
    c_count
ORDER BY
    custdist DESC,
    c_count DESC;

```

Figura A.13 – Consulta referente *Query #13*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição.

```

SELECT
    100.00 * SUM(CASE
        WHEN p_type LIKE 'PROMO%'
            THEN l_extendedprice * (1 - l_discount)
        ELSE 0
    END) / SUM(l_extendedprice * (1 - l_discount)) AS promo_revenue
FROM
    lineitem,
    part
WHERE
    l_partkey = p_partkey
    AND l_shipdate >= DATE '1996-12-01'
    AND l_shipdate < DATE '1996-12-01' + INTERVAL '1' MONTH;

```

Figura A.14 – Consulta referente *Query #14*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição.

```

SELECT
    l_suppkey,
    SUM(l_extendedprice * (1 - l_discount))
FROM
    lineitem
WHERE
    l_shipdate >= DATE '1997-07-01'
    AND l_shipdate < DATE '1997-07-01' + INTERVAL '3' MONTH
GROUP BY
    l_suppkey;

SELECT
    s_suppkey,
    s_name,
    s_address,
    s_phone,
    total_revenue
FROM
    supplier,
    revenue0
WHERE
    s_suppkey = supplier_no
    AND total_revenue = (
        SELECT
            MAX(total_revenue)
        FROM
            revenue0
    )
ORDER BY
    s_suppkey;

```

Figura A.15 – Consulta referente *Query #15*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que não foram desconsideradas.

```

SELECT
    p_brand,
    p_type,
    p_size,
    COUNT(distinct ps_suppkey) AS supplier_cnt
FROM
    partsupp,
    part
WHERE
    p_partkey = ps_partkey
    AND p_brand <> 'Brand#34'
    AND p_type not like 'LARGE BRUSHED%'
    AND p_size in (48, 19, 12, 4, 41, 7, 21, 39)
    AND ps_suppkey NOT IN (
        SELECT
            s_suppkey
        FROM
            supplier
        WHERE
            s_comment LIKE '%Customer%Complaints%'
    )
GROUP BY
    p_brand,
    p_type,
    p_size
ORDER BY
    supplier_cnt DESC,
    p_brand,
    p_type,
    p_size;

```

Figura A.16 – Consulta referente *Query #16*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que ão foram desconsideradas.

```
SELECT
    SUM(l_extendedprice) / 7.0 AS avg_yearly
FROM
    lineitem,
    part
WHERE
    p_partkey = l_partkey
    AND p_brand = 'Brand#44'
    AND p_container = 'WRAP PKG'
    AND l_quantity < (
        SELECT
            0.2 * AVG(l_quantity)
        FROM
            lineitem
        WHERE
            l_partkey = p_partkey
    );
```

Figura A.17 – Consulta referente *Query #17*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição.

```

SELECT
    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice,
    sum(l_quantity)
FROM
    customer,
    orders,
    lineitem
WHERE
    o_orderkey in (
        SELECT
            l_orderkey
        FROM
            lineitem
        GROUP BY
            l_orderkey having
                sum(l_quantity) > 314
    )
    AND c_custkey = o_custkey
    AND o_orderkey = l_orderkey
GROUP BY
    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice
ORDER BY
    o_totalprice DESC,
    o_orderdate
LIMIT 100;

```

Figura A.18 – Consulta referente *Query #18*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição.

```

SELECT
    SUM(l_extendedprice* (1 - l_discount)) AS revenue
FROM
    lineitem,
    part
WHERE
(
    p_partkey = l_partkey
    AND p_brand = 'Brand#52'
    AND p_container IN ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
    AND l_quantity >= 4 AND l_quantity <= 4 + 10
    AND p_size BETWEEN 1 AND 5
    AND l_shipmode IN ('AIR', 'AIR REG')
    AND l_shipinstruct = 'DELIVER IN PERSON'
)
OR
(
    p_partkey = l_partkey
    AND p_brand = 'Brand#11'
    AND p_container IN ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
    AND l_quantity >= 18 AND l_quantity <= 18 + 10
    AND p_size BETWEEN 1 AND 10
    AND l_shipmode IN ('AIR', 'AIR REG')
    AND l_shipinstruct = 'DELIVER IN PERSON'
)
OR
(
    p_partkey = l_partkey
    AND p_brand = 'Brand#51'
    AND p_container IN ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
    AND l_quantity >= 29 AND l_quantity <= 29 + 10
    AND p_size BETWEEN 1 AND 15
    AND l_shipmode IN ('AIR', 'AIR REG')
    AND l_shipinstruct = 'DELIVER IN PERSON'
);

```

Figura A.19 – Consulta referente *Query #19*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição.

```

SELECT
    s_name,
    s_address
FROM
    supplier,
    nation
WHERE
    s_suppkey IN (
        SELECT
            ps_suppkey
        FROM
            partsupp
        WHERE
            ps_partkey IN (
                SELECT
                    p_partkey
                FROM
                    part
                WHERE
                    p_name LIKE 'green%'
            )
        AND ps_availqty > (
            SELECT
                0.5 * SUM(l_quantity)
            FROM
                lineitem
            WHERE
                l_partkey = ps_partkey
                AND l_suppkey = ps_suppkey
                AND l_shipdate >= DATE '1993-01-01'
                AND l_shipdate < DATE '1993-01-01'+INTERVAL '1' YEAR
        )
    )
    AND s_nationkey = n_nationkey
    AND n_name = 'ALGERIA'
ORDER BY
    s_name;

```

Figura A.20 – Consulta referente *Query #20*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que não foram desconsideradas.

```

SELECT
    s_name,
    count(*) AS numwait
FROM
    supplier,
    lineitem l1,
    orders,
    nation
WHERE
    s_suppkey = l1.l_suppkey
    AND o_orderkey = l1.l_orderkey
    AND o_orderstatus = 'F'
    AND l1.l_receiptdate > l1.l_commitdate
    AND EXISTS (
        SELECT
            *
        FROM
            lineitem l2
        WHERE
            l2.l_orderkey = l1.l_orderkey
            AND l2.l_suppkey <> l1.l_suppkey
    )
    AND NOT EXISTS (
        SELECT
            *
        FROM
            lineitem l3
        WHERE
            l3.l_orderkey = l1.l_orderkey
            AND l3.l_suppkey <> l1.l_suppkey
            AND l3.l_receiptdate > l3.l_commitdate
    )
    AND s_nationkey = n_nationkey
    AND n_name = 'EGYPT'
GROUP BY
    s_name
ORDER BY
    numwait DESC,
    s_name
LIMIT 100;

```

Figura A.21 – Consulta referente *Query #21*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição. Colunas e tabelas destacadas com sublinhado tachado representam aquelas que ão foram desconsideradas.

