Automatização de Operações de Registros de Banco de Dados em Equipamentos do Setor de Telecomunicações

Jean Francisco Giareta, Eder Pazinatto

Instituto de Tecnologia – Curso de Ciência da Computação
Universidade de Passo Fundo (UPF)
BR 285 Km 292,7 – Bairro São José – Passo Fundo, RS, Brasil

{172959, ederpazinatto}@upf.br

Abstract. *This article presents the development of an automated script for massive insertion and deletion of records of damaged equipment in an Oracle database, applied to the telecommunications sector. The solution, developed in Python using SQLPlus, aims to optimize the management of large volumes of data, ensuring integrity, consistency, and scalability of operations. The tests conducted demonstrated a reduction of up to 99.67% in execution time compared to the manual method for volumes of up to 10,000 records. In addition to increasing operational efficiency, the script significantly reduces the likelihood of human errors and ensures high-quality data processing, providing a robust and adaptable solution to the growing demand for real-time data management in telecommunications networks.*

Resumo. *Este artigo apresenta o desenvolvimento de um script automatizado para a inserção e remoção massiva de registros de equipamentos com danos em um banco de dados Oracle, aplicado ao setor de telecomunicações. A solução, desenvolvida em Python utilizando SQLPlus, busca otimizar o gerenciamento de grandes volumes de dados, garantindo integridade, consistência e escalabilidade das operações. Os testes realizados demonstraram uma redução de até 99,67% no tempo de execução em comparação ao método manual para volumes de até 10.000 registros. Além de aumentar a eficiência operacional, o script reduz significativamente a probabilidade de erros humanos e assegura alta qualidade no processamento dos dados, oferecendo uma solução robusta e adaptável ao crescimento da demanda por gerenciamento em tempo real em redes de telecomunicações.*

1. Introdução

No ambiente corporativo moderno, especialmente em setores intensivos em dados como telecomunicações, o gerenciamento eficiente de grandes volumes de informações é um desafio crescente. A expansão contínua das redes e infraestruturas digitais resulta em um aumento exponencial na geração de dados, demandando soluções robustas para armazenamento, manipulação e atualização dessas informações. Segundo Torres (2018), sistemas de gerenciamento de banco de dados (SGBDs) desempenham um papel central na organização e manutenção da integridade dos dados, particularmente em cenários onde a confiabilidade e a consistência são cruciais.

No setor de telecomunicações, a integridade e a atualização dos dados relacionados à infraestrutura, como equipamentos, cabos e conectores, são essenciais para garantir a operação eficiente e ininterrupta dos serviços. Contudo, processos manuais para inserir ou remover grandes volumes de dados em bancos de dados de grande porte apresentam limitações significativas: demandam tempo considerável, exigem equipes especializadas e estão sujeitos a erros humanos. Estas limitações se tornam ainda mais evidentes em situações críticas, como em casos de danos massivos

em equipamentos, onde a agilidade no gerenciamento dos dados é imprescindível para evitar interrupções nos serviços.

A automação de operações de inserção e remoção massiva em bancos de dados surge como uma solução estratégica para mitigar essas dificuldades. De acordo com Elmasri & Navathe (2011), a automação é um fator determinante para alcançar escalabilidade, eficiência e maior controle sobre grandes volumes de dados. No caso específico de empresas de telecomunicações, essa abordagem não apenas reduz o tempo de processamento, mas também minimiza a dependência de intervenção manual, mitigando riscos associados a erros e promovendo a continuidade operacional.

Neste contexto, este artigo aborda o desenvolvimento e a implementação de um script automatizado voltado para a inserção e remoção de registros relacionados a danos massivos em equipamentos de telecomunicações em um banco de dados Oracle. A solução proposta resolve os principais desafios enfrentados no processo manual, reduzindo drasticamente o tempo necessário para manipular grandes volumes de dados e garantindo a integridade e consistência das informações.

A relevância deste trabalho reside na capacidade de otimizar o gerenciamento de dados críticos de infraestrutura em larga escala. Empresas de telecomunicações frequentemente lidam com cenários de alta complexidade, onde a demora na atualização de dados pode levar a interrupções de serviço, diagnósticos incorretos ou falhas em ações de manutenção. Conforme Stonebraker e Hellerstein (2005) destacam, a eficiência no gerenciamento de dados é um pilar fundamental para operações críticas. A automação, como demonstrado neste estudo, não apenas supera os desafios associados ao processo manual, mas também estabelece um novo padrão de eficiência e confiabilidade no setor.

2. Revisão Bibliográfica

Nesta seção, falamos um pouco mais sobre SGBDs, sua importância e trabalhos relacionados.

2.1 Banco de Dados e sua Importância no Setor de Telecomunicações

Os Sistemas de Gerenciamento de Banco de Dados (SGBDs) desempenham um papel crucial em qualquer ambiente que dependa do armazenamento, manipulação e recuperação eficiente de grandes volumes de dados. Eles são utilizados em várias indústrias, incluindo finanças, saúde, comércio e, principalmente, telecomunicações, onde o volume de dados gerados e armazenados cresce de forma exponencial. Segundo Elmasri e Navathe (2011), os SGBDs são projetados para fornecer acesso eficiente a dados estruturados, garantir a integridade transacional e oferecer suporte a operações simultâneas de usuários. No setor de telecomunicações, eles são utilizados para gerenciar uma variedade de informações, como registros de clientes, estado da rede, equipamentos de infraestrutura e dados de falhas.

No contexto de redes de telecomunicações, a eficiência de um banco de dados afeta diretamente a operação da rede. SGBDs como Oracle, MySQL e PostgreSQL são amplamente utilizados, sendo o Oracle um dos mais preferidos em ambientes corporativos devido à sua capacidade de lidar com grandes volumes de dados e operações de alta complexidade. Conforme Sanches et al., (2022) ressalta, a escolha de

um SGBD confiável é essencial para o desempenho da rede e para a manutenção da integridade dos dados, especialmente em casos de infraestrutura crítica como redes de fibra óptica e sistemas de conectividade.

