UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

LEONARDO TEIXEIRA QUEIROZ

# Um *Benchmark* para Avaliação de Técnicas de Busca no Contexto de Análise de Mutantes SQL

Goiânia
2013

**TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR AS TESES E DISSERTAÇÕES ELETRÔNICAS (TEDE) NA BIBLIOTECA DIGITAL DA UFG**

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), sem ressarcimento dos direitos autorais, de acordo com a Lei nº 9610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou *download*, a título de divulgação da produção científica brasileira, a partir desta data.

**1. Identificação do material bibliográfico**: **[X] Dissertação** **[ ] Tese**

**2. Identificação da Tese ou Dissertação**

| | | | | | | |
|---|---|---|---|---|---|---|
| Autor (a): | Leonardo Teixeira Queiroz | | | | | |
| E-mail: | leonardo.queiroz@gmail.com | | | | | |
| Seu e-mail pode ser disponibilizado na página? [X]Sim [ ] Não | | | | | | |
| Vínculo empregatício do autor | SEGPLAN-GO | | | | | |
| Agência de fomento: | Fundação de Amparo à Pesquisa do Estado de Goiás | | | Sigla: | FAPEG | |
| País: | Brasil | UF: | GO | CNPJ: | 08.156.102/0001-02 | |
| Título: | Um *Benchmark* para Avaliação de Técnicas de Busca no Contexto de Análise de Mutantes SQL. | | | | | |
| Palavras-chave: | Testes de Aplicações de Banco de Dados; Análise de Mutantes SQL; Redução de Bases de Dados; Benchmark | | | | | |
| Título em outra língua: | A Benchmark to Evaluation of Search Techniques in the Context of SQL Mutation Analysis. | | | | | |
| Palavras-chave em outra língua: | Data Base Aplicattion Test; SQL Mutation Analysis; Data Reduction; Benchmark. | | | | | |
| Área de concentração: | Metodologia e Técnicas de Computação - Engenharia de Software - Teste de Software | | | | | |
| Data defesa: (dd/mm/aaaa) | 02/08/2013 | | | | | |
| Programa de Pós-Graduação: | Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás | | | | | |
| Orientador (a): | Prof. Dr. Cássio Leonardo Rodrigues | | | | | |
| E-mail: | cassio@inf.ufg.br | | | | | |
| Co-orientador (a):* | Prof. Dr. Celso Gonçalves Camilo Júnior | | | | | |
| E-mail: | celso@inf.ufg.br | | | | | |

*Necessita do CPF quando não constar no SisPG

**3. Informações de acesso ao documento:**

Concorda com a liberação total do documento [ X ] SIM [ ] NÃO[1]

Havendo concordância com a disponibilização eletrônica, torna-se imprescindível o envio do(s) arquivo(s) em formato digital PDF ou DOC da tese ou dissertação.

O sistema da Biblioteca Digital de Teses e Dissertações garante aos autores, que os arquivos contendo eletronicamente as teses e ou dissertações, antes de sua disponibilização, receberão procedimentos de segurança, criptografia (para não permitir cópia e extração de conteúdo, permitindo apenas impressão fraca) usando o padrão do Acrobat.

_____ Data: _____ / _____ / _____

Assinatura do (a) autor (a)

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. A extensão deste prazo suscita justificativa junto à coordenação do curso. Os dados do documento não serão disponibilizados durante o período de embargo.

LEONARDO TEIXEIRA QUEIROZ

# Um *Benchmark* para Avaliação de Técnicas de Busca no Contexto de Análise de Mutantes SQL

Dissertação apresentada ao Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Mestre em Computação.

**Área de concentração:** Metodologia e Técnicas de Computação - Engenharia de Software - Teste de Software.

**Orientador:** Prof. Dr. Cássio Leonardo Rodrigues

**Co-Orientador:** Prof. Dr. Celso Gonçalves Camilo Júnior

Goiânia
2013

**Dados Internacionais de Catalogação na Publicação (CIP)**
**GPT/BC/UFG**

Queiroz, Leonardo Teixeira.
Q384b Um benchmark para avaliação de técnicas de busca no contexto de análise de mutantes SQL [manuscrito] / Leonardo Teixeira Queiroz. - 2013.
171 f. : il., tabs.

Orientador: Prof. Dr. Cássio Leonardo Rodrigues; Co-orientador: Prof. Dr. Celso Gonçalves Camilo Júnior.
Dissertação (Mestrado) - Universidade Federal de Goiás, Instituto de Informática, 2013.
Bibliografia.
Inclui lista de figuras e tabelas.
Apêndices.

1. Banco de dados – Teste de aplicações. 2. Bases de dados – Redução. I. Título.

CDU: 004.415.535

Leonardo Teixeira Queiroz

**Um *Benchmark* para Avaliação de Técnicas de Busca no Contexto de Análise de Mutantes SQL**

Dissertação defendida no Programa de Pós-Graduação do Instituto de Informática da Universidade Federal de Goiás como requisito parcial para obtenção do título de Mestre em Ciência da Computação, aprovada em 02 de agosto de 2013, pela Banca Examinadora constituída pelos professores:

---

**Prof. Dr. Cássio Leonardo Rodrigues**
Instituto de Informática - UFG
Presidente da Banca

---

**Prof. Dr. Celso Gonçalves Camilo Júnior**
Instituto de Informática – UFG

---

**Prof. Dr. Plínio de Sá Leitão Júnior**
Instituto de Informática – UFG

---

**Prof. Dr. Arilo Cláudio Dias Neto**
Universidade Federal do Amazonas

**Leonardo Teixeira Queiroz**

Graduou-se em Análise de Sistemas pela Universidade Salgado de Oliveira, Campus Goiânia, recebendo o prêmio "Destaque Acadêmico Marlene Salgado de Oliveira 2º Lugar". Como bolsista na mesma instituição especializou-se em Tecnologia da Informação e Comércio Eletrônico com Habilitação para Docência Universitária. Durante o Mestrado foi bolsista da FAPEG - Fundação de Amparo a Pesquisa do Estado de Goiás. Já atuou como docente universitário e como analista de sistemas em empresas públicas e privadas. Atualmente é Gestor de Tecnologia da Informação da Secretaria de Gestão e Planejamento do Estado de Goiás.

Dedico este trabalho à memória de meu saudoso pai, Odilio Queiroz de Oliveira.

# Agradecimentos

Agradeço a Deus pela oportunidade, pelos desafios e pela força para superá-los. Aos meus pais, Maria e Odilio, fica meu imenso agradecimento pela dedicação, inspiração e exemplos. Não tenho dúvida que as minhas conquistas também são conquistas deles. Agradeço a minha amiga e esposa, Aline, pelo amor, companherismo, ajuda e paciência durante a realização do mestrado. Seu carinho e incentivos foram fundamentais para me dar forças nas horas certas. Agradeço, também, a todos de minha família, principlamente aos meus afilhados e sobrinhos, que souberam entender meus momentos de ausência.

Agradeço ao professor Plínio por ter me recebido no programa de pós-graduação e por me acompanhar durante toda minha jornada neste mestrado, sempre ajudando e orientando nos problemas mais difíceis. Ao professor Cássio agradeço pelos ensinamentos, pela orientação e pelas palavras de serenidade nos momentos certos. Ao professor Celso agradeço pela dedicação e colaboração com a pesquisa, fazendo grande diferença no resultado final. Ao professor Auri, agradeço pela ajuda e conhecimento que enriqueceram nosso trabalho.

Agradeço aos meus amigos Max, Gilmar e Luiz Gonzaga, por toda ajuda que recebi para realizar o mestrado, mas principalmente agradeço pelo incentivo, pelos "puxões de orelha" e pelas palavras de motivação. Também agradeço muito aos meus novos amigos e companheiros de mestrado: Adailton, Ana Cláudia, Jorge, Alex e Sandino. Foram amizades que conquistei durante este período e todos, de uma forma ou outra, contribuiram para o meu crescimento acadêmico e pessoal. Em especial agradeço ao amigo André, que com muita sabedoria me ajudou a enfrenter as últimas dificuldades do processo de construção da dissertação.

Agradeço também a todos colegas de trabalho que me incentivaram e "seguraram as pontas" quando precisei me ausentar. Aos chefes e gerentes que foram compreensivos, e à SEGPLAN pela minha liberação para frequentar as aulas.

Agradeço à Fundação de Amparo à Pesquisa do Estado de Goiás (FAPEG) pelo suporte financeiro a esta pesquisa.

"O homem é do tamanho do seu sonho."

**Fernando Pessoa**,
*Poeta, filósofo e escritor português*.

# Resumo

Teixeira Queiroz, Leonardo. **Um *Benchmark* para Avaliação de Técnicas de Busca no Contexto de Análise de Mutantes SQL**. Goiânia, 2013. 171p. Dissertação de Mestrado. Instituto de Informática, Universidade Federal de Goiás.

Uma das preocupações no teste de Aplicações de Bancos de Dados (ABD) é manter o custo operacional e computacional baixo. No contexto das ABD, uma das maneiras de colaborar com essa premissa é garantir que as bases de dados de teste (BDT) sejam pequenas, porém, eficazes na revelação de defeitos de instruções SQL. Tais bases podem ser construídas ou obtidas pela redução de grandes bases de dados de produção (BDP). No caso da redução, estão envolvidos aspectos combinatórios que exigem o uso de alguma técnica para a sua realização. Neste contexto, em resposta a uma carência identificada na literatura, o presente trabalho tem como objetivo construir e disponibilizar um *benchmark* para possibilitar a avaliação de desempenho, utilizando a Análise de Mutantes SQL, de qualquer técnica de busca que se proponha a realizar reduções de bases de dados. Sendo assim, para exercitar as técnicas de busca, o *benchmark* foi construído com dois cenários, onde cada um é composto por uma BDP e um conjunto de instruções SQL. Além disso, como uma referência para as técnicas de busca, ele é composto também por resultados de desempenho de bases de dados reduzidas aleatoriamente. Como objetivo secundário deste trabalho, a partir dos experimentos conduzidos na construção do *benchmark*, foram feitas análises dos resultados obtidos para responder importantes questões sobre quais fatores estão envolvidos na complexidade de instruções SQL no contexto da Análise de Mutantes. Uma das principais conclusões neste sentido foi sobre a restritividade dos comandos SQL, sendo este o fator que mais influencia na complexidade das instruções.

**Palavras–chave**

Teste de Aplicações de Banco de Dados; Análise de Mutantes SQL; Redução de Bases de Dados; *Benchmark*.

# Abstract

Teixeira Queiroz, Leonardo. ***A Benchmark to Evaluation of Search Techniques in the Context of SQL Mutation Analysis***. Goiânia, 2013. 171p. MSc. Dissertation. Instituto de Informática, Universidade Federal de Goiás.

*One of the concerns in test Applications Database (ADB) is to keep the operating and computational costs low. In the context of the ADB, one way to collaborate with this assumption is ensuring that the Test Databases (TDB) are small, but effective in revealing defects of SQL statements. Such bases can be constructed or obtained by the reduction of Production Databases (PDB). In the reductions case, there are combinatorial aspects involved that require the use of a specific technique for their implementation. In this context, in response to a deficiency identified in the literature, this work aims to build and provide a benchmark to enable performance evaluation, using SQL Mutation Analysis, any search technique that intends to conduct databases reductions. Therefore, to exercise the search techniques, the benchmark was built with two scenarios where each one is composed of a PDB and a set of SQL statements. In addition, as a reference for search techniques, it also contains performance of data database randomly reduced. As a secondary objective of this work, from the experiments conducted in the construction of the benchmark, analyses were made with the results obtained to answer important questions about what factors are involved in the complexity of SQL statements in the context of Test Mutation. A key finding in this regard was on the restrictiveness of SQL commands, and this is the factor that most influences the complexity of statements.*

**Keywords**

*Data Base Aplicattion Test; SQL Mutation Analysis; Data Reduction; Benchmark.*

# Sumário

# Lista de Figuras

# Lista de Tabelas

CAPÍTULO 1

# Introdução

*"Se houver uma vontade, haverá um caminho."*
*Witness Lee*

Nas últimas décadas, a Engenharia de Software tornou-se uma importante área de conhecimento para a sociedade. Grandes avanços sociais, econômicos e tecnológicos só estão sendo possíveis graças à disseminação e uso de softwares, cada vez mais complexos, em diversos segmentos.

Um dos objetivos e desafios da Engenharia de Software é garantir que os seus produtos tenham qualidade aceitável e funcionamento correto, independente da complexidade envolvida. Para isso são empregadas inúmeras atividades, métodos e técnicas durante o processo de desenvolvimento do software.

Dentro deste processo, uma das áreas para garantir a qualidade é conhecida como Verificação e Validação. Esta área é composta por diversas atividades, sendo uma das mais utilizadas e importantes o Teste de Software, que tem como principal objetivo revelar a presença de falhas [2].

Existem diversas técnicas para realizar a atividade de teste de software. As três principais técnicas utilizadas são: (i) Técnica Funcional (caixa preta): o software é avaliado considerando a sua especificação de requisitos; (ii) Técnica Estrutural (caixa branca): são usados os aspectos estruturas da implementação para a realização do teste; e (iii) Técnica Baseada em Erros: são considerados os erros mais comuns durante o desenvolvimento de software para conduzir a atividade de teste.

Porém, independente da técnica utilizada, um dos grandes problemas da atividade de teste é o custo. Ela é uma das mais onerosas dentro do desenvolvimento de software [40]. Sendo assim, o grande desafio é adotar critérios e estratégias para conduzir a

atividade de teste que consiga manter um custo reduzido mas sem prejuízos na sua capacidade de revelação de defeitos, com isso garantindo a qualidade esperada do software.

Neste sentido, desde a década de 70, inúmeros estudos e trabalhos vem sendo realizados com o objetivo de minimizar o custo do teste de software nos diferentes contextos onde ele está inserido. Várias técnicas, abordagens e estratégias já foram propostas e ainda hoje esta é uma área de pesquisa muito importante e com diversas contribuições sendo publicadas todos os anos.

Indo ao encontro dessas iniciativas, o presente trabalho visa contribuir com os esforços atuais para a redução de custos na área de Teste de Software, mais especificamente em softwares que envolve persistência de dados, o qual são denominados Aplicações de Bancos de Dados (ABD).

## 1.1 Contexto e Motivação

Para a realização de testes em Aplicações de Banco de Dados são necessários casos de teste[1] formados por instâncias de dados, conhecidas como Bases de Dados de Teste (BDTs). Um dos objetivos das BDTs é evidenciar possíveis defeitos presentes nas instruções SQL das ABD.

Espera-se que uma BDT tenha tamanho reduzido, impactando positivamente no custo da aplicação dos testes, e que tenha um conteúdo adequado, possibilitando que defeitos das instruções SQL sejam identificados. Quando uma BDT não tem qualidade, ou seja, não é adequada para a revelação de defeitos, o teste das instruções SQL fica comprometido. Uma forma de atestar a qualidade de uma BDT é usando a Análise de Mutantes SQL[2].

Sendo assim, as BDTs devem ser construídas considerando estes dois fatores: tamanho e conteúdo. Uma forma de realizar esta construção é através da seleção de subconjuntos de dados a partir de uma Base de Dados de Produção (BDP).

A redução de uma BDP, para formar uma BDT pequena e com dados adequados, pode ter um custo computacional elevado dependendo da complexidade envolvida, tornando este processo (redução de bases de dados) um problema relevante. A complexi-

[1] Um caso de teste define (formalmente) um conjunto específico de valores de entrada para o teste, as condições sob as quais o teste deve ser executado e os resultados esperados pela execução.

[2] Análise de Mutantes SQL é uma técnica de teste baseado em erros, específico para aplicações de banco de dados, onde consideram-se os erros mais frequentes durante o desenvolvimento de instruções SQL para orientar na definição dos casos de teste.

dade deste problema está relacionada diretamente com as características da BDP (volume e dados), das instruções SQL sendo testadas e dos seus mutantes.

Uma alternativa para tratar este problema é o uso de técnicas de busca que visam encontrar boas soluções (BDTs pequenas e adequadas) dentro do espaço de busca (BDP). A aplicação de técnicas de busca para resolver problemas da Engenharia de Software é chamada de *Search Based Software Engineering* (SBSE) [25]. Qualquer técnica que seja adaptada ou construída para tratar o problema de redução, vai em algum momento precisar fazer uso de um ambiente para atestar o seu funcionamento e avaliar o seu desempenho.

Na presente pesquisa, o critério de avaliação da qualidade das BDTs está voltada para técnicas que usem a Análise de Mutantes. Dessa forma, o ambiente deve conter no mínimo: (i) Uma BDP de grande volume; (ii) Instruções SQL para testes; (iii) Os mutantes das instruções; e (iv) calcúlo de escore de mutação de cada instrução SQL usando a BDP como dado de entrada.

Além da demanda de tempo e esforço para construir um ambiente com essas características, existem outros problemas que devem ser considerados sobre os diferentes ambientes que cada pesquisador pode usar para aplicar uma técnica de busca. Os mais relevantes para este trabalho são: (i) Uso de ambientes sem cenários desafiadores o suficiente para exercitar as técnicas de busca; e (ii) Uso de ambientes não padronizados, inviabilizando comparações e avaliações de desempenho entre as técnicas de busca.

Portanto, tais problemas são fortes argumentos para justificar a construção e disponibilização de um *benchmark* para funcionar como um ambiente padrão de referência na avaliação de desempenho de técnicas de busca para redução de bases de dados.

## 1.2 Objetivos e Contribuições

O **principal objetivo** da pesquisa consiste na construção e na disponibilização de um *benchmark*[3], que funcione como um ambiente de referência a fim de contribuir no fornecimento de todos os elementos necessários para avaliações de desempenho das técnicas de busca, no contexto de redução de bases de dados para testes de instruções SQL.

[3]Por definição, considera-se um *benchmark* como sendo a composição de uma estrutura (cenário) juntamente com um conjunto de instâncias de problemas. Além disso ele também é composto por alguns resultados gerados por métodos aleatórios a fim de fornecer uma referência padrão de complexidade para cada instância de problema.

O ambiente deste *benchmark* deve ser padronizado e possuir instâncias de problemas invariáveis e bem definidas. Espera-se também que ele tenha uma estrutura com grande volume de dados formado por uma ou mais BDPs e um conjunto de instruções SQL com seus respectivos mutantes criados a partir da ferramenta *SQLMutation* [48]. Todos estes componentes do *benchmark* serão disponibilizados fisicamente através de entregáveis[4] organizados para facilitar o seu entendimento e uso.

Como **objetivo secundário**, mas não menos importante, serão realizadas para cada instância de problema *(BDP, Instrução SQL)* do *benchmark*, diversos experimentos que visam conduzir reduções aleatórias da BDP a fim de gerar conjuntos de BDTs de diferentes tamanhos. A qualidade de cada BDT, e consequentemente de cada conjunto, será calculada e avaliada utilizando a Análise de Mutantes SQL.

A realização destes experimentos vai possibilitar a geração de diversas informações que indiquem a complexidade de todas as instruções SQL do *benchmark*, permitindo assim uma interpretação que nos leve à elaboração de entregáveis (planilhas de resultados, gráficos e principalmente tabelas de ranqueamento em diferentes perspectivas) que sirvam como importantes instrumentos de referências para as técnicas de busca. Complementarmente, outras interpretações dos resultados podem ser construídas para ajudar na identificação de quais situações e/ou fatores estão relacionados com a complexidade das instâncias de problemas.

Além disso, com todos os resultados e conclusões obtidos a partir do *benchmark* e dos experimentos realizados, espera-se, também, que pesquisadores da área de Análise de Mutantes SQL passem a contar com um conjunto considerável de informações para servir como insumos e parâmetros relevantes em diferentes outras pesquisas da área, sendo esta também uma importante contribuição do presente trabalho.

Por fim, outra contribuição consiste na disponibilização do ambiente e da ferramenta utilizados na construção do *benchmark* e realização dos experimentos. Além da BDP, este ambiente é também composto por uma base de dados usada para registrar os experimentos e seus resultados (BDE), e por uma base de dados para simular as BDT. Para facilitar o controle e manipulação destas bases, uma ferramenta para automatizar a execução dos experimentos foi projetada e implementada, fazendo as reduções aleatórias da BDP, manipulando as BDTs geradas e realizando os cálculos de escore de mutação.

Desta forma, a disponibilização do ambiente de construção, juntamente com a

[4] Além do documento de dissertação, todo material produzido e disponibilizado ao final da pesquisa é considerado como um entregável deste trabalho, visando complementar as contribuições do mesmo. Neste contexto, os principais entregáveis da pesquisa são os componentes do *benchmark* e os resultados obtidos a partir dos experimentos realizados.

ferramenta, configura-se como mais um entregável deste trabalho permitindo que novos experimentos, com novos cenários e instâncias de problemas, possam ser realizados em outras pesquisas.

## 1.3 Metodologia

Considerando os objetivos definidos, foram conduzidas 4 etapas para a execução e finalização do trabalho, sendo elas:

1. **Entendimento do escopo do problema** - O primeiro passo da pesquisa foi obter um entendimento em relação aos problemas envolvidos. A medida que este entendimento foi amadurecendo, trabalhos relacionados na área foram sendo pesquisados e avaliados. Para auxiliar nesta etapa, também foi realizada uma revisão bibliográfica sobre Teste de Software, Análise de Mutantes SQL e Técnicas de *Benchmark*.
2. **Definição das diretrizes e características do *benchmark*** - Após o entendimento do escopo do problema e da proposta de trabalho (construção de um *benchmark*), foram definidas as características e diretrizes para orientar a construção e evolução do *benchmark* proposto. Também nesta etapa foram estabelecidos os parâmetros e um processo sistemático de trabalho para a realização dos experimentos de reduções aleatórias.
3. **Construção Inicial do *Benchmark* e Realização dos Experimentos** - Seguindo as diretrizes e características estabelecidas, foram criados dois cenários para o *benchmark*. O primeiro cenário, a partir da geração de dados aleatórios e o segundo, utilizando dados de uma aplicação real. Também foram definidas as instruções SQL iniciais para cada cenário, assim como os seus mutantes. A medida que os experimentos foram realizados, novas instruções SQL foram elaboradas considerando os resultados obtidos. Um ambiente foi definido e implementado, juntamente com uma ferramenta desenvolvida em linguagem Java, para dar suporte na construção do *benchmark* e realização dos experimentos. Os experimentos foram executados de acordo com as diretrizes e parâmetros definidos. Estes parâmetros indicam o comportamento de cada experimento (instrução SQL, quantidade de BDTs, tamanho das BDTs). Todos os resultados foram registrados na base de dados de experimentos (BDE) para consultas futuras.
4. **Análise dos resultado e Geração dos Entregáveis** - Uma importante atividade do trabalho foi o tratamento e análise do grande volume de resultados gerados pelos experimentos. Nesta etapa, foram definidas quais informações seriam relevantes e importantes e a partir delas, construídos diversos entregáveis do *benchmark*. Além

disso, foi feita uma série de análises em cima dos resultados para identificar quais os fatores estão relacionados com o grau de dificuldade de uma instrução SQL no contexto da Análise de Mutantes.

## 1.4 Organização do Trabalho

O restante deste trabalho está organizado como descrito a seguir:

No **Capítulo 2** é apresentada a fundamentação teórica do trabalho que relaciona os conceitos de Técnica de *Benchmark*, a Análise de Mutantes e a Análise de Mutantes SQL.

No **Capítulo 3** são apresentados alguns trabalhos relacionados, detalhando suas característica e resultados obtidos. Também são apresentadas neste capítulo algumas discussões importantes, extraídas destes trabalhos, que ajudam a justificar a presente pesquisa.

No **Capítulo 4** são apresentados maiores explicações sobre os objetivos do *Benchmark*, sua composição, suas características, seus entregáveis, as diretrizes para a sua construção e o processo desenvolvido para a execução dos experimentos.

No **Capítulo 5** são detalhados os resultados obtidos a partir dos experimentos realizados com o método aleatório para a geração de BDTs. Estes resultados foram sintetizados, interpretados e materializados nos principais entregáveis deste trabalho, servindo como importantes instrumentos de referência para as técnicas de busca. Além disso, neste capítulo os resultados são analisados em diferentes perspectivas para responder questões relevantes dentro do contexto da Análise de Mutantes SQL.

O **Capítulo 6** contém as considerações finais do trabalho, com as principais contribuições e trabalhos futuros.

CAPÍTULO 2

# Fundamentação Teórica

*"Se você tem conhecimento, deixe os outros acenderem suas velas nele."*
*Margarete Fuller*

Este capítulo é iniciado com a apresentação de conceitos básicos sobre Análise de Desempenho e Técnica de *Benchmark*. Posteriormente é feita uma revisão bibliográfica sobre a Análise de Mutantes, abordando suas características, objetivos e processo. Para finalizar, o capítulo apresenta uma explicação geral sobre Análise de Mutantes SQL.

## 2.1 Análise de Desempenho

Avaliar, medir e comparar o desempenho de produtos que propõem realizar uma mesma tarefa é uma necessidade presente desde os primórdios da ciência da computação, sejam produtos do tipo *hardware* ou do tipo *software* [28].

Análise de desempenho é um dos mecanismos mais utilizados para realizar essas medições e comparações. Basicamente, esta abordagem mede a eficiência que os produtos efetuam determinadas tarefas. Tal eficiência é expressa através de alguma medida escalar (tempo, distância, tamanho, etc.) e, facilmente, pode-se realizar um ranqueamento dos produtos sendo avaliados usando operadores relacionais básicos (maior, menor e igual) por meio dessas medidas escalares. Na área da Ciência da Computação, dentre as subáreas que utilizam Análise de Desempenho destacam-se: Sistemas Operacionais; Arquitetura de Computadores; Teoria da Computação e Banco de Dados [13]. Neste cenário, espera-se que a técnica ajude a resolver questões do tipo:

- Avaliar a capacidade máxima de um produto;

- Comparar diferentes tecnologias;
- Avaliar a viabilidade de um produto para uma capacidade específica;
- Avaliar a relação custo x benefício de um produto.

A aplicação da Análise de Desempenho é realizada utilizando um dos seguintes modelos: Modelo Analítico; Modelo de Simulação ou Modelo Experimental. O Modelo Experimental obtém os resultados de desempenho utilizando o próprio produto sendo analisado. Duas técnicas principais fazem parte deste modelo: Monitoração e *Benchmark* [13].

A técnica de monitoração utiliza ferramentas de avaliação e/ou estatísitica disponíveis nos produtos em análise. Por não existir uma padronização nestas ferramentas, os resultados são específicos e com isso limitados. Outro agravante é que pela falta de padronização a técnica de certa forma inviabiliza a comparação de resultados gerados em diferente produtos. Por outro lado, a técnica de *benchmark* funciona executando um conjunto fixo e padronizado de tarefas sobre um produto para avaliar o seu desempenho [20] .

### 2.1.1 Técnica de *Benchmark*

Esta técnica é amplamente utilizada para avaliação de desempenho em sistemas computacionais [30]. Ela pode ser aplicada para comparar com precisão a eficiência de diferentes produtos, uma vez que é **padronizada** e os seus **testes são invariáveis** e **bem definidos** [5].

Segundo Berry et al. [4], a técnica de *benchmark* é geralmente utilizada para medir a performance de alguns aspectos de sistemas computacionais de forma sistemática. O rigor nas medições visa o controle do ambiente no qual as avaliações são realizadas.

O principal componente da técnica é um *benchmark*. Ele é composto por uma série representativa de testes funcionais e de desempenho, os quais são efetuados em um determinado subconjunto de dados, simulando assim um ambiente de aplicação alvo. De forma geral pretende-se medir o quão eficiente um sistema computacional efetua um determinado conjunto de tarefas[1] (testes) em um determinado cenário [13]. Segundo Ciferre [13], espera-se que um *benchmark* tenha as seguintes características:

- Ser **relevante** para a aplicação alvo a qual representa;

[1]Em alguns contextos, o *benchmark* é composto por um conjunto de instâncias de problemas, onde pretende-se medir a qualidade do resultado gerado para cada um deles.

- Ser **portátil** entre diferentes arquiteturas;
- Ser **escalável**, podendo ser executado em diferentes sistemas computacionais e;
- Ser, na medida do possível, **simples** de entender para manter a sua credibilidade.

No início a técnica de *benchmark* foi muito utilizada para avaliação de computadores. Com o passar do tempo, a técnica começou a ser utilizada para outros tipos de avaliações dentro da Ciência da Computação. É importante ressaltar a necessidade da técnica ser especializada de acordo com o propósito de sua utilização. Uma das especializações mais utilizadas são os *Benchmark*s de Bancos de Dados, onde uma base de dados (cenário) e um conjunto de transações (tarefas) são bem definidos e, posteriormente, as transações são executadas utilizando o(s) sistema(s) em avaliação para obter os resultados de desempenho [21].

Um dos primeiros *benchmark* para avaliar Sistemas de Gerenciamento de Banco de Dados (SGBD) foi o TPC-B, criado em 1990 pelo TCP (*Transaction Processing Performance Council*)[2]. Basicamente o TPC-B era usado para medir quantas transações por segundo um SGBD podia realizar. [14]

O TPC-B é considerado obsoleto desde 1995. Ele foi substituído pelo TPC-C que é um *benchmark* para OLTP (*Online Transaction Processing*), onde foram incorporados múltiplos tipos de transações e bases de dados mais complexas. O objetivo principal do TPC-C é definir um conjunto de requisitos funcionais, que podem ser executados em qualquer sistema OLTP independente do sistema operacional e do *hardware*. Ele simula um ambiente computacional completo, como se existissem vários usuários executando transações no banco de dados.

As duas principais métricas do TPC-C são a **tpmC** que mede o desempenho geral do processamento das transações. O seu valor representa quantas transações por minuto o SGBD gera em paralelo a todas as outras cargas de trabalho realizadas pelo *benchmark*. A outra métrica é a **$/tpmC** que mede a relação custo/performance do processamento. Além do TPC-C, também foram desenvolvidos pelo TPC outras especializações de *benchmark*. Sendo elas:

- **TPC-DS** - *Benchmark* para sistemas de suporte a decisão
- **TPC-E** - *Benchmark* para simular OLTP de um ambiente corporativo (empresa de corretora de seguros)
- **TPC-H** - *Benchmark* para sistemas de suporte a decisão *ad-hoc*

[2]O TPC é um comitê que define e mantém *benchmark*s de avaliação de desempenho de bancos de dados. Ele fornece resultados confiáveis como referências para a indústria. Seus principais membros são a Oracle, Microsoft, IBM, Intel, RedHat e HP.

- **TPC-VMS** - *Benchmark* para bancos de dados virtualizados

Um outro exemplo de *Benchmark* para Banco de Dados é o *ANSI SQL Standard Scalable and Portable* (AS3AP). Os objetivos definidos para a elaboração do AS3AP são ótimos exemplos de características que se esperam encontrar, de forma genérica, em *benchmark*s no geral [45]. Os objetivos do AS3AP :

- Fornecer um conjunto de testes para processamento de banco de dados que seja abrangente e, ao mesmo tempo, tratável;
- Ser escalável e portável, permitindo que ele seja utilizado para testar uma grande variedade de sistemas;
- Minimizar o esforço humano na implementação e execução de testes de *benchmark*;
- Fornecer uma métrica uniforme para uma relação de bancos de dados equivalentes, permitindo uma interpretação simples e sem ambiguidade dos resultados do *benchmark*;

Além de Bancos de Dados, outras especializações da técnica podem ser realizadas desde que o cenário e as tarefas do *benchmark* sejam planejadas para o fim específico. Outro ponto a ser definido para um *benchmark* são as métricas de avaliação de desempenho. Dessa forma, torna-se possível a avaliação do desempenho de qualquer tipo de produto e/ou solução que tenha capacidade de realizar as tarefas definidas em um *benchmark*.

## 2.2 Análise de Mutantes

O sucesso da atividade de testes em revelar defeitos em um programa está diretamente relacionado com a qualidade dos casos de teste usados para avaliar o programa. Segundo Maldonado et al. [32], avaliar os casos de teste é um ponto crucial na atividade de teste. O objetivo é conseguir identificar e projetar casos de teste que tenham maior poder de revelar uma grande quantidade de defeitos com o menor tempo e esforço possível.

Uma maneira de alcançar tal objetivo é utilizando técnicas de teste que auxiliam na avaliação e geração dos casos de teste, bem como na seleção daqueles que deverão ser utilizados para aumentar a probabilidade de revelar defeitos no programa testado com o menor custo possível. Um dos critérios utilizados para este fim é a **Análise de Mutantes** (*Mutation Analysis*), também conhecida como **Teste de Mutação** *(Mutation Testing)*.

A Análise de Mutantes é classificada como uma técnica de **Teste Baseado em Erros**, onde consideram-se os erros mais frequentes durante o desenvolvimento do

*software* para orientar na definição dos testes. Ela surgiu ainda na década de 70, e um dos primeiros trabalhos que detalham a ideia básica da Análise de Mutantes foi publicado em um artigo por DeMillo et al. [18] em 1978.

No artigo em questão, foi apresentado o conceito conhecido como **Hipótese do Programador Competente**, que é um dos fundamentos básicos da Análise de Mutantes. Esta hipótese considera que todo programador escreve códigos de boa qualidade, gerando produtos corretos (ou próximo disso). Partindo deste princípio, é assumido que os defeitos presentes nos programas escritos são pequenos enganos sintáticos cometidos pelos programadores. Estes pequenos enganos, embora não causem erros sintáticos, podem mudar a semântica do programa levando a um comportamento não esperado.

DeMillo et al. [18] também apresentam um outro conceito associado ao critério da Análise de Mutantes, que é o **Efeito de Acoplamento**. Ele considera que defeitos complexos estão relacionados com defeitos simples, como os cometidos pelos programadores competentes. Desta forma, revelar defeitos simples leva naturalmente à descoberta de defeitos maiores e mais complexos.

Partindo deste pressuposto, espera-se então que uma técnica como a Análise de Mutantes avalie um conjunto de casos de teste para verificar se os mesmos estão aptos para encontrar os menores e mais simples defeitos do programa, tendo como principal referência de busca os defeitos mais comuns cometidos pelos programadores. De acordo com Jia et al. [27], essa habilidade da Análise de Mutantes em mensurar e avaliar a aptidão (qualidade) dos casos de teste é expressa em um valor chamado **Score de Mutação** *(Mutation Score)* que, por sua vez, é um dos resultantes finais do processo de Análise de Mutantes.

### 2.2.1 Processo de Análise de Mutantes

De forma genérica, o processo de Análise de Mutantes funciona da seguinte maneira: são inseridas pequenas alterações no código de *P* (programa em teste) sendo que essas pequenas modificações são realizadas por meio de operadores de mutação, resultando em programas *P'* ligeiramente diferentes do original. Esses programas modificados são chamados de mutantes. Posteriormente verifica se tais modificações são perceptíveis pelo conjunto de casos de teste *T*, cuja adequação deseja-se avaliar. Quanto maior o número de modificações identificadas, maior é a qualidade de *T*.

Esta identificação é feita comparando o comportamento da execução do programa original *P* usando *T*, com o comportamento da execução de cada mutante *P'* também usando *T*. Se o comportamento do *P'* for diferente do comportamento de *P* diz-se

que o mutante está morto, ou seja, a modificação feita em *P'* foi identificada por *T*. Se o comportamento de *P* e *P'* forem iguais, diz-se que o mutante está vivo. Neste caso existem duas possibilidades: (i) *P'* é semanticamente equivalente a *P* ou; (ii) *T* não teve capacidade de revelar a modificação feita em *P'*. Geralmente é necessária uma avaliação humana para analisar e identificar o motivo na sobrevivência de um mutante.

O Processo de Análise de Mutantes descrito pode ser dividido nas seguintes macro etapas:

1. Geração dos Mutantes *P'*;
2. Execução do Programa *P* e de cada mutante *P'*;
3. Análise dos Mutantes.

**Etapa 1 - Geração dos mutantes P'**

A partir do programa *P* a ser testado, são feitas nele pequenas modificações sintáticas, gerando uma coleção de novos programas similares *P1, P2,...,Pn*. Cada novo programa (*P'*) gerado é um **mutante** de *P*. A geração de cada mutante é feita aplicando um **operador de mutação** *(mutant operator)* em um local/elemento de *P*.

A Tabela 2.1 apresenta um exemplo ilustrativo de um mutante *P'* gerado por um operador de mutação que alterou do sinal **>** presente no programa original *P*.

**Tabela 2.1:** *Exemplo simples de mutante*

| Programa original *P* | Mutante gerado *P'* |
|---|---|
| ...<br>*if (x > y) return true*<br>... | ...<br>*if (x < y) return true*<br>... |

Os operadores de mutação são regras que definem quais as alterações devem ser aplicadas no programa original *P* para gerar o conjunto de *P'* . Eles induzem mudanças sintáticas simples com base nos erros típicos cometidos pelos programadores (Hipótese do Programador Competente), com o intuito de inserir defeitos pequenos e simples que se identificados também revelam defeitos complexos (Efeito de Acoplamento).

Outro aspecto que orienta na definição e escolha dos operadores de mutação é a linguagem na qual o programa *P* foi escrito. Em geral, essa definição de operadores é uma das partes mais importantes para garantir eficiência com o uso da Análise de Mutantes. Um indício dessa assertiva, consiste na existência de vários esforços para aprimorar e otimizar o conjunto de operadores para linguagens específicas [3].

A princípio, um mutante é criado aplicando um operador de mutação por vez, neste caso são chamados de mutantes de ordem 1 ou mutantes de primeira ordem. Mutantes de mais alta ordem são criados aplicando mais de um defeito por vez, porém estes geralmente não são considerados por dois motivos: (i) Aumentam exponencialmente o número de mutantes, consequentemente aumentando o custo do teste; (ii) Mutantes de ordem mais alta não contribuem para melhoria significativa dos casos de teste, como foi comprovado nos estudos de Demilli e Offut [16].

Ao final, a quantidade de mutantes gerados vai depender diretamente de dois fatores: (i) Conjunto de operadores de mutação escolhidos para o processo de geração dos mutantes; (ii) Complexidade do programa sendo testado, sendo que quanto maior o número de elementos que estão no domínio dos operadores, maior vai ser a quantidade de mutantes.

**Etapa 2 - Execução do Programa *P* e de cada mutante *P'***

O primeiro passo desta etapa é executar o programa original *P* usando os casos de teste de *T*. Neste momento é verificado se o resultado de *P* está dentro do esperado (correto). Caso esteja, o processo de Análise de Mutantes pode continuar com a execução do conjunto de mutantes *P'*, usando o resultado de *P* como referência.

Cada mutante *P'* é executado com os dados de entrada do conjunto de casos de teste *T*. Se para qualquer caso de teste $t \in T$ o resultado do mutante *P'* for diferente do programa original *P*, diz-se que *P'* é morto pelo conjunto *T*. Se o resultado for igual, diz-se que o mutante continua vivo. Este processo é feito para todos mutantes *P'* gerados.

**Etapa 3 - Análise dos Mutantes**

Ao final da execução de todos os mutantes, alguns estarão mortos e outros não. Um mutante *P'* continua vivo quando, para todos os casos de teste de *T*, o resultado da sua execução é exatamente igual ao programa *P*. Se *P'*, apesar da mutação, sempre tiver resultado igual a *P*, então possivelmente *P'* é um **mutante equivalente**.

Neste momento pode ser necessária uma intervenção humana para definir se um mutante sobrevivente é ou não equivalente, comparando o código de *P* e *P'* para atestar a equivalência semântica de ambos. Caso sejam semanticamente iguais o mutante é classificado como equivalente. Porém, caso não se trate de um mutante equivalente, significa que *T* falhou em identificá-lo.

Na Tabela 2.2 é apresentado um exemplo simples de mutante equivalente. Apesar

de ter ocorrido uma mudança sintática (troca do operador ***menor*** pelo operador ***diferente***), o comportamento semântico do programa será o mesmo. A execução só sairá do laço *for* quando *i* for igual a 3, seja no programa original *P* ou no mutante equivalente *P'*.

**Tabela 2.2:** *Exemplo de mutante equivalente*

| Programa original *P* | Mutante equivalente *P'* |
|---|---|
| for ( *int i* = 0; *i* < 3; *i*++)<br>{<br>*bloco de comandos;*<br>} | for (*int i* = 0; *i* <> 3; *i*++)<br>{<br>*bloco de comandos;*<br>} |

Com a relação de mutantes mortos, mutantes vivos e mutantes equivalentes é possível mensurar a qualidade do conjunto de casos de teste *T*. Demillo [17] destaca a capacidade que a Análise de Mutantes tem em fornecer uma medida objetiva do nível de confiança da adequação dos casos de teste analisados, através da definição de um *score* de mutação (*mutation score*), que relaciona o número de mutantes mortos com o número de mutantes gerados. Seu cálculo é feito da seguinte forma:

$$ms(P,T) = \frac{DM(P,T)}{M(P) - EM(P)}$$

Sendo:

$DM(P,T)$ : quantidade de mutantes mortos pelos casos de teste em $T$

$M(P)$ : quantidade total de mutantes gerados.

$EM(P)$ : número de mutantes equivalentes a $P$.

O valor de $ms(P,T)$ varia entre 0 e 1, sendo que quanto maior este valor, mais adequado é o conjunto de casos de teste $T$ para o programa $P$ sendo testado.

Com o score de mutação calculado, cabe ao testador decidir se o teste deve continuar ou não. Se o valor do score é igual a 1 ou bem próximo disso, os testes podem ser finalizados considerando que os casos de teste de $T$ estão satisfatoriamente adequados para testar $P$.

Nas situações onde o valor do score de mutação de $T$ for baixo, ou seja, a qualidade de $T$ é insatisfatória dentro dos critérios esperados pelo testador, pode ser feita uma avaliação dos mutantes sobreviventes para analisar porque $T$ não foi capaz de identificar os defeito inseridos nos mutantes. A partir desta análise é possível melhorar a qualidade de $T$, incrementando novos casos de teste mais adequados para matar os mutantes sobreviventes. O processo se repete a partir da Etapa 2.

### 2.2.2 Os Problemas da Análise de Mutantes

Existem alguns problemas que podem inviabilizar a aplicação da Análise de Mutantes. Dentre os mais relevantes vale citar o alto custo computacional para a execução da enorme quantidade de mutantes gerados para o programa em teste. Reduzir a quantidade de mutantes sem ter perca significativa na capacidade de detecção de defeitos se tornou um problema muito pesquisado [27]. Algumas técnicas derivadas da Análise de Mutantes foram criadas, onde se procura selecionar um subconjunto do total de mutantes gerados, reduzindo o custo associado, mas com expectativa de não se reduzir a eficácia do critério. Dentre estas técnicas tem-se: a Mutação Aleatória; Mutação Restrita; Mutação Seletiva [32].

Outro problema relevante na Análise de Mutantes é a avaliação dos mutantes equivalentes. Detectar se um mutante é equivalente ou não é um problema indecidível, como foi provado nos trabalhos de Budd et al. [9]. Isso implica que esta avaliação muitas vezes deve ser feita por humanos. Porém, também existem muitos trabalhos no sentido de tentar automatizar este processo [22] [35]. Diversos outros trabalhos e pesquisas com o intuito de minimizar problemas da Análise de Mutantes foram relacionados por Jia [27].

## 2.3 Análise de Mutantes para Instruções SQL

Análise de Mutantes foi proposta inicialmente para ser usada em linguagens de programação como o *Fortran* [8], *Ada* [6] e *Linguagem C* [1] [15]. Posteriormente a técnica começou a ser aplicada em outros tipos de linguagens, interfaces, especificações e modelos. No final da década de 90, por exemplo, foram realizados alguns trabalhos para a aplicação da técnica na linguagem Java [29] [31]. No inicio da década passada, em 2000 e 2001, foram realizadas as primeiras pesquisas na aplicação da Análise de Mutantes em software Orientado a Objetos [26]. Para cada nova necessidade, novos e específicos operadores de mutação foram criados para adequação da técnica.

Em tese, tendo definidos os operadores de mutação, o processo da Análise de Mutantes continua o mesmo independente da linguagem e/ou interface. Sendo assim, a partir do momento que existam operadores específicos, a técnica poderia ser usada para testes de comandos SQL, e com isso avaliar a qualidade de casos de teste específicos para aplicações de banco de dados. Neste caso, o programa que sofre a mutação é uma instrução SQL.

A expectativa é que os operadores de mutação gerem mutantes SQL que consigam simular os defeitos mais comuns cometidos nesta linguagem. Estes defeitos podem

ser vistos, por exemplo, no trabalho de Brass et al. [7], que em 2006 classificou erros semânticos em instruções SQL baseados em diversos programas escritos por programadores com pouca experiência. Ele destaca os problemas típicos que ocorreram em todos os defeitos encontrados nas instruções, que justamente são aqueles considerados nos operadores de mutação para instruções SQL, como pode ser visto em vários trabalhos [50] [11] [27] [23] [43].

Um exemplo do uso de operadores de mutação pode ser observado nas instruções a seguir, onde a partir da única instrução SQL do Código 2.1 foram gerados 305 mutantes automaticamente utilizando uma ferramenta específica chamada *SQLMutantion* [48], que usa um conjunto específico de operadores de mutação definidos por Tuya et al. [50]. Os Códigos 2.2 e 2.3 são apenas dois exemplos de mutantes dentre todos os gerados. No Código 2.2, com o uso do operador de mutação **SEL**, foi aplicada uma modificação na cláusula *SELECT* acrescentando *DISTINCT* e no Código 2.3 foi alterado o primeiro campo da cláusula *SELECT* através do operador de mutação **IRC**.

**Código 2.1:** *Instrução SQL Original*

```
SELECT fname, lname, salary, dname, pname, plocation, hours
FROM department, employee, project, works_on
WHERE dno = dnumber AND essn = ssn
AND pno = pnumber AND salary > 1000
AND hours > 6 AND plocation NOT LIKE '%-GO'
AND ssn IN ( SELECT DISTINCT superssn FROM employee )
ORDER BY pname, fname
```

**Código 2.2:** *Primeiro Exemplo de Mutante Gerado.*

```
SELECT DISTINCT fname , lname , salary , dname , pname ,
    plocation , hours
FROM department , employee , project , works_on
WHERE dno = dnumber AND essn = ssn
AND pno = pnumber AND salary > 1000
AND hours > 6 AND plocation NOT LIKE '%-GO'
AND ssn IN ( SELECT DISTINCT superssn FROM employee )
ORDER BY pname , fname
```

**Código 2.3:** *Segundo Exemplo de Mutante Gerado.*

```
SELECT employee.lname AS fname , lname , salary , dname , pname
    , plocation , hours
FROM department , employee , project , works_on
WHERE dno = dnumber AND essn = ssn
```

```
AND pno = pnumber AND salary > 1000
AND hours > 6 AND plocation NOT LIKE '%-GO'
AND ssn IN ( SELECT DISTINCT superssn FROM employee )
ORDER BY pname , fname
```

### 2.3.1 Casos de Teste para Análise de Mutantes SQL

Além dos operadores de mutação, outro grande aspecto singular e que merece destaque na Análise de Mutantes SQL são os Casos de Teste. Cabeça et al. [10] definem que no contexto de aplicações de banco de dados, um caso de teste é formado por:

- Dados de entrada para o programa;
- Uma ou mais instâncias de bancos de dados;
- A saída esperada e a instância esperada do banco para estes dados de entrada.

A qualidade de uma instância de banco de dados utilizada como um Caso de Teste na Análise de Mutantes SQL é determinante para permitir uma boa cobertura nos testes. A instância, também conhecida como Base de Dados de Teste (BDT), é usada na execução da instrução SQL original e na execução dos mutantes. Por este motivo ela deve ser composta por tuplas, que sejam sensíveis o suficiente para distinguir a instrução SQL original das suas variações sintáticas representadas pelos defeitos inseridos nos mutantes.

Para melhor entendimento sobre a importância de um caso de teste adequado, é dada, como exemplo, uma BDT composta por uma única tabela ***employee*** ilustrada na Tabela 2.3.

**Tabela 2.3:** *Exemplo de uma BDT - Tabela employee*

| SSN | Fname | Salary | ... |
|---|---|---|---|
| 37 | Walt Kowalski | $7,250.00 | ... |
| 55 | Frankie Dunn | $4,930.00 | ... |
| 82 | Frank Morris | $5,500.00 | ... |
| 83 | Robert Kincaid | $3,700.00 | ... |
| 94 | William Munny | $4,125.00 | ... |

Executando na BDT, representada pela Tabela 2.3, uma instrução SQL Original *iSQL* representada no Código 2.4, seriam retornados os dados apresentado na Tabela 2.4

**Código 2.4:** *iSQL - Instrução SQL Original*

```
SELECT ssn, fname, salary
FROM employee
WHERE salary > 5000
ORDER BY salary
```

**Tabela 2.4:** *Retorno da consulta realizada com o Código 2.4*

| SSN | Fname | Salary | ... |
|---|---|---|---|
| 82 | Frank Morris | $5.500 | ... |
| 37 | Walt Kowalski | $7.250 | ... |

O Código 2.5 seria um possível mutante *M1* para a instrução *iSQL*. Neste exemplo o sinal **>** foi alterado para **<**. Executando este mutante na mesma BDT utilizada pela *iSQL* seriam obtidas as tuplas representadas na Tabela 2.5. Este retorno é diferente daquele obtido pela execução da *iSQL*. Consequentemente, conclui-se que a BDT conseguiu matar o mutante *M1*.

**Código 2.5:** *Mutante M1 da instrução iSQL*

```
SELECT ssn, fname, salary
FROM employee
WHERE salary < 5000
ORDER BY salary
```

**Tabela 2.5:** *Retorno da consulta realizada com o Código 2.5*

| SSN | Fname | Salary | ... |
|---|---|---|---|
| 83 | Robert Kincaid | $3.700 | ... |
| 94 | William Munny | $4.125 | ... |
| 55 | Frankie Dunn | $4.930 | ... |

O Código 2.6 seria um outro possível mutante *M2* para a instrução *iSQL*. Neste exemplo o sinal **>** foi alterado para **>=**. Executando este mutante na mesma BDT utilizada pela *iSQL* o retorno seria exatamente o mesmo representado na Tabela 2.4 pois não existe nenhuma tupla com o valor da coluna *salary* igual a $5.000. Consequentemente este mutante não seria morto pelo caso de teste, ou seja, a BDT não tem a capacidade de identificar o defeito.

**Código 2.6:** *Mutante M2 da instrução iSQL*

```
SELECT ssn, fname, salary
FROM employee
WHERE salary >= 5000
ORDER BY salary
```

Para identificar o defeito do *M2* seria necessário apenas modificar a BDT inserindo uma única tupla com o valor de *salary* igual a $5.000 , ficando a tabela *employee* como demostrado na Tabela 2.6 .

Construir uma boa Base de Dados de Testes não é trivial. Além de aspectos relacionados com o conteúdo da BDT (capacidade de identificação dos defeitos), estão

**Tabela 2.6:** *BDT Modifica - Tabela employee*

| SSN | Fname | Salary | ... |
|---|---|---|---|
| 37 | Walt Kowalski | $7.250 | ... |
| 55 | Frankie Dunn | $4.930 | ... |
| 82 | Frank Morris | $5.500 | ... |
| 83 | Robert Kincaid | $3.700 | ... |
| 94 | William Munny | $4.125 | ... |
| 100 | Nick Pulovski | $5.000 | ... |

envolvidas questões relacionadas com o seu tamanho. Quanto maior a quantidade de tuplas de uma BDT, maior o custo de execução dos testes [47].

Basicamente existem duas maneiras de providenciar BDTs. Elas podem ser geradas ou podem ser selecionadas (extraídas) de uma Base de Dados de Produção (BDP). No caso da geração, este processo pode ser *ah-hoc* ou ser orientado por alguma técnica. O mesmo vale para a seleção. Contudo, a seleção a partir da extração de dados pode ser feita utilizando 100% da BDP ou apenas um pequeno subconjunto da mesma, ou seja, fazendo uma redução da BDP para formar a BDT. Essa redução pode ser aleatória ou conduzida por alguma técnica que se proponha a manter uma boa qualidade da BDT.

Técnicas para redução procuram selecionar um subconjunto dos dados de teste, de forma que a sua cobertura se mantenha a mesma dos dados de teste originais. Essas abordagens têm o objetivo em comum de diminuir o custo do teste de regressão. Existem várias pesquisas que abordam o problema de geração e de seleção/redução de dados de teste para Análise de Mutantes SQL. No próximo capítulo serão apresentados alguns destes trabalhos, bem como os resultados alcançados por eles na realização de experimentos.

## 2.4 Considerações Finais

Neste capítulo foram apresentados importantes conceitos e definições que subsidiarão o entendimento geral dos outros capítulos deste trabalho.

Para uma melhor assimilação sobre os objetivos e vantagens do uso de um *benchmark*, foi apresentada a técnica de *benchmark* como uma maneira eficiente de realizar avaliações de desempenho de produtos da ciência da computação.

Por fim, Análise de Mutantes foi abordada no capítulo como critério de teste, apresentando seus fundamentos, seus objetivos e o seu processo de funcionamento. Tais conhecimentos auxiliaram na introdução dos conceitos específicos acerca da Análise de Mutantes SQL, principalmente sobre os Casos de Teste para esta abordagem.

CAPÍTULO 3

# Trabalhos Relacionados

*"Se eu vi mais longe, foi por estar de pé sobre ombros de gigantes."*
*Isaac Newton*

Neste capítulo são apresentados trabalhos publicados até o ano de 2013 que abordam pesquisas relevantes para o contexto desta dissertação.

O problema atribuído ao presente trabalho está relacionado aos seguintes domínios: (i) construção de bases de dados de teste, pela redução de bases de produção, no contexto do teste de aplicações que utilizam a SQL; (ii) emprego da análise de mutantes para medir a qualidade de bases de de dados de teste; e (iii) utilização de um *benchmark*, com cenários promissores para a aplicação de técnicas de busca, com o objetivo suportar a avaliação dessas técnicas no contexto da redução do custo do teste. Na literatura, não foram identificadas iniciativas de pesquisa que envolvessem todos os domínios simultaneamente. Diante disso, pesquisou-se por trabalhos que explorem tais domínios.

A técnica de *benchmark* funciona executando um conjunto fixo e padronizado de tarefas sobre um produto para avaliar o seu desempenho [20], sendo amplamente utilizada para avaliação de desempenho em sistemas computacionais [30]. Também não foi identificado o uso de *benchmark*s pertinente ao Domínio (iii) acima mencionado.

O capítulo está dividido em cinco seções. A primeira seção explora o Domínio (ii) e relaciona trabalhos sobre a Análise de Mutantes SQL envolvendo criação de operadores SQL e aplicações da técnica. A segunda e terceira seções expandem o Domínio (i), abordando trabalhos mais específicos sobre geração de dados de teste, e trabalhos de redução/seleção de dados de teste, ambos no contexto de Análise de Mutantes SQL. A quarta seção procura destacar algumas características e conclusões de cada trabalho que merecem maior atenção para o contexto da presente pesquisa. A quinta seção

apresenta considerações finais sobre os trabalhos expostos neste capítulo.

Para facilitar a organização e exposição dos trabalhos relacionados, foi criada uma subseção para cada artigo apresentado neste capítulo.

## 3.1 Análise de Mutantes SQL

### 3.1.1 *Fault-Based Testing of Database Application Programs with Conceptual Data Model*

O primeiro trabalho na literatura a projetar operadores de mutação para a linguagem SQL foi feito por Chan et al. [11] em 2005, quando foram propostos sete operadores baseados em modelos de entidade e relacionamento [27]. Para a criação dos mutantes, a abordagem explora informações semânticas do modelo de dados conceitual, tais como relações entre os atributos armazenados e derivados, de acordo com a família de operadores de substituição definida.

Os operadores foram criados somente para os comandos SQL do tipo *SELECT* e podem ser observados como foram originalmente propostos na Tabela 3.1.

**Tabela 3.1:** *Operadores de Mutação proposto por Chan et al. [11]*

| **Operadores Semânticos de Mutação** | **Sigla** | **Descrição** |
|---|---|---|
| Reposições de Restrições de Particionamento | PTCR | Altera os requisitos de participação dos tipos de entidade na tabela. |
| Reposições de Restrições de Cardinalidade | CDCR | Altera os requisitos de cardinalidade dos tipos de entidade na tabela. |
| Identificação / Substituição de entidades do tipo fraca | IWKR | Substituir uma expressão do tipo de identificação por uma expressão do tipo de entidade fraca, ou vice versa. |
| Substituição de Atributos | ATTR | Substituir uma expressão de atributo(s) por uma expressão de outro(s) atributo(s) de um tipo compatível. |
| Substituição de Generalização / Especialização de Integralidade | GSCR | Substituir uma expressão parcial de uma superclasse por uma expressão em uma subclasse na forma de negação da superclasse. |
| Substituição de Generalização / Especialização de disjunção | GSDR | Substituir uma expressão de tipo de entidade irmão por uma expressão de outro(s) tipo de entidade irmão na mesma superclasse. |
| Substituição de integridade do tipo união | UTCR | Substituir um tipo de entidade por uma subclasse e / ou superclasses da subclasse, de tal forma que estas superclasses têm a mesma restrição do tipo união |

Nas conclusões deste trabalho foi apresentado que para avaliar a efetividade da abordagem (operadores) propostos, Chan et al. realizaram um pequeno estudo de caso onde foram gerados 35 mutantes não equivalentes a partir de uma única instrução SQL.

A abordagem deles conseguiu detectar 89.5% dos defeitos, contra 78.9% de uma outra abordagem convencional usada como comparação.

### 3.1.2 *SQLMutation: A tool to generate mutants of sql database queries*

Em 2006, Tuya et al. [48] desenvolveram uma ferramenta chamada *SQLMutation* para automatizar a geração de mutantes a partir de instruções SQL. Uma característica interessante da *SQLMutation* é a identificação de mutantes equivalentes durante o processo de geração. A ferramenta implementa um conjunto de operadores que foi formalizado posteriormente em outro trabalho [50]. Muitos trabalhos usam essa ferramenta em seus experimentos [34] [19] [50] [38] [44] [47].

Os autores realizaram um experimento para avaliar a ferramenta. O experimento consistia em gerar mutantes SQL a partir de quatro instruções que foram criadas por um grupo de 7 estudantes durante um exercício. Os próprio estudantes também criaram a base de dados de teste (BDT). Ao final, usando os mutantes gerados e a BDT, o escore de mutação médio foi de 78,5%. Sendo que a categoria de operadores que apresentou menor escore de mutação foi a SC, com 59,6%.