```

SELECT
    cntrycode,
    COUNT(*) AS numcust,
    SUM(c_acctbal) AS totacctbal
FROM
(
    SELECT
        SUBSTRING(c_phone FROM 1 for 2) AS cntrycode,
        c_acctbal
    FROM
        customer
    WHERE
        SUBSTRING(c_phone FROM 1 for 2) IN
            ('20', '40', '22', '30', '39', '42', '21')
        AND c_acctbal > (
            SELECT
                AVG(c_acctbal)
            FROM
                customer
            WHERE
                c_acctbal > 0.00
                AND SUBSTRING(c_phone FROM 1 for 2) IN
                    ('20', '40', '22', '30', '39', '42', '21')
        )
    AND NOT EXISTS (
        SELECT
            *
        FROM
            orders
        WHERE
            o_custkey = c_custkey
    )
) AS custsale
GROUP BY
    cntrycode
ORDER BY
    cntrycode;

```

Figura A.22 – Consulta referente *Query #22*. Colunas e tabelas destacadas em negrito representam aquelas consideradas no processo de otimização. Colunas e tabelas sublinhadas representam aquelas que possuem chaves primárias por definição.

APÊNDICE B – PLANO DE EXECUÇÃO DE TODAS AS CONSULTAS NO ESTADO INICIAL DO ESQUEMA E COM O MELHOR INDIVÍDUO ENCONTRADO CONFIGURADO NO ESQUEMA.

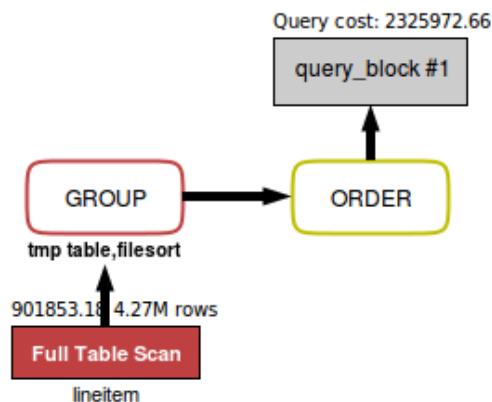


Figura B.1 – Plano de execução da consulta 1 no estado inicial do esquema (somente com PK e FK).

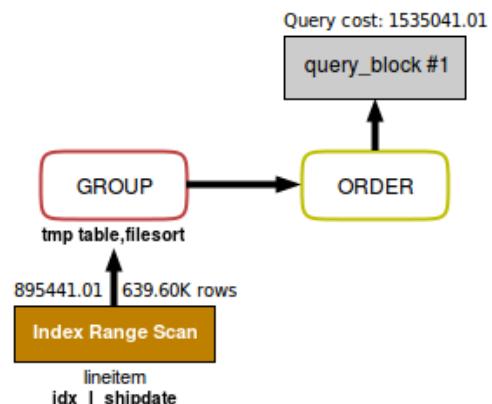


Figura B.2 – Plano de execução da consulta 1 com o melhor indivíduo.

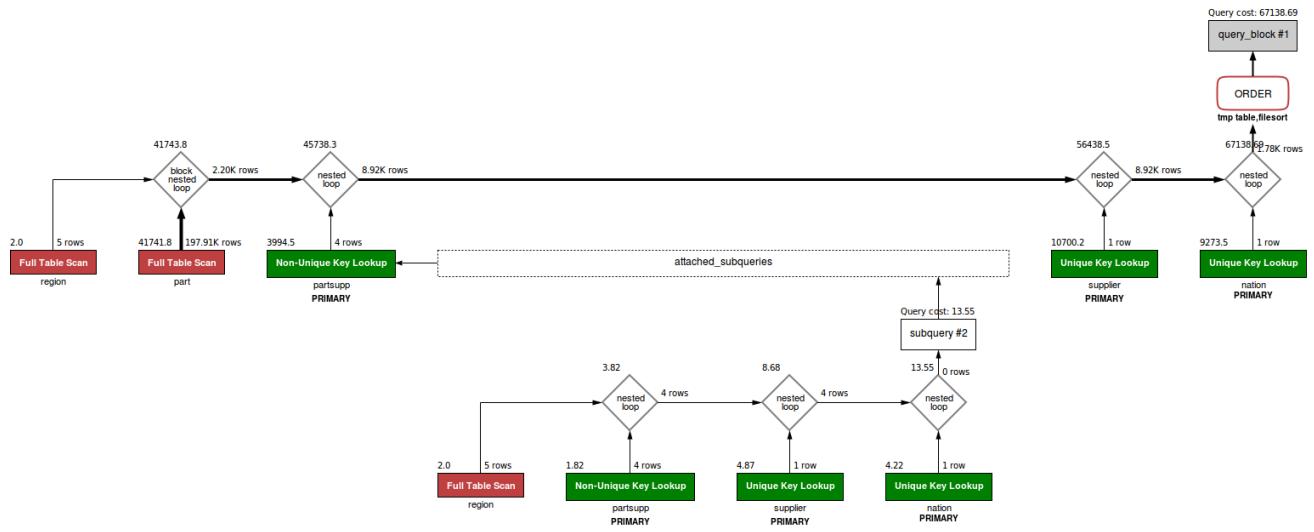


Figura B.3 – Plano de execução da consulta 2 no estado inicial do esquema (somente com PK e FK).

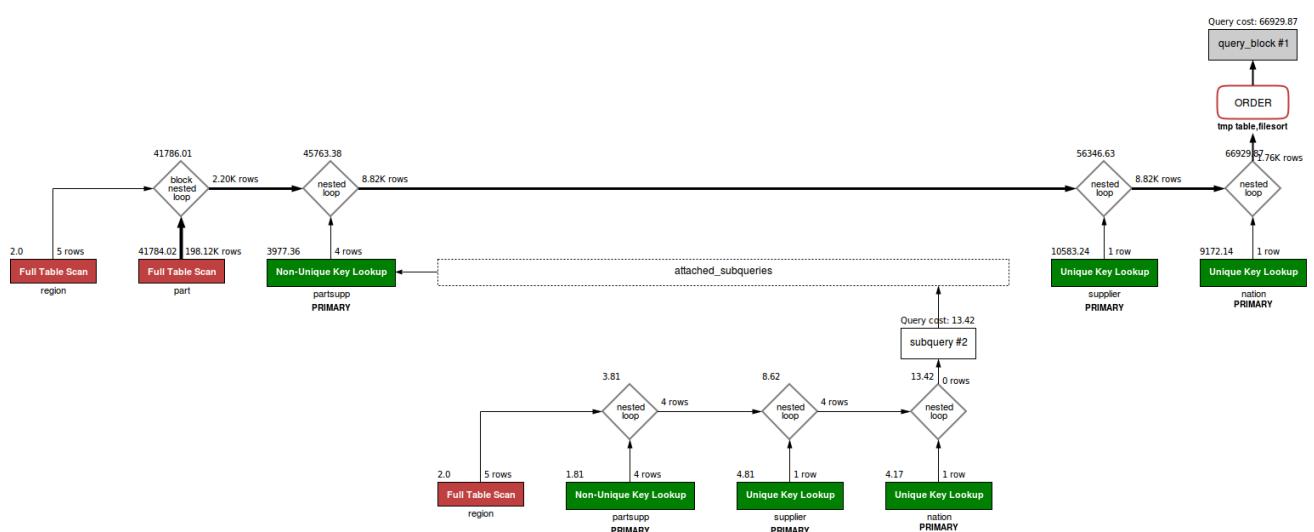


Figura B.4 – Plano de execução da consulta 2 com o melhor indivíduo.

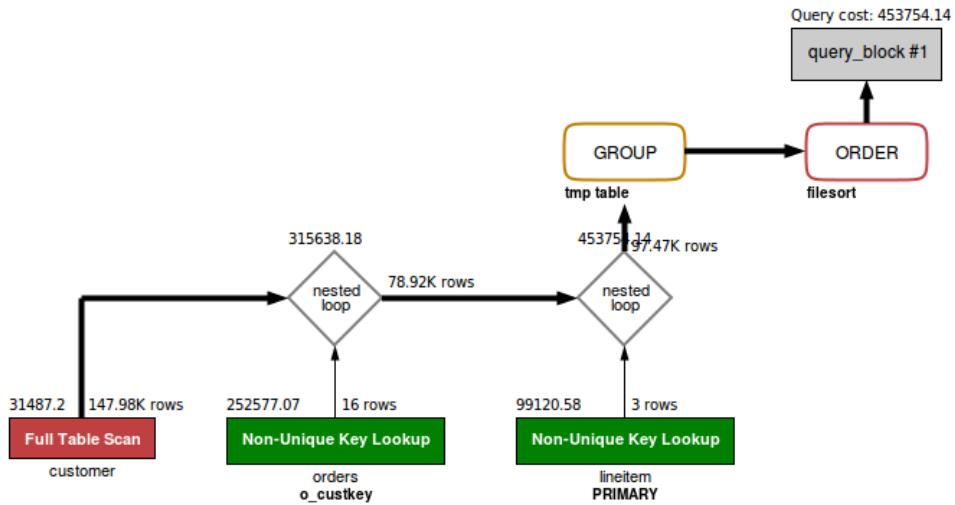


Figura B.5 – Plano de execução da consulta 3 no estado inicial do esquema (somente com PK e FK).

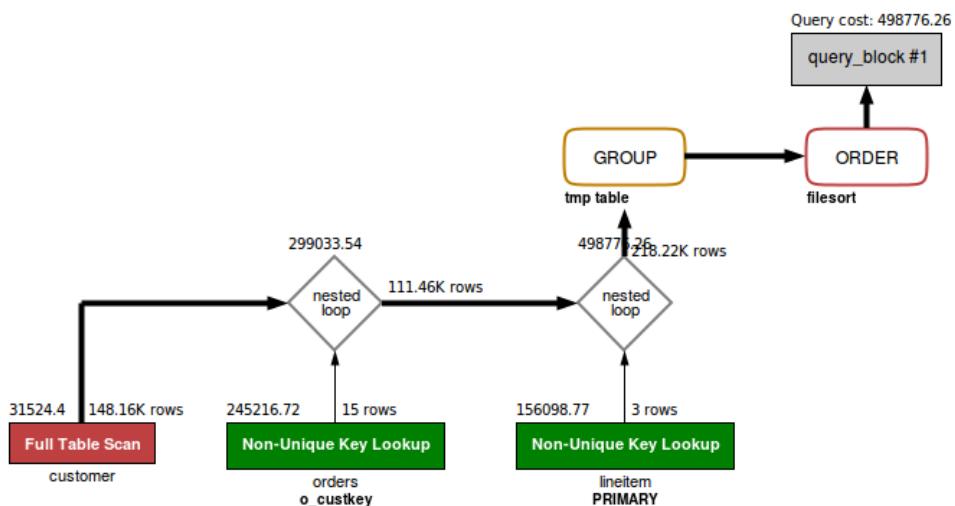


Figura B.6 – Plano de execução da consulta 3 com o melhor indivíduo.