O uso de um SGBD completo como o Oracle neste trabalho se justifica pela sua escalabilidade e recursos avançados de gerenciamento de dados. O Oracle é capaz de lidar com grandes transações simultâneas e tem um sistema sofisticado de controle de concorrência, essencial para garantir que as operações em larga escala mantenham a integridade e consistência dos dados, como mencionado por Garcia-Molina et al. (2009). Esse tipo de suporte é particularmente importante em redes de telecomunicações, onde qualquer inconsistência pode levar à interrupção dos serviços ou à perda de dados críticos para a operação.

2.2 Automação na Gerência de Operações e Dados

Com o aumento exponencial de dados gerados e a crescente complexidade das operações em redes de telecomunicações, a automação de processos no gerenciamento de banco de dados tornou-se indispensável. A automação reduz a necessidade de intervenção manual, minimiza erros e acelera significativamente o processamento de grandes volumes de dados. Segundo Elmasri e Navathe (2011), a automação não só aprimora a eficiência operacional, mas também assegura que o desempenho de sistemas complexos, como os de telecomunicações, se mantenha em níveis elevados. No contexto deste trabalho, a automação é aplicada na inserção e remoção massiva de registros de danos em equipamentos de redes.

O conceito de "batch processing", discutido por Silberschatz et al. (2011), destaca a importância de processar múltiplas operações de forma agrupada para aumentar a eficiência. Esse tipo de processamento é crucial quando se lida com milhares de registros, permitindo que transações sejam realizadas em blocos, reduzindo a sobrecarga no banco de dados e garantindo que as operações sejam realizadas de maneira mais rápida e eficiente. No setor de telecomunicações, onde grandes volumes de dados precisam ser inseridos ou removidos frequentemente, a automação desse processo reduz o tempo de operação e o risco de inconsistências causadas por falhas humanas.

Além disso, o uso de scripts automatizados também ajuda a garantir a integridade transacional, como discutido por Hammes et al., (2021). Transações em banco de dados devem seguir o princípio ACID (Atomicidade, Consistência, Isolamento e Durabilidade) para garantir que todas as operações sejam concluídas de maneira confiável, mesmo em caso de falhas. O script desenvolvido neste trabalho foi projetado para seguir essas diretrizes, garantindo que, em caso de erro, o banco de dados possa reverter as operações para manter um estado consistente.

2.3 Trabalhos Similares

A automação no gerenciamento de banco de dados é um tema amplamente explorado na literatura, especialmente no que diz respeito ao manuseio de grandes volumes de dados em setores de infraestrutura crítica, como telecomunicações. Stonebraker et al. (2010) investigaram a escalabilidade de sistemas de banco de dados em ambientes de Big Data, destacando a importância de arquiteturas distribuídas para suportar grandes volumes de dados. A pesquisa revelou que soluções como o processamento massivo de dados em

paralelo, automatizado por scripts, podem melhorar o desempenho em operações de inserção e remoção de dados, características essenciais em redes de telecomunicações.

A escalabilidade e a eficiência no manuseio de grandes volumes de dados também foram discutidas por Abadi et al. (2009), que destacaram a importância de sistemas de gerenciamento de dados tolerantes a falhas e escaláveis para infraestruturas críticas. No contexto de telecomunicações, a automação desses processos facilita o gerenciamento de dados de forma eficiente, garantindo que as operações de inserção, atualização e remoção possam ser realizadas sem comprometer a integridade e a consistência das informações.

Esses trabalhos, em conjunto, mostram que a automação de operações de banco de dados, como inserção e remoção massiva de registros, não só facilita o gerenciamento de grandes volumes de dados, mas também assegura maior eficiência e robustez nos sistemas. A presente pesquisa avança ao oferecer uma solução focada em operações críticas para o setor de telecomunicações, como a gestão de equipamentos danificados em redes, destacando os benefícios da automação no ambiente corporativo.

3. Materiais e Métodos

Nesta seção, são brevemente apresentados as ferramentas aplicadas na concepção desse projeto.

3.1. Ferramentas Utilizadas

Para o desenvolvimento deste script, foi necessário utilizar um conjunto de ferramentas que permitem a manipulação eficiente de grandes volumes de dados em um ambiente de telecomunicações. A seguir, são descritas as principais ferramentas utilizadas:

Oracle Database: O Oracle é um dos Sistemas de Gerenciamento de Banco de Dados (SGBD) mais utilizados em ambientes corporativos. Sua escolha foi motivada por sua capacidade de lidar com grandes volumes de dados, suporte a transações complexas e controle de concorrência. Para este estudo, foi utilizada a versão Oracle 19c, que oferece recursos avançados de escalabilidade e performance, essenciais para operações de inserção e remoção massiva de dados.

SQLPlus: Ferramenta de linha de comando utilizada para interação com o banco de dados Oracle. Foi empregada para executar comandos SQL e scripts de inserção e remoção de dados diretamente no Oracle, sendo essencial para a execução dos scripts desenvolvidos.

Python 3.6: A linguagem Python foi escolhida pela sua simplicidade e versatilidade na automação de processos. As bibliotecas padrão de Python, como *subprocess*, *argparse*, *re, defaultdict e csv*, foram usadas para gerenciar as interações entre os scripts, banco de dados e os arquivos de entrada/saída.

PyInstaller: Ferramenta utilizada para compilar os scripts Python em executáveis, facilitando a distribuição e a execução dos scripts em ambientes sem a necessidade de instalar o interpretador Python.

Linux (CentOS/Red Hat): O sistema operacional Linux foi utilizado como ambiente de desenvolvimento e execução dos scripts. O Linux oferece um ambiente

estável e otimizado para o gerenciamento de grandes processos, especialmente em servidores corporativos.

3.2. Diagrama do Processo

A figura 1 descreve o fluxo geral do processo de inserção e remoção massiva de registros de equipamentos com danos massivos em um banco de dados Oracle.