### 3.1.3 *Mutating Database Queries*

No ano de 2007, foi proposto um outro conjunto de operadores por Tuya et al. [50] que pode ser observado na Tabela 3.2. Este conjunto foi organizado em quatro categorias, sendo elas: **SC** - Mutação de cláusulas SQL; **OR** - Mutação de operadores em condições e expressões; **NL** - Mutação de manipulação de valores nulos e **IR** - Mutação de Identificadores. Os operadores foram criados baseados nas principais dificuldades e erros de elaboração de instruções *SELECT*.

Para demostrar a aplicação da técnica de Análise de Mutantes, foram realizados experimentos utilizando os operadores propostos no artigo. Os pesquisadores utilizaram o ambiente de teste *NIST SQL Conformance Test*, desenvolvido pela NIST (*National Institute of Standards and Technology*). Trata-se de um *test suite* para avaliar produtos comerciais de SQL a fim de validar a compatibilidade com os padrões ISO, ANSI, e FIPS SQL. Este ambiente é composto por módulos, que são pequenos programas com instruções SQL. Para os testes foram selecionados somente os módulos que realizam consultas no banco de dados. Os experimentos (cálculos de escore de mutação) foram realizados em cinco passos:

**Tabela 3.2:** *Operadores de Mutação propostos por Tuya et al. [50]*

| Categoria | Tipo | Descrição |
|---|---|---|
| SC | SEL | Alterna as palavras chaves SELECT e SELECT DISTINCT |
| | JOI | Cada palavra chave (INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN, CROSS JOIN) é substituida por outra |
| | SUB | Mutação de predicatos de sub-consultas |
| | GRU | A expressão GROUP BY é removida |
| | AGR | Cada função de agregação {MIN, MAX, AVG, AVG(DISTINCT), SUM, SUM(DISTINCT), COUNT, COUNT(DISTINCT)} é substituida por outra |
| | UNI | Alterna as palavras chaves UNION e UNION ALL ou remove as consultas de união |
| | ORD | Alterna as palavras chaves ASC e DESC ou as remove da expressão ORDER BY |
| OR | ROR | Cada operador relacional {=, <>, <, <=, >,>=} é substituido por outro |
| | LCR | Cada operador lógico {AND, OR} é substituido por outro |
| | UOI | Cada expressão aritmética (e) ou referência a um número (e) é substituido por - e, e + 1 e e – 1 |
| | ABS | Cada expressão aritmética (e) ou referência a um número (e) é substituido por ABS(e) e - ABS(e) |
| | AOR | Cada operador aritmético { =, -, *, /, %} é substituido por outro |
| | BTW | Cada condição a BETWEEN x AND y é substituido por a>x AND a<=y, e por a>=x AND a<y |
| | LIKE | Mutações da expressão a LIKE s (removendo, substituindo, adicionando opções {%,_}, ...) |
| NL | NLF | Alterar os predicatos NULL e IS NOT NULL |
| | NLS | Cada referência a coluna na lista do SELECT é substituida por uma função que altera o valor da coluna para outro quando é encontrado valor nulo na coluna. |
| | NLI | Força um valor verdadeiro de uma condição onde existe um valor nulo |
| | NLO | Cada atributo x na condição C é substituido por NOT C OR x IS NULL, x IS NULL, x IS NOT NULL |
| IR | IRC | Cada referência a coluna é substituida por outra referência de coluna, constante ou parâmetro (tipos compatíveis) da cláusula SELECT. |
| | IRT | Cada constante é substituida por outra referência de coluna, constante ou parâmetro (tipos compatíveis) da cláusula SELECT |
| | IRP | Cada parâmetro é substituida por outra referência de coluna, constante ou parâmetro (tipos compatíveis) da cláusula SELECT |
| | IRH | Cada referência a um atributo de coluna é substituido por outro (tipos compatíveis) definidos na tabela, mas não aplicável para os operadores IRC, IRT ou IRP |

1. No primeiro passo, para calcular os escores de mutação, foram gerados os mutantes a partir das instruções SQL originais dos módulos selecionados. Foi utilizada a base de dados original do ambiente de teste da NIST. Neste passo foi alcançado um escore de mutação de 69,6%.
2. No segundo passo foi mantida a mesma base de dados utilizada no passo anterior. Porém foram alterados os parâmetros das instruções SQL. Nesta etapa o escore de mutação obteve uma melhoria, ficando com o valor de 79,7%.
3. No Passo 3 foram mantidas as instruções SQL alteradas no passo anterior. Porém neste passo foi feita uma cópia da base de dados original e foram aplicadas nela algumas alterações. O escore de mutação aumentou para 83,3%.
4. Por fim, no quarto e último passo foram mantidas as mesmas instruções SQL e feita uma segunda alteração, com a inclusão de valores nulos, na base de dados utilizada no passo anterior. O escore de mutação alcançou o valor de 85,6%.

A cada passo os resultados dos experimentos foram agrupados e apresentados por categoria e subcategoria dos operadores de mutação. Ao final, a categoria de operadores SC obteve o menor escore de mutação, demostrando que os mutantes gerados a partir dos operadores desta categoria foram mais difíceis de serem mortos.

Tuya et al. relatam também a necessidade de reduzir o custo de teste da Análise de Mutantes SQL. São sugeridas duas maneiras de reduzir este custo:

1. Reduzindo o número de mutantes: Foi proposto que a redução ocorra excluindo operadores de mutação que geram muitos mutantes com pouca representatividade para identificação de defeitos;
2. Reduzindo o número de casos de teste: Essa redução deve acontecer a partir do momento que os casos de teste são priorizados, ordenando-os de forma que os que matam mais mutantes são executados primeiro.

### 3.1.4 Análise de Mutantes em Aplicações SQL de Banco de Dados

Em 2008, Cabeça et al. [10] definem um outro conjunto de operadores de mutação para a maioria dos comandos de consulta e de atualização de dados pertencentes ao padrão SQL 3 cuja a sintaxe fosse passível de mutação, não restringindo somente para instruções com o comando *SELECT*, mesmo sendo este comando o mais utilizado em instruções SQL nas aplicações comerciais [39].

Os operadores foram divididos em cinco categorias de acordo com a funcionalidade do comanto SQL e podem ser observados na Tabela 3.3.

**Tabela 3.3:** *Operadores de Mutação propostos por Cabeça et al. [10]*

| **Categoria** | **Operador** | **Descrição** |
|---|---|---|
| Operadores de SQL | tOpMt | Troca de Operador Matemático |
| | tOpCp | Troca de Operador de Comparação |
| | tOpCj | Troca de Operador Conjuntivo |
| | tOpLg | Troca de Operador de Lógico |
| | iNot | Inserção de Operador de Negação |
| | rNot | Retirada de Operador de Negação |
| Miscelânea | tPoAt | Troca de Posição de Atributo |
| | rAtr | Retirada de Atributo |
| | iAtr | Inserção de Atributo |
| | tPoVr | Troca de Posição de Valor |
| | tVr | Troca de Valor |
| | tTpVar | Troca de Tipo de Variável |
| | tNmTb | Troca de Nome de Tabela |
| | tNmRole | Troca de Nome de ROLE |
| | iRole | Inserção de ROLE |
| | rRole | Retirada de ROLE |
| | tNmCursor | Troca de Nome de Cursor |
| | tFuAg | Troca de Função de Agregação |
| | tInSec | Troca de Intersecção |
| | tJoin | Troca de Join |
| Fluxo de Controle | tBlCmEstRep | Troca de Bloco de Comandos nas estruturas de condição e repetição |
| | rCmBlRep | Retirada de Comando do Bloco de Repetição/Condição |
| | iCmBlRep | Inserção de Comando do Bloco de Repetição/Condição |
| | tPosLeave | Troca de Posição do *Leave* no Bloco de Comandos |
| | rLeave | Retirada de *LEAVE* |
| | iLeave | Inserção de *LEAVE* |
| Controle de Transações | iCM | Inserção de *COMMIT* |
| | rCM | Retirada de *COMMIT* |
| | iRb | Inserção de *ROLLBACK* |
| | rRb | Retirada de ROLLBACK |
| | tCmRb | Troca de *COMMIT* por *ROLLBACK* |
| | tRbCm | Troca de *ROLLBACK* por *COMMIT* |
| | tNmSP | Troca de Nome do *SAVEPOINT* |
| | tPerm | Troca de Permissão |
| | tPriv | Troca de Privilégio |
| | tGrRe | Troca de *GRANT* por *REVOKE* |
| | tReGr | Troca de *REVOKE* por *GRANT* |
| | tNmUsr | Troca de Nome de Usuário |
| Funções, Procedimentos e *Triggers* | tNm | Troca de Nome da Função, Procedimento, *VIEW* ou *TRIGGER* |
| | tPoReFu | Troca Posição de Retorno da Função |
| | rReFu | Retirada de Retorno da Função |
| | tPaPro | Troca de Parâmetros da *PROCEDURE* |
| | tEv | Troca de Evento na *TRIGGER* |

Para aplicar a técnica e os operadores de mutação, foram conduzidos 3 experimentos. O objetivo principal dos experimentos foi investigar a aplicabilidade e a habilidade de detecção de defeitos dos operadores propostos. Nos experimentos foram utilizados três sistemas do mundo real: (i) Aplicação de gestão de equipamentos de informática; (ii) Software de controle de empréstimo de materiais; e (iii) Aplicação de suporte para padrões de desenvolvimento de software.

Cada experimento foi executado mais de uma vez, sendo que em cada uma delas com uma BDT diferente. Em alguns momentos foram empregadas as BDTs já utilizadas pela equipe de desenvolvimento de cada sistema. Outro cenário foi o uso da base de dados de produção original de cada sistema como BDT. E para finalizar foram usadas versões reduzidas das bases de dados de produção.

Os experimentos apresentaram resultados animadores em relação às bases de dados reduzidas. Em dois experimentos foram alcançados escores de mutação mostrando que as bases de dados reduzidas pelas empresas possuem habilidade igual ou superior de detecção de defeitos do que a base de produção. O escore de mutação para ambas as bases de dados é similar em muitos casos de teste, podendo indicar que a base reduzida possui eficiência de detecção de defeitos, no mínimo, similar à da base de produção. Em outro experimento a base de dados de teste conseguiu matar mutantes que a base de produção não matou.

### 3.1.5 *Music: Mutation-based SQL Injection Vulnerability Checking*

No mesmo ano de 2008, Shahriar e Zulkernine [43] propuseram um conjunto de nove operadores de mutação específicos para testar um tipo de vulnerabilidade de segurança em aplicações de banco de dados conhecida como Injeção SQL *(SQL Injection)*. Essas vulnerabilidades são muito exploradas em aplicações disponibilizadas na plataforma web *(web-based applications)*. A Tabela 3.4 apresenta os operadores propostos.

A intenção é que esses operadores gerem mutantes que só serão mortos se o(s) caso(s) de teste contenham dados que simulem ataques de injeção SQL. Para automatizar a geração dos mutantes eles também criaram uma ferramenta chamada MUSIC *(MUtation-based SQL Injection vulnerabilities Checking)*.

Shahriar e Zulkernine conduziram um conjunto de experimentos usando 5 aplicações web disponíveis em repositórios *open source*. Foi utilizado também um *benchmark* específico que contém casos de teste simulando ataques de injeção SQL.

Foram gerados para todas as aplicações 654 mutantes usando a ferramenta *MUSIC*. A primeira bateria de experimentos usou uma BDT $T_1$ com um conjunto de

**Tabela 3.4:** *Operadores de Mutação propostos por Shahriar e Zulkernine [43]*

| Categoria | Operador | Descrição |
|---|---|---|
| WC | RMWH | Remove a palavra chave WHERE e suas condições. |
| | NEGC | Nega cada uma das expressões unitárias dentro da condição WHERE. |
| | FADP | Adiciona a expressão "FALSE AND" depois da palavra chave WHERE. |
| | UNPR | Desorganiza os parênteses das condições da expressão WHERE. |
| AMC | MQFT | Marca vários indicadores de execução da consulta como TRUE. |
| | OVCR | Elimina opções de COMMIT e ROLLBACK. |
| | SMRZ | Define para infinito o número máximo de registros retornador por um resultado. |
| | SQDZ | Define para infinito o tempo de execução de uma consulta. |
| | OVEP | Elimina indicadores(*flags*) de saida(*ecape*) do processamento. |

tuplas geradas sem um critério específico. O escore de mutação alcançado foi de 0%, mostrando que $T_1$ não tinha capacidade para identificar vulnerabilidades de *SQL Injection*. A segunda bateria de experimentos foi feita com uma outra BDT $T_2$ contendo casos de teste (tuplas) retirados do *benchmark* usado nos experimentos. Esses casos de teste basicamente simulam ataques de injeção SQL. Desta vez, com a BDT $T_2$ foi alcançado um escore de mutação médio de 71%.

Verificando os resultados, foi identificado que a BDT $T_2$ não possuía casos de teste para matar os mutantes da categoria UNPR. Avaliando os mutantes sobreviventes, os pesquisadores conseguiram criar tuplas apropriadas e incrementaram $T_2$ aumentando sua qualidade. Realizando os experimentos novamente o escore de mutação alcançou o valor de 100%.

### 3.1.6 *An Experimental Case Study to Applying Mutation Analysis for SQL Queries*

Derezinska em 2009 [19] desenvolve um trabalho onde o principal foco foi a avaliação (utilidade e performance) dos operadores de mutação para instruções SQL, através da realização de experimentos utilizando os operadores propostos por Tuya et al. [50]. O artigo apresenta também uma pequena, porém interessante, revisão sobre os conceitos básicos da Análise de Mutantes, Análise de Mutantes SQL e alguns trabalhos relacionados.

Foram realizados experimentos a partir de uma base de dados reduzida de um sistema de seguros utilizado por uma empresa real. A redução foi realizada sem nenhum critério pré-estabelecido e os experimentos utilizaram instruções SQL de 5 módulos do

sistema. Os mutantes foram gerados através a ferramenta *SQLMutation* [49] e, ao total, para todos os cinco módulos, foram gerados 1.159 mutantes.

Os experimentos foram executados com o suporte de um programa desenvolvido em linguagem *Perl* e o SGBD utilizado foi o Oracle. Neste programa o escore de mutação foi calculado comparando o resultado da instrução original com o resultado dos mutantes. Esta comparação foi feita gerando uma *hash* de cada resultado através um algoritmo SHA[1]. Com isso o programa comparava somente as *hashs* dos resultados. Outro critério usado para matar os mutantes foi o tempo padrão de resposta do SGBD para a execução das instruções. Quando o tempo fosse acima do definido o mutante era considerado morto.

No geral, os escores de mutação alcançados tiveram um bom desempenho, ficando com um valor médio de 92%. Distribuindo os resultados pelas categorias dos operadores de mutação foram alcançados os valores apresentados na Tabela 3.5.

**Tabela 3.5:** *Resultados dos experimentos realizados por Derezinska*

| Categoria | Número de Mutantes | Mutantes Mortos | Escore de mutação (%) |
|---|---|---|---|
| SC | 37 | 27 | 73 |
| OR | 365 | 320 | 88 |
| NL | 142 | 113 | 80 |
| IR | 615 | 604 | 98 |

A distribuição dos resultados entre as categorias de operadores foi comparada com a distribuição dos resultados alcançados por Tuya et al. no artigo *Mutating Database Queries* [50]. Derezinska destaca que podem ser observadas muitas similaridades entre os resultados dos dois trabalhos. Em ambos os casos a média de escore de mutação mais alta foi alcançada com a categoria de operadores de mutação IR, e a pior média com a categoria OR. Mutantes gerados pelos operadores (ROR, NLO, IRP) foram facilmente mortos em ambos os casos.

### 3.1.7 *JDAMA: Java Database Application Mutation Analyser*

No ano de 2011, Chixiang Zhou e Phyllis Frankl [51] apresentaram uma técnica para aplicar Análise de Mutantes SQL em programas desenvolvidos em linguagem Java com JDBC (*Java Database Connectivity*). A técnica estende a ideia básica dos operadores de mutação em instruções SQL proposta por Tuya et al. [50]. Essa extensão ocorre integrando a abordagem de Tuya et al. com a análise e modificação das aplicações Java/JDBC em nível de *bytecode* .

[1] SHA (*Secure Hash Algorithm*) é uma função de criptografia muito utilizada em protocolos de segurança

Para aplicar a técnica foi desenvolvida uma ferramenta, *Java Database Application Mutation Analyser* (JDAMA). Esta ferramenta pode ser usada tanto para realizar testes de mutação em aplicações Java/JDBC como para dar suporte na realização de experimentos de avaliação de técnicas de teste para aplicações de banco de dados.

Uma característica importante considerada neste trabalho é o fato de que na maioria das aplicações Java/JDBC as instruções SQL são dinamicamente construídas e realizadas em tempo de execução. Desta forma, a estrutura da instrução executada e os seus parâmetros ficam dependentes do caminho que o programa percorreu para montar a consulta e dos dados fornecidos para a sua execução. Segundo os autores, esse cenário representa uma dificuldade na aplicação da Análise de Mutantes para estas instruções.

Uma forma de amenizar esta dificuldade foi criando o conceito de **instruções abstratas**, que podem ser interpretadas como instruções SQL que contêm fragmentos que serão fornecidos durantes a execução (*runtime*) do programa. A técnica passa então a considerar estas instruções abstratas para realizar o processo de mutação.

Os experimentos realizados no artigo tiveram dois objetivos específicos:

1. Usar a ferramenta JDAMA para avaliar técnicas de geração de casos de teste;
2. Usar a ferramenta JDAMA para avaliar critérios de adequação de teste.

Em ambos os casos o critério de avaliação foi feito usando a Análise de Mutantes, tendo os escores de mutação calculados pela JDAMA.

Para o Objetivo 1, foi feita a comparação de casos de teste gerados a partir de um método aleatório com casos de teste gerados com uma ferramenta chamada AGENDA[2]. Foram usados programas de dois sistemas para os experimentos: (i) Uma aplicação Java/JDBC do tutorial oficial do Java chamada *Coffee*; e (ii) Uma implementação Java que usa o modelo de dados proposto pelo o *benchmark* TPC-C.

Somente em um dos casos utilizando um programa do sistema *Coffee*, a geração com o método aleatório obteve um desempenho maior. Para todos os outros experimentos o escore de mutação ficou mais alto quando os casos de teste foram gerados usando a ferramenta AGENDA.

[2]AGENDA é uma ferramenta que foi desenvolvida por um grupo de pesquisadores para testes de aplicações em banco de dados [12]

## 3.2 Geração de Dados para Análise de Mutantes SQL

### 3.2.1 *Populating Test Databases for Testing SQL Queries*

Em 2010 Suárez et al. [44] apresentam uma abordagem de **geração** automática de dados para testes de instruções SQL. Neste trabalho eles ressaltam sobre a grande dificuldade para geração manual de dados para teste, principalmente nos casos onde as estruturas de entrada são complexas, como por exemplo aplicações de banco de dados.

Para realizar a geração dos dados, a técnica se orienta pelas instruções SQL sendo testadas usando um critério de cobertura específico para linguagem SQL chamado *SQLFpc* [46]. Outro critério utilizado como regra para a geração dos dados de teste é o esquema do banco de dados, considerando as seguintes características e restrições descritas nele: Chaves primárias; Chaves estrangeiras; Valores Nulos; Valores de domínio.

Com o objetivo de demostrar a técnica de geração de dados, Suárez et al. realizaram um experimento (estudo de caso) para geração de uma BDT utilizando um sistema real de *Helpdesk*. Para este estudo de caso, foram extraídas do sistema três instruções SQL com diferentes níveis de complexidade, com isso foi considerado somente a parte do esquema do banco que é usada nas consultas selecionadas. A partir destas instruções e do esquema, foram realizadas as gerações dos dados.

O desempenho da BDT gerada (com apenas 10 tuplas) foi comparada com o desempenho de uma parte da BPD do sistema de *helpdesk* (com 1805 tuplas). Os critérios usados para realizar essa comparação foi a Cobertura *SQLFpc* e a Análise de Mutantes.

Os resultados se mostraram mais favoráveis quando avaliados com a *SQLFpc*. Para as três instruções SQL utilizadas no experimento, foram geradas 20 regras de cobertura do critério *SQLFpc*. Com a BDP houve um percentual médio de cobertura de 85%. Usando a BDT gerada o percentual médio de cobertura foi de 100%.

Usando a Análise de Mutantes como critério de comparação de desempenho, foram gerados 323 mutantes através da ferramenta *SQLMutation* [48]. Com a BDP o escore de mutação médio foi de 87%. Com a BDT gerada o score médio foi de 82,35%. Apesar do escore menor, o resultado ainda sim é muito interessante considerando a imensa diferença de tamanho entre a BDP e a BDT.

### 3.2.2 *Generating Test Data for Killing SQL Mutants: A constraint-based approach*

Em 2011, Shah et al. [42], dando continuidade a um trabalho anterior [23], abordaram o problema de **geração** de bases de dados para detecção de defeitos através da Análise de Mutantes. Segundo os autores, o objetivo da geração é minimizar o esforço humano para testes gerando pequenos e intuitivos casos de teste.

Os defeitos focados neste trabalho estão relacionados com a classe de comandos *Join/Out Join*. Também são abordados defeitos envolvendo a cláusula *WHERE*. Para dar suporte na abordagem, os autores definiram um conjunto de regras/operadores para orientar a criação dos mutantes. De acordo com os elementos de cada grupo de regras, os mutantes são gerados realizando subtituições destes elementos por outros elementos do mesmo grupo. Os grupos de regras são:

- **Mutantes de tipos de junções**: São considerados os tipos de junção: *Inner Join; Left outer-join; right outer-join* e *full outer-join.*
- **Mutantes de operadores de comparação**: São os operadores ($=, <, >, <=, >=$ , $<>$).
- **Mutantes de operadores de comparação**: São considerados os operadores M*AX, MIN, SUM, AVG, COUNT, SUM(DISTINCT), AVG (DISTINCT)* e *COUNT(DISTINCT)*.

A técnica de geração proposta funciona basicamente alterando as instruções SQL originais e gerando os mutantes de acordo com as regras estabelecidas. A partir destes mutantes são executados algoritmos que procuram gerar dados que atendam os critérios de seleção dos mutantes. Neste processo podem ser geradas várias BDTs.

Shah et al. conduziram experimentos para demostrar a eficiência da técnica no aspecto de poder de detecção de defeito e tempo de execução dos algoritmos envolvidos. Foram usadas instruções SQL com até 7 relacionamentos de tabelas. Infelizmente os resultados apresentados focaram somente no tempo em segundos para as gerações da BDTs. Porém, nas conclusões os autores afirmam que os experimentos geraram pequenas e eficientes BDTs.

### 3.2.3 *Automatic Test Generation for Mutation Testing on Database Applications*

Em 2013, Pan et al. [38] propuseram uma abordagem chamada *MutaGen* que conduz a geração de testes para Análise de Mutantes SQL. Essa abordagem é baseada em um *framework* chamado *SynDB* [37] para transformar códigos de uma aplicação de banco de dados em uma intrução SQL "pura". Em seguida, são gerados mutantes SQL a partir desta instrução com a ferramenta *SQLMutation* [48]. Baseados nesses mutantes são criados alguns critérios que orientam na geração dos dados de teste que tenham capacidade de matar estes mutantes.

Foram realizados experimentos para a avaliação da proposta utilizando duas aplicações de banco de dados *open source*: (i) *RiskIt* e (ii) *UnixUsage*. Para a primeira aplicação foram gerados 270 mutantes SQL, e para a segunda 75.

Inicialmente fizeram experimentos calculando o escore de mutação com uma BDT original. Em seguida, realizaram a geração de dados incrementando esta BDT. Em resumo, os resultados mostraram que quando os dados gerados pela abordagem foram inseridas na BDT, o escore de mutação da aplicação (i) aumentou 21,3% e da aplicação (ii) o aumento foi de 33,5%.

## 3.3 Redução de Dados para Análise de Mutantes SQL

### 3.3.1 *Query-aware shrinking test databases*

No ano de 2009, Tuya et al. [47] desenvolvem um novo trabalho com foco na **redução** de bases de dados para favorecer tarefas de teste. Apresentam uma técnica para realização da redução de forma automática a partir de uma base de dados inicial que pode ser uma cópia de uma BDP.

A abordagem é baseada nos critérios de cobertura MC/DC[3], onde são selecionadas tuplas que atendam certas condições de acordo com a(s) instrução(ões) SQL sendo avaliadas. O objetivo é que a redução aconteça preservando o poder de detecção de defeitos em relação a base de dados inicial.

Para dar suporte na abordagem, foi criada uma ferramenta chamada *QA Shirink Tool*. Ela automatiza o processo de redução proposto pelos autores e tem como principal

[3]O critério MC/DC foi proposto em 2006 por Tuya et al. no artigo *A practical guide to SQL white-box testing* [46]

objetivo selecionar o número mínimo de tuplas de uma BDP que atendam aos critérios de cobertura.

A técnica teve sua eficiência avaliada através da realização de experimentos (estudo de caso) com um sistema real de *help-desk*. Na época o sistema possuía uma BDP em SQL Server com 137.490 tuplas divididas em 31 tabelas. Os experimentos foram realizados com essa base de dados e com 198 instruções SQL capturadas do sistema.

Com os experimentos pretendeu-se responder principalmente as seguintes questões:

1. Qual o grau de redução pode ser atigindo?
2. A habilidade de detecção de defeitos da base de dados original é preservada na base de dados reduzida?

Para avaliar a qualidade da base de dados reduzida os autores utilizaram a ideia básica da técnica de Análise de Mutantes. Porém, ao invés de usar mutantes tradicionais foram usados como "mutantes" versões das instruções SQL originais criadas a partir das regras do critério MC/DC.

Outra característica relevante sobre os experimentos foi como acontece fisicamente a seleção das tuplas da BDP para formar a base reduzida. Durante o processo de redução a ferramenta mantem em memória informações das tuplas que vão compor a nova base de dados reduzida, porém, somente as chaves primárias de cada tabela são armazenadas em memória. No momento de gerar fisicamente a base reduzida, a ferramenta seleciona da BDP as tuplas de acordo com as chaves primárias selecionadas.

Os resultados da abordagem proposta no artigo se mostraram extremamente animadores. Com uma redução de 99,84% do banco de produção foi mantido um escore de mutação apenas 0,5% menor do que com a versão completa do banco.

### 3.3.2 *Applying genetic algorithms to data selection for sql mutation analysis*

No ano de 2013 Monção et al. [34] apresentam uma abordagem para testes de instruções SQL usando Análise de Mutantes juntamente com o emprego de um Algoritmo Genético (AG) para realizar a **seleção** de dados na formação de BDTs a partir da **redução** de uma BDP. O objetivo é conseguir, de forma heurística, selecionar um conjunto de dados de uma BDP que consiga ter um bom poder de detecção de defeitos de instruções SQL de uma aplicação.

Os AGs podem ser definidos como uma técnica de busca baseada numa metáfora do processo biológico de evolução. Baseiam-se nos mecanismos de seleção natural e genética para criar/encontrar boas soluções para um problema. Cada solução no AG é considerada como um indivíduo dentro de uma população. Com isso a técnica combina a sobrevivência entre os melhores com uma forma estruturada de troca de informação genética entre indivíduos, para formar uma estrutura heurística de busca.

Para viabilizar a abordagem proposta, foi definido e apresentado no trabalho uma especialização de AG que permitiu tratar um caso de teste (BDT) como uma solução. Além disso foi criado um modelo cromossomial, que é capaz de representar instâncias de banco de dados como indivíduos dentro do contexto de AG. O principal objetivo do modelo é possibilitar a representação de um subconjunto de dados (tabelas e tuplas) que está contido em uma BDP. A partir deste subconjunto é possível instanciar uma BDT para ser usada na Análise de Mutantes.

Para verificar a viabilidade da proposta, os autores realizaram experimentos com o objetivo principal de comparar resultados obtidos usando o AG com os obtidos a partir de métodos aleatórios. Com isso, pretendiam demonstrar os ganhos obtidos com a técnica. A expectativa era que com o AG fossem geradas pequenas BDTs que alcancem resultados (escores) iguais ou melhores do que aqueles encontrados usando toda a BDP. Os mutantes foram gerados usando a ferramenta *SQLMutantion* [48].

Ao final da primeira bateria de experimentos, os autores puderam comparar para cada instrução SQL, os seguintes valores (resultados) de escore de mutação:

- Score usando toda a BDP;
- Média e Maior score usando BDT de 100 tuplas gerada com o AG;
- Média e Maior score usando BDT de 100 tuplas gerada aleatoriamente;
- Média e Maior score usando BDT de 10.00 tuplas gerada aleatoriamente;

Para 11 instruções SQL avaliadas, três mostraram escores de mutação similares entre a seleção aleatória (100 tuplas) e a seleção do AG (100 tuplas). Outras três tiveram uma pequena melhora com o AG. Quatro instruções demonstraram um ganho considerável e uma delas conseguiu alcançar com o AG (100 tuplas) um score igual ao score da BDP (100.000 tuplas).

Na tentativa de encontrar melhores resultados, Monção et al. aplicaram mudanças consideráveis nos parâmetros do AG e realizaram novos experimentos para 3 instruções das 11 utilizadas anteriormente. Os parâmetros utilizados estão apresentados na Tabela 3.6.

**Tabela 3.6:** *Novos parâmetros do Algoritmo Genético usado nos experimentos*

| **Parâmetro** | **Valor** |
|---|---|
| Gerações | 30 |
| Indivíduos | 100 |
| Genes | 100 |
| *Crossover* | Ponto de corte aleatório |
| Taxa de mutação | 3% por gene |
| Elitismo | Os dois melhores indivíduos |

O escore de mutação de uma das instruções não sofreu mudança em relação ao escore alcançado pelo primeiro experimento (primeira versão do AG). Já para as outras duas instruções, foram geradas BDTs com escores de mutação melhores que os alcançados anteriormente chegando mais próximos do escore de mutação da BDP.

## 3.4 Questões Importantes

De modo geral, os trabalhos que foram relacioandos neste capítulo possuem algumas características, experiências e conclusões que merecem ser destacadas considerando o contexto e objetivos da pesquisa.

No trabalho de Tuya et al. [50] é colocado em destaque a vantagem e a viabilidade de se usar uma base de dados reduzida para realizar testes em instruções SQL. Os autores afirmam que um simples e pequeno caso de teste muitas vezes é suficiente para simular diferentes situações de dados e com isso conseguir detectar muitos defeitos representados pelos mutantes. Relatam também que existe uma dificuldade e um enorme esforço para a geração manual de casos de testes (dados de teste). Em contrapartida, eles afirmam que usar dados de teste de forma aleatória aumenta o risco de omitir defeitos.

Um possível problema em relação aos experimentos realizados foi levantado por Tuya et al [50]. Segundo eles a ausência de dados reais torna o experimento incerto em relação à representatividade de cada consulta, em termos das combinações de características que podem ser encontradas no mundo real, impossibilitando a comparação da aplicabilidade dos operadores com sistemas reais. Esta afirmação destaca a necessidade de existir um ambiente de teste que seja composto por dados de sistemas reais.

Derezinska [19] destaca sobre a desvantagem do uso de uma base de dados reduzida não apropriada para os experimentos. Afirma também que se fosse usada a base completa muitos mutantes sobreviventes, provavelmente, teriam morrido, porém o custo (tempo de execução) seria muito alto. Reforça a necessidade de realizar reduções que mantenham o mesmo poder de detecção de defeito da base real. Outro ponto levantado

como um possível problema é a falta de conclusividade nos resultados principalmente pela baixa quantidade de experimentos realizados.

Um problema previsto por Tuya et al. [47] foi a possível violação de integridade referencial durante o processo de redução. Tuplas selecionadas para formar a BDT podiam fazer referência a tuplas que não foram selecionadas, podendo causar inconsistência no banco de dados sendo criado. Esta situação foi evitada simplesmente incluindo, quando necessário, as tuplas referenciadas pelas tuplas selecionadas. Tuya et al. novamente destacam neste trabalho a importância da base de dados reduzida para uso em testes de aplicações de banco de dados.

Durantes os experimentos, Chixiang Zhou e Phyllis Frankl [51] se depararam com uma situação onde alguns mutantes somente poderiam ser mortos se fossem inseridos dados que infringissem restrições de unicidade do esquema do banco de dados (chaves primárias). Como isso não é possível, estes mutantes foram considerados nos experimentos como mutantes equivalentes.

Para realizar os experimentos e verificar o desempenho da técnica, Monção et al. [34] tiveram que dedicar um enorme tempo e esforço na criação de todo um ambiente de teste composto pela base de dados de grande volume (BDP), as instruções SQL com seus mutantes e as inúmeras reduções aleatórias para servirem como parâmetros de comparação com os resultados alcançados com a técnica de busca empregada por eles.

Esforço semelhante foi necessário também para todas as pesquisas que trabalharam com seleção/redução de dados no contexto de Análise de Mutantes SQL. Além disso, o desempenho dos resultados encontrados nestes artigos infelizmente não podem ser comparados de forma imparcial entre eles. A comparação das técnicas somente seria possível caso todas utilizassem um mesmo ambiente de teste (BDP e instruções SQL).

A avaliação de todos os trabalhos apresentados neste capítulo nos levou a contrução da Tabela 3.7 que correlaciona os trabalhos de acordo com algumas características em comum.

## 3.5 Considerações Finais

Neste capítulo, foram apresentados importantes trabalhos sobre a Análise de Mutantes SQL, envolvendo principalmente a criação dos operadores de mutação e a aplicação da técnica. Também foram detalhados trabalhos que tratam sobre a geração ou redução de dados para construção de bons casos de teste no contexto da Análise de Mutantes SQL.

**Tabela 3.7:** *Correlação entre os Trabalhos Relacionados*

| **Características** | **Trabalhos Relacionados** | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | [10] | [11] | [19] | [23] | [34] | [43] | [44] | [47] | [48] | [50] | [51] | [38] | [42] |
| Propõem operadores de mutação para instruções SQL | X | X | | X | | X | | | | X | | | X |
| Realizam experimentos aplicando a técnica de Análise de Mutantes | X | | X | X | X | | X | X | | X | X | X | X |
| Constroem algum tipo de ferramenta para dar suporte na aplicação da técnica de Análise de Mutantes SQL | X | | X | | | X | | | X | X | X | | |
| Abordam sobre redução/seleção de bases de dados no contexto de Análise de Mutantes SQL | | | | | X | | | X | | | | | |
| Abordam sobre geração de bases de dados no contexto de Análise de Mutantes SQL | | | | X | | | X | | | | | X | X |

Além disso, o capítulo destaca algumas importantes características e/ou conclusões de cada trabalho que merecem maior atenção para esta dissertação. Neste sentido, a importância do uso de bases de dados reduzidas para a realização de testes pode ser destacada, bem como o esforço na geração de ambientes de teste para a aplicação da técnica e algumas dificuldades na realização de experimentos.

CAPÍTULO 4

# Proposta de *Benchmark*

*"Todo bom desempenho começa com objetivos claros."*
*Ken Blanchard*

Este capítulo apresenta uma proposta de *benchmark*[1] para possibilitar que técnicas de busca para redução/seleção de dados, no contexto de Análise de Mutantes SQL, possam ser avaliadas e comparadas.

Inicialmente, realiza-se uma contextualização do problema envolvido, apresentando explicações sobre a complexidade e custo para realizar reduções de bases de dados. A definição de uma instância do problema é exposta e detalhada. Também são listadas as características e necessidades de um ambiente adequado para avaliação das técnicas de busca no contexto do problema.

Além do objetivo, da composição e das caractarísticas definidas para o *benchmark*, este capítulo também apresenta o processo e as atividades que foram estabelecidas para orientar a construção do *benchmark*, bem como para orientar a execução dos experimentos necessários para a geração de seus entregáveis. Ao final do capítulo é apresentada uma ferramenta de apoio implementada para auxiliar o processo de construção e execução dos experimentos.

[1] Por definição, considera-se um *benchmark* como sendo a composição de uma estrutura (cenário) juntamente com um conjunto de instâncias de problemas. Além disso ele também é composto por alguns resultados gerados por métodos aleatórios a fim de fornecer uma referência padrão de complexidade para cada instância de problema.

## 4.1 Contextualização

Instruções SQL (*Structured Query Language*) são componentes essenciais em Aplicações de Banco de Dados (ABDs), por tratar justamente da interface de acesso entre a aplicação e o repositório de dados. Apesar da linguagem SQL ser relativamente simples e bem consolidada, erros sintáticos ou semânticos são muito comuns durante a sua codificação [7]. Por este motivo, empregar esforços para identificar a presença de defeitos nestas instruções é tão importante quanto testar qualquer outro componente do software.

Porém, considerando as particularidades das ABDs, devem ser adotadas técnicas e/ou critérios de teste que consigam garantir uma boa cobertura, ou seja, casos de teste com um grande poder de revelação de defeitos em instruções SQL. Outro fator importante é o uso de técnicas que mantenham um custo computacional e operacional aceitáveis, mas sem afetar negativamente na qualidade dos casos de teste utilizados.

A Análise de Mutantes é um critério que pode ser utilizado para este propósito, funcionando como uma maneira de mensurar o quanto um conjunto de testes está adequado e, consequentemente, orientando na evolução e definição de novos testes [32].

Em 2005, Chan et al. [11] desenvolveram uma abordagem onde a Análise de Mutantes foi utilizada para testes de instruções de consulta SQL através de operadores de mutação específicos para tal finalidade, fazendo com que o programa *P* em teste seja uma instrução SQL. Diversos outros trabalhos [50] [10] [42] [43] também foram realizados com o objetivo de criar e propor novos operadores de mutação para instruções SQL.

Além dos operadores de mutação, outro fator essencial para possibilitar a aplicação da Análise de Mutantes em ABD são os dados de entrada utilizados como casos de teste. No caso de instruções SQL, os dados de entrada são instâncias de um banco de dados denominadas Bases de Dados de Teste (BDTs). Dois aspectos pertinentes às BDTs são de grande relevância [10]:

1. O **tamanho** da BDT, pois impacta diretamente no custo de aplicação do teste;
2. O **conteúdo** da BDT, pois determina quais defeitos das instruções SQL poderão ser revelados.

Basicamente existem duas maneiras de construir BDTs. Elas podem ser geradas ou podem ser selecionadas de uma Base de Dados de Produção (BDP). No caso da geração, este processo acontece de forma *ah-hoc* ou orientado por alguma abordagem/técnica específica que realiza a geração usando regras para chegar em resultados que consigam matar muitos mutantes [44] [42]. Já a seleção, funciona a partir da extração dos dados de

uma BDP para compor os dados da BDT. Ou seja, reduzindo a BDP para criar a BDT.

### 4.1.1 Redução de Bases de Dados

Considerando que ambientes de produção contêm grandes volumes de dados, compor uma BDT com todas as tuplas de uma BDP pode ser extremamente oneroso para a execução dos testes [19]. Isto seria o mesmo que usar a própria BDP como base de dados de teste. Neste caso, durante a Análise de Mutantes SQL por exemplo, todos os mutantes de uma instrução SQL teriam que ser executados diretamente na BDP, causando grande impacto no custo. A Figua 4.1 apresenta uma representação desta situação.

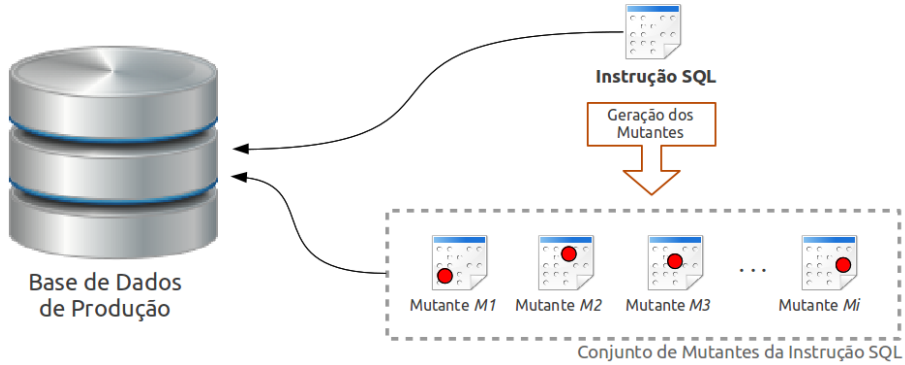


**Figura 4.1:** *Base de Dados de Produção usada como base de dados de teste na Análise de Mutantes SQL*

Sendo assim, o ideal é que uma base de teste seja formada a partir da seleção de um pequeno subconjunto de dados da BDP, ou seja, fazendo uma **redução** da base de produção para formar a BDT, diminundo assim o custo do teste, tanto para a Análise de Mutantes SQL, quanto para outras técnicas de teste para ABD. A Figura 4.2 ilustra esse cenário onde o custo do teste seria reduzido.

Por outro lado, apesar do custo mais elevado, uma BDP tem a vantagem de ser um bom caso de teste para matar mutantes, pois ela é o ambiente que representa o cenário real de dados para os comandos SQL em teste, possibilitando um número maior de situações que sensibilizem os defeitos presentes nos mutantes. Porém, Tuya et al. [50] [47] afirmam que um simples e pequeno caso de teste (BDT) muitas vezes é suficiente para simular diferentes situações de dados e, com isso, conseguir detectar muitos defeitos, mas destacam o risco de omissão de defeitos caso a BDT não seja adequada.

Muitas vezes uma simples redução aleatória da BDP não é suficiente para garantir que sejam selecionadas tuplas (e suas combinações) que conseguirão colocar em evidência os defeitos presentes nas instruções SQL sendo testadas. Derezinska [19] reforça a importância de realizar reduções que mantenham o mesmo poder de detecção de

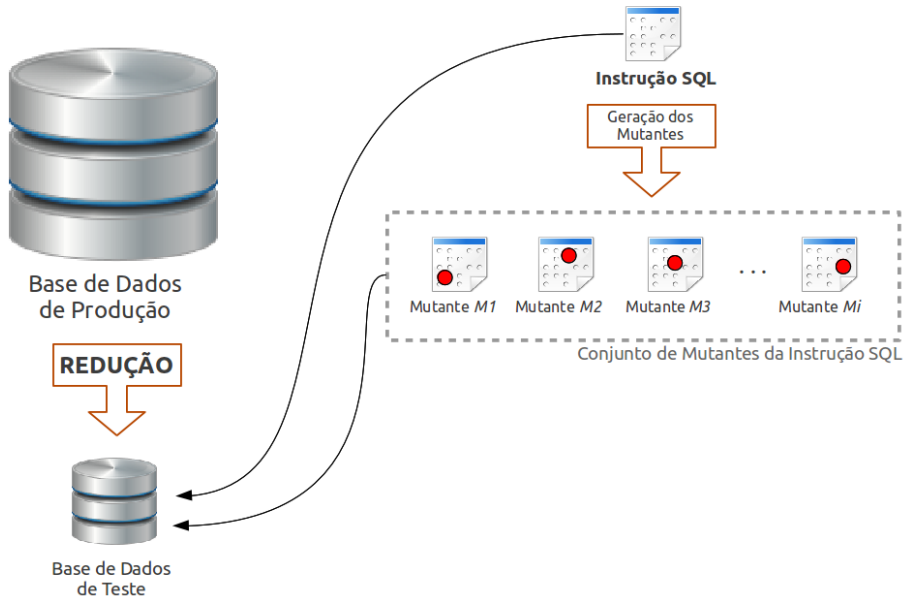


**Figura 4.2:** *Base de Dados Reduzida usada como base de teste na Análise de Mutantes SQL*

defeito da BDP. Sendo assim, realizar a redução de uma BDP, garantindo que não haverá perdas significativas com respeito à descoberta de defeitos, é um problema relevante em qualquer técnica de teste para Aplicações de Banco de Dados.

### 4.1.2 Espaço de Busca e Instância do Problema

O contexto principal do problema no qual esta dissertação está inserida é a criação de **BDTs adequadas**, obtidas a partir da **redução** de uma BDP, utilizando a Análise de Mutantes SQL como critério de avaliação da qualidade das BDTs geradas. Ou seja, não basta simplesmente fazer uma redução aleatória da BDP, o desafio é realizar reduções da BDP, gerando pequenas BDTs que mantenham o custo do teste mais baixo, porém com o poder de revelação de defeitos tão adequado quanto da BDP.

Outra faceta importante do problema é o próprio custo necessário para a realização da redução. Dependendo do volume de dados da BDP, do grau de redução que se pretende aplicar, bem como as instruções SQL que estão sendo testadas, a complexidade envolvida pode ser muito elevada e consequentemente o custo também.

O primeiro fator que vai impactar diretamente neste custo é o espaço de busca[2] de onde pretende-se encontrar boas BDTs. Usando a Teoria dos Conjuntos, pode-se considerar a BDP como sendo um conjunto de tuplas e seus possíveis subconjuntos

[2]Espaço de busca é o conjunto de todas as possíveis soluções de um problema de onde pretende-se encontrar as boas solução. Cada possível solução é considerada como um ponto no espaço de busca.

seriam as BDTs. Em um primeiro momento, sem determinar restrições de tamanho destes subconjuntos, usando uma BDP com $n$ tuplas, o espaço de busca seria e $2^n$ BDTs (pontos). A Figura 4.3 ilustra como seria um espaço de busca no contexto da pesquisa.

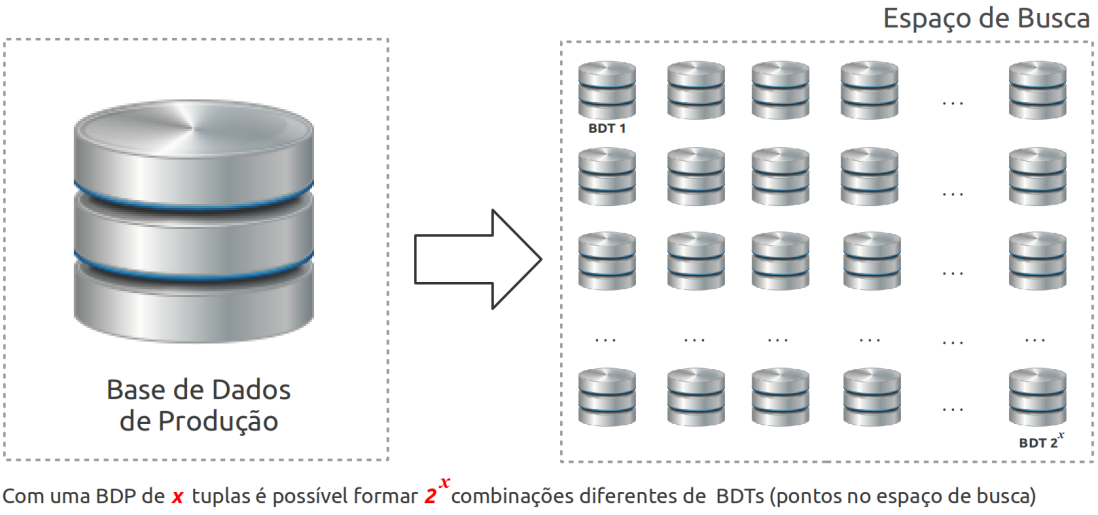


**Figura 4.3:** *Espaço de Busca com possíveis pontos de solução de reduções da BDP*

Todavia, tratando-se de redução de dados, subconjuntos com tamanhos grandes não são desejáveis. Logo, o ideal é que seja limitado o tamanho máximo que uma BDT pode ter, considerando que tamanhos maior do que o estabelecido não vão proporcionar os benefícios esperados da redução. Neste caso, com a restrição de um tamaho específico do subconjunto, é usada uma combinação simples para mensusar o espaço de busca:

$$C_{n,p} = \frac{n!}{p!(n-p)!}$$

Sendo:

$n$ : quantidade de tuplas na BDP

$p$ : quantidade de tuplas em cada BDT (tamanho do subconjunto)

Mesmo com a combinação simples, o problema ainda é exponencial, gerando números astronômicos quando se tratando de bases com grandes volumes de dados. Imaginando uma BDP com 100.000 tuplas e considerando que pretende-se formar uma BDT com 1% destas tuplas, ou seja, com apenas 1000 tuplas, o espaço de busca seria de aproximadamente[3] $99^{1000}$ pontos, que seriam possíveis combinações de BDTs. Percorrer e

[3]Para uma melhor noção do tamanho do espaço de busca, foi feita matematicamente uma conversão aproximada da fórmula de combinação simples para uma forma exponencial expressando um número menor. Porém, mesmo sendo menor, ainda se trata de um número astronômico.

avaliar todos estes pontos para “descobrir” quais são as BDTs promissoras é computacionalmente intratável. Portanto, são necessárias outras abordagens para resolver este problema de busca, e nestas abordagens, quanto maior o espaço de busca, maior o custo computacional.

Além do tamanho do espaço de busca, outro fator que compõe a complexidade deste problema são as instruções SQL sendo testadas juntamente com os seus mutantes. A combinação da “BDP + Instrução SQL com seus Mutantes” forma uma **instância do problema**. Dependendo das características desta instância, a quantidade de pontos do espaço de busca que conseguem resultar em uma boa solução pode aumentar ou diminuir. Quanto menos pontos adequados para a instância do problema no espaço de busca, maior a dificuldade de encontrar boas soluções, logo, maior a complexidade da instância do problema.

A complexidade das instâncias do problema pode ser muito variável. Um único mutante no conjunto de vários mutantes, é suficiente para determinar se a instância do problema é mais fácil ou mais difícil. Pode-se ter um determinado exemplo onde um mutante é extremamente difícil de ser morto, ou seja, são pouquíssimos pontos no espaço de busca que tem capacidade de revelar o defeito do mutante. A instância do problema do qual ele participa tem então uma complexidade mais alta. Se esse único mutante for retirado, não fazendo mais parte da instância do problema, tem-se agora uma outra instância do problema, porém com uma complexidade mais baixa. Da mesma forma, se uma única tupla for retirada ou acrescentada da BDP será possível criar uma outra instância do problema com maior ou menor complexidade.

### 4.1.3 Técnicas de Busca e Ambientes de Avaliação

Dependendo da complexidade da instância do problema, podem estar envolvidos aspectos combinatórios que vão gerar grandes espaços de busca com poucos pontos de boas soluções. No contexto do presente trabalho, seriam poucas BDTs que consigam matar todos ou a maioria dos mutantes SQL.

Este cenário vai exigir o uso de alguma abordagem que explore de forma inteligente o espaço de busca a fim de realizar a seleção/redução dos dados na formação de BDTs adequadas. Neste caso, uma das alternativas seria o uso de **técnicas de busca**[4].

[4]Técnicas de Busca são metaheurísticas que visam encontrar uma boa solução, eventualmente a ótima, para problemas onde o uso de métodos exatos se torna restrito. A técnica funciona através de algum processo que converge rapidamente para soluções interessantes dentro de um espaço de busca.

Como exemplos de técnicas de busca, pode-se citar as metaheurística de busca local, onde a exploração do espaço de soluções é feita por meio de movimentos, os quais são aplicados a cada passo sobre a solução corrente, gerando outra solução promissora na sua vizinhança. Outro tipo de técnica são as metaheurísticas de busca populacional, que consiste em manter um conjunto de boas soluções e combiná-las de forma a tentar produzir soluções ainda melhores.

Entretanto, independente de qual técnica de busca seja adaptada e/ou construída para resolver o problema da redução de dados, todas deverão inevitavelmente utilizar um ambiente que possibilite a aplicação da técnica afim de atestar o seu funcionamento e avaliar seu desempenho na geração de BDTs de qualidade.

Espera-se que este ambiente tenha no mínimo:

1. Uma Base de Dados com grande volume para ser a BDP de onde os dados serão selecionados para formar as BDTs;
2. Um conjunto de instruções SQL para usar como objeto alvo de onde serão investigados os defeitos;
3. O conjunto de mutantes de cada instrução SQL para permitir o uso da Análise de Mutantes como critério de avaliação de qualidade das BDTs;
4. Cálculo do escore de mutação para cada instrução SQL usando a BDP como base de teste.

A técnica de busca deve atuar neste ambiente explorando o espaço de busca realizando reduções na BDP para gerar as BDTs. Com cada BDT aplica-se a análise de mutantes na instrução SQL sendo testada, verificando quais mutantes a BDT consegue matar. Basicamente, o resultado (escore de mutação) vai indicar a qualidade da base reduzida (BDT). Quanto mais próximo do escore de mutação da BDP, melhor o poder de detecção de defeitos da BDT e mais adequada ela está. A Figura 4.4 apresenta um esquema básico de um ambiente onde seria possível avaliar o desempenho de uma técnica de busca através da avaliação das BDTs geradas.

Ainda sobre a Figura 4.4, o Passo 1 representa a exploração do espaço de busca e a seleção de tuplas da BDP, o Passo 2 é a criação da BDT a partir dos dados selecionados e o Passo 3 representa a aplicação da técnica de Análise de Mutantes para verificar a qualidade da BDT. O Passo 4 mostra que este ciclo pode acontecer várias vezes, gerando diversas BDTs. Ao final, além de verificar individualmente a adequação de cada BDT através do escore de mutação, é possível também verificar a média do escore de mutação de todas as BDTs geradas.

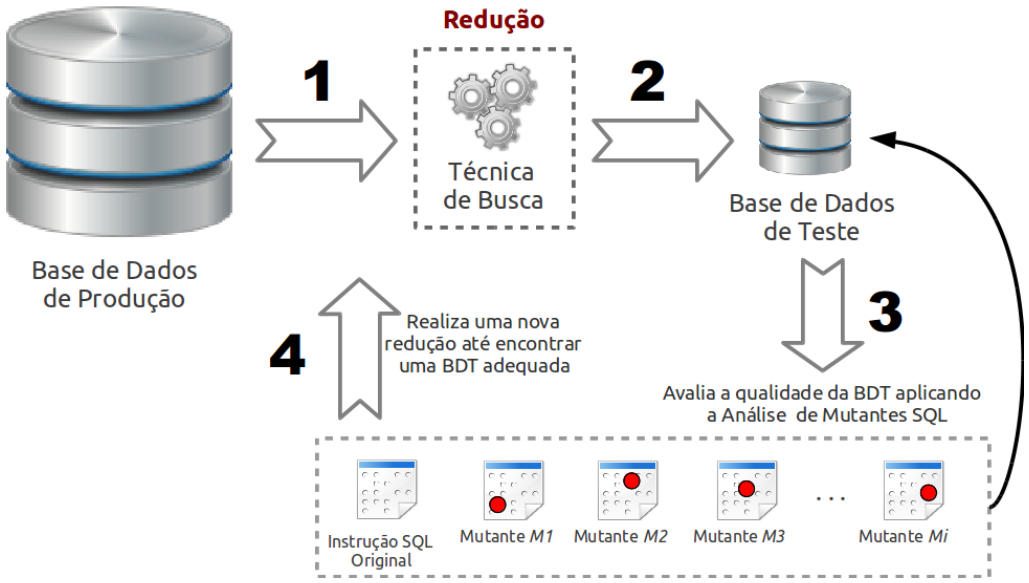


**Figura 4.4:** *Um Ambiente mínimo para avaliar o desempenho de uma técnica de busca no contexto de redução de bases de dados*

A Figura 4.5 apresenta o resultado de um exemplo hipotético, onde alguma técnica de busca (por exemplo, um Algoritmo Genético) em um ambiente semelhante ao apresentado na Figura 4.4, criou para uma instância de problema cinco BDTs. Ao final, cada BDT gerada apresenta um valor de escore de mutação que pode ser comparado com o escore de mutação da BDP. Neste exemplo, a média do escore de mutação das 5 BDTs foi de 0,65, ficando bem abaixo do escore da BDP. Porém, a BDT4 alcançou um escore de 0,83, conseguindo uma qualidade de detecção de defeitos semelhante à BDP, que é a principal referência de comparação.

Além do valor do escore de mutação da BDP para cada instrução SQL, seria também desejável compor o ambiente de avaliação com uma relação de valores de escore de mutação obtidos a partir de BDTs geradas aleatoriamente, ou seja, através de um Método Aleatório (MA). Considerando que o método aleatório tem o menor custo computacional [33], espera-se então que outros métodos mais caros, como as técnica de busca, tenham melhores resultados, e isso pode ser verificado comparando os escores alcançados com a técnica de busca com os escores do MA.

Porém, é necessário realizar várias medições com o método aleatório para ter conclusividade nos resultados, por este motivo é necessário gerar para cada instância de problema diversas BDTs aleatórias. Com este conjunto de BDTs, é possível ter uma média do escore de mutação através do MA.

Na Figura 4.6 é apresentado novamente o mesmo resultado ilustrado na Figura 4.5, porém acrescido de mais um valor para referência, que é a média do escore de

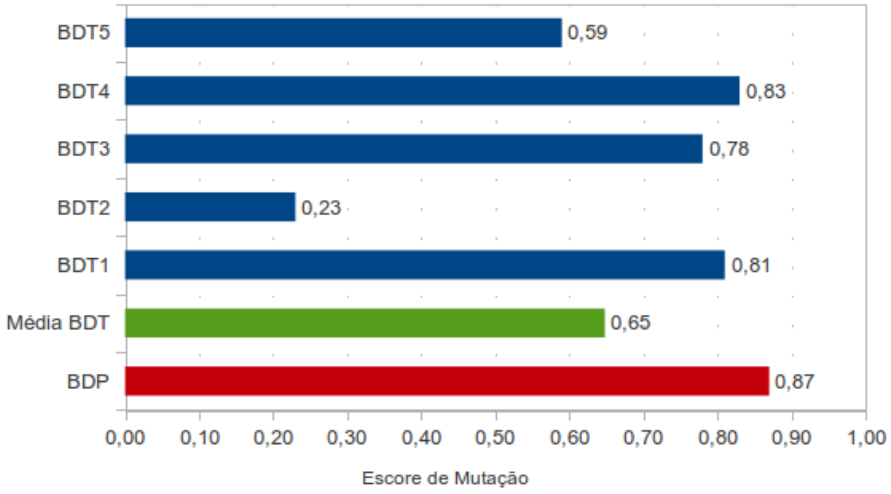


**Figura 4.5:** *Exemplo de avaliação de desempenho de BDTs geradas*

mutação alcançado por supostas BDTs geradas pelo MA. É possível observar agora que a técnica de busca teve a competência de gerar de BDTs com uma média de escore de mutação bem acima do MA, sinalizando com isso uma certa propensão a ter bons resultados.