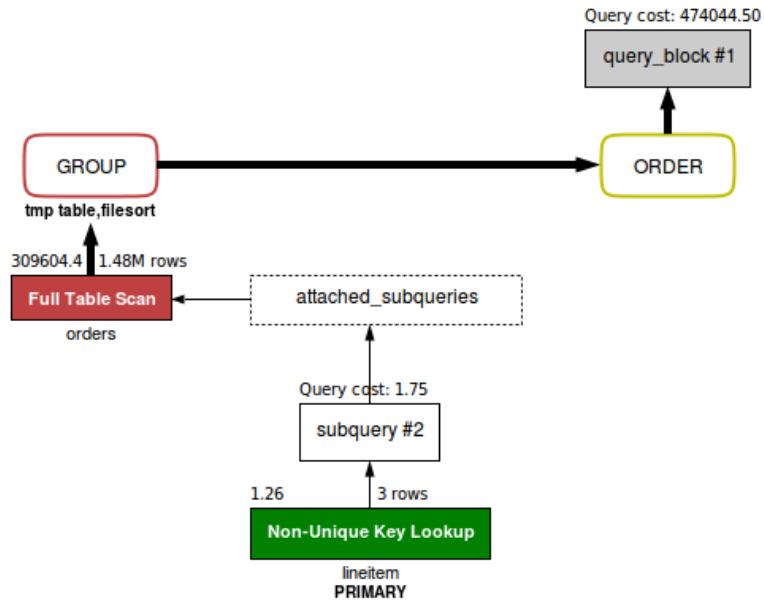


Figura B.7 – Plano de execução da consulta 4 no estado inicial do esquema (somente com PK e FK).

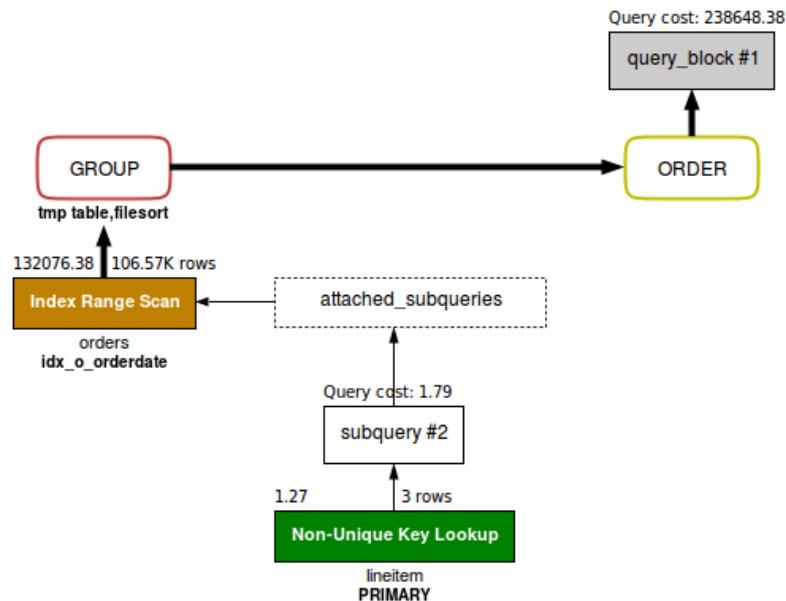


Figura B.8 – Plano de execução da consulta 4 com o melhor indivíduo.

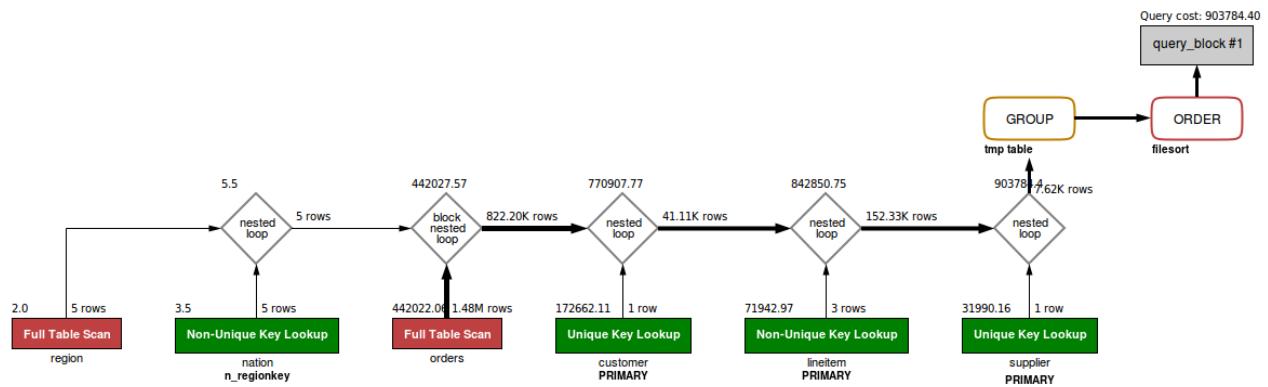


Figura B.9 – Plano de execução da consulta 5 no estado inicial do esquema (somente com PK e FK).

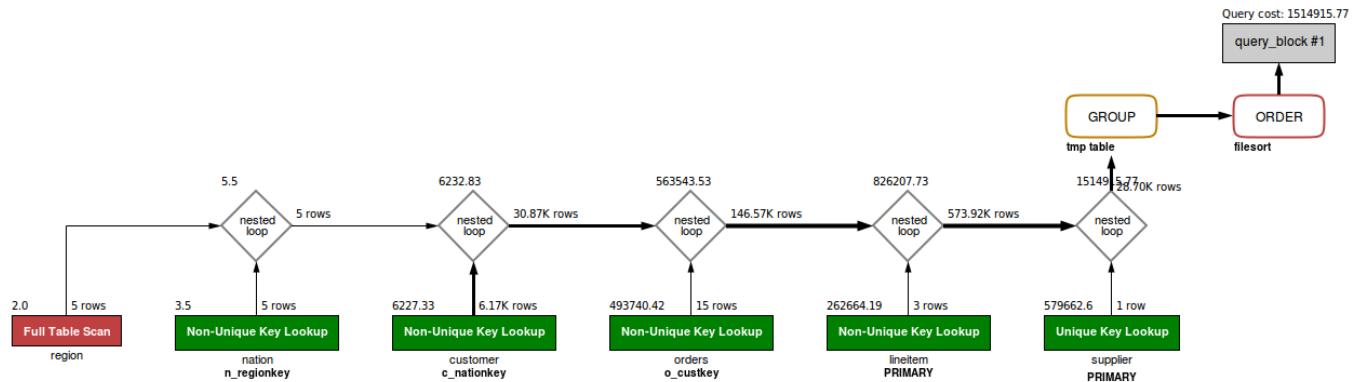


Figura B.10 – Plano de execução da consulta 5 com o melhor indivíduo.

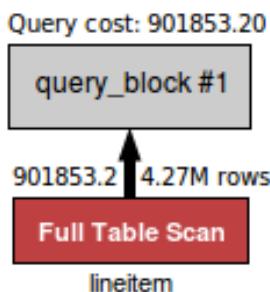


Figura B.11 – Plano de execução da consulta 6 no estado inicial do esquema (somente com PK e FK).

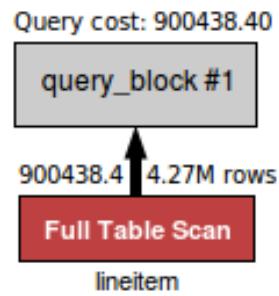


Figura B.12 – Plano de execução da consulta 6 com o melhor indivíduo.

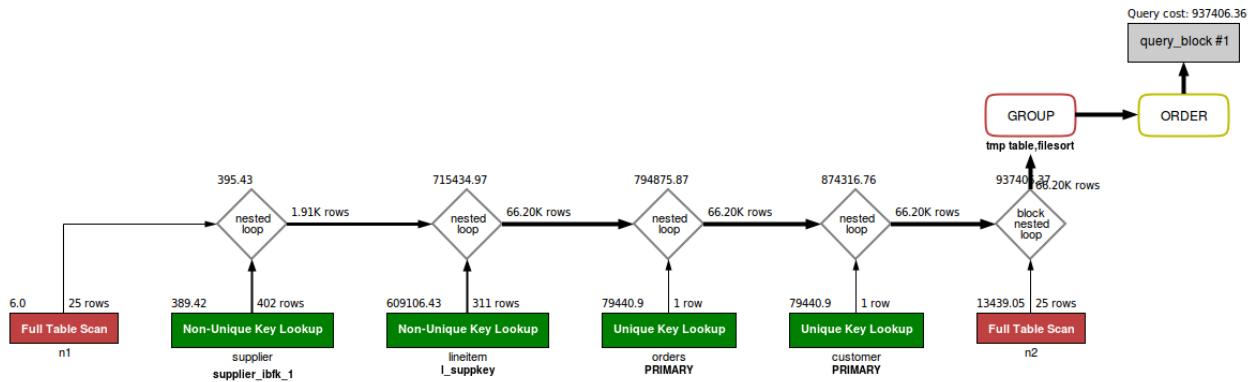


Figura B.13 – Plano de execução da consulta 7 no estado inicial do esquema (somente com PK e FK).

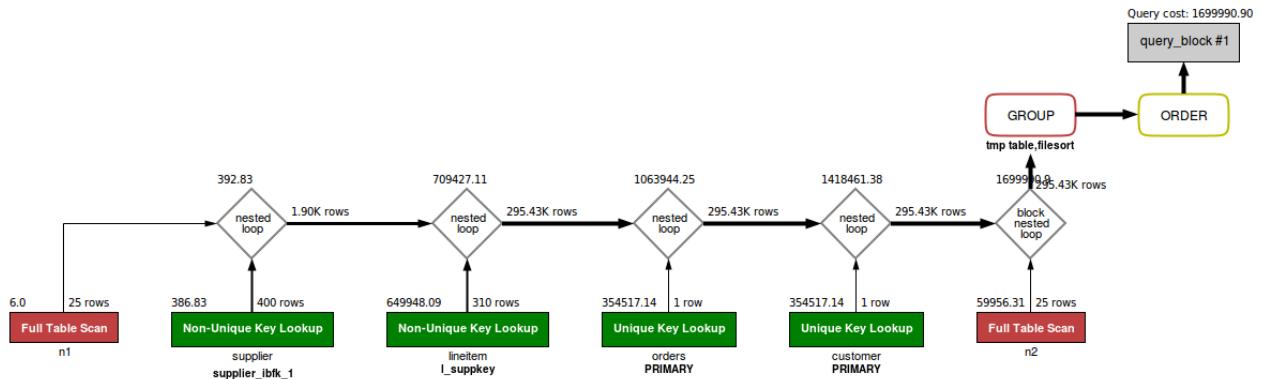