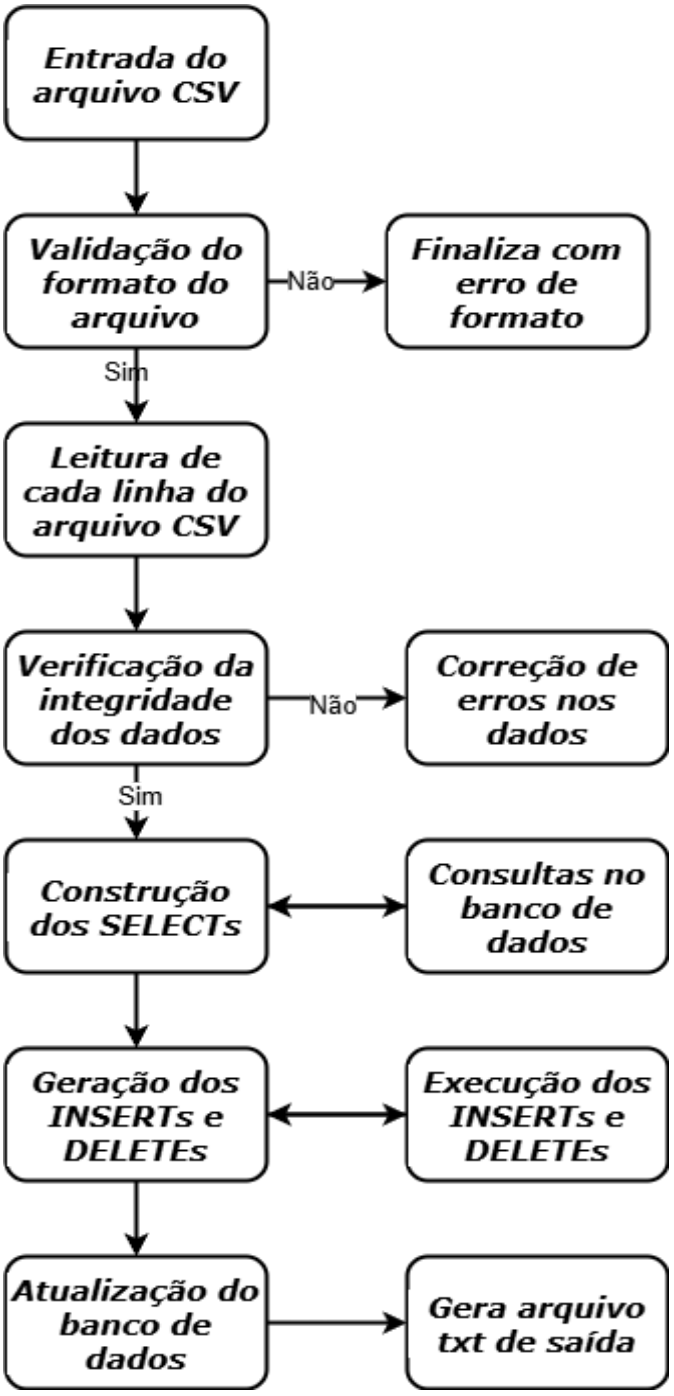


Figura 1. Diagrama de processos do script

Entrada do arquivo CSV: A entrada do arquivo CSV dá início ao processo, fornecendo dados essenciais para a execução das operações no banco de dados. Esse arquivo geralmente contém informações organizadas em colunas como Localidade, Site

e nome do equipamento. Estes campos permitem que o script identifique cada ocorrência de forma precisa e busque o restante das informações necessárias nas tabelas de equipamentos.

Validação do formato do arquivo: O script verifica se o arquivo CSV está no formato esperado. Se o formato estiver incorreto, o fluxo segue para a fase de Finalização com erro de formato. Caso contrário, o processo continua.

Leitura de cada linha do arquivo CSV: Se o arquivo passar na validação, cada linha do CSV é lida para ser processada.

Verificação da integridade dos dados: Após a leitura das linhas, o script verifica a integridade dos dados. Se houver problemas, o fluxo segue para a correção de erros nos dados. Caso os dados estejam corretos, o fluxo continua.

Construção de Consultas: Com os dados verificados, o script monta os comandos SQL do tipo SELECT, que serão usados para consulta no banco de dados.

Consultas no banco de dados: O script realiza as consultas no banco de dados para verificar os registros existentes e determinar quais comandos serão necessários.

Geração dos comandos de manipulação de dados: Com base nas consultas realizadas, o script gera os comandos SQL de INSERT ou DELETE para inserir ou remover registros no banco de dados.

Execução dos comandos de manipulação de dados: Após gerar os comandos, o script executa os INSERTs e DELETEs no banco de dados.

Atualização do banco de dados: Com a execução dos comandos, o banco de dados é atualizado conforme as alterações necessárias (inserção e/ou remoção de registros).

Geração de arquivo txt de saída: Após a atualização do banco, um arquivo de saída no formato .txt é gerado, contendo os logs ou resultados das operações realizadas.

Esse fluxograma reflete um processo automatizado de gerenciamento de dados em um banco de dados, desde a entrada e validação do arquivo de dados até a atualização e geração de logs ou resultados.

3.3. Desenvolvimento

O desenvolvimento da aplicação focou na automação dos processos de inserção e remoção de dados em massa, otimizando operações manuais que anteriormente demandavam muito tempo e eram suscetíveis a erros. Os scripts foram projetados para lidar com grandes volumes de dados, realizando operações seguras e eficientes de forma automatizada.

3.3.1 Estrutura dos Scripts

Script de Inserção (DM_Insert.py): O anexo 1 mostra uma parte do código deste processo.

O objetivo deste script é automatizar a inserção de novos registros de danos massivos em equipamentos de telecomunicações. Ele lê os dados de um arquivo CSV, realiza as validações nos dados de entrada, procura pela localidade e site inicial do equipamento, verifica qual tipo de equipamento é e faz o relacionamento com as tabelas

buscando o cabo origem, contagem e localidade do equipamento. Após isso gera os inserts e insere os registros na tabela FTTX_MASSIVE_DAMAGE, gerando "Massive Numbers" para cada equipamento danificado. Caso algum erro ocorra durante a execução, o script garante que a transação será revertida, mantendo a consistência do banco de dados.

Script de Remoção (DM_Remove.py): O anexo 2 mostra uma parte do código deste processo.

Este script executa a remoção de registros de equipamentos que já tiveram seus danos massivos reparados. Assim como no script de inserção, a remoção é realizada de forma automatizada e eficiente, com base nos dados fornecidos pelo arquivo CSV.

