Trabalho de Banco de Dados 2

Tema: Oracle DB

Conteúdos: transações, visões, visões materializadas e desempenho

Nomes dos integrantes

- Allan Bastos da Silva
  - allan.bastos@aluno.ifsp.edu.br
- Lucas dos Santos Padilha
  - lucas.padilha@aluno.ifsp.edu.br
- Mateus Carvalho Lucas
  - mateus.lucas1@aluno.ifsp.edu.br
- Victor Probio Lopes
  - victor.probio@aluno.ifsp.edu.br

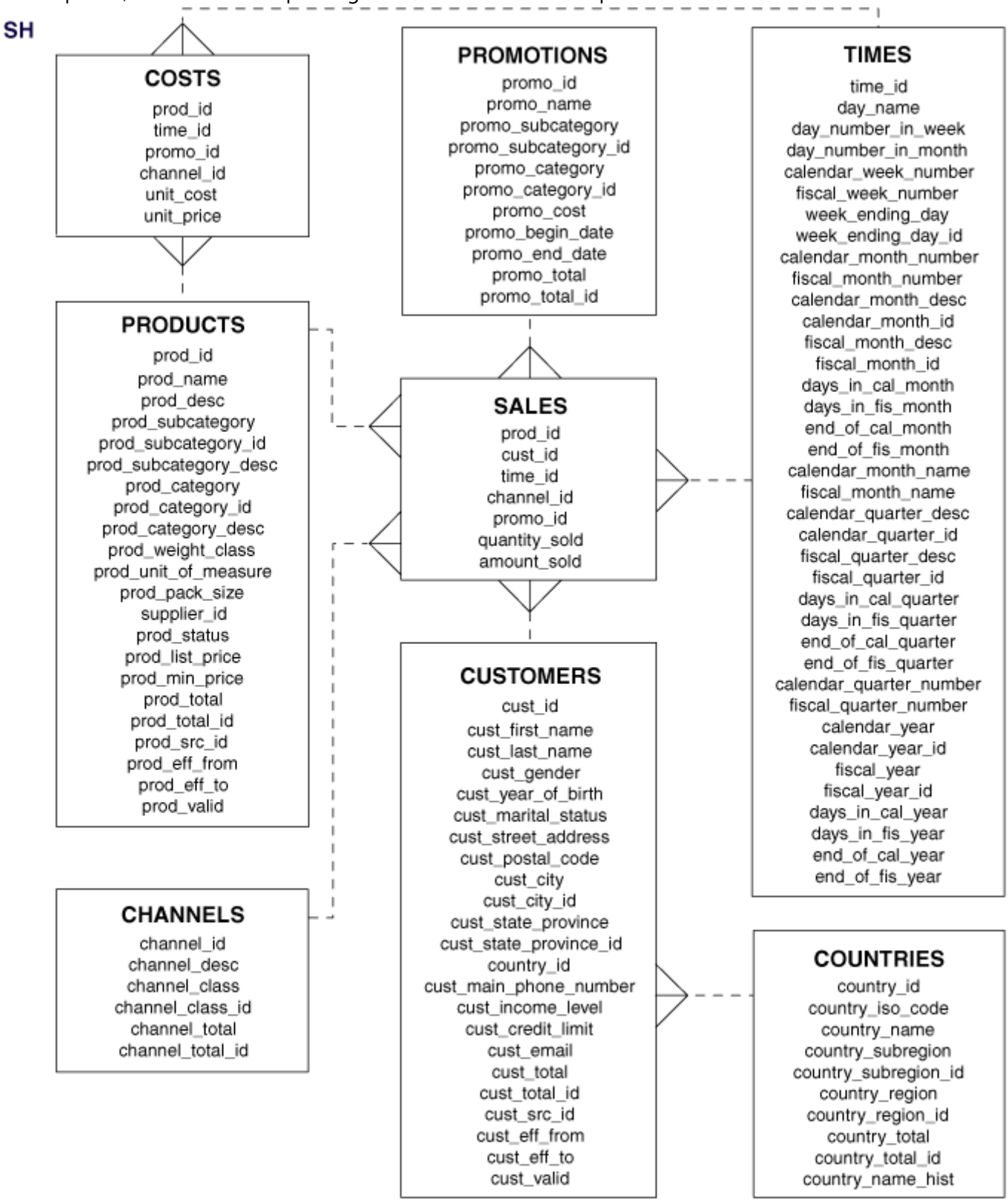
Sumário

1. Definição do Banco de Dados
2. Análise da Viabilidade
3. Definição do Esquema e Carga de Dados
4. Consultas Simples
5. Transações
6. Visões (views)
7. Visões materializadas (materialized views)
8. Desempenho
9. Conclusão
10. Referências

1. Definição do Banco de Dados

Como base de dados, o grupo optou por selecionar uma amostra fornecida pela própria Oracle para o Oracle DB, com documentação disponível no site da empresa. Várias amostras estão disponíveis, cada uma com um propósito de testes diferente, mas que podem ser utilizadas de forma independente. O banco de dados escolhido foi o esquema "Sales History", que foi projetado para demonstrações com grandes quantidades de dados. Tal escolha se baseou no conteúdo do projeto, que trata sobre visões, visões materializadas e

desempenho, tornando um esquema grande um candidato ideal para as atividades.



O repositório do banco se encontra no GitHub, e a versão utilizada foi a mais recente: 23c. Os testes iniciais com os computadores do laboratório usavam a versão 18c do Oracle DB. Embora a versão da base de dados usada não seja oficialmente compatível com a versão instalada do banco (18c), os testes realizados pelo grupo concluíram que os scripts podem ser executados sem problemas. A versão mais nova foi escolhida no lugar da versão oficialmente compatível com o Oracle DB mais antigo por duas razões: na versão mais antiga do banco, todos os esquemas de amostras precisam ser instalados juntos (já na mais nova cada um é independente), e o processo de instalação antigo é demorado e trabalhoso, precisando também de alterações manuais para cada computador onde os esquemas precisam ser instalados. No final, levar em consideração a

versão do Oracle DB compatível com os computadores do laboratório tornou-se irrelevante, algo explicado melhor mais adiante. Por ser uma amostra com uma quantidade de dados extremamente volumosa, a parte de carregamento de dados é dividida em dois formatos: uma parte é inserida no próprio arquivo SQL, e outras dependem de arquivos CSV que vêm inclusos ao realizar o download do repositório da amostra.