Figura B.14 – Plano de execução da consulta 7 com o melhor indivíduo.

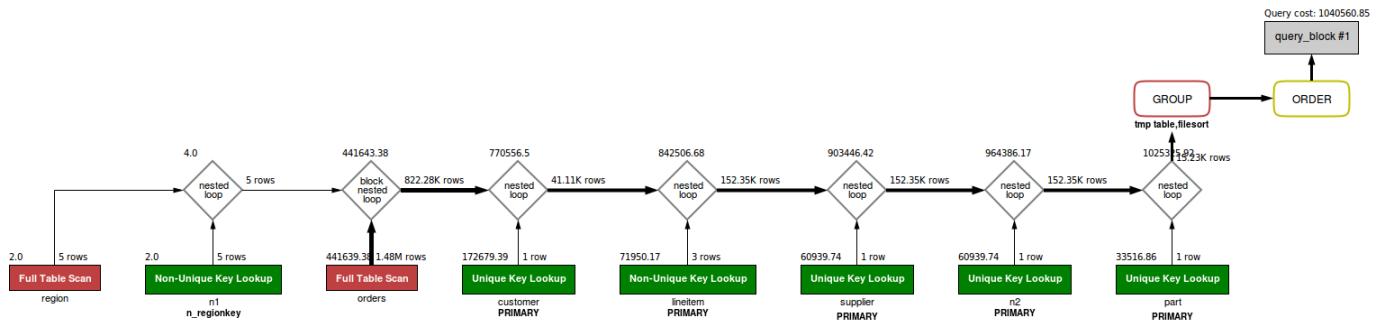


Figura B.15 – Plano de execução da consulta 8 no estado inicial do esquema (somente com PK e FK).

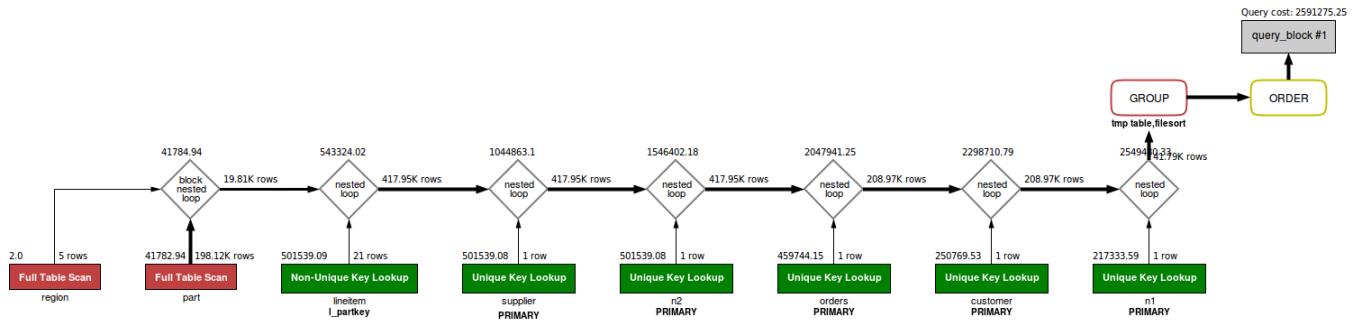


Figura B.16 – Plano de execução da consulta 8 com o melhor indivíduo.

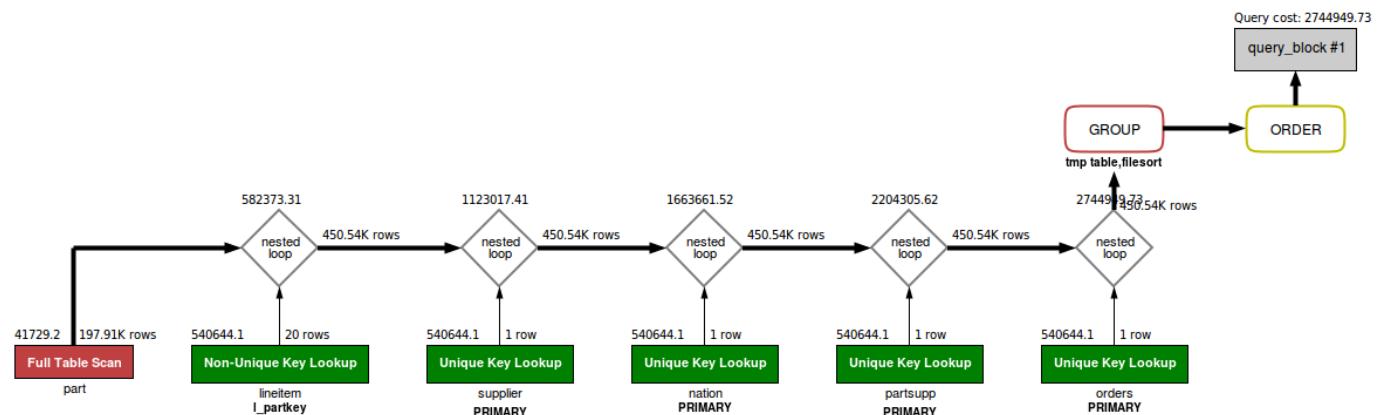


Figura B.17 – Plano de execução da consulta 9 no estado inicial do esquema (somente com PK e FK).

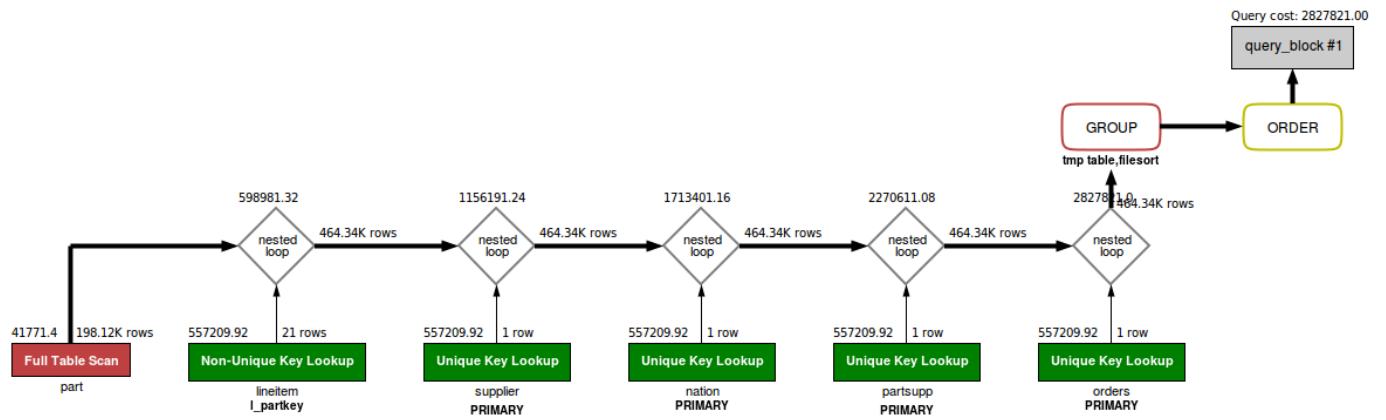


Figura B.18 – Plano de execução da consulta 9 com o melhor indivíduo.

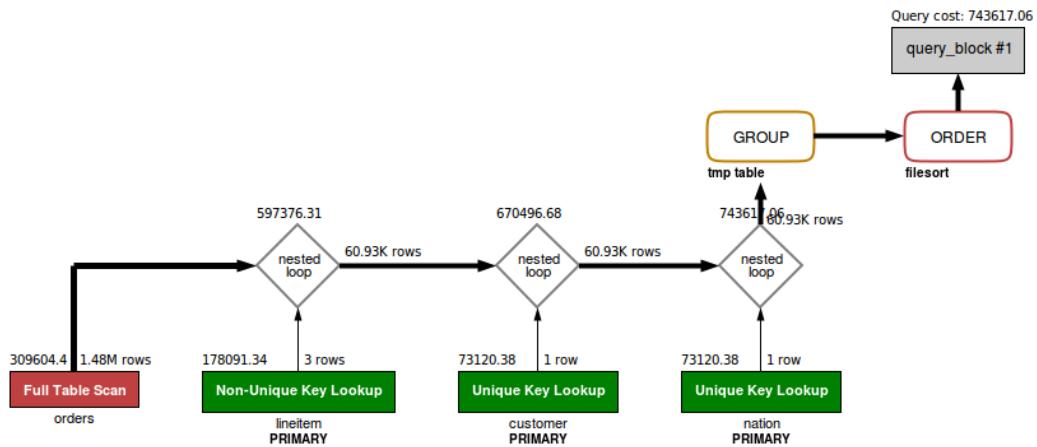


Figura B.19 – Plano de execução da consulta 10 no estado inicial do esquema (somente com PK e FK).

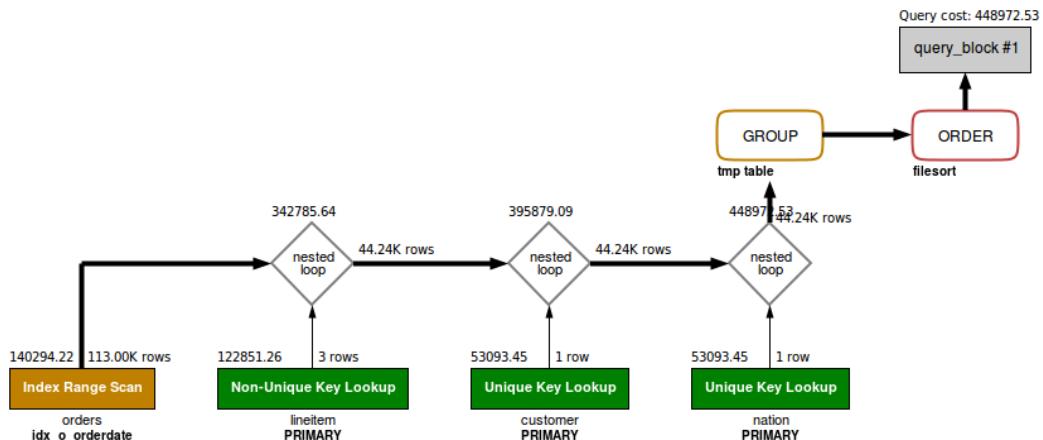


Figura B.20 – Plano de execução da consulta 10 com o melhor indivíduo.

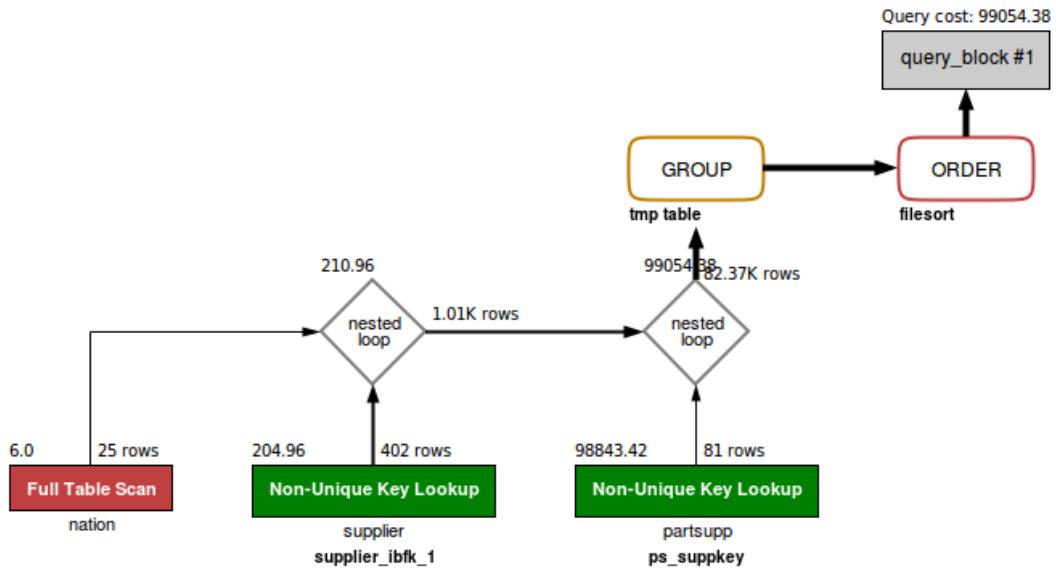


Figura B.21 – Plano de execução da consulta 11 no estado inicial do esquema (somente com PK e FK).

