EasyBase - Uma Ferramenta Para Auxiliar na Otimização de Coleta de Estatística no SGBD *Oracle*

Vinicius Irale Cavalheiro

Orientadora: Tanisi Pereira de Carvalho

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul - *Campus* Porto Alegre (IFRS)

CEP 90.030-040 - Av Cel Vicente, 281, Porto Alegre, RS – Brasil

viniirale@hotmail.com, tanisi.carvalho@poa.ifrs.edu.br

Resumo: Os sistemas gerenciadores de banco de dados estão aumentando cada vez mais sua complexidade em virtude de sua ampla adoção em larga escala, com regras cada vez mais complexas neste meio. Isto faz com que administradores, desenvolvedores, estudantes e professores de cursos da área de tecnologia da informação, acabem encontrando barreiras em otimizar os processos de seu banco de dados devido às ferramentas e regras complexas existentes. Tendo em vista estas dificuldades no dia-a-dia de quem trabalha neste segmento, o sistema proposto pelo *EasyBase* tem por objetivo integrar-se diretamente ao banco de dados *Oracle* do cliente, e com o uso de inteligência artificial, sugerir coleta de estatísticas com base nas estruturas das tabelas e histogramas. Dentre suas funcionalidades, destaca-se a sugestão de inserção e alteração de índices, notificações para a coleta de estatísticas e histogramas, fazendo uso das práticas de otimização baseadas em custo. O objetivo central é fornecer uma ferramenta que, além de executar otimizações, também exponha para o usuário os processos executados que o levaram até essa otimização, exibindo práticas recomendadas para uma melhor manutenção de desempenho dos bancos *Oracle*.

Palavras-chave: CBO, Oracle, Otimização Por Custo, SQL, Tuning, UX.

1 Introdução

A criação e modelagem de bancos de dados utilizando *Structured Query Language* (*SQL*) é uma atividade executada comumente pelos desenvolvedores de sistemas e administradores de banco de dados - *Database Administrator* (*DBA*). Estes profissionais visam pela aplicação de boas práticas e otimização de desempenho das operações de um banco de dados, atividades estas que se tornam complexas e desafiadoras em virtude dos relacionamentos presentes nas bases de dados. (Poletto; Santos; Júnior, 2013).

Os Sistemas Gerenciadores de Banco de Dados (SGBD) atuais possuem documentações extensas e complexas, com muitos processos e regras específicas, constantemente atualizadas. Isso requer profissionais altamente qualificados e especializados, o que, segundo Poletto, dificulta a entrada e o aprendizado de novos profissionais. Mesmo com ferramentas de manuseio para banco de dados, com diversas otimizações e configurações, como SQL Developer e DBeaver, ainda se tem um deficit de orientação e melhores práticas no manuseio de SGBD (Bryla; Loney, 2009).

Tendo em vista os problemas citados acima, este trabalho tem como proposta desenvolver uma ferramenta para auxiliar os usuários especializados, como desenvolvedores e *DBAs* na otimização de consulta a partir de coleta de estatísticas no SGBD *Oracle* de forma instrutiva, denominado *EasyBase*. Este programa tem como propósito indicar falhas estruturais no banco de dados, como índices faltantes, falta de chaves estrangeiras e outras inconsistências. Além disso, ele exibe os comandos necessários para monitorar e melhorar as estatísticas do banco de dados, sugerindo o melhor horário para a coleta de estatísticas, de acordo com volume de tráfego do SGBD, processando os dados coletados com a inteligência artificial da *OpenAI* e gerando um relatório com base nas estatísticas do SGBD para o usuário especializado.

Com as funcionalidades descritas acima, o DBA terá um maior controle da sua base de dados, garantindo a manutenibilidade do SGBD, gerando *insights* relevantes para que a coleta de estatísticas seja realizada com o menor impacto possível aos usuários deste banco.

O artigo está organizado da seguinte forma: a seção 2 apresenta conceitos de bancos de dados *SQL* e *Oracle*, expondo a importância do *tuning* de consultas e os diferentes tipos de otimizadores deste tipo de operação. A seção 3 apresenta a comparação das aplicações analisadas em contraste com o *EasyBase*. Em seguida, a seção 4 descreve a metodologia utilizada para o desenvolvimento e elaboração deste trabalho de conclusão. Na seção 5 , serão descritas as tecnologias, modelagem do sistema e objetivos empregados no desenvolvimento da ferramenta proposta. Após, na seção 6 é exibida uma coleta de métricas relacionada a consultas ao banco de dados para observar os resultados obtidos com a aplicação proposta. E por ultimo, a conclusão deste trabalho se encontra na seção 7.

2 Fundamentação Teórica

Nesta seção serão explicadas as especificações e conceitos dos bancos de dados *SQL*, *Oracle*, assim como os dois tipos de otimizadores baseado em regra e custo e do software de inteligência artificial utilizado.

2.1 Banco de dados *SQL*

De acordo com Silberschatz, Sundarshan & Korth (2016) o *SQL* foi criado no começo dos anos 1974 por pesquisadores da *International Business Machines Corporation (IBM)*, Raymond Boyce e Donald Chamberlin com intuito de implementar relações entre dados. Isso possibilitou definir regras de negócios atreladas aos dados e realizar um armazenamento de forma eficiente (Silberschatz; Sundarshan; Korth, 2016).

Um banco de dados *SQL* é uma coleção de dados organizada e estruturada armazenados em um sistema de computador de acordo com Miletto *et al.* (2014), Oracle (2024b). Ele é controlado por um SGBD onde se utiliza a linguagem *SQL*. Esta linguagem conta com processos de *tuning* que significa otimizar o uso dos recursos do banco para aprimoramento da performance e tem como objetivos:

- Minimizar o tempo de resposta e recuperação de dados;
- Minimizar a concorrência de acesso aos dados;
- Otimizar a taxa de transferência;
- Otimizar a capacidade de carga do Banco de Dados;
- Utilização de histogramas para melhorar o otimizador.

Além disto, o SQL possuí diferentes SGBDs à sua disposição, como *Oracle*, *MySQL*, *PostgreSQL*, etc. E como mostrado no *DB-Engines (2024)*, o *Oracle* é o banco de dados mais utilizado do mundo, sendo este o fator de escolha para avaliação deste SGBD neste trabalho.

2.2 Banco de dados *Oracle*

O banco de dados *Oracle* é um SGBD relacional criado em 1977 por Larry Ellison, Bob Miner e Ed Oates. O objetivo era a criação de um *software* empresarial com foco no modelo de bancos de dados relacionais que na época, não era tão comercializado (Oracle, 2024a) e somente desenvolvedores altamente treinados da época conseguiam utilizar sistemas e softwares complexos para gerenciar os dados.