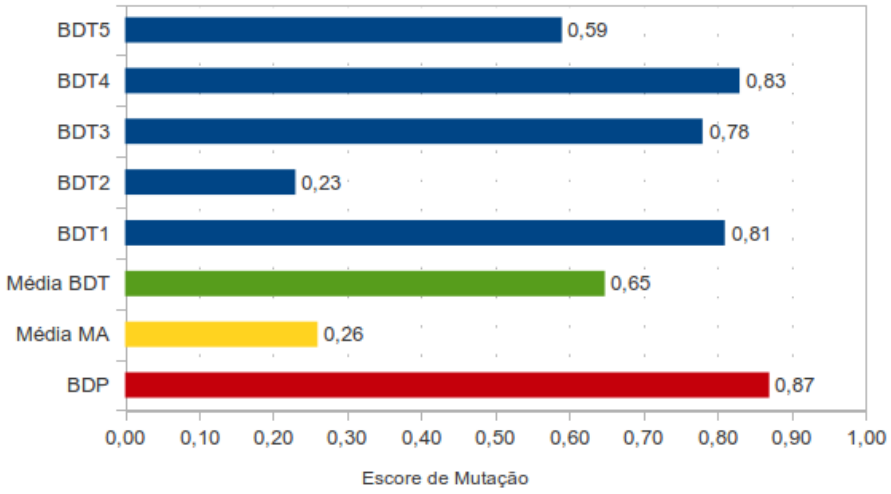


**Figura 4.6:** *Exemplo de avaliação de desempenho de BDTs geradas comparando com Método Aleatório*

As BDTs e resultados apresentados nos exemplos anteriores, foram supostamente geradas e avaliadas dentro de um ambiente específico para esta finalidade. Todavia, existem alguns problemas que devem ser considerados em relação aos diferentes ambientes, que podem ser usados na avaliação do desempenho de técnicas de busca para redução.

Dentre eles, vale destacar como os mais relevantes no contexto desta pesquisa:

1. Demanda de tempo e esforço para a construção de um ambiente adequado, tanto no que diz respeito a BDP, às instruções SQL, aos mutantes e às gerações de BDTs através de métodos aleatórios;
2. Uso de um ambiente sem instâncias do problema desafiadoras o suficiente para avaliar e exercitar de forma criteriosa a técnica de busca. Com um ambiente desfavorável, corre-se o risco de avaliar erroneamente uma técnica, podendo até fazer com que a mesma tenha tendência a obter somente bons resultados;
3. Falta de um ambiente padronizado, inviabilizando comparações e avaliações de desempenho entre técnicas de busca que usaram ambientes distintos.

Problemas semelhantes aos citados foram enfrentados na pesquisa realizada por Monção et al. [34]. Durante o desenvolvimento deste trabalho, procurou-se na literatura algum ambiente adequado a este contexto, onde fosse possível aplicar a técnica de busca desenvolvida, neste caso, um Algoritimo Genético (AG). Na Seção 3.3.2 foram apresentados maiores detalhes sobre as características e condução desta pesquisa.

A principal necessidade era realmente utilizar o AG em um ambiente que permitisse exercitar a técnica e avaliar o seu desempenho para redução de dados. Na época, não foram encontrados trabalhos e pesquisas com características que atendessem à necessidade da pesquisa [34]. Diante disso, um grande esforço para a criação de um ambiente foi empregado, e somente depois foi possível dar início efetivo aos trabalhos para construção do AG. Além disso, devido à falta de um ambiente padronizado, não foi possível comparar os resultados obtidos com resultados de outras abordagens.

Esta ausência de referências foi um dos principais motivadores para a realização da pesquisa desta dissertação, que resultaria na construção do ambiente adequado à necessidade presente na época.

## 4.2 Objetivos e Composição do *Benchmark*

Considerando os problemas e motivações citados na seção anterior, o principal objetivo da pesquisa consiste na construção e disponibilização de um *benchmark*, que funcione como um **ambiente de referência**, contribuindo para fornecer todos os elementos necessários para apoiar a **avaliação de desempenho** de qualquer técnica de busca, no contexto de redução de bases de dados para testes de instruções SQL.

O ambiente deste *benchmark* se propõe a ser padronizado, possuindo instâncias do problema invariáveis e bem definidas. Espera-se também que ele tenha uma estrutura

com grande volume de dados, formado por uma ou mais BDPs, e um conjunto de instruções SQL com seus respectivos mutantes. Além disso, para cada instância do problema do *benchmark*, serão realizados diversos experimentos que visam conduzir reduções aleatórias da BDP, afim de gerar conjuntos de BDTs de diferentes tamanhos. A qualidade de cada BDT, e consequentemente de cada conjunto, será calculada e avaliada utilizando a Análise de Mutantes SQL.

O segundo objetivo, a partir da realização dos experimentos, consiste na geração de informações que indiquem a complexidade de cada instrução SQL do *benchmark*, permitindo assim interpretações que levem à elaboração de entregáveis, sendo eles, planilhas de resultados, gráficos e principalmente tabelas de ranqueamento em diferentes perspectivas. Servindo não só como importantes instrumentos de referências para as técnicas de busca mas também ajudando a analisar e responder importantes questões como por exemplo:

1. Quais são as situações que geram instruções SQL mais difíceis no contexto de Análise de Mutantes SQL?
2. Quais características influenciam na complexidade das instâncias do problema?
3. Quais propriedades de uma base de teste pode influenciar na sua capacidade de matar mutantes?

Além disso, espera-se que as informações e resultados gerados na presente pesquisa forneçam relevantes insumos e parâmetros para outros pesquisadores da área de Análise de Mutantes SQL, sendo este então um outro importante objetivo que se deseja alcançar com a criação do *benchmark*.

## Composição

A proposta é construir e disponibilizar um *benchmark* que seja formado por dois cenários, como apresentado na Figura 4.7. No primeiro cenário serão utilizados dados gerados especificamente para compor o *benchmark*. No segundo cenário serão usados dados extraídos da BDP de uma aplicação real.

Em ambos os cenários, a composição padrão será constituída pelos seguintes componentes: (i) Banco de Dados de Produção; (ii) Instruções SQL e Mutantes; e (iii) Resultados do Método Aleatório. A Figura 4.8 ilustra uma abstração simples da composição de cada cenário do *benchmark*.

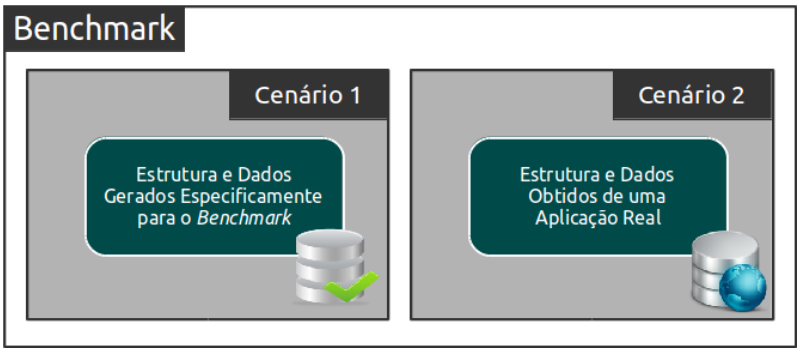


**Figura 4.7:** *Representação gráfica da composição do Benchmark*

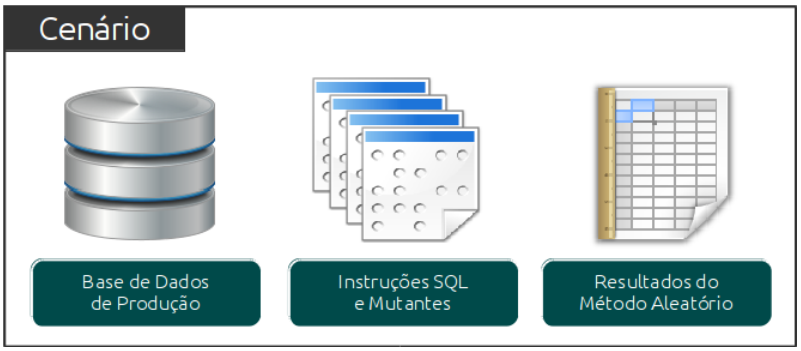


**Figura 4.8:** *Representação gráfica da composição de cada cenário do Benchmark*

Para cada componente serão gerados entregáveis que vão permitir o entendimento e utilização do *benchmark* dentro do que ele se propõe. O conjunto de todos estes entregáveis foi definido e organizado para fisicamente formar o *benchmark*. A listagem detalhada dos entregáveis está disponibilizada no Apêndice B.

## Base de Dados de Produção

Cada cenário possuirá uma instância de banco de dados, composta por tabelas povoadas com grande quantidade de tuplas que formarão a BDP. Como já foi visto, a BDP tem uma relação direta com o **espaço de busca** da pesquisa, pois é dela que serão selecionadas as combinações de tuplas para geração das BDTs.

## Instruções SQL e Mutantes

Baseado na estrutura da BDP, cada cenário do *benchmark* terá um conjunto de instruções SQL de consulta, que servirá como objeto alvo de testes. O objetivo é identificar os defeitos destas instruções através da Análise de Mutantes SQL. Sendo assim, para cada instrução SQL serão gerados os seus mutantes.

Os mutantes serão gerados utilizando a ferramenta *SQLMutantion* [48], que implementa os operadores de mutação propostos por Tuya et al. [50]. A escolha desta ferramenta se justifica por ela ser a mais utilizada na literatura para automatização de geração de mutantes SQL. Além disso, a ferramenta tem a capacidade de eliminar muitos mutantes equivalentes.

## Resultados do Método Aleatório (MA)

No contexto do presente trabalho, o problema consiste na seleção de um subconjunto de tuplas da BDP para formar uma BDT. Conforme citado anteriormente, pode-se considerar que o menor custo computacional para realizar esta geração seria a seleção aleatória de tuplas da BDP para formar a BDT. Sendo assim, justifica-se a adoção de qualquer outro método mais caro para resolver este problema se, e somente se, este método obtiver um resultado significativamente melhor que o MA.

Serão geradas então várias medições aleatórias para cada instância de problema dos dois cenários do *benchmark*. Os resultados obtidos serão importantes e necessários principalmente para:

1. **Verificar a complexidade de cada instância do problema *(BDP + Instrução SQL + Mutantes)*** - Se os resultados do MA forem bons, a instância do problema sendo avaliada tem complexidade baixa, ou seja, é um problema muito fácil de resolver. A verificação desta complexidade é feita comparando o escore de mutação médio do MA com o escore de mutação alcançado pela BDP. Quanto mais distante o escore do MA do escore da BDP, mais complexo e difícil é a instância do problema. Essa avaliação vai permitir ranquear as instâncias pela sua complexidade, além de auxiliar na definição de instruções SQL mais adequadas para compor o *benchmark*, a medida que as medições pelo MA sejam realizadas.

2. **Comparar o resultado de uma técnica sendo avaliada com o MA -** Considerando que uma instância do problema não obteve bons resultados com o MA, justifica-se então o uso de uma técnica mais cara com a expectativa de obter melhores resultados. O desempenho desta técnica será avaliado comparando o resultado alcançado por ela com o resultado do MA e o resultado da BDP. Quanto mais distante do MA e mais próximo da BDP, melhor o desempenho da técnica para aquela instância do problema.

## 4.3 Características do *Benchmark* e de seus cenários

Após avaliar o contexto do problema e as motivações envolvidas, várias decisões foram tomadas acerca de como conduzir a construção do *benchmark* e, principalmente, quais seriam as características necessárias para alcançar o objetivo estabelecido. Também foram definidas algumas diretrizes e características sobre a execução dos experimentos com o Método Aleatório.

Conforme já descrito, uma das primeiras decisões tomadas foi a construção do *benchmark* com dois cenários distintos. O primeiro cenário com uma BDP inspirada em em uma referência na literatura e com os dados gerados especificamente para formar o *benchmark*. E o segundo cenário com uma BDP inspirada e formada por dados obtidos de alguma aplicação do mundo real.

Ter estes dois cenários é uma das características mais importantes do *benchmark*, pois acredita-se que eles permitirão uma variabilidade interessante nos dados, nas instâncias do problema e consequentemente nos resultados obtidos através dos experimentos com o Método Aleatório.

Além da origem dos dados da BDP, cada cenário do *benchmark* contém outras características específicas que os diferem. A condução da construção e execução dos experimentos de cada cenário também têm suas particularidades. Porém, de forma geral, os dois cenários compactuam com a maioria das características e diretrizes estabelecidas para a construção de cada componente e a realização dos experimentos.

### 4.3.1 Base de Dados de Produção (BDP)

Em ambos os cenários, para formar a BDP, optou-se por utilizar modelos de dados com pelo menos duas tabelas, permitindo assim o uso de instruções SQL de consulta com comandos do tipo *JOIN*. Além disso, cada tabela deve possuir uma quantidade elevada de tuplas para simular uma BDP de grande volume. Sendo assim, decidiu-se que cada BDP deve ter no mínimo 100.000 tuplas.

#### Cenário 1 - EMPRESA

Para o primeiro cenário, foi realizada uma pesquisa na literatura em busca de um modelo conceitual que atendesse às necessidades da pesquisa e que fosse de fácil entendimento. Optou-se pela utilização do modelo de banco de dados **EMPRESA** proposto no livro *Fundamentals of Database Systems* de Elmasri e Navathe [41]. Este

modelo é muito utilizado em materiais acadêmicos e é de fácil compreensão por se tratar de um negócio relativamente simples. A Figura 4.9 apresenta o modelo conceitual de entidade-relacionamento do banco EMPRESA que foi criado por Elmasri e Navathe.

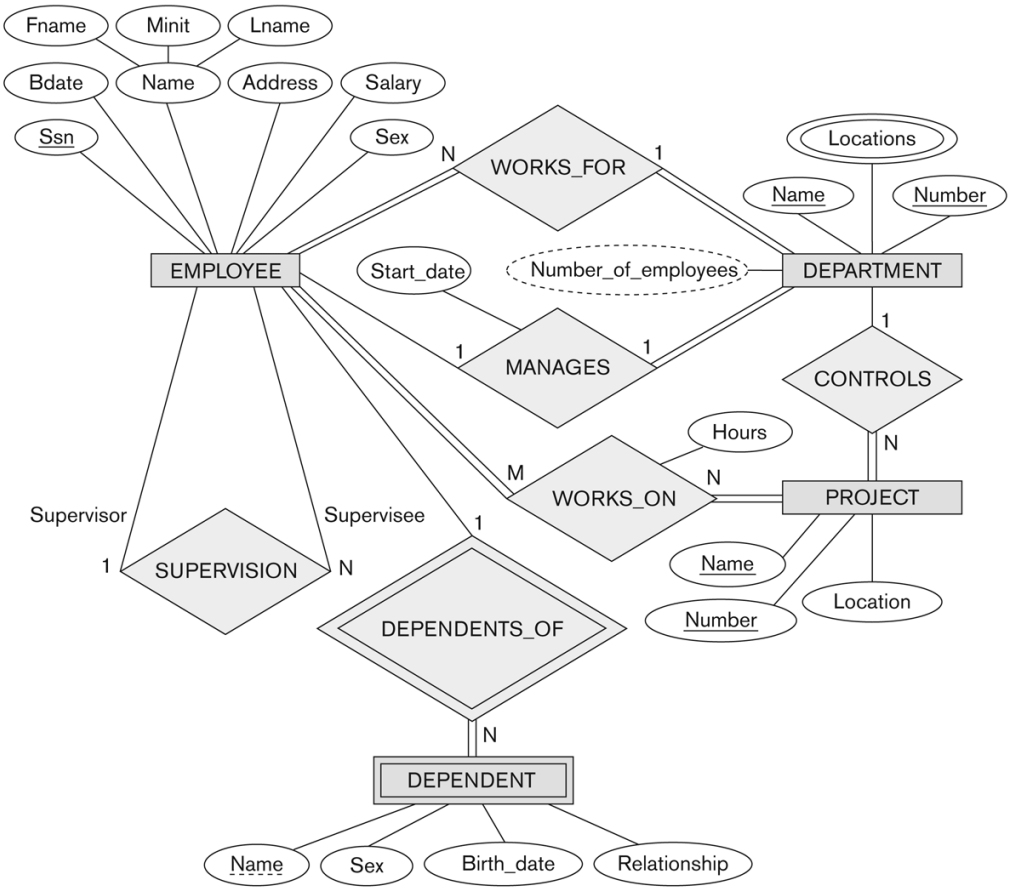


**Figura 4.9:** *Modelo de Entidade e Relacionamento EMPRESA*

Também é proposto por Elmasri e Navathe uma definição física de todas as tabelas, contendo a descrição e características dos campos, restrições e chaves. A Figura 4.10 apresenta o esquema relacional das tabelas criado pelos autores a partir do mapeamento do modelo EMPRESA [41].

Sendo assim, a estrutura da BDP do Cenário 1 foi baseada no modelo e nas definições físicas das tabelas. Porém, apesar do modelo EMPRESA possuir seis tabelas, optou-se em criar a BDP utilizando somente duas tabelas: *EMPLOYEE* e *DEPARTMENT*. Esta foi uma decisão estratégica durante a elaboração do *benchmark* com o objetivo de evitar problemas de integridade referencial e facilitar o processo de seleção aleatória de dados da BDP. Outra decisão realizada, também procurando evitar problemas de integridade, foi o relaxamento da restrição do auto-relacionamento (*supervision*) da tabela

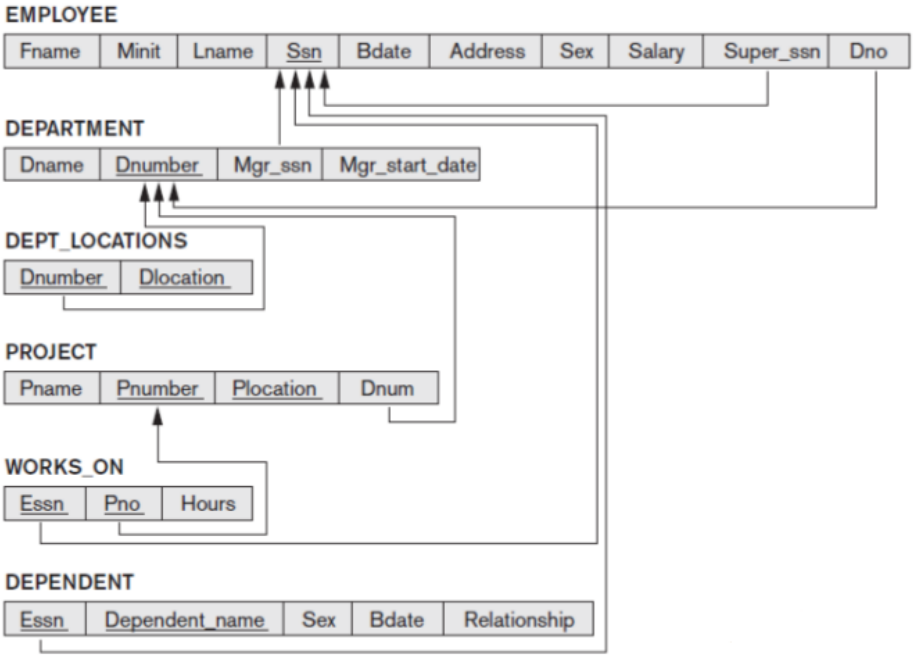


**Figura 4.10:** *Esquema Relacional de Tabelas do Banco EMPRESA*

*EMPLOYEE*.

Elmasri e Navathe não fornecem ou propõe nenhum tipo de carga padrão para as tabelas do banco EMPRESA. Realizou-se então, sistematicamente, a geração aleatória de tuplas especificamente para popular as tabelas usadas na BDP. Neste processo, foram respeitadas as regras e restrições de campos estabelecidas no modelo físico. Além disso, apesar de ser aleatória, procurou-se gerar tuplas com valores de cada coluna obedecendo regras pré-estabelecidas, que buscam manter similaridade com um ambiente de uma aplicação real. Ao total, foram geradas aproximadamente 100.000 tuplas para a BDP.

Diante da liberdade que o Cenário 1 forneceu para gerar e manipular os dados, especificamente para o *benchmark*, decidiu-se que na BDP existirão todas as tuplas necessárias para matar todos os mutantes gerados para este Cenário. Desta forma, para todas as instruções SQL do Cenário 1, os escores de mutação usando a BDP serão iguais à 100%. Para isso, à medida que os experimentos forem realizados, a BDP será manipulada para garantir o escore de 100%. Com cada nova instrução SQL será feito o cálculo de escore de mutação usando a BDP. Os mutantes sobreviventes serão avaliados e a partir disso a BDP será modificada para ter a capacidade de matá-los. Esta modificação pode acontecer incluindo novas tuplas ou modificando tuplas já existentes.

Quando não for possível modificar a BDP para matar um mutante sobrevivente, este mutante será classificado como equivalente. Essa situação pode acontecer quando a modificação necessária na BDP infringe alguma restrição do modelo de dados. Por exemplo, pode ser gerado um mutante que só é possível ser morto se existir pelo menos

uma tupla com o campo da chave primária igual a NULO. Como esta situação não é permitida, considerou-se que este mutante é equivalente.

**Cenário 2 - UFG**

Para o Cenário 2, foram encontradas algumas dificuldades na obtenção da BDP de uma aplicação real. O fato é que na maioria dos casos, mesmo sendo para pesquisas acadêmicas, existe uma forte restrição por parte das empresas e instituições em fornecer uma base de dados que possua informações de seus clientes, fornecedores, funcionários, etc. Para amenizar este receio, decidiu-se modificar as BDPs sugerindo que fossem retirados e/ou mascaradas todas as informações que poderiam identificar ou expor qualquer pessoa.

A primeira tentativa foi a solicitação de parte da BDP do sistema de folha de pagamento do estado de Goiás (RHNET). Foi enviado um ofício à Superintendência de Tecnologia da Informação da SEGPLAN[5]. Neste ofício, foi exposto a necessidade da BDP para a pesquisa destacando que não seria necessário na BDP informações pessoais, financeiras ou qualquer tipo de dado que pudesse identificar os servidores (funcionários) do estado.

Em paralelo, foi feita também uma solicitação semelhante à Universidade Federal de Goiás (UFG). Neste caso, solicitou-se parte da BDP do sistema acadêmico da UFG, também sem nenhuma informação que pudesse expor a privacidade dos alunos e professores.

Foi obtido um retorno mais rápido por parte da UFG, que por sua vez, forneceu um conjunto de dados do sistema acadêmico, referente ao ano de 2011 e em forma de arquivos. Este conjunto de dados é o mesmo que é enviado ao MEC[6] todos os anos, contendo informações do histórico de participação dos alunos e professores nas disciplinas, sendo que não existe qualquer informação neste conjunto de dados que possibilite a identificação (nome, documento, etc) dos alunos ou professores.

Baseando-se nos arquivos fornecidos, criou-se a BDP do Cenário 2. Da mesma forma que foi feito para o Cenário 1, utilizou-se somente duas tabelas para esta base. A Figura 4.11 apresenta o modelo físico das tabelas da base de dados UFG. Ao total, a quantidade de tuplas da BDP do Cenário 2 é de aproximadamente 250.000 tuplas.

Outra decisão importante para o Cenário 2 foi manter os dados da BDP da mesma

[5]Secretaria de Gestão e Planejamento do Estado de Goiás
[6]Ministério da Educação e Cultura

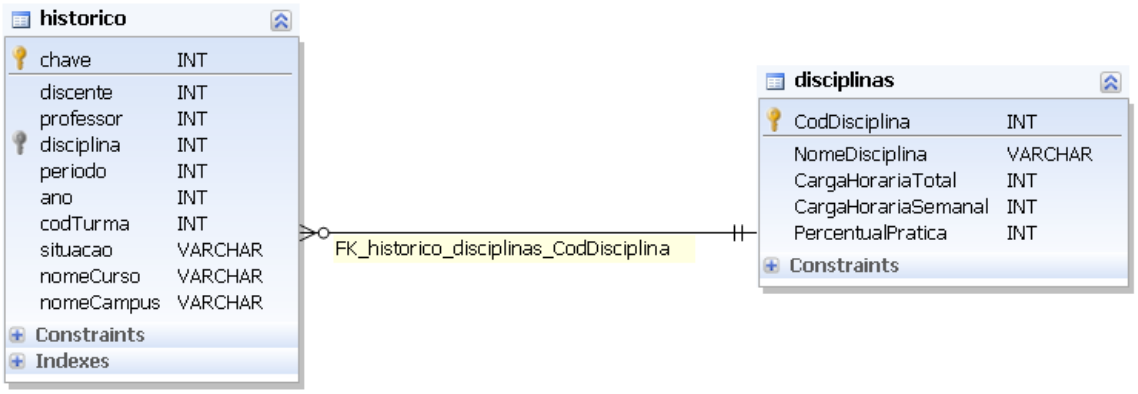


**Figura 4.11:** *Esquema Relacional de Tabelas do Banco UFG*

forma que foram extraídos da aplicação real. Com isso, diferente do Cenário 1, o escore de mutação da BDP pode ser diferente para cada instância do problema, pois não são feitas intervenções na BDP para matar todos os mutantes. Também não são classificados como equivalentes mutantes que não foram mortos, independente dos resultados da Análise de Mutantes com a BDP.

### 4.3.2 Instruções SQL e Mutantes

As escolhas das instruções SQL que irão compor cada Cenário do *benchmark* é muito importante pois impacta diretamente na qualidade das instâncias do problema. Uma das contribuições do *benchmark* é justamente fornecer um conjunto de instâncias de problemas que levem à uma avaliação criteriosa das técnicas de busca.

Por este motivo, surgiu a necessidade de elaborar algumas diretrizes para guiar a escolha e definições das instruções SQL. Os dois principais objetivos observados para isso foram: (i) Exercitar, pelo menos uma vez, todos os operadores de mutação previstos na referência adotada; e (ii) Criar, em sua maioria, instruções SQL que gerem mutantes resistentes.

No primeiro momento, adotou-se como meta a criação de no mínimo 10 instruções SQL por Cenário. A partir destas instruções, os mutantes serão gerados na ferramenta *SQLMutation* usando os operadores de mutação propostos por Tuya et al. [50]. Caso não fosse suficiente utilizar todos os operadores de mutação com as 10 instruções, seriam criadas outras instruções até alcançar esse objetivo. Da mesma forma, enquanto não houvesse instruções SQL com mutantes desafiadores, novas instruções também seriam criadas.

As instruções SQL utilizam somente comandos do padrão ANSI[7] SQL3, ficando

[7] *American National Standards Institute*

assim compatíveis com qualquer SBGD relacional. Optou-se também por usar somente instruções SQL de consulta (*SELECT*). Esta escolha se deu por dois motivos: (i) O comando *SELECT* é o mais utilizado em instruções SQL de aplicações comerciais [39]; e (ii) O conjunto de operadores de mutação propostos por Tuya et al [50] utiliza somente comandos do tipo *SELECT*.

Importante destacar também que a intenção é elaborar uma maioria de instruções SQL que gerem mutantes resistentes, levando a criar instâncias de problemas difíceis. Porém, instruções SQL de instâncias fáceis também fazem parte do *benchmark*, ajudando assim em outros tipos de interpretações, como por exemplo, de quais comandos da linguagem SQL não são difíceis de identificar os defeitos.

Uma instrução SQL permite que sejam usados parâmetros em suas consultas, porém, no *benchmark*, todas as instruções foram construídas com valores constantes. Por este motivo o operador de mutação IRP não foi usando em nenhum dos cenários. No Cenário 2, além do operador IRP, também não foram usados os operadores NLF, NLI e NLO, pois estes são exercitados quando utilizado o valor NULL nas instruções SQL, o que não ocorreu devido às características da BDP deste Cenário.

**Cenário 1 - EMPRESA**

Para o Cenário 1, decidiu-se criar as instruções SQL em duas etapas. Na primeira etapa foi criado um grupo de instruções SQL baseado nas diversas instruções de exemplo elaboradas por Elmasri e Navathe [41] para o banco EMPRESA. Estas instruções foram escolhidas priorizando o uso dos operadores de mutação e considerando o acesso somente das tabelas escolhidas para formar a BDP.

Foram selecionadas e adaptadas 9 instruções SQL para o primeiro grupo. Com estas instruções, gerou-se os mutantes e executou-se os experimentos para elas. Tendo em mãos a relação dos mutantes e os resultados dos experimentos, definiu-se manualmente um segundo grupo de instruções SQL inspirados nos: (i) Operadores de mutação que ainda não foram exercitados; e (ii) nas instruções SQL do grupo anterior que tiveram uma maior incidência de mutantes resistentes, caracterizando instâncias do problema mais difíceis. Este segundo grupo foi criado com outras 11 instruções SQL. A Figura 4.12 ilustra esse processo de criação das instruções SQL.

Ao total, para o Cenário 1, foram geradas 20 instruções SQL, que estão listadas no Apêndice C. Maiores detalhes sobre as instruções e a relação dos mutantes gerados, estão disponibilizados nos entregáveis conforme previso na Seção B.0.2.

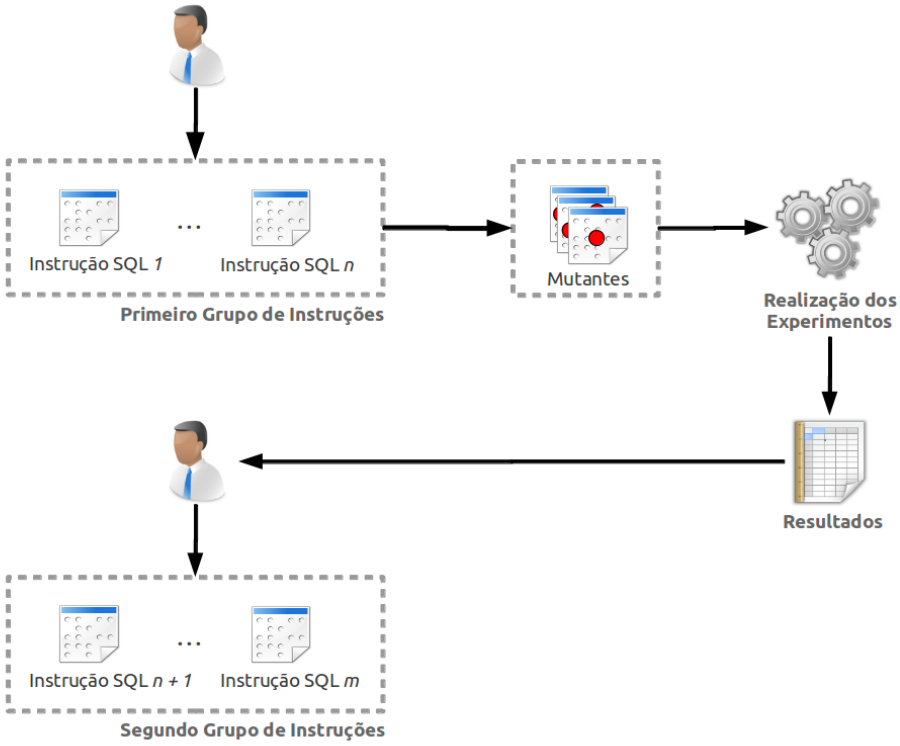


**Figura 4.12:** *Criação das Instruções SQL em duas etapas, sendo a segunda inspirada nos resultados da primeira.*

**Cenário 2 - UFG**

Para o Cenário 2, desejou-se inicialmente criar as instruções SQL inspiradas na mesma aplicação real no qual a BDP foi extraída. Porém, não se teve acesso ao código fonte da aplicação, não sendo possível então avaliar as instruções SQL usadas. Sendo assim, decidiu-se gerar as instruções do Cenário 2 inspiradas nas instruções do Cenário 1. Considerou-se então as características e os tipos de comandos utilizados e fez-se uma adaptação para a BDP do Cenário 2.

Da mesma forma, também priorizou-se cobrir todos os operadores de mutação possíveis e ter em sua maioria instruções SQL com mutantes resistentes. Ao total, 15 instruções SQL para o Cenário 2 foram criadas (listadas no Apêndice C).

### 4.3.3 Experimentos e Resultados do Método Aleatório (MA)

Para cada instrução SQL do *benchmark*, serão realizados vários experimentos para gerar as BDTs através das reduções pelo MA, que são importantes componente do *benchmark*. Cada BDT tem sua qualidade mensurada, fornecendo importantes resultados para a criação dos principais entregáveis.

Por este motivo, devido à sua importância no processo de construção do *benchmark*, decidiu-se realizar a execução dos experimentos em um ambiente que permita o registro de todos os parâmetros, informações e resultados em uma base de dados. A esta base, deu-se o nome de Base de Dados de Experimentos (BDE). Neste ambiente, além da BDE, existe também uma instância de banco de dados para a BDP e uma outra para a geração das BDTs, sendo todas fisicamente independentes e separadas.

O objetivo de cada experimento é a geração de um conjunto de BDTs, com uma determinada quantidade de tuplas, para testar uma única instrução SQL. A quantidade de BDTs no conjunto e a quantidade de tuplas por BDT serão determinadas como parâmetros de cada experimento. Estes parâmetros foram escolhidos considerando a importância e necessidade de:

1. Criar conjuntos de diversos tamanhos para avaliar o comportamento dos resultados em cenários diferentes em relação aos conjuntos de BDTs.
2. O tamanho da BDT (percentual de redução) deve ser significativamente menor que a BDP, até mesmo para justificar a redução de dados para diminuição do custo.
3. Como já foi visto anteriormente, são necessárias várias gerações com o MA para ter conclusividade nos resultados.
4. Não ignorar as restrições e limitações de tempo e processamento para realização dos experimentos.

Em um primeiro momento, de forma empírica, definiu-se um padrão de parâmetros com a quantidade de conjuntos por instrução SQL variando de 1 a 18, a quantidade de BDTs por conjunto variando de 5 a 30 e a quantidade de tuplas por BDT de 1%, 2%, 3%, 5% e 7% do tamanho da BDP. Entretando, para tornar resultados gerados conclusivos, é necessário a realização de várias medições com esta abordagem a fim de se ter, ao final, uma média de resultado com o MA. Estes parâmetros foram determinados para ambos os cenários do *benchmark*.

A intenção foi escolher uma configuração padrão de parâmetros para testes iniciais, e após estes testes verificar se os parâmetros são adequados para os objetivos e restrições da pesquisa. Os testes foram realizados e de modo geral os parâmetros escolhidos tiveram um comportamento interessante. A única alteração necessária em relação à proposta original foi a inclusão de mais um possível tamanho de BDT de 0,1%.

A primeira configuração padrão, definida para os experimentos de cada instrução SQL, está apresentada na Tabela 4.1. Cada linha da tabela é um experimento a ser realizado para cada instrução SQL. A primeira linha por exemplo indica que serão geradas aleatoriamente 5 BDTs para a instrução SQL, e cada BDT conterá 0,1% das tuplas da BDP. Ao total, para cada instrução SQL podem ser geradas 270 BDTs pelo MA.

| Quantidade de BDTs | Percentual da Redução |
|---|---|
| 5 | 0,1% |
| 10 | 0,1% |
| 30 | 0,1% |
| 5 | 1% |
| 10 | 1% |
| 30 | 1% |
| 5 | 2% |
| 10 | 2% |
| 30 | 2% |
| 5 | 3% |
| 10 | 3% |
| 30 | 3% |
| 5 | 5% |
| 10 | 5% |
| 30 | 5% |
| 5 | 7% |
| 10 | 7% |
| 30 | 7% |

**Tabela 4.1:** *Primeiro Conjunto Padrão de Parâmetros dos Experimentos*

Porém, após a execução dos experimentos baseados nos parâmetros da Tabela 4.1, identificou-se algumas instruções SQL que o escore de mutação (média e/ou maior) com o MA, em nenhuma situação, aproximou do escore de mutação da BDP. Para estas situações, decidiu-se expandir os parâmetros e avaliar o comportamento destas instruções SQL com novos experimentos. A Tabela 4.2 apresenta o segundo conjunto de parâmetros de experimentos.

| Quantidade de BDTs | Percentual da Redução |
|---|---|
| 5 | 8% |
| 10 | 8% |
| 30 | 8% |
| 5 | 9% |
| 10 | 9% |
| 30 | 9% |
| 5 | 10% |
| 10 | 10% |
| 30 | 10% |

**Tabela 4.2:** *Segundo Conjunto Padrão de Parâmetros dos Experimentos*

Considerando o primeiro e o segundo grupo de parâmetros, ao total, para cada instrução SQL, podem ser geradas até 405 BDTs de diferentes tamanhos. Para facilitar a avaliação da evolução do escore de mutação destas BDTs, os parâmetros foram definidos de forma crescente. Sendo assim, serão realizados primeiro para cada instrução SQL os experimentos que geram BDTs menores, e em seguida os experimentos com BDTs maiores. A expectativa é que a cada grupo de experimentos com BDTs maiores o escore

de mutação médio da instrução SQL fique mais alto.

Porém, nem todos os experimentos possíveis para uma instrução SQL necessariamente devem ser executados. Definiu-se alguns **critérios de parada** que limitarão as execuções quando resultados com o MA já obtiverem escores de mutação aceitáveis.

Por exemplo, se com a maioria das BDTs de 5% já for possível matar a mesma quantidade de mutantes da BDP, não se justifica criar BDTs maiores, consequentemente mais caras, para avaliar o seu comportamento. Esta avaliação é feita observando-se três valores: (i) média do escore de mutação; (ii) maiores escores de mutação; e (iii) o escore de mutação de um conjunto de BDTs. Nestes casos, quando um destes valores alcança ou se aproxima muito do escore da BDP, o critério de parada é atingido, sinalizando que não é mais necessário realizar novos experimentos para aquela instrução SQL.

A Figura 4.13 apresenta um gráfico onde mostra a evolução do escore de mutação de uma instrução SQL onde o escore com a BDP é igual a 0,9545. Pode-se observar no gráfico que este mesmo escore da BDP já foi alcançado em algum experimento por uma BDT de apenas 1% da BDP, porém a média do escore das BDTs ainda estava distante. Com 5% a média de escores das BDTs geradas aleatoriamente já está muito próxima do escore de mutação da BDP, sinalizando com isso que não é mais necessário avaliar BDTs maiores.

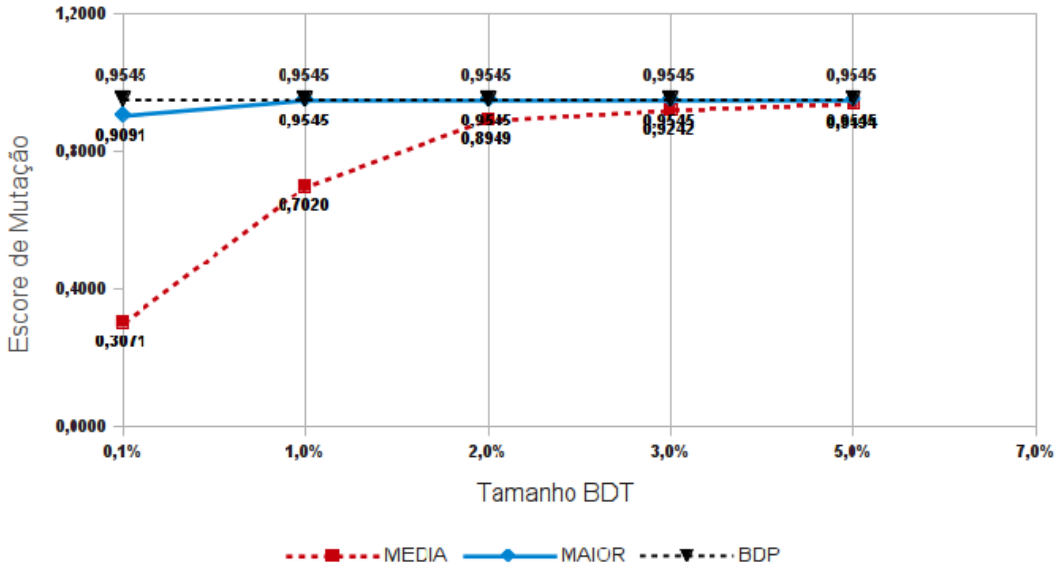


**Figura 4.13:** *Evolução da média do escore de mutação com o Método Aleatório*

Outro critério de parada é o próprio tamanho máximo da BDT que foi definido nos parâmentos, pois conforme visto anteriormente, não se justifica realizar reduções de dados gerando subconjuntos grandes. A intenção é reduzir o máximo possível para obter

os benefícios esperados. Sendo assim, não foram geradas BDTs com mais de 10% da BDP, independente dos resultados alcançados.

## 4.4 Processo de Construção do *Benchmark* e Execução dos Experimentos

Como foi visto nas seções anteriores, o *benchmark* tem componentes com características específicas e bem definidas para alcançar o seu objetivo. Construir todos esses componentes, além dos entregáveis, não é uma tarefa trivial. Portanto, um **processo incremental** foi definido para que, de forma sistemática, os componentes do *benchmark* fossem evoluindo a cada ciclo de execução dos experimentos. Por este motivo, a construção do *benchmark* está totalmente relacionada com a realização dos experimentos, sendo possível expressar tudo em um único processo.

Estão envolvidos neste processo três ambientes distintos de banco de dados. Em um dos ambientes será construído a BDP, em outro serão realizadas as inúmeras gerações de BDTs e no terceiro ambiente serão controlados os experimentos e registrados os resultados obtidos.

Para facilitar a explicação e entendimento, o processo de construção do *benchmark* e execução dos experimentos foi dividido em quatro macro atividades sendo elas:

1. Criar BDP inicial;
2. Definir e registrar instruções SQL e mutantes;
3. Calcular com a BDP o escore de mutação de cada instrução SQL;
4. Cadastrar e executar experimentos;

A Figura 4.14 apresenta o fluxo de execução das macro atividades. A primeira macro atividade do processo consiste na criação da BDP inicial. Neste momento, após definido o modelo de dados, as tabelas são criadas e populadas em um ambiente específico de acordo com a estratégia definida: Cenário 1 - Geração específica dos dados para a BDP do *benchmark*; Cenário 2 - Cópia dos dados de uma BDP de uma aplicação real.

Estão envolvidas na segunda macro atividade a criação das Instruções SQL e a geração de seus mutantes usando a ferramenta *SQLMutantion*. Todas as instruções e mutantes são cadastradas na Base de Dados de Experimentos (BDE). Esta macro atividade é executada no mínimo duas vezes, uma para cada grupo de instruções SQL conforme previsto na seção 4.3.2.

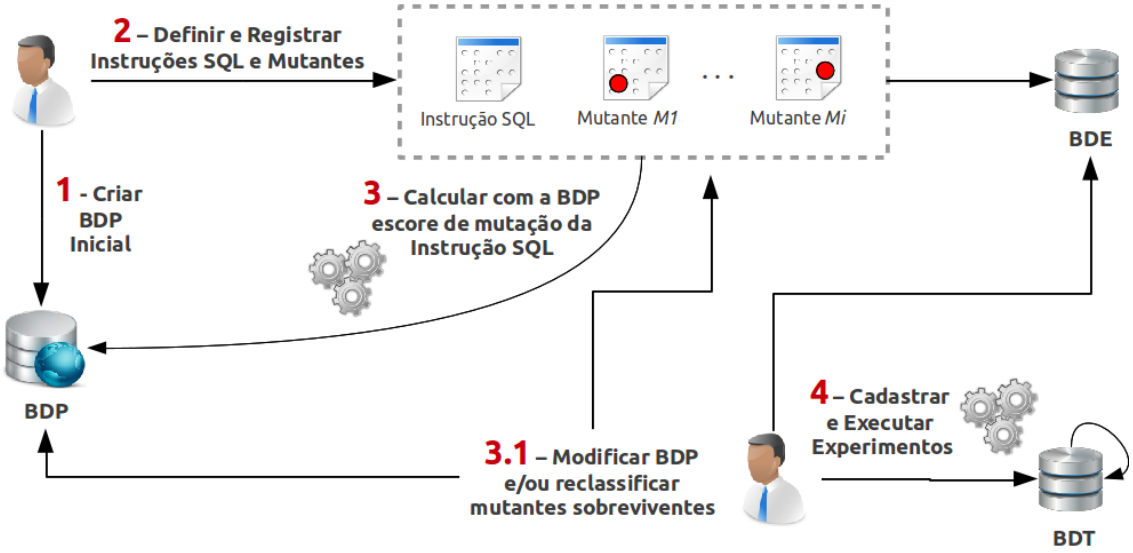


**Figura 4.14:** *Macro processo da construção do Benchmark*

Na terceira macro atividade, para cada instrução SQL, é calculado o escore de mutação utilizando toda a BDP como base de teste. Este valor é registrado na BDE para futuras consultas e comparações. Para atender às características do Cenário 1, foi criada uma sub-atividade específica (3.1) para garantir que o escore de mutação de todas as instruções SQL seja igual a 100%. Portanto, para todo mutante sobrevivente da Macro atividade 3 é feita uma avaliação manual para entender porque ele não foi morto. Se houver condições, a BDP é alterada para conseguir matar este mutante, senão, este mutante é reclassificado como equivalente. O escore de mutação é calculado novamente após as alterações na BDP e/ou reclassificação dos mutantes sobreviventes. A quarta macro atividade é apresentada na seção seguinte.

### 4.4.1 Cadastrar e Executar Experimentos

A quarta e última macro atividade, **4-Cadastrar e Executar Experimentos**, é a mais complexa de todo o processo. Por este motivo ela está melhor detalhada em um diagrama de atividades apresentado na Figura 4.15, onde são destacadas as iterações (ciclos de execução) por instrução SQL, por experimento, por BDT e por mutante.

O primeiro passo é definir e cadastrar na BDE os experimentos para cada instrução SQL. Além da indicação de qual instrução SQL envolvida, cada experimento possui dois parâmetros que indicam a quantidade de BDTs e a quantidade de tuplas, que é um valor calculado de acordo com o percentual de redução da BDP. Serão considerados os parâmetros padrões definidos na Seção 4.3.3.

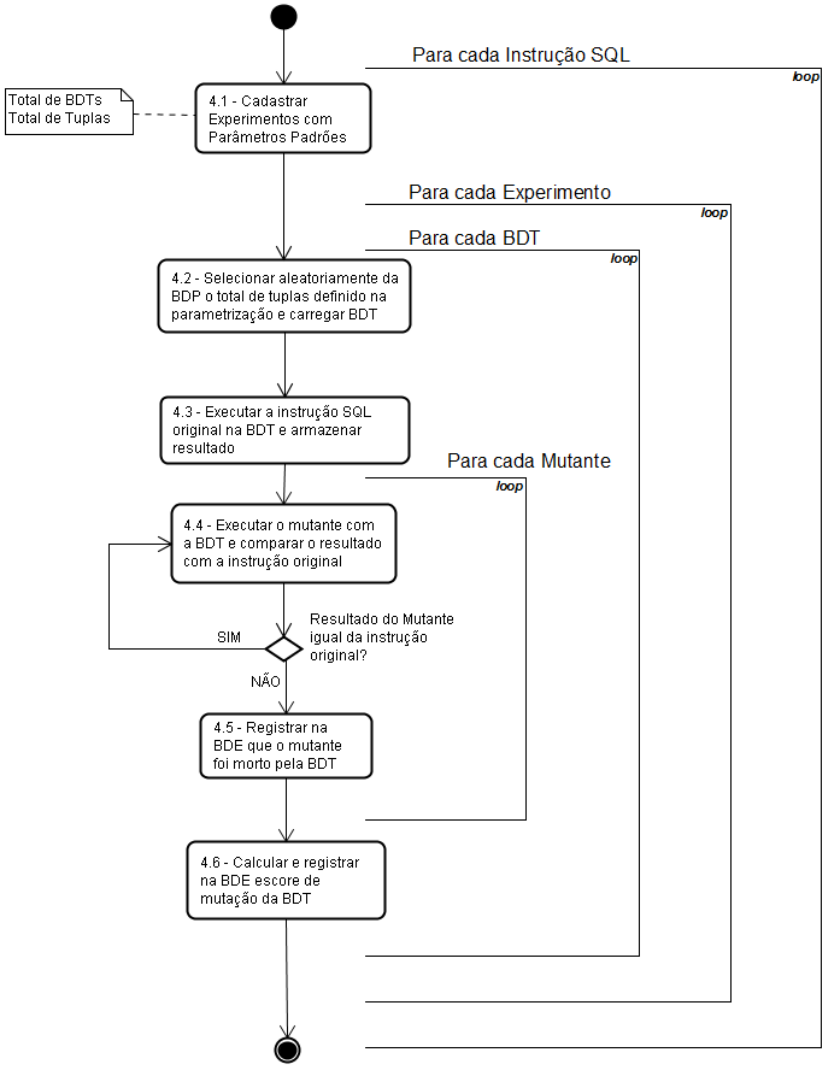


**Figura 4.15:** *Processo da macro atividade de Execução dos Experimentos*

Considerando os critérios de parada, nem todos os experimentos previstos e cadastrados serão executados. Dependendo dos resultados obtidos, em algumas instruções SQL somente os primeiros experimentos já serão suficientes para alcançar um escore de mutação igual ou próximo da BDP. Por outro lado, algumas instruções SQL vão executar todos os experimentos e mesmo assim não serão alcançados escores altos.

Após os experimentos cadastrados, é iniciada a execução de cada um. Para cada experimento são geradas as BDTs previstas, e para cada BDT é calculado o escore de mutação, comparando o resultado do mutante com o resultado da instrução original. O resultado é registrado na BDE, e cada mutante que a BDT matar também é registrado na BDE. Ao final, é possível calcular o escore de mutação de um conjunto consultando os mutantes mortos de todos as BDTs que participam do conjunto.

## 4.5 Ferramenta de apoio

Para facilitar a execução dos experimentos, além da ferramenta *SQLMutation* que foi implementada e disponibilizada por Tuya et al. [48], foi projetada e implementada uma outra ferramenta que automatizou as atividades mais complexas apresentadas na seção anterior. Em complemento à ferramenta, foi projetada também a base de dados de experimentos (BDE) para armazenar todas as informações e resultados, como previsto no processo de execução dos experimentos.

Basicamente o motor da ferramenta tem duas funções principais para executar os experimentos: (i) A ferramenta lê os dados dos experimentos que devem ser executados, e de acordo com os parâmetros ela **seleciona aleatoriamente tuplas da BDP para formar as BDT**; e (ii) para cada BDT **realiza os cálculos de escore de mutação** e registra todos os resultados na BDE. Sendo assim, a ferramenta tem acesso aos três ambientes (BDP, BDT, BDE) utilizados nos experimentos, funcionando como uma mediadora e controladora do estado de cada um. A Figura 4.16 é uma representação visual das interações da ferramenta com os ambientes e com o usuário que realiza/gerencia o processo.

A ferramenta não possui interface para cadastro das instruções SQL, mutantes e experimentos. Tudo isso é realizado diretamente na BDE através de *scripts* SQL. Também não faz parte do escopo desta versão inicial da ferramenta nenhum tipo de funcionalidade para povoar ou manipular a BDP. Estas atividades também são feitas diretamente/manualmente na BDP através de *scripts* SQL.

O modelo de dados definido para a BDE permite que a ferramente leia os parâmetros dos experimentos e registre todas as informações geradas durante os mesmos. O Apêndice F apresenta o modelo físico da BDE. Nas tabelas definidas é possível registrar as seguintes informações:

- Os experimentos realizados e/ou à serem realizados;
- Resultados dos experimentos realizados;
- As instruções SQL;
- Os mutantes de cada instrução e suas classificações (normal ou equivalente);
- Os operadores de mutação e a relação mutante *vs.* operador;
- As tuplas selecionadas aleatoriamente para cada BDT gerada;
- A relação de mutantes mortos por cada versão da BDP;
- A relação de mutantes mortos por cada BDT gerada;
- Versões da BDP juntamente com as modificações realizadas e suas justificativas (no

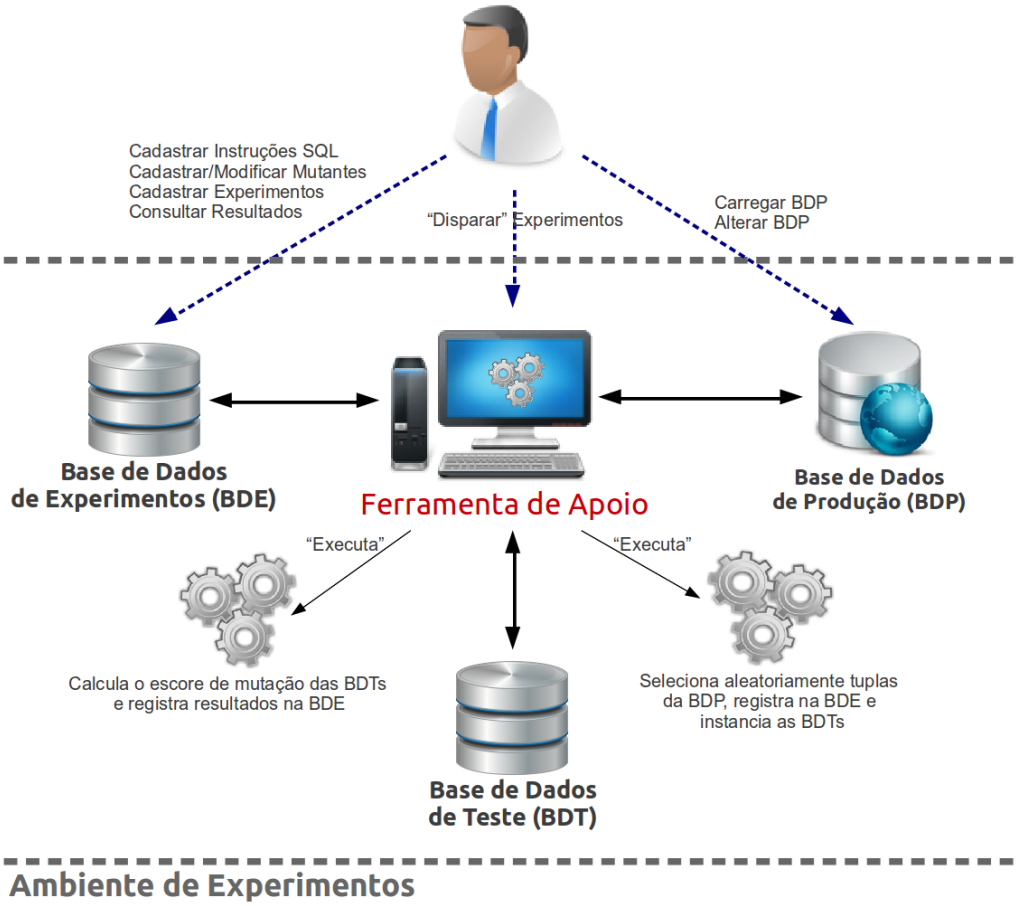


**Figura 4.16:** *Interações da ferramenta de apoio durante execução dos experimentos*

caso do Cenário 1);

Além da disponibilização do *benchmark*, outra contribuição do trabalho é a disponibilização da ferramenta apresentada juntamente com todo o ambiente da BDE e um pequeno guia de utilização de ambos. Isso vai possibilitar que outros pesquisadores realizem experimentos semelhantes com outros parâmetros e até mesmo com outras instruções SQL, possibilitando assim a expansão do *benchmark* conforme a necessidade dos seus utilizadores.

## 4.6 Considerações Finais

Após explicações sobre o contexto do problema e os principais conceitos envolvidos, este capítulo apresentou uma proposta de *benchmark* para servir como um ambiente de referência na avaliação de técnicas de busca no contexto de reduções de bases de dados para Análise de Mutantes SQL.

O *benchmark* é composto por dois cenários distintos, sendo que cada cenário possui os seguintes componentes: (i) uma base de dados de produção; (ii) um conjunto de instruções SQL com seus mutantes; e (iii) resultados da geração de bases de dados de teste usando método aleatório.

Para cada cenário, foram detalhadas as características dos seus componentes, explicadas a origem da estrutura e dos dados da BDP, a definição das instruções SQL e os parâmetros que conduziriam os experimentos. Também é detalhado no capítulo o processo que foi definido e usado para construir, de forma incremental, os componentes do *benchmark* e seus entregáveis. Além disso, a ferramenta implementada para dar suporte nos experimentos foi apresentada e disponibilizada como entregável desta dissertação.

CAPÍTULO 5

# Resultados do Método Aleatório

*"O óbvio é aquilo que nunca é visto até que alguém o manifeste com simplicidade."*
*Khalil Gibran*

Foram realizados, com cada instrução SQL do *benchmark*, diversos experimentos para gerar aleatoriamente, a partir da redução da BDP, bases de teste de diferentes tamanhos. Cada BDT gerada foi avaliada para mensurar a sua capacidade de matar os mutantes da instrução SQL e, consequentemente, foi calculado seu escore de mutação. Como já foi visto nos capítulos anteriores, os resultados da análise de mutantes são importantes referências para avaliar as técnicas de busca.

Além de gerar um ambiente de referência, com os resultados dos experimentos pretende-se também responder algumas importantes questões que podem contribuir em conhecimento na área da Análise de Mutantes SQL, sendo as principais:

**Q1** - Quais propriedades de uma BDT que influenciam no seu escore de mutação?
**Q2** - É possível determinar e ranquear a complexidade[1] das instruções SQL?
**Q3** - Quais fatores influenciam na complexidade de uma instrução SQL?
- Q3.1 - Quantidade de mutantes?
- Q3.2 - Operadores de mutação?
- Q3.3 - Comando e/ou função SQL?
- Q3.4 - Conteúdo da Base de Dados de Produção?

[1] Considera-se que quanto maior a dificuldade para matar os mutantes da instrução SQL, maior é a sua complexidade. Consequentemente também, mais difícil é a instância do problema.

Na tentativa de responder essas questões, foram definidas diferentes situações de experimentos, que procuram avaliar os fatores que podem ou não influenciar na complexidade das instruções SQL e nos escores de mutação. Sendo assim, cada experimento foi realizado com uma parametrização que determina a instrução SQL a ser testada, a quantidade de BDTs que devem ser geradas e o tamanho de cada BDT. Todos estes parâmetros foram explicados e definidos na Seção 4.3.3.