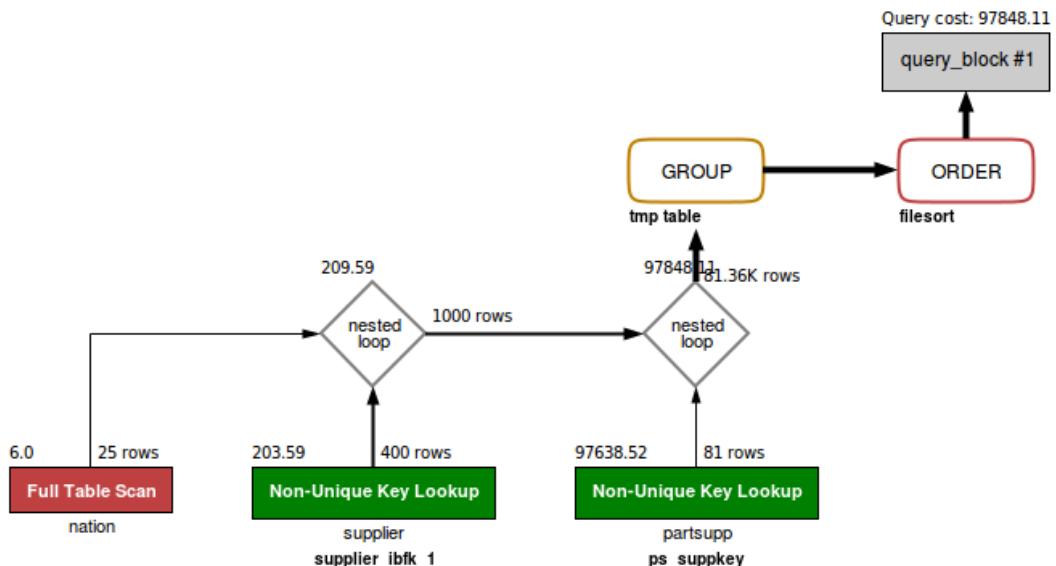


Figura B.22 – Plano de execução da consulta 11 com o melhor indivíduo.

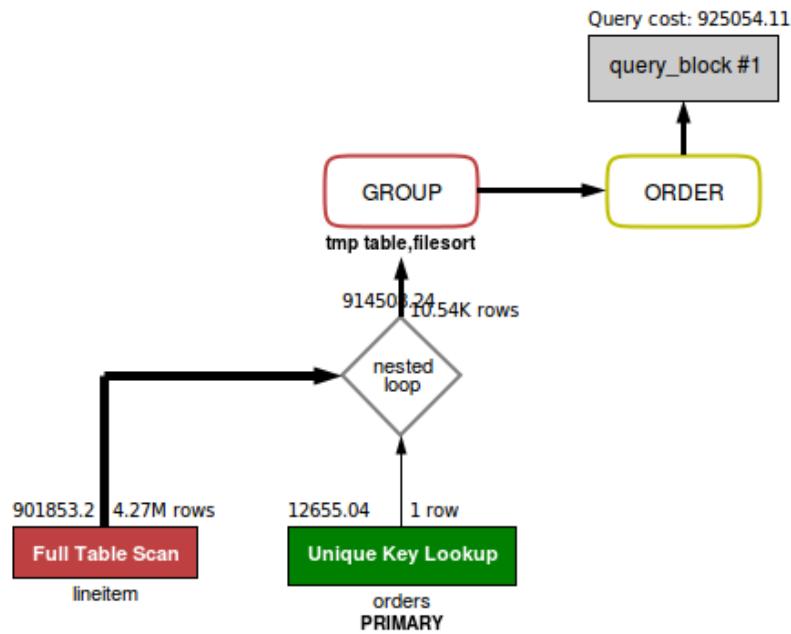


Figura B.23 – Plano de execução da consulta 12 no estado inicial do esquema (somente com PK e FK).

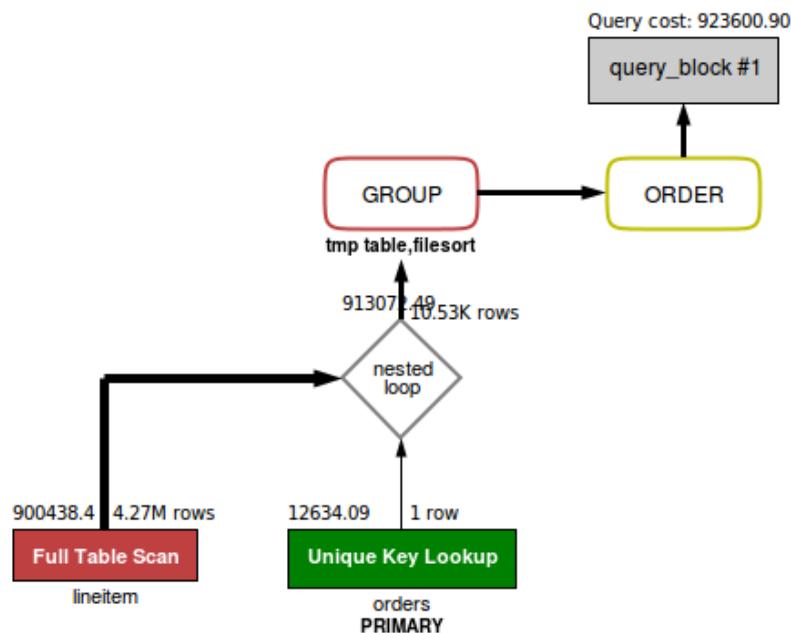


Figura B.24 – Plano de execução da consulta 12 com o melhor indivíduo.

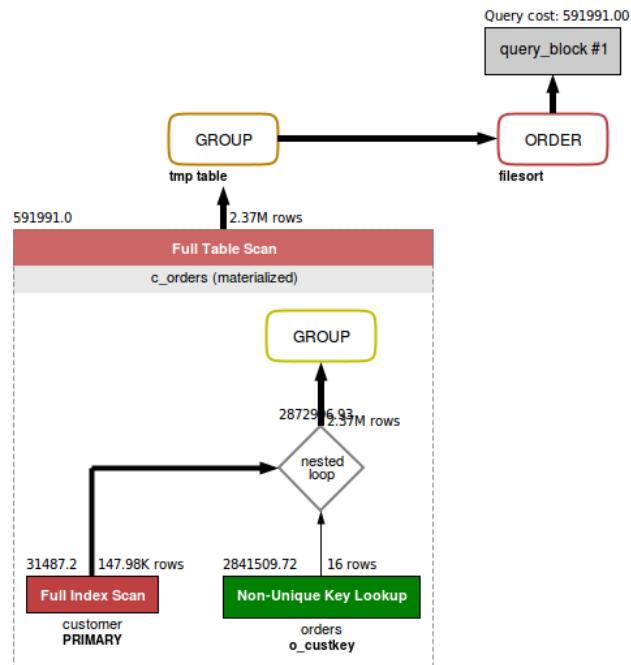


Figura B.25 – Plano de execução da consulta 13 no estado inicial do esquema (somente com PK e FK).

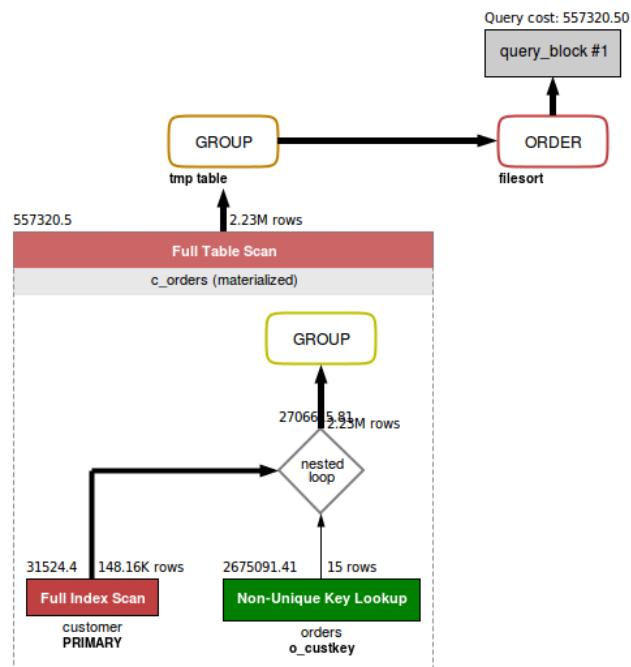


Figura B.26 – Plano de execução da consulta 13 com o melhor indivíduo.

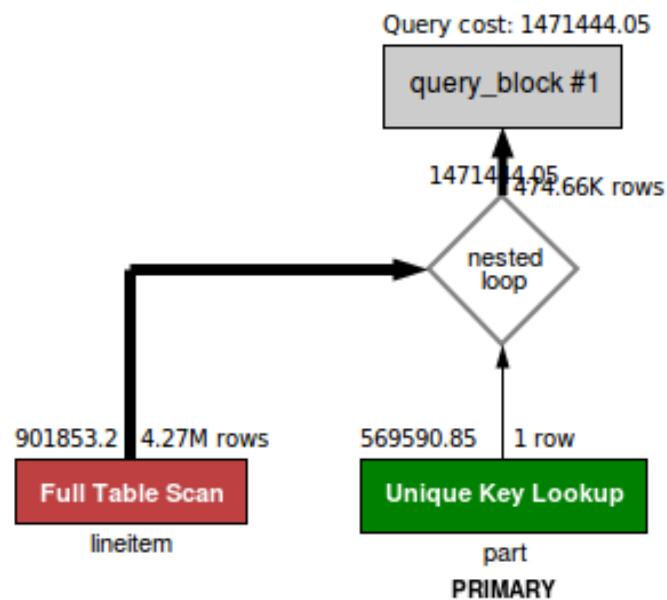


Figura B.27 – Plano de execução da consulta 14 no estado inicial do esquema (somente com PK e FK).

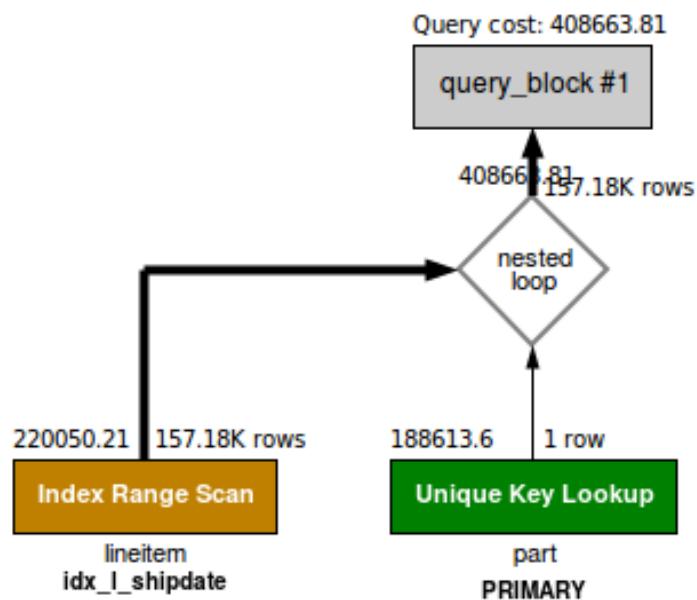


Figura B.28 – Plano de execução da consulta 14 com o melhor indivíduo.

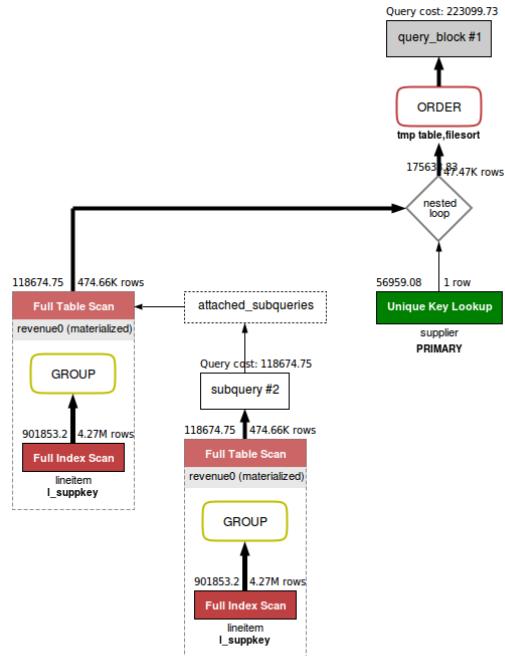


Figura B.29 – Plano de execução da consulta 15 no estado inicial do esquema (somente com PK e FK).