Após a sua criação, a *Oracle* teve uma grande adesão no mundo todo por ser uma solução mais acessível em comparação aos seus concorrentes da época (Oracle, 2024a). Este dispõe de uma documentação de *SQL tuning*, onde vemos um guia atualizado de boas práticas e execução de processos para aumentar a performance do seu banco de dados *SQL*, existem diferentes guias dependendo da versão do seu Banco *Oracle*.

Para aumentar a performance nas consultas, o *Oracle* implementou otimizadores de consulta cuja função é decidir o melhor plano de execução para a instrução *SQL*, essenciais para o *tuning*. No *Oracle* existem dois tipos de otimizadores, o *Rule-Based Optimizer* (*RBO*) e o *Cost-Based Optimizer* (*CBO*) (Nunes, 2009; Oracle, 2023), os quais iremos apresentar no capítulo a seguir.

2.3 Otimizadores baseados em regras (*RBO*)

O *RBO* leva em consideração regras internas para decidir o melhor caminho de execução. Ao encontrar a primeira rota possível, executa-a, ignorando outras direções viáveis, sem verificar se existem alternativas que oferecem melhor eficiência. *RBO* não realiza cálculo de custo, não utiliza estatísticas nem histogramas. Nunes (2009) adverte que a *Oracle* deixou de oferecer suporte ao *RBO* desde a versão *Oracle* 10g em 2003 e o fornece somente ao *CBO*. Entretanto, o *Oracle* mantém a funcionalidade de *RBO* para os usuários especializados que a utilizam,para não ter problema de compatibilidade.

2.4 Otimizadores baseados em custo (*CBO*)

O *CBO* utiliza estatísticas e histogramas disponibilizados pelo SGBD, onde contém dados estatísticos de tabelas, colunas, índices e informações do sistema, como processador e memória, que estão armazenados no banco de dados (Nunes, 2009). Com base nos histogramas e dados estatísticos gerados, o otimizador gera um conjunto de planos de execução para comparar e escolher o que possui o menor custo *Oracle* (Nunes, 2009; Prado, 2024).

Este tipo de otimizador conta com um recurso de histograma, um recurso exclusivo do *CBO*, que ajuda o otimizador de consultas a estimar com mais precisão a cardinalidade, ou seja, o número de linhas que uma consulta ou um filtro específico retornará. Uma estimativa de cardinalidade precisa é essencial para o *Oracle* escolher o melhor plano de execução (Nunes, 2009; Prado, 2024). O histograma pode ser definido somente nas colunas que possuem índices, onde ele analisa a distribuição dos dados nas colunas. Caso a distribuição de uma coluna for altamente irregular, ou seja, que contenha muitos valores diferentes (conhecido como distribuição com *skew*), é mais provável que o *Oracle* decida criar um histograma para essa coluna (Prado, 2024). Conforme explicado por Prado (2024), o otimizador baseado em custo executa uma consulta 44,4% mais rápida em comparação ao otimizador baseado em regra. Ele também

adverte que as estatísticas devem estar atualizadas para que o SGBD *Oracle* siga um plano de execução eficiente.

2.5 *Chat Generative Pre-trained Transformer* (*ChatGPT*)

ChatGPT é um *chatbot* que utiliza inteligência artificial desenvolvido pela *OpenAI*, um laboratório de pesquisa de inteligência artificial(IA) estadunidense, que conduz pesquisas visando promover e desenvolver uma IA de fácil entendimento (OpenAI, 2024). Atualmente o *ChatGPT* dispõe de uma Interfaces de Programação de Aplicação - *Application Programming Interface*(*API*) - e é uma das mais utilizadas segundo *Forbes (2024)*.

A configuração da API do chatGPT fornecida OpenAI, conforme Alves (2023) mostra suas funcionalidades, contando com uma versão gratuita que dá acesso à versão *GPT-3.5* e outra versão paga que dá acesso às versões *GPT-4* e *GPT-4o* que realizam análises de dados mais avançados. As aplicações relacionadas utilizam a *API* referente ao *GPT-3.5* para diversos propósitos e o *EasyBase* fornece dados das estatísticas do SGBD do usuário especializado para a *API* da *OpenAI*, que processa os dados, e então o SGBD coleta estes dados para gerar um relatório personalizado.

3 Aplicações Relacionadas

Ao realizar o levantamento de aplicações relacionadas, foram encontrados quatro aplicativos com funcionalidades similares a desta proposta que serão apresentados a seguir.

O ***AskYourDatabase*** é uma plataforma com foco no ambiente empresarial, oferecendo funcionalidades como *dashboards*, códigos para criação de banco de dados e *report* pautado em regras de negócio. Disponível nas versões *web* e *desktop*, ambas com interface intuitivas, a plataforma permite que o usuário especializado execute tarefas por meio de comandos inseridos no *prompt*. O serviço não possui uma versão gratuita ou de testes, e seus planos de assinatura variam entre $29 e $49 mensais, que variam conforme a versão da *API* do *OpenAI* utilizada. Além de ser compatível com banco de dados relacionais *SQL*, ele também suporta banco de dados não relacionais.

O ***SQLAI.ai*** além de oferecer otimização de consulta, ele também gera códigos *SQL* a partir de textos digitados pelo usuário, cria gráficos informativos a partir das consultas *SQLs* digitadas no sistema, além de possuir o diferencial de sempre utilizar a *API* do *GPT-4* em todas as suas requisições, gerando um resultado mais eficaz que a versão base do *GPT-3.5*. Este aplicativo possui um plano de teste de sete dias e com preços mais acessíveis, variando entre $6 à $18 dólares mensais. Ele também suporta *Oracle* e outros tipos de bancos relacionais e não relacionais.

Já ***DB Sensei*** gera automaticamente consultas complexas, corrige erros, explica o funcionamento das consultas e formata o código para melhor legibilidade, além de permitir a importação ilimitada de bancos de dados. Ele possui uma versão de teste de sete dias e três opções pagas, onde os preços variam entre $9 à $29 dólares mensais. *DB Sensei* é compatível com diversos bancos de dados além do *Oracle*.

EverSQL é uma ferramenta que otimiza consultas utilizando IA e algoritmos próprios. Além disso, oferece a possibilidade do usuário especializado exportar informações do banco manualmente como estruturas do seu esquema, relatório das consultas, envio de estatísticas e histogramas, onde a partir destas informações ele realiza recomendações para reduzir custos como a remoção de índices redundantes e otimizações de esquema. Apesar de existir um plano gratuito limitado, os planos pagos começam em $129 dólares podendo ultrapassar $2000 mensais nos seus planos mais avançados, contudo, o aplicativo não possuí integração para o *Oracle*.

Tendo em vista os principais pontos de otimização por custo abordados pelo Nunes (2009), foram escolhidas as seguintes funcionalidades a serem comparadas no Quadro **1**.