Resumindo

O software usado foi o Oracle DB 21c, enquanto o banco de dados escolhido foi a amostra oficial da Oracle "Sales History", em sua versão 23c.

2. Análise da Viabilidade

Quanto a análise da viabilidade, no dia 08/05/2023, foi realizado um teste de instalação do banco nos computadores do laboratório. O principal problema encontrado foi uma incompatibilidade da versão mais recente (21c) do Oracle DB com o Windows 7 instalados nas máquinas. Foi preciso recorrer a uma versão bem mais antiga do banco (18c). Apesar da demora na instalação, a princípio obteve-se êxito. Outro detalhe importante é que o Oracle DB Express (a versão gratuita utilizada pelo grupo) não acompanha um SGBD, sendo necessário baixar o Oracle SQLDeveloper a parte, ou usar uma alternativa como o DBeaver. Ainda foi preciso instalar a interface de linha de comando SQLcl, da Oracle, para instalar a amostra escolhida. O grupo estudou a possibilidade de hospedar o banco em algum sistema de nuvem, possibilitando que haja preocupação apenas com a liberação de acesso para o restante da turma no dia da apresentação, sendo necessário instalar apenas o DBeaver ou o SQL Developer. Até então, existiam duas possibilidades: usar o serviço gratuito da AWS ou usar o Oracle Cloud, que também possui serviços gratuitos. A ideia foi usar aquele que ofereça os melhores recursos, e que facilite a integração do banco. A escolha do serviço AWS foi realizada, e após dias de dificuldades com a execução do script no banco, esse foi importado com sucesso para a nuvem. Porém, o grupo descobriu da pior forma possível, na aula do dia 05/06/2023, que o serviço gratuito do AWS, não era gratuito (pelo menos não para Oracle). Assim, surgiu uma outra ideia: rodar o banco em uma máquina pessoal na rede do Instituto Federal, e usar apenas o DBeaver nas outras máquinas para se conectar ao servidor. Essa ideia foi testada com sucesso no mesmo dia, e até agora se provou como a melhor opção em todos os quesitos: é gratuita, fácil de configurar, possui controle ilimitado, não depende da internet para funcionar, é rápida de configurar em uma nova máquina, não precisa de nenhum software adicional nos outros computadores alem do DBeaver e possibilita executar a última versão do Oracle DB (21c) no servidor e a versão mais atual do script, evitando a necessidade de utilizar as versões mais antigas dos softwares por conta das limitações do laboratório.

Resumindo

Uma máquina pessoal será usada como servidor, executando a versão 21c do Oracle DB. Os clientes se conectarão usando o DBeaver.

3. Definição do Esquema e Carga de Dados

Foram anotadas as instruções executadas no computador servidor para instalar o banco e carregar nele os dados

1. **Preparação:**

- Baixar o Oracle Express Edition 21c, os esquemas de demonstração em sua versão 23c, o SQLcl 23.1, o SQLDeveloper e salvar seus arquivos em alguma pasta "segura" no computador;
- Extrair os conteúdos do arquivo zip do Oracle Express para uma pasta qualquer "OracleXE21" (ou algum outro nome);
- Abrir o arquivo zip do esquema, a pasta "db-sample-schemas-23.1" e extrair a subpasta "sales_history" para uma pasta qualquer no computador (nesse exemplo, dentro de uma pasta qualquer chamada "Banco");
- Extrair os conteúdos do arquivo zip do SQLcl para uma pasta qualquer (nesse exemplo uma pasta chamada "Programas").
- Extrair os conteúdos do arquivo zip do SQLDeveloper para uma pasta qualquer (nesse exemplo a mesma pasta "Programas").

2. **Instalação do Oracle DB 21c:**
    - Abrir a pasta onde o Oracle 21c foi extraído e executar o arquivo "setup.exe";
    - Prosseguir com a instalação normalmente, definindo uma senha quando solicitado;
    - Após a instalação, encontrar a pasta onde o Oracle foi instalado (normalmente "C:\app[USUARIO]\product\21c\homes\OraDB21Home1\network\admin");
    - Na pasta do Oracle, abrir a pasta network\admin e abrir os arquivos "listener.ora" e "tnsnames.ora" com o bloco de notas;
    - Nos arquivos, trocar os IPs por "localhost" (sem as aspas) e salvar os arquivos;
    - Reiniciar os serviços do Oracle no Windows (no menu iniciar, digitar "services.msc" e procurar por "OracleServiceXE" e "OracleXETNSListener", clicar com o botão direito e reiniciar os dois);
    - (Opcional) - Excluir a pasta de instalação do Oracle do computador, e o arquivo zip original.
3. **Programas adicionais:**
    - (Opcional/Solução de problemas) - Baixar e instalar o runtime do Java (baixar e abrir o arquivo instalador do Java e prosseguir com a instalação normalmente, adicionando o caminho do Java ao PATH do Windows quando solicitado);
    - (Opcional) - adicionar o caminho do executável do SQLcl ao PATH do Windows (no menu iniciar, digitar "variáveis de ambiente" e clicar em "Editar as variáveis de ambiente do sistema", clicar em "Variáveis de ambiente", selecionar a variável "Path" e clicar em "Editar", clicar em "Novo" e adicionar o caminho da pasta "Programas\sqlcl\bin" e clicar em "OK" em todas as janelas);
4. **Criação do banco de dados de amostra:**
    - Abrir o SQLcl ("Programas\SQLcl\bin\sql.exe") e conectar-se ao banco de dados com o usuário "SYSTEM" e a senha definida na instalação;
    - Navegar até a pasta "Banco/sales_history" (no SQLcl, digitar `cd [CAMINHO NO PC]/Banco/sales_history` e pressionar enter, sempre usando barras invertidas);
    - Rodar o seguinte comando e pressionar enter: `alter session set "_ORACLE_SCRIPT"=true;`
    - Rodar o seguinte comando e pressionar enter: `@sh_install.sql;`
    - Quando solicitado, digitar uma senha para o usuário "SH" e pressionar enter;
    - Quando solicitado o tablespace, pressionar enter para usar o tablespace padrão;
    - Aguardar a instalação do banco de dados de amostra e verificar se o banco foi instalado corretamente (a quantidade de linhas inseridas em cada tabela e a quantidade de linhas esperadas serão exibidas no final da instalação).
5. **Criação da conexão com o banco:**
    - Abrir o SQLDeveloper ("Programas\SQL_Developer\sqldeveloper.exe");
    - Clicar em "New Connection" e preencher os campos com os seguintes valores:
        - Connection Name: Sales History