3.3.2 Estrutura do Arquivo CSV

O arquivo CSV utilizado pelos scripts segue uma estrutura pré-definida, contendo as colunas "*Localidade*", "*Site*" e "*Codigo do equipamento terminal*". Este arquivo pode ser gerado no módulo do sistema de gerência da planta chamado Reports Manager. Onde é possível gerar o arquivo CSV a partir de um cabo de origem que alimenta os equipamentos. Ou por localidade, site onde lista todos os equipamentos daquela região. Na figura 4 está um exemplo de como os dados devem ser fornecidos para o correto funcionamento dos scripts.

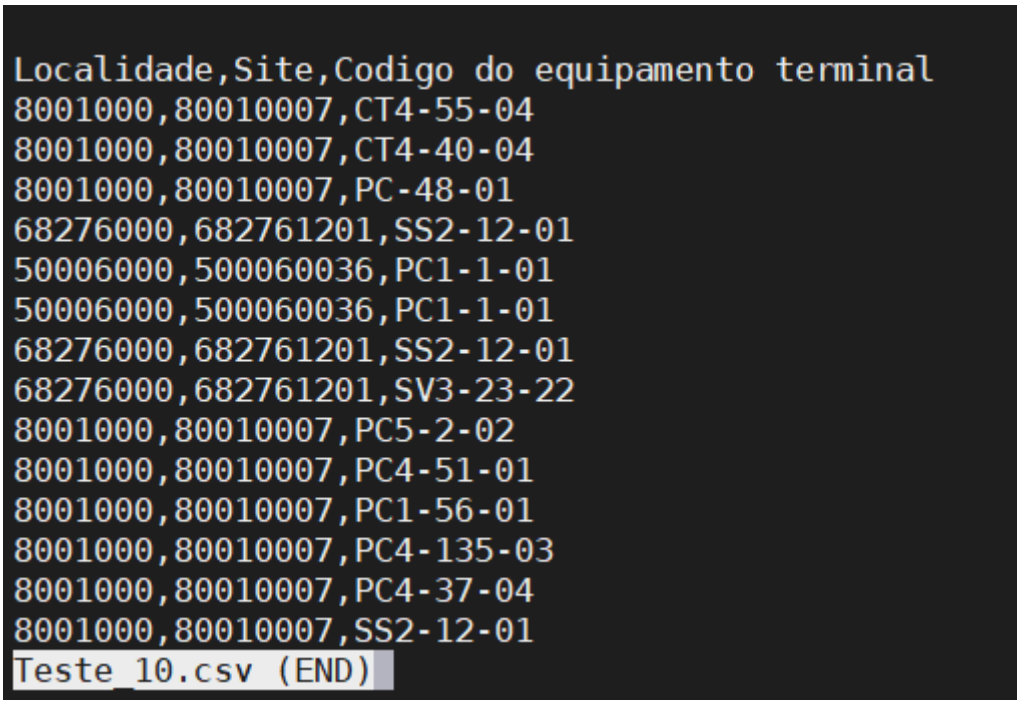


Figura 2. Exemplo de arquivo .csv

Cada linha representa um equipamento que será inserido ou removido da base de dados, conforme a execução do script.

3.3.3 Geração de "Massive Numbers"

A geração dos "Massive Numbers", que identificam os danos massivos nos equipamentos, é feita por meio de uma *sequence* no banco de dados Oracle. Essa sequence é definida pela seguinte instrução SQL:

```
SQL>
SQL>
SQL> CREATE SEQUENCE SEQ_FTTX_MASSIVE_NUMBER
START WITH 1000
INCREMENT BY 1
NOCACHE
NOCYCLE;  2    3    4    5

Sequence created.
```

Figura 3. SQL de criação da sequence

Esta sequência (sequence) garante que cada registro inserido no banco de dados tenha um identificador único, o que é crucial para o gerenciamento correto dos dados.

3.3.4 Saída do Script

O arquivo de log (arquivo_de_saida) conterá uma lista de mensagens indicando se as operações foram bem-sucedidas ou se ocorreram erros.

```
Equipamento PC-48-01 ja possui dano massivo. MassiveNumber: PE1776
Equipamento SS2-12-01 ja possui dano massivo. MassiveNumber: PE1777
Equipamento CT4-55-04 ja possui dano massivo. MassiveNumber: PE1774
Equipamento CT4-40-04 ja possui dano massivo. MassiveNumber: PE1775
Equipamento PC1-1-01 ja possui dano massivo. MassiveNumber: PE1778
Equipamento SV3-23-22, Localidade 68276000, Site 682761201 nao encontrado
Sucesso, DM inserido no equipamento: CT7-36-04, Localidade: 8001000, Site: 80010007, MassiveNumber: PE1779
Sucesso, DM inserido no equipamento: CT2-37-04, Localidade: 8001000, Site: 80010007, MassiveNumber: PE1780
Sucesso, DM inserido no equipamento: CT1-37-04, Localidade: 8001000, Site: 80010007, MassiveNumber: PE1781
Sucesso, DM inserido no equipamento: CT8-34-01, Localidade: 8001000, Site: 80010007, MassiveNumber: PE1782
```

Figura 4. Mensagens de sucesso de inserção

A Figura 6 exibe as mensagens de status geradas durante o processo de inserção de dados no banco. Essas mensagens fornecem informações sobre o sucesso das operações e sobre possíveis conflitos devido a registros já existentes nos equipamentos juntamente com o MassiveNumber correspondente (por exemplo, "PE1776" para o equipamento "PC-48-01"). Isso permite ao usuário identificar rapidamente quais equipamentos já foram processados anteriormente, facilitando o rastreamento e a filtragem desses registros para evitar duplicidade. Também é possível realizar a geração de relatórios via Reports Management para verificar os assinantes e áreas afetadas pelos danos.

```
Sucesso, DM removido no equipamento: CT4-55-04, Localidade: 8001000, Site: 80010007, MassiveNumber: PE1774
Sucesso, DM removido no equipamento: CT4-40-04, Localidade: 8001000, Site: 80010007, MassiveNumber: PE1775
Sucesso, DM removido no equipamento: PC-48-01, Localidade: 8001000, Site: 80010007, MassiveNumber: PE1776
Sucesso, DM removido no equipamento: SS2-12-01, Localidade: 68276000, Site: 682761201, MassiveNumber: PE1777
Equipamento SV3-23-22, Localidade 68276000, Site 682761201 nao encontrado.
Equipamento PC1-10-02, Localidade 50006000, Site 500060036 nao tem dano massivo ativo.
Sucesso, DM removido no equipamento: CT7-36-04, Localidade: 8001000, Site: 80010007, MassiveNumber: PE1779
Sucesso, DM removido no equipamento: CT2-37-04, Localidade: 8001000, Site: 80010007, MassiveNumber: PE1780
```

Figura 5. Mensagens de sucesso de remoção

A Figura 7 exibe as mensagens de status geradas durante o processo de remoção de danos massivos (DM) no banco de dados. Essas mensagens fornecem informações detalhadas sobre a remoção bem-sucedida dos registros e sobre situações em que a remoção não pôde ser realizada.