Quadro 1 – Quadro comparativo

Funcionalidades	***AskYourDatabase***	***SQLAI.ai***	***DB Sensei***	***EverSQL***	***EasyBase***
Integração com banco de dados	✓	✓	✓		✓
Sugestão de melhorias de índices	✓	✓	✓	✓	✓
Monitoramento e sugestão de estatísticas				✓	✓
Compatível com banco *Oracle*	✓	✓	✓		✓
Todas as funcionalidades gratuitas					✓

Fonte: Elaborado pelo autor (2024).

O *AskYourDatabase*, *SQLAI.ai* e *DB Sensei* possuem conexão com o banco de dados do cliente diretamente. Com exceção do *EverSQL*, os outros aplicativos não fazem uma otimização sugerindo coletas de dados, análise de estatísticas e histogramas das tabelas. Entre os quatro aplicativos, todos estão disponíveis somente na língua inglesa, além deles oferecerem a sugestão de melhorias de índices.

Dessa forma, o aplicativo ***EasyBase*** se destacará por ser projetado para ser simples, realizando a otimização com poucas instruções e disponibilizando explicações claras sobre os processos executados, voltadas para o usuário especializado. Além disso, será relevante por ser uma opção totalmente gratuita e disponibilizada em português.

4 Metodologia

Para o desenvolvimento do *EasyBase*, foi realizada uma pesquisa exploratória do *Google* Acadêmico, utilizando um protocolo de revisão bibliográfica. Foram levantados estudos sobre otimização baseada em custo, *tuning* de consultas no banco de dados *Oracle* e artigos que fundamentaram o trabalho, como o de Nunes (2009) que explica os ganhos do uso de otimizadores baseado em custo, comparando-os com os otimizadores baseado em regra. Além disso, realizou-se uma pesquisa exploratória na internet para identificar aplicativos similares ao *EasyBase*.

Com base na pesquisa descrita, deu-se inicio ao levantamento de requisitos funcionais que vão estar no aplicativo *EasyBase*, seguido da elaboração do diagrama de casos de uso na Figura 1, levando em conta o tempo disponível para a elaboração das entregas de funcionalidades do *EasyBase*.

4.1 Tecnologias utilizadas

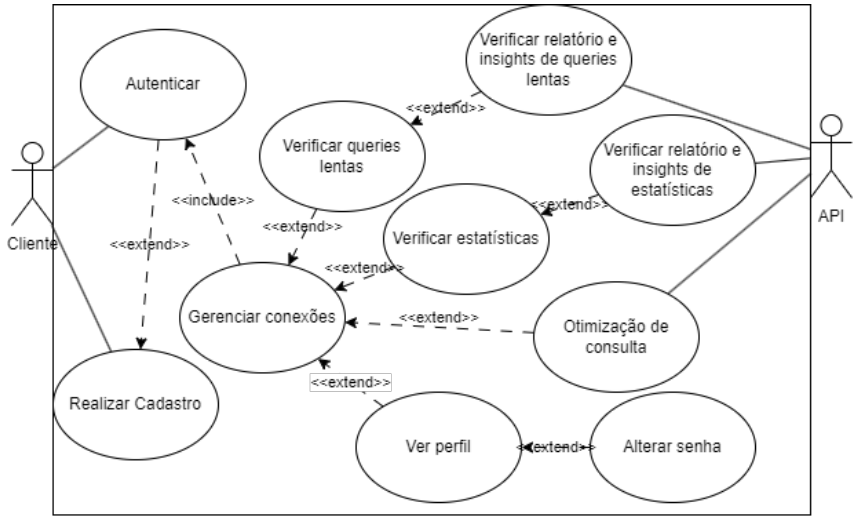
Entre as tecnologias que estão empregadas no desenvolvimento do aplicativo *EasyBase*, estão a codificação através da *Integrated Software Environment* (IDE) *Visual Studio Code*, onde o programa foi codificado utilizando a linguagem Python com o *framework Django* para desenvolvimento do *Front-end* e do *Back-end*. A escolha do *Django* se justifica por sua excelente conectividade com o banco de dados *Oracle* e pela disponibilidade nativa de uma interface de login, o que facilita o desenvolvimento da aplicação.

Além disso, foi utilizada a *API* do *OpenAI ChatGPT* 3.5 que pode ser acessada gratuitamente, embora com limitações de uso. O banco de dados escolhido, o *Oracle*, está alinhado com os objetivos propostos no artigo. O diagrama de casos de uso foi elaborado com o uso da ferramenta online *DrawIo* e, para o gerenciamento de projeto e versionamento, foi usado o *GitHub*.

5 O Aplicativo *EasyBase*

Este capítulo detalha o projeto da ferramenta *EasyBase*, suas principais funcionalidades, a proposta de modelagem do sistema e o funcionamento do programa disponível com diagrama de caso de uso.

Figura 1 – Diagrama de casos de uso do sistema



Fonte: Elaborado pelo autor (2024).

O diagrama de casos de uso possui o cliente e a *API* do *OpenAI* como atores, onde o cliente pode realizar seu cadastro, autenticação e alteração de senha para ter suas funcionalidades de gerenciamento de conta. O cliente será capaz de gerenciar suas conexões e verificar suas estatísticas de desempenho de cada banco, obtendo informações de possíveis otimizações a partir do sistema. Já o ator da *API*, realiza a coleta de informações das estatísticas e histogramas e retorna possíveis melhorias ao usuário especializado.

5.1 Requisitos funcionais

O intuito do *EasyBase* é auxiliar o usuário especializado na otimização de sua consulta, executando o processo de otimização e expondo as etapas processadas, sendo assim, a ferramenta deve atender os seguintes requisitos:

a) Integração do banco de dados do cliente com o *EasyBase*;

b) Sugestão de melhoria de consulta, tendo conhecimento das estruturas atuais do banco de dados do usuário especializado, como tabelas, índices, estatísticas;