        - Username: sh
        - Password: [SENHA DEFINIDA NA INSTALAÇÃO]
        - Hostname: localhost
        - Port: 1521
        - SID: xe
    - Clicar em "Test" e aguardar a mensagem de sucesso;
    - Clicar em "Save" e em "Connect";
    - (Opcional) - Executar consultas no banco de dados de amostra para verificar se a instalação foi bem sucedida;
    - (Opcional/Alternativa) - Usar o DBeaver (explicado em seguida) ao invés do SQLDeveloper.

Preparação - Instalação dos programas nas máquinas que serão clientes (explicado melhor mais adiante), opcional para o servidor

1. **Preparação:**
    - Baixar o instalador do DBeaver e salvar o arquivo em alguma pasta.
2. **Instalação do DBeaver:**
    - Abrir o arquivo de *setup* do DBeaver e prosseguir com a instalação.

Descrição do esquema do banco de dados de amostra

A amostra escolhida é composta por 8 tabelas, sendo elas: costs, products, channels, promotions, sales, costumers, times, countries. A importação do esquema é feita por um script chamado sh_install.sql, que chamará e executará os outros scripts referentes a criação da tabela e carga de dados. No projeto são os arquivos sh_create.sql, sh_populate.sql, respectivamente. O script sh_create.sql também define constraints, chaves estrangeiras, comentários e algumas visões e visões materializadas.

Alguns comandos usados pelos scripts para criar os esquemas são

1. **Tabela Countries:**
    - Todas as colunas criadas para essa tabela são determinadas com a constraint `NOT NULL`
    - `CONSTRAINT countries_pk PRIMARY KEY (country_id) //define a PK da tabela para a coluna country_id`
2. **Tabela Costumers:**
    - `CONSTRAINT customers_pk PRIMARY KEY (cust_id) //define a PK da tabela para a coluna cust_id`
    - `CONSTRAINT customers_country_fk FOREIGN KEY (country_id) REFERENCES countries (country_id) //a coluna country_id é uma chave estrangeira que referencia a coluna country_id da tabela Country`
3. **Tabela Promotions:**
    - Todas as colunas desta tabela possuem a constraint `NOT NULL` declarada.
    - `CONSTRAINT promo_pk PRIMARY KEY (promo_id) //definição da chave primária`
4. **Tabela Products:**
    - Algumas colunas desta tabela possuem a constraint `NOT NULL` declarada.
    - `CONSTRAINT products_pk PRIMARY KEY (prod_id) //definição da chave primária`
5. **Tabela Times:**
    - Todas as colunas desta tabela possuem a constraint `NOT NULL` declarada.
    - `CONSTRAINT times_pk PRIMARY KEY (time_id) //definição da chave primária`

6. **Tabela Channels:**
    - Todas as colunas desta tabela possuem a constraint `NOT NULL` declarada.
    - `CONSTRAINT channels_pk PRIMARY KEY (channel_id)//definição da chave primária`
7. **Tabela Sales:**
    - Todas as colunas desta tabela possuem a constraint `NOT NULL` declarada.
    - 6 das 7 colunas desta tabela são chaves estrangeiras que referenciam as chaves primárias das tabelas anteriores.
8. **Tabela Costs:**
    - Todas as colunas desta tabela possuem a constraint `NOT NULL` declarada.
    - Semelhante a tabela Sales, possui 4 de suas colunas referenciando chaves primárias de outras tabelas

4. Consultas Simples

Ao fim da execução do script, são exibidas quantas entradas foram inseridas, em cada uma das tabelas, comparando os valores com os números esperados

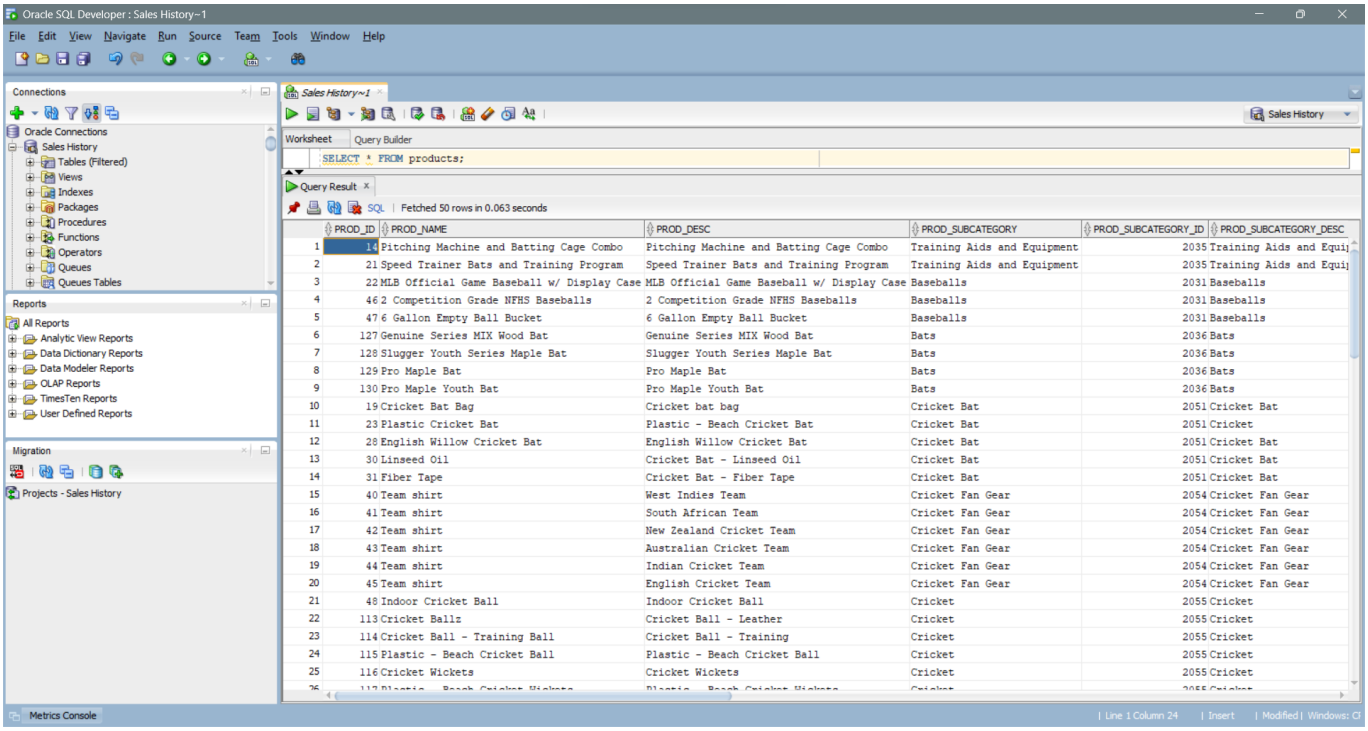
```
Installation verification
_________________________
Verification:

Table                          provided    actual
_____________________________ ___________ _________
channels                               5         5
costs                              82112     82112
countries                             35        35
customers                          55500     55500
products                              72        72
promotions                           503       503
sales                             918843    918843
times                               1826      1826
supplementary_demographics          4500      4500

Thank you!
_____________________________________________________
The installation of the sample schema is now finished.
Please check the installation verification output above.
```