O tamanho da BDT é representado por um valor proporcional da BDP, e os possíveis valores são 0,1%, 1%, 2%, 3%, 5%, 7%, 8%, 9% e 10%. Para cada possível tamanho de BDT são executados três experimentos, sendo o primeiro gerando 5 BDTs, o segundo 10 e o terceiro 30. O objetivo de ter diferentes quantidade de BDTs é avaliar o escore de mutação por conjunto de BDTs. Neste caso o escore é calculado avaliando a quantidade de mutantes da instrução SQL que todas as BDTs do conjunto conseguem matar. Espera-se que o escore de mutação seja melhor com conjuntos maiores.

Porém, independente dos conjuntos, a BDT com maior escore de mutação é identificada dentre todas as BDTs de mesmo tamanho. Também são calculados outros resultados, sendo o um dos mais importantes a média do escore de mutação por tamanho de BDTs. A Figura 5.1 exemplifica esse processo apresentando o esquema padrão de experimentos para gerar BDTs de 0,1% para uma determinada instrução SQL. Este mesmo esquema se repete para todas as combinações "Instrução SQL + Tamanho de BDT."

Durante a execução dos experimentos, são registradas na BDE informações que permitem agrupar e interpretar os resultados em diferentes perspectivas. Neste capítulo, para cada cenário do *benchmark*, procura-se apresentar e analisar os resultados dos experimentos sob a ótica de: (i) Instruções SQL; (ii) Mutantes; e (iii) Operadores de Mutação. Em cada perspectiva, os resultados estão agrupados e consolidados para fornecer informações relevantes, principalmente no que diz respeito à complexida envolvida.

As informações estão expressas em diferentes **tabelas de ranqueamento**. O principal objetivo dessas tabelas é inferir, em cada perspectiva, as situações com maior complexidade dentro do *benchmark*, seja a relação das instruções SQL que geram instâncias mais difíceis ou a relação dos operadores e/ou mutantes mais resistentes. O inverso também é analisado, procurando entender as situações que geram instâncias de problema mais fáceis.

Cada tabela de ranquemento foi construída baseada em alguma característica e/ou resultado que supostamente pode influenciar na complexidade das instâncias do problema. Utilizam-se dois critérios para mensuar os resultados e gerar o ranqueamento. O primeiro e mais importante é o próprio escore de mutação, que já foi explicado

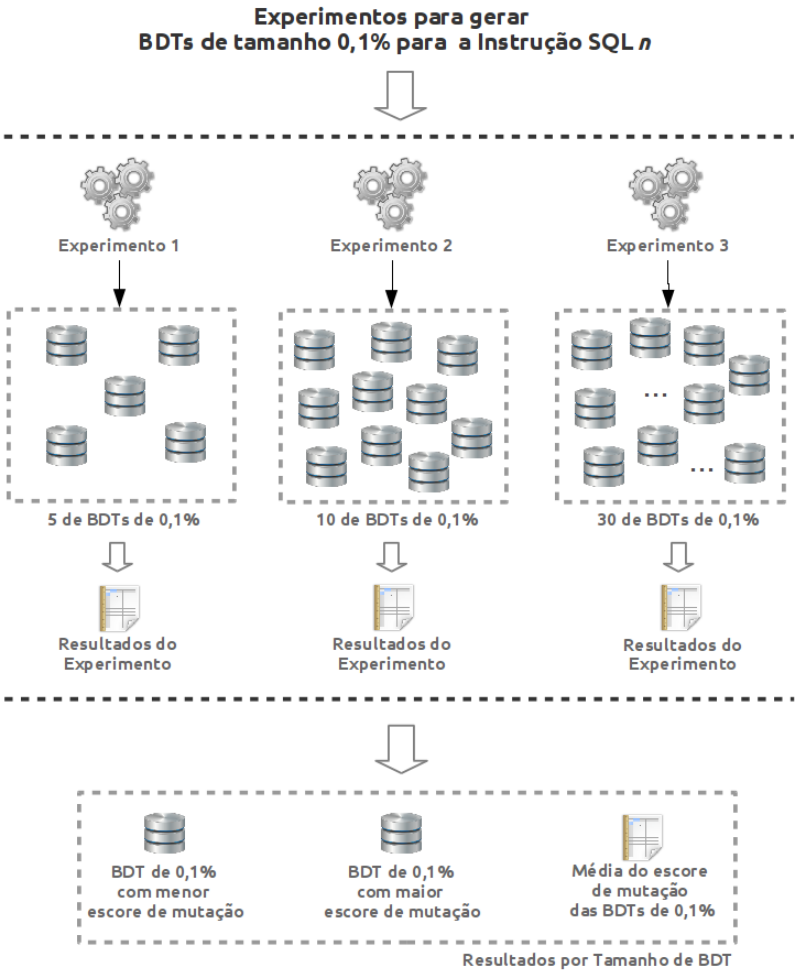


**Figura 5.1:** *Esquema de experimentos para tamanho de BDT igual a 0,1%*

nos capítulos anteriores. A partir do escore de mutação foi proposto um novo valor denominado **Espaço de Melhoria**, que é a diferença entre o escore de mutação das BDTs do MA e o escore de mutação da BDP. O segundo critério adotado foi a **Taxa de Mortalidade**, que é um índice proposto por Tuya et al. [50] que mede o quanto um mutante é resistente, avaliando a quantidade de BDTs que conseguem matá-lo. Ambos critérios serão melhores explicados nas seções a seguir.

Além de fornecer um panorama sobre a complexidade das instâncias de problemas do *benchmark*, a análise dos resultados e das diferentes tabelas de ranqueamento, ajuda responder às questões declaradas no inicio deste capítulo. Nas próximas seções, são apresentados para cada cenário, as tabelas de ranqueamento e as análises realizadas, bem como os resultados consolidados nas diferentes perspectivas.

Todos resultados gerados a partir dos experimentos estão fornecidos integralmente na forma de entregáveis do *benchmark*, conforme previsto e detalhado na Seção

B.0.3 do Apêndice B.

## 5.1 Cenário 1 - EMPRESA

### 5.1.1 Resultados na perspectiva das Instruções SQL

Para o Cenário 1, foram criadas as 20 intruções SQL listadas no Apêndice C. A Tabela 5.1 apresenta para cada instrução as seguintes informações: (i) quantidade de operadores de mutação utilizados; (ii) quantidade de mutantes não equivalentes; e (iii) quantidade de BDTs que foram geradas no MA especificamente para a instrução SQL.

**Tabela 5.1:** *Cenário 1 - Quantidade de operadores, mutantes e BDTs por instrução SQL em relação*

| Instrução | Operadores | Mutantes não Equivalentes | BDTs |
|---|---|---|---|
| 1 | 7 | 73 | 405 |
| 2 | 10 | 66 | 180 |
| 3 | 9 | 62 | 95 |
| 4 | 5 | 7 | 225 |
| 5 | 7 | 33 | 270 |
| 6 | 6 | 20 | 45 |
| 7 | 8 | 70 | 270 |
| 8 | 10 | 56 | 270 |
| 9 | 12 | 98 | 45 |
| 10 | 15 | 85 | 405 |
| 11 | 7 | 73 | 270 |
| 12 | 7 | 32 | 405 |
| 13 | 7 | 29 | 405 |
| 14 | 4 | 18 | 405 |
| 15 | 10 | 78 | 405 |
| 16 | 6 | 45 | 180 |
| 17 | 11 | 58 | 405 |
| 18 | 14 | 94 | 405 |
| 19 | 10 | 56 | 405 |
| 20 | 11 | 43 | 315 |

Foi planejado para cada instrução, um conjunto de experimentos. Ao total, para todas as 20 instruções, foram realizados 387 experimentos, gerando e calculando o escore de mutação de 5.795 BDTs. Tentar analisar o escore de todas as BDTs seria impraticável e pouco produtivo. Por este motivo, o primeiro passo foi sintetizar todo o volume de resultados gerados, e a partir deles extrair as seguintes informações relevantes por experimento:

1. Maior escore de mutação encontrado;
2. Menor escore de mutação encontrado;
3. Escore médio de mutação;
4. Escore do conjunto de BDTs;
5. Desvio padrão dos escores de mutação;

Essas informações, para todos os experimentos realizados, estão apresentadas na Seção D.1 do Apêndice D. Para ilustrar, relacionamos na Tabela 5.2 somente os resultados dos experimentos realizados para a Instrução SQL 1.

**Tabela 5.2:** *Cenário 1 - Resultados de todos os Experimentos da Instrução SQL 1*

| | | Escore de Mutação | | | | |
|---|---|---|---|---|---|---|
| **Qtde BDT** | **Qtde Tuplas** | **Maior** | **Menor** | **Média** | **Conjunto** | **Desvio Padrão** |
| 5 | 0,1% | 0,0000 | 0,0000 | 0,0000 | 0,0000 | 0,0000 |
| 10 | 0,1% | 0,0274 | 0,0274 | 0,0274 | 0,0274 | 0,0000 |
| 30 | 0,1% | 0,1233 | 0,0274 | 0,0754 | 0,1233 | 0,0480 |
| 5 | 1% | 0,0274 | 0,0274 | 0,0274 | 0,0548 | 0,0000 |
| 10 | 1% | 0,1233 | 0,0274 | 0,0462 | 0,1233 | 0,0313 |
| 30 | 1% | 0,1233 | 0,0274 | 0,0397 | 0,1918 | 0,0277 |
| 5 | 2% | 0,1233 | 0,0274 | 0,0630 | 0,1233 | 0,0320 |
| 10 | 2% | 0,8356 | 0,0274 | 0,1400 | 0,8767 | 0,2475 |
| 30 | 2% | 0,1233 | 0,0274 | 0,0564 | 0,2877 | 0,0380 |
| 5 | 3% | 0,1918 | 0,0274 | 0,0986 | 0,2192 | 0,0633 |
| 10 | 3% | 0,0548 | 0,0274 | 0,0466 | 0,0548 | 0,0126 |
| 30 | 3% | 0,8356 | 0,0274 | 0,0904 | 0,9452 | 0,1442 |
| 5 | 5% | 0,1233 | 0,0274 | 0,0630 | 0,1233 | 0,0320 |
| 10 | 5% | 0,8630 | 0,0548 | 0,3164 | 0,9452 | 0,3426 |
| 30 | 5% | 0,9041 | 0,0274 | 0,1630 | 0,9863 | 0,2384 |
| 5 | 7% | 0,8356 | 0,0548 | 0,2219 | 0,8767 | 0,3076 |
| 10 | 7% | 0,8767 | 0,0274 | 0,1644 | 0,9452 | 0,2427 |
| 30 | 7% | 0,9041 | 0,0548 | 0,3018 | 0,9863 | 0,3286 |
| 5 | 8% | 0,8630 | 0,0548 | 0,4110 | 0,9178 | 0,3552 |
| 10 | 8% | 0,8219 | 0,0548 | 0,2438 | 0,9863 | 0,2898 |
| 30 | 8% | 0,9041 | 0,0274 | 0,2247 | 1,0000 | 0,2836 |
| 5 | 9% | 0,1233 | 0,0548 | 0,0822 | 0,1507 | 0,0336 |
| 10 | 9% | 0,8356 | 0,0548 | 0,2452 | 0,9726 | 0,2983 |
| 30 | 9% | 0,8767 | 0,0274 | 0,3306 | 1,0000 | 0,3418 |
| 5 | 10% | 0,8356 | 0,0548 | 0,2521 | 0,9178 | 0,2962 |
| 10 | 10% | 0,9178 | 0,0548 | 0,2575 | 1,0000 | 0,3319 |
| 30 | 10% | 0,9315 | 0,0548 | 0,2667 | 1,0000 | 0,3143 |

Cada linha da Tabela 5.2 é um experimento que foi realizado para a Instrução SQL 1. A última linha por exemplo, relaciona um experimento que gerou aleatoriamente 30 BDTs, sendo cada BDT com tamanho de 10% em relação ao tamanho da BDP. Dentre as BDTs geradas, o melhor desempenho alcançado foi um escore de mutação de 0,9315 e o pior desempenho foi de 0,0548. A média do escore de mutação das 30 BDTs geradas foi de 0,2667 e usando todas as 30 BDTs, o escore de mutação (escore do conjunto) foi de 1. Para finalizar, a última coluna indica o desvio padrão entre todos os escores de mutação encontrados.

Considerando a grande quantidade de BDTs geradas, com os resultados dos experimentos apresentados na seção D.1 do Apêndice D, já é possível afirmar para a Questão **Q1**, mesmo que sendo óbvio, que o escore de mutação de uma BDT está diretamente relacionada com o seu tamanho. Quanto maior a quantidade de tuplas de uma BDT, a tendência é que mais mutantes ela tem a capacidade de matar. Além disso, observando o comportamento dos experimentos, pode-se concluir que a quantidade de BDTs utilizada para avaliar uma instrução SQL também influencia diretamente nos

resultados. Quanto maior o número de BDTs de um conjunto de casos de teste, maior o escore de mutação alcançado pelo conjunto.

Além de uma visão e análise global dos resultados, é necessário também avaliar individualmente o comportamento de cada instrução SQL, verificando principalmente a evolução do escore de mutação em relação à quantidade de tuplas de uma BDT. Outro aspecto importante a ser avaliado é a complexidade de cada instância do problema, que como já foi relatado anteriormente, está diretamente relacionado com as instruções SQL.

Sendo assim, a partir dos resultados de todos os experimentos, os valores foram agrupados e consolidados por instrução SQL. Para ilustrar, são apresentados na Tabela 5.3 os resultados da instrução SQL 1 agrupados por tamanho de BDT.

**Tabela 5.3:** *Cenário 1 - Resultados Agrupados por Tamanho de BDT da Instrução SQL 1*

| | **Escore de Mutação** | | | |
|---|---|---|---|---|
| **Qtde Tuplas** | **Média** | **Maior** | **Menor** | **Desvio Padrão** |
| 0,1% | 0,0594 | 0,1233 | 0,0274 | 0,0452 |
| 1% | 0,0394 | 0,1233 | 0,0274 | 0,0271 |
| 2% | 0,0766 | 0,8356 | 0,0274 | 0,1281 |
| 3% | 0,0816 | 0,8356 | 0,0274 | 0,1212 |
| 5% | 0,1860 | 0,9041 | 0,0274 | 0,2644 |
| 7% | 0,2624 | 0,9041 | 0,0274 | 0,3145 |
| 8% | 0,2496 | 0,9041 | 0,0274 | 0,2994 |
| 9% | 0,2840 | 0,8767 | 0,0274 | 0,3226 |
| 10% | 0,2630 | 0,9315 | 0,0548 | 0,3164 |

A última linha da Tabela 5.3 indica que todas as BDTs de tamanho 10%, que foram geradas pelos experimentos da instrução SQL 1, obtiveram uma média de escore de mutação igual a 0,2630. O melhor desempenho alcançado obeteve escore de 0,9315, e o pior de 0,0548. O desvio padrão do escore de mutação destas BDTs foi de 0,3164. Nos entregáveis do *benchmark* estão disponíveis os resultados de todas as instruções SQL.

Para facilitar a visualização dos resultados e da evolução do escore de mutação em cada instrução SQL, a partir desta perspectiva de agrupamento apresentada no exemplo da Tabela 5.3, foram criados gráficos para cada instrução SQL. Nestes gráficos estão plotados o escore médio e o maior escore por tamanho de BDT. Como referência, também é plotada a linha que representa o escore de mutação da BDP. As Figuras 5.2 até 5.7 apresentam gráficos da evolução dos escores de mutação de algumas instruções SQL do Cenário 1. Os gráficos de todas as instruções estão apresentados na Seção D.2 do Apêndice D.

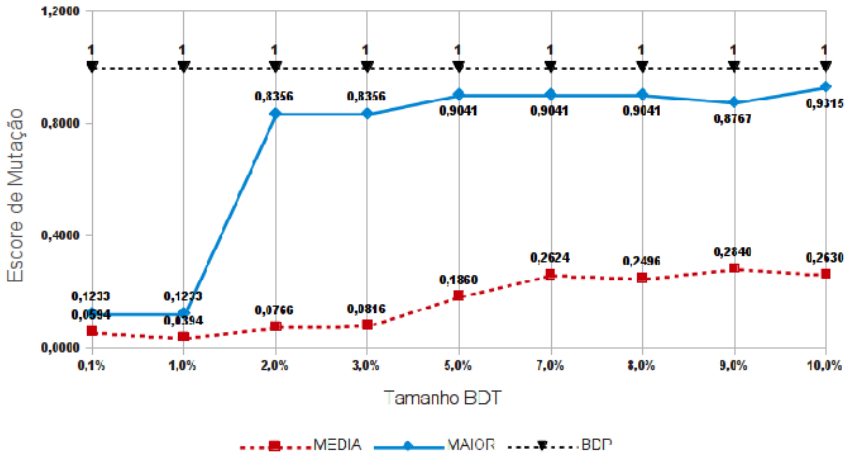


**Figura 5.2:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 1*

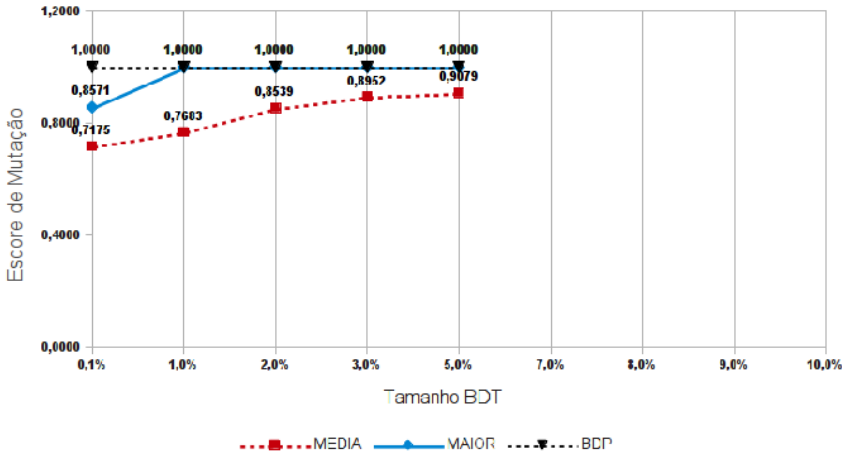


**Figura 5.3:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 4*

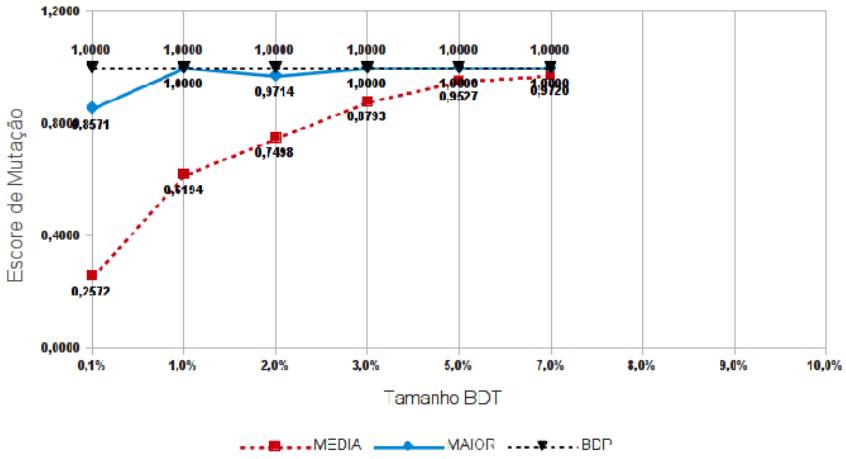


**Figura 5.4:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 7*

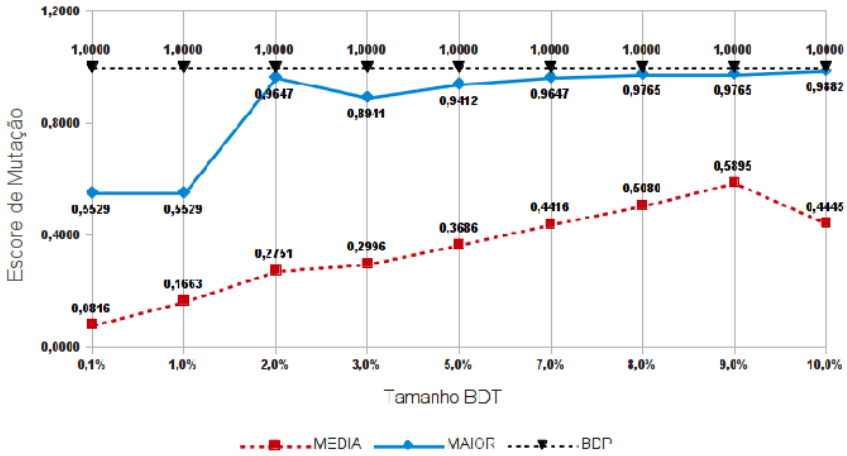


**Figura 5.5:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 10*

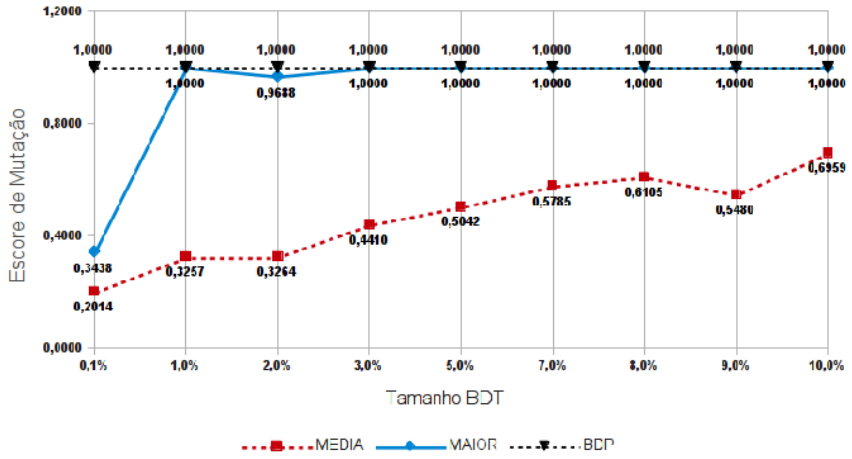


**Figura 5.6:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 12*

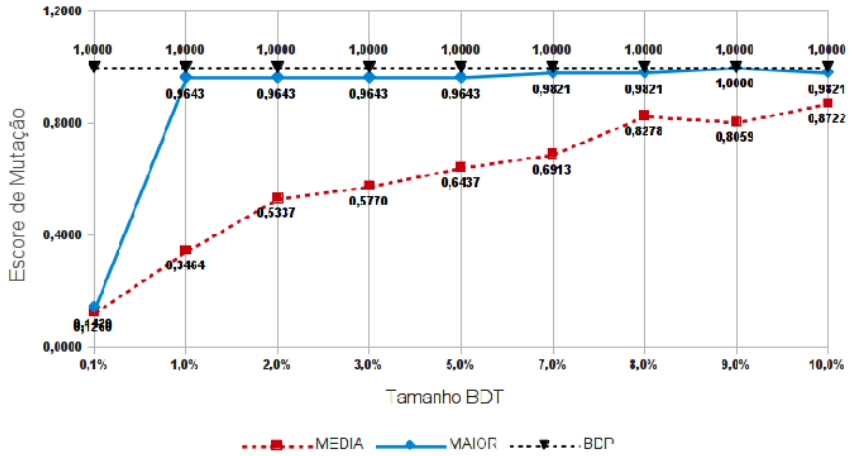


**Figura 5.7:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 19*

### Espaço de Melhoria Médio

Com os resultados consolidados por instrução SQL, foram definidos critérios para inferir sobre a dificuldade das instâncias de problema, de tal modo que fosse possível comparar de forma imparcial (independente do escore da BDP) a complexidade de diferentes instruções SQL. Consequentemente foram definidas regras para ranqueamento das instruções.

A proposta para uma primeira tabela de ranqueamento das instruções SQL foi baseada em dois valores: (i) Escore do MA - Um valor que representa o escore de mutação

médio das BDTs geradas pelo método aleatório; e (ii) Escore BDP - Valor do escore de mutação da BDP para a instrução SQL.

O objetivo é mensurar o quanto o Método Aleatório ficou aquém do escore da BDP, sendo que quanto maior a distância entre o Escore BDP e o Escore do MA, mais complexa é a instância do problema, pois foi difícil encontrar boas soluções com o MA. O valor desta distância é chamado de **Espaço de Melhoria Médio** da instrução SQL, sendo que seu valor máximo é o valor do escore de mutação da BDP.

Exemplificando com a instrução SQL 1, o cálculo do Espaço de Melhoria Médio é feito considerando a média dos valores da segunda coluna da Tabela 5.3. Visualizando no gráfico da Figura 5.8, o Espaço de Melhoria Médio está em destaque, representando a distância média entre a linha dos valores médios de escore das BDTs e a linha que representa o valor de escore da BDP.

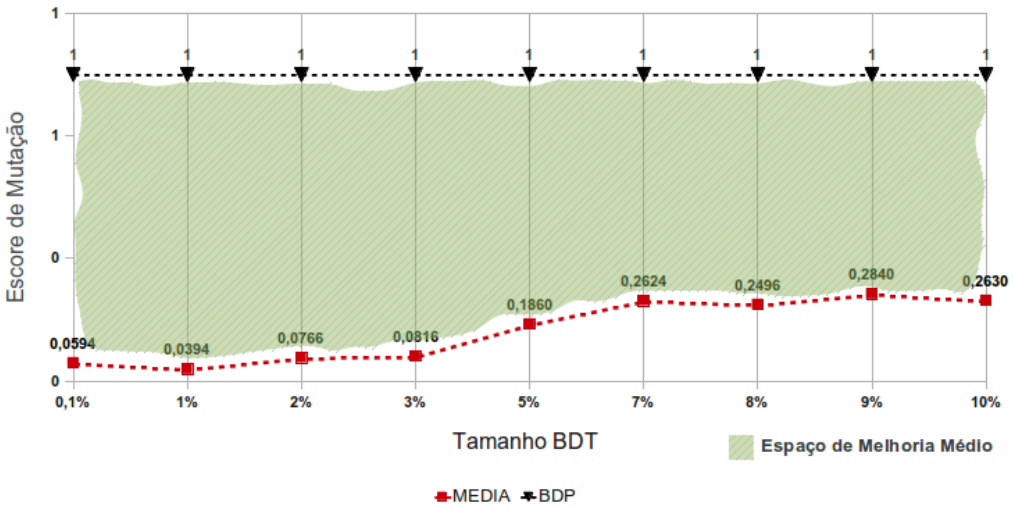


**Figura 5.8:** *Cenário 1 - Espaço de Melhoria Médio da Instrução SQL 1*

A Tabela 5.4 apresenta a primeira proposta de ranqueamento, listando e ordenando as instruções SQL de acordo com o valor do Espaço de Melhoria Médio de cada uma. Quanto maior o Espaço de Melhoria Médio, significa que maior a complexidade da instrução SQL. As mesmas informações são apresentadas em forma de gráfico na Figura 5.9.

Analisando os resultados da Tabela 5.4, foi observado que as instruções com maior complexidade (topo da tabela) e as instruções com menor complexidade (base da tabela) não mantém nenhum padrão que demonstre uma influência na quantidade de mutantes com a complexidade da instância. As três primeiras instruções mais difícies

**Tabela 5.4:** *Cenário 1 - Instrução SQL ranqueadas pelo Espaço de Melhoria Médio*

| POSIÇÃO | Instrução | Qtd. Mutantes | Qtd. Operadores | Média BDT | Escore BDP | Espaço de Melhoria Médio |
|---|---|---|---|---|---|---|
| 1 | 1 | 73 | 7 | 0,16367 | 1 | 0,83632 |
| 2 | 18 | 94 | 14 | 0,23883 | 1 | 0,76116 |
| 3 | 14 | 18 | 4 | 0,35930 | 1 | 0,64069 |
| 4 | 10 | 85 | 15 | 0,36554 | 1 | 0,63445 |
| 5 | 17 | 58 | 11 | 0,41334 | 1 | 0,58665 |
| 6 | 20 | 43 | 11 | 0,44755 | 1 | 0,55244 |
| 7 | 12 | 32 | 7 | 0,47225 | 1 | 0,52774 |
| 8 | 11 | 73 | 7 | 0,49138 | 1 | 0,50861 |
| 9 | 5 | 33 | 7 | 0,608918 | 1 | 0,39108 |
| 10 | 19 | 56 | 10 | 0,61943 | 1 | 0,38056 |
| 11 | 3 | 62 | 9 | 0,67571 | 1 | 0,32428 |
| 12 | 15 | 78 | 10 | 0,68336 | 1 | 0,31663 |
| 13 | 2 | 66 | 10 | 0,70720 | 1 | 0,29279 |
| 14 | 7 | 18 | 4 | 0,74143 | 1 | 0,25856 |
| 15 | 8 | 56 | 10 | 0,80287 | 1 | 0,19712 |
| 16 | 4 | 7 | 5 | 0,83299 | 1 | 0,16700 |
| 17 | 16 | 45 | 6 | 0,84803 | 1 | 0,15196 |
| 18 | 13 | 29 | 7 | 0,88196 | 1 | 0,11803 |
| 19 | 6 | 20 | 6 | 1,00000 | 1 | 0,00000 |
| 20 | 9 | 98 | 12 | 1,00000 | 1 | 0,00000 |

contêm respectivamente 73, 94 e 18 mutantes. E as três mais fáceis contêm 98, 20 e 29. Ou seja, em ambos os casos existem situações com muitos mutantes e situações com poucos mutantes. Considerando a Questão **Q3.1**, pode-se concluir que pelo menos para o contexto do primeiro Cenário do *benchmark*, a quantidade de mutantes não está relacionada diretamente com a complexidade das instâncias.

O mesmo raciocínio para avaliar a quantidade de mutantes pode ser utilizado para analisar a influência da quantidade de operadores de mutação (Questão **Q3.2**) utilizados na geração dos mutantes de uma instrução SQL. E de acordo com o ranqueamento da Tabela 5.4, a princípio também não existem padrões que relacionem a quantidade de operadores de mutação das instruções com a complexidade envolvida. De qualquer forma, esta análise é melhor verificada ao avaliar os resultados na perspectiva de operadores de mutação.

### Espaço de Melhoria Máxima

Além do valor médio dos escores de mutação, também estão disponíveis nos resultados o maior valor de escore de mutação encontrado em um grupo de BDTs do mesmo tamanho. Utilizou-se então esta informação para gerar um outro valor de Espaço de Melhoria baseado na média das BDTs com maior escore de mutação por tamanho de BDT. Consequentemente uma segunda proposta de tabela de ranqueamento das instruções SQL foi construída.

Exemplificando novamente com a Instrução SQL 1, o cálculo do **Espaço de**

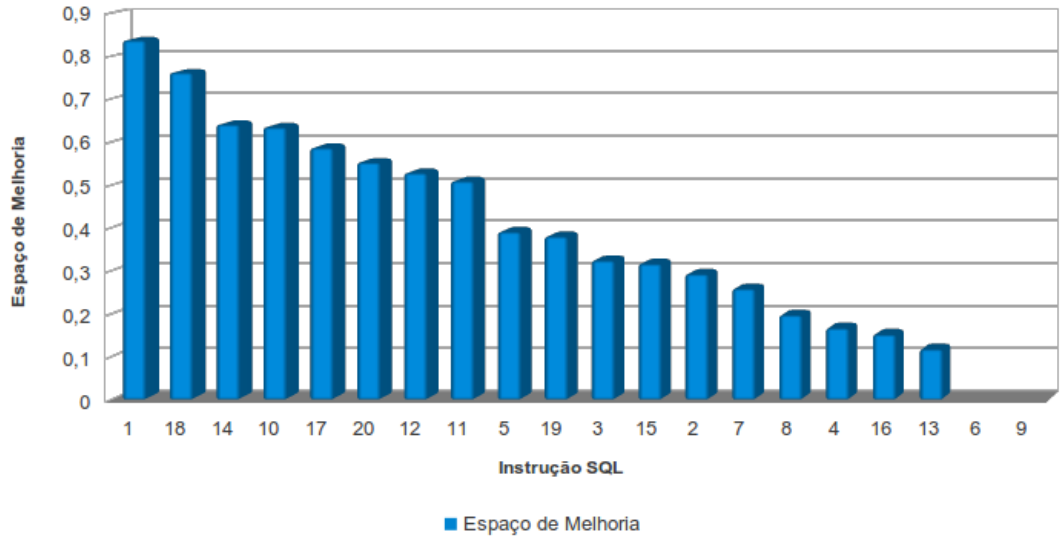


**Figura 5.9:** *Cenário 1 - Espaço de Melhoria Médio das Instruções SQL*

**Melhoria Máxima** é feito considerando a média dos valores da terceira coluna da Tabela 5.3. Visualizando o gráfico na Figura 5.10, o Espaço de Melhoria Máxima está em destaque, representando a distância média entre a linha dos valores máximos de escore das BDTs e a linha que representa o valor de escore da BDP.

A Tabela 5.5 apresenta o novo ranqueamento, listando e ordenando as instruções SQL de acordo com o valor do Espaço de Melhoria Máxima. As mesmas informações são apresentadas em forma de gráfico na Figura 5.11.

**Tabela 5.5:** *Cenário 1 - Instruções SQL ranqueadas pelo Espaço de Melhoria Máxima*

| POSIÇÃO | Instrução | Média Escore BDT Máxima | Escore BDP | Espaço de Melhoria Máxima |
|---|---|---|---|---|
| 1 | 17 | 0,44064 | 1 | 0,55935 |
| 2 | 18 | 0,60755 | 1 | 0,39244 |
| 3 | 14 | 0,65432 | 1 | 0,34567 |
| 4 | 1 | 0,71536 | 1 | 0,28463 |
| 5 | 5 | 0,82828 | 1 | 0,17171 |
| 6 | 20 | 0,86048 | 1 | 0,13951 |
| 7 | 10 | 0,86796 | 1 | 0,13203 |
| 8 | 19 | 0,88293 | 1 | 0,11706 |
| 9 | 15 | 0,90454 | 1 | 0,09545 |
| 10 | 12 | 0,92362 | 1 | 0,07637 |
| 11 | 11 | 0,92465 | 1 | 0,07535 |
| 12 | 13 | 0,96167 | 1 | 0,03832 |
| 13 | 8 | 0,96428 | 1 | 0,03571 |
| 14 | 7 | 0,97141 | 1 | 0,02858 |
| 15 | 4 | 0,97142 | 1 | 0,02858 |
| 16 | 16 | 0,97777 | 1 | 0,02222 |
| 17 | 3 | 0,98923 | 1 | 0,01076 |
| 18 | 2 | 0,99620 | 1 | 0,0038 |
| 19 | 6 | 1,00000 | 1 | 0,00000 |
| 20 | 9 | 1,00000 | 1 | 0,00000 |

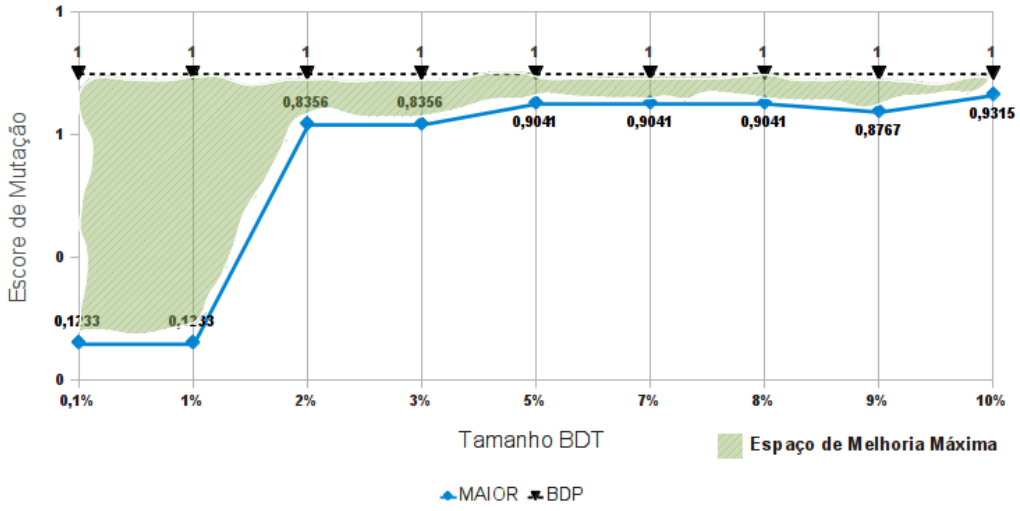


**Figura 5.10:** *Cenário 1 - Espaço de Melhoria Máxima da Instrução SQL 1*

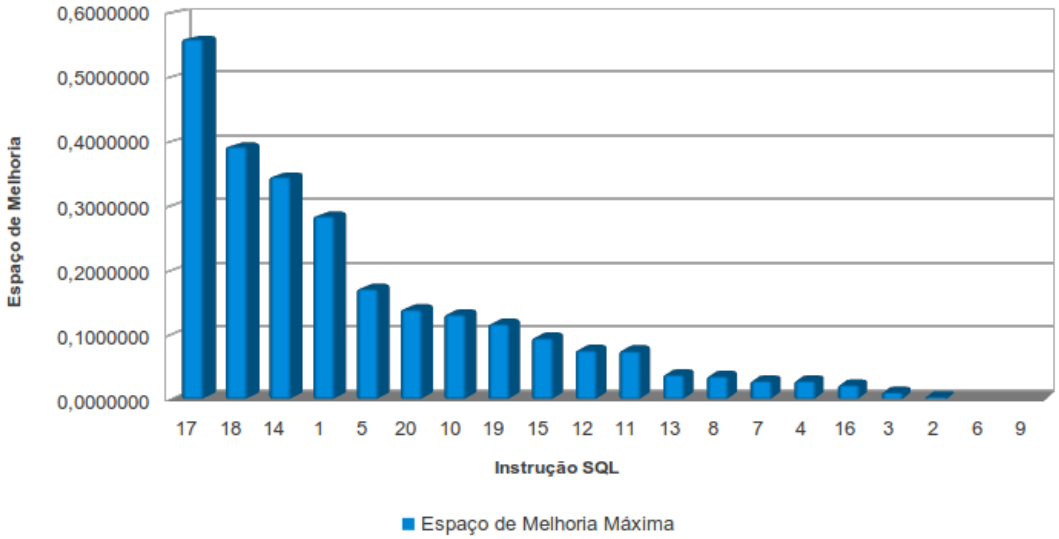


**Figura 5.11:** *Cenário 1 - Espaço de Melhoria Máxima das Instruções SQL*

Comparando o ranqueamento anterior da Tabela 5.4 com o novo ranqueamento do Espaço de Melhoria Máxima da Tabela 5.5, pode-se observar que 6 instruções SQL mantiveram a mesma posição, duas tiveram mudanças de apenas uma posição a menor e as outras doze tiveram suas classificações alteradas em no mínimo duas posições a menor e no máximo seis a maior. Em média, a diferença de posicionamento das instruções SQL foi de 2,3 posições.

Pode-se concluir que, apesar das duas abordagens não serem correlatas, de modo

geral ambas mantém um comportamento muito parecido no ranqueamento das instruções, indicando assim, eficiência na determinação da complexidade das instruções SQL (Questão **Q2**) usando o critério do Espaço de Melhoria. A mudança mais significativa foi a alteração do posicionamento da Instrução 17, que pelo critério anterior foi classificada como a quinta mais difícil e neste novo critério foi classificada como a mais difícil dentre as 20. Entender porque esta instrução foi tão beneficiada nesta abordagem merece melhor investigação em outras perspectivas de resultados.

### Espaço de Melhoria Médio por Tamanho de BDT

Uma outra possibilidade interessante é usar o Espaço de Melhoria para avaliar o comportamento das instruções SQL com diferentes tamanhos de BDT, ou seja, avaliar a complexidade de uma instância de problema nas situações em que foram criadas BDTs de um determinado tamanho.

Neste caso, o Espaço de Melhoria é calculado com a média do escore de mutação de todos os BDTs de mesmo tamanho por instrução SQL. Sendo assim, é possível ranquear as instâncias de problemas nas diferentes situações. Para facilitar o entendimento, fica convencionado que neste contexto uma **situação** representa uma instrução SQL sendo avaliada com BDTs de um determinado tamanho.

A Tabela 5.6 relaciona as 50 situções que apresentaram maior dificuldade com o MA. Teoricamente, a situação mais difícil no *benchmark* é encontrar uma BDT adequada para a instrução SQL 1 com tamanho de 1%.

Analisando os resultados apresentados na Tabela 5.6, concluímos que uma situação difícil do *benchmark* não está relacionada somente com a instrução SQL, mas também com o tamanho da BDT. Pode-se observar que as quatro situações mais difíceis estão relacionadas com a instrução SQL 1, que é a instrução posicionada como a mais difícil pelo ranqueamento do Espaço de Melhoria Médio. Porém, as duas situações seguintes (quinta e sexta) não estão relacionadas com as instruções 18 e 14, que são, respectivamente, a segunda e terceira instrução mais difícil pelo ranqueamento.

Em outro exemplo, consideremos a complexidade da quarta e quinta situações da Tabela 5.6. Pode-se observar que apesar de serem situações com instruções e tamanhos de BDTs diferentes, ambas praticamente apresentaram o mesmo grau de dificuldade.

**Tabela 5.6:** *Cenário 1 - Ranqueamento das 50 situações com maior complexidade no benchmark*

| POSIÇÃO | Instrução | Qtde Tuplas | Espaço de Melhoria Médio |
|---|---|---|---|
| 1 | 1 | 1% | 0,960613 |
| 2 | 1 | 0,1% | 0,940633 |
| 3 | 1 | 2% | 0,923423 |
| 4 | 1 | 3% | 0,918411 |
| 5 | 10 | 0,1% | 0,918400 |
| 6 | 19 | 0,1% | 0,873977 |
| 7 | 18 | 0,1% | 0,867109 |
| 8 | 17 | 0,1% | 0,857467 |
| 9 | 11 | 0,1% | 0,852651 |
| 10 | 5 | 0,1% | 0,848500 |
| 11 | 10 | 1% | 0,833698 |
| 12 | 18 | 1% | 0,829087 |
| 13 | 20 | 0,1% | 0,827378 |
| 14 | 1 | 5% | 0,813998 |
| 15 | 12 | 0,1% | 0,798607 |
| 16 | 15 | 0,1% | 0,788047 |
| 17 | 18 | 2% | 0,780629 |
| 18 | 2 | 0,1% | 0,774091 |
| 19 | 18 | 5% | 0,773764 |
| 20 | 18 | 3% | 0,760056 |
| 21 | 1 | 8% | 0,750376 |
| 22 | 7 | 0,1% | 0,742847 |
| 23 | 1 | 7% | 0,737591 |
| 24 | 1 | 10% | 0,736980 |
| 25 | 18 | 7% | 0,735224 |
| 26 | 3 | 0,1% | 0,731211 |
| 27 | 10 | 2% | 0,724947 |
| 28 | 1 | 9% | 0,715978 |
| 29 | 18 | 8% | 0,711109 |
| 30 | 18 | 9% | 0,703313 |
| 31 | 20 | 1% | 0,701280 |
| 32 | 14 | 0,1% | 0,701264 |
| 33 | 10 | 3% | 0,700378 |
| 34 | 11 | 1% | 0,692242 |
| 35 | 12 | 1% | 0,674280 |
| 36 | 12 | 2% | 0,673578 |
| 37 | 14 | 2% | 0,665464 |
| 38 | 14 | 1% | 0,661758 |
| 39 | 14 | 3% | 0,660522 |
| 40 | 14 | 5% | 0,660522 |
| 41 | 19 | 1% | 0,653580 |
| 42 | 20 | 2% | 0,641849 |
| 43 | 10 | 5% | 0,631364 |
| 44 | 14 | 9% | 0,623482 |
| 45 | 5 | 1% | 0,614151 |
| 46 | 14 | 8% | 0,608662 |
| 47 | 18 | 10% | 0,606389 |
| 48 | 11 | 2% | 0,593313 |
| 49 | 14 | 10% | 0,576564 |
| 50 | 16 | 0,1% | 0,566402 |

### 5.1.2 Resultados na perspectiva dos Mutantes

Durante a realização dos experimentos, para cada BDT avaliada, são registrados na BDE quais mutantes a BDT conseguiu matar. Além de possibilitar o cálculo do escore de mutação, essa informação também permite medir a resistência do mutante.

Adotamos a **taxa de mortalidade** como critério para mensurar o quanto um mutante é resistente. A taxa de mortalidade de um mutante indica o percentual das BDTs

geradas que tem a capacidade de matá-lo [50]. Por exemplo, se para uma instrução SQL, considerando todos os experimentos, foram geradas ao total 300 BDTs, e se para um dos mutantes desta instrução somente 30 das BDTs geradas conseguem matá-lo, pode-se dizer que a taxa de mortalidade deste mutante é de 10%.

Sendo assim, quanto menor a taxa de mortalidade, mais difícil é matar o mutante, consequentemente, mais resistente ele é. Baseado então nesta taxa, foi construída uma tabela de ranqueamento com todos os 1.181 mutantes não equivalentes gerados no cenário. Na Tabela 5.7, para ilustrar, são apresentados somente os 50 mutantes mais resistentes. O ranqueamento completo está disponibilizado nos entregáveis do *benchmark*.

Com a Tabela de Ranqueamento 5.7, é possível fazer algumas análises e inferências sobre quais fatores podem influenciar na taxa de mortalidade dos mutantes. Inicialmente, pode-se observar que os 30 mutantes mais resistentes são da Instrução 17. Isso confirma e justifica porque esta instrução foi classificada como a mais difícil pelo ranqueamento do Espaço de Melhoria Máxima da Tabela 5.5. O alto número de mutantes que nunca foram mortos por uma BDT deixa claro que, mesmo com as melhores BDTs encontradas pelo MA, ainda existe um grande espaço que pode ser explorado até o escore da BDP. A Figura 5.12 apresenta o gráfico de resultados da Instrução 17. Fica fácil observar que a linha do maior BDT se manteve sempre distante da BDP.

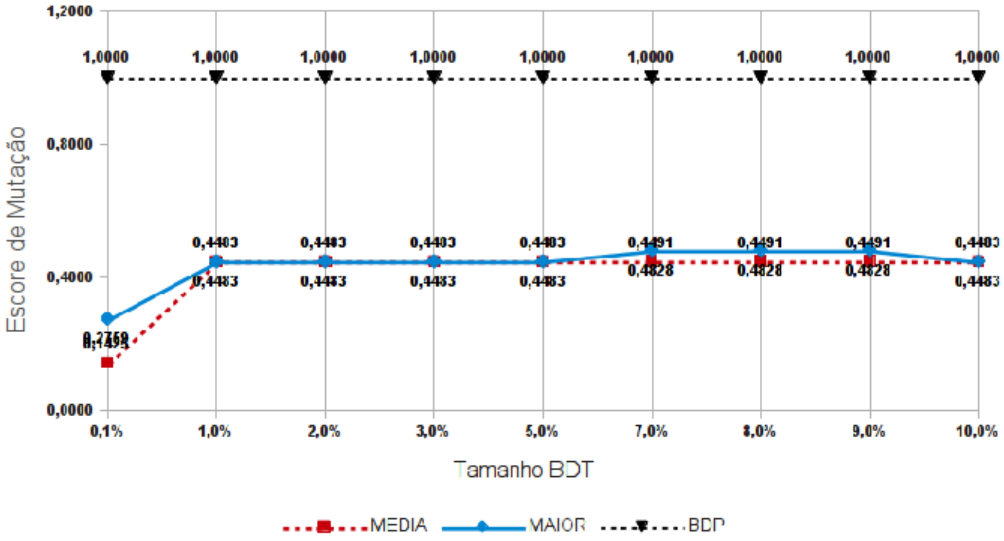


**Figura 5.12:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 17*

No contexto desta pesquisa, é muito importante entender quais fatores fizeram com que estes 30 mutantes nunca fossem mortos e também porque alguns mutantes raramente são mortos, mesmo com a grande quantidade de BDTs geradas pelo MA.

**Tabela 5.7:** *Cenário 1 - Tabela de Ranqueamento por Taxa de Mortalidade com os 50 mutantes mais resistentes*

| POSIÇÃO | Mutantes | Operador | Instrução | Taxa de Mortalidade |
|---|---|---|---|---|
| 1 | 1092 | NLI | 17 | 0 |
| 2 | 1094 | NLO | 17 | 0 |
| 3 | 1096 | IRC | 17 | 0 |
| 4 | 1097 | IRD | 17 | 0 |
| 5 | 1098 | IRD | 17 | 0 |
| 6 | 1099 | IRD | 17 | 0 |
| 7 | 1100 | IRD | 17 | 0 |
| 8 | 1101 | IRD | 17 | 0 |
| 9 | 1102 | IRD | 17 | 0 |
| 10 | 1109 | ROR | 17 | 0 |
| 11 | 1111 | ABS | 17 | 0 |
| 12 | 1112 | UOI | 17 | 0 |
| 13 | 1113 | UOI | 17 | 0 |
| 14 | 1114 | UOI | 17 | 0 |
| 15 | 1115 | IRC | 17 | 0 |
| 16 | 1116 | IRC | 17 | 0 |
| 17 | 1117 | IRC | 17 | 0 |
| 18 | 1120 | LCR | 17 | 0 |
| 19 | 1125 | UOI | 17 | 0 |
| 20 | 1126 | UOI | 17 | 0 |
| 21 | 1129 | IRD | 17 | 0 |
| 22 | 1130 | ROR | 17 | 0 |
| 23 | 1134 | ROR | 17 | 0 |
| 24 | 1136 | ROR | 17 | 0 |
| 25 | 1137 | IRT | 17 | 0 |
| 26 | 1138 | IRT | 17 | 0 |
| 27 | 1141 | IRC | 17 | 0 |
| 28 | 1142 | IRD | 17 | 0 |
| 29 | 1149 | ROR | 17 | 0 |
| 30 | 1150 | IRT | 17 | 0 |
| 31 | 1241 | AGR | 18 | 0 |
| 32 | 1249 | NLI | 18 | 0 |
| 33 | 1274 | ABS | 18 | 0 |
| 34 | 324 | SEL | 5 | 0,3704 |
| 35 | 1143 | ROR | 17 | 0,7407 |
| 36 | 1146 | ROR | 17 | 0,7407 |
| 37 | 1315 | LKE | 18 | 0,7407 |
| 38 | 1281 | NLI | 18 | 0,9877 |
| 39 | 619 | SEL | 10 | 1,2346 |
| 40 | 710 | SEL | 10 | 1,2346 |
| 41 | 388 | AGR | 7 | 1,4815 |
| 42 | 431 | AGR | 7 | 1,4815 |
| 43 | 709 | ABS | 10 | 1,4815 |
| 44 | 800 | ABS | 10 | 1,4815 |
| 45 | 72 | UOI | 20 | 3,1746 |
| 46 | 81 | BTW | 20 | 3,1746 |
| 47 | 1234 | IRC | 18 | 3,4568 |
| 48 | 1235 | AGR | 18 | 3,4568 |
| 49 | 1236 | AGR | 18 | 3,4568 |
| 50 | 1237 | AGR | 18 | 3,4568 |

Considerando apenas os 50 mutantes mais resistentes apresentados na Tabela 5.7, foram avaliadas algumas características em busca de um padrão que justifique a baixa taxa de mortalidade.

Verificando inicialmente os operadores de mutação, é difícil concluir se existe uma relação forte entre eles e a resistência dos mutantes. Isso porque os operadores que geraram os mutantes mais resistentes também geraram mutantes pouco resistentes.

É necessário então avaliar os resultados de todas as instruções SQL na perspectiva de operadores de mutação. Essa avaliação pode revelar padrões a partir dos resultados.

Além dos operadores de mutação, na tentativa novamente de encontrar algum padrão, avaliou-se também a sintaxe dos mutantes mais resistentes e de suas instruções originais. Portanto, pode-se concluir com essa avaliação que um forte fator que impacta na resistência dos mutantes é a **restritividade** da instrução SQL e/ou mutante. Quando a consulta SQL (instrução original ou mutante) procura recuperar um pequeno e muito específico grupo de tuplas, o mutante se torna mais resistente, pois a probabilidade de encontrar a(s) tupla(s) certa(s) dentro da BDP é menor.

Sendo assim, pode-se concluir (Questão **Q3.4**) que um dos fatores mais relevantes na taxa de mortalidade é a quantidade de tuplas na BDP que atende aos critérios de seleção (restritividade) da instrução SQL original e/ou dos mutantes. É necessário então entender quais outros fatores, além da BDP, estão relacionados diretamente com esta restritividade.

Portanto, também são agrupados e ranqueados os mutantes por instrução SQL. Isso permite avaliar se o comportamento de restritividade observado na Tabela 5.7 se repete individualmente em cada instrução. Para ilustrar, apresentamos na Tabela 5.8 todos os mutantes da Instrução 13 (Código 5.1) ordenados pela taxa de mortalidade. Esta mesma informação, para as demais instruções SQL, está disponibilizada nos entregáveis do *benchmark*.

**Código 5.1:** *Instrução SQL 13 do Cenário 1*

```
SELECT SSN FROM EMPLOYEE
WHERE SALARY BETWEEN 1000 AND 1500
```

Na Tabela 5.8, pode-se observar que o mutante mais resistente é o 940, pois ele foi morto por apenas 4,93% das BDTs criadas para a Instrução SQL 13. Já o Mutante 945 é um dos 7 mutantes que foi morto por todas as BDTs criadas, ou seja, são os mutantes extremamente fáceis de serem identificados.

Avaliando melhor o Mutante 940, pode-se reforçar a conclusão anterior sobre a relação da taxa de mortalidade com a sintaxe do mutante no que diz respeito à sua restritividade no retorno de tuplas. Neste exemplo, o Mutante 940 somente é morto se houver na BDT uma tupla com o valor de *SALARY* exatamente igual a 1000. Se na BDP houver muitas ocorrências de tuplas com este valor, é mais fácil encontrar uma BDT que mata o mutante. Porém, se tiver por exemplo só um registro com essa característica, o mutante se torna muito resistente, demonstrando uma relação direta entre o conteúdo da BDP com a taxa de mortalidade dos mutantes.

**Tabela 5.8:** *Cenário 1 - Relação de mutantes da instrução 13 ordenados pela Taxa de Mortalidade*

| Mutante | Comando SQL | Operador | Taxa de Mortalidade |
|---|---|---|---|
| 940 | SELECT SSN FROM EMPLOYEE WHERE ( ( SALARY >= 1000 ) AND ( SALARY < 1500 ) ) | BTW | 4,9383 |
| 939 | SELECT SSN FROM EMPLOYEE WHERE ( ( SALARY > 1000 ) AND ( SALARY <= 1500 ) ) | BTW | 24,6914 |
| 925 | SELECT SSN FROM EMPLOYEE WHERE (employee.salary IS NULL OR SALARY BETWEEN 1000 AND 1500 ) | NLI | 45,4321 |
| 932 | SELECT SSN FROM EMPLOYEE WHERE ((SALARY)-1) BETWEEN 1000 AND 1500 | UOI | 48,1481 |
| 931 | SELECT SSN FROM EMPLOYEE WHERE ((SALARY)+1) BETWEEN 1000 AND 1500 | UOI | 51,358 |
| 942 | SELECT SSN FROM EMPLOYEE WHERE SALARY BETWEEN employee.salary AND 1500 | IRT | 98,2716 |
| 917 | SELECT ABS(SSN) AS SSN FROM EMPLOYEE WHERE SALARY BETWEEN 1000 AND 1500 | ABS | 98,7654 |
| 918 | SELECT (-ABS(SSN)) AS SSN FROM EMPLOYEE WHERE SALARY BETWEEN 1000 AND 1500 | ABS | 98,7654 |
| 919 | SELECT ((SSN)+1) AS SSN FROM EMPLOYEE WHERE SALARY BETWEEN 1000 AND 1500 | UOI | 98,7654 |
| 920 | SELECT ((SSN)-1) AS SSN FROM EMPLOYEE WHERE SALARY BETWEEN 1000 AND 1500 | UOI | 98,7654 |
| 921 | SELECT (-(SSN)) AS SSN FROM EMPLOYEE WHERE SALARY BETWEEN 1000 AND 1500 | UOI | 98,7654 |
| 922 | SELECT employee.salary FROM EMPLOYEE WHERE SALARY BETWEEN 1000 AND 1500 | IRC | 98,7654 |
| 923 | SELECT 1000 FROM EMPLOYEE WHERE SALARY BETWEEN 1000 AND 1500 | IRC | 98,7654 |
| 924 | SELECT 1500 FROM EMPLOYEE WHERE SALARY BETWEEN 1000 AND 1500 | IRC | 98,7654 |
| 927 | SELECT SSN FROM EMPLOYEE WHERE (employee.salary IS NULL) | NLO | 98,7654 |
| 930 | SELECT SSN FROM EMPLOYEE WHERE (-ABS(SALARY)) BETWEEN 1000 AND 1500 | ABS | 98,7654 |
| 933 | SELECT SSN FROM EMPLOYEE WHERE (-(SALARY)) BETWEEN 1000 AND 1500 | UOI | 98,7654 |
| 938 | SELECT SSN FROM EMPLOYEE WHERE SALARY BETWEEN ( 1500 ) AND ( 1000 ) | BTW | 98,7654 |
| 941 | SELECT SSN FROM EMPLOYEE WHERE SALARY BETWEEN employee.ssn AND 1500 | IRT | 98,7654 |
| 943 | SELECT SSN FROM EMPLOYEE WHERE SALARY BETWEEN 1500 AND 1500 | IRT | 98,7654 |
| 946 | SELECT SSN FROM EMPLOYEE WHERE SALARY BETWEEN 1000 AND 1000 | IRT | 98,7654 |
| 934 | SELECT SSN FROM EMPLOYEE WHERE employee.ssn BETWEEN 1000 AND 1500 | IRC | 99,2593 |
| 926 | SELECT SSN FROM EMPLOYEE WHERE (employee.salary IS NULL OR NOT SALARY BETWEEN 1000 AND 1500 ) | NLO | 100 |
| 928 | SELECT SSN FROM EMPLOYEE WHERE (employee.salary IS NOT NULL) | NLO | 100 |
| 935 | SELECT SSN FROM EMPLOYEE WHERE 1000 BETWEEN 1000 AND 1500 | IRC | 100 |
| 936 | SELECT SSN FROM EMPLOYEE WHERE 1500 BETWEEN 1000 AND 1500 | IRC | 100 |
| 937 | SELECT SSN FROM EMPLOYEE WHERE SALARY NOT BETWEEN 1000 AND 1500 | BTW | 100 |
| 944 | SELECT SSN FROM EMPLOYEE WHERE SALARY BETWEEN 1000 AND employee.ssn | IRT | 100 |
| 945 | SELECT SSN FROM EMPLOYEE WHERE SALARY BETWEEN 1000 AND employee.salary | IRT | 100 |

Outra faceta que pode ser inferida sobre a restritividade é o comando ou função SQL sendo utilizado. Para verificar essa possibilidade, foram avaliados os mutantes mais resistentes das instruções SQL mais difícies. As características destes mutantes podem

apresentar padrões em suas sintaxes, que permitam concluir sobre os elementos SQL relacionados com a restritividade.