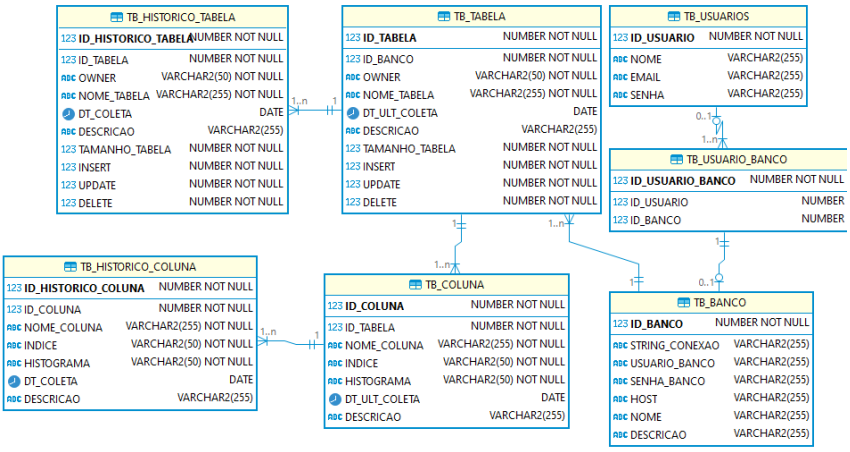
c) Sugestão de inserção e alteração de índices, baseado nas estatísticas das colunas;

d) Envio de notificações para realizar a coleta de estatística com base na data da última coleta além de sugerir a melhor janela de execução da coleta de estatística, conforme o tráfego do SGBD;

5.2 Modelagem do sistema

O *EasyBase* conecta-se no SGBD do usuário especializado via *string* de conexão, onde os dados são armazenados consoante o modelo de banco de dados descrito no diagrama lógico, apresentado na **Figura 2**.

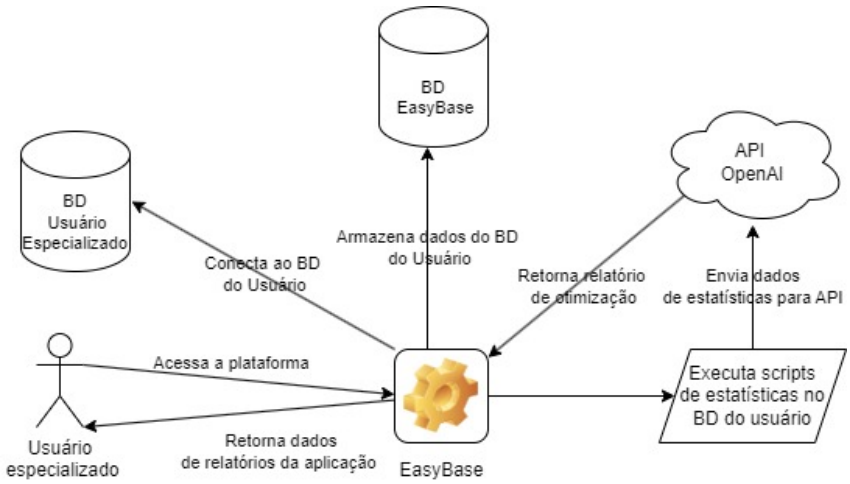
Figura 2 – Diagrama lógico da aplicação com o banco do usuário especializado.



Fonte: Elaborado pelo autor (2024).

O diagrama lógico representa as tabelas do aplicativo *EasyBase*, onde cada usuário especializado possui uma ou mais conexões de banco de dados seguindo a cardinalidade 1:N. A conexão se mostra necessária, pois o *EasyBase* utiliza informações do SGBD do usuário para a coleta de estatísticas e fornece essas informações para a *API* da *OpenAI*, onde processa os dados e retorna relatórios das estatísticas com sugestões de melhorias, como inserção de índices, aprimoramento em consultas que estão gerando sobrecarga para o banco, além do resumo das estatísticas das tabelas e colunas.

Figura 3 – Esquema do sistema



Fonte: Elaborado pelo autor (2024).

O esquema do sistema descrito na **Figura 7** exibe o funcionamento do sistema, onde o usuário especializado acessa o *EasyBase*, cadastra o seu SGBD para que a aplicação execute os comandos *SQLs*, para obter os dados do banco do usuário especializado. Então, os dados são enviados para a *API* da *OpenAI* para serem processados e, a partir daí, são gerados relatórios sobre o SGBD. Em seguida, o *EasyBase* retorna para o usuário os comandos necessários para efetuar a coleta de estatísticas e outras informações para otimizar suas consultas.

5.3 Códigos *SQLs* utilizados

Para o *EasyBase* obter as informações das últimas coletas de estatísticas das tabelas, ele executa os seguintes comandos *SQL* no banco de dados do usuário especializado, conforme a **Figura 4**.

Figura 4 – Comando a ser executado pelo *EasyBase* para a coleta de informações das estatísticas e tamanho das tabelas:

```
-- Este comando obtem informacoes das ultimas estatisticas
SELECT owner , table_name, last_analyzed
FROM DBA_TAB_STATISTICS
WHERE object_type = 'TABLE'; -- seleciona somente os objetos do tipo TABELA

--Este comando exibe os horarios onde o SGBD possui menos carga
WITH horas AS (
SELECT LEVEL - 1 AS HORA
FROM DUAL
CONNECT BY LEVEL <= 24)
SELECT
LPAD(horas.HORA, 2, '0') || '-' || NVL(SESSION\_COUNT.SESSIONS, 0)
AS "HORAS|SESSOES"
FROM horas
LEFT JOIN ( SELECT TO_CHAR(LOGON_TIME, 'HH24') AS HORA,
            COUNT(*) AS SESSIONS
            FROM V$SESSION GROUP BY TO_CHAR(LOGON_TIME, 'HH24')) SESSION_COUNT
ON horas.HORA = SESSION_COUNT.HORA
ORDER BY horas.HORA;

--este comando obtem informacoes sobre histogramas das colunas
SELECT  owner, table_name, column_name , num_distinct , histogram, last_analyzed
FROM DBA_TAB_COL_STATISTICS;

-- Este comando obtem o numero de linhas contendo na tabela
SELECT count(*)
FROM <nome_tabela>;
```

Fonte: Elaborado pelo autor (2024).

Onde:

a) ***owner***: Nome do proprietário da tabela;

b) ***table_name***: Nome da tabela;

c) ***last_analyzed***: Última coleta de estatística ou do histograma;

d) ***object_type***: para retornar somente objetos do tipo Tabela;

e) ***column_name***: Nome da coluna;

f) ***num_distinct***: Retorna o numero de dados distintos da coluna, o que vai influenciar no tipo de histograma que pode ser criado;

g) ***histogram***: Valor que retorna o tipo de histograma, que pode ser híbrido, frequência, balanceado ou vazio, caso a coluna não tenha um histograma definido;

h) ***count***: Retorna o número de registros da tabela.

Essas informações são fundamentais, pois o primeiro *script SQL* apresentado na **Figura 4** serve para verificar quando foi executada a última coleta de estatística. Este código procura verificar a data e o horário da última coleta de estatísticas realizada. O segundo *script* gera uma lista contendo a média de acessos por intervalo de tempo, permitindo ao usuário especializado identificar o período de menor carga no Sistema de Gerenciamento de Banco de Dados (SGBD). Isso possibilita a recomendação de uma janela de execução para a coleta de estatísticas, caso o

tempo desde a última execução seja elevado. Na execução do terceiro *script*, o programa fornece informações sobre a presença de histogramas nas colunas das tabelas e os valores distintos de cada coluna. No *Oracle*, o tipo de histograma é definido automaticamente durante a coleta de estatísticas, visando otimizar o desempenho das consultas. E por último, o quarto *script* realiza a contagem do número de registros em uma tabela específica, dado que essa informação é essencial para obter a estimativa de tempo necessário para a execução da coleta de estatísticas.