O comando executado pelo script foi

```
SELECT 'Verification:' AS "Installation verification" FROM dual;

SELECT 'channels' AS "Table", 5 AS "provided", count(1) AS "actual" FROM channels
UNION ALL
SELECT 'costs' AS "Table", 82112 AS "provided", count(1) AS "actual" FROM costs
UNION ALL
```

```
SELECT 'countries' AS "Table", 35 AS "provided", count(1) AS "actual" FROM
countries
UNION ALL
SELECT 'customers' AS "Table", 55500 AS "provided", count(1) AS "actual" FROM
customers
UNION ALL
SELECT 'products' AS "Table", 72 AS "provided", count(1) AS "actual" FROM products
UNION ALL
SELECT 'promotions' AS "Table", 503 AS "provided", count(1) AS "actual" FROM
promotions
UNION ALL
SELECT 'sales' AS "Table", 918843 AS "provided", count(1) AS "actual" FROM sales
UNION ALL
SELECT 'times' AS "Table", 1826 AS "provided", count(1) AS "actual" FROM times
UNION ALL
SELECT 'supplementary_demographics' AS "Table", 4500 AS "provided", count(1) AS
"actual" FROM supplementary_demographics;
```

Para uma verificação manual mais detalhada, o comando `SELECT * FROM products;` foi executado, retornando as informações de 72 produtos inseridos no banco