#### Ranqueamento de Instruções SQL por Taxa de Mortalidade

Avaliando todos os mutantes da Instrução 13, podemos observar que a **média da taxa de mortalidade** dos mutantes é de 88,05%. Este valor, teoricamente, consegue representar uma expectativa de complexidade de uma instrução SQL em função da resistência de seus mutantes.

Esse raciocínio motivou a criação de uma terceira proposta de ranqueamento das instruções SQL. A Tabela 5.9 lista as instruções SQL ranqueadas pela média da taxa de mortalidade de seus mutantes. Este novo critério de ranqueamento ajuda a comprovar, apesar de óbvio, a relação da taxa de mortalidade dos mutantes com o escore de mutação das BDTs. Na Seção 5.2.4 todas as tabelas de ranqueamento propostas serão comparadas.

**Tabela 5.9:** *Cenário 1 - Instruções SQL ranqueadas pela Média da Taxa de Mortalidade dos Mutantes*

| POSIÇÃO | Instrução | Total Mutantes | Média Taxa de Mortalidade |
|---|---|---|---|
| 1 | 1 | 73 | 15,83288 |
| 2 | 18 | 94 | 24,81481 |
| 3 | 10 | 85 | 35,27379 |
| 4 | 14 | 18 | 36,59806 |
| 5 | 17 | 58 | 41,45593 |
| 6 | 20 | 43 | 44,15651 |
| 7 | 12 | 32 | 47,01387 |
| 8 | 11 | 73 | 50,40081 |
| 9 | 5 | 33 | 57,97977 |
| 10 | 19 | 56 | 59,95592 |
| 11 | 3 | 62 | 65,33104 |
| 12 | 2 | 66 | 66,79295 |
| 13 | 15 | 78 | 67,67016 |
| 14 | 7 | 70 | 73,84130 |
| 15 | 8 | 56 | 79,53042 |
| 16 | 4 | 7 | 82,85714 |
| 17 | 16 | 45 | 83,86418 |
| 18 | 13 | 29 | 88,05447 |
| 19 | 6 | 20 | 100,0000 |
| 20 | 9 | 98 | 100,0000 |

### 5.1.3 Resultados na perspectiva dos Operadores de Mutação

Todo mutante cadastrado na BDE tem a informação de qual é o seu operador de mutação. Com essa relação, podemos agrupar os resultados de taxa de mortalidade por operadores de mutação. Essa perspectiva de resultado permite inferir sobre a capacidade dos operadores de gerar mutantes resistentes no contexto do Cenário 1, além de ajudar na identificação da relação entre os operadores de mutação e a restritividade dos mutantes. A Tabela 5.10 apresenta um ranqueamento dos operadores de mutação considerando a

taxa de mortalidade dos seus mutantes. Sendo assim, por este ranqueamento, conclui-se quais são os operadores que geram os mutantes mais resistentes e os menos resistentes do Cenário 1.

**Tabela 5.10:** *Cenário 1 - Operadores de Mutação Ranqueados pela Taxa de Mortalidade*

| POSIÇÃO | Operador | Categoria | Mutantes | Média Taxa de Mortalidade |
|---|---|---|---|---|
| 1 | ORD | SC | 2 | 6,91360 |
| 2 | AOR | OR | 10 | 20,04940 |
| 3 | NLS | NL | 14 | 21,22221 |
| 4 | NLI | NL | 14 | 28,38148 |
| 5 | SEL | SC | 13 | 28,46914 |
| 6 | IRT | IR | 117 | 40,38107 |
| 7 | BTW | OR | 12 | 47,83950 |
| 8 | IRC | IR | 183 | 48,67232 |
| 9 | UOI | OR | 102 | 49,64304 |
| 10 | IRD | IR | 247 | 51,95760 |
| 11 | ABS | OR | 72 | 56,30009 |
| 12 | LKE | OR | 44 | 60,96176 |
| 13 | UNI | SC | 6 | 61,48148 |
| 14 | AGR | SC | 42 | 63,68609 |
| 15 | ROR | OR | 182 | 69,47203 |
| 16 | LCR | OR | 78 | 71,55226 |
| 17 | NLO | NL | 42 | 75,49871 |
| 18 | SUB | SC | 36 | 84,77023 |
| 19 | NLF | NL | 2 | 87,90120 |
| 20 | JOI | SC | 13 | 96,93828 |
| 21 | GRU | SC | 10 | 99,30864 |

Existem dois aspectos importantes a serem considerados sobre os resultados da Tabela 5.10 que limitam a sua validade:

- Os operadores de mutação que geraram poucos mutantes não podem ser conclusivos em relação à taxa de mortalidade. Logo, decidiu-se ignorar do ranqueamento todos os operadores que geraram menos de 10 mutantes, sendo eles: ORD; UNI; NFL;
- Os resultados apresentados se limitam às instâncias de problemas do Cenário 1. Sendo assim, não é possível afirmar que o comportamento dos operadores de mutação em relação à taxa de mortalidade será o mesmo em outro contexto (BDP, Instruções SQL).

A Figura 5.13 apresenta um gráfico com as taxas de mortalidade dos operadores de mutação. Quanto menor a barra vertical, mais resistentes são os mutantes daquele operador.

A Tabela 5.10 apresentou, independente da instrução SQL, a média geral da taxa de mortalidade dos mutantes de cada operador. Porém, é possível avaliar também, essa média por instrução SQL, na perspectiva de: (i) avaliar todos os operadores de uma instrução SQL ou, (ii) avaliar todas as instruções SQL por operador.

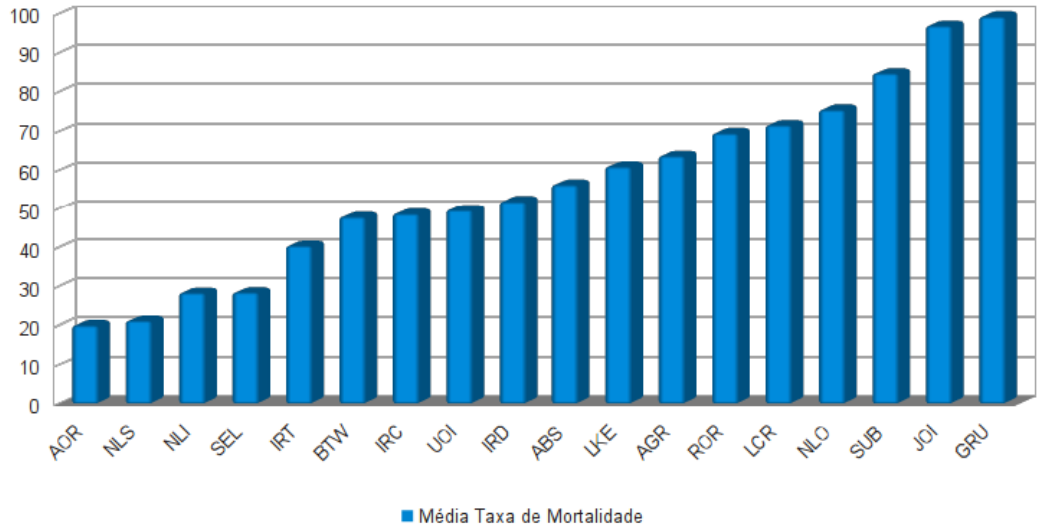


**Figura 5.13:** *Cenário 1 - Taxa de Mortalidade dos Operadores de Mutação*

Para ilustrar, a Tabela 5.11 apresenta, para cada operador utilizado na Instrução SQL 10, a média da taxa de mortalidade dos seus mutantes gerados. O resultados para todas as outras instruções estão disponíveis nos entregáveis do *benchmark*

**Tabela 5.11:** *Cenário 1 - Taxa de Mortalidade média por operador usado na instrução SQL 10*

| Instrução | Operador | Qtde Mutantes | Taxa Média de Mortalidade |
|---|---|---|---|
| 10 | SEL | 2 | 1,2346 |
| 10 | ORD | 2 | 6,9136 |
| 10 | IRT | 30 | 19,6625 |
| 10 | AOR | 10 | 20,0494 |
| 10 | UOI | 24 | 21,8415 |
| 10 | NLS | 4 | 25,6790 |
| 10 | ABS | 10 | 27,4074 |
| 10 | IRC | 24 | 32,5926 |
| 10 | IRD | 20 | 33,5555 |
| 10 | ROR | 14 | 60,3880 |
| 10 | LCR | 20 | 61,9259 |
| 10 | NLO | 6 | 63,6214 |
| 10 | NLF | 2 | 87,9012 |
| 10 | SUB | 2 | 97,0370 |

Na Tabela 5.11, podemos observar que o operador SEL gerou dois mutantes da Instrução SQL 10 que em média foram mortos por somente 1,2346% das BDTs. E no outro extremo, o operador SUB gerou dois mutantes que em média foi morto por 97,037% das BDTs.

Uma análise interessante a ser feita é avaliar o comportamento de um operador em todas as instruções SQL, e observar se é mantido um padrão. Essa análise pode ajudar a confirmar o grau de relação entre o operador de mutação e a complexidade das instâncias de problemas.

Para ilustrar, a Tabela 5.12 apresenta a média da taxa de mutação dos mutantes do operador SEL nas instruções SQL em que ele foi utilizado. O resultado de todos os operadores de mutação utilizados no Cenário 1 estão disponíveis nos entregáveis do *benchmark*.

**Tabela 5.12:** *Cenário 1 - Taxa de Mutação média por instrução SQL do operador de mutação SEL*

| Operador | Instrução | Qtde Mutantes | Taxa Média de Mortalidade |
|---|---|---|---|
| SEL | 4 | 1 | 56,4444 |
| SEL | 5 | 1 | 0,3704 |
| SEL | 8 | 1 | 14,4444 |
| SEL | 10 | 2 | 1,2346 |
| SEL | 16 | 2 | 62,7778 |

O exemplo da Tabela 5.12 mostra que o mutante gerado com o operador SEL para a Instrução SQL 5 foi morto por apenas 0,37% das BDTs. E em constraste, no caso da instrução SQL 16, os dois mutantes gerados foram mortos por 62,77% das BDTs. Isso pode evidenciar que, apesar de existir, a influência do operador de mutação na resistência do mutante pode não ser tão forte.

### 5.1.4 Ranqueamento Final das Instruções SQL

A avaliação da complexidade dos mutantes e dos operadores de mutação foi baseada somente no critério da Taxa de Mortalidade. Sendo assim, somente uma tabela de ranqueamento foi proposta para cada caso.

Já para as instruções SQL foram propostas três abordagens de ranqueamento. A primeira é baseada no Espaço de Melhoria calculado com o valor médio dos escores de mutação. A segunda teve o mesmo raciocínio da anterior, porém considerando a média do escore de mutação das BDTs com melhor desempenho. E por último, a terceira tabela de ranqueamento foi baseada na Taxa de Mortalidade dos seus mutantes.

O mais intuitivo é comparar como cada instrução SQL foi ranqueada nas três diferentes propostas, confrontando assim os posicionamentos apresentados. Isso pode reforçar os resultados e trazer mais conclusividade em relação a quais são as instruções que geram instâncias de problemas mais difíceis no *benchmark*. A Tabela 5.27 relaciona para cada instrução SQL qual foi a sua posição em cada abordagem de ranqueamento.

Analisando a Tabela 5.13 pode-se observar que de forma geral, todas as instruções mantiveram um padrão muito próximo de ranqueamento nas três abordagens adotadas. Isso evidencia credibilidade nos critérios utilizados para os ranqueamentos. No caso das Abordagens 1 e 3, houve uma grande similaridade nos resultados. Esse fato reforça a

**Tabela 5.13:** *Cenário 1 - Ranqueamento das Instruções SQL para cada Abordagem de Ranqueamento*

| | Posição por Abordagem de Ranqueamento | | |
|---|---|---|---|
| **Instrução** | **(1)Espaço de Melhoria Médio** | **(2)Espaço de Melhoria Máximo** | **(3)Taxa de Mortalidade** |
| 1 | 1 | 4 | 1 |
| 2 | 13 | 18 | 12 |
| 3 | 11 | 17 | 11 |
| 4 | 16 | 15 | 16 |
| 5 | 9 | 5 | 9 |
| 6 | 19 | 19 | 19 |
| 7 | 14 | 14 | 14 |
| 8 | 15 | 13 | 15 |
| 9 | 20 | 20 | 20 |
| 10 | 4 | 7 | 3 |
| 11 | 8 | 11 | 8 |
| 12 | 7 | 10 | 7 |
| 13 | 18 | 12 | 18 |
| 14 | 3 | 3 | 4 |
| 15 | 12 | 9 | 13 |
| 16 | 17 | 16 | 17 |
| 17 | 5 | 1 | 5 |
| 18 | 2 | 2 | 2 |
| 19 | 10 | 8 | 10 |
| 20 | 6 | 6 | 6 |

forte relação entre a Taxa de Mortalidade dos mutantes e o Espaço de Melhoria Médio, que é baseada no Escore de Mutação das BDTs.

A Abordagem 2 obteve algumas diferenças em relação as demais, pois levou-se em conta no ranqueamento somente os maiores escores de mutação, que são aqueles valores mais "raros" de serem gerados com o MA.

Considerando então a maior precisão das Abordagens 1 e 3, uma tabela de ranqueamento final foi proposta. A soma da posição dessas duas abordagens vai definir a posição final na nova tabela, sendo que quanto menor o valor, mais complexa e difícil a instrução. Como critério de desempate, foi utilizado o ranqueamento da Abordagem 2.

A Tabela 5.14 apresenta novamente todas as instruções e suas posições nos ranqueamentos, porém, agora acrescida a coluna com a posição do ranqueamento final definido. A Tabela 5.15 apresenta o ranqueamento final das instruções SQL do Cenário 1.

## 5.2 Cenário 2 - UFG

Na seção anterior, as tabelas de resultado e ranqueamento foram explicadas e apresentadas. Nesta seção, os resultados dos experimentos realizados no segundo Cenário do *benchmark*, são apresentados de forma análoga ao primeiro cenário, porém com maior simplicidade, sem detalhar novamente as regras de consolidação dos resultados e tabelas de ranqueamento.

**Tabela 5.14:** *Cenário 1 - Ranqueamento das Instruções SQL para cada Abordagem de Ranqueamento acrescido do ranqueamento final*

| | Posição por Abordagem de Ranqueamento | | | |
|---|---|---|---|---|
| **Instrução** | **(1)Espaço de Melhoria Médio** | **(2)Espaço de Melhoria Máximo** | **(3)Taxa de Mortalidade** | **(4)Final** |
| 1 | 1 | 4 | 1 | 1 |
| 2 | 13 | 18 | 12 | 13 |
| 3 | 11 | 17 | 11 | 11 |
| 4 | 16 | 15 | 16 | 16 |
| 5 | 9 | 5 | 9 | 9 |
| 6 | 19 | 19 | 19 | 19 |
| 7 | 14 | 14 | 14 | 14 |
| 8 | 15 | 13 | 15 | 15 |
| 9 | 20 | 20 | 20 | 20 |
| 10 | 4 | 7 | 3 | 4 |
| 11 | 8 | 11 | 8 | 8 |
| 12 | 7 | 10 | 7 | 7 |
| 13 | 18 | 12 | 18 | 18 |
| 14 | 3 | 3 | 4 | 3 |
| 15 | 12 | 9 | 13 | 12 |
| 16 | 17 | 16 | 17 | 17 |
| 17 | 5 | 1 | 5 | 5 |
| 18 | 2 | 2 | 2 | 2 |
| 19 | 10 | 8 | 10 | 10 |
| 20 | 6 | 6 | 6 | 6 |

**Tabela 5.15:** *Cenário 1 - Ranqueamento Final de todas as Instruções SQL*

| **POSIÇÃO** | **Instrução SQL** | **Operadores** | **Mutantes** | **Espaço de Melhoria Médio** |
|---|---|---|---|---|
| 1 | 1 | 7 | 73 | 0,83632 |
| 2 | 18 | 14 | 94 | 0,76116 |
| 3 | 14 | 4 | 18 | 0,64069 |
| 4 | 10 | 15 | 85 | 0,63445 |
| 5 | 17 | 11 | 58 | 0,58665 |
| 6 | 20 | 11 | 43 | 0,55244 |
| 7 | 12 | 7 | 32 | 0,52774 |
| 8 | 11 | 7 | 73 | 0,50861 |
| 9 | 5 | 7 | 33 | 0,39108 |
| 10 | 19 | 10 | 56 | 0,38056 |
| 11 | 3 | 9 | 62 | 0,32428 |
| 12 | 15 | 10 | 78 | 0,31663 |
| 13 | 2 | 10 | 66 | 0,29279 |
| 14 | 7 | 8 | 70 | 0,25856 |
| 15 | 8 | 10 | 56 | 0,19712 |
| 16 | 4 | 5 | 7 | 0,16700 |
| 17 | 16 | 6 | 45 | 0,15196 |
| 18 | 13 | 7 | 29 | 0,11803 |
| 19 | 6 | 6 | 20 | 0 |
| 20 | 9 | 12 | 98 | 0 |

### 5.2.1 Resultados na perspectiva das Instruções SQL

Para o Cenário 2, foram criadas as 15 instruções SQL listadas na Seção E.1 Apêndice C. A Tabela 5.16 apresenta para cada instrução as seguintes informações: (i) quantidade de operadores de mutação utilizados; (ii) quantidade de mutantes não equivalentes; e (iii) quantidade de BDTs que foram geradas no MA especificamente para a instrução SQL.

Ao total, para as 15 instruções do Cenário 2, foram realizados 238 experimentos,

**Tabela 5.16:** *Cenário 2 - Quantidade de operadores, mutantes e BDTs por instrução SQL*

| Instrução | Operadores | Mutantes não Equivalentes | BDTs |
|---|---|---|---|
| 1 | 7 | 25 | 270 |
| 2 | 8 | 101 | 270 |
| 3 | 4 | 12 | 90 |
| 4 | 7 | 32 | 270 |
| 5 | 2 | 4 | 90 |
| 6 | 8 | 67 | 270 |
| 7 | 8 | 50 | 270 |
| 8 | 9 | 98 | 270 |
| 9 | 7 | 67 | 270 |
| 10 | 10 | 129 | 270 |
| 11 | 9 | 95 | 270 |
| 12 | 11 | 96 | 270 |
| 13 | 10 | 77 | 180 |
| 14 | 6 | 22 | 225 |
| 15 | 9 | 134 | 270 |

gerando e calculando o escore de mutação de 3.555 BDTs. As informações de todos os experimentos estão apresentadas na Seção E.1 do Apêndice E. Para ilustrar, relacionamos na Tabela 5.17 somente os resultados dos experimentos realizados para a Instrução SQL 15.

**Tabela 5.17:** *Cenário 2 - Resultados de todos os Experimentos da Instrução SQL 15*

| | | Escore de Mutação | | | | |
|---|---|---|---|---|---|---|
| Qtde BDT | Qtde Tuplas | Maior | Menor | Média | Conjunto | Desvio Padrão |
| 5 | 0,1% | 0,0746 | 0,0672 | 0,0702 | 0,0821 | 0,0036 |
| 10 | 0,1% | 0,1418 | 0,0672 | 0,0762 | 0,1567 | 0,0223 |
| 30 | 0,1% | 0,5970 | 0,0672 | 0,1279 | 0,6045 | 0,1463 |
| 5 | 1% | 0,6343 | 0,5448 | 0,5731 | 0,6343 | 0,0367 |
| 10 | 1% | 0,6269 | 0,0896 | 0,3463 | 0,6418 | 0,2279 |
| 30 | 1% | 0,6119 | 0,0746 | 0,3836 | 0,6493 | 0,2163 |
| 5 | 2% | 0,6343 | 0,0970 | 0,5089 | 0,6343 | 0,2089 |
| 10 | 2% | 0,7910 | 0,1642 | 0,4933 | 0,8134 | 0,2179 |
| 30 | 2% | 0,7463 | 0,0970 | 0,5259 | 0,7687 | 0,1755 |
| 5 | 3% | 0,7687 | 0,1716 | 0,5403 | 0,7687 | 0,1994 |
| 10 | 3% | 0,6343 | 0,1716 | 0,4731 | 0,6418 | 0,1868 |
| 30 | 3% | 0,7463 | 0,1716 | 0,5736 | 0,7687 | 0,1289 |
| 5 | 5% | 0,7687 | 0,6045 | 0,6507 | 0,7687 | 0,0602 |
| 10 | 5% | 0,7687 | 0,6045 | 0,6530 | 0,7687 | 0,0591 |
| 30 | 5% | 0,8209 | 0,5821 | 0,6435 | 0,8582 | 0,0533 |
| 5 | 7% | 0,8134 | 0,6119 | 0,6940 | 0,8134 | 0,0811 |
| 10 | 7% | 0,7687 | 0,6045 | 0,6478 | 0,7687 | 0,0418 |
| 30 | 7% | 0,7687 | 0,2239 | 0,6298 | 0,8657 | 0,0900 |

Semelhante ao primeiro cenário, com os resultados dos experimentos do Cenário 2, pode-se afirmar para a Questão **Q1** que existe uma relação entre o tamanho da BDT com o escore de mutação. Também pode-se concluir que a quantidade de BDTs utilizada para avaliar uma instrução SQL influencia diretamente nos resultados. Quanto maior o número de BDTs de um conjunto de casos de teste, maior o escore de mutação.

A partir dos resultados de todos os experimentos, os valores foram agrupados e consolidados por instrução SQL. Para ilustrar, é apresentado na Tabela 5.18 os resultados da Instrução SQL 15 agrupados por tamanho de BDT. As Figuras 5.14 até 5.18 apresentam

gráficos de escores de mutação de algumas instruções do Cenário 2. Os gráficos de todas as instruções estão apresentados na Seção E.2 do Apêndice E.

**Tabela 5.18:** *Cenário 2 - Resultados Agrupados por Tamanho de BDT da Instrução 15*

| | Escore de Mutação | | | |
|---|---|---|---|---|
| **Qtde Tuplas** | **Média** | **Maior** | **Menor** | **Desvio Padrão** |
| 0,1% | 0,1100 | 0,5970 | 0,0672 | 0,1226 |
| 1% | 0,3963 | 0,6343 | 0,0746 | 0,2168 |
| 2% | 0,5167 | 0,7910 | 0,0970 | 0,1900 |
| 3% | 0,5476 | 0,7687 | 0,1716 | 0,1579 |
| 5% | 0,6464 | 0,8209 | 0,5821 | 0,0556 |
| 7% | 0,6410 | 0,8134 | 0,2239 | 0,0832 |

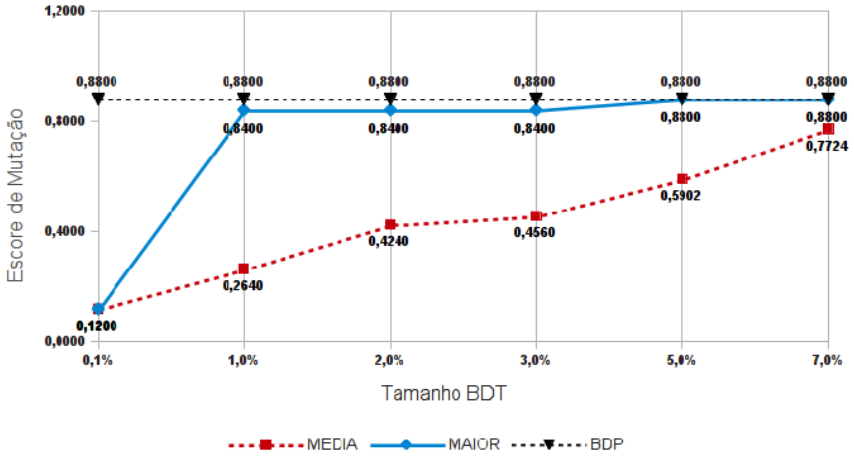


**Figura 5.14:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 1*

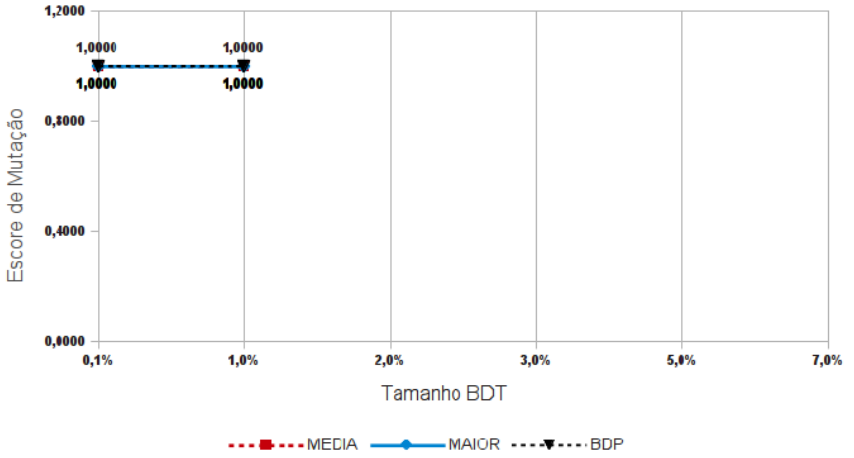


**Figura 5.15:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 5*

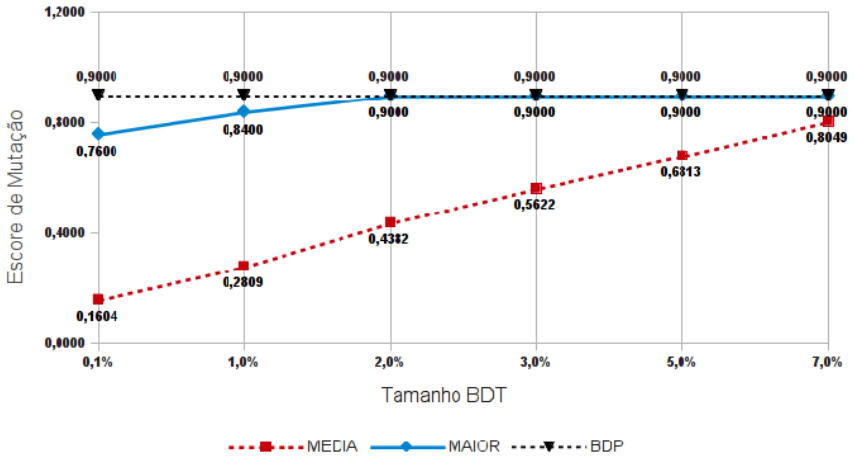


**Figura 5.16:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 7*

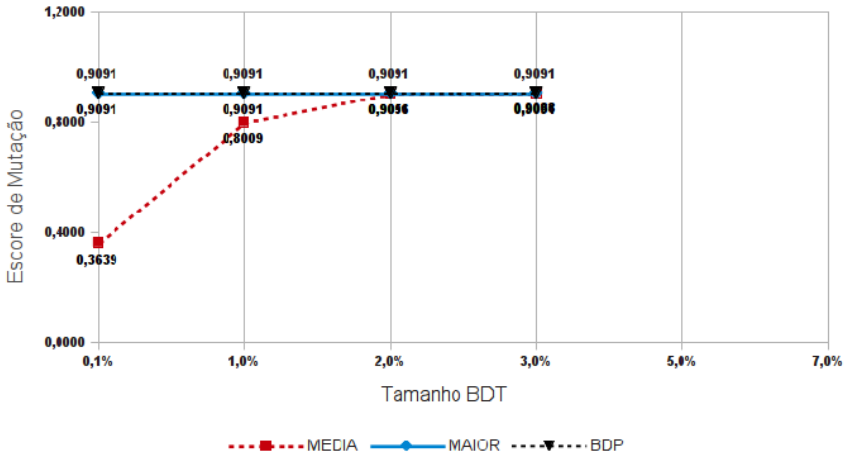


**Figura 5.17:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 13*

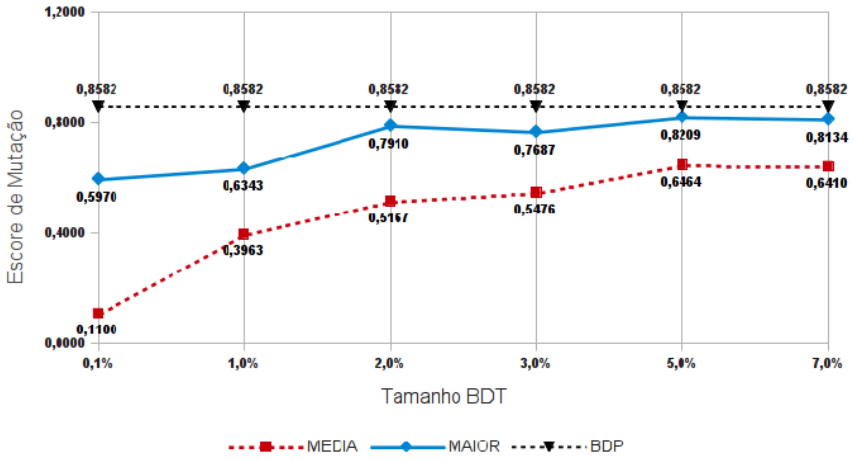


**Figura 5.18:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 15*

### Espaço de Melhoria Médio

A Tabela 5.19 apresenta a primeira proposta de ranqueamento, listando e ordenando as instruções SQL de acordo com o valor do Espaço de Melhoria Médio. As mesmas informações são apresentadas em forma de gráfico na Figura 5.19.

**Tabela 5.19:** *Cenário 2 - Instrução SQL ranqueadas pelo Espaço de Melhoria Médio*

| POSIÇÃO | Instrução | Média BDT | Escore BDP | Espaço de Melhoria Médio |
|---|---|---|---|---|
| 1 | 9 | 0,30498 | 0,94030 | 0,63532 |
| 2 | 4 | 0,28441 | 0,90625 | 0,62184 |
| 3 | 2 | 0,33930 | 0,92079 | 0,58150 |
| 4 | 6 | 0,37675 | 0,92537 | 0,54862 |
| 5 | 10 | 0,39331 | 0,89922 | 0,50592 |
| 6 | 8 | 0,41366 | 0,90816 | 0,49450 |
| 7 | 12 | 0,48975 | 0,93750 | 0,44775 |
| 8 | 1 | 0,43422 | 0,88000 | 0,44578 |
| 9 | 7 | 0,49122 | 0,90000 | 0,40878 |
| 10 | 15 | 0,47840 | 0,85821 | 0,37981 |
| 11 | 11 | 0,48282 | 0,72632 | 0,24349 |
| 12 | 14 | 0,75180 | 0,95455 | 0,20275 |
| 13 | 13 | 0,75609 | 0,90909 | 0,15300 |
| 14 | 3 | 0,91670 | 0,91670 | 0,00000 |
| 15 | 5 | 1,00000 | 1,00000 | 0,00000 |

Na Tabela 5.20 são demonstradas as quantidades de operadores e mutantes para cada instrução ranqueada pelo Espaço de Melhoria Médio. Avaliando a Questão **Q3.1**, a princípio também não existe uma padrão bem definido que possa relacionar a quantidade de mutantes e/ou operadores (Questão **Q3.2**) com a complexidade da instrução SQL. Porém, no Cenário 2, as duas instruções menos complexas neste ranqueamento são as que possuem menor quantidade de operadores e mutantes. Em contrapartida, as instruções mais complexas não são as que contém maior número de operadores e mutantes.

**Tabela 5.20:** *Cenário 2 - Quantidade de Operadores e Mutantes por Instrução SQL*

| POSIÇÃO | Instrução | Operadores | Mutantes |
|---|---|---|---|
| 1 | 4 | 7 | 32 |
| 2 | 9 | 7 | 67 |
| 3 | 2 | 8 | 101 |
| 4 | 6 | 8 | 67 |
| 5 | 10 | 10 | 129 |
| 6 | 8 | 9 | 98 |
| 7 | 1 | 7 | 25 |
| 8 | 15 | 9 | 134 |
| 9 | 11 | 9 | 95 |
| 10 | 7 | 8 | 50 |
| 11 | 12 | 11 | 96 |
| 12 | 13 | 10 | 77 |
| 13 | 14 | 6 | 22 |
| 14 | 3 | 4 | 12 |
| 15 | 5 | 2 | 4 |

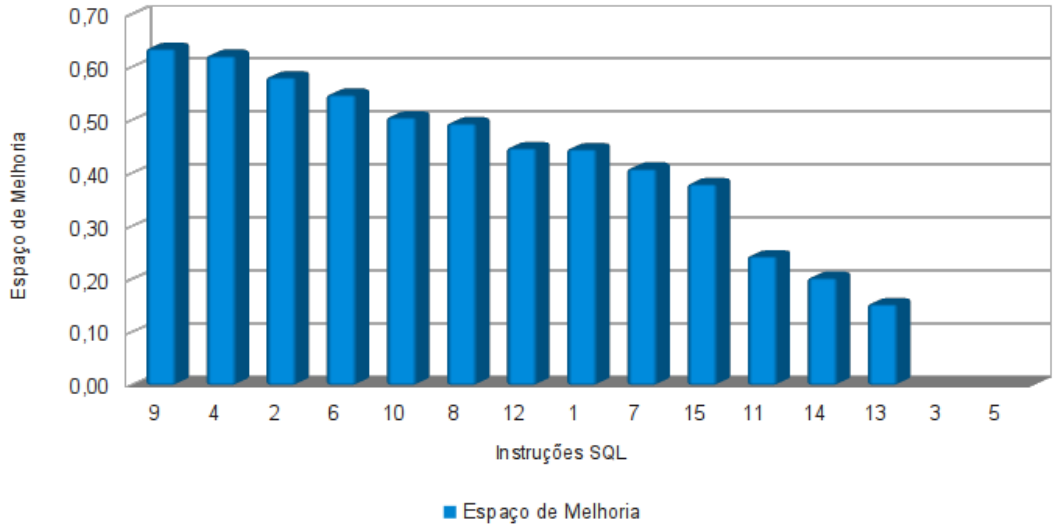


**Figura 5.19:** *Cenário 2 - Espaço de Melhoria Médio das Instruções SQL*

### Espaço de Melhoria Máxima

A Tabela 5.21 apresenta o ranqueamento pelo Espaço de Melhoria Máxima, listando e ordenando as instruções SQL de acordo com este valor. As mesmas informações são apresentadas em forma de gráfico na Figura 5.20.

Pode-se observar na Figura 5.21 o comportamento da Instrução SQL 4, que é a mais complexa pelo critério do Espaço de Melhoria Máxima.

**Tabela 5.21:** *Cenário 2 - Instrução SQL ranqueadas pelo Espaço de Melhoria Máxima*

| POSIÇÃO | Instrução | Média Escore BDT Máxima | Escore BDP | Espaço de Melhoria Máxima |
|---|---|---|---|---|
| 1 | 4 | 0,58858 | 0,90625 | 0,31767 |
| 2 | 10 | 0,74675 | 0,89922 | 0,15247 |
| 3 | 8 | 0,75850 | 0,90816 | 0,14966 |
| 4 | 1 | 0,73333 | 0,88000 | 0,14667 |
| 5 | 15 | 0,73755 | 0,85821 | 0,12066 |
| 6 | 2 | 0,81353 | 0,92079 | 0,10726 |
| 7 | 9 | 0,87315 | 0,94030 | 0,06715 |
| 8 | 7 | 0,86667 | 0,90000 | 0,03333 |
| 9 | 11 | 0,71053 | 0,72632 | 0,01578 |
| 10 | 6 | 0,91040 | 0,92537 | 0,01497 |
| 11 | 12 | 0,92710 | 0,93750 | 0,01040 |
| 12 | 14 | 0,94542 | 0,95455 | 0,00913 |
| 13 | 3 | 0,91670 | 0,91670 | 0,00000 |
| 14 | 5 | 1,00000 | 1,00000 | 0,00000 |
| 15 | 13 | 0,90910 | 0,90909 | 0,00000 |

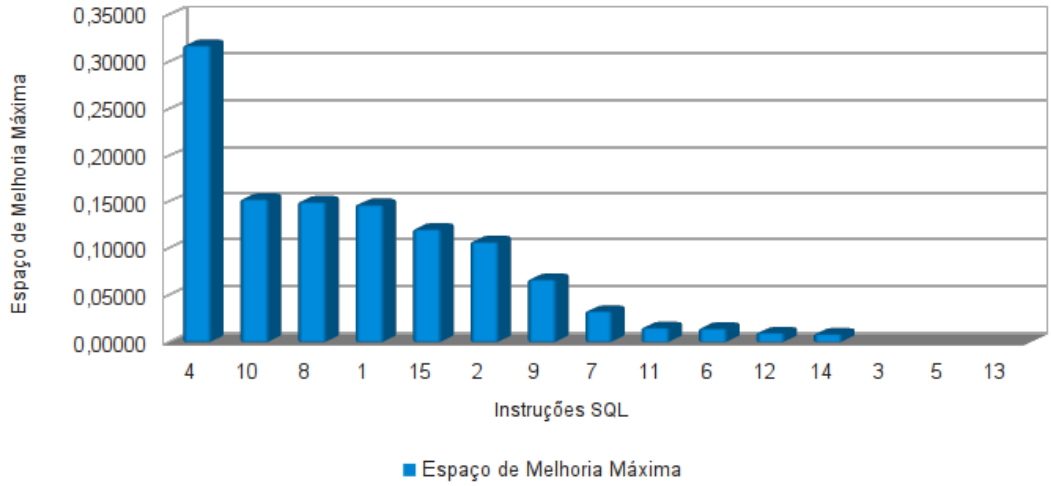


**Figura 5.20:** *Cenário 2 - Espaço de Melhoria Máxima das Instruções SQL*

### Espaço de Melhoria Médio por Tamanho de BDT

A Tabela 5.22 relaciona as 25 situações que apresentaram maior dificuldade com o MA no Cenário 2. Teoricamente, a situação mais difícil no *benchmark* é encontrar uma BDT adequada para a Instrução SQL 2 com tamanho de 0,1%.

O ranqueamento das situações mais complexas do Cenário 2 reforça as conclusões obtidas no mesmo contexto do Cenário 1, indicando que a Instrução SQL não é o único fator envolvido na complexidade. Pode-se observar que as três primeiras situações mais complexas não estão relacionadas com as três instruções SQL mais complexas pelo ranqueamento do Espaço de Melhoria Médio.

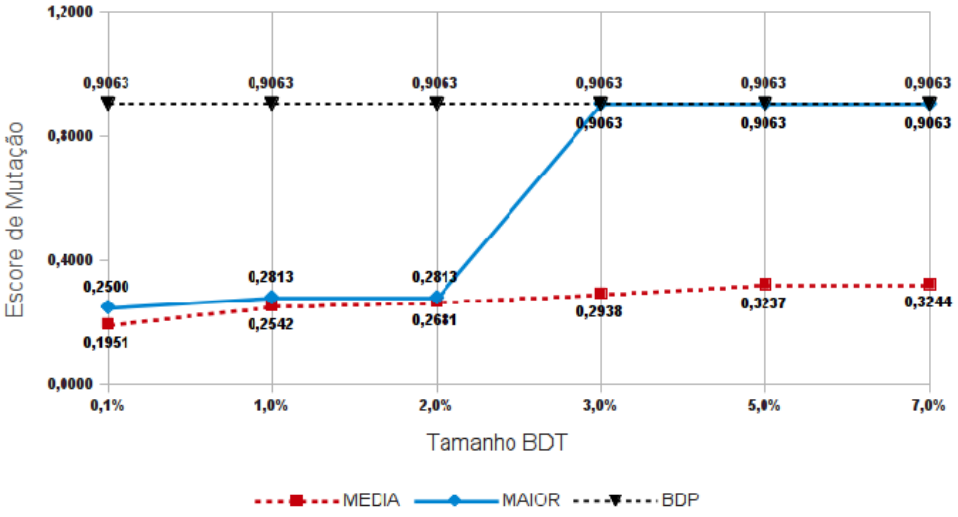


**Figura 5.21:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 4*

**Tabela 5.22:** *Cenário 2 - Ranqueamento das 25 situações com maior complexidade no benchmark*

| POSIÇÃO | Instrução | Qtde Tuplas | Espaço de Melhoria Médio |
|---|---|---|---|
| 1 | 2 | 0,1% | 0,81783 |
| 2 | 6 | 0,1% | 0,80296 |
| 3 | 1 | 0,1% | 0,76000 |
| 4 | 9 | 0,1% | 0,75257 |
| 5 | 15 | 0,1% | 0,74824 |
| 6 | 12 | 0,1% | 0,74790 |
| 7 | 10 | 0,1% | 0,74058 |
| 8 | 7 | 0,1% | 0,73956 |
| 9 | 4 | 0,1% | 0,71110 |
| 10 | 9 | 1% | 0,70282 |
| 11 | 9 | 2% | 0,68424 |
| 12 | 6 | 1% | 0,67760 |
| 13 | 8 | 0,1% | 0,66215 |
| 14 | 10 | 1% | 0,66204 |
| 15 | 2 | 1% | 0,66096 |
| 16 | 4 | 1% | 0,65206 |
| 17 | 14 | 0,1% | 0,64749 |
| 18 | 4 | 2% | 0,63816 |
| 19 | 2 | 2% | 0,63632 |
| 20 | 9 | 3% | 0,63549 |
| 21 | 12 | 1% | 0,63030 |
| 22 | 2 | 3% | 0,62577 |
| 23 | 7 | 1% | 0,61911 |
| 24 | 9 | 5% | 0,61691 |
| 25 | 1 | 1% | 0,61600 |

### 5.2.2 Resultados na perspectiva dos Mutantes

Baseado na taxa de mortalidade, foi construída uma tabela de ranqueamento com todos os 1.009 mutantes não equivalentes gerados no Cenário 2. Ao total, existem 111 mutantes com taxa de mortalidade igual a zero, ou seja, nunca foram mortos. Este alto número é justificado pelo fato do Cenário 2 não ter tido a BDP modificada para

matar todos os mutantes gerados, consequentemente, as BDTs também não conseguirão matar tais mutantes. Na Tabela 5.23, para ilustrar, são apresentados os 50 mutantes mais resistentes com taxa de mortalidade maior que zero. O ranqueamento completo está disponibilizado nos entregáveis do *benchmark*.

**Tabela 5.23:** *Cenário 2 - Lista dos 50 mutantes mais resistentes com taxa de mortalidade maior que zero*

| POSIÇÃO | Mutantes | Operador | Instrução | Taxa de Mortalidade |
|---|---|---|---|---|
| 112 | 999 | IRT | 15 | 0,3704 |
| 113 | 805 | JOI | 13 | 0,5556 |
| 114 | 1004 | IRT | 15 | 1,8519 |
| 115 | 1005 | IRT | 15 | 1,8519 |
| 116 | 1006 | IRT | 15 | 1,8519 |
| 117 | 1007 | IRT | 15 | 1,8519 |
| 118 | 1008 | IRT | 15 | 1,8519 |
| 119 | 1009 | IRT | 15 | 1,8519 |
| 120 | 141 | ABS | 4 | 2,5926 |
| 121 | 142 | UOI | 4 | 2,5926 |
| 122 | 143 | UOI | 4 | 2,5926 |
| 123 | 144 | UOI | 4 | 2,5926 |
| 124 | 145 | IRC | 4 | 2,5926 |
| 125 | 146 | IRC | 4 | 2,5926 |
| 126 | 147 | IRD | 4 | 2,5926 |
| 127 | 148 | IRD | 4 | 2,5926 |
| 128 | 149 | IRD | 4 | 2,5926 |
| 129 | 150 | IRD | 4 | 2,5926 |
| 130 | 151 | IRD | 4 | 2,5926 |
| 131 | 153 | ABS | 4 | 2,5926 |
| 132 | 156 | UOI | 4 | 2,5926 |
| 133 | 158 | IRD | 4 | 2,5926 |
| 134 | 160 | IRD | 4 | 2,5926 |
| 135 | 161 | IRD | 4 | 2,5926 |
| 136 | 162 | IRD | 4 | 2,5926 |
| 137 | 169 | ROR | 4 | 2,5926 |
| 138 | 976 | IRT | 15 | 2,5926 |
| 139 | 977 | IRT | 15 | 2,5926 |
| 140 | 978 | IRT | 15 | 2,5926 |
| 141 | 979 | IRT | 15 | 2,5926 |
| 142 | 980 | IRT | 15 | 2,5926 |
| 143 | 981 | IRT | 15 | 2,963 |
| 144 | 350 | IRC | 8 | 3,3333 |
| 145 | 351 | IRC | 8 | 3,3333 |
| 146 | 352 | IRC | 8 | 3,3333 |
| 147 | 355 | ROR | 8 | 3,3333 |
| 148 | 359 | ROR | 8 | 3,3333 |
| 149 | 369 | LCR | 8 | 3,3333 |
| 150 | 556 | IRT | 10 | 3,3333 |
| 151 | 983 | IRT | 15 | 3,3333 |
| 152 | 984 | IRT | 15 | 3,3333 |
| 153 | 985 | IRT | 15 | 3,3333 |
| 154 | 986 | IRT | 15 | 3,3333 |
| 155 | 987 | IRT | 15 | 3,3333 |
| 156 | 988 | IRT | 15 | 3,3333 |
| 157 | 170 | IRT | 4 | 4,8148 |
| 158 | 624 | IRD | 11 | 4,8148 |
| 159 | 372 | UOI | 8 | 5,1852 |
| 160 | 382 | ROR | 8 | 5,1852 |
| 161 | 515 | ABS | 10 | 5,1852 |

É possível avaliar a resistência dos mutantes por instrução SQL. Para ilustrar, apresentamos na Tabela 5.24 todos os mutantes da Instrução SQL 4 (Código 5.2) orde-

nados pela taxa de mortalidade. Esta mesma informação, para as demais instruções SQL, está disponibilizada nos entregáveis do *benchmark*.

**Código 5.2:** *Instrução SQL 4 do Cenário 2*

```
SELECT chave FROM historico
WHERE professor = 1597
```

**Tabela 5.24:** *Cenário 2 - Relação de mutantes da Instrução SQL 4 ordenados pela Taxa de Mortalidade*

| Mutante | Comando SQL | Operador | Taxa de Mortalidade |
|---|---|---|---|
| 139 | SELECT DISTINCT chave FROM historico WHERE professor = 1597 | SEL | 0 |
| 140 | SELECT ABS(chave) AS chave FROM historico WHERE professor = 1597 | ABS | 0 |
| 152 | SELECT chave FROM historico WHERE ABS(professor) = 1597 | ABS | 0 |
| 141 | SELECT (-ABS(chave)) AS chave FROM historico WHERE professor = 1597 | ABS | 2,5926 |
| 142 | SELECT ((chave)+1) AS chave FROM historico WHERE professor = 1597 | UOI | 2,5926 |
| 143 | SELECT ((chave)-1) AS chave FROM historico WHERE professor = 1597 | UOI | 2,5926 |
| 144 | SELECT (-(chave)) AS chave FROM historico WHERE professor = 1597 | UOI | 2,5926 |
| 145 | SELECT historico.professor FROM historico WHERE professor = 1597 | IRC | 2,5926 |
| 146 | SELECT 1597 FROM historico WHERE professor = 1597 | IRC | 2,5926 |
| 147 | SELECT historico.discente FROM historico WHERE professor = 1597 | IRD | 2,5926 |
| 148 | SELECT historico.disciplina FROM historico WHERE professor = 1597 | IRD | 2,5926 |
| 149 | SELECT historico.periodo FROM historico WHERE professor = 1597 | IRD | 2,5926 |
| 150 | SELECT historico.ano FROM historico WHERE professor = 1597 | IRD | 2,5926 |
| 151 | SELECT historico.codTurma FROM historico WHERE professor = 1597 | IRD | 2,5926 |
| 153 | SELECT chave FROM historico WHERE (-ABS(professor)) = 1597 | ABS | 2,5926 |
| 156 | SELECT chave FROM historico WHERE (-(professor)) = 1597 | UOI | 2,5926 |
| 158 | SELECT chave FROM historico WHERE historico.discente = 1597 | IRD | 2,5926 |
| 160 | SELECT chave FROM historico WHERE historico.periodo = 1597 | IRD | 2,5926 |
| 161 | SELECT chave FROM historico WHERE historico.ano = 1597 | IRD | 2,5926 |
| 162 | SELECT chave FROM historico WHERE historico.codTurma = 1597 | IRD | 2,5926 |
| 169 | SELECT chave FROM historico WHERE (1=0) | ROR | 2,5926 |
| 170 | SELECT chave FROM historico WHERE professor = historico.chave | IRT | 4,8148 |
| 157 | SELECT chave FROM historico WHERE historico.chave = 1597 | IRC | 5,5556 |
| 155 | SELECT chave FROM historico WHERE ((professor)-1) = 1597 | UOI | 70,3704 |
| 159 | SELECT chave FROM historico WHERE historico.disciplina = 1597 | IRD | 76,2963 |
| 154 | SELECT chave FROM historico WHERE ((professor)+1) = 1597 | UOI | 81,1111 |
| 163 | SELECT chave FROM historico WHERE professor <> 1597 | ROR | 100 |
| 164 | SELECT chave FROM historico WHERE professor > 1597 | ROR | 100 |
| 165 | SELECT chave FROM historico WHERE professor < 1597 | ROR | 100 |
| 166 | SELECT chave FROM historico WHERE professor >= 1597 | ROR | 100 |
| 167 | SELECT chave FROM historico WHERE professor <= 1597 | ROR | 100 |
| 168 | SELECT chave FROM historico WHERE (1=1) | ROR | 100 |

Com o ranqueamento dos mutantes apresentados na Tabela 5.24 fica novamente evidente que o grande fator que determina a complexidade de uma instrução SQL é a restritividade envolvida, que pode ser definida pelo comando SQL e/ou conteúdo da BDP.

O Mutante 139 é um ótimo exemplo para mostrar a restritividade causada pela BDP. Nesca caso, o mutante só seria morto se houvesse duas tuplas com o valor 1597 no campo *professor* = 1597. Como na BDP só existe uma tupla com esta característica, este mutante nunca é morto com a BDP, consequentemente com nenhuma BDT. Outro exemplo é o Mutante 140. Ele só seria morto se houvesse alguma tupla na tabela *historico* com valor negativo no campo *chave*.

#### Ranqueamento de Instruções SQL por Taxa de Mortalidade

A Tabela 5.25 lista as instruções SQL ranqueadas pela média da taxa de mortalidade de seus mutantes.

**Tabela 5.25:** *Cenário 2 - Instruções SQL ranqueadas pela Média da Taxa de Mortalidade dos Mutantes*

| POSIÇÃO | Instrução | Total Mutantes | Média Taxa de Mortalidade |
|---|---|---|---|
| 1 | 4 | 32 | 27,65047 |
| 2 | 9 | 67 | 30,40908 |
| 3 | 2 | 101 | 33,45802 |
| 4 | 6 | 67 | 38,65671 |
| 5 | 10 | 129 | 39,38844 |
| 6 | 8 | 98 | 41,17535 |
| 7 | 1 | 25 | 43,77778 |
| 8 | 15 | 134 | 47,63406 |
| 9 | 11 | 95 | 48,60429 |
| 10 | 7 | 50 | 48,80001 |
| 11 | 12 | 96 | 49,43671 |
| 12 | 13 | 77 | 74,48053 |
| 13 | 14 | 22 | 75,43435 |
| 14 | 3 | 12 | 91,66667 |
| 15 | 5 | 4 | 100,00000 |

### 5.2.3 Resultados na perspectiva dos Operadores de Mutação

A Tabela 5.26 apresenta um ranqueamento dos operadores de mutação considerando a taxa de mortalidade dos seus mutantes. Sendo assim, por este ranqueamento, pode-se concluir quais são os operadores que geram os mutantes mais resistentes e menos resistentes do Cenário 2.

**Tabela 5.26:** *Cenário 2 - Operadores de Mutação Ranqueados pela Taxa de Mortalidade*

| POSIÇÃO | Operador | Categoria | Mutantes | Média Taxa de Mortalidade |
|---|---|---|---|---|
| 1 | UNI | SC | 3 | 10,98767 |
| 2 | SEL | SC | 11 | 14,72727 |
| 3 | ABS | OR | 88 | 19,99663 |
| 4 | IRT | IR | 101 | 32,25889 |
| 5 | BTW | OR | 8 | 32,91666 |
| 6 | LKE | OR | 5 | 37,48148 |
| 7 | UOI | OR | 132 | 38,47531 |
| 8 | IRD | IR | 209 | 41,10225 |
| 9 | IRC | IR | 147 | 42,63038 |
| 10 | JOI | SC | 10 | 46,75926 |
| 11 | ORD | SC | 1 | 49,33330 |
| 12 | AGR | SC | 14 | 56,61375 |
| 13 | GRU | SC | 6 | 59,62963 |
| 14 | ROR | OR | 182 | 65,18396 |
| 15 | LCR | OR | 70 | 74,32804 |
| 16 | AOR | OR | 6 | 78,88890 |
| 17 | SUB | SC | 16 | 86,04166 |

### 5.2.4 Ranqueamento Final das Instruções SQL

A Tabela 5.27 relaciona, para cada instrução SQL, a sua posição em cada abordagem de ranqueamento. A primeira abordagem é o ranqueamento pelo Espaço de Melhoria Médio, a segunda pelo Espaço de Melhoria Máxima e a terceira pela Taxa de Mortalidade.

**Tabela 5.27:** *Cenário 2 - Ranqueamento das Instruções SQL para cada Abordagem de Ranqueamento*

| | Posição por Abordagem de Ranqueamento | | |
|---|---|---|---|
| Instrução | (1)Espaço de Melhoria Médio | (2)Espaço de Melhoria Máximo | (3)Taxa de Mortalidade |
| 1 | 8 | 4 | 7 |
| 2 | 3 | 6 | 3 |
| 3 | 14 | 13 | 14 |
| 4 | 2 | 1 | 1 |
| 5 | 15 | 14 | 15 |
| 6 | 4 | 10 | 4 |
| 7 | 9 | 8 | 10 |
| 8 | 6 | 3 | 6 |
| 9 | 1 | 7 | 2 |
| 10 | 5 | 2 | 5 |
| 11 | 11 | 9 | 9 |
| 12 | 7 | 11 | 11 |
| 13 | 13 | 15 | 12 |
| 14 | 12 | 12 | 13 |
| 15 | 10 | 5 | 8 |

Analisando a Tabela 5.27 pode-se observar que no Cenário 2, em relação ao Cenário 1, houve uma maior diferença entre a primeira e a segunda abordagem. No caso das Abordagens 1 e 3, houve uma grande similaridade nos resultados, semelhante ao comportamento do Cenário 1. Novamente pode-se reforçar a relação entre a Taxa de Mortalidade dos mutantes e o Espaço de Melhoria Médio, demonstrando também para a Questão **Q2** que ambos critérios são adequados para determinar e ranquear a complexidade das instruções SQL.

A Tabela 5.28 apresenta novamente todas as instruções e suas posições nos ranqueamentos, acrescida a coluna com a posição do ranqueamento final definido. A Tabela 5.29 apresenta o ranqueamento final das instruções SQL do Cenário 2.

**Tabela 5.28:** *Cenário 2 - Ranqueamento das Instruções SQL para cada Abordagem de Ranqueamento acrescido do ranqueamento final*

| | Posição por Abordagem de Ranqueamento | | | |
|---|---|---|---|---|
| **Instrução** | **(1)Espaço de Melhoria Médio** | **(2)Espaço de Melhoria Máximo** | **(3)Taxa de Mortalidade** | **(4)Final** |
| 1 | 8 | 4 | 7 | 7 |
| 2 | 3 | 6 | 3 | 3 |
| 3 | 14 | 13 | 14 | 14 |
| 4 | 2 | 1 | 1 | 1 |
| 5 | 15 | 14 | 15 | 15 |
| 6 | 4 | 10 | 4 | 4 |
| 7 | 9 | 8 | 10 | 10 |
| 8 | 6 | 3 | 6 | 6 |
| 9 | 1 | 7 | 2 | 2 |
| 10 | 5 | 2 | 5 | 5 |
| 11 | 11 | 9 | 9 | 11 |
| 12 | 7 | 11 | 11 | 9 |
| 13 | 13 | 15 | 12 | 13 |
| 14 | 12 | 12 | 13 | 12 |
| 15 | 10 | 5 | 8 | 8 |

**Tabela 5.29:** *Cenário 2 - Ranqueamento Final de todas as Instruções SQL*

| **POSIÇÃO** | **Instrução SQL** | **Operadores** | **Mutantes** | **Espaço de Melhoria Médio** |
|---|---|---|---|---|
| 1 | 4 | 7 | 32 | 0,54862 |
| 2 | 9 | 7 | 67 | 0,40878 |
| 3 | 2 | 8 | 101 | 0,62184 |
| 4 | 6 | 8 | 67 | 0,49450 |
| 5 | 10 | 10 | 129 | 0,37981 |
| 6 | 8 | 9 | 98 | 0,44578 |
| 7 | 1 | 7 | 25 | 0,63532 |
| 8 | 15 | 9 | 134 | 0,00000 |
| 9 | 12 | 11 | 96 | 0,20275 |
| 10 | 7 | 8 | 50 | 0,44775 |
| 11 | 11 | 9 | 95 | 0,24349 |
| 12 | 14 | 6 | 22 | 0,00000 |
| 13 | 13 | 10 | 77 | 0,15300 |
| 14 | 3 | 4 | 12 | 0,58150 |
| 15 | 5 | 2 | 4 | 0,50592 |

## 5.3 Considerações Finais

Os experimentos realizados para cada cenário do *benchmark* geraram diversos resultados que foram analisados em diferentes perspectivas. No geral, apesar das características distintas, ambos os cenários apresentaram comportamento semelhante no que diz respeito aos fatores que influenciam na complexidade das instâncias dos problemas. A Tabela 5.30 apresenta um comparativo inicial entre os dois cenários.