Após a execução dos *scripts*, o *EasyBase* fornece ao usuário especializado uma recomendação sobre o comando de coleta mais adequado, uma estimativa do tempo de execução e a melhor janela de tempo para minimizar o impacto sobre o desempenho do SGBD.

Figura 5 – Comando a ser executado pelo usuário especializado para executar a coleta de estatísticas

```
DECLARE -- Este comando obtem a coleta de estatisticas
  nome_schema VARCHAR2(30) := 'SCHEMA_EXEMPLO';
  nome_tabela  VARCHAR2(30) := 'TABELA_EXEMPO';
BEGIN
  DBMS_STATS.GATHER_TABLE_STATS(
    ownname          => nome_schema,
    tabname          => nome_tabela,
    cascade          => TRUE,
    method_opt       => 'FOR ALL COLUMNS SIZE AUTO',
    degree           => DBMS_STATS.DEFAULT_DEGREE,
    granularity      => 'AUTO',
    estimate_percent => DBMS_STATS.AUTO_SAMPLE_SIZE,
    no_invalidate    => FALSE);
  DBMS_OUTPUT.PUT_LINE('Estatisticas coletadas.');
EXCEPTION --Tratamento de excecoes
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Erro ao coletar estatisticas: '||SQLERRM);
END;/
```

Fonte: Elaborado pelo autor (2024).

Onde:

a) ***ownname***: Nome do proprietário da tabela;

b) ***tabname***: Nome da tabela;

c) ***cascade***: Quando ***TRUE***, também coleta estatísticas dos índices da tabela;

d) ***method_opt***: Define as opções de coleta de estatísticas para colunas, o parâmetro ***FOR ALL COLUMNS SIZE AUTO*** coleta estatísticas de todas as colunas e o *Oracle* decide automaticamente quais colunas terão histogramas e quantos *buckets* serão usados;

e) ***degree***: define o grau de paralelismo. ***DBMS_STATS.DEFAULT_DEGREE*** usa o grau de paralelismo padrão;

f) ***granularity***: define a granularidade da coleta. ***AUTO*** permite que o *Oracle* determine a granularidade;

g) ***estimate_percent***: especifica a porcentagem de linhas a serem amostradas para a coleta de estatísticas. Quando você define em ***DBMS_STATS.AUTO_SAMPLE_SIZE***, permite que o *Oracle* determine a melhor porcentagem de amostragem para garantir a coleta de estatísticas precisas sem afetar significativamente o desempenho. O *Oracle* usa algoritmos internos para decidir a amostragem, balanceando precisão e custo de coleta;

h) *no_invalidate*: quando ***FALSE***, invalida os cursores dependentes após a coleta de estatísticas.

Após a execução dos *scripts* de coleta de dados das tabelas, nos dados coletados, dados de histogramas e outras informações do banco, o *EasyBase* junta as informações e formula uma frase de entrada padrão para enviar à *API* do *OpenAI*. Esta entrada será melhor detalhada na próxima seção, mostrando a geração de *insights* e recomendações personalizadas para o ambiente do banco de dados do cliente.

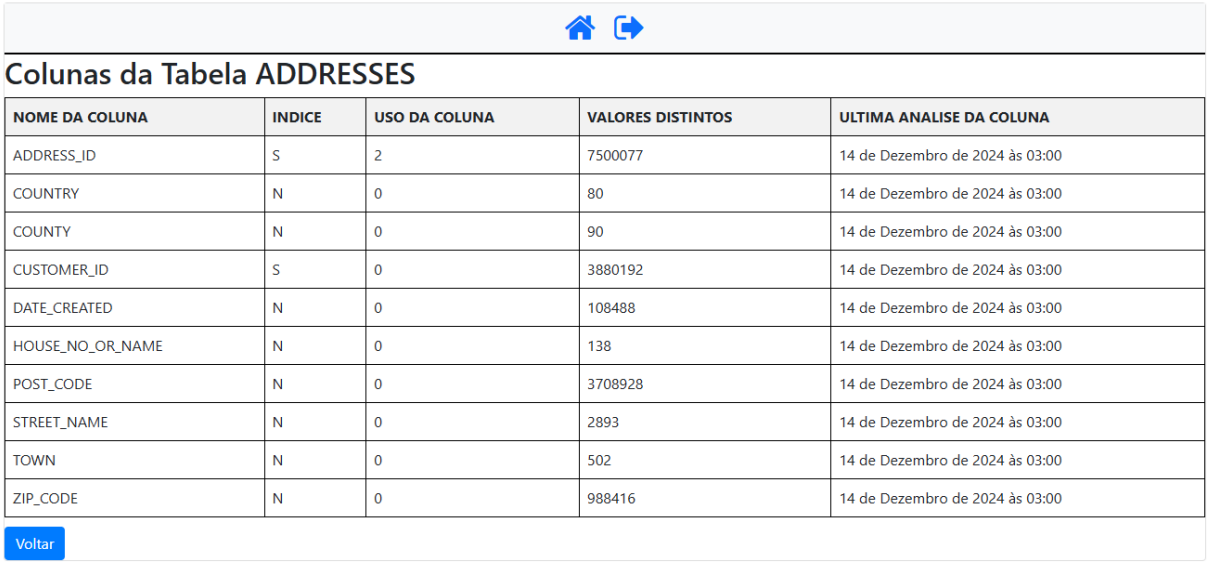
O *EasyBase* lista todas as colunas, depois exibe se elas são índices ou não, juntamente com suas métricas de uso baseadas nos planos de execuções de todos os usuários do banco de dados. Além disso, ele mostra a quantidade de valores distintos que a coluna possui naquele momento, baseado em sua coleta de estatística, como indicado na **Figura 6**.