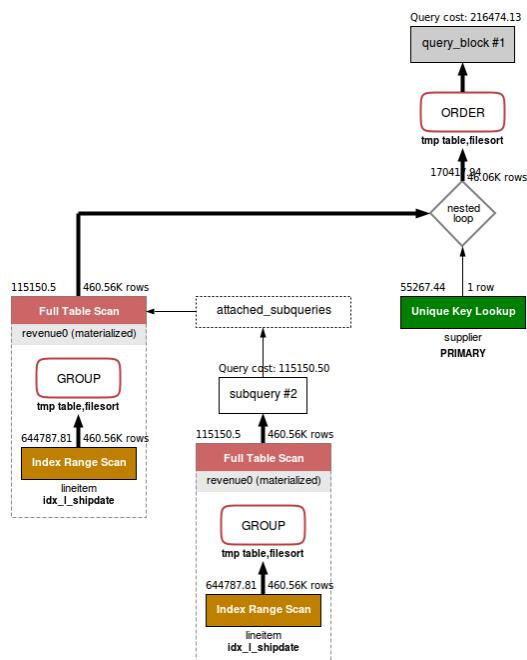


Figura B.30 – Plano de execução da consulta 15 com o melhor indivíduo.

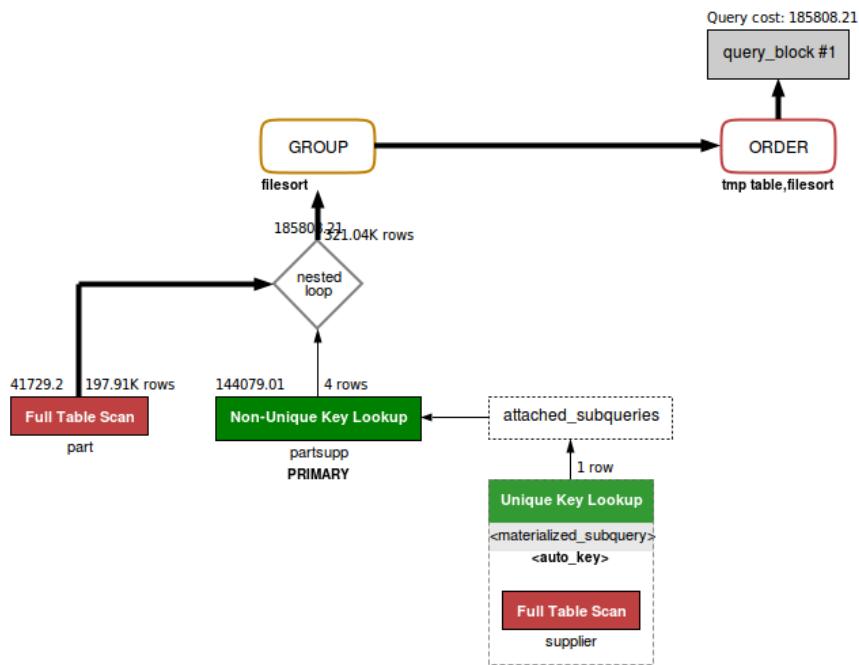


Figura B.31 – Plano de execução da consulta 16 no estado inicial do esquema (somente com PK e FK).

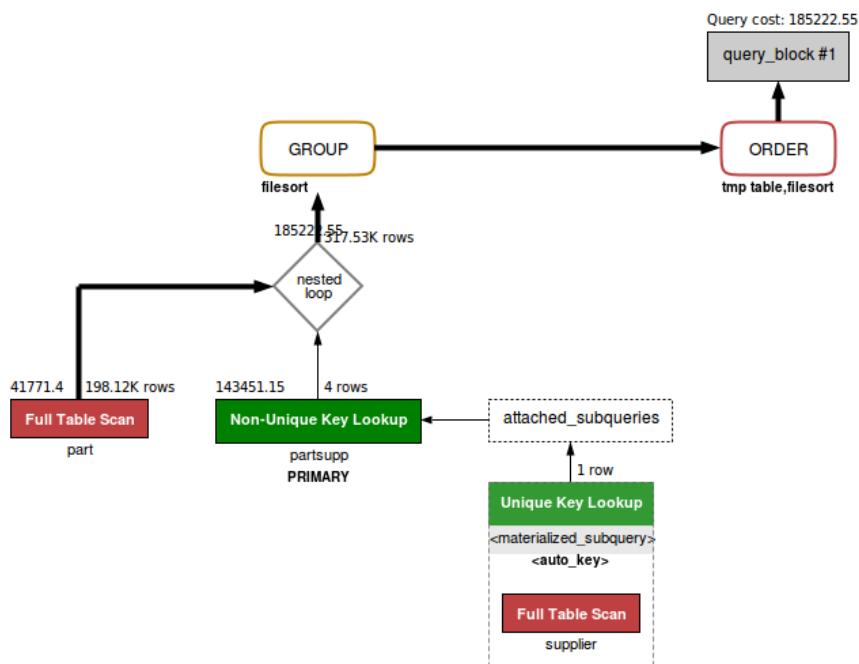


Figura B.32 – Plano de execução da consulta 16 com o melhor indivíduo.

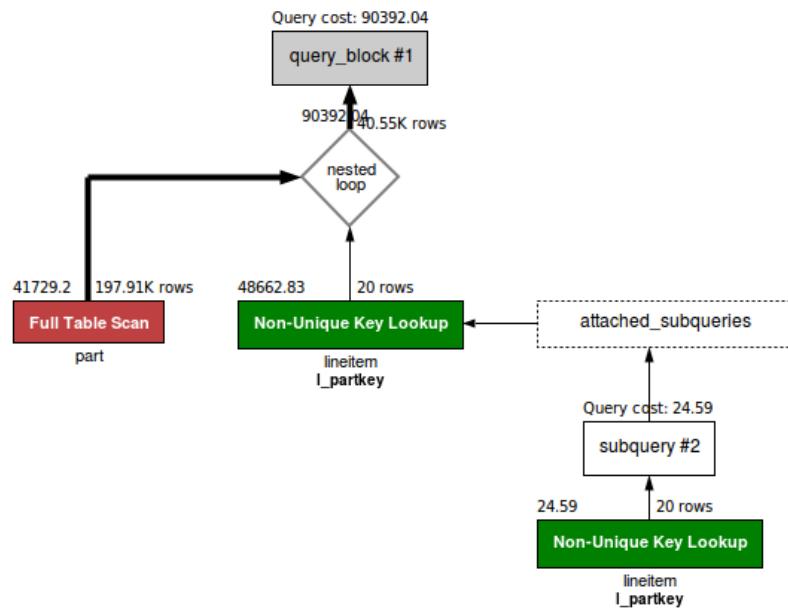


Figura B.33 – Plano de execução da consulta 17 no estado inicial do esquema (somente com PK e FK).

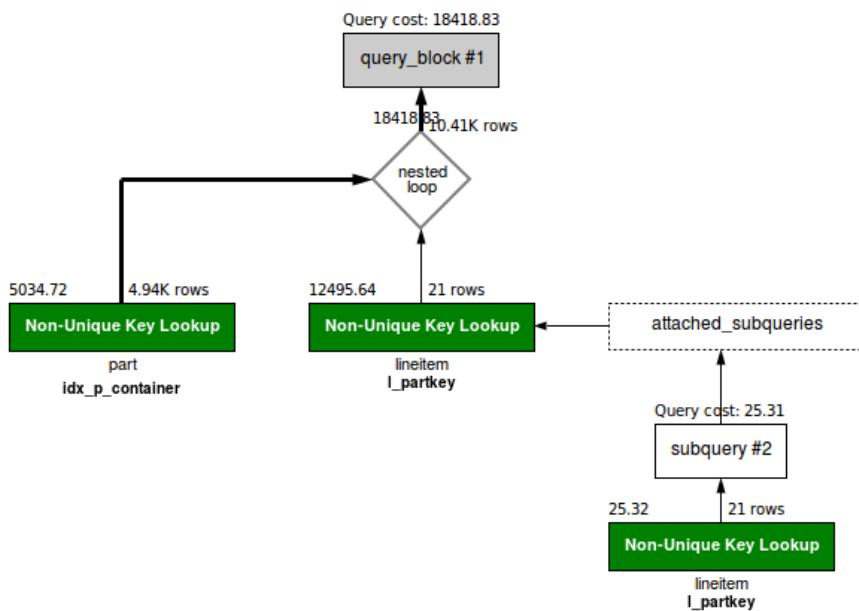


Figura B.34 – Plano de execução da consulta 17 com o melhor indivíduo.

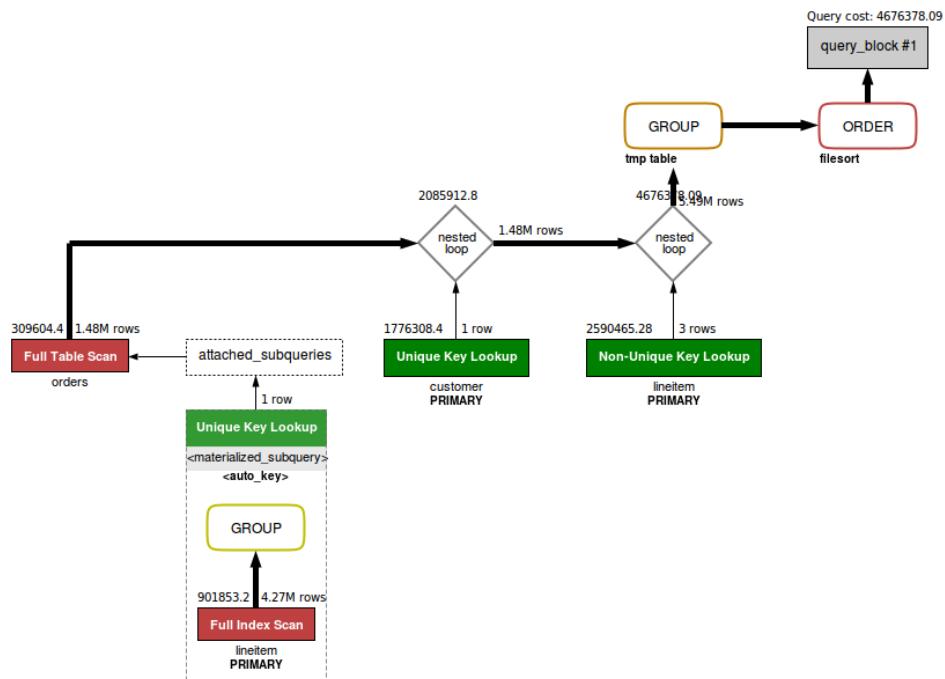


Figura B.35 – Plano de execução da consulta 18 no estado inicial do esquema (somente com PK e FK).

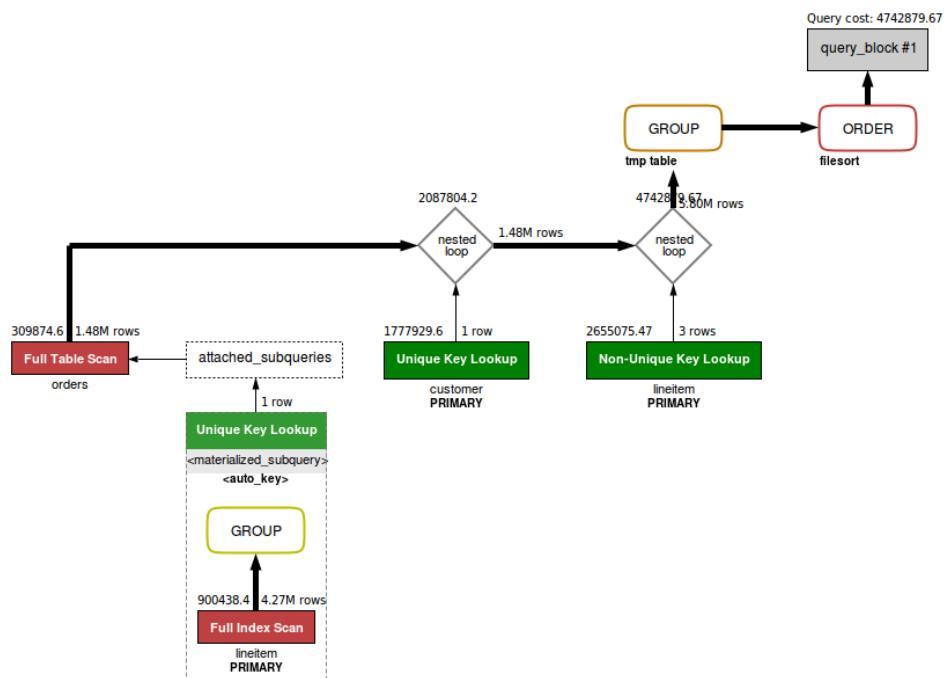


Figura B.36 – Plano de execução da consulta 18 com o melhor indivíduo.

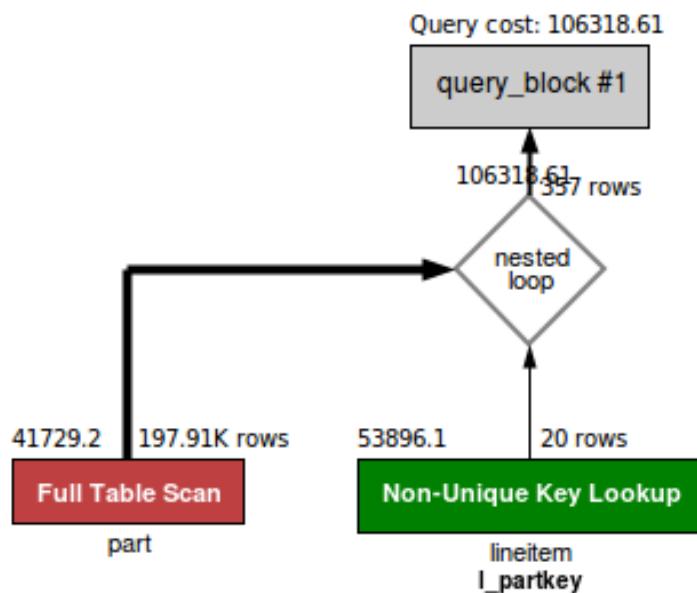