**Tabela 5.30:** *Comparativo de características entre os dois cenários que compõem o benchmark*

| **Característica** | **Cenário 1 - EMPRESA** | **Cenário 2 - UFG** |
|---|---|---|
| Origem BDP | Literatura | Aplicação Real |
| Dados da BDP | Gerados Aleatoriamente | Extraídos de uma BDP real |
| Tamanho da BDP | Aproximadamente 100.000 tuplas | Aproximadamente 250.000 tuplas |
| Quantidade de Instruções SQL | 20 | 15 |
| Quantidade de Mutantes | 1181 | 1009 |
| Quantidade de Experimentos Realizados | 387 | 238 |
| Quantidade de BDTs geradas | 5795 | 3555 |

Considerando os dois cenários, foram realizados para o *benchmark* 625 experimentos, utilizando 35 instruções SQL e 2.190 mutantes. Aleatoriamente foram geradas ao total 9.350 BDTs, extraídas a partir de aproximadamente 350.000 tuplas.

Como já relatado anteriormente, analisar individualmente o escore de mutação de cada BDT seria impraticável e pouco produtivo. Sendo assim, a análise dos experimentos foi baseada na sintetização dos resultados gerados. A partir desta análise, pretendeu-se ranquear as instruções SQL e responder aos questionamentos declarados no inicio deste capítulo.

Em cada cenário foram identificados padrões e evidências que levaram à conclusões principalmente sobre a complexidade das instruções SQL e, consequentemente, das instâncias dos problemas. Além disso, outras conclusões foram realizadas, gerando informações relevantes no contexto da Análise de Mutantes SQL.

### Q1 - Quais as propriedades de uma base de dados de teste que influenciam no seu escore de mutação?

Com a grande quantidade de experimentos realizados, é possível afirmar, por mais que seja óbvio, que quanto maior o tamanho das BDTs a tendência é que maior seja o escore de mutação. A Instrução 7 do Cenário 2 é um exemplo que mostra esta caracterísica. Porém, para algumas instruções SQL utilizadas no *benchmark*, nota-se que este aumento do escore é mais discreto e lento, como por exemplo as Instruções 13 e 14 do Cenário 1.

Apesar desta característica, o aumento do escore é limitado. A princípio, o limite do escore de mutação de uma BDT é o escore de mutação da BDP[2]. Algumas BDTs

[2]Nos experimentos deste *benchmark* não foi identificado nenhum caso onde o escore da BDT ultrapassou o escore da BDP

atingem este valor máximo de escore com tamanhos (quantidade de tuplas) menores. Nestes casos, pode-se afirmar que essas instruções geram mutantes mais fáceis de morrer. Um bom exemplo é a Instrução 3 do Cenário 1. Com apenas 2% do tamanho da BDP, as BDTs geradas conseguem atingir o escore de mutação máximo.

**Q2 - É possível determinar e ranquear a complexidade[3] das instruções SQL? Como?**

Intuitivamente, uma das formas para verificar a complexidade da instrução SQL é verificar o escore de mutação de uma BDT. Como citado anteriormente, BDTs pequenas que atingem escore de mutação alto indicam que a instrução SQL é menos complexa. Quanto maior a BDT necessária para atingir o escore de mutação máximo, mais complexa é a instrução SQL.

Porém, somente este critério não é adequado para verificar a complexidade das instruções com detalhamento suficiente para ranquear e comparar instruções SQL. É necessário criar e avaliar várias BDTs, de diferente tamanhos, geradas aleatoriamente. A partir dos seus resultados de todas as BDTs, é possível verificar a complexidade da instrução SQL.

Foram propostas duas maneiras para analisar estes resultados. Uma através do Espaço de Melhoria, que é uma medida para verificar o quanto o escore das BDTs geradas aleatoriamente para uma instrução SQL ficou distante do escore da BDP. E a outra maneira, verificando a Taxa de Mortalidade dos mutantes da instrução SQL. Como já foi visto, a Taxa de mortalidade indica o quanto um mutante é resistente.

Tanto o Espaço de Melhoria quanto a Taxa de Mutação foram eficientes critérios para mensurar a complexidade das instruções SQL e consequentemente ranqueá-las. Os resultados de ambas foram muito semelhantes.

**Q3 - Quais fatores influenciam na complexidade de uma instrução SQL?**

Em ambos os cenários, foi possível verificar que a quantidade de mutantes não está relacionada com a complexidade das instruções. Houve casos de instruções com poucos mutantes, que foram classificadas e ranqueadas como muito complexas, e casos de instruções com muitos mutantes, que também foram ranqueadas como muito complexas. O mesmo comportamento foi observado em relação à quantidade de operadores SQL,

[3]Considera-se que quanto maior a dificuldade para matar os mutantes da instrução SQL, maior é a sua complexidade. Consequentemente também, mais difícil é a instância do problema.

sendo assim, conclui-se que esta característica também não tem relação direta com a complexidade das instruções SQL.

Porém, independente da quantidade, os operadores de mutação foram analisados sob uma perspectiva específica. Notou-se que alguns operadores, dentro do contexto de cada cenário deste *benchmark*, tem maior propensão a gerar mutantes com maior ou menor resistência. Baseado nesta tendências, os operadores foram ranqueado. Na Tabela 5.31 são comparados o ranqueamento de cada operador em cada um dos cenários.

**Tabela 5.31:** *Comparação do ranqueamento dos operadores de mutação nos diferentes cenários do benchmark*

| | Ranqueamento | |
|---|---|---|
| | Cenário 1 | Cenário 2 |
| ABS | 11 | 3 |
| AGR | 14 | 12 |
| AOR | 2 | 16 |
| BTW | 7 | 5 |
| GRU | 21 | 13 |
| IRC | 8 | 9 |
| IRD | 10 | 8 |
| IRT | 6 | 4 |
| JOI | 20 | 10 |
| LCR | 16 | 15 |
| LKE | 12 | 6 |
| NLF | 19 | - |
| NLI | 4 | - |
| NLO | 17 | - |
| NLS | 3 | - |
| ORD | 1 | 11 |
| ROR | 15 | 14 |
| SEL | 5 | 2 |
| SUB | 18 | 17 |
| UNI | 13 | 1 |
| UOI | 9 | 7 |

Nota-se que houve uma diferença muito grande entre os três operadores mais difíceis de cada cenário. Em contrapartida, operadores que geraram mutantes menos resistentes foram classificados de forma semelhante em ambos os cenários, como por exemplo o AGR, GRU, o ROR e o SUB.

Bem mais relevante que os operadores de mutação, a **restritividade** das consultas foi o fator que mais influenciou na complexidade das instruções SQL utilizadas no *benchmark*. Quando a consulta SQL (instrução original ou mutante) procura recuperar um pequeno e muito específico grupo de tuplas, o mutante se torna mais resistente, pois a probabilidade de encontrar a(s) tupla(s) certa(s) dentro da BDP é menor. Foram identificados três fatores que estão relacionados com a restritividade:

1. Comando SQL
2. Operador de Mutação
3. Conteúdo da BDP

No geral, o que causou maior restritividade foram os comandos que de alguma forma limitam os resultados, como por exemplo o *BETWEEN*, *HAVING*, *LIKE* e o *IN*. Também deve-se destacar que os elementos observados que causaram grande restritividade nas consultas foram os operadores relacionais, com destaque para o **operador de igualdade**.

Entretanto, observou-se que somente estes comandos SQL e/ou o operador de igualdade não são suficientes para tornar o mutante resistente. Como exemplo, existem mutantes que usam tais elementos mas possuem uma alta taxa de mortalidade. Concluiu-se que são os comandos SQL, aliados ao conteúdo da BDP, que exercem grande influência na resistência dos mutantes.

Um ótimo exemplo que mostra como a restritividade é influênciada pelo comando SQL e pelo conteúdo da BDP são os Mutantes 947 e 961 da Instrução SQL 14 do Cenário 1. O Mutante 947 tem a taxa de mortalidade de apenas 4,19%, ou seja, um mutante muito difícil de ser morto, sendo que das 405 BDTs geradas para a instrução, somente 17 conseguiram matá-lo. Já o Mutante 961 tem a taxa de mortalidade de 96,54%.

Pode-se observar no Código 5.3 e no Código 5.4 que a única diferença entre os dois mutantes é o operador relacional usado no campo *ssn* da cláusula *WHERE*. No primeiro comando SQL é usado o sinal de igualdade, sendo assim, somente uma tupla da BDP atende essa restrição, tornando a consulta altamente restritiva. Já no segundo é usado o sinal de menor (<), possibilitando que todas as tuplas da BDP com *ssn* menor que 1003 possam ser recuperadas.

**Código 5.3:** *Mutante SQL 947 do cenário 1*

```
SELECT employee.minit
FROM EMPLOYEE
WHERE ssn = 1003
```

**Código 5.4:** *Mutante SQL 961 do cenário 1*

```
SELECT employee.minit
FROM EMPLOYEE
WHERE ssn < 1003
```

Além da restritividade, pode-se observar outros fatores relacionados especificamente com a complexidade baixa. Avaliando as instruções menos complexas de cada cenário, identificou-se alguns comandos SQL que tornam os mutantes destas instruções menos resistentes, sendo os principais deles o *GROUP BY* e o *DISTINCT*.

Os operadores de mutação, que estão relacionados diretamente com os elementos do SQL que causam maior ou menor restritividade, são um terceiro fator envolvido que podem gerar mutantes mais restritivos ainda. De qualquer forma, os resultados mostram que os comandos SQL, conteúdo da BDP e a combinação deles são os principais fatores que tornam as instruções SQL com maior ou menor complexidade, sendo esta a conclusão mais importante deste capítulo.

CAPÍTULO 6

# Considerações Finais

*"Transportai um punhado de terra todos os dias e fareis uma montanha."*
*Confúcio*

Esta dissertação apresentou um trabalho que contribui com os esforços atuais da literatura para a redução de custos na área de Teste de Software, mais especificamente em softwares que envolvem persistência de dados, o qual são denominados Aplicações de Bancos de Dados (ABD).

O trabalho gerado agregou conhecimento aos seguintes domínios: (i) construção de bases de dados de teste (BDTs), pela redução de bases de produção, para testes de ABD; (ii) uso da Análise de Mutantes SQL para mensurar a qualidade de bases de teste; (iii) avaliação de desempenho de técnicas de busca que visam realizar a construção de bases de teste de boa qualidade.

Sendo assim, o principal objetivo alcançado foi a construção e disponibilização de um *benchmark* que funciona como um ambiente de referência, fornecendo todos os elementos necessários para avaliar o desempenho das técnicas de busca (domínio iii) no contexto de redução de bases de dados para testes de ABD (domínio i), utilizando a Análise de Mutantes SQL como critério de avaliação (domínio ii).

Para a construção do *benchmark*, foram criados dois cenários, sendo cada um composto por uma base de dados de produção (BDP) e um conjunto de instruções SQL com seus devidos mutantes gerados pela ferramenta *SQLMutantion*. Foram realizados 625 experimentos, criando aleatoriamente 9.350 BDTs de diferentes tamanhos a partir da redução das BDPs de cada cenário. Considerando que o método aleatório é o que possui menor custo computacional, os resultados de escore de mutação destas BDTs são as principais referências para avaliar as técnicas de busca.

Para disponibilizar outra importante informação como referência no *benchmark*, foram definidas neste trabalho formas de determinar o grau de complexidade de cada

instrução SQL. Neste sentido, foi proposto um novo critério, baseado no escore de mutação, chamado de Espaço de Melhoria. Outro critério utilizado foi a Taxa de Mortalidade, proposta por Tuya et al. [50]. Utilizando estes dois critérios, a complexidade da cada instrução foi mensurada e comparada.

Com estes valores e também a partir dos experimentos gerados, foi atingido o segundo objetivo deste trabalho, que é a análise e interpretações dos resultados para inferir sobre importantes questões da Análise de Mutantes SQL. Sendo assim, definiu-se no contexto desta pesquisa, quais são os principais fatores envolvidos na complexidade das instruções SQL.

Em resumo, a restritividade das consultas se destaca como sendo o fator mais determinante na complexidade das instruções SQL. Esta restritividade está relacionada com os comandos SQL utilizados e com o conteúdo da base de dados. Quanto mais específico e restrito o retorno da consulta, a tendência é que mais complexa seja a instrução SQL no contexto da análise de mutantes.

## 6.1 Contribuições

Pode-se destacar como principais contribuições deste trabalho:

- Disponibilizar na literatura um *benchmark* específico e padronizado para que outros pesquisadores possam avaliar técnicas de busca no contexto de redução de bases de dados para testes de aplicações SQL. Com este ambiente de referência é possível comparar, de maneira imparcial e criteriosa, diferentes técnicas que visam reduzir bases de dados mantendo o seu mesmo poder de revelação de defeitos.
- Construção e disponibilização de uma ferramenta para possibilitar que pesquisadores da área conduzam experimentos semelhantes aos realizados neste trabalho, podendo utilizar outros parâmetros e/ou outras instruções SQL, expandindo o *benchmark* conforme a necessidade dos seus utilizadores.
- A partir da análise do grande volume de resultados gerados pelo Método Aleatório, este trabalho fornece um conjunto de interpretações sobre o comportamento das instruções SQL, dos operadores de mutação e das bases de dados de teste. Estas interpretações ajudaram a responder importantes questionamentos sobre a Análise de Mutantes SQL, principalmente no que diz respeito à complexidade das instruções SQL.

## 6.2 Limitações

Considerando os principais objetivos, o presente trabalho possui algumas limitações que devem ser condideradas, sendo elas:

- Não foram utilizadas técnicas avançadas de estatística na avaliação dos resultados gerados. Para inferir sobre a complexidade das instâncias de problemas, as análises se limitaram no uso de médias simples e desvios padrões.
- Como valores de referências, o presente trabalho somente fornece resultados gerados pelo Método Aleatório. Não foram realizados e incorporados resultados de experimentos com técnicas de busca e/ou metaheurísticas.
- Estão disponíveis na literatura diversos grupos operadores de mutação para instruções SQL, porém foram utilizados neste trabalho somente um grupo de operadores de mutação.
- Apesar do uso de instruções SQL construidas a partir do padrão ANSI SQL3, que a princípio garante a compatibilidade das instruções com qualquer SGBD relacional, os experimentos deste trabalho foram realizados somente no SGBD MySQL.

## 6.3 Trabalhos Futuros

Durante o desenvolvimento desta dissertação, algumas possibilidades de trabalhos foram identificadas. Sendo assim, o propósito inicial do *benchmark* pode ser expandido em diferentes aspectos, permitindo outras contribuições interessantes. Como principais trabalhos futuros pode-se destacar:

- Utilizar o *benchmark* deste trabalho para avaliar o Algoritmo Genético proposto por Monção et al. [34]. Com os resultados obtidos, gerar mais uma referência de comparações para ser incorporada ao *benchmark*.
- Além da redução de bases de dados, outra área de pesquisa muito ativa no que se refere ao custo do teste é a redução de mutantes e/ou casos de teste. Sendo assim, o *benchmark* aqui proposto pode ser adaptado e utilizado também para avaliar técnicas que tenham como foco estas reduções.
- Conforme apresentado nos capítulos anteriores, o grupo de operadores proposto por Tuya et al. [50] é o mais utilizado em trabalhas da área. Porém, outros operadores foram propostos na literatura. Uma possibilidade é usar as mesmas BDPs e as mesmas instruções SQL do *benchmark* para criar novos mutantes utilizando outros conjutos de operadores de mutação. A partir destes novos mutantes, novos experimentos podem ser conduzidos e comparados com os experimentos já realizados neste trabalho.

Ao final seria possível responder qual é o grupo de operadores de mutação SQL mais adequado na revelação de defeitos.

- Offutt et al. [16] afirmam que mutantes de ordem mais alta não contribuem para melhora significativa dos casos de teste. Já Jia et al. [24] afirmam que mutantes de mais alta ordem são potencialmente melhores para simular defeitos reais e também identificar erros que mutantes de primeira ordem não conseguem simular. Uma possibilidade então seria estender o *benchmark* incluindo mutantes de segunda ordem e avaliar se no contexto de instruções SQL estes mutantes seriam realmente mais eficientes na identificação de defeitos.
- Em 1996, Offutt e Hayes [36] propuseram o conceito de Tamanho Semântico para mensurar a resistência de mutantes. Eles afirmam que mutantes mais resistentes (*hard-to-kill*) podem conduzir a construção de casos de teste mais eficientes. Um trabalho interessante a ser realizado seria avaliar, por outra perspectiva, os resultados gerados no *benchmark* para validar a teoria de Offut e Hayes no contexto de mutantes SQL.

# Referências Bibliográficas


[1] AGRAWAL, H.; DEMILLO, R.; HATHAWAY, R.; HSU, W.; HSU, W.; KRAUSER, E.; MARTIN, R. J.; MATHUR, A.; SPAFFORD, E. **Design of mutant operators for the c programming language**. Technical report, Technical Report SERC-TR-41-P, Software Engineering Research Center, Purdue University, 1989.

[2] AMMANN, P.; OFFUTT, J. **Introduction to software testing**. Cambridge University Press, 2008.

[3] BARBOSA, E. F.; VINCENZI, A. M.; MALDONADO, J. C. **Uma contribuição para a determinação de um conjunto essencial de operadores de mutação no teste de programas c**. *Simpósio Brasileiro de Engenharia de Software–SBES*, 98:33–34, 1998.

[4] BERRY, M.; CYBENKO, G.; LARSON, J. **Scientific benchmark characterizations**. *Parallel Computing*, 17(10):1173–1194, 1991.

[5] BORAL, H.; DEWITT, D. J. **A methodology for database system performance evaluation**, volume 14. ACM, 1984.

[6] BOWSER, J. H. **Reference manual for ada mutant operators**. *Georiga Institute of Technology, Atlanta, Georgia, Technique Report GITSERC-88/02*, 1988.

[7] BRASS, S.; GOLDBERG, C. **Semantic errors in sql queries: A quite complete list**. *Journal of Systems and Software*, 79(5):630–644, 2006.

[8] BUDD, T. A.; DEMILLO, R. A.; LIPTON, R. J.; SAYWARD, F. G. **The design of a prototype mutation system for program testing**. In: *Proceedings of National Computer Conference*, p. 623–627, 1978.

[9] BUDD, T. A. **Mutation analysis of program test data [ph. d. thesis]**. 1980.

[10] CABEÇA, A. G.; JINO, M.; LEITAO-JUNIOR, P. S. **Análise de mutantes em aplicações sql de banco de dados**. *VII Simpósio Brasileiro de Qualidade de Software*, 2008.

[11] CHAN, W. K.; CHEUNG, S.; TSE, T. H. **Fault-based testing of database application programs with conceptual data model**. *Quality Software, International Conference on*, 0:187–196, 2005.

[12] CHAYS, D.; DENG, Y.; FRANKL, P. G.; DAN, S.; VOKOLOS, F. I.; WEYUKER, E. J. **An agenda for testing relational database applications**. *Software Testing, verification and reliability*, 14(1):17–44, 2004.

[13] CIFERRI, R. R. **Um benchmark voltado a análise de desempenho de sistemas de informações geográficas**. 1995.

[14] COUNCIL, T. P. P. **Tpc benchmark b**. *Standard Specification, Waterside Associates, Fremont, CA*, 1990.

[15] DELAMARO, M.; MALDONADO, J. C. **Interface mutation: Assessing testing quality at interprocedural level**. In: *Computer Science Society, 1999. Proceedings. SCCC'99. XIX International Conference of the Chilean*, p. 78–86. IEEE, 1999.

[16] DEMILLI, R.; OFFUTT, A. **Constraint-based automatic test data generation**. *Software Engineering, IEEE Transactions on*, 17(9):900–910, 1991.

[17] DEMILLO, R. A. **Mutation analysis as a tool for software quality assurance**. In: *COMPSAC80*, p. 390–393, Chicago, IL, October 1980.

[18] DEMILLO, R.; LIPTON, R.; SAYWARD, F. **Hints on test data selection: Help for the practicing programmer**. *Computer*, 11(4):34 –41, april 1978.

[19] DEREZINSKA, A. **An experimental case study to applying mutation analysis for sql queries**. In: *Computer Science and Information Technology, 2009. IMCSIT'09. International Multiconference on*, p. 559–566. IEEE, 2009.

[20] DEWITT, D. J. **Benchmarking database systems: Past effords and future diretions**. *Database Engineering Bulletin*, 8(1):2–9, 1985.

[21] GRAY, J. **Benchmark handbook: for database and transaction processing systems**. Morgan Kaufmann Publishers Inc., 1992.

[22] GRUN, B. J.; SCHULER, D.; ZELLER, A. **The impact of equivalent mutants**. In: *Software Testing, Verification and Validation Workshops, 2009. ICSTW'09. International Conference on*, p. 192–199. IEEE, 2009.

[23] GUPTA, B. P.; VIRA, D.; SUDARSHAN, S. **X-data: Generating test data for killing sql mutants**. In: *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, p. 876–879. IEEE, 2010.

[24] HARMAN, M.; JIA, Y.; LANGDON, W. B. **A manifesto for higher order mutation testing**. In: *Software Testing, Verification, and Validation Workshops (ICSTW), 2010 Third International Conference on*, p. 80–89. IEEE, 2010.

[25] HARMAN, M.; MCMINN, P.; DE SOUZA, J. T.; YOO, S. **Search based software engineering: Techniques, taxonomy, tutorial**. In: *Empirical Software Engineering and Verification*, p. 1–59. Springer, 2012.

[26] HU, J.; LI, N.; OFFUTT, J. **An analysis of oo mutation operators**. In: *Software Testing, Verification and Validation Workshops (ICSTW), 2011 IEEE Fourth International Conference on*, p. 334–341. IEEE, 2011.

[27] JIA, Y.; HARMAN, M. **An analysis and survey of the development of mutation testing**. *Software Engineering, IEEE Transactions on*, 37(5):649 –678, sept.-oct. 2011.

[28] JOSLIN, E. O.; CHAIRMAN-HITTI, R. **Evaluation and performance of computers: application benchmarks: the key to meaningful computer evaluations**. In: *Proceedings of the 1965 20th national conference*, p. 27–37. ACM, 1965.

[29] KIM, S.-W.; CLARK, J. A.; MCDERMID, J. A. **Investigating the effectiveness of object-oriented testing strategies using the mutation method**. *Software Testing, Verification and Reliability*, 11(4):207–225, 2001.

[30] LUCAS JR, H. **Performance evaluation and monitoring**. *ACM Computing Surveys (CSUR)*, 3(3):79–91, 1971.

[31] MA, Y.-S.; OFFUTT, J.; KWON, Y. R. **Mujava: an automated class mutation system**. *Software Testing, Verification and Reliability*, 15(2):97–133, 2005.

[32] MALDONADO, J. C.; BARBOSA, E. F.; VINCENZI, A. M. R.; DELAMARO, M. E.; SOUZA, S.; JINO, M. **Introdução ao teste de software**. *São Carlos*, 2004.

[33] MANTERE, T.; ALANDER, J. T. **Evolutionary software engineering, a review**. *Applied Soft Computing*, 5(3):315–331, 2005.

[34] MONÇÃO, A. C. B. L.; QUEIROZ, L. T.; CAMILO JUNIOR, C. G.; RODRIGUES, C. L.; LEITÃO JÚNIOR, P.; VINCENZI, A. M. **Applying genetic algorithms to data selection for sql mutation analysis**. *Genetic and Evolutionary Computation Conference*, 2013.

[35] OFFUTT, A. J.; CRAFT, W. M. **Using compiler optimization techniques to detect equivalent mutants**. *Software Testing, Verification and Reliability*, 4(3):131–154, 1994.

[36] OFFUTT, A. J.; HAYES, J. H. **A semantic model of program faults**. *SIGSOFT Softw. Eng. Notes*, 21(3):195–200, May 1996.

[37] PAN, K.; WU, X.; XIE, T. **Guided test generation for database applications via synthesized database interactions**. Technical report, Technical report, UNC Charlotte, 2012, http://coitweb. uncc. edu/˜ xwu/DBGen.

[38] PAN, K.; WU, X.; XIE, T. **Automatic test generation for mutation testing on database applications**. 2013.

[39] PÖNIGHAUS, R. **'favourite'sql-statements—an empirical analysis of sql-usage in commercial applications**. In: *Information Systems and Data Management*, p. 75–91. Springer, 1995.

[40] PRESSMAN, R. S.; INCE, D. **Software engineering: a practitioner's approach**, volume 5. McGraw-hill New York, 1992.

[41] RAMEZ, E.; N., S. B. **Fundamentals of Database Systems**. Pearson, 6ª edition, 2005.

[42] SHAH, S.; SUDARSHAN, S.; KAJBAJE, S.; PATIDAR, S.; GUPTA, B.; VIRA, D. **Generating test data for killing sql mutants: A constraint-based approach**. In: *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, p. 1175 –1186, april 2011.

[43] SHAHRIAR, H.; ZULKERNINE, M. **Music: Mutation-based sql injection vulnerability checking**. In: *Quality Software, 2008. QSIC'08. The Eighth International Conference on*, p. 77–86. IEEE, 2008.

[44] SUÁREZ-CABAL, M.; DE LA RIVA, C.; TUYA, J. **Populating test databases for testing sql queries**. *Latin America Transactions, IEEE (Revista IEEE America Latina)*, 8(2):164–171, 2010.

[45] TURBYFILL, C.; ORJI, C.; BITTON, D. **As3ap: An ansi sql standard scaleable and portable benchmark for relational database systems**. *The Benchmark Handbook for Database and Transaction Processing Systems*, p. 317–358, 1993.

[46] TUYA, J.; SUÁREZ-CABAL, M. J.; DE LA RIVA, C. **A practical guide to sql white-box testing**. *ACM SIGPLAN Notices*, 41(4):36–41, 2006.

[47] TUYA, J.; SUÁREZ-CABAL, M. J.; DE LA RIVA, C. **Query-aware shrinking test databases**. In: *Proceedings of the Second International Workshop on Testing Database Systems*, p. 6. ACM, 2009.

[48] TUYA, J.; SUAREZ-CABAL, M. J.; DE LA RIVA, C. **Sqlmutation: A tool to generate mutants of sql database queries**. In: *Mutation Analysis, 2006. Second Workshop on*, p. 1–1. IEEE, 2006.

[49] TUYA, J.; SUAREZ-CABAL, M. J.; DE LA RIVA, C. **Sqlmutation: A tool to generate mutants of sql database queries**. *Mutation Analysis, Workshop on*, 0:1, 2006.

[50] TUYA, J.; SUÁREZ-CABAL, M. J.; DE LA RIVA, C. **Mutating database queries**. *Information and Software Technology*, 49(4):398 – 417, 2007.

[51] ZHOU, C.; FRANKL, P. **Jdama: Java database application mutation analyser**. *Software Testing, Verification and Reliability*, 21(3):241–263, 2011.

APÊNDICE A

# Glossário

**ABD** - Aplicações de Banco de Dados

**AG** - Algoritmo Genético

**ANSI** - *American National Standards Institute*

**AS3AP** - *ANSI SQL Standard Scalable and Portable*

**BDE** – Base de Dados de Experimentos

**BDP** – Base de Dados de Produção

**BDT** – Base de Dados de Teste

**FAPEG** – Fundação de Amparo à Pesquisa do Estado de Goiás

**JDBC** - *Java Database Connectivity*

**MA** - *Método Aleatório*

**MEC** - *Ministério da Educação e Cultura*

**NIST** - *National Institute of Standards and Technology*

**UFG** – Universidade Federal de Goiás

**SEGPLAN** – Secretaria de Estado de Gestão e Planejamento

**SGBD** - Sistemas de Gerenciamento de Banco de Dados

**SHA** - *Secure Hash Algorithm*

**SQL** – *Structured Query Language*

**TPC** - *Transaction Processing Performance Council*

APÊNDICE B

# Entregáveis do *Benchmark*

O *Benchmark* é formado por três componentes principais: (i) Banco de Dados de Produção; (ii) Instruções SQL e Mutantes e (iii) Resultados do Método Aleatório. Para cada componente, serão listados um conjunto de entregáveis previstos que vão fisicamente formar o *benchmark*. Além disso, também estão disponibilizados como entregáveis, todo o ambiente (ferramenta e BDE) que foi utilizado para conduzir os experimentos.

Para facilitar a sua organização e disponibilização física, adotou-se um padrão de nomenclatura para os entregáveis. Por exemplo, **C2-BDP-E1** significa: Entregável 1 (E1) do componente Banco de Dados de Produção (BDP) do segundo cenário (C2).

Neste apêndice, para cada componente, são relacionados os entregáveis previstos, bem como o detalhamento do conteúdo de cada um deles. Fisicamente, todos os entregáveis estarão disponibilizados em meio digital anexado nesta dissertação.

### B.0.1 Base de Dados de Produção

Os entregáveis para este componente devem possibilitar que os interessados em utilizar o *benchmark* consigam facilmente entender, instanciar e utilizar a BDP. A Tabela B.1 relaciona quais são estes entregáveis.

**Tabela B.1:** *Relação de entregáveis da Base de Dados de Produção*

| **Rastreabilidade** | **Entregável** |
|---|---|
| BDP-E1 | Diagrama de modelo conceitual do banco de dados |
| BDP-E2 | Diagrama do modelo físico do banco de dados |
| BDP-E3 | *Scripts* SQL de criação das tabelas |
| BDP-E4 | *Scripts* SQL de carga de dados |
| BDP-E5 | Arquivo XML com o esquema físico da BDP |

### B.0.2 Instruções SQL e Mutantes

Na Tabela B.2 estão relacionados todos os entregáveis das Instruções SQL e Mutantes.

**Tabela B.2:** *Relação de entregáveis das Instruções SQL e Mutantes*

| Rastreabilidade | Entregável |
|---|---|
| SQL-E1 | Planilha de Instruções SQL |
| SQL-E2 | Planilha de Mutantes SQL |

No entregável **SQL-E1**, serão relacionadas todas as instruções SQL usadas no *benchmark*. Para cada instrução serão listadas as seguintes informações:

1. Identificação numérica única da Instrução;
2. Comando SQL;
3. Origem da instrução, indicando se ela foi inspirada em alguma instrução já existente ou se foi criada especificamente para o *benchmark*;
4. Quantidade de mutantes gerados a partir da instrução;
5. Quantidade de operadores de mutação exercitados pela instrução;
6. Escore de Mutação usando a BDP como dado de entrada;
7. Espaço de Melhoria (Medida de complexidade).
8. Média Taxa de Mortalidade (Medida de complexidade).

No entregável **SQL-E2**, serão relacionados por mutante as seguintes informações:

1. Identificação da instrução SQL que originou o mutante;
2. Identificação numérica única do mutante;
3. Comando SQL do mutante;
4. Identificação do operador de mutação utilizado para a geração do mutante.
5. Taxa de Mortalidade (Medida de complexidade)

### B.0.3 Resultados do Método Aleatório

Com os valores calculados após a finalização dos experimentos com o método aleatório, será possível consolidar e disponibilizar os resultados através dos entregáveis listados na Tabela B.3.

Serão listadas no entregável **RST-E1** as informações mais básicas, porém essenciais, sobre os resultados obtidos por cada experimento realizado.

Para cada experimento, serão relacionadas as seguintes informações no RST-E1:

**Tabela B.3:** *Relação de entregáveis dos Resultados do Método Aleatório*

| Rastreabilidade | Entregável |
|---|---|
| RST-E1 | Planilha de Resultados Gerais por Experimento |
| RST-E2 | Planilha de Resultados por Tamanho de BDT |
| RST-E3 | Planilha de Ranqueamento por Taxa de Mortalidade dos Mutantes |
| RST-E4 | Planilha de Resultados por Operador |
| RST-E5 | Planilha de Resultados por Operador/Instrução |
| RST-E6 | Relatório Consolidado por Instrução |

1. Identificação numérica única do experimento;
2. Identificação da instrução SQL que será objeto de teste no experimento;
3. Quantidade de BDTs geradas (tamanho do conjunto);
4. Quantidade de tuplas de cada BDT (percentual de redução da BDP);
5. Maior escore de mutação alcançado no experimento;
6. Menor escore de mutação alcançado no experimento;
7. Escore de mutação médio;
8. Escore de mutação do conjunto;
9. Desvio padrão entre os escores de mutação alcançados.

O **RST-E2** apresenta os valores de escore de mutação agrupados por instrução SQL e por tamanho da BDT, independente dos experimentos que geraram as BDTs. Esta forma de consolidação de resultado é interessante para avaliar o comportamento do escore de mutação a medida que o tamanho da BDT é incrementado. Para cada par (instrução SQL - tamanho de BDT) serão apresentadas no entregável RST-E2 as seguintes informações:

1. Identificação da instrução SQL
2. Tamanho (quantidade de tuplas) da BDT
3. Escore de mutação médio
4. Maior escore de mutação
5. Escore de Mutação da BDP
6. Menor escore de mutação
7. Desvio padrão

O **RST-E3** relaciona para todos os mutantes do cenário (independente da instrução SQL) a quantidade de vezes que ele foi morto durante os experimentos e a sua taxa de mortalidade.

O **RST-E4** apresenta alguns resultados consolidados e agrupados pelos operadores de mutação. Estas informações já fornecem um indicativo em relação as características

dos operadores que já permitiria algumas conclusões sobre o comportamento deles.

Para cada operador de mutação utilizado nos experimentos, serão listas as seguintes informações no entregável RST-E4:

1. Nome do operador de mutação;
2. Categoria do operador de mutação;
3. Quantidade de mutantes gerados;
4. Quantidade de mutantes equivalentes;
5. Quantidade de mutantes mortos;
6. Média da taxa de mortalidade dos mutantes do operador.

No **RST-E5**, basicamente serão fornecidas as mesmas informações do RST-E4, porém, agrupadas por instrução SQL e operador. Desta forma será possível avaliar o comportamento de um operador em cada instrução SQL específica. Serão apresentadas as seguintes informações:

1. Identificação da instrução SQL;
2. Nome do operador de mutação;
3. Quantidade de mutantes gerados;
4. Média da taxa de mortalidade dos mutantes do operador;

O mais natural e relevante é que sejam avaliados os resultados do MA na perspectiva das instruções SQL, principalmente porque elas compoem as instâncias dos problemas. No **RST-E6** serão apresentadas informações já listadas em outros entregáveis, porém agrupadas e consolidadas por cada instrução SQL, facilitando a avaliação e interpretação da complexidade envolvida. Neste entregável será possível visualizar as seguintes informações por instrução:

1. Resultados de todos experimentos da instrução SQL;
2. Resultados por tamanho de BDT;
3. Resultados de mortalidade por operador;
4. Gráfico de evolução do escore de mutação por tamanho do BDT, sendo plotadas para cada tamanho de BDT informações sobre a média do MA, maior escore do MA e escore da BDP.

### B.0.4 Ferramenta e Base de Dados de Experimentos (BDE)

O ambiente utilizado para conduzir os experimentos é formado por uma ferramenta implementada em linguagem Java e uma Base de Dados de Experimentos, onde

são registrados todos os resultados obtidos. Todos os entregáveis deste contexto estão relacionados na Tabela B.4.

**Tabela B.4:** *Relação de entregáveis da Ferramenta e BDE*

| **Rastreabilidade** | **Entregável** |
|---|---|
| BDE-E1 | Código Fonte da Ferramenta |
| BDE-E2 | Script de criação e carga da BDE |

APÊNDICE C

# Instruções SQL Geradas para o *Benchmark*

## C.1 Cenário 1 - EMPRESA

A Tabela C.1 lista as instruções SQL criadas para o cenário 1 do *benchmark*.

**Tabela C.1:** *Instruções SQL do Cenário 1*

| Id | Instrução |
|---|---|
| 1 | SELECT BDATE, ADDRESS FROM EMPLOYEE WHERE FNAME='JOAO' AND minit='J' AND lname='RAMIRO' |
| 2 | SELECT FNAME,LNAME,ADDRESS FROM EMPLOYEE, DEPARTMENT WHERE dname='SECTOR 155' AND DNUMBER=DNO |
| 3 | SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME FROM EMPLOYEE AS E, EMPLOYEE AS S WHERE E.SUPERSSN=S.SSN |
| 4 | SELECT DISTINCT SALARY FROM EMPLOYEE |
| 5 | SELECT FNAME, LNAME FROM EMPLOYEE WHERE ADDRESS LIKE '%Goiânia - GO%' |
| 6 | SELECT DNO, COUNT(*), AVG(SALARY) FROM EMPLOYEE GROUP BY DNO |
| 7 | SELECT SUM(SALARY), MAX(SALARY), MIN(SALARY), AVG(SALARY) FROM (EMPLOYEE JOIN DEPARTMENT ON DNO=DNUMBER) WHERE DNAME='SECTOR 155' |
| 8 | SELECT LNAME, FNAME FROM EMPLOYEE WHERE SALARY > ALL (SELECT SALARY FROM EMPLOYEE WHERE DNO=5) |
| 9 | SELECT DNUMBER, COUNT(*) FROM DEPARTMENT, EMPLOYEE WHERE DNUMBER=DNO AND SALARY>10000 AND DNO IN (SELECT DNO FROM EMPLOYEE GROUP BY DNO HAVING COUNT(*) > 4) GROUP BY DNUMBER |
| 10 | SELECT BDATE, salary as gross_salary, (salary*0.15) as tax FROM EMPLOYEE WHERE dno in (5, 155) and ((salary < 1000 ) or salary is null) ORDER BY SALARY DESC |
| 11 | SELECT FNAME, LNAME FROM EMPLOYEE WHERE ADDRESS LIKE '%Goiânia - GO%' UNION SELECT FNAME, LNAME FROM EMPLOYEE WHERE ADDRESS LIKE '%Brasília - DF%' |
| 12 | SELECT * FROM employee WHERE MINIT='J' AND LNAME LIKE 'RAMIRO' |
| 13 | SELECT SSN FROM EMPLOYEE WHERE SALARY BETWEEN 1000 AND 1500 |
| 14 | SELECT fname FROM EMPLOYEE WHERE ssn = 1003 |
| 15 | SELECT SSN FROM EMPLOYEE WHERE (DNO BETWEEN 10 AND 15 OR DNO = 5) AND (SALARY < 1500) |
| 16 | select DISTINCT fname as NOME from employee where fname LIKE 'LEO%' UNION ALL select DISTINCT lname as NOME from employee where lname LIKE 'LEO%' |
| 17 | SELECT DNAME, COUNT(*) FROM EMPLOYEE, DEPARTMENT WHERE DNO = dnumber AND SALARY < 1000 GROUP BY DNAME HAVING COUNT(*) > 4 ORDER BY DNAME |
| 18 | SELECT SEX, COUNT(*), avg(salary) TOTAL FROM EMPLOYEE WHERE superssn IN (SELECT SSN FROM EMPLOYEE WHERE DNO = 5 AND ADDRESS LIKE '%GO') GROUP BY SEX |
| 19 | SELECT SSN, DNAME FROM EMPLOYEE INNER JOIN department on EMPLOYEE.ssn = department.mgrssn WHERE dname like 'MANAGEMENT%' and dno < 150 |
| 20 | SELECT * FROM EMPLOYEE WHERE (SALARY BETWEEN 1000 AND 1500) AND DNO=5 |

## C.2 Cenário 2 - UFG

A Tabela C.2 lista as instruções SQL criadas para o cenário 2 do *benchmark*.

**Tabela C.2:** *Instruções SQL do Cenário 2*

| Id | Instrução |
|---|---|
| 1 | SELECT codTurma, nomecampus FROM historico WHERE nomeCurso LIKE 'MESTRADO EM QUÍMICA%' |
| 2 | SELECT discente, professor, disciplina, situacao FROM disciplinas, historico WHERE nomeDisciplina = 'FOTOJORNALISMO' and situacao <> 'APROVADO' and codDisciplina = disciplina |
| 3 | SELECT DISTINCT professor FROM historico |
| 4 | SELECT chave FROM historico WHERE professor = 1597 |
| 5 | SELECT situacao , count(*) FROM HISTORICO GROUP BY situacao |
| 6 | SELECT DISCENTE, PROFESSOR, SITUACAO FROM HISTORICO JOIN DISCIPLINAS ON CODDISCIPLINA = DISCIPLINA WHERE NOMEDISCIPLINA = 'TEORIA DOS JOGOS' |
| 7 | select * from historico where professor = 5962 and nomeCurso = 'ZOOTECNIA' AND SITUACAO = 'APROVADO' |
| 8 | SELECT * FROM HISTORICO WHERE SITUACAO = 'REPROVADO POR NOTA' AND disciplina IN (SELECT CODDISCIPLINA FROM DISCIPLINAS WHERE PercentualPratica > 50 AND CargaHorariaTotal > 200) |
| 9 | SELECT CHAVE, DISCENTE AS RESPONSAVEL FROM HISTORICO WHERE NOMECURSO = 'CULTURA VISUAL' UNION SELECT CHAVE, PROFESSOR AS RESPONSAVEL FROM HISTORICO WHERE NOMECURSO = 'EDUCAÇÃO MUSICAL - CANTO' |
| 10 | SELECT CHAVE, DISCENTE, SITUACAO FROM HISTORICO JOIN DISCIPLINAS ON CODDISCIPLINA = DISCIPLINA WHERE PERCENTUALPRATICA BETWEEN 90 AND 100 AND CODDISCIPLINA < 125 |
| 11 | SELECT DISCENTE, CODTURMA FROM historico WHERE (CODTURMA BETWEEN 46600 AND 46610 OR CODTURMA = 34763) AND SITUACAO = 'CANCELADA' |
| 12 | select nomeCurso, nomeCampus, avg(cargaHorariaTotal) from historico, disciplinas where coddisciplina = disciplina and professor in (133) and situacao = 'CANCELADA' group by nomeCurso |
| 13 | SELECT nomeCurso, sum(cargaHorariaTotal)/4 media_carga_ano FROM HISTORICO, DISCIPLINAS WHERE CODDISCIPLINA = DISCIPLINA AND nomeCurso = 'MÚSICA' GROUP BY nomeCurso |
| 14 | SELECT distinct nomeCurso from historico WHERE disciplina = 22 order by nomeCurso desc |
| 15 | SELECT discente, situacao from historico where DISCIPLINA = 22 AND nomeCampus <> 'CAMPUS DE CATALAO' AND situacao = 'APROVADO' AND discente not in (78661, 80836, 79782, 86615, 85339) |

APÊNDICE D

# Resultados dos Experimentos do Cenário 1

## D.1 Resultados por Experimento

**Tabela D.1:** *Experimentos Realizados no Cenário 1*

| | | | | Escore de Mutação | | | | |
|---|---|---|---|---|---|---|---|---|
| **Exp.** | **Instrução** | **Qtde BDTs** | **Qtde Tuplas** | **Maior** | **Menor** | **Média** | **Conjunto** | **Desvio Padrão** |
| 1 | 1 | 5 | 0,1% | 0,0000 | 0,0000 | 0,0000 | 0,0000 | 0,0000 |
| 2 | 1 | 10 | 0,1% | 0,0274 | 0,0274 | 0,0274 | 0,0274 | 0,0000 |
| 3 | 1 | 30 | 0,1% | 0,1233 | 0,0274 | 0,0754 | 0,1233 | 0,0480 |
| 4 | 1 | 5 | 1% | 0,0274 | 0,0274 | 0,0274 | 0,0548 | 0,0000 |
| 5 | 1 | 10 | 1% | 0,1233 | 0,0274 | 0,0462 | 0,1233 | 0,0313 |
| 6 | 1 | 30 | 1% | 0,1233 | 0,0274 | 0,0397 | 0,1918 | 0,0277 |
| 7 | 1 | 5 | 2% | 0,1233 | 0,0274 | 0,0630 | 0,1233 | 0,0320 |
| 8 | 1 | 10 | 2% | 0,8356 | 0,0274 | 0,1400 | 0,8767 | 0,2475 |
| 9 | 1 | 30 | 2% | 0,1233 | 0,0274 | 0,0564 | 0,2877 | 0,0380 |
| 10 | 1 | 5 | 3% | 0,1918 | 0,0274 | 0,0986 | 0,2192 | 0,0633 |
| 11 | 1 | 10 | 3% | 0,0548 | 0,0274 | 0,0466 | 0,0548 | 0,0126 |
| 12 | 1 | 30 | 3% | 0,8356 | 0,0274 | 0,0904 | 0,9452 | 0,1442 |
| 13 | 1 | 5 | 5% | 0,1233 | 0,0274 | 0,0630 | 0,1233 | 0,0320 |
| 14 | 1 | 10 | 5% | 0,8630 | 0,0548 | 0,3164 | 0,9452 | 0,3426 |
| 15 | 1 | 30 | 5% | 0,9041 | 0,0274 | 0,1630 | 0,9863 | 0,2384 |
| 16 | 1 | 5 | 7% | 0,8356 | 0,0548 | 0,2219 | 0,8767 | 0,3076 |
| 17 | 1 | 10 | 7% | 0,8767 | 0,0274 | 0,1644 | 0,9452 | 0,2427 |
| 18 | 1 | 30 | 7% | 0,9041 | 0,0548 | 0,3018 | 0,9863 | 0,3286 |
| 19 | 1 | 5 | 8% | 0,8630 | 0,0548 | 0,4110 | 0,9178 | 0,3552 |
| 20 | 1 | 10 | 8% | 0,8219 | 0,0548 | 0,2438 | 0,9863 | 0,2898 |
| 21 | 1 | 30 | 8% | 0,9041 | 0,0274 | 0,2247 | 1,0000 | 0,2836 |
| 22 | 1 | 5 | 9% | 0,1233 | 0,0548 | 0,0822 | 0,1507 | 0,0336 |
| 23 | 1 | 10 | 9% | 0,8356 | 0,0548 | 0,2452 | 0,9726 | 0,2983 |
| 24 | 1 | 30 | 9% | 0,8767 | 0,0274 | 0,3306 | 1,0000 | 0,3418 |
| 25 | 1 | 5 | 10% | 0,8356 | 0,0548 | 0,2521 | 0,9178 | 0,2962 |
| 26 | 1 | 10 | 10% | 0,9178 | 0,0548 | 0,2575 | 1,0000 | 0,3319 |
| 27 | 1 | 30 | 10% | 0,9315 | 0,0548 | 0,2667 | 1,0000 | 0,3143 |
| 28 | 2 | 5 | 0,1% | 0,9848 | 0,1515 | 0,3182 | 0,9848 | 0,3333 |
| 29 | 2 | 10 | 0,1% | 0,1515 | 0,1515 | 0,1515 | 0,1515 | 0,0000 |
| 30 | 2 | 30 | 0,1% | 0,9848 | 0,1515 | 0,2353 | 0,9848 | 0,2498 |
| 31 | 2 | 5 | 1% | 1,0000 | 0,9848 | 0,9878 | 1,0000 | 0,0061 |
| 32 | 2 | 10 | 1% | 0,9848 | 0,1515 | 0,6515 | 0,9848 | 0,4082 |
| 33 | 2 | 30 | 1% | 1,0000 | 0,1515 | 0,6540 | 1,0000 | 0,4103 |

| | | | | Escore de Mutação | | | | |
|---|---|---|---|---|---|---|---|---|
| **Exp.** | **Instrução** | **Qtde BDTs** | **Qtde Tuplas** | **Maior** | **Menor** | **Média** | **Conjunto** | **Desvio Padrão** |
| 34 | 2 | 5 | 2% | 1,0000 | 0,9848 | 0,9909 | 1,0000 | 0,0074 |
| 35 | 2 | 10 | 2% | 1,0000 | 0,1515 | 0,9030 | 1,0000 | 0,2505 |
| 36 | 2 | 30 | 2% | 1,0000 | 0,1515 | 0,7641 | 1,0000 | 0,3694 |
| 37 | 2 | 5 | 3% | 1,0000 | 0,9848 | 0,9909 | 1,0000 | 0,0074 |
| 38 | 2 | 10 | 3% | 1,0000 | 0,1515 | 0,9030 | 1,0000 | 0,2505 |
| 39 | 2 | 30 | 3% | 1,0000 | 0,1515 | 0,9363 | 1,0000 | 0,2099 |
| 40 | 3 | 5 | 0,1% | 0,1935 | 0,1774 | 0,1871 | 0,2258 | 0,0079 |
| 41 | 3 | 10 | 0,1% | 0,9677 | 0,1774 | 0,2580 | 0,9677 | 0,2366 |
| 42 | 3 | 30 | 0,1% | 0,9677 | 0,1774 | 0,2860 | 0,9677 | 0,2675 |
| 43 | 3 | 5 | 1% | 1,0000 | 1,0000 | 1,0000 | 1,0000 | 0,0000 |
| 44 | 3 | 10 | 1% | 1,0000 | 1,0000 | 1,0000 | 1,0000 | 0,0000 |
| 45 | 3 | 30 | 1% | 1,0000 | 0,9677 | 0,9989 | 1,0000 | 0,0058 |
| 46 | 3 | 5 | 2% | 1,0000 | 1,0000 | 1,0000 | 1,0000 | 0,0000 |
| 47 | 4 | 5 | 0,1% | 0,7143 | 0,7143 | 0,7143 | 0,7143 | 0,0000 |
| 48 | 4 | 10 | 0,1% | 0,7143 | 0,7143 | 0,7143 | 0,7143 | 0,0000 |
| 49 | 4 | 30 | 0,1% | 0,8571 | 0,7143 | 0,7191 | 0,8571 | 0,0256 |
| 50 | 4 | 5 | 1% | 1,0000 | 0,7143 | 0,8286 | 1,0000 | 0,1069 |
| 51 | 4 | 10 | 1% | 0,8571 | 0,7143 | 0,7714 | 1,0000 | 0,0700 |
| 52 | 4 | 30 | 1% | 1,0000 | 0,7143 | 0,7571 | 1,0000 | 0,0837 |
| 53 | 4 | 5 | 2% | 1,0000 | 0,8571 | 0,8857 | 1,0000 | 0,0572 |
| 54 | 4 | 10 | 2% | 1,0000 | 0,7143 | 0,8571 | 1,0000 | 0,0639 |
| 55 | 4 | 30 | 2% | 1,0000 | 0,7143 | 0,8476 | 1,0000 | 0,0731 |
| 56 | 4 | 5 | 3% | 0,8571 | 0,8571 | 0,8571 | 0,8571 | 0,0000 |
| 57 | 4 | 10 | 3% | 1,0000 | 0,8571 | 0,9000 | 1,0000 | 0,0655 |
| 58 | 4 | 30 | 3% | 1,0000 | 0,7143 | 0,9000 | 1,0000 | 0,0837 |
| 59 | 4 | 5 | 4000 | 1,0000 | 0,8571 | 0,9143 | 1,0000 | 0,0700 |
| 60 | 4 | 10 | 4000 | 1,0000 | 0,8571 | 0,9286 | 1,0000 | 0,0715 |
| 61 | 4 | 30 | 4000 | 1,0000 | 0,7143 | 0,9000 | 1,0000 | 0,0752 |
| 62 | 5 | 5 | 0,1% | 0,1515 | 0,1515 | 0,1515 | 0,1515 | 0,0000 |
| 63 | 5 | 10 | 0,1% | 0,1515 | 0,1515 | 0,1515 | 0,1515 | 0,0000 |
| 64 | 5 | 30 | 0,1% | 0,1515 | 0,1515 | 0,1515 | 0,1515 | 0,0000 |
| 65 | 5 | 5 | 1% | 0,9394 | 0,1515 | 0,4667 | 0,9394 | 0,3860 |
| 66 | 5 | 10 | 1% | 0,9394 | 0,1515 | 0,2364 | 0,9697 | 0,2350 |
| 67 | 5 | 30 | 1% | 0,9394 | 0,1515 | 0,4222 | 0,9697 | 0,3662 |
| 68 | 5 | 5 | 2% | 0,9394 | 0,1515 | 0,7818 | 0,9394 | 0,3152 |
| 69 | 5 | 10 | 2% | 0,9394 | 0,1515 | 0,6303 | 0,9697 | 0,3789 |
| 70 | 5 | 30 | 2% | 0,9697 | 0,1515 | 0,4717 | 0,9697 | 0,3842 |
| 71 | 5 | 5 | 3% | 0,9394 | 0,9394 | 0,9394 | 0,9394 | 0,0000 |
| 72 | 5 | 10 | 3% | 0,9697 | 0,1515 | 0,5545 | 0,9697 | 0,3914 |
| 73 | 5 | 30 | 3% | 0,9697 | 0,1515 | 0,6091 | 0,9697 | 0,3884 |
| 74 | 5 | 5 | 5% | 0,9697 | 0,9394 | 0,9515 | 0,9697 | 0,0148 |
| 75 | 5 | 10 | 5% | 0,9697 | 0,9394 | 0,9485 | 0,9697 | 0,0139 |
| 76 | 5 | 30 | 5% | 0,9697 | 0,1515 | 0,8212 | 0,9697 | 0,2892 |
| 77 | 5 | 5 | 7% | 0,9697 | 0,9394 | 0,9455 | 0,9697 | 0,0121 |
| 78 | 5 | 10 | 7% | 0,9697 | 0,1515 | 0,7970 | 0,9697 | 0,3230 |
| 79 | 5 | 30 | 7% | 0,9697 | 0,1515 | 0,9303 | 1,0000 | 0,1454 |
| 80 | 6 | 5 | 0,1% | 1,0000 | 1,0000 | 1,0000 | 1,0000 | 0,0000 |
| 81 | 6 | 10 | 0,1% | 1,0000 | 1,0000 | 1,0000 | 1,0000 | 0,0000 |
| 82 | 6 | 30 | 0,1% | 1,0000 | 1,0000 | 1,0000 | 1,0000 | 0,0000 |
| 83 | 7 | 5 | 0,1% | 0,2143 | 0,2143 | 0,2143 | 0,2143 | 0,0000 |
| 84 | 7 | 10 | 0,1% | 0,2143 | 0,2143 | 0,2143 | 0,2143 | 0,0000 |
| 85 | 7 | 30 | 0,1% | 0,8571 | 0,2143 | 0,2786 | 0,8571 | 0,1928 |
| 86 | 7 | 5 | 1% | 0,9714 | 0,8571 | 0,8800 | 0,9714 | 0,0457 |
| 87 | 7 | 10 | 1% | 0,9714 | 0,2143 | 0,5586 | 0,9714 | 0,3465 |

| | | | | Escore de Mutação | | | | |
|---|---|---|---|---|---|---|---|---|
| **Exp.** | **Instrução** | **Qtde BDTs** | **Qtde Tuplas** | **Maior** | **Menor** | **Média** | **Conjunto** | **Desvio Padrão** |
| 88 | 7 | 30 | 1% | 1,0000 | 0,2143 | 0,5962 | 1,0000 | 0,3596 |
| 89 | 7 | 5 | 2% | 0,9714 | 0,2143 | 0,6228 | 0,9714 | 0,3362 |
| 90 | 7 | 10 | 2% | 0,9714 | 0,2143 | 0,7100 | 0,9714 | 0,3279 |
| 91 | 7 | 30 | 2% | 0,9714 | 0,2143 | 0,7843 | 0,9714 | 0,2598 |
| 92 | 7 | 5 | 3% | 0,9714 | 0,8571 | 0,9485 | 0,9714 | 0,0457 |
| 93 | 7 | 10 | 3% | 0,9714 | 0,2143 | 0,8728 | 0,9714 | 0,2241 |
| 94 | 7 | 30 | 3% | 1,0000 | 0,2143 | 0,8700 | 1,0000 | 0,2238 |
| 95 | 7 | 5 | 5% | 0,9714 | 0,9714 | 0,9714 | 0,9714 | 0,0000 |
| 96 | 7 | 10 | 5% | 0,9714 | 0,8571 | 0,9600 | 0,9714 | 0,0343 |
| 97 | 7 | 30 | 5% | 1,0000 | 0,2143 | 0,9471 | 1,0000 | 0,1362 |
| 98 | 7 | 5 | 7% | 0,9714 | 0,9714 | 0,9714 | 0,9714 | 0,0000 |
| 99 | 7 | 10 | 7% | 1,0000 | 0,9714 | 0,9743 | 1,0000 | 0,0086 |
| 100 | 7 | 30 | 7% | 0,9714 | 0,9714 | 0,9714 | 0,9714 | 0,0000 |
| 101 | 8 | 5 | 0,1% | 0,4464 | 0,4107 | 0,4178 | 0,4464 | 0,0143 |
| 102 | 8 | 10 | 0,1% | 0,8750 | 0,4107 | 0,4714 | 0,9286 | 0,1356 |
| 103 | 8 | 30 | 0,1% | 0,8750 | 0,4107 | 0,4357 | 0,9286 | 0,0831 |
| 104 | 8 | 5 | 1% | 0,9643 | 0,4464 | 0,7250 | 0,9643 | 0,2294 |
| 105 | 8 | 10 | 1% | 0,9464 | 0,3214 | 0,6625 | 0,9643 | 0,2636 |
| 106 | 8 | 30 | 1% | 0,9643 | 0,4464 | 0,7077 | 1,0000 | 0,2270 |
| 107 | 8 | 5 | 2% | 0,9821 | 0,9286 | 0,9464 | 0,9821 | 0,0226 |
| 108 | 8 | 10 | 2% | 0,9643 | 0,4643 | 0,8804 | 1,0000 | 0,1399 |
| 109 | 8 | 30 | 2% | 0,9464 | 0,4464 | 0,8363 | 1,0000 | 0,1821 |
| 110 | 8 | 5 | 3% | 0,9643 | 0,9107 | 0,9214 | 0,9821 | 0,0214 |
| 111 | 8 | 10 | 3% | 0,9643 | 0,8929 | 0,9232 | 0,9643 | 0,0196 |
| 112 | 8 | 30 | 3% | 0,9643 | 0,3214 | 0,8494 | 1,0000 | 0,1772 |
| 113 | 8 | 5 | 5% | 1,0000 | 0,9107 | 0,9428 | 1,0000 | 0,0327 |
| 114 | 8 | 10 | 5% | 0,9464 | 0,9286 | 0,9357 | 0,9643 | 0,0087 |
| 115 | 8 | 30 | 5% | 0,9821 | 0,9107 | 0,9482 | 1,0000 | 0,0203 |
| 116 | 8 | 5 | 7% | 0,9464 | 0,9107 | 0,9321 | 0,9464 | 0,0134 |
| 117 | 8 | 10 | 7% | 0,9821 | 0,9464 | 0,9607 | 1,0000 | 0,0134 |
| 118 | 8 | 30 | 7% | 1,0000 | 0,9286 | 0,9548 | 1,0000 | 0,0194 |
| 119 | 9 | 5 | 0,1% | 1,0000 | 1,0000 | 1,0000 | 1,0000 | 0,0000 |
| 120 | 9 | 10 | 0,1% | 1,0000 | 1,0000 | 1,0000 | 1,0000 | 0,0000 |
| 121 | 9 | 30 | 0,1% | 1,0000 | 1,0000 | 1,0000 | 1,0000 | 0,0000 |
| 122 | 10 | 5 | 0,1% | 0,0471 | 0,0118 | 0,0330 | 0,0471 | 0,0173 |
| 123 | 10 | 10 | 0,1% | 0,1765 | 0,0118 | 0,0647 | 0,2118 | 0,0547 |
| 124 | 10 | 30 | 0,1% | 0,5529 | 0,0118 | 0,0953 | 0,5588 | 0,1209 |
| 125 | 10 | 5 | 1% | 0,5529 | 0,0471 | 0,2259 | 0,5529 | 0,1710 |
| 126 | 10 | 10 | 1% | 0,1882 | 0,0471 | 0,1259 | 0,1882 | 0,0644 |
| 127 | 10 | 30 | 1% | 0,5529 | 0,0471 | 0,1698 | 0,5882 | 0,1044 |
| 128 | 10 | 5 | 2% | 0,9647 | 0,1765 | 0,3412 | 0,9647 | 0,3120 |
| 129 | 10 | 10 | 2% | 0,8941 | 0,1765 | 0,3765 | 0,9765 | 0,2990 |
| 130 | 10 | 30 | 2% | 0,7882 | 0,0471 | 0,2302 | 0,9176 | 0,1582 |
| 131 | 10 | 5 | 3% | 0,5882 | 0,1765 | 0,3412 | 0,5882 | 0,1881 |
| 132 | 10 | 10 | 3% | 0,2118 | 0,1765 | 0,1836 | 0,2471 | 0,0141 |
| 133 | 10 | 30 | 3% | 0,8941 | 0,1765 | 0,3314 | 0,9765 | 0,2368 |
| 134 | 10 | 5 | 5% | 0,5529 | 0,1765 | 0,3482 | 0,6118 | 0,1686 |
| 135 | 10 | 10 | 5% | 0,9412 | 0,1765 | 0,5177 | 0,9765 | 0,2928 |
| 136 | 10 | 30 | 5% | 0,8235 | 0,1765 | 0,3224 | 0,9647 | 0,2175 |
| 137 | 10 | 5 | 7% | 0,8353 | 0,2353 | 0,6565 | 0,9647 | 0,2291 |
| 138 | 10 | 10 | 7% | 0,9529 | 0,1765 | 0,4153 | 0,9765 | 0,2784 |
| 139 | 10 | 30 | 7% | 0,9647 | 0,1765 | 0,4145 | 0,9647 | 0,2624 |
| 140 | 10 | 5 | 8% | 0,9529 | 0,1765 | 0,5365 | 0,9765 | 0,3158 |
| 141 | 10 | 10 | 8% | 0,8235 | 0,1765 | 0,4953 | 0,9647 | 0,2560 |