Figura 6 – Consulta de uso de índice

```
-- Este comando coleta informacoes de uso de indice
WITH indexedcolumns AS
(   SELECT c.column_name,
    CASE WHEN i.column_name IS NOT NULL THEN 'S' ELSE 'N' END AS indice
    , c.num_distinct
    , c.last_analyzed FROM dba_tab_columns c
    LEFT JOIN user_ind_columns i ON c.table_name = i.table_name
    AND c.column_name = i.column_name
    WHERE c.table_name = 'TABELA_EXEMPO'
),
indexusage AS
(   SELECT ic.column_name
    , COUNT(*) AS usage_count
    FROM dba_ind_columns ic
    JOIN  v$sql_plan p ON ic.index_name = p.object_name
    WHERE
    ic.table_name      = 'TABELA_EXEMPO'
    AND p.object_owner = 'SCHEMA_EXEMPLO'
    GROUP BY ic.column_name
),
predicateusage AS
(   SELECT c.column_name
    , COUNT(*) AS usage_count FROM dba_tab_columns c
    JOIN  v$sql_plan p ON (
    p.filter_predicates LIKE '%'|| c.column_name || '%'
    OR p.access_predicates LIKE '%' || c.column_name || '%'
    ) WHERE c.table_name       = 'TABELA_EXEMPO'
    AND p.object_owner = 'SCHEMA_EXEMPLO'
    GROUP BY c.column_name
)
SELECT ic.column_name AS "NOME␣DA␣COLUNA"
, ic.indice
, CASE WHEN ic.indice = 'S' THEN NVL(iu.usage_count, 0) ELSE NVL(pu.usage_count, 0)
END AS "USO␣DA␣COLUNA"
, ic.num_distinct AS "VALORES␣DISTINTOS"
, ic.last_analyzed AS "ULTIMA␣ANALISE␣DA␣COLUNA" FROM
indexedcolumns ic
LEFT JOIN indexusage iu     ON ic.column_name = iu.column_name
LEFT JOIN predicateusage pu ON ic.column_name = pu.column_name
ORDER BY ic.column_name
```

Fonte: Elaborado pelo autor (2024).

Figura 7 – Tela Apresentando o Resultado das Colunas



Colunas da Tabela ADDRESSES

NOME DA COLUNA	INDICE	USO DA COLUNA	VALORES DISTINTOS	ULTIMA ANALISE DA COLUNA
ADDRESS_ID	S	2	7500077	14 de Dezembro de 2024 às 03:00
COUNTRY	N	0	80	14 de Dezembro de 2024 às 03:00
COUNTY	N	0	90	14 de Dezembro de 2024 às 03:00
CUSTOMER_ID	S	0	3880192	14 de Dezembro de 2024 às 03:00
DATE_CREATED	N	0	108488	14 de Dezembro de 2024 às 03:00
HOUSE_NO_OR_NAME	N	0	138	14 de Dezembro de 2024 às 03:00
POST_CODE	N	0	3708928	14 de Dezembro de 2024 às 03:00
STREET_NAME	N	0	2893	14 de Dezembro de 2024 às 03:00
TOWN	N	0	502	14 de Dezembro de 2024 às 03:00
ZIP_CODE	N	0	988416	14 de Dezembro de 2024 às 03:00

Voltar

Fonte: Elaborado pelo autor (2024).

O usuário especializado pode tomar a decisão de inserir ou remover um índice, baseado nas informações fornecidas pelo *EasyBase*.

5.4 Comunicação com o *GPT*

A partir dos *scripts* mencionados na seção anterior, o *EasyBase* estrutura os *prompts* para enviá-los a *API* do *ChatGPT*. Nota-se que o prompt irá mudar caso o usuário tiver acesso administrativos do banco de dados, onde ele terá uma gama maior de comandos para retornar informações, como o horário mais indicado para realizar a coleta de estatística.

Figura 8 – Código utilizado para estruturar o *prompt* do *GPT*

```
# Construir a mensagem do prompt

if conexao_db.usuario_banco.upper() == 'SYS':
    prompt = f"""
    Olá! eu executei esta query no meu banco de dados oracle no owner {owner} "{query1}" e obtive como retorno da tabela
    {resultado_query1}

    Também executei a query para verificar quais as últimas alterações que a tabela teve
    "{query2}" e obtive o seguinte resultado:
    {resultado_query2}

    Também executei esta query para obter o melhor intervalo do dia para fazer a estatística com base no menor uso do banco
    "{query3}"
    E recebi o seguinte resultado:
    {resultado_query3}

    Considerando os dados, acha que é necessário executar a coleta de estatística?  caso sim diga qual seria o melhor horário para executar as estatísticas e
    pode me oferecer o código para efetuar a coleta de estatísticas?
    Caso não precise efetuar a coleta de estatísstica, pode me dizer o motivo?
    """
else:
    prompt = f"""
    Olá! eu executei esta query no meu banco de dados oracle no owner {owner} "{query1}" e obtive como retorno da tabela
    {resultado_query1}

    Também executei a query para verificar quais as últimas alterações que a tabela teve
    "{query2}" e obtive o seguinte resultado:
    {resultado_query2}

    Considerando os dados, acha que é necessário executar a coleta de estatística?  Caso sim pode me oferecer o código para efetuar a coleta de estatísticas?
    Caso não precise efetuar a coleta de estatísstica, pode me dizer o motivo?
    """
# resposta retornada da API do ChatGPT após o envio do prompt
resposta_chatgpt = chatgpt_api_call(prompt)
```

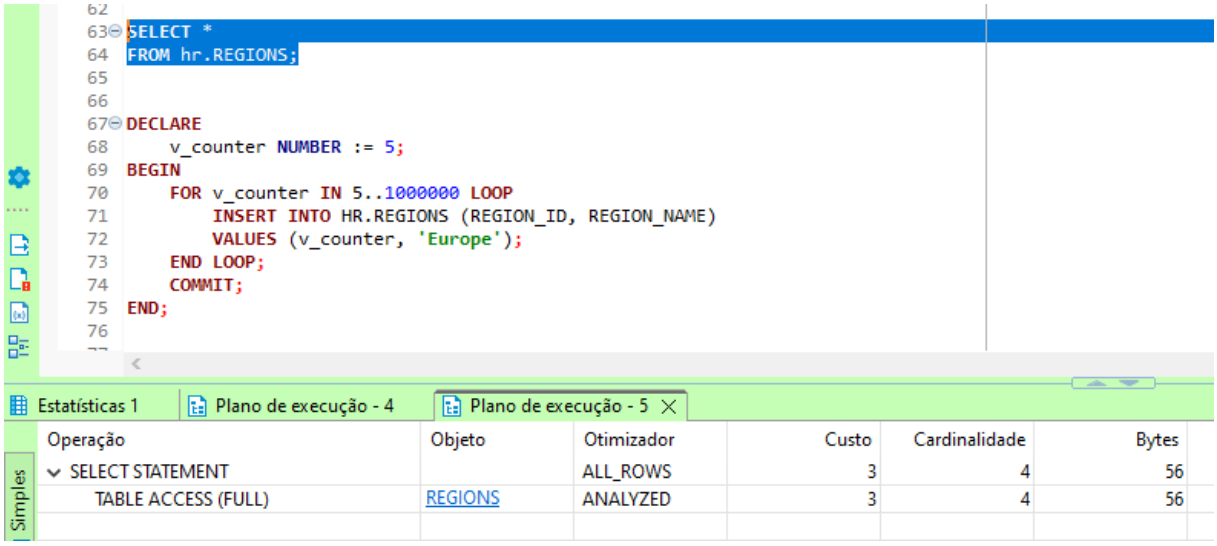
Fonte: Elaborado pelo autor (2024).