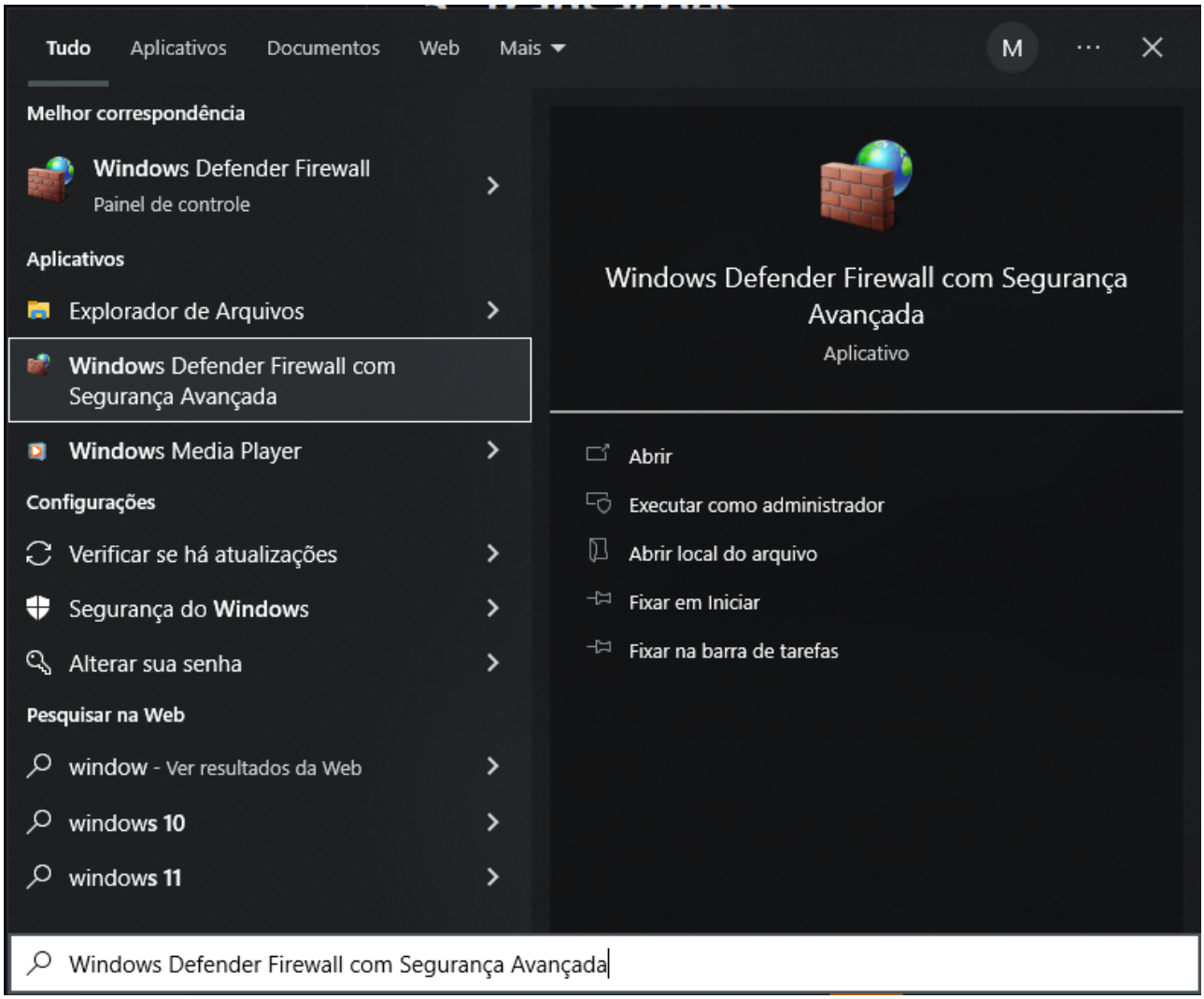


A tabela products foi escolhida devido a sua quantidade menor de entradas, mas o mesmo teste foi realizado com a tabela promotions, de 503 entradas (`SELECT * FROM promotions;`).

5. Transações

Como já citado, o grupo realizou um teste utilizando uma máquina pessoal como servidor. Para isso, as seguintes instruções foram seguidas:

1. Abrir a porta 1521 na máquina usada como servidor. Será preciso liberar no "Windows Defender Firewall com Segurança Avançada"



2. Vamos criar uma regra de entrada e saída para a porta 1521. Para isso, é só selecionar "Regra de Entrada" ou "Regra de saída" com o botão direito e clicar em "Nova regra" e seguir os passos conforme as imagens abaixo.

Windows Defender Firewall com Segurança Avançada

Arquivo Ação Exibir Ajuda

Windows Defender Firewall com Segurança Avançada em Computador Local

- Regras de Entrada
- Regras de Saída
- Regras de Segurança de Co
- Monitoramento

O Windows Defender Firewall com Segurança Avançada oferece segurança de rede a c

Visão geral

Perfil do Domínio

- O Windows Defender Firewall está ativado.
- Conexões de entrada que não correspondem a uma regra são bloqueadas.
- Conexões de saída que não correspondem a uma regra são permitidas.

Perfil Particular

- O Windows Defender Firewall está ativado.
- Conexões de entrada que não correspondem a uma regra são bloqueadas.
- Conexões de saída que não correspondem a uma regra são permitidas.

Perfil Público Ativo

- O Windows Defender Firewall está ativado.
- Conexões de entrada que não correspondem a uma regra são bloqueadas.
- Conexões de saída que não correspondem a uma regra são permitidas.

Propriedades do Windows Defender Firewall

Introdução

Autenticar comunicações entre computadores

Crie regras de segurança de conexão para especificar como e quando as conexões entre os c autenticadas e protegidas usando a segurança IPsec.

Regras de Segurança de Conexão

Exibir e criar regras de firewall

Crie regras de firewall para permitir ou bloquear conexões a portas ou programas especificados. permitir uma conexão somente se estiver autenticada ou se vier de um usuário, grupo ou compu padrão, as conexões de entrada são bloqueadas, a menos que coincidam com uma regra que conexões de saída são permitidas, a menos que correspondam a uma regra que as bloqueie.

Ações

Windows Defender Firewall com Segurança Avançada em ...

- Importar Política...
- Exportar Política...
- Restaurar Política Padrão
- Diagnosticar/Reparar
- Exibir
- Atualizar
- Propriedades
- Ajuda

Assistente para Nova Regra de Entrada

Tipo de regra

Selecionar o tipo de regra de firewall a ser criada.

Etapas:

- Tipo de regra
- Protocolo e Portas
- Ação
- Perfil
- Nome

Que tipo de regra você deseja criar?

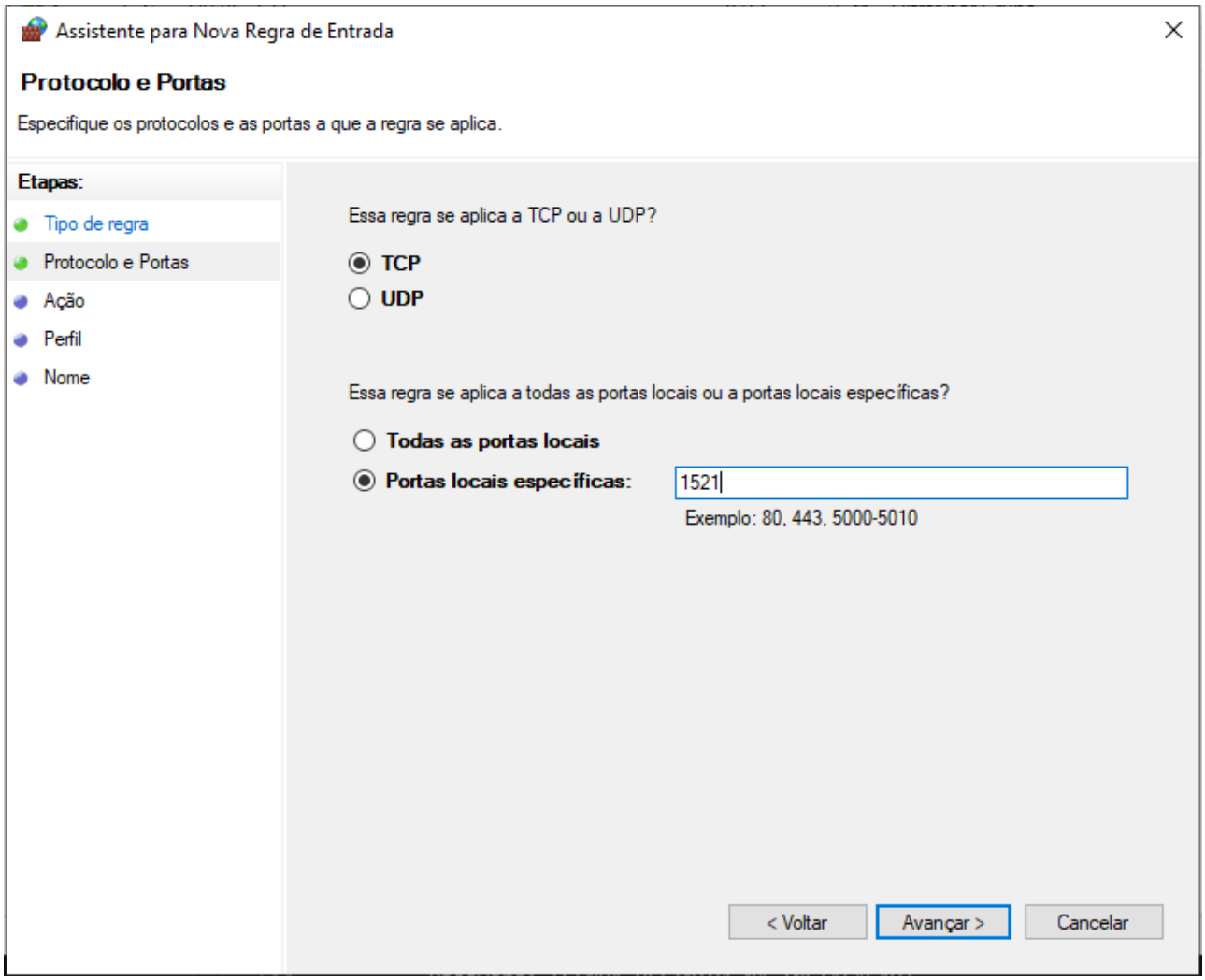
( ) **Programa**
Regra que controla conexões para um programa.