3.3.5 Funcionamento do Script

O script desenvolvido visa automatizar o processo de inserção massiva de registros de danos em equipamentos no setor de telecomunicações, utilizando uma base de dados Oracle e a linguagem Python para processar os dados de entrada. Ele é estruturado com várias funções e métodos que trabalham em conjunto para garantir a execução eficiente do processo.

3.3.6 Os principais métodos e funções incluem:

parse_args: responsável por interpretar os argumentos de linha de comando que especificam os arquivos CSV de entrada e saída, permitindo flexibilidade na operação do script.

executar_comando_sqlplus: função central para a comunicação com o banco de dados Oracle, executando os comandos SQL via SQL*Plus e retornando os resultados ou erros das operações.

buscar_cr_site e *parsear_cr_site*: responsáveis por gerar e processar as consultas SQL para obter os identificadores da localidade e do site (REGION_ID e CO_ID) dos equipamentos. Essas informações são essenciais para localizar os equipamentos na base de dados.

*dividir_lista***:** divide grandes listas de equipamentos em sublistas menores, permitindo que as operações em lote sejam processadas de maneira mais eficiente. Essa técnica de processamento em lote é comumente aplicada em operações de grande volume de dados, pois melhora o desempenho e evita sobrecargas no sistema (Sadalage & Fowler, 2013). Os lotes foram configurados com um tamanho de 200 registros para evitar sobrecarga no sistema e considerando a limitação do SQLPLUS em processar adequadamente os dados de retorno ao script.

buscar_dados_equipamento_batch e *parsear_dados_equipamento_batch*: essas funções criam os comandos SQL que buscam dados dos equipamentos em lote a partir de diferentes fontes de dados (como tabelas de visualização), e processam os resultados para extrair os detalhes dos equipamentos, como o ID, nome e valores de fibra. Isso garante que os dados corretos sejam inseridos no sistema de gerenciamento de danos.

inserir_dano_massivo_batch: função responsável pela inserção dos registros de danos massivos no banco de dados em lotes, utilizando a técnica de "batch processing" para otimizar a inserção massiva de dados, reduzindo o tempo total de execução, evitando sobrecarga do sistema e garantindo que múltiplos registros sejam processados de uma só vez (Oracle, 2024).

verificar_dano_massivo_batch: verifica a existência de registros de danos massivos já inseridos para determinados equipamentos, evitando a duplicidade de dados e garantindo a integridade das informações no banco de dados. A verificação prévia é uma prática recomendada para assegurar a integridade dos dados (Oracle, 2024).

gerar_massive_number_batch: gera números exclusivos de danos massivos (MASSIVE_NUMBER) para os novos registros, assegurando a criação de identificadores únicos. O uso de sequências para gerar identificadores únicos em banco de dados é amplamente recomendado para evitar conflitos de chaves primárias (Silberschatz, Korth & Sudarshan, 2011).

3.3.7. Compilar o Script em um Executável

Compilar o script em um executável é uma maneira prática de distribuir e rodar o programa em diferentes máquinas sem precisar instalar Python ou suas bibliotecas. Isso torna o uso do script mais acessível, especialmente para quem não tem conhecimento técnico de programação.

Antes de transformar o script em um executável, é necessário instalar algumas ferramentas que ajudam a empacotar todo o código e as dependências em um único arquivo. As principais ferramentas utilizadas são *wheel* e *PyInstaller*, que preparam o script para ser executado de forma independente (Python Software Foundation, 2024).

Com as ferramentas instaladas, o próximo passo é usar o *PyInstaller*. Essa ferramenta converte o script Python em um executável que pode ser rodado em qualquer computador, sem a necessidade de Python instalado. O PyInstaller gera um arquivo que contém tudo o que é necessário para rodar o programa, facilitando o uso em diferentes sistemas operacionais.

Após o processo de compilação, o PyInstaller cria uma pasta chamada *dist*, onde estarão os arquivos executáveis finais. Esses arquivos podem ser distribuídos e usados sem qualquer configuração adicional de ambiente.

É importante garantir que o executável seja compilado em um ambiente compatível com o sistema onde ele será usado. Por exemplo, se o programa será executado em um servidor Linux mais antigo, é necessário que o executável seja gerado em um ambiente com uma versão de bibliotecas como a *glibc* compatível (Linux Foundation, 2024). Isso evita problemas de compatibilidade, garantindo que o executável funcione corretamente.

Essa abordagem simplifica a distribuição e uso do script, especialmente em ambientes onde a instalação de Python ou bibliotecas adicionais pode ser complicada, garantindo que o programa funcione de maneira independente e eficiente.

3.4. Protocolo de Testes

Após o desenvolvimento dos scripts, foi estabelecido um protocolo de testes para validar sua eficiência, intensidade e comportamento em diferentes cenários de inserção e remoção de dados. A tabela 1 descreve os principais testes realizados.

Tabela 1.Tabela de testes executados

Tipo de Teste	Objetivo	Procedimento
Performance	Avaliar a eficiência do script ao processar grandes volumes de dados.	Realização de testes com arquivos CSV contendo diferentes volumes de registros (de 1 a 10.000 linhas).
Consistência	Garantir que os dados inseridos ou removidos estejam corretos e consistentes.	Após cada execução, consultas SQL foram realizadas no banco de dados para verificar se os registros estavam corretamente inseridos ou removidos.
Reversão	Testar o comportamento do script em caso de erros durante a execução.	Simulação de cenários com falhas durante inserção ou remoção, para garantir que o banco de dados revertesse para um estado consistente, sem perda de dados.
Validação	Verificar se o script identifica registros inválidos ou duplicados no CSV.	Utilização de arquivos CSV com registros inconsistentes ou duplicados, verificando se o script gerava alertas e evitava a inserção de dados incorretos no banco.