As telas do aplicativo com o resultado gerado pela API encontram-se esquematizadas no apêndice A. Após clicar para exibir o relatório do *GPT*, o *EasyBase* informa que a tabela selecionada possuí um número grande de dados na tabela que as estatísticas foram coletadas 3 meses atrás, porém não houve inserção ou alteração de dados então ele recomenda não fazer a coleta de estatística. Apesar disto, ele indica o comando caso o usuário especializado queira executar manualmente.

6 Métricas Coletadas

Para demonstrar a importância da coleta de estatística, usamos de exemplo a tabela ”HR.REGIONS”onde constava 4 registros diferentes, após a inserção de um milhão de registros, o plano de execução continua como se estivesse apenas com 4 valores, o que pode fazer o Oracle tomar uma decisão ruim e gerar uma consulta mais lenta.

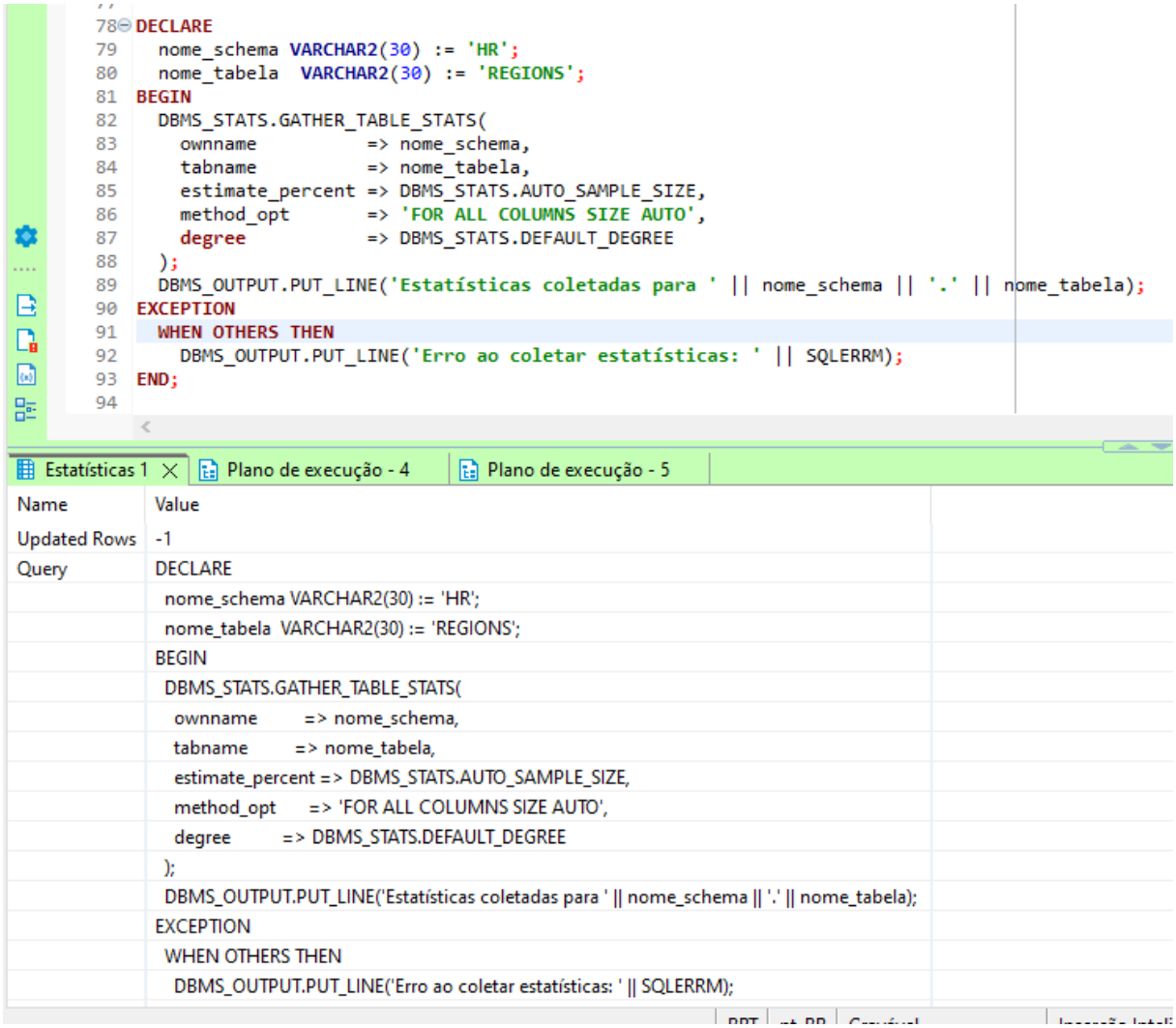
Figura 9 – Plano de execução antes da coleta de estatística



Fonte: Elaborado pelo autor (2024).

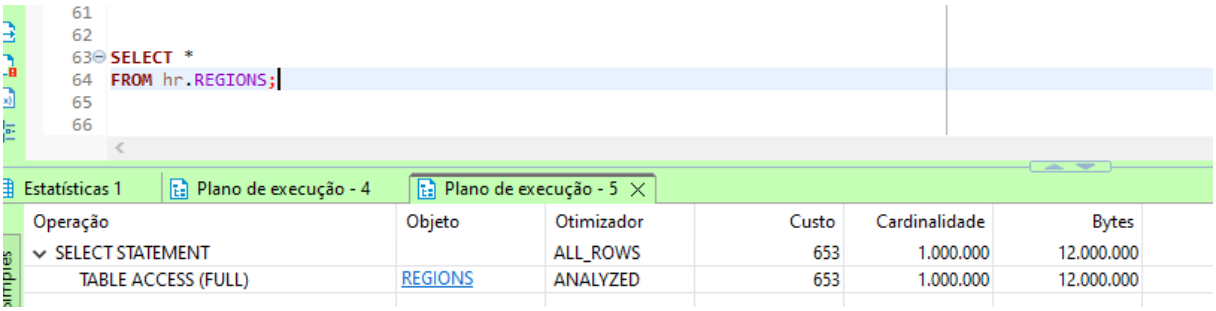
Após a execução da coleta de estatística, vimos que os valores de custo, cardinalidade e *bytes* foram devidamente atualizados nas figuras 10 e 11.