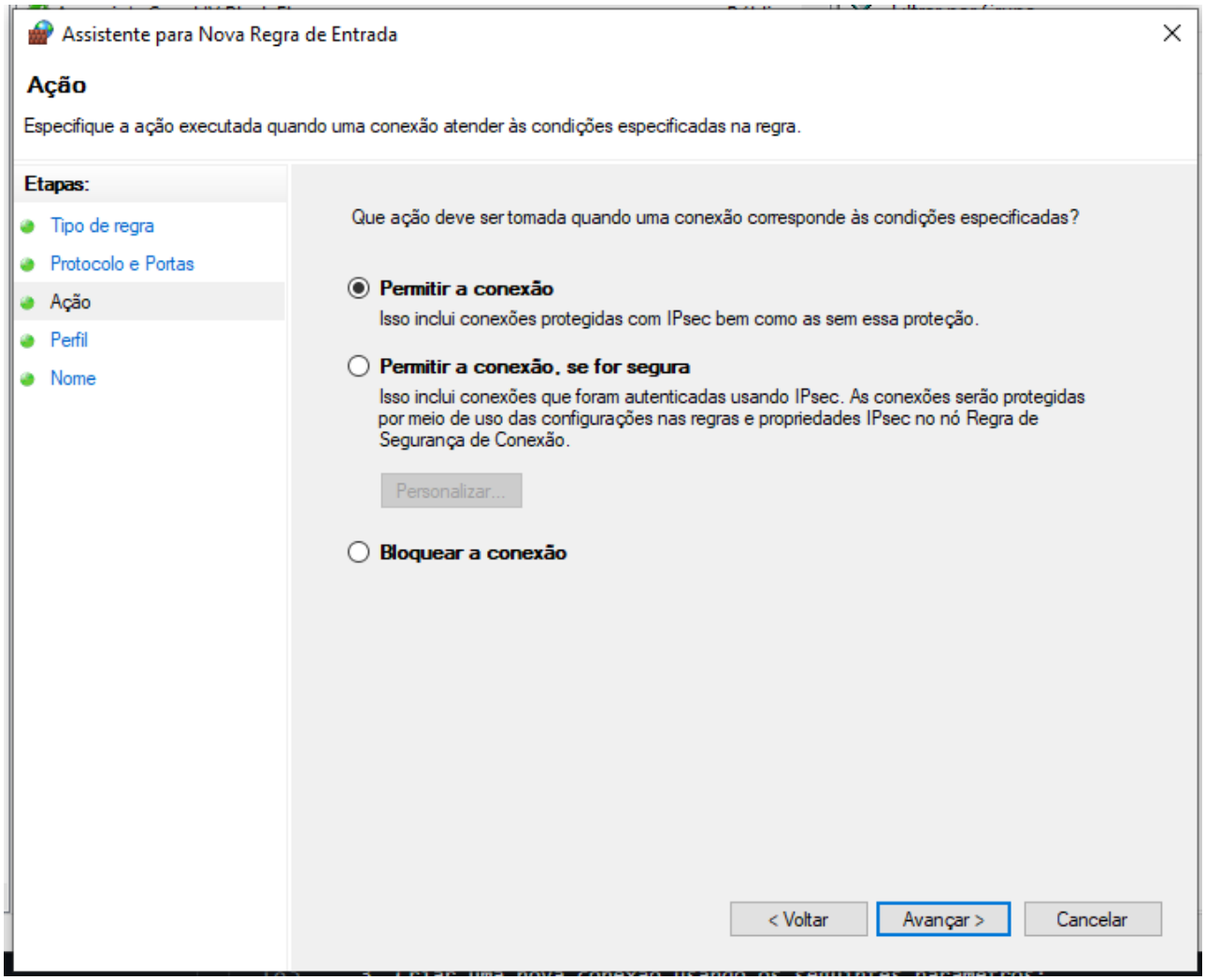
(•) **Porta**
Regra que controla conexões para uma porta TCP ou UDP.