4. Resultados

Nesta seção, são apresentados os resultados deste projeto.

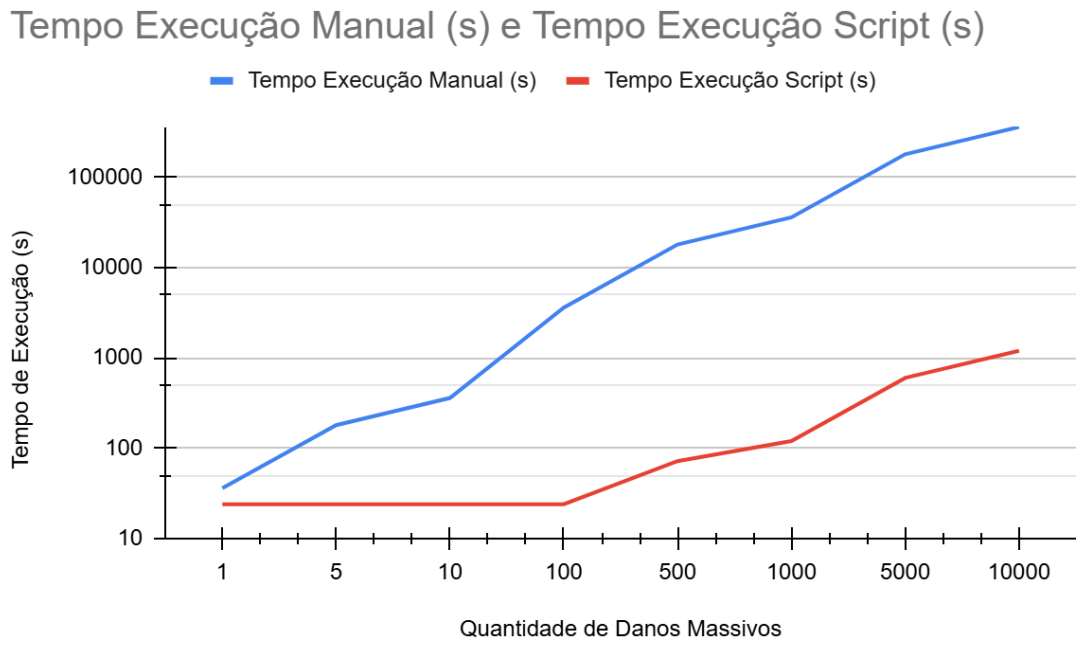
4.1. Performance

Um dos objetivos principais do projeto era aumentar a eficiência das operações de inserção e remoção de dados em massa, reduzindo o tempo gasto em processos manuais e, ao mesmo tempo, garantindo que grandes volumes de dados pudessem ser manipulados sem afetar a integridade ou consistência do banco de dados.

Tabela 2. Comparativo de execução manual e via script.

Quantidade de Danos Massivos	Tempo Execução Manual (s)	Tempo Execução Script (s)	Diferença (%)
1	36	24	33,33
5	180	24	86,67
10	360	24	93,33
100	3600	24	99,33
500	18000	72	99,60
1000	36000	120	99,67
5000	180000	600	99,67
10000	360000	1200	99,67

Gráfico 1. Comparativo de execução manual e via script.



A tabela 2 e o gráfico 1 mostram a comparação entre os tempos de execução dos dois métodos para diferentes quantidades de danos massivos. A diferença percentual destaca a significativa redução no tempo de execução ao utilizar o script, com os ganhos mais expressivos à medida que o volume de dados aumenta. Esses resultados

evidenciam a eficiência e escalabilidade do script, tornando-o uma solução preferencial para processamento de grandes volumes de dados.

Os resultados mostraram que o tempo médio de processamento por registro permaneceu praticamente constante, mesmo com o aumento no volume de dados. Isso demonstra que os scripts possuem boa escalabilidade, sendo capazes de lidar com crescimento de volume sem degradação significativa de desempenho.

Além disso, foi observado que a execução das operações de inserção e remoção em lotes (batch processing) contribuiu significativamente para a redução do tempo total de execução. O processamento em lotes permitiu que múltiplas operações fossem agrupadas e executadas em uma única transação, diminuindo o número de interações com o banco de dados e, consequentemente, o tempo total necessário para concluir a operação.

4.2. Validação de Dados e Integridade

Outro resultado importante foi a validação e integridade dos dados manipulados pelos scripts. Ao processar os arquivos CSV, os scripts realizam uma verificação detalhada dos dados, garantindo que apenas registros válidos sejam inseridos ou removidos do banco de dados. Durante os testes, arquivos contendo dados inconsistentes ou duplicados foram corretamente identificados, com os scripts gerando relatórios de erro e evitando a inclusão de informações incorretas no banco de dados.

Além disso, a geração de "Massive Numbers" através de uma sequence exclusiva no Oracle garantiu que cada registro inserido no banco de dados tivesse um identificador único, evitando conflitos de duplicação. Esse mecanismo de geração sequencial foi crucial para assegurar que as inserções em massa fossem realizadas sem falhas, mesmo com a execução simultânea de múltiplos processos.

4.3. Escalabilidade e Robustez

Em todos os testes, os scripts demonstraram uma alta capacidade de escalar, processando lotes de dados de maneira eficiente e mantendo tempos de execução constantes. Esse comportamento é fundamental para empresas de telecomunicações que lidam com grandes volumes de dados, garantindo que o script possa crescer sem a necessidade de revisões frequentes ou adaptações significativas.

A robustez dos scripts também foi validada durante os testes, com os scripts sendo executados em ambientes controlados e de produção simulada. Mesmo em casos de falha no script ou em interrupções durante o processo, o banco de dados manteve sua integridade, e as operações foram corretamente revertidas ou completadas.