| | | | | Escore de Mutação | | | | |
|---|---|---|---|---|---|---|---|---|
| **Exp.** | **Instrução** | **Qtde BDTs** | **Qtde Tuplas** | **Maior** | **Menor** | **Média** | **Conjunto** | **Desvio Padrão** |
| 142 | 10 | 30 | 8% | 0,9765 | 0,1765 | 0,5074 | 0,9882 | 0,2772 |
| 143 | 10 | 5 | 9% | 0,8235 | 0,5529 | 0,7153 | 0,9647 | 0,1279 |
| 144 | 10 | 10 | 9% | 0,8353 | 0,1765 | 0,4765 | 0,9882 | 0,2418 |
| 145 | 10 | 30 | 9% | 0,9765 | 0,1765 | 0,6063 | 1,0000 | 0,2700 |
| 146 | 10 | 5 | 10% | 0,5765 | 0,2000 | 0,3553 | 0,6118 | 0,1715 |
| 147 | 10 | 10 | 10% | 0,9647 | 0,2000 | 0,5647 | 0,9882 | 0,2667 |
| 148 | 10 | 30 | 10% | 0,9882 | 0,1765 | 0,4192 | 1,0000 | 0,2742 |
| 149 | 11 | 5 | 0,1% | 0,1781 | 0,1370 | 0,1452 | 0,1781 | 0,0164 |
| 150 | 11 | 10 | 0,1% | 0,5616 | 0,1370 | 0,1795 | 0,5616 | 0,1274 |
| 151 | 11 | 30 | 0,1% | 0,1370 | 0,1370 | 0,1370 | 0,1370 | 0,0000 |
| 152 | 11 | 5 | 1% | 0,5616 | 0,1370 | 0,2219 | 0,5616 | 0,1698 |
| 153 | 11 | 10 | 1% | 0,5616 | 0,1370 | 0,2685 | 0,5890 | 0,1923 |
| 154 | 11 | 30 | 1% | 0,9863 | 0,1370 | 0,3352 | 0,9863 | 0,2625 |
| 155 | 11 | 5 | 2% | 0,9863 | 0,1370 | 0,4767 | 0,9863 | 0,3178 |
| 156 | 11 | 10 | 2% | 0,9863 | 0,1370 | 0,4384 | 1,0000 | 0,3792 |
| 157 | 11 | 30 | 2% | 1,0000 | 0,1370 | 0,3845 | 1,0000 | 0,2340 |
| 158 | 11 | 5 | 3% | 0,5890 | 0,5616 | 0,5671 | 1,0000 | 0,0110 |
| 159 | 11 | 10 | 3% | 1,0000 | 0,1370 | 0,5726 | 1,0000 | 0,3222 |
| 160 | 11 | 30 | 3% | 1,0000 | 0,1370 | 0,6242 | 1,0000 | 0,2097 |
| 161 | 11 | 5 | 5% | 0,9863 | 0,1370 | 0,5616 | 0,9863 | 0,2686 |
| 162 | 11 | 10 | 5% | 1,0000 | 0,1370 | 0,7411 | 1,0000 | 0,2823 |
| 163 | 11 | 30 | 5% | 1,0000 | 0,1781 | 0,7287 | 1,0000 | 0,2493 |
| 164 | 11 | 5 | 7% | 0,9863 | 0,5616 | 0,7424 | 1,0000 | 0,1994 |
| 165 | 11 | 10 | 7% | 1,0000 | 0,5616 | 0,8699 | 1,0000 | 0,1901 |
| 166 | 11 | 30 | 7% | 1,0000 | 0,5616 | 0,8507 | 1,0000 | 0,2007 |
| 167 | 12 | 5 | 0,1% | 0,3438 | 0,1875 | 0,2188 | 0,3438 | 0,0625 |
| 168 | 12 | 10 | 0,1% | 0,3438 | 0,1875 | 0,2031 | 0,3438 | 0,0469 |
| 169 | 12 | 30 | 0,1% | 0,3438 | 0,1875 | 0,1979 | 0,3438 | 0,0390 |
| 170 | 12 | 5 | 1% | 0,4063 | 0,1875 | 0,3250 | 0,4063 | 0,0729 |
| 171 | 12 | 10 | 1% | 0,9688 | 0,1875 | 0,3500 | 1,0000 | 0,2218 |
| 172 | 12 | 30 | 1% | 1,0000 | 0,1875 | 0,3177 | 1,0000 | 0,1899 |
| 173 | 12 | 5 | 2% | 0,4063 | 0,1875 | 0,3375 | 0,4063 | 0,0801 |
| 174 | 12 | 10 | 2% | 0,9688 | 0,1875 | 0,3594 | 1,0000 | 0,2148 |
| 175 | 12 | 30 | 2% | 0,9688 | 0,1875 | 0,3136 | 1,0000 | 0,1465 |
| 176 | 12 | 5 | 3% | 0,4063 | 0,3438 | 0,3813 | 0,4063 | 0,0306 |
| 177 | 12 | 10 | 3% | 0,9688 | 0,3438 | 0,5063 | 1,0000 | 0,2325 |
| 178 | 12 | 30 | 3% | 1,0000 | 0,1875 | 0,4292 | 1,0000 | 0,2233 |
| 179 | 12 | 5 | 5% | 0,9688 | 0,4063 | 0,5188 | 1,0000 | 0,2250 |
| 180 | 12 | 10 | 5% | 1,0000 | 0,1875 | 0,4719 | 1,0000 | 0,2426 |
| 181 | 12 | 30 | 5% | 1,0000 | 0,3438 | 0,5125 | 1,0000 | 0,2449 |
| 182 | 12 | 5 | 7% | 0,4063 | 0,3438 | 0,3813 | 0,4063 | 0,0306 |
| 183 | 12 | 10 | 7% | 1,0000 | 0,3438 | 0,4532 | 1,0000 | 0,1839 |
| 184 | 12 | 30 | 7% | 1,0000 | 0,3438 | 0,6532 | 1,0000 | 0,2956 |
| 185 | 12 | 5 | 8% | 1,0000 | 0,3438 | 0,6313 | 1,0000 | 0,3019 |
| 186 | 12 | 10 | 8% | 1,0000 | 0,3438 | 0,7532 | 1,0000 | 0,2966 |
| 187 | 12 | 30 | 8% | 1,0000 | 0,3438 | 0,5594 | 1,0000 | 0,2638 |
| 188 | 12 | 5 | 9% | 1,0000 | 0,4063 | 0,7625 | 1,0000 | 0,2909 |
| 189 | 12 | 10 | 9% | 1,0000 | 0,4063 | 0,4657 | 1,0000 | 0,1781 |
| 190 | 12 | 30 | 9% | 1,0000 | 0,3438 | 0,5396 | 1,0000 | 0,2520 |
| 191 | 12 | 5 | 10% | 1,0000 | 0,4063 | 0,7625 | 1,0000 | 0,2909 |
| 192 | 12 | 10 | 10% | 1,0000 | 0,4063 | 0,6438 | 1,0000 | 0,2909 |
| 193 | 12 | 30 | 10% | 1,0000 | 0,4063 | 0,7021 | 1,0000 | 0,2959 |
| 194 | 13 | 5 | 0,1% | 0,8621 | 0,8276 | 0,8345 | 0,8966 | 0,0138 |
| 195 | 13 | 10 | 0,1% | 0,8276 | 0,2759 | 0,6586 | 0,8276 | 0,2436 |

| | | | | Escore de Mutação | | | | |
|---|---|---|---|---|---|---|---|---|
| **Exp.** | **Instrução** | **Qtde BDTs** | **Qtde Tuplas** | **Maior** | **Menor** | **Média** | **Conjunto** | **Desvio Padrão** |
| 196 | 13 | 30 | 0,1% | 0,8621 | 0,2759 | 0,7908 | 0,9310 | 0,1296 |
| 197 | 13 | 5 | 1% | 0,9310 | 0,8276 | 0,8759 | 0,9310 | 0,0352 |
| 198 | 13 | 10 | 1% | 0,8621 | 0,8276 | 0,8380 | 0,8966 | 0,0158 |
| 199 | 13 | 30 | 1% | 0,9310 | 0,8276 | 0,8483 | 0,9655 | 0,0245 |
| 200 | 13 | 5 | 2% | 0,8966 | 0,8276 | 0,8690 | 0,9655 | 0,0258 |
| 201 | 13 | 10 | 2% | 0,9655 | 0,8276 | 0,8621 | 0,9655 | 0,0436 |
| 202 | 13 | 30 | 2% | 0,9310 | 0,8276 | 0,8667 | 1,0000 | 0,0305 |
| 203 | 13 | 5 | 3% | 0,9655 | 0,8276 | 0,8759 | 0,9655 | 0,0516 |
| 204 | 13 | 10 | 3% | 0,9310 | 0,8276 | 0,8724 | 1,0000 | 0,0409 |
| 205 | 13 | 30 | 3% | 0,9655 | 0,8276 | 0,8759 | 1,0000 | 0,0352 |
| 206 | 13 | 5 | 5% | 0,9655 | 0,8276 | 0,8966 | 0,9655 | 0,0487 |
| 207 | 13 | 10 | 5% | 1,0000 | 0,8621 | 0,9035 | 1,0000 | 0,0457 |
| 208 | 13 | 30 | 5% | 0,9655 | 0,8276 | 0,8920 | 1,0000 | 0,0405 |
| 209 | 13 | 5 | 7% | 0,9655 | 0,8621 | 0,9035 | 0,9655 | 0,0507 |
| 210 | 13 | 10 | 7% | 0,9310 | 0,8621 | 0,8931 | 0,9655 | 0,0186 |
| 211 | 13 | 30 | 7% | 0,9655 | 0,8621 | 0,9115 | 0,9655 | 0,0363 |
| 212 | 13 | 5 | 8% | 0,9655 | 0,9310 | 0,9379 | 1,0000 | 0,0138 |
| 213 | 13 | 10 | 8% | 0,9655 | 0,8966 | 0,9345 | 1,0000 | 0,0241 |
| 214 | 13 | 30 | 8% | 1,0000 | 0,8276 | 0,9149 | 1,0000 | 0,0395 |
| 215 | 13 | 5 | 9% | 0,9655 | 0,8621 | 0,9241 | 0,9655 | 0,0402 |
| 216 | 13 | 10 | 9% | 0,9655 | 0,8966 | 0,9276 | 1,0000 | 0,0241 |
| 217 | 13 | 30 | 9% | 0,9655 | 0,8621 | 0,9184 | 0,9655 | 0,0259 |
| 218 | 13 | 5 | 10% | 0,9655 | 0,8966 | 0,9310 | 0,9655 | 0,0308 |
| 219 | 13 | 10 | 10% | 1,0000 | 0,8966 | 0,9379 | 1,0000 | 0,0338 |
| 220 | 13 | 30 | 10% | 0,9655 | 0,8621 | 0,9184 | 1,0000 | 0,0350 |
| 221 | 14 | 5 | 0,1% | 0,3333 | 0,3333 | 0,3333 | 0,3333 | 0,0000 |
| 222 | 14 | 10 | 0,1% | 0,3333 | 0,2222 | 0,2889 | 0,3333 | 0,0544 |
| 223 | 14 | 30 | 0,1% | 0,3333 | 0,2222 | 0,2963 | 0,3333 | 0,0524 |
| 224 | 14 | 5 | 1% | 0,3889 | 0,3333 | 0,3444 | 0,3889 | 0,0222 |
| 225 | 14 | 10 | 1% | 0,3889 | 0,3333 | 0,3444 | 0,3889 | 0,0222 |
| 226 | 14 | 30 | 1% | 0,3889 | 0,3333 | 0,3352 | 0,3889 | 0,0100 |
| 227 | 14 | 5 | 2% | 0,3333 | 0,3333 | 0,3333 | 0,3333 | 0,0000 |
| 228 | 14 | 10 | 2% | 0,3333 | 0,3333 | 0,3333 | 0,3333 | 0,0000 |
| 229 | 14 | 30 | 2% | 0,3889 | 0,3333 | 0,3352 | 0,3889 | 0,0100 |
| 230 | 14 | 5 | 3% | 0,3333 | 0,3333 | 0,3333 | 0,3333 | 0,0000 |
| 231 | 14 | 10 | 3% | 0,3889 | 0,3333 | 0,3444 | 0,4444 | 0,0222 |
| 232 | 14 | 30 | 3% | 0,3889 | 0,3333 | 0,3389 | 0,4444 | 0,0167 |
| 233 | 14 | 5 | 5% | 0,3333 | 0,3333 | 0,3333 | 0,3333 | 0,0000 |
| 234 | 14 | 10 | 5% | 0,3889 | 0,3333 | 0,3389 | 0,3889 | 0,0167 |
| 235 | 14 | 30 | 5% | 0,3889 | 0,3333 | 0,3407 | 0,4444 | 0,0189 |
| 236 | 14 | 5 | 7% | 0,3889 | 0,3333 | 0,3444 | 0,3889 | 0,0222 |
| 237 | 14 | 10 | 7% | 0,3889 | 0,3333 | 0,3667 | 0,5000 | 0,0272 |
| 238 | 14 | 30 | 7% | 1,0000 | 0,3333 | 0,4981 | 1,0000 | 0,2776 |
| 239 | 14 | 5 | 8% | 0,3889 | 0,3333 | 0,3444 | 0,3889 | 0,0222 |
| 240 | 14 | 10 | 8% | 0,3889 | 0,3333 | 0,3389 | 0,3889 | 0,0167 |
| 241 | 14 | 30 | 8% | 1,0000 | 0,3333 | 0,4167 | 1,0000 | 0,1960 |
| 242 | 14 | 5 | 9% | 1,0000 | 0,3333 | 0,4778 | 1,0000 | 0,2620 |
| 243 | 14 | 10 | 9% | 1,0000 | 0,3333 | 0,4055 | 1,0000 | 0,1988 |
| 244 | 14 | 30 | 9% | 0,4444 | 0,3333 | 0,3500 | 0,4444 | 0,0326 |
| 245 | 14 | 5 | 10% | 0,4444 | 0,3333 | 0,3889 | 0,5000 | 0,0351 |
| 246 | 14 | 10 | 10% | 0,3889 | 0,3333 | 0,3389 | 0,3889 | 0,0167 |
| 247 | 14 | 30 | 10% | 1,0000 | 0,3333 | 0,4574 | 1,0000 | 0,2450 |
| 248 | 15 | 5 | 0,1% | 0,7179 | 0,1538 | 0,3179 | 0,7179 | 0,2080 |
| 249 | 15 | 10 | 0,1% | 0,2821 | 0,1538 | 0,1795 | 0,2821 | 0,0513 |

| | | | | Escore de Mutação | | | | |
|---|---|---|---|---|---|---|---|---|
| **Exp.** | **Instrução** | **Qtde BDTs** | **Qtde Tuplas** | **Maior** | **Menor** | **Média** | **Conjunto** | **Desvio Padrão** |
| 250 | 15 | 30 | 0,1% | 0,2821 | 0,1538 | 0,2051 | 0,2821 | 0,0629 |
| 251 | 15 | 5 | 1% | 0,7436 | 0,2821 | 0,6359 | 0,9231 | 0,1772 |
| 252 | 15 | 10 | 1% | 0,7564 | 0,2821 | 0,4667 | 0,7821 | 0,2160 |
| 253 | 15 | 30 | 1% | 0,7436 | 0,2821 | 0,4363 | 0,9359 | 0,2067 |
| 254 | 15 | 5 | 2% | 0,7436 | 0,2821 | 0,4795 | 0,7821 | 0,2109 |
| 255 | 15 | 10 | 2% | 0,7436 | 0,2821 | 0,4667 | 0,7564 | 0,2106 |
| 256 | 15 | 30 | 2% | 0,8974 | 0,2821 | 0,5838 | 0,9872 | 0,2245 |
| 257 | 15 | 5 | 3% | 0,8718 | 0,7436 | 0,7769 | 0,9872 | 0,0484 |
| 258 | 15 | 10 | 3% | 0,9103 | 0,2821 | 0,6257 | 0,9487 | 0,2329 |
| 259 | 15 | 30 | 3% | 0,9487 | 0,2821 | 0,6244 | 0,9615 | 0,2522 |
| 260 | 15 | 5 | 5% | 0,9359 | 0,3333 | 0,6949 | 0,9615 | 0,1970 |
| 261 | 15 | 10 | 5% | 0,9231 | 0,3077 | 0,7128 | 0,9744 | 0,2122 |
| 262 | 15 | 30 | 5% | 0,9359 | 0,2949 | 0,7675 | 1,0000 | 0,1661 |
| 263 | 15 | 5 | 7% | 0,9615 | 0,7308 | 0,8769 | 0,9744 | 0,0825 |
| 264 | 15 | 10 | 7% | 0,9615 | 0,3205 | 0,8359 | 0,9615 | 0,1812 |
| 265 | 15 | 30 | 7% | 0,9487 | 0,7308 | 0,8547 | 1,0000 | 0,0819 |
| 266 | 15 | 5 | 8% | 0,9487 | 0,7949 | 0,8872 | 0,9872 | 0,0558 |
| 267 | 15 | 10 | 8% | 0,9359 | 0,7308 | 0,8539 | 0,9872 | 0,0879 |
| 268 | 15 | 30 | 8% | 0,9872 | 0,7308 | 0,8769 | 1,0000 | 0,0763 |
| 269 | 15 | 5 | 9% | 0,9487 | 0,7564 | 0,8744 | 0,9872 | 0,0863 |
| 270 | 15 | 10 | 9% | 0,9359 | 0,7308 | 0,8526 | 0,9872 | 0,0833 |
| 271 | 15 | 30 | 9% | 0,9615 | 0,3590 | 0,8372 | 1,0000 | 0,1482 |
| 272 | 15 | 5 | 10% | 0,9744 | 0,7564 | 0,8923 | 0,9872 | 0,0776 |
| 273 | 15 | 10 | 10% | 0,9744 | 0,8846 | 0,9346 | 1,0000 | 0,0278 |
| 274 | 15 | 30 | 10% | 0,9744 | 0,7436 | 0,9009 | 1,0000 | 0,0736 |
| 275 | 16 | 5 | 0,1% | 0,9111 | 0,2000 | 0,5511 | 0,9333 | 0,2250 |
| 276 | 16 | 10 | 0,1% | 0,5556 | 0,1778 | 0,4334 | 0,9333 | 0,1678 |
| 277 | 16 | 30 | 0,1% | 0,8667 | 0,1333 | 0,4141 | 0,9556 | 0,2060 |
| 278 | 16 | 5 | 1% | 1,0000 | 0,9556 | 0,9822 | 1,0000 | 0,0218 |
| 279 | 16 | 10 | 1% | 1,0000 | 0,6222 | 0,9311 | 1,0000 | 0,1059 |
| 280 | 16 | 30 | 1% | 1,0000 | 0,6222 | 0,9341 | 1,0000 | 0,1059 |
| 281 | 16 | 5 | 2% | 1,0000 | 0,9778 | 0,9867 | 1,0000 | 0,0109 |
| 282 | 16 | 10 | 2% | 1,0000 | 0,6222 | 0,9578 | 1,0000 | 0,1122 |
| 283 | 16 | 30 | 2% | 1,0000 | 0,9556 | 0,9933 | 1,0000 | 0,0117 |
| 284 | 16 | 5 | 3% | 1,0000 | 0,9778 | 0,9956 | 1,0000 | 0,0089 |
| 285 | 16 | 10 | 3% | 1,0000 | 1,0000 | 1,0000 | 1,0000 | 0,0000 |
| 286 | 16 | 30 | 3% | 1,0000 | 0,9778 | 0,9970 | 1,0000 | 0,0075 |
| 287 | 17 | 5 | 0,1% | 0,1379 | 0,0690 | 0,1103 | 0,1379 | 0,0338 |
| 288 | 17 | 10 | 0,1% | 0,1552 | 0,0690 | 0,1345 | 0,1552 | 0,0334 |
| 289 | 17 | 30 | 0,1% | 0,2759 | 0,0690 | 0,1506 | 0,2759 | 0,0481 |
| 290 | 17 | 5 | 1% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |
| 291 | 17 | 10 | 1% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |
| 292 | 17 | 30 | 1% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |
| 293 | 17 | 5 | 2% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |
| 294 | 17 | 10 | 2% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |
| 295 | 17 | 30 | 2% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |
| 296 | 17 | 5 | 3% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |
| 297 | 17 | 10 | 3% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |
| 298 | 17 | 30 | 3% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |
| 299 | 17 | 5 | 5% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |
| 300 | 17 | 10 | 5% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |
| 301 | 17 | 30 | 5% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |
| 302 | 17 | 5 | 7% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |
| 303 | 17 | 10 | 7% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |

| | | | | Escore de Mutação | | | | |
|---|---|---|---|---|---|---|---|---|
| **Exp.** | **Instrução** | **Qtde BDTs** | **Qtde Tuplas** | **Maior** | **Menor** | **Média** | **Conjunto** | **Desvio Padrão** |
| 304 | 17 | 30 | 7% | 0,4828 | 0,4483 | 0,4495 | 0,4828 | 0,0062 |
| 305 | 17 | 5 | 8% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |
| 306 | 17 | 10 | 8% | 0,4828 | 0,4483 | 0,4518 | 0,4828 | 0,0104 |
| 307 | 17 | 30 | 8% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |
| 308 | 17 | 5 | 9% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |
| 309 | 17 | 10 | 9% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |
| 310 | 17 | 30 | 9% | 0,4828 | 0,4483 | 0,4495 | 0,4828 | 0,0062 |
| 311 | 17 | 5 | 10% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |
| 312 | 17 | 10 | 10% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |
| 313 | 17 | 30 | 10% | 0,4483 | 0,4483 | 0,4483 | 0,4483 | 0,0000 |
| 314 | 18 | 5 | 0,1% | 0,1383 | 0,1277 | 0,1298 | 0,1383 | 0,0042 |
| 315 | 18 | 10 | 0,1% | 0,2021 | 0,1277 | 0,1458 | 0,2021 | 0,0285 |
| 316 | 18 | 30 | 0,1% | 0,1383 | 0,1277 | 0,1291 | 0,1383 | 0,0036 |
| 317 | 18 | 5 | 1% | 0,2021 | 0,1383 | 0,1766 | 0,2021 | 0,0313 |
| 318 | 18 | 10 | 1% | 0,2021 | 0,1383 | 0,1808 | 0,2660 | 0,0285 |
| 319 | 18 | 30 | 1% | 0,2021 | 0,1383 | 0,1667 | 0,2660 | 0,0307 |
| 320 | 18 | 5 | 2% | 0,2660 | 0,2021 | 0,2149 | 0,2660 | 0,0256 |
| 321 | 18 | 10 | 2% | 0,3191 | 0,2021 | 0,2287 | 0,3404 | 0,0378 |
| 322 | 18 | 30 | 2% | 0,3085 | 0,2021 | 0,2170 | 0,3511 | 0,0276 |
| 323 | 18 | 5 | 3% | 0,2660 | 0,2021 | 0,2319 | 0,3298 | 0,0256 |
| 324 | 18 | 10 | 3% | 0,3191 | 0,2021 | 0,2394 | 0,3404 | 0,0434 |
| 325 | 18 | 30 | 3% | 0,8511 | 0,2021 | 0,2415 | 0,9362 | 0,1172 |
| 326 | 18 | 5 | 5% | 0,2660 | 0,2021 | 0,2149 | 0,2660 | 0,0256 |
| 327 | 18 | 10 | 5% | 0,2660 | 0,2021 | 0,2213 | 0,3298 | 0,0242 |
| 328 | 18 | 30 | 5% | 0,3085 | 0,2021 | 0,2298 | 0,3511 | 0,0321 |
| 329 | 18 | 5 | 7% | 0,2660 | 0,2021 | 0,2319 | 0,3511 | 0,0217 |
| 330 | 18 | 10 | 7% | 0,3085 | 0,2021 | 0,2447 | 0,3298 | 0,0404 |
| 331 | 18 | 30 | 7% | 0,8617 | 0,2021 | 0,2770 | 0,9468 | 0,1146 |
| 332 | 18 | 5 | 8% | 0,3191 | 0,2021 | 0,2681 | 0,3723 | 0,0463 |
| 333 | 18 | 10 | 8% | 0,3298 | 0,2021 | 0,2766 | 0,3511 | 0,0395 |
| 334 | 18 | 30 | 8% | 0,8936 | 0,2021 | 0,2965 | 0,9468 | 0,1617 |
| 335 | 18 | 5 | 9% | 0,3191 | 0,2021 | 0,2489 | 0,3723 | 0,0439 |
| 336 | 18 | 10 | 9% | 0,3404 | 0,2021 | 0,2606 | 0,3936 | 0,0490 |
| 337 | 18 | 30 | 9% | 0,9149 | 0,2021 | 0,3167 | 0,9681 | 0,1643 |
| 338 | 18 | 5 | 10% | 0,3404 | 0,2234 | 0,2787 | 0,3936 | 0,0438 |
| 339 | 18 | 10 | 10% | 0,9043 | 0,2234 | 0,3553 | 0,9468 | 0,1865 |
| 340 | 18 | 30 | 10% | 0,9149 | 0,2021 | 0,4255 | 0,9574 | 0,2657 |
| 341 | 19 | 5 | 0,1% | 0,1250 | 0,1250 | 0,1250 | 0,1250 | 0,0000 |
| 342 | 19 | 30 | 0,1% | 0,1429 | 0,1250 | 0,1262 | 0,1429 | 0,0045 |
| 343 | 19 | 5 | 1% | 0,7143 | 0,2321 | 0,4286 | 0,9643 | 0,1589 |
| 344 | 19 | 10 | 1% | 0,9107 | 0,2321 | 0,3571 | 0,9643 | 0,2041 |
| 345 | 19 | 30 | 1% | 0,9643 | 0,1429 | 0,3292 | 0,9821 | 0,1615 |
| 346 | 19 | 5 | 2% | 0,9643 | 0,2321 | 0,6143 | 0,9643 | 0,2972 |
| 347 | 19 | 10 | 2% | 0,9643 | 0,2321 | 0,5464 | 0,9821 | 0,2109 |
| 348 | 19 | 30 | 2% | 0,9643 | 0,2321 | 0,5161 | 0,9821 | 0,2361 |
| 349 | 19 | 5 | 3% | 0,4643 | 0,2500 | 0,4036 | 0,4821 | 0,0842 |
| 350 | 19 | 10 | 3% | 0,9643 | 0,2500 | 0,5429 | 0,9643 | 0,2203 |
| 351 | 19 | 30 | 3% | 0,9643 | 0,2321 | 0,6173 | 0,9643 | 0,2614 |
| 352 | 19 | 5 | 5% | 0,9643 | 0,4643 | 0,5643 | 0,9643 | 0,2000 |
| 353 | 19 | 10 | 5% | 0,9643 | 0,4643 | 0,6214 | 0,9643 | 0,2246 |
| 354 | 19 | 30 | 5% | 0,9643 | 0,4464 | 0,6643 | 0,9643 | 0,2451 |
| 355 | 19 | 5 | 7% | 0,9643 | 0,4464 | 0,6643 | 0,9643 | 0,2452 |
| 356 | 19 | 10 | 7% | 0,9821 | 0,4643 | 0,7214 | 0,9821 | 0,2466 |
| 357 | 19 | 30 | 7% | 0,9643 | 0,4464 | 0,6857 | 0,9643 | 0,2438 |

| | | | | Escore de Mutação | | | | |
|---|---|---|---|---|---|---|---|---|
| **Exp.** | **Instrução** | **Qtde BDTs** | **Qtde Tuplas** | **Maior** | **Menor** | **Média** | **Conjunto** | **Desvio Padrão** |
| 358 | 19 | 5 | 8% | 0,9643 | 0,4821 | 0,8679 | 0,9643 | 0,1929 |
| 359 | 19 | 10 | 8% | 0,9821 | 0,4643 | 0,7768 | 1,0000 | 0,2373 |
| 360 | 19 | 30 | 8% | 0,9821 | 0,4643 | 0,8381 | 1,0000 | 0,2176 |
| 361 | 19 | 5 | 9% | 0,9643 | 0,4821 | 0,7714 | 0,9643 | 0,2362 |
| 362 | 19 | 10 | 9% | 0,9821 | 0,4643 | 0,7196 | 0,9821 | 0,2519 |
| 363 | 19 | 30 | 9% | 1,0000 | 0,4643 | 0,8405 | 1,0000 | 0,2191 |
| 364 | 19 | 5 | 10% | 0,9821 | 0,9643 | 0,9714 | 0,9821 | 0,0087 |
| 365 | 19 | 10 | 10% | 0,9821 | 0,9643 | 0,9679 | 0,9821 | 0,0071 |
| 366 | 19 | 30 | 10% | 0,9821 | 0,4643 | 0,8238 | 1,0000 | 0,2290 |
| 367 | 20 | 5 | 0,1% | 0,3256 | 0,0233 | 0,1675 | 0,3256 | 0,0958 |
| 368 | 20 | 10 | 0,1% | 0,7209 | 0,0233 | 0,1954 | 0,7442 | 0,1849 |
| 369 | 20 | 30 | 0,1% | 0,3256 | 0,0233 | 0,1659 | 0,4186 | 0,0665 |
| 370 | 20 | 5 | 1% | 0,3256 | 0,1628 | 0,1954 | 0,3256 | 0,0651 |
| 371 | 20 | 10 | 1% | 0,6977 | 0,1628 | 0,3186 | 0,8372 | 0,1482 |
| 372 | 20 | 30 | 1% | 0,8605 | 0,1628 | 0,3093 | 0,8605 | 0,1280 |
| 373 | 20 | 5 | 2% | 0,3721 | 0,3256 | 0,3442 | 0,4186 | 0,0228 |
| 374 | 20 | 10 | 2% | 0,8372 | 0,1628 | 0,3535 | 0,8605 | 0,1764 |
| 375 | 20 | 30 | 2% | 0,8605 | 0,1628 | 0,3620 | 1,0000 | 0,1410 |
| 376 | 20 | 5 | 3% | 0,8605 | 0,3721 | 0,6605 | 0,9302 | 0,2356 |
| 377 | 20 | 10 | 3% | 0,8605 | 0,3256 | 0,5023 | 0,9070 | 0,2028 |
| 378 | 20 | 30 | 3% | 0,8605 | 0,3256 | 0,4659 | 1,0000 | 0,2100 |
| 379 | 20 | 5 | 5% | 0,8372 | 0,2326 | 0,5163 | 0,8837 | 0,2566 |
| 380 | 20 | 10 | 5% | 0,8605 | 0,3256 | 0,5535 | 0,9070 | 0,2332 |
| 381 | 20 | 30 | 5% | 0,8605 | 0,3256 | 0,5116 | 1,0000 | 0,2219 |
| 382 | 20 | 5 | 7% | 0,8372 | 0,3256 | 0,5442 | 0,8605 | 0,2305 |
| 383 | 20 | 10 | 7% | 0,8837 | 0,3721 | 0,6721 | 0,8837 | 0,2315 |
| 384 | 20 | 30 | 7% | 0,9070 | 0,3256 | 0,5791 | 1,0000 | 0,2302 |
| 385 | 20 | 5 | 8% | 0,8605 | 0,3721 | 0,7582 | 0,8837 | 0,1932 |
| 386 | 20 | 10 | 8% | 0,8605 | 0,3256 | 0,5651 | 0,9535 | 0,2328 |
| 387 | 20 | 30 | 8% | 0,9535 | 0,3256 | 0,6581 | 1,0000 | 0,2444 |

## D.2 Gráficos de Evolução dos Escores de Mutação

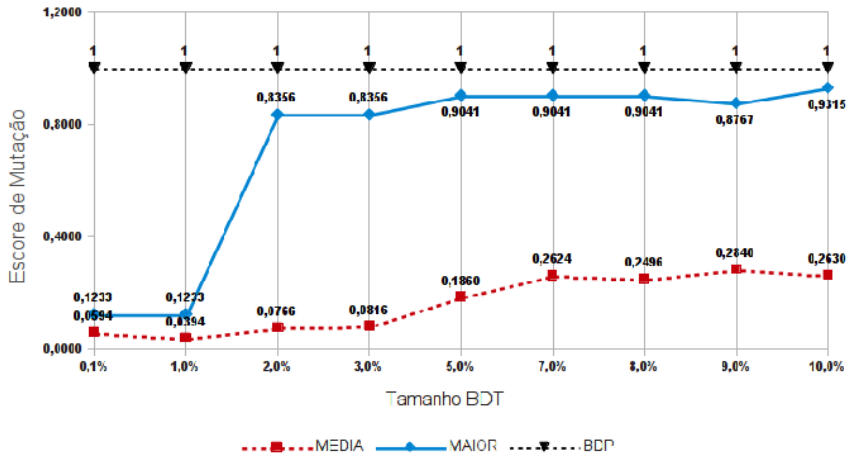


**Figura D.1:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 1*

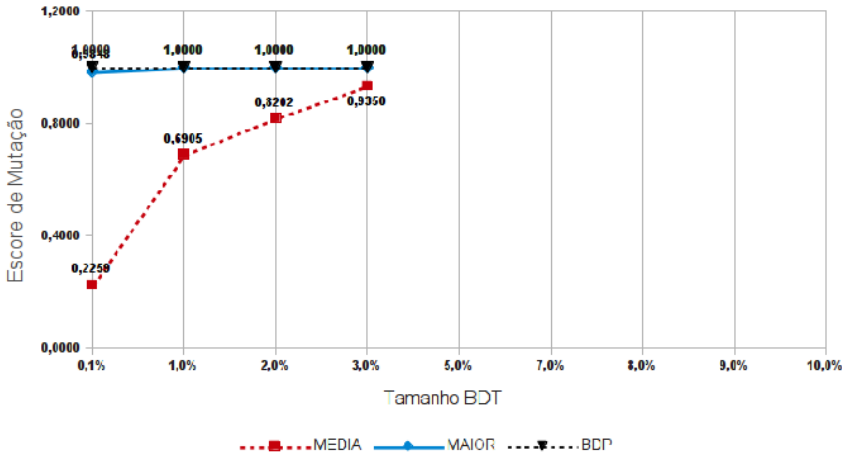


**Figura D.2:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 2*

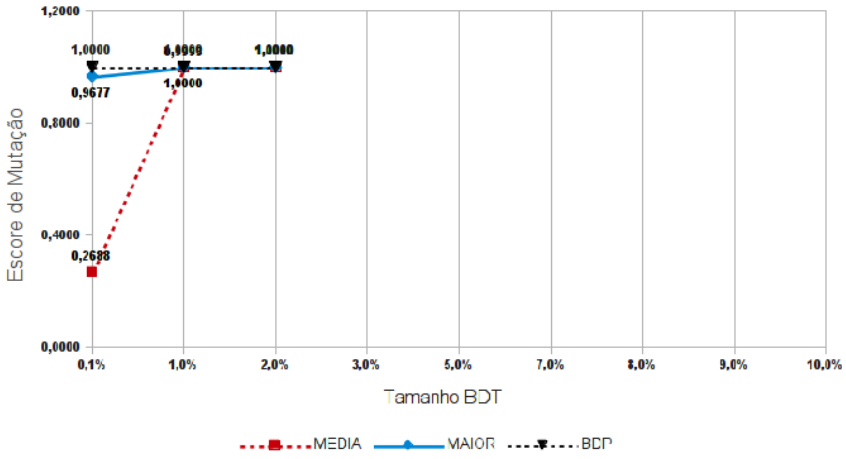


**Figura D.3:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 3*

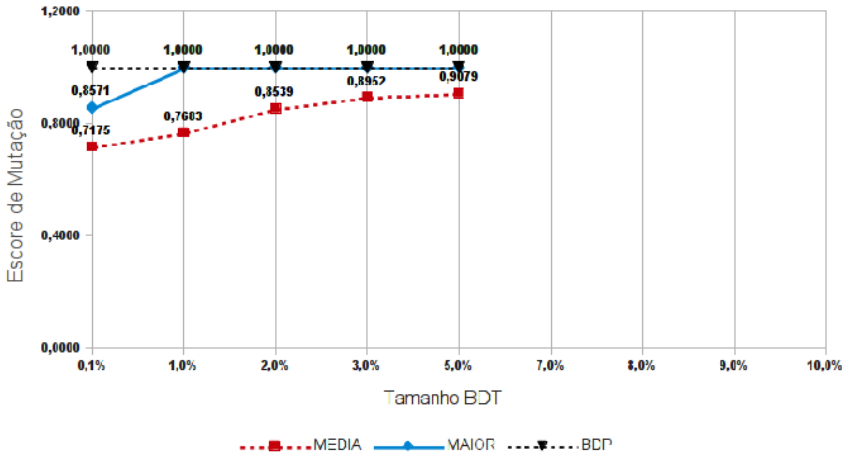


**Figura D.4:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 4*

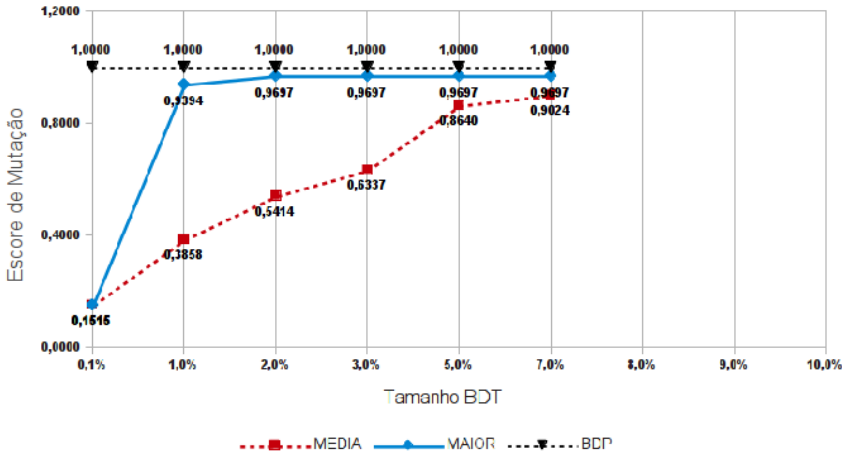


**Figura D.5:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 5*

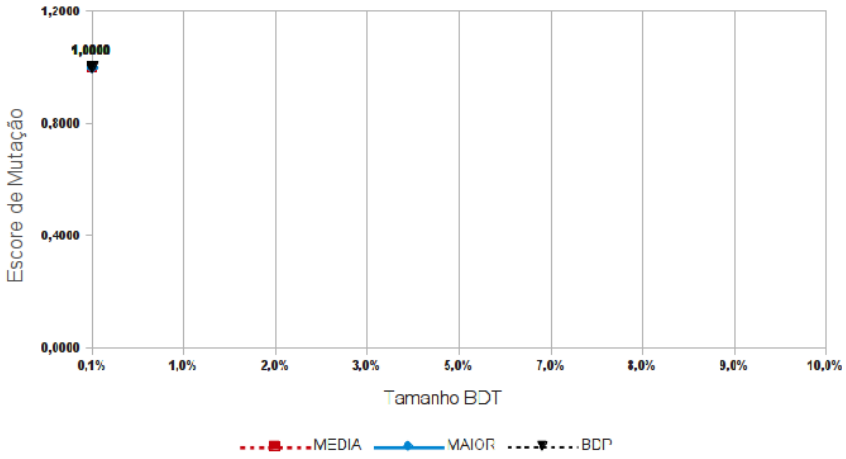


**Figura D.6:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 6*

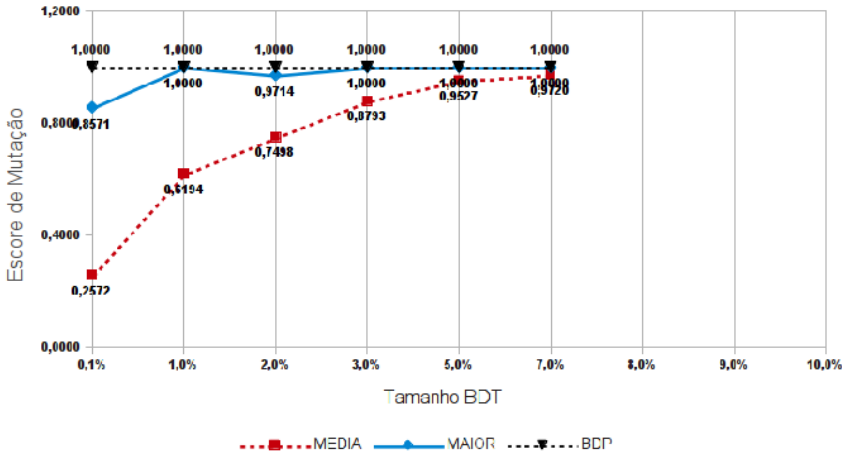


**Figura D.7:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 7*

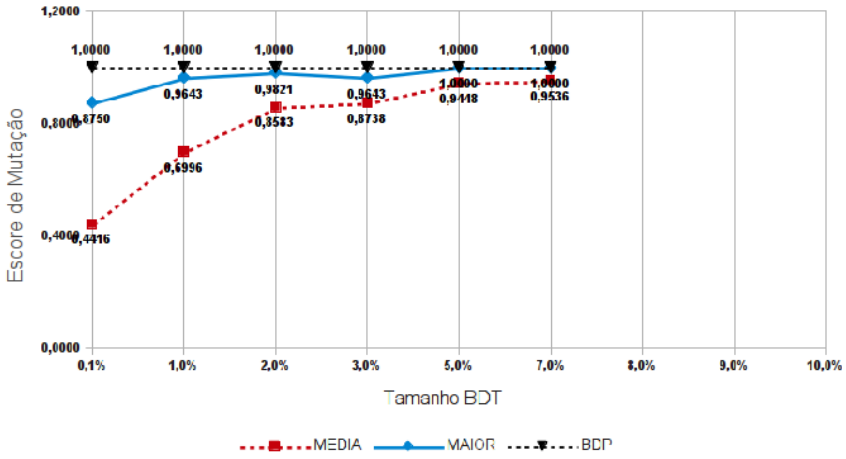


**Figura D.8:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 8*

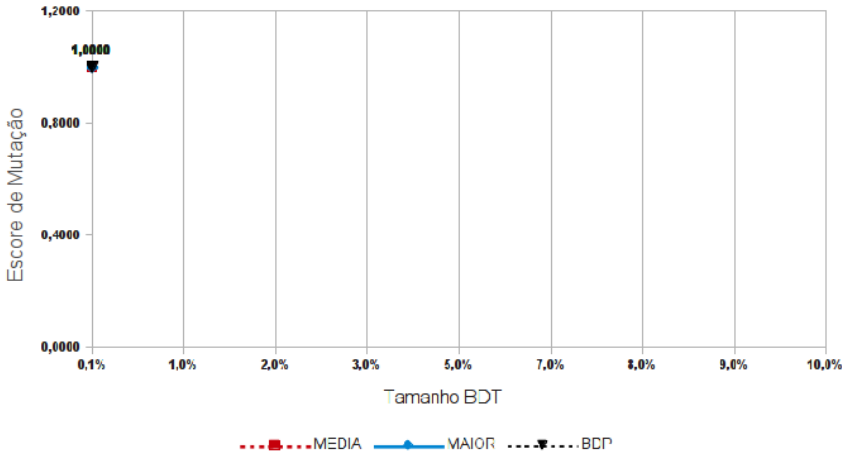


**Figura D.9:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 9*

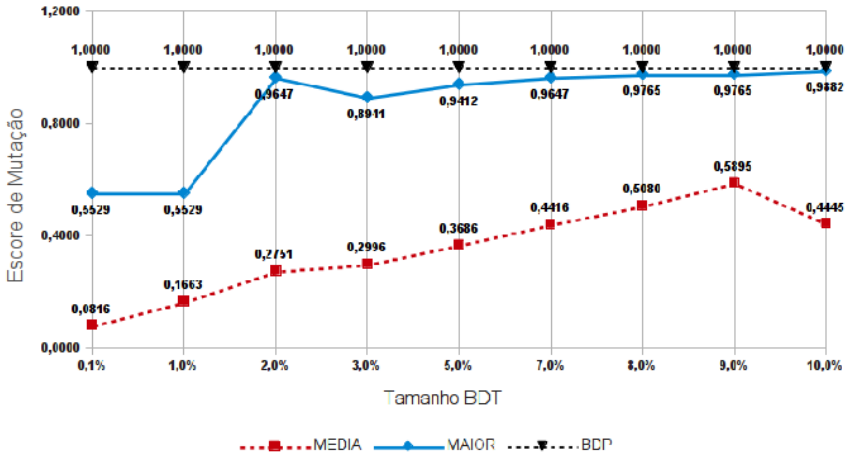


**Figura D.10:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 10*

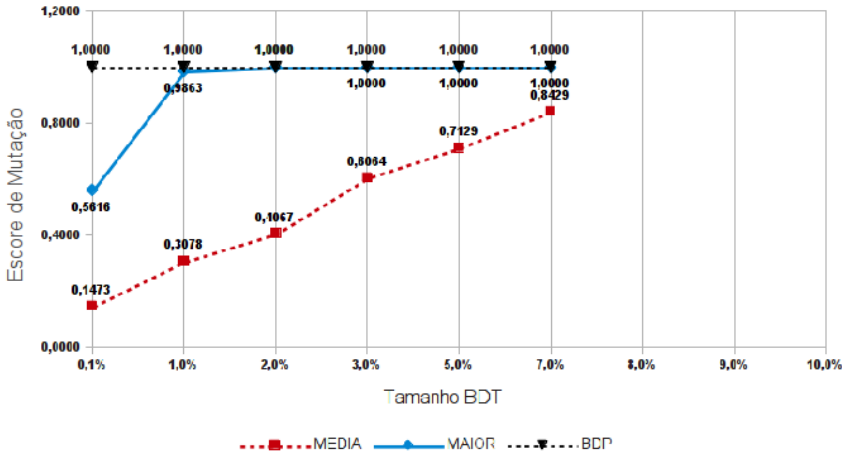


**Figura D.11:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 11*

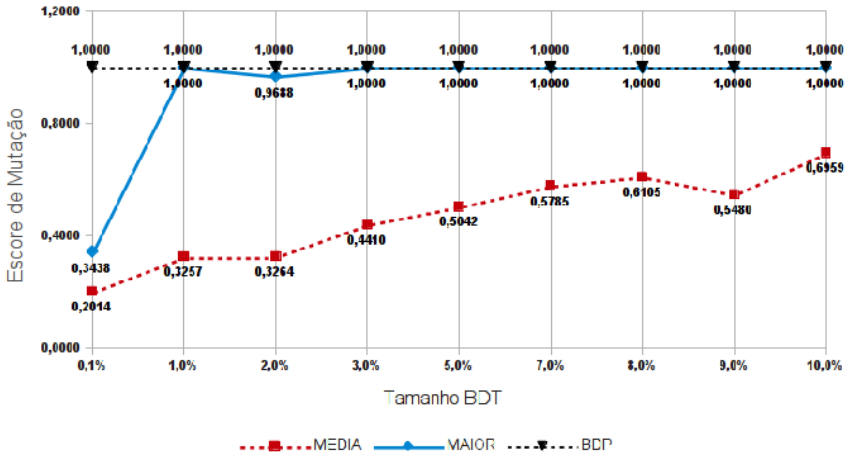


**Figura D.12:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 12*

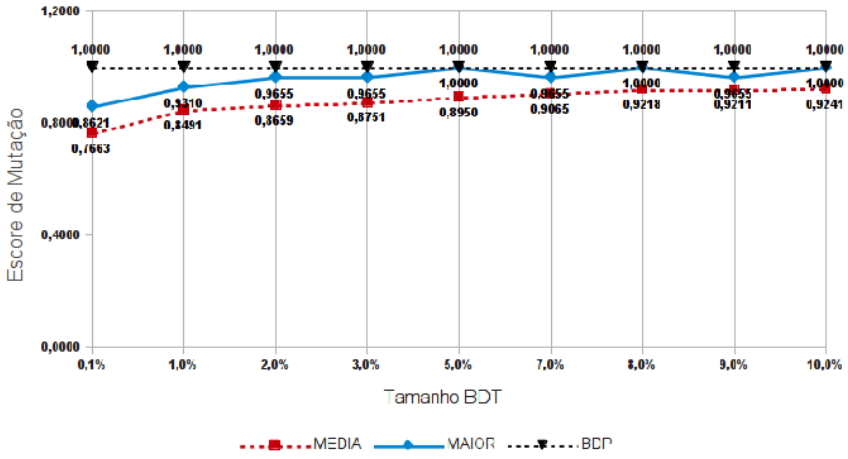


**Figura D.13:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 13*

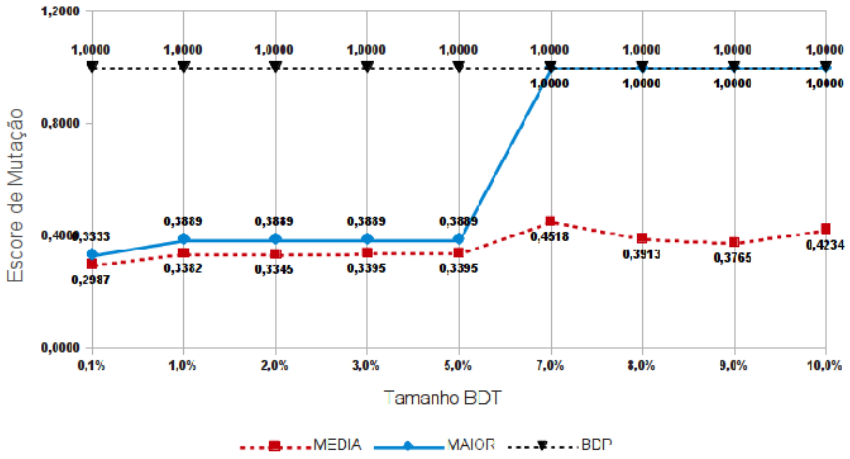


**Figura D.14:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 14*

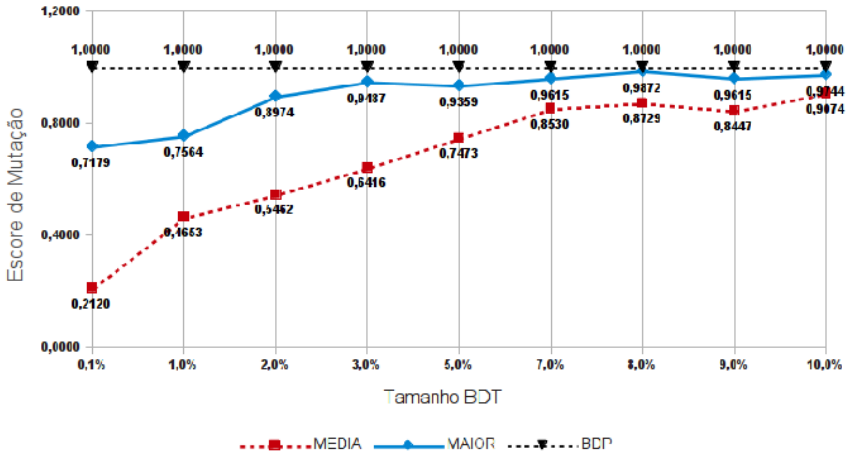


**Figura D.15:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 15*

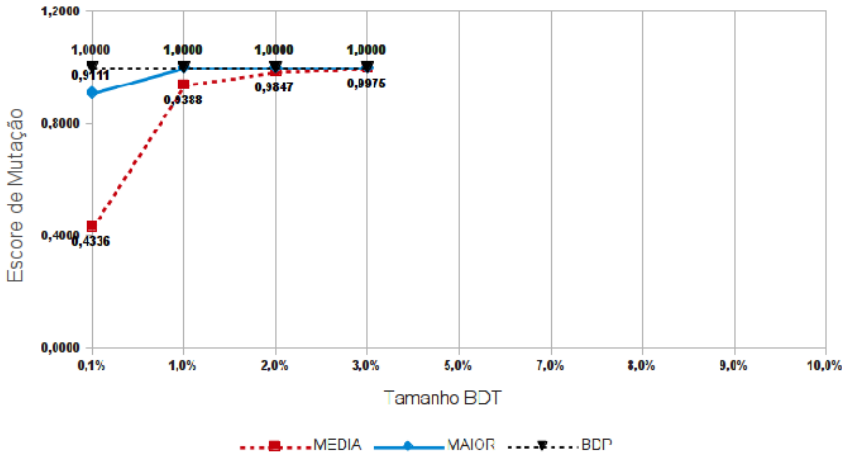


**Figura D.16:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 16*

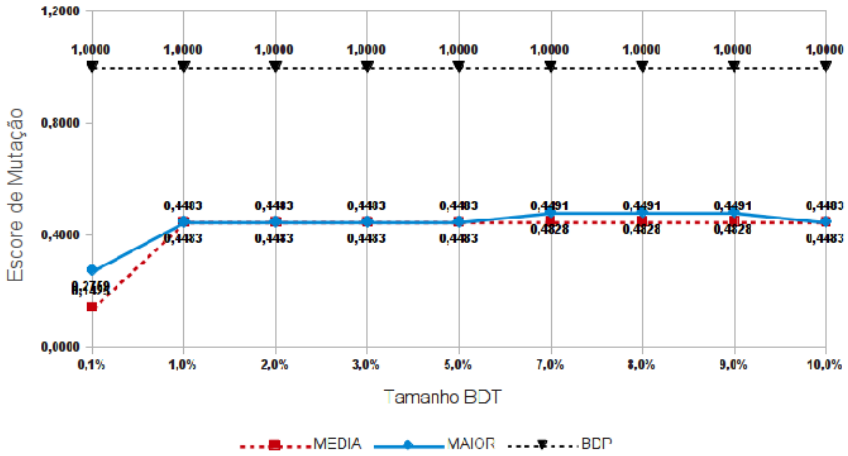


**Figura D.17:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 17*

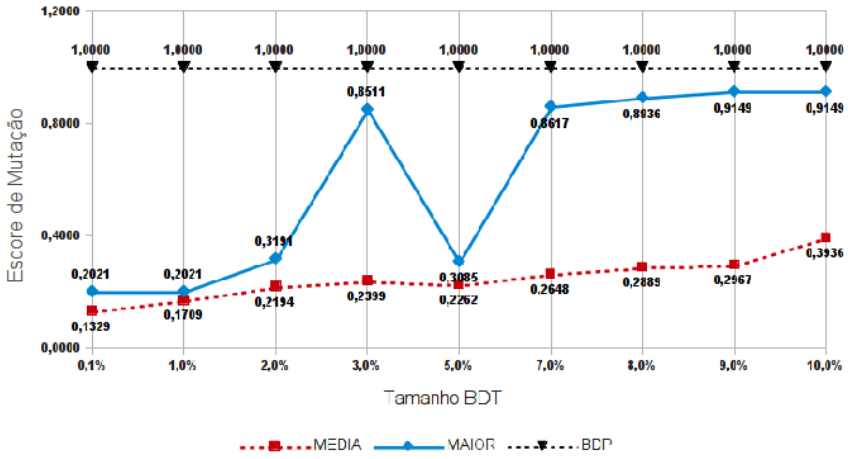


**Figura D.18:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 18*

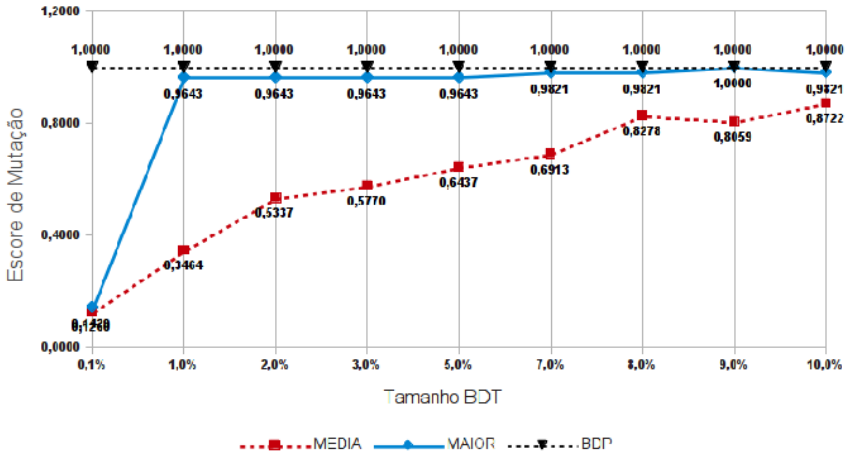


**Figura D.19:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 19*

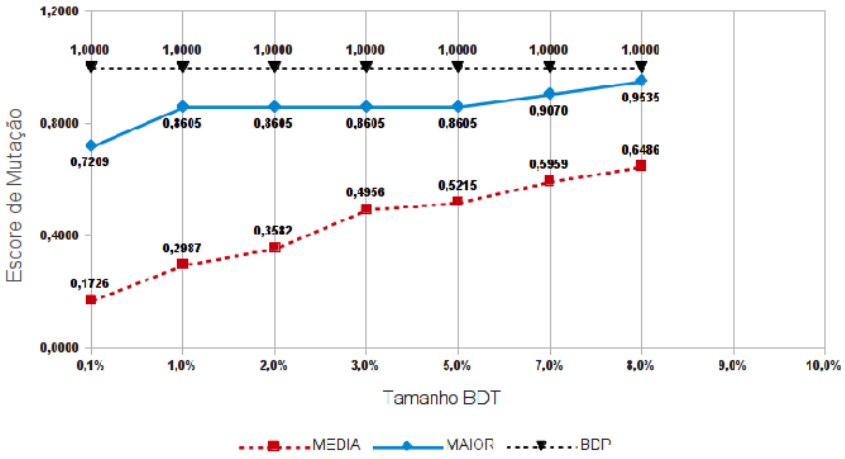


**Figura D.20:** *Cenário 1 - Evolução do Escore de Mutação da Instrução 20*

APÊNDICE E

# Resultados dos Experimentos do Cenário 2

## E.1 Resultados por Experimento

**Tabela E.1:** *Experimentos Realizados no Cenário 2*

| | | | | Escore de Mutação | | | | |
|---|---|---|---|---|---|---|---|---|
| **Exp.** | **Instrução** | **Qtde BDTs** | **Qtde Tuplas** | **Maior** | **Menor** | **Média** | **Conjunto** | **Desvio Padrão** |
| 1 | 1 | 5 | 0,1% | 0,1200 | 0,1200 | 0,1200 | 0,1200 | 0,0000 |
| 2 | 1 | 10 | 0,1% | 0,1200 | 0,1200 | 0,1200 | 0,1200 | 0,0000 |
| 3 | 1 | 30 | 0,1% | 0,1200 | 0,1200 | 0,1200 | 0,1200 | 0,0000 |
| 4 | 1 | 5 | 1% | 0,8400 | 0,1200 | 0,2640 | 0,8400 | 0,2880 |
| 5 | 1 | 10 | 1% | 0,8400 | 0,1200 | 0,2640 | 0,8400 | 0,2880 |
| 6 | 1 | 30 | 1% | 0,8400 | 0,1200 | 0,2640 | 0,8400 | 0,2880 |
| 7 | 1 | 5 | 2% | 0,8400 | 0,1200 | 0,4080 | 0,8400 | 0,3527 |
| 8 | 1 | 10 | 2% | 0,8400 | 0,1200 | 0,4080 | 0,8400 | 0,3527 |
| 9 | 1 | 30 | 2% | 0,8400 | 0,1200 | 0,4320 | 0,8400 | 0,3568 |
| 10 | 1 | 5 | 3% | 0,8400 | 0,1200 | 0,4080 | 0,8400 | 0,3527 |
| 11 | 1 | 10 | 3% | 0,8400 | 0,1200 | 0,4080 | 0,8400 | 0,3527 |
| 12 | 1 | 30 | 3% | 0,8400 | 0,1200 | 0,4800 | 0,8400 | 0,3600 |
| 13 | 1 | 5 | 5% | 0,8400 | 0,1200 | 0,5520 | 0,8400 | 0,3527 |
| 14 | 1 | 10 | 5% | 0,8800 | 0,1200 | 0,5600 | 0,8800 | 0,3596 |
| 15 | 1 | 30 | 5% | 0,8800 | 0,1200 | 0,6067 | 0,8800 | 0,3444 |
| 16 | 1 | 5 | 7% | 0,8800 | 0,8400 | 0,8640 | 0,8800 | 0,0196 |
| 17 | 1 | 10 | 7% | 0,8800 | 0,1200 | 0,7840 | 0,8800 | 0,2221 |
| 18 | 1 | 30 | 7% | 0,8800 | 0,1200 | 0,7533 | 0,8800 | 0,2490 |
| 19 | 2 | 5 | 0,1% | 0,0990 | 0,0990 | 0,0990 | 0,0990 | 0,0000 |
| 20 | 2 | 10 | 0,1% | 0,0990 | 0,0990 | 0,0990 | 0,0990 | 0,0000 |
| 21 | 2 | 30 | 0,1% | 0,2772 | 0,0990 | 0,1049 | 0,2772 | 0,0320 |
| 22 | 2 | 5 | 1% | 0,2772 | 0,0990 | 0,2059 | 0,2772 | 0,0873 |
| 23 | 2 | 10 | 1% | 0,9208 | 0,0990 | 0,2723 | 0,9208 | 0,2327 |
| 24 | 2 | 30 | 1% | 0,9208 | 0,0990 | 0,2647 | 0,9208 | 0,2563 |
| 25 | 2 | 5 | 2% | 0,8416 | 0,0990 | 0,3228 | 0,9208 | 0,2728 |
| 26 | 2 | 10 | 2% | 0,8416 | 0,0990 | 0,3723 | 0,9208 | 0,2405 |
| 27 | 2 | 30 | 2% | 0,9208 | 0,0990 | 0,2488 | 0,9208 | 0,1907 |
| 28 | 2 | 5 | 3% | 0,2970 | 0,0990 | 0,2455 | 0,2970 | 0,0737 |
| 29 | 2 | 10 | 3% | 0,9208 | 0,0990 | 0,3069 | 0,9208 | 0,2166 |
| 30 | 2 | 30 | 3% | 0,9208 | 0,0990 | 0,2993 | 0,9208 | 0,1682 |
| 31 | 2 | 5 | 5% | 0,9208 | 0,0990 | 0,5030 | 0,9208 | 0,3481 |
| 32 | 2 | 10 | 5% | 0,9208 | 0,2772 | 0,4851 | 0,9208 | 0,2856 |
| 33 | 2 | 30 | 5% | 0,9208 | 0,2772 | 0,4700 | 0,9208 | 0,2788 |

| | | | | Escore de Mutação | | | | |
|---|---|---|---|---|---|---|---|---|
| **Exp.** | **Instrução** | **Qtde BDTs** | **Qtde Tuplas** | **Maior** | **Menor** | **Média** | **Conjunto** | **Desvio Padrão** |
| 34 | 2 | 5 | 7% | 0,9208 | 0,2970 | 0,6713 | 0,9208 | 0,3056 |
| 35 | 2 | 10 | 7% | 0,9208 | 0,2772 | 0,5495 | 0,9208 | 0,3035 |
| 36 | 2 | 30 | 7% | 0,9208 | 0,2772 | 0,5871 | 0,9208 | 0,3122 |
| 37 | 3 | 5 | 0,1% | 0,9167 | 0,9167 | 0,9167 | 0,9167 | 0,0000 |
| 38 | 3 | 10 | 0,1% | 0,9167 | 0,9167 | 0,9167 | 0,9167 | 0,0000 |
| 39 | 3 | 30 | 0,1% | 0,9167 | 0,9167 | 0,9167 | 0,9167 | 0,0000 |
| 40 | 3 | 5 | 1% | 0,9167 | 0,9167 | 0,9167 | 0,9167 | 0,0000 |
| 41 | 3 | 10 | 1% | 0,9167 | 0,9167 | 0,9167 | 0,9167 | 0,0000 |
| 42 | 3 | 30 | 1% | 0,9167 | 0,9167 | 0,9167 | 0,9167 | 0,0000 |
| 55 | 4 | 5 | 0,1% | 0,2188 | 0,1875 | 0,1938 | 0,2188 | 0,0125 |
| 56 | 4 | 10 | 0,1% | 0,2188 | 0,1875 | 0,1906 | 0,2188 | 0,0094 |
| 57 | 4 | 30 | 0,1% | 0,2500 | 0,1875 | 0,1969 | 0,2813 | 0,0165 |
| 58 | 4 | 5 | 1% | 0,2813 | 0,2188 | 0,2438 | 0,2813 | 0,0234 |
| 59 | 4 | 10 | 1% | 0,2813 | 0,2188 | 0,2563 | 0,3125 | 0,0188 |
| 60 | 4 | 30 | 1% | 0,2813 | 0,1875 | 0,2552 | 0,3125 | 0,0230 |
| 61 | 4 | 5 | 2% | 0,2813 | 0,2188 | 0,2688 | 0,3125 | 0,0250 |
| 62 | 4 | 10 | 2% | 0,2813 | 0,2500 | 0,2688 | 0,2813 | 0,0153 |
| 63 | 4 | 30 | 2% | 0,2813 | 0,2188 | 0,2677 | 0,3125 | 0,0209 |
| 64 | 4 | 5 | 3% | 0,9063 | 0,2813 | 0,4063 | 0,9063 | 0,2500 |
| 65 | 4 | 10 | 3% | 0,2813 | 0,2500 | 0,2782 | 0,2813 | 0,0094 |
| 66 | 4 | 30 | 3% | 0,3125 | 0,2500 | 0,2803 | 0,3125 | 0,0098 |
| 67 | 4 | 5 | 5% | 0,9063 | 0,2813 | 0,4125 | 0,9063 | 0,2472 |
| 68 | 4 | 10 | 5% | 0,9063 | 0,2813 | 0,3500 | 0,9063 | 0,1858 |
| 69 | 4 | 30 | 5% | 0,9063 | 0,2500 | 0,3000 | 0,9063 | 0,1134 |
| 70 | 4 | 5 | 7% | 0,2813 | 0,2813 | 0,2813 | 0,2813 | 0,0000 |
| 71 | 4 | 10 | 7% | 0,9063 | 0,2500 | 0,3438 | 0,9063 | 0,1880 |
| 72 | 4 | 30 | 7% | 0,9063 | 0,2813 | 0,3250 | 0,9063 | 0,1555 |
| 73 | 5 | 5 | 0,1% | 1,0000 | 1,0000 | 1,0000 | 1,0000 | 0,0000 |
| 74 | 5 | 10 | 0,1% | 1,0000 | 1,0000 | 1,0000 | 1,0000 | 0,0000 |
| 75 | 5 | 30 | 0,1% | 1,0000 | 1,0000 | 1,0000 | 1,0000 | 0,0000 |
| 76 | 5 | 5 | 1% | 1,0000 | 1,0000 | 1,0000 | 1,0000 | 0,0000 |
| 77 | 5 | 10 | 1% | 1,0000 | 1,0000 | 1,0000 | 1,0000 | 0,0000 |
| 78 | 5 | 30 | 1% | 1,0000 | 1,0000 | 1,0000 | 1,0000 | 0,0000 |
| 91 | 6 | 5 | 0,1% | 0,1045 | 0,1045 | 0,1045 | 0,1045 | 0,0000 |
| 92 | 6 | 10 | 0,1% | 0,1045 | 0,1045 | 0,1045 | 0,1045 | 0,0000 |
| 93 | 6 | 30 | 0,1% | 0,9104 | 0,1045 | 0,1314 | 0,9104 | 0,1447 |
| 94 | 6 | 5 | 1% | 0,9104 | 0,1045 | 0,2657 | 0,9104 | 0,3224 |
| 95 | 6 | 10 | 1% | 0,9104 | 0,1045 | 0,1851 | 0,9104 | 0,2418 |
| 96 | 6 | 30 | 1% | 0,9104 | 0,1045 | 0,2657 | 0,9104 | 0,3224 |
| 97 | 6 | 5 | 2% | 0,9104 | 0,1045 | 0,2657 | 0,9104 | 0,3224 |
| 98 | 6 | 10 | 2% | 0,9104 | 0,1045 | 0,3463 | 0,9104 | 0,3693 |
| 99 | 6 | 30 | 2% | 0,9104 | 0,1045 | 0,3463 | 0,9104 | 0,3693 |
| 100 | 6 | 5 | 3% | 0,9104 | 0,1045 | 0,4269 | 0,9104 | 0,3948 |
| 101 | 6 | 10 | 3% | 0,9104 | 0,1045 | 0,5880 | 0,9104 | 0,3948 |
| 102 | 6 | 30 | 3% | 0,9104 | 0,1045 | 0,4806 | 0,9104 | 0,4021 |
| 103 | 6 | 5 | 5% | 0,9104 | 0,1045 | 0,7492 | 0,9104 | 0,3224 |
| 104 | 6 | 10 | 5% | 0,9104 | 0,1045 | 0,3493 | 0,9104 | 0,3674 |
| 105 | 6 | 30 | 5% | 0,9104 | 0,1045 | 0,5079 | 0,9104 | 0,4025 |
| 106 | 6 | 5 | 7% | 0,9104 | 0,1194 | 0,2895 | 0,9104 | 0,3109 |
| 107 | 6 | 10 | 7% | 0,9104 | 0,1194 | 0,7537 | 0,9104 | 0,3134 |
| 108 | 6 | 30 | 7% | 0,9104 | 0,1045 | 0,6214 | 0,9104 | 0,3799 |
| 109 | 7 | 5 | 0,1% | 0,1600 | 0,0400 | 0,1120 | 0,1600 | 0,0588 |
| 110 | 7 | 10 | 0,1% | 0,2800 | 0,0400 | 0,1600 | 0,2800 | 0,0537 |
| 111 | 7 | 30 | 0,1% | 0,7600 | 0,0400 | 0,1687 | 0,8400 | 0,1157 |

| | | | | Escore de Mutação | | | | |
|---|---|---|---|---|---|---|---|---|
| **Exp.** | **Instrução** | **Qtde BDTs** | **Qtde Tuplas** | **Maior** | **Menor** | **Média** | **Conjunto** | **Desvio Padrão** |
| 112 | 7 | 5 | 1% | 0,8400 | 0,1600 | 0,3200 | 0,8400 | 0,2641 |
| 113 | 7 | 10 | 1% | 0,7600 | 0,1600 | 0,2440 | 0,9000 | 0,1784 |
| 114 | 7 | 30 | 1% | 0,8400 | 0,1600 | 0,2867 | 0,9000 | 0,2242 |
| 115 | 7 | 5 | 2% | 0,8400 | 0,1600 | 0,5600 | 0,8400 | 0,2817 |
| 116 | 7 | 10 | 2% | 0,8400 | 0,1600 | 0,4560 | 0,8400 | 0,2690 |
| 117 | 7 | 30 | 2% | 0,9000 | 0,1600 | 0,4120 | 0,9000 | 0,3048 |
| 118 | 7 | 5 | 3% | 0,8400 | 0,2800 | 0,6160 | 0,9000 | 0,2743 |
| 119 | 7 | 10 | 3% | 0,9000 | 0,1600 | 0,5700 | 0,9000 | 0,2988 |
| 120 | 7 | 30 | 3% | 0,9000 | 0,1600 | 0,5507 | 0,9000 | 0,2726 |
| 121 | 7 | 5 | 5% | 0,8400 | 0,7600 | 0,8240 | 0,8400 | 0,0320 |
| 122 | 7 | 10 | 5% | 0,8400 | 0,2800 | 0,6100 | 0,9000 | 0,2522 |
| 123 | 7 | 30 | 5% | 0,9000 | 0,1600 | 0,6813 | 0,9000 | 0,2557 |
| 124 | 7 | 5 | 7% | 0,8400 | 0,3800 | 0,7320 | 0,9000 | 0,1787 |
| 125 | 7 | 10 | 7% | 0,9000 | 0,2800 | 0,6800 | 0,9000 | 0,2435 |
| 126 | 7 | 30 | 7% | 0,9000 | 0,7600 | 0,8587 | 0,9000 | 0,0390 |
| 127 | 8 | 5 | 0,1% | 0,3265 | 0,2143 | 0,2408 | 0,3469 | 0,0431 |
| 128 | 8 | 10 | 0,1% | 0,3265 | 0,1224 | 0,2612 | 0,3469 | 0,0789 |
| 129 | 8 | 30 | 0,1% | 0,3367 | 0,1224 | 0,2418 | 0,3571 | 0,0687 |
| 130 | 8 | 5 | 1% | 0,4082 | 0,3061 | 0,3490 | 0,4694 | 0,0338 |
| 131 | 8 | 10 | 1% | 0,8061 | 0,3265 | 0,3990 | 0,8776 | 0,1385 |
| 132 | 8 | 30 | 1% | 0,7959 | 0,3265 | 0,3506 | 0,8367 | 0,0834 |
| 133 | 8 | 5 | 2% | 0,3571 | 0,3265 | 0,3408 | 0,3571 | 0,0122 |
| 134 | 8 | 10 | 2% | 0,7959 | 0,3265 | 0,4418 | 0,8367 | 0,1782 |
| 135 | 8 | 30 | 2% | 0,8367 | 0,3265 | 0,3955 | 0,8367 | 0,1410 |
| 136 | 8 | 5 | 3% | 0,8163 | 0,3265 | 0,4347 | 0,8163 | 0,1911 |
| 137 | 8 | 10 | 3% | 0,8163 | 0,3367 | 0,4459 | 0,8367 | 0,1857 |
| 138 | 8 | 30 | 3% | 0,8163 | 0,3265 | 0,4619 | 0,8367 | 0,1945 |
| 139 | 8 | 5 | 5% | 0,3878 | 0,3571 | 0,3673 | 0,3878 | 0,0112 |
| 140 | 8 | 10 | 5% | 0,8163 | 0,3367 | 0,4490 | 0,8163 | 0,1842 |
| 141 | 8 | 30 | 5% | 0,8776 | 0,3367 | 0,4758 | 0,8980 | 0,1913 |
| 142 | 8 | 5 | 7% | 0,8163 | 0,3673 | 0,6367 | 0,8163 | 0,2118 |
| 143 | 8 | 10 | 7% | 0,8776 | 0,3571 | 0,6520 | 0,8980 | 0,2277 |
| 144 | 8 | 30 | 7% | 0,8367 | 0,3571 | 0,5020 | 0,8980 | 0,1933 |
| 145 | 9 | 5 | 0,1% | 0,1791 | 0,1791 | 0,1791 | 0,1791 | 0,0000 |
| 146 | 9 | 10 | 0,1% | 0,5672 | 0,1791 | 0,2179 | 0,5672 | 0,1164 |
| 147 | 9 | 30 | 0,1% | 0,1791 | 0,1791 | 0,1791 | 0,1791 | 0,0000 |
| 148 | 9 | 5 | 1% | 0,1791 | 0,1791 | 0,1791 | 0,1791 | 0,0000 |
| 149 | 9 | 10 | 1% | 0,9254 | 0,1791 | 0,2537 | 0,9254 | 0,2239 |
| 150 | 9 | 30 | 1% | 0,9254 | 0,1791 | 0,2418 | 0,9403 | 0,1955 |
| 151 | 9 | 5 | 2% | 0,5672 | 0,1791 | 0,2567 | 0,5672 | 0,1552 |
| 152 | 9 | 10 | 2% | 0,5672 | 0,1791 | 0,2567 | 0,5672 | 0,1552 |
| 153 | 9 | 30 | 2% | 0,9254 | 0,1791 | 0,2557 | 0,9403 | 0,1810 |
| 154 | 9 | 5 | 3% | 0,9403 | 0,1791 | 0,3313 | 0,9403 | 0,3045 |
| 155 | 9 | 10 | 3% | 0,9254 | 0,1791 | 0,2925 | 0,9403 | 0,2406 |
| 156 | 9 | 30 | 3% | 0,9254 | 0,1791 | 0,3045 | 0,9403 | 0,2619 |
| 157 | 9 | 5 | 5% | 0,5672 | 0,1791 | 0,2567 | 0,5672 | 0,1552 |
| 158 | 9 | 10 | 5% | 0,9403 | 0,1791 | 0,3687 | 0,9403 | 0,3046 |
| 159 | 9 | 30 | 5% | 0,9403 | 0,1791 | 0,3194 | 0,9403 | 0,2502 |
| 160 | 9 | 5 | 7% | 0,9254 | 0,1791 | 0,6329 | 0,9403 | 0,2777 |
| 161 | 9 | 10 | 7% | 0,9254 | 0,1791 | 0,4448 | 0,9403 | 0,2932 |
| 162 | 9 | 30 | 7% | 0,9403 | 0,1791 | 0,5189 | 0,9403 | 0,3407 |
| 163 | 10 | 5 | 0,1% | 0,1705 | 0,1163 | 0,1457 | 0,1783 | 0,0242 |
| 164 | 10 | 10 | 0,1% | 0,1705 | 0,1550 | 0,1597 | 0,1705 | 0,0052 |
| 165 | 10 | 30 | 0,1% | 0,1705 | 0,1163 | 0,1605 | 0,1783 | 0,0098 |

| | | | | Escore de Mutação | | | | |
|---|---|---|---|---|---|---|---|---|
| **Exp.** | **Instrução** | **Qtde BDTs** | **Qtde Tuplas** | **Maior** | **Menor** | **Média** | **Conjunto** | **Desvio Padrão** |
| 166 | 10 | 5 | 1% | 0,1860 | 0,1705 | 0,1767 | 0,1860 | 0,0058 |
| 167 | 10 | 10 | 1% | 0,8527 | 0,1705 | 0,2449 | 0,8527 | 0,2027 |
| 168 | 10 | 30 | 1% | 0,8527 | 0,1628 | 0,2447 | 0,8527 | 0,2028 |
| 169 | 10 | 5 | 2% | 0,8682 | 0,1783 | 0,4481 | 0,8682 | 0,3275 |
| 170 | 10 | 10 | 2% | 0,8682 | 0,1783 | 0,3845 | 0,8682 | 0,3099 |
| 171 | 10 | 30 | 2% | 0,8527 | 0,1705 | 0,3597 | 0,8527 | 0,2973 |
| 172 | 10 | 5 | 3% | 0,8527 | 0,1783 | 0,5829 | 0,8527 | 0,3304 |
| 173 | 10 | 10 | 3% | 0,8527 | 0,1783 | 0,3837 | 0,8527 | 0,3070 |
| 174 | 10 | 30 | 3% | 0,8527 | 0,1783 | 0,4499 | 0,8527 | 0,3289 |
| 175 | 10 | 5 | 5% | 0,8527 | 0,1860 | 0,4527 | 0,8527 | 0,3266 |
| 176 | 10 | 10 | 5% | 0,8682 | 0,1783 | 0,5860 | 0,8682 | 0,3298 |
| 177 | 10 | 30 | 5% | 0,8682 | 0,1783 | 0,5633 | 0,8682 | 0,3311 |
| 178 | 10 | 5 | 7% | 0,8527 | 0,1783 | 0,4511 | 0,8527 | 0,3279 |
| 179 | 10 | 10 | 7% | 0,8682 | 0,1860 | 0,7217 | 0,8682 | 0,2679 |
| 180 | 10 | 30 | 7% | 0,8682 | 0,1783 | 0,5638 | 0,8682 | 0,3330 |
| 181 | 11 | 5 | 0,1% | 0,1579 | 0,1579 | 0,1579 | 0,1579 | 0,0000 |
| 182 | 11 | 10 | 0,1% | 0,6737 | 0,1579 | 0,2095 | 0,6737 | 0,1547 |
| 183 | 11 | 30 | 0,1% | 0,6737 | 0,1579 | 0,1754 | 0,6737 | 0,0925 |
| 184 | 11 | 5 | 1% | 0,6737 | 0,1579 | 0,2863 | 0,7053 | 0,1952 |
| 185 | 11 | 10 | 1% | 0,7053 | 0,1579 | 0,4432 | 0,7158 | 0,2627 |
| 186 | 11 | 30 | 1% | 0,7053 | 0,1579 | 0,3400 | 0,7263 | 0,2273 |
| 187 | 11 | 5 | 2% | 0,7158 | 0,7053 | 0,7095 | 0,7158 | 0,0051 |
| 188 | 11 | 10 | 2% | 0,7158 | 0,2211 | 0,5621 | 0,7158 | 0,2210 |
| 189 | 11 | 30 | 2% | 0,7158 | 0,1579 | 0,4734 | 0,7263 | 0,2481 |
| 190 | 11 | 5 | 3% | 0,7158 | 0,1789 | 0,4105 | 0,7158 | 0,2456 |
| 191 | 11 | 10 | 3% | 0,7053 | 0,1789 | 0,4169 | 0,7158 | 0,2362 |
| 192 | 11 | 30 | 3% | 0,7158 | 0,2211 | 0,6049 | 0,7158 | 0,1897 |
| 193 | 11 | 5 | 5% | 0,7158 | 0,2421 | 0,6169 | 0,7158 | 0,1874 |
| 194 | 11 | 10 | 5% | 0,7158 | 0,2316 | 0,5653 | 0,7263 | 0,2075 |
| 195 | 11 | 30 | 5% | 0,7263 | 0,2421 | 0,6635 | 0,7263 | 0,1385 |
| 196 | 11 | 5 | 7% | 0,7158 | 0,7053 | 0,7095 | 0,7158 | 0,0051 |
| 197 | 11 | 10 | 7% | 0,7158 | 0,2526 | 0,6642 | 0,7263 | 0,1375 |
| 198 | 11 | 30 | 7% | 0,7263 | 0,2316 | 0,6818 | 0,7263 | 0,1191 |
| 199 | 12 | 5 | 0,1% | 0,9063 | 0,1146 | 0,2834 | 0,9271 | 0,3121 |
| 200 | 12 | 10 | 0,1% | 0,1667 | 0,1146 | 0,1250 | 0,1667 | 0,0208 |
| 201 | 12 | 30 | 0,1% | 0,8750 | 0,1146 | 0,1955 | 0,9271 | 0,2203 |
| 202 | 12 | 5 | 1% | 0,9063 | 0,1667 | 0,3146 | 0,9271 | 0,2958 |
| 203 | 12 | 10 | 1% | 0,8958 | 0,1667 | 0,3854 | 0,9375 | 0,3207 |
| 204 | 12 | 30 | 1% | 0,8854 | 0,1146 | 0,2799 | 0,9271 | 0,2628 |
| 205 | 12 | 5 | 2% | 0,9063 | 0,1667 | 0,4646 | 0,9375 | 0,3522 |
| 206 | 12 | 10 | 2% | 0,9063 | 0,1667 | 0,3938 | 0,9375 | 0,3288 |
| 207 | 12 | 30 | 2% | 0,9271 | 0,1667 | 0,4608 | 0,9375 | 0,3513 |
| 208 | 12 | 5 | 3% | 0,8750 | 0,1667 | 0,4604 | 0,9167 | 0,3388 |
| 209 | 12 | 10 | 3% | 0,9167 | 0,1667 | 0,4646 | 0,9479 | 0,3439 |
| 210 | 12 | 30 | 3% | 0,9375 | 0,1667 | 0,5545 | 0,9375 | 0,3527 |
| 211 | 12 | 5 | 5% | 0,8958 | 0,1875 | 0,6083 | 0,9167 | 0,3394 |
| 212 | 12 | 10 | 5% | 0,9375 | 0,1667 | 0,8146 | 0,9479 | 0,2173 |
| 213 | 12 | 30 | 5% | 0,9271 | 0,1667 | 0,7004 | 0,9479 | 0,3098 |
| 214 | 12 | 5 | 7% | 0,8750 | 0,8646 | 0,8729 | 0,8958 | 0,0042 |
| 215 | 12 | 10 | 7% | 0,9063 | 0,1875 | 0,6104 | 0,9271 | 0,3412 |
| 216 | 12 | 30 | 7% | 0,9479 | 0,1667 | 0,8264 | 0,9479 | 0,2156 |
| 217 | 13 | 5 | 0,1% | 0,8701 | 0,1558 | 0,7272 | 0,8961 | 0,2857 |
| 218 | 13 | 10 | 0,1% | 0,8701 | 0,1558 | 0,2987 | 0,8961 | 0,2857 |
| 219 | 13 | 30 | 0,1% | 0,9091 | 0,1558 | 0,3251 | 0,9481 | 0,3059 |

| | | | | Escore de Mutação | | | | |
|---|---|---|---|---|---|---|---|---|
| **Exp.** | **Instrução** | **Qtde BDTs** | **Qtde Tuplas** | **Maior** | **Menor** | **Média** | **Conjunto** | **Desvio Padrão** |
| 220 | 13 | 5 | 1% | 0,9091 | 0,1558 | 0,7506 | 0,9351 | 0,2978 |
| 221 | 13 | 10 | 1% | 0,9091 | 0,1558 | 0,6740 | 0,9351 | 0,3395 |
| 222 | 13 | 30 | 1% | 0,9091 | 0,1558 | 0,8515 | 0,9351 | 0,1863 |
| 223 | 13 | 5 | 2% | 0,9091 | 0,9091 | 0,9091 | 0,9091 | 0,0000 |
| 224 | 13 | 10 | 2% | 0,9091 | 0,8961 | 0,9078 | 0,9091 | 0,0039 |
| 225 | 13 | 30 | 2% | 0,9091 | 0,8701 | 0,9043 | 0,9091 | 0,0119 |
| 226 | 13 | 5 | 3% | 0,9091 | 0,8961 | 0,9065 | 0,9091 | 0,0052 |
| 227 | 13 | 10 | 3% | 0,9091 | 0,9091 | 0,9091 | 0,9091 | 0,0000 |
| 228 | 13 | 30 | 3% | 0,9091 | 0,9091 | 0,9091 | 0,9091 | 0,0000 |
| 235 | 14 | 5 | 0,1% | 0,8182 | 0,1818 | 0,3364 | 0,8636 | 0,2433 |
| 236 | 14 | 10 | 0,1% | 0,9091 | 0,1818 | 0,3591 | 0,9091 | 0,2669 |
| 237 | 14 | 30 | 0,1% | 0,8182 | 0,1818 | 0,2848 | 0,8636 | 0,1403 |
| 238 | 14 | 5 | 1% | 0,9545 | 0,3636 | 0,5909 | 0,9545 | 0,2456 |
| 239 | 14 | 10 | 1% | 0,9091 | 0,3182 | 0,6591 | 0,9545 | 0,2458 |
| 240 | 14 | 30 | 1% | 0,9545 | 0,2273 | 0,7348 | 0,9545 | 0,2249 |
| 241 | 14 | 5 | 2% | 0,9545 | 0,8182 | 0,9000 | 0,9545 | 0,0530 |
| 242 | 14 | 10 | 2% | 0,9545 | 0,7727 | 0,8863 | 0,9545 | 0,0711 |
| 243 | 14 | 30 | 2% | 0,9545 | 0,7727 | 0,8970 | 0,9545 | 0,0663 |
| 244 | 14 | 5 | 3% | 0,9545 | 0,9545 | 0,9545 | 0,9545 | 0,0000 |
| 245 | 14 | 10 | 3% | 0,9545 | 0,8182 | 0,9318 | 0,9545 | 0,0419 |
| 246 | 14 | 30 | 3% | 0,9545 | 0,7727 | 0,9166 | 0,9545 | 0,0576 |
| 247 | 14 | 5 | 5% | 0,9545 | 0,9545 | 0,9545 | 0,9545 | 0,0000 |
| 248 | 14 | 10 | 5% | 0,9545 | 0,8182 | 0,9227 | 0,9545 | 0,0539 |
| 249 | 14 | 30 | 5% | 0,9545 | 0,8182 | 0,9484 | 0,9545 | 0,0255 |
| 253 | 15 | 5 | 0,1% | 0,0746 | 0,0672 | 0,0702 | 0,0821 | 0,0036 |
| 254 | 15 | 10 | 0,1% | 0,1418 | 0,0672 | 0,0762 | 0,1567 | 0,0223 |
| 255 | 15 | 30 | 0,1% | 0,5970 | 0,0672 | 0,1279 | 0,6045 | 0,1463 |
| 256 | 15 | 5 | 1% | 0,6343 | 0,5448 | 0,5731 | 0,6343 | 0,0367 |
| 257 | 15 | 10 | 1% | 0,6269 | 0,0896 | 0,3463 | 0,6418 | 0,2279 |
| 258 | 15 | 30 | 1% | 0,6119 | 0,0746 | 0,3836 | 0,6493 | 0,2163 |
| 259 | 15 | 5 | 2% | 0,6343 | 0,0970 | 0,5089 | 0,6343 | 0,2089 |
| 260 | 15 | 10 | 2% | 0,7910 | 0,1642 | 0,4933 | 0,8134 | 0,2179 |
| 261 | 15 | 30 | 2% | 0,7463 | 0,0970 | 0,5259 | 0,7687 | 0,1755 |
| 262 | 15 | 5 | 3% | 0,7687 | 0,1716 | 0,5403 | 0,7687 | 0,1994 |
| 263 | 15 | 10 | 3% | 0,6343 | 0,1716 | 0,4731 | 0,6418 | 0,1868 |
| 264 | 15 | 30 | 3% | 0,7463 | 0,1716 | 0,5736 | 0,7687 | 0,1289 |
| 265 | 15 | 5 | 5% | 0,7687 | 0,6045 | 0,6507 | 0,7687 | 0,0602 |
| 266 | 15 | 10 | 5% | 0,7687 | 0,6045 | 0,6530 | 0,7687 | 0,0591 |
| 267 | 15 | 30 | 5% | 0,8209 | 0,5821 | 0,6435 | 0,8582 | 0,0533 |
| 268 | 15 | 5 | 7% | 0,8134 | 0,6119 | 0,6940 | 0,8134 | 0,0811 |
| 269 | 15 | 10 | 7% | 0,7687 | 0,6045 | 0,6478 | 0,7687 | 0,0418 |
| 270 | 15 | 30 | 7% | 0,7687 | 0,2239 | 0,6298 | 0,8657 | 0,0900 |

## E.2 Gráficos de Evolução dos Escores de Mutação

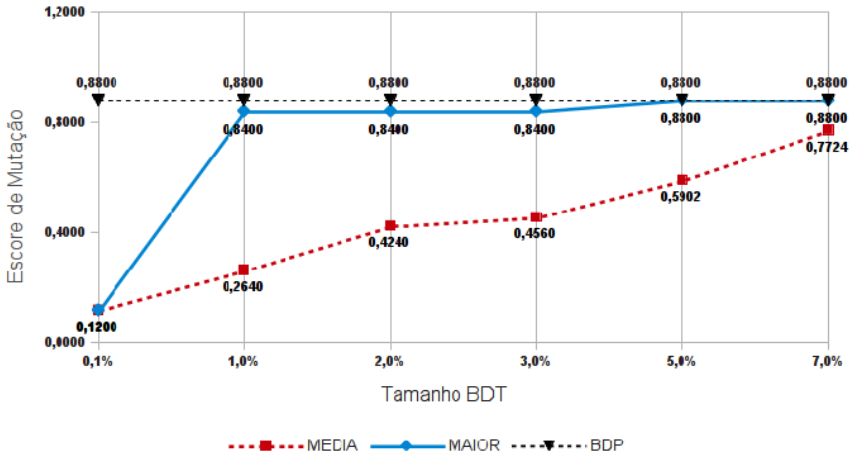


**Figura E.1:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 1*

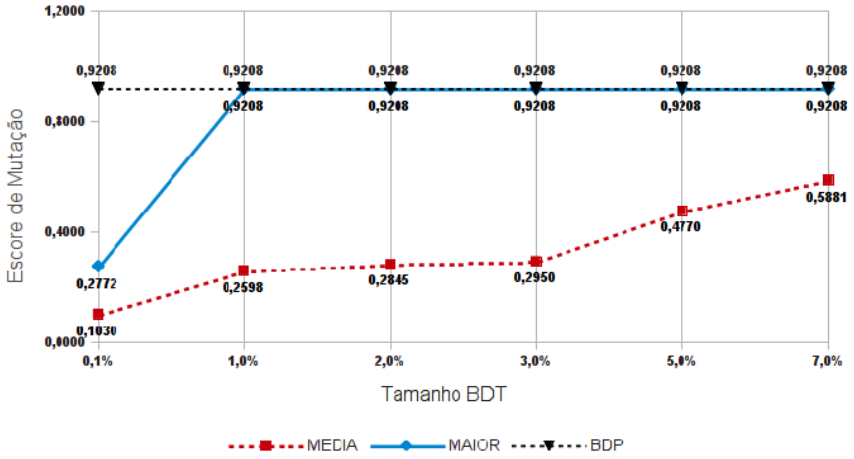


**Figura E.2:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 2*

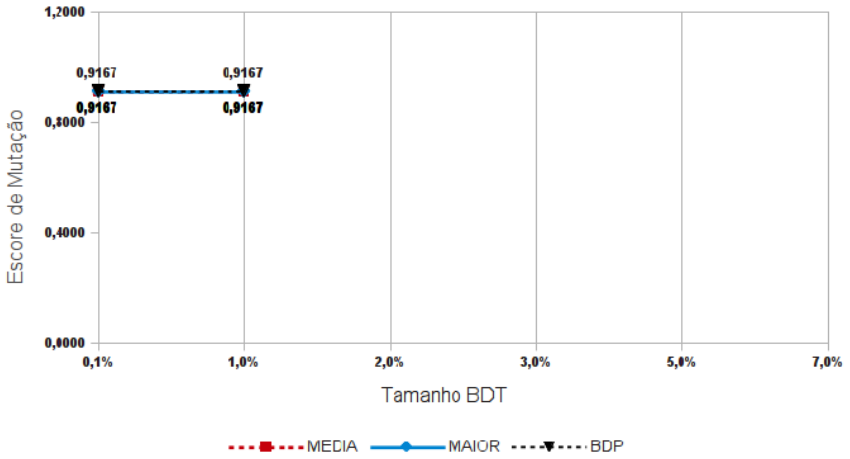


**Figura E.3:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 3*

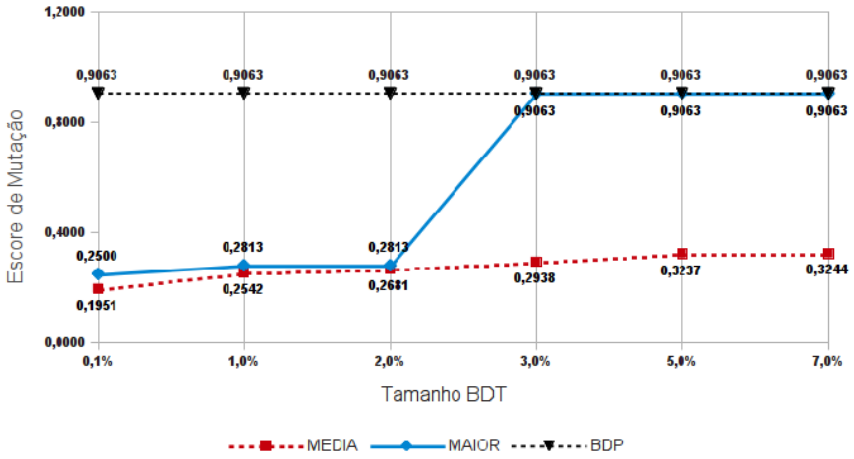


**Figura E.4:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 4*

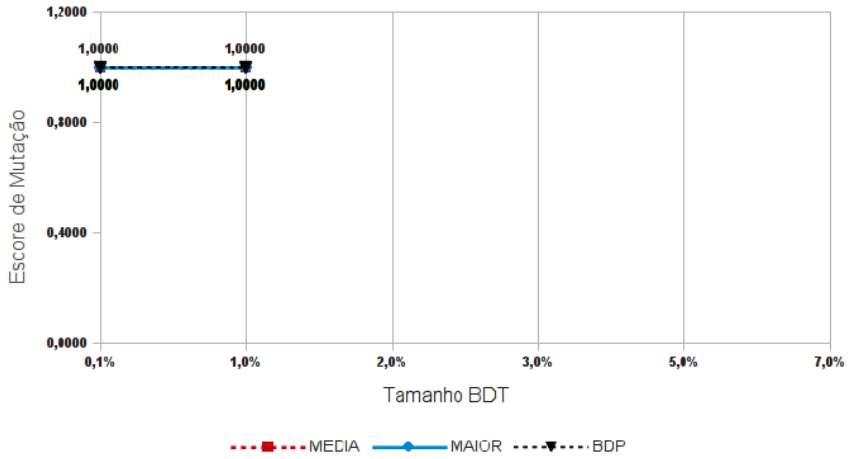


**Figura E.5:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 5*

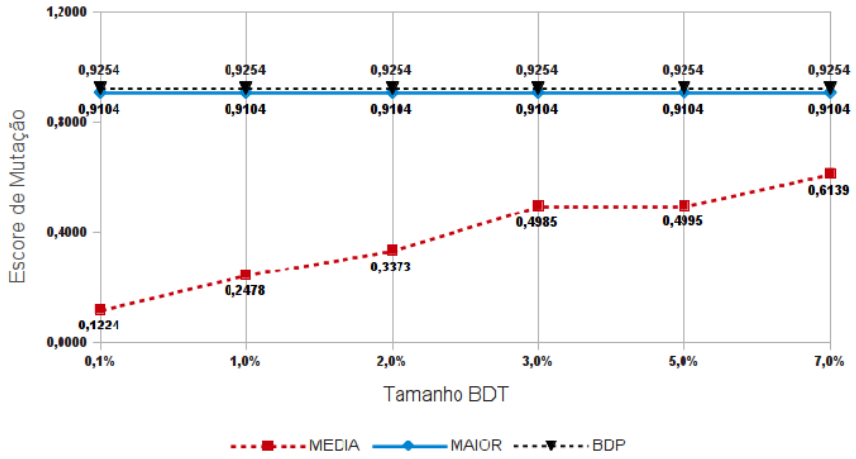


**Figura E.6:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 6*

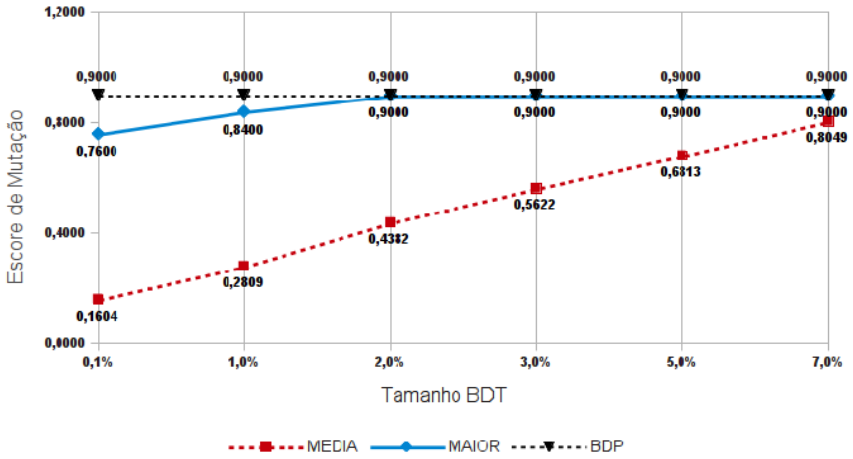


**Figura E.7:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 7*

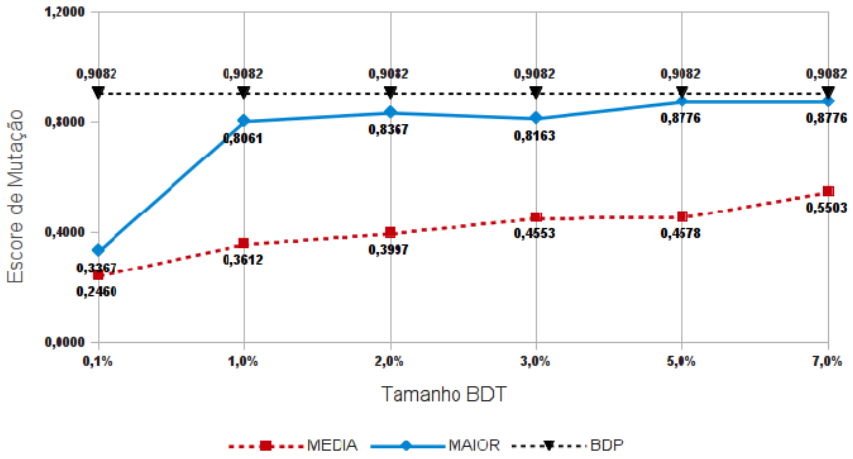


**Figura E.8:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 8*

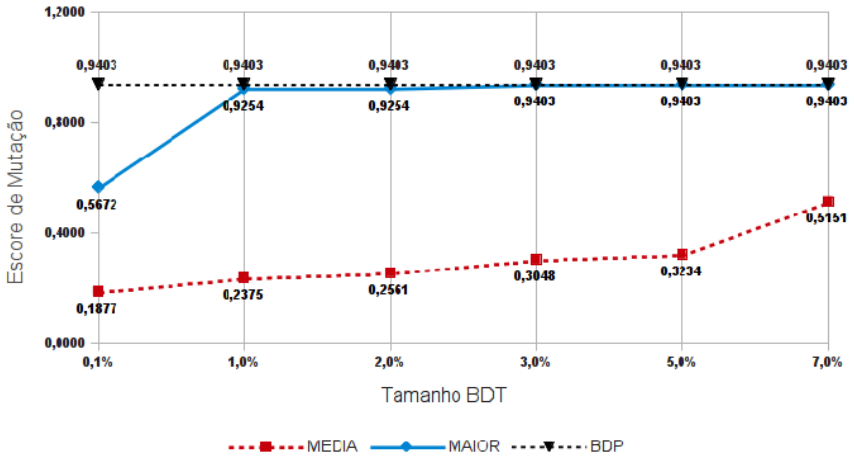


**Figura E.9:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 9*

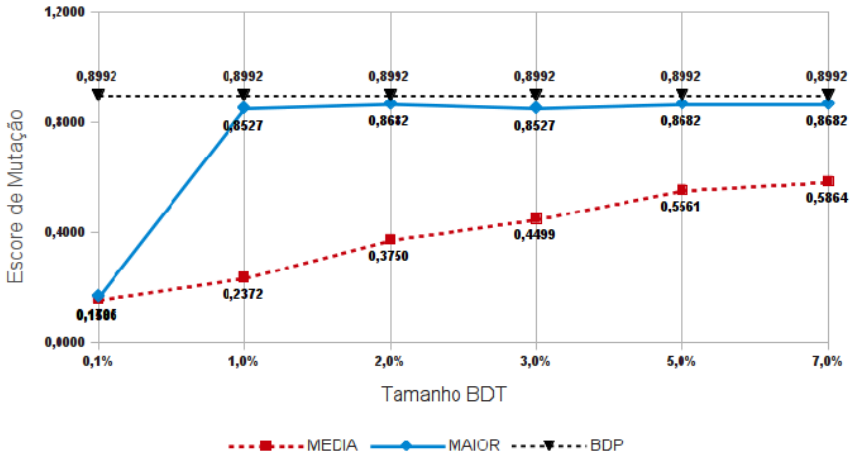


**Figura E.10:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 10*

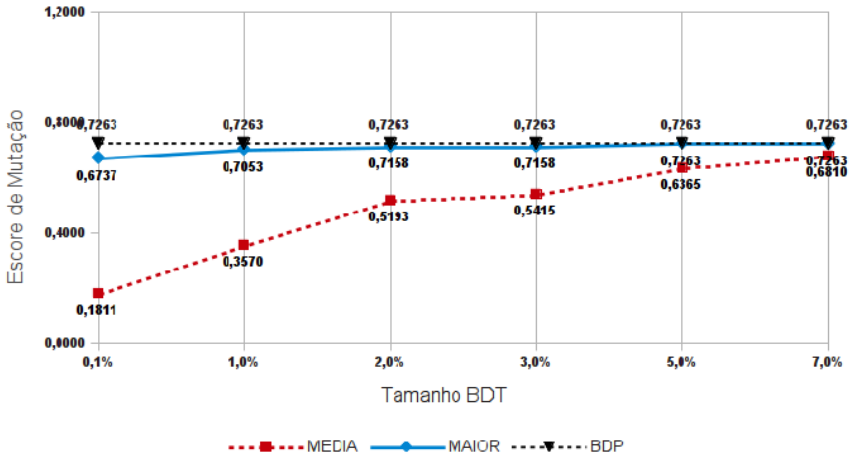


**Figura E.11:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 11*

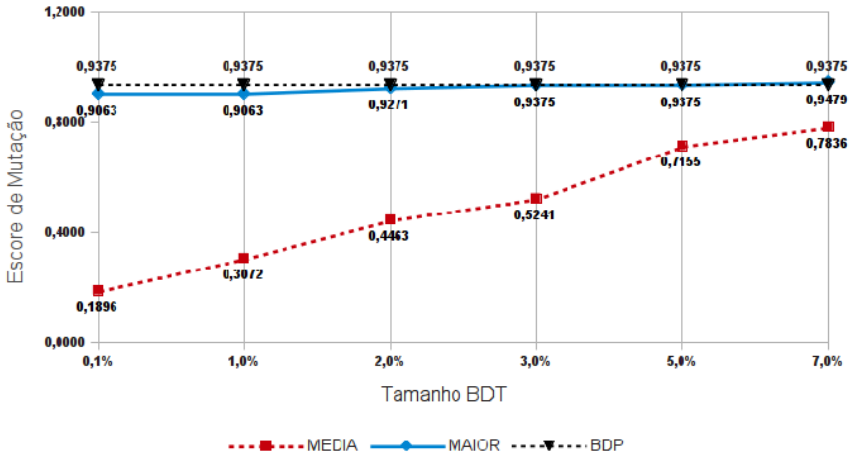


**Figura E.12:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 12*

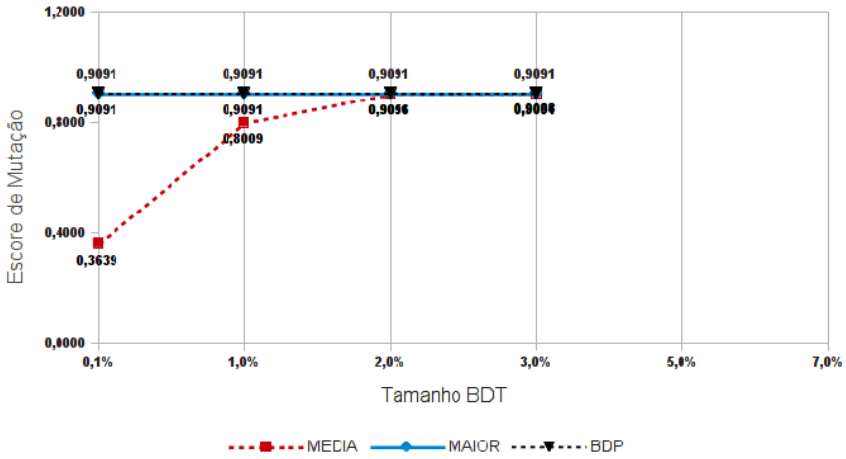


**Figura E.13:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 13*

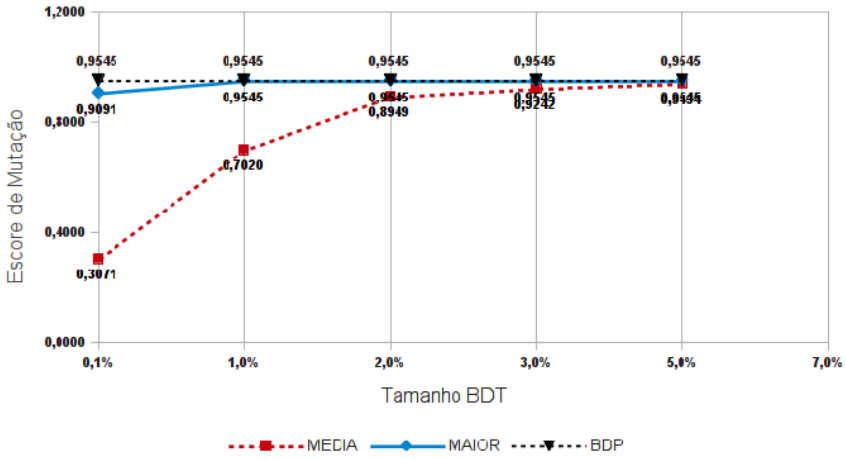


**Figura E.14:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 14*

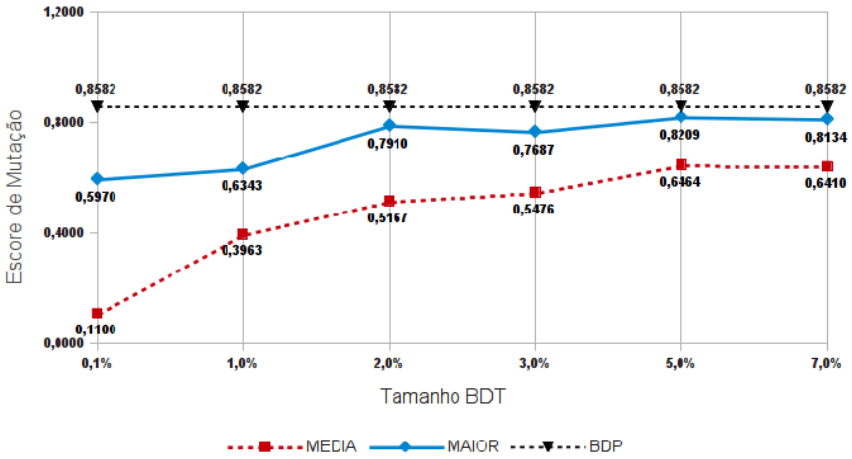


**Figura E.15:** *Cenário 2 - Evolução do Escore de Mutação da Instrução 15*

APÊNDICE F

# Base de Dados de Experimentos

A Figura F.1 apresenta o modelo físico criado para a Base de Dados de Experimentos.

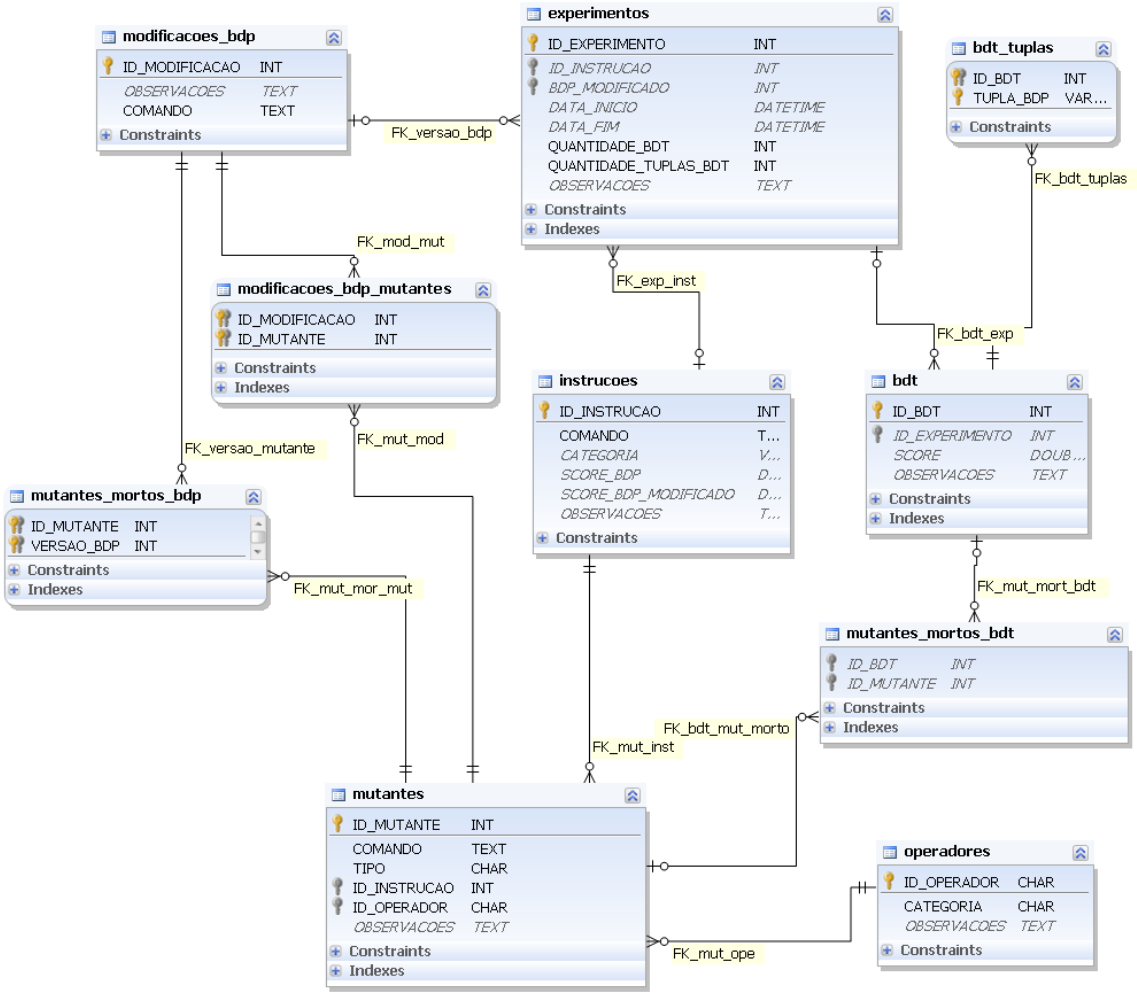


**Figura F.1:** *Diagrama do Modelo Físico da BDE*