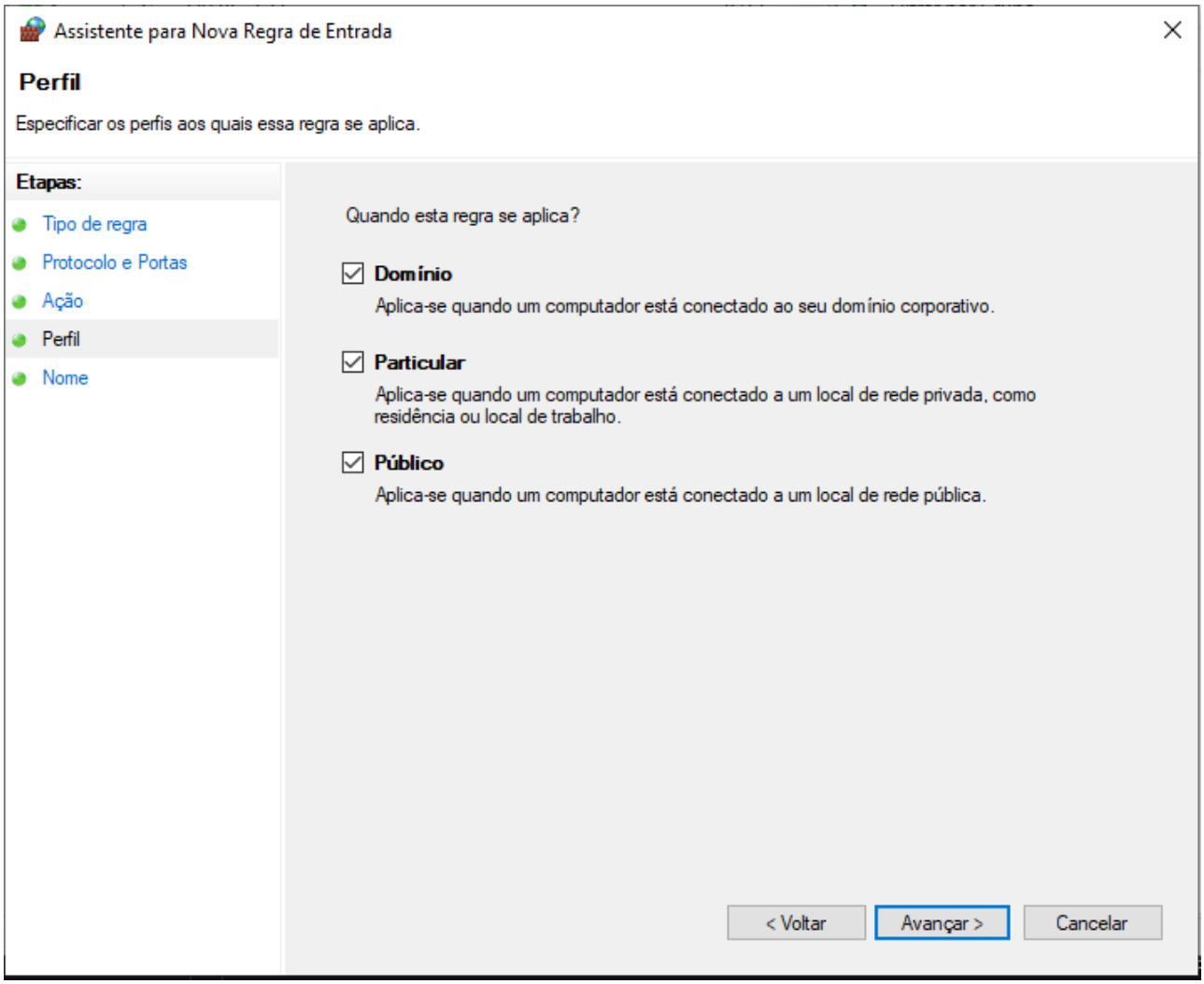
( ) **Predefinida:**
@FirewallAPI.dll,-80200
Regra que controla conexões para uma experiência do Windows.

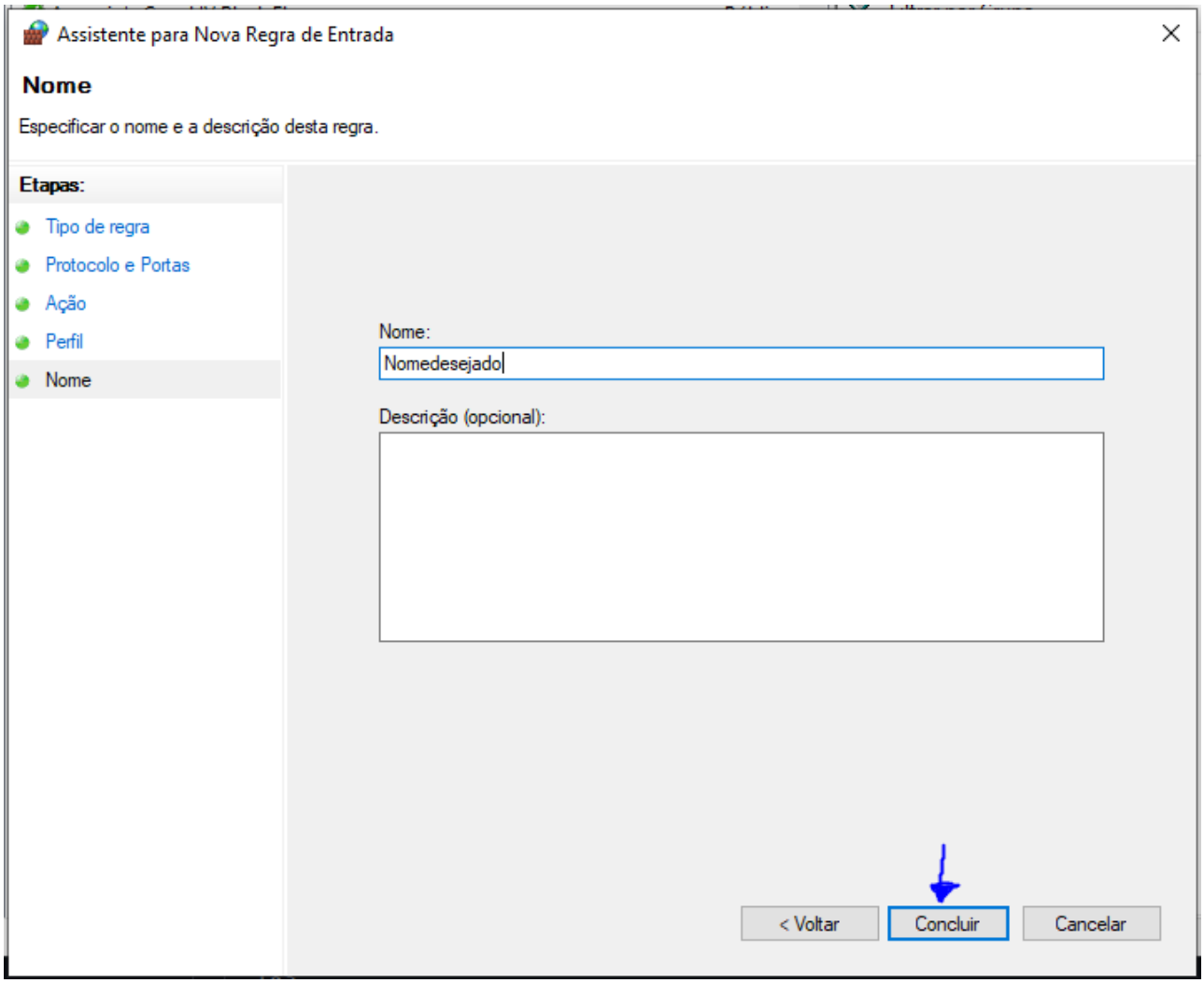
( ) **Personalizado**
Regra personalizada.