5. Conclusão

Este artigo apresentou o desenvolvimento e a aplicação de scripts automatizados para a inserção e remoção massiva de registros de danos em equipamentos de redes de telecomunicações, utilizando o banco de dados Oracle. A solução proposta demonstrou ser uma abordagem eficaz para enfrentar os desafios impostos pela necessidade de manipulação de grandes volumes de dados, algo comum em empresas de telecomunicações.

Antes da automação com os scripts, as operações de inserção e remoção de dados em massa eram realizadas manualmente, o que implicava em um alto risco de erros humanos, além de demandar tempo considerável da equipe técnica. Com a automação, foi possível reduzir significativamente a quantidade de erros, ao mesmo tempo que o tempo necessário para concluir as operações foi otimizado.

Os scripts se mostraram uma solução prática e eficaz para automatizar tarefas repetitivas e críticas, permitindo que a equipe de TI das empresas de telecomunicações concentrasse seus esforços em outras áreas mais estratégicas. Este aumento de produtividade foi um dos principais benefícios identificados, com uma redução considerável no tempo de resposta às demandas de manutenção de redes.

A implementação dos scripts trouxe benefícios em termos de eficiência, escalabilidade e integridade dos dados. A automação reduz expressivamente o tempo de execução, permitindo o processamento rápido de grandes volumes de dados. O uso de transações atômicas garantiu a confiabilidade do script, mesmo em caso de falhas. A geração de "Massive Numbers" através de uma sequence no Oracle assegurou a unicidade e rastreabilidade dos registros, contribuindo para a robustez do sistema.

Além disso, a execução dos scripts em lotes e a capacidade de escalar conforme o aumento do volume de dados demonstraram a adequação da solução para ambientes corporativos com grandes bancos de dados de infraestrutura de telecomunicações. A adoção dos scripts resultou em uma redução significativa de erros humanos e no aumento da produtividade das equipes responsáveis pela manutenção das redes.

Em suma, este trabalho oferece uma solução automatizada robusta e eficiente para o setor de telecomunicações, que pode servir de modelo para outras áreas que enfrentam desafios semelhantes, como bancos de dados corporativos e grandes data centers.

5.1. Considerações Finais

Entre os desafios enfrentados durante o desenvolvimento, a manipulação de grandes volumes de dados e a interação eficiente com o banco de dados Oracle foram os mais significativos. O processamento de grandes volumes de registros exigiu a implementação de um sistema de lotes, que divide o processamento em blocos menores, garantindo que o desempenho não fosse prejudicado e que as operações fossem escaláveis.

Além disso, o tratamento de erros e a garantia de que transações incompletas não afetam a integridade do banco de dados foram aspectos críticos. O uso de mecanismos de controle transacional do Oracle, combinados com a automação oferecida pelo Python, garantiram que o script fosse robusto e capaz de lidar com operações complexas.

5.2. Trabalhos Futuros

Dado o bom resultado do script apresentado, futuras melhorias podem incluir a otimização do desempenho em ambientes ainda mais robustos, a implementação de algoritmos que lidam com paralelismo de operações e o desenvolvimento de sistemas de monitoramento automático para o acompanhamento em tempo real das operações. Dessa forma, seria possível expandir ainda mais o impacto positivo da solução e atender a demandas cada vez mais complexas no setor de telecomunicações.

6. Referências

ABADI, Daniel J.; et al. Scalable semantic web data management using vertical partitioning. In: Proceedings of the 33rd international conference on Very Large Data Bases. 2007.

ELMASRI, R.; NAVATHE, S. B. Fundamentals of Database Systems. Pearson, 2011.

GARCIA-MOLINA, H.; ULLMAN, J. D.; WIDOM, J. Database Systems: The Complete Book. Pearson, 2009.

HAMMES, M. R.; BRUM, A. L.; FRANCK, D. M.; BRIZOLLA, M. M. B.; TEIXEIRA, M. H. Impactos da falta de comunicação entre sistemas de informação em uma empresa do setor metalomecânico: um estudo de caso. Research, Society and Development, v. 10, n. 1, p. e18710111572-e18710111572, 2021.

LINUX FOUNDATION. Glibc Documentation. 2024. Disponível em: https://www.gnu.org/software/libc. Acesso em: 12 nov. 2024.

ORACLE CORPORATION. SQLPlus User's Guide and Reference. 2024. Disponível em: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlplus/index.html. Acesso em: 12 nov. 2024.

PYTHON SOFTWARE FOUNDATION. PyInstaller Documentation. 2024. Disponível em: https://www.pyinstaller.org. Acesso em: 12 nov. 2024.

SADALAGE, P. J.; FOWLER, M. NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Pearson Education, 2013.

SANCHES, Bruno; et al. Desenvolvimento de um modelo de sistema BI para o acompanhamento do desempenho do setor industrial e monitoramento ergonômico do ambiente laboral em uma empresa do ramo de telecomunicação. 2022.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. Database System Concepts. 6. ed. McGraw-Hill, 2011.

STONEBRAKER, M.; et al. MapReduce and parallel DBMSs: friends or foes? Communications of the ACM, v. 53, n. 1, p. 64-71, 2010.

STONEBRAKER, M.; HELLERSTEIN, J. M. Readings in Database Systems. MIT Press, 2005.

TORRES, Claudio. A Bíblia do Marketing Digital: Tudo o que você queria saber sobre marketing e publicidade na internet e não tinha a quem perguntar. Novatec Editora, 2018.

Anexo 1 - Script de inserção.