Figura B.37 – Plano de execução da consulta 19 no estado inicial do esquema (somente com PK e FK).

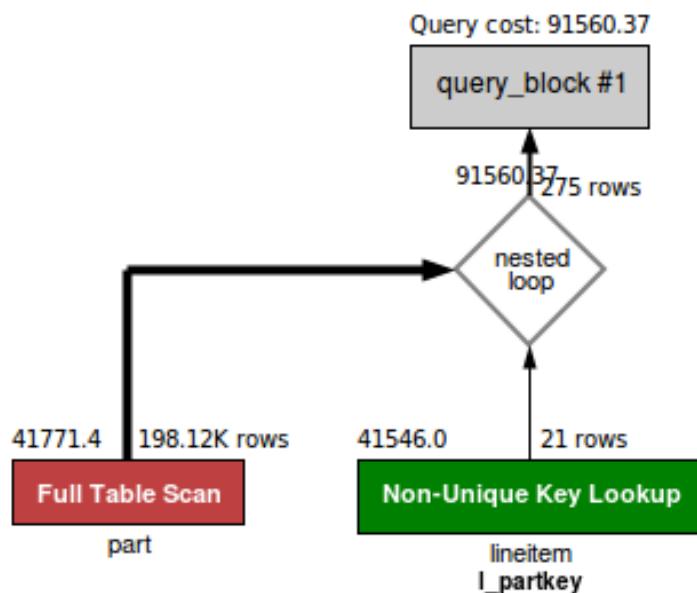


Figura B.38 – Plano de execução da consulta 19 com o melhor indivíduo.

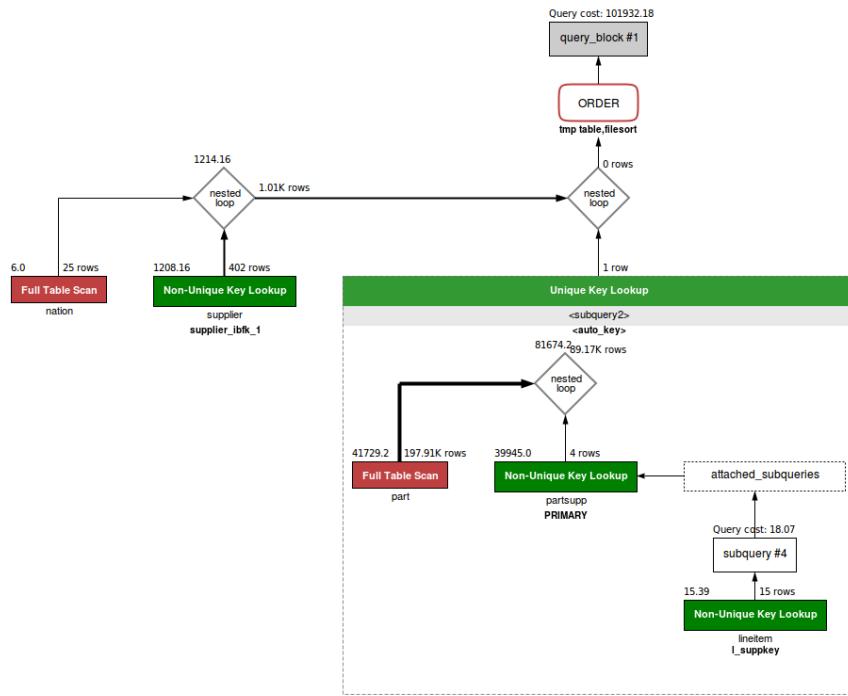


Figura B.39 – Plano de execução da consulta 20 no estado inicial do esquema (somente com PK e FK).

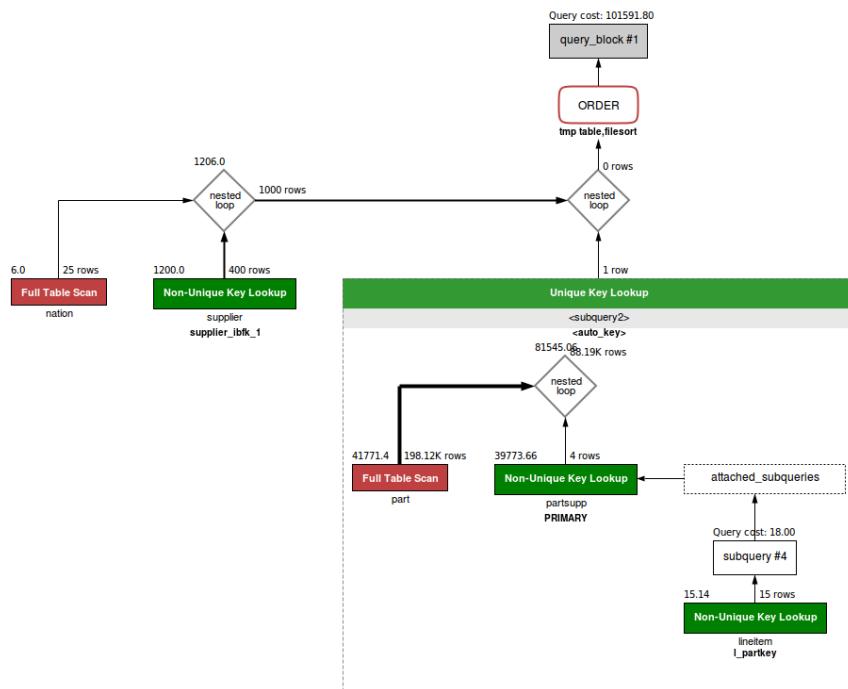


Figura B.40 – Plano de execução da consulta 20 com o melhor indivíduo.

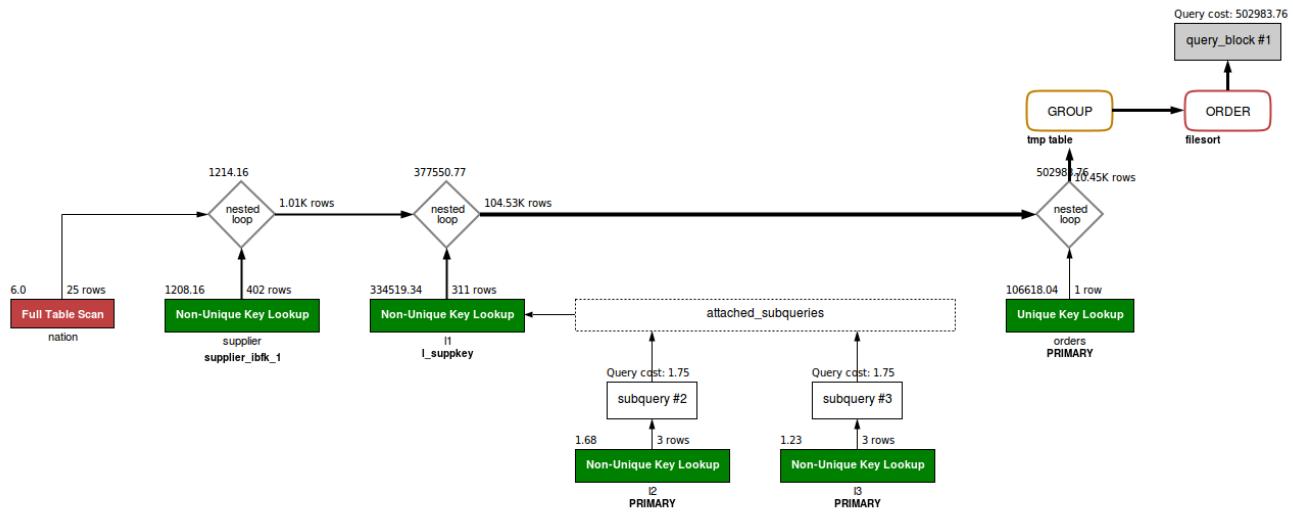


Figura B.41 – Plano de execução da consulta 21 no estado inicial do esquema (somente com PK e FK).

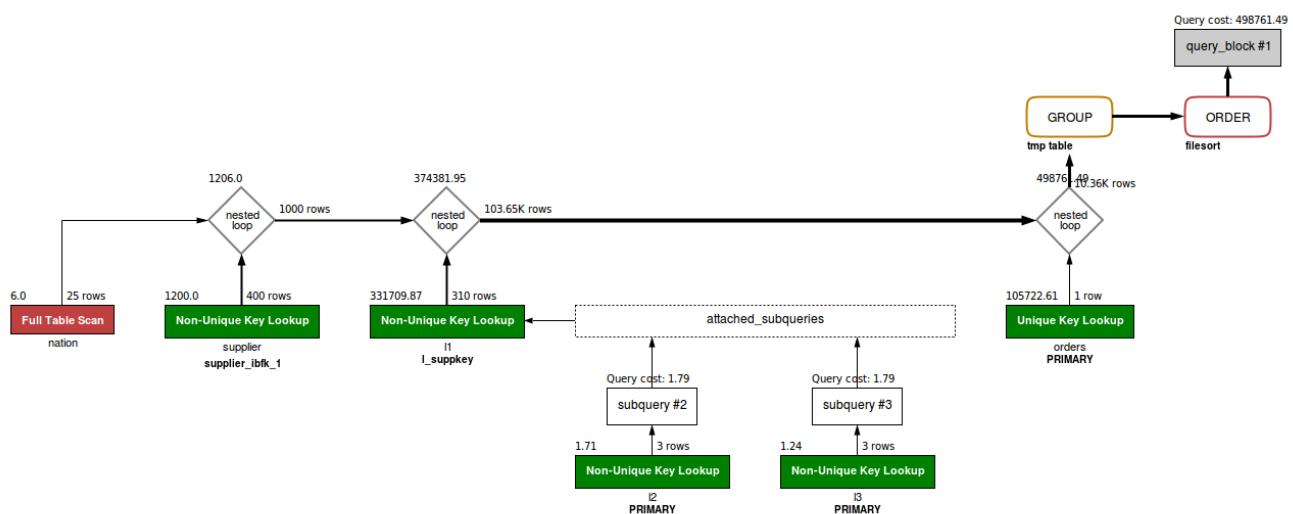


Figura B.42 – Plano de execução da consulta 21 com o melhor indivíduo.

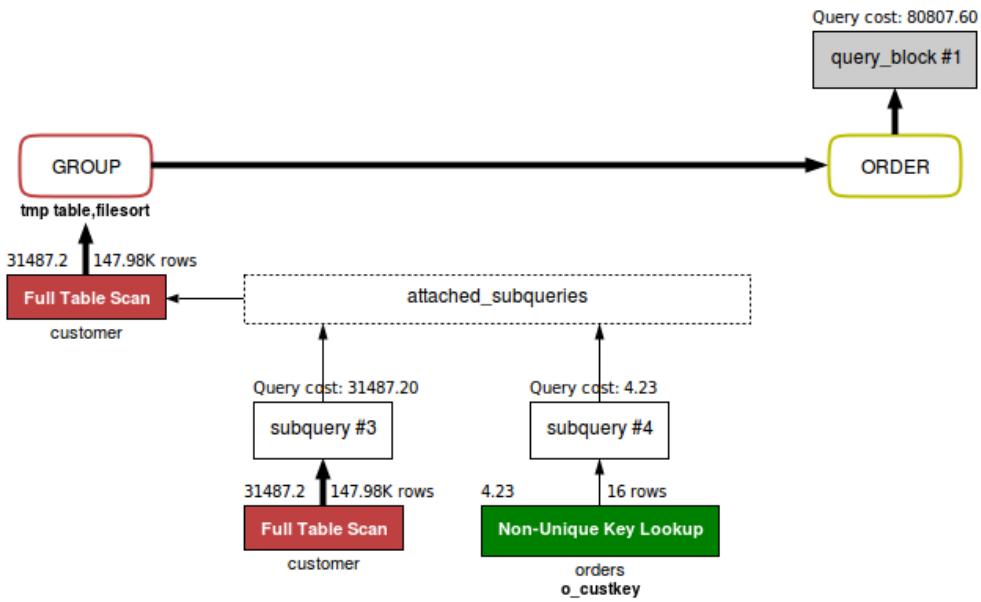


Figura B.43 – Plano de execução da consulta 22 no estado inicial do esquema (somente com PK e FK).

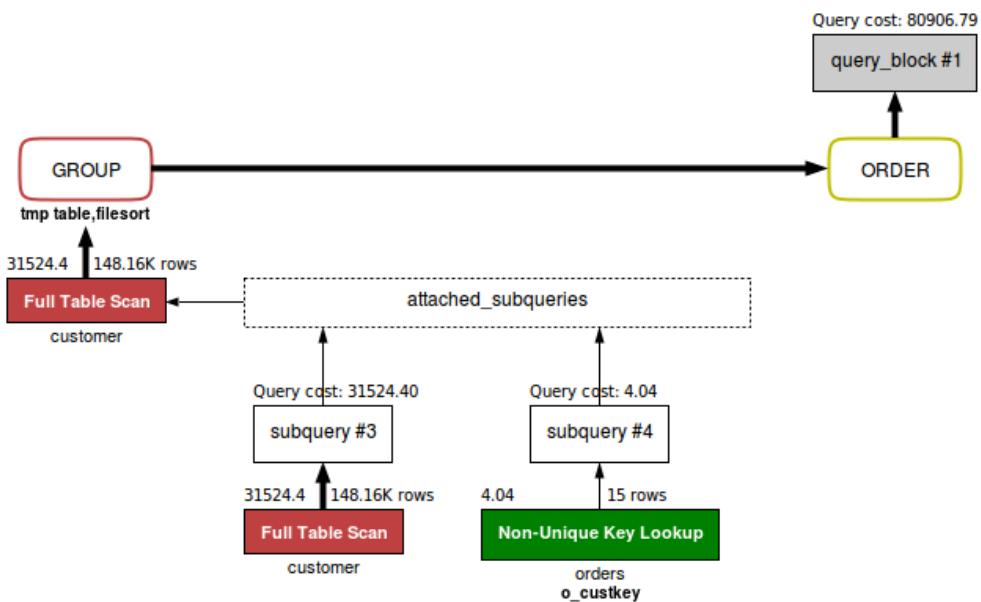


Figura B.44 – Plano de execução da consulta 22 com o melhor indivíduo.