< Voltar | Avançar > | Cancelar


Assistente para Nova Regra de Entrada
Protocolo e Portas
Especifique os protocolos e as portas a que a regra se aplica.
Etapas:
Tipo de regra
Protocolo e Portas
Ação
Perfil
Nome
Essa regra se aplica a TCP ou a UDP?
TCP
UDP
Essa regra se aplica a todas as portas locais ou a portas locais específicas?
Todas as portas locais
Portas locais específicas:
1521
Exemplo: 80, 443, 5000-5010
< Voltar
Avançar >
Cancelar


Assistente para Nova Regra de Entrada
Ação
Especifique a ação executada quando uma conexão atender às condições especificadas na regra.
Etapas:
Tipo de regra
Protocolo e Portas
Ação
Perfil
Nome
Que ação deve ser tomada quando uma conexão corresponde às condições especificadas?
Permitir a conexão
Isso inclui conexões protegidas com IPsec bem como as sem essa proteção.
Permitir a conexão, se for segura
Isso inclui conexões que foram autenticadas usando IPsec. As conexões serão protegidas por meio de uso das configurações nas regras e propriedades IPsec no nó Regra de Segurança de Conexão.
Personalizar...
Bloquear a conexão
< Voltar
Avançar >
Cancelar


Assistente para Nova Regra de Entrada
Perfil
Especificar os perfis aos quais essa regra se aplica.
Etapas:
Tipo de regra
Protocolo e Portas
Ação
Perfil
Nome
Quando esta regra se aplica?
Domínio
Aplica-se quando um computador está conectado ao seu domínio corporativo.
Particular
Aplica-se quando um computador está conectado a um local de rede privada, como residência ou local de trabalho.
Público
Aplica-se quando um computador está conectado a um local de rede pública.
< Voltar
Avançar >
Cancelar



3. Criar uma nova conexão usando os seguintes parâmetros:
    - Username: sh
    - Password: [SENHA DEFINIDA NA INSTALAÇÃO]
    - Hostname: [IP DO SERVIDOR]
    - Port: 1521
    - SID: xe
4. Realizar consultas simples para testar a conexão.

Após o fim do teste, o grupo iniciou a pesquisa sobre transações no Oracle. No Oracle, transações são unidades lógicas atômicas de trabalho que contém um ou mais comandos SQL. Essas transações são então grupos de comandos que podem ser aplicados ao banco de dados em sua totalidade (*commit*) ou desfeitos em sua totalidade (*rollback*). O Oracle atribui um ID único para cada transação. As transações do Oracle obedecem às propriedades ACID, acrônimo que significa:

- **Atomicidade**
    - Ou todas as tarefas são executadas, ou nenhuma é. Transações não são parciais, e caso haja uma falha no meio de sua execução, a transação é revertida em sua totalidade.
- **Consistência**
    - A transação deve tirar o banco de dados de um estado consistente e o levar para outro estado consistente. Falhas não podem resultar em inconsistência de dados.
- **Isolação**
    - As transações são isoladas entre si, ou seja, os efeitos de uma transação não são visíveis para outras até que essa receba um *commit*.

- **Durabilidade**
  - Mudanças feitas por transações que receberam um *commit* são permanentes, e não são perdidas no banco.

Transações em um banco de dados possuem um ou mais comandos, e um começo e fim. No Oracle, o começo de uma transação é demarcado pelo primeiro comando SQL executável encontrado. Quando uma nova transação começa, o Oracle DB a associa a um conjunto de dados para serem "desfeitos", e depois atribui à transação um ID. Já o fim de uma transação pode ser uma das seguintes situações:

- O usuário executa um comando `COMMIT` ou `ROLLBACK` sem haver um *savepoint* prévio.
- O usuário executa um comando de definição de dados, como `CREATE`, `DROP`, `RENAME` ou `ALTER`. Nesse caso, o Oracle implicitamente executa um comando `COMMIT` na transação.
- O usuário sai da aplicação. Nesse caso, o Oracle implicitamente executa um comando `COMMIT` na transação.
- Um processo termina abruptamente. Nesse caso, a transação é implicitamente desfeita pelo Oracle.

Uma transação pode começar implicitamente ao ser nomeada, usando o comando SQL `SET TRANSACTION NAME 'nome';` (onde "nome" é um nome qualquer). O controle de transações no Oracle funciona da seguinte forma:

- Um comando `COMMIT` encerra a transação atual, apaga todos os *savepoints* e torna as mudanças efetuadas permanentes.
- Um comando `ROLLBACK` desfaz todas as mudanças desde o início da transação. Um comando `ROLLBACK TO SAVEPOINT nome` desfaz as mudanças efetuadas na transação desde o *savepoint* chamado "nome", mas não a encerra.
- Um comando `SAVEPOINT nome` (onde "nome" é um nome qualquer) cria um savepoint, para o qual é possível retornar depois.

Exemplo de transações

Para criar um novo usuário, que fará as operações simples ao banco, é necessário

1. Logar como usuário SYSTEM no Oracle

2. Garantir ao usuário sh permissão para criar usuários e manipular o banco com o comando `GRANT ALL PRIVILEGES to sh;`, após rodar `alter session set "_ORACLE_SCRIPT"=true;`

3. Voltar na conexão do usuário sh e criar o novo usuário (com a senha "usuario") usando o comando `CREATE USER sh_usuario IDENTIFIED BY "usuario";`