Figura 10 – Execução da Coleta de Estatística



Fonte: Elaborado pelo autor (2024).

Figura 11 – Plano de execução após coleta



Fonte: Elaborado pelo autor (2024).

7 Conclusão

O *EasyBase* cumpre com seu propósito de oferecer recomendações assertivas e baseadas em dados concretos, melhorando a tomada de decisão do usuário especializado. Podemos observar na seção 6 que seguindo os passos sugeridos pela aplicação, houve melhoria nos processos de consulta ao banco de dados. A facilidade de uso do programa é um de seus pontos fortes, além de incentivar seus usuários no uso de otimização de estatísticas e histogramas de acordo com as boas práticas do *tuning* guide da *Oracle*. Tendo em vista que a aplicação também faz as sugestões de horários para a coleta de estatística, é garantido que o processo de coleta para a otimização das consultas terá seu impacto minimizado, evitando problemas de acesso/lentidão a base de dados e, portanto, garantindo uma boa prática segura.

Conforme recebemos novas funcionalidades nos SGBDs e atualizações em seus processos, é importante garantirmos que o processo de atualização tecnológica não impacte o dia-a-dia do trabalho de um DBA. Neste sentido, ferramentas de caráter assistencialista como o *EasyBase* devem ganhar força nesse cenário, agilizando processos e garantindo a manutenibilidade de SGBDs de forma fácil, prática e segura.

Referências

ALVES, A. **Introdução à API da OpenAI**. 2023. Disponível em: https://www.researchgate.net/publication/374165121_Introducao_a_API_da_OpenAI. Acesso em: 03 nov. 2024.

ASKYOURDATABASE. **AskYourDatabase Página Inicial**. 2024. Disponível em: https://www.askyourdatabase.com/. Acesso em: 16 jul. 2024.

BRYLA, B.; LONEY, K. **Oracle Database 11g: manual do DBA**. Biblioteca do Instituto de Informática UFRGS Porto Alegre, RS: Bookman Editora, 2009.

DB-ENGINES. **DB-Engines Ranking Database**. 2024. Disponível em: https://db-engines.com/en/ranking. Acesso em: 16 jul. 2024.

EVERSQL. **EverSQL Página Inicial**. 2024. Disponível em: https://db-engines.com/en/ranking. Acesso em: 16 jul. 2024.

FORBES. **10 AI Tools In 2024**. 2024. Disponível em: https://www.forbes.com/advisor/business/ai-tools/. Acesso em: 16 jul. 2024.

MILETTO, E. M.; BERTAGNOLLI, S. D. C.; HUBLER, P. N.; CARVALHO, T. P. D.; NICOLAO, M. **Desenvolvimento de Software II: Introdução ao Desenvolvimento Web com HTML, CSS, JavaScript e PHP-Eixo: Informação e Comunicação-Série Tekne**. [S.l.]: Bookman Editora, 2014. Disponível em: https://www.researchgate.net/publication/263220660_Desenvolvimento_de_Software_II_-_Introducao_ao_Desenvolvimento_Web_com_HTML_CSS_JavaScript_e_PHP. Acesso em: 03 nov. 2024.

NUNES, J. P. G. **Diferenças entre os otimizadores de consulta do banco de dados Oracle baseado em regras e baseado em custo**. 2009. Disponível em: https://lyceumonline.usf.edu.br/salavirtual/documentos/1673.pdf. Acesso em: 03 nov. 2024.

OPENAI. **About us OpenAI**. 2024. Disponível em: https://openai.com/about/. Acesso em: 13 dez. 2024.

ORACLE. **SQL Tuning Guide**. 2023. Disponível em: https://docs.oracle.com/en/database/oracle/oracle-database/21/tgsql/sql-tuning-guide.pdf. Acesso em: 16 jul. 2024.

ORACLE. **Defying Conventional Wisdom**. 2024. Disponível em: https://www.oracle.com/us/corporate/profit/p27anniv-timeline-151918.pdf. Acesso em: 13 dez. 2024.

ORACLE. **O que é um Banco de Dados?** 2024. Disponível em: https://www.oracle.com/br/database/what-is-database/. Acesso em: 16 jul. 2024.

POLETTO, A. S. R. de S.; SANTOS, E. C.; JÚNIOR, J. R. de A. **Avaliação do grau de difuldade do aprendizado da modelagem de banco dados**. 2013. Disponível em: https://turing.pro.br/anais/COBENGE-2013/pdf/116141_1.pdf. Acesso em: 03 nov. 2024.

PRADO, F. **Coletando estatísticas para o otimizador de queries do Oracle**. 2024. Disponível em: https://www.fabioprado.net/2012/04/coletando-estatisticas-para-o.html. Acesso em: 05 ago. 2024.

SENSEI, D. **DB Sensei Página Inicial**. 2024. Disponível em: https://dbsensei.com/. Acesso em: 16 jul. 2024.

SILBERSCHATZ, A.; SUNDARSHAN, S.; KORTH, H. **Sistema de banco de dados**. [S.l.]: Elsevier Brasil, 2016. Disponível em: https://www.researchgate.net/publication/374165121_Introducao_a_API_da_OpenAI. Acesso em: 03 nov. 2024.

Apêndices

APÊNDICE A

Fonte: Elaborado pelo autor (2024).

APÊNDICE A

Com base nos dados fornecidos, parece que a tabela 'VENDAS_MENSAIS_PRODUTO' possui um número significativo de linhas (52.077) e a última vez que as estatísticas foram coletadas foi em 18 de setembro de 2024. No entanto, não há registro de modificações na tabela desde então.

Por padrão, o Oracle coleta automaticamente estatísticas quando a tabela é criada ou quando uma quantidade significativa de dados é modificada. Como não houve nenhuma modificação na tabela desde a última coleta de estatísticas, pode não ser necessário executar uma nova coleta nesse momento.

No entanto, se você deseja forçar uma nova coleta de estatísticas para garantir que o otimizador de consultas tenha informações atualizadas, você pode executar o seguinte código:

```
BEGIN
DBMS_STATS.GATHER_TABLE_STATS(
ownname => 'VENDAS_MENSAIS_PRODUTO',
tabname => 'VENDAS_MENSAIS_PRODUTO',
estimate_percent => DBMS_STATS.AUTO_SAMPLE_SIZE,
method_opt => 'FOR ALL COLUMNS SIZE AUTO',
cascade => TRUE);
END;
/
```

Isso irá coletar estatísticas para a tabela 'VENDAS_MENSAIS_PRODUTO' e suas colunas, utilizando um tamanho de amostra automático e atualizando todas as estatísticas relacionadas. Por favor, verifique se essa ação é necessária antes de executá-la, pois a coleta frequente de estatísticas pode afetar o desempenho do sistema.

Fonte: Elaborado pelo autor (2024).