DM_Insert.py ● DM_Remove.py ●

C: > Users > jeang > Downloads > DM_Insert.py > processar_csv

```
251 def processar_csv(arquivo_csv, arquivo_saida):
456     if infos_para_inserir:
457         # Dividir a geração de MassiveNumbers em lotes de no máximo 200
458         tamanho_max_lote_massive = 200
459         massive_numbers_total = []
460         total_inserts = len(infos_para_inserir)
461         idx_info = 0
462 
463         while idx_info < total_inserts:
464             n_massive_numbers = min(tamanho_max_lote_massive, total_inserts - idx_info)
465             massive_number_command = gerar_massive_number_batch(n_massive_numbers)
466             label = f"massive_numbers_lote{idx_info // tamanho_max_lote_massive}"
467             comandos_sql_massive = adicionar_separadores(massive_number_command, label=label)
468             # Print para verificar o select
469             # print(f"Select command for label {label}:\n{massive_number_command}\n")
470 
471             stdout_massive, stderr_massive = executar_comando_sqlplus(comandos_sql_massive)
472             if stderr_massive:
473                 with open(arquivo_saida, 'a', encoding='utf-8') as f:
474                     f.write(f"Erro ao executar comandos SQL:\n{stderr_massive}\n")
475 
476             # Parsear MassiveNumbers e atribuir aos infos
477             pattern_massive = re.compile(r'###INICIO_RESULTADO_(.*?)###(.*?)###FIM_RESULTADO_\1###', re.DOTALL)
478             result_blocks_massive = pattern_massive.findall(stdout_massive)
479             results_dict_massive = {label: content for label, content in result_blocks_massive}
480 
481             massive_numbers = parsear_massive_number_batch(results_dict_massive.get(label, ''))
482             massive_numbers_total.extend(massive_numbers)
483 
484             idx_info += n_massive_numbers
485 
486         if len(massive_numbers_total) != len(infos_para_inserir):
487             with open(arquivo_saida, 'a', encoding='utf-8') as f:
488                 f.write("Erro ao obter MassiveNumbers suficientes.\n")
489             for info in infos_para_inserir:
490                 info['error'] = f"Erro ao obter MassiveNumber para o equipamento {info['equipamento']}"
491         else:
492             # Atribuir MassiveNumbers aos infos
493             for info, massive_number in zip(infos_para_inserir, massive_numbers_total):
494                 info['massive_number'] = massive_number
495 
496             # Inserir danos massivos em lotes de no máximo 200 inserts
497             for idx, infos_batch in enumerate(dividir_lista(infos_para_inserir, tamanho_max_lote)):
498                 insert_commands = inserir_dano_massivo_batch(infos_batch)
499                 # Print para verificar os inserts
500                 # print(f"Insert commands for batch {idx}:\n{insert_commands}\n")
501 
502                 stdout_insert, stderr_insert = executar_comando_sqlplus(insert_commands)
503                 if stderr_insert and "ORA-" in stderr_insert:
504                     with open(arquivo_saida, 'a', encoding='utf-8') as f:
505                         f.write(f"Erro ao inserir danos massivos:\n{stderr_insert}\n")
506                     for info in infos_batch:
507                         info['error'] = f"Erro ao inserir dano massivo para o equipamento {info['equipamento']}"
508                 else:
509                     for info in infos_batch:
510                         info['mensagem_sucesso'] = f"Sucesso, DM inserido no equipamento: {info['equipamento']}, Localidade: {info['localidade']},
511                           Site: {info['site']}, MassiveNumber: {info['massive_number']}"
```

Anexo 2 - Script de remoção

DM_Insert.py ● DM_Remove.py ●

C: > Users > jeang > Downloads > DM_Remove.py > processar_csv

```
216  def processar_csv(arquivo_csv, arquivo_saida):
384      # Terceira fase: Verificar danos massivos existentes em lote
385      equipment_ids = [info['cr_equipment_id'] for info in batch_info if info['cr_equipment_id'] and not info['error']]
386
387      massive_data = {}
388      if equipment_ids:
389          for idx, equipment_ids_batch in enumerate(dividir_lista(equipment_ids, tamanho_max_lote)):
390              check_command = verificar_dano_massivo_batch(equipment_ids_batch)
391              label = f"check_massive_damage_lote{idx}"
392              comandos_sql_dm = adicionar_separadores(check_command, label=label)
393              # Print para verificar o select
394              # print(f"Select command for label {label}:\n{check_command}\n")
395
396              stdout_dm, stderr_dm = executar_comando_sqlplus(comandos_sql_dm)
397              if stderr_dm:
398                  with open(arquivo_saida, 'a', encoding='utf-8') as f:
399                      f.write(f"Erro ao executar comandos SQL:\n{stderr_dm}\n")
400
401              # Parsear resultados de verificação de dano massivo
402              pattern_dm = re.compile(r'###INICIO_RESULTADO_(.*?)###(.*?)###FIM_RESULTADO_\1###', re.DOTALL)
403              result_blocks_dm = pattern_dm.findall(stdout_dm)
404              results_dict_dm = {label: content for label, content in result_blocks_dm}
405
406              massive_data_batch = parsear_verificacao_dano_massivo_batch(results_dict_dm.get(label, ''))
407              massive_data.update(massive_data_batch)
408
409      for info in batch_info:
410          if info['error']:
411              continue  # Pular se já houve erro
412          if info['cr_equipment_id'] in massive_data:
413              info['massive_number'] = massive_data[info['cr_equipment_id']]
414          else:
415              info['error'] = f"Equipamento {info['equipamento']}, Localidade {info['localidade']}, Site {info['site']} nao tem dano massivo ativo."
416
417      # Quarta fase: Remover danos massivos em lote
418      infos_para_remover = [info for info in batch_info if not info.get('error')]
419
420      if infos_para_remover:
421          equipment_ids_remover = [info['cr_equipment_id'] for info in infos_para_remover]
422          for idx, equipment_ids_batch in enumerate(dividir_lista(equipment_ids_remover, tamanho_max_lote)):
423              delete_command = remover_dano_massivo_batch(equipment_ids_batch)
424              # Print para verificar o comando de remoção
425              # print(f"Delete command for batch {idx}:\n{delete_command}\n")
426
427              stdout_delete, stderr_delete = executar_comando_sqlplus(delete_command)
428              if stderr_delete and "ORA-" in stderr_delete:
429                  with open(arquivo_saida, 'a', encoding='utf-8') as f:
430                      f.write(f"Erro ao eliminar danos massivos:\n{stderr_delete}\n")
431                  for info in infos_para_remover:
432                      if info['cr_equipment_id'] in equipment_ids_batch:
433                          info['error'] = f"Erro ao eliminar dano massivo para o equipamento {info['equipamento']}"
434              else:
435                  for info in infos_para_remover:
436                      if info['cr_equipment_id'] in equipment_ids_batch:
437                          info['mensagem_sucesso'] = f"Sucesso, DM eliminado no equipamento: {info['equipamento']}, Localidade: {info['localidade']},
438                          Site: {info['site']}, MassiveNumber: {info['massive_number']}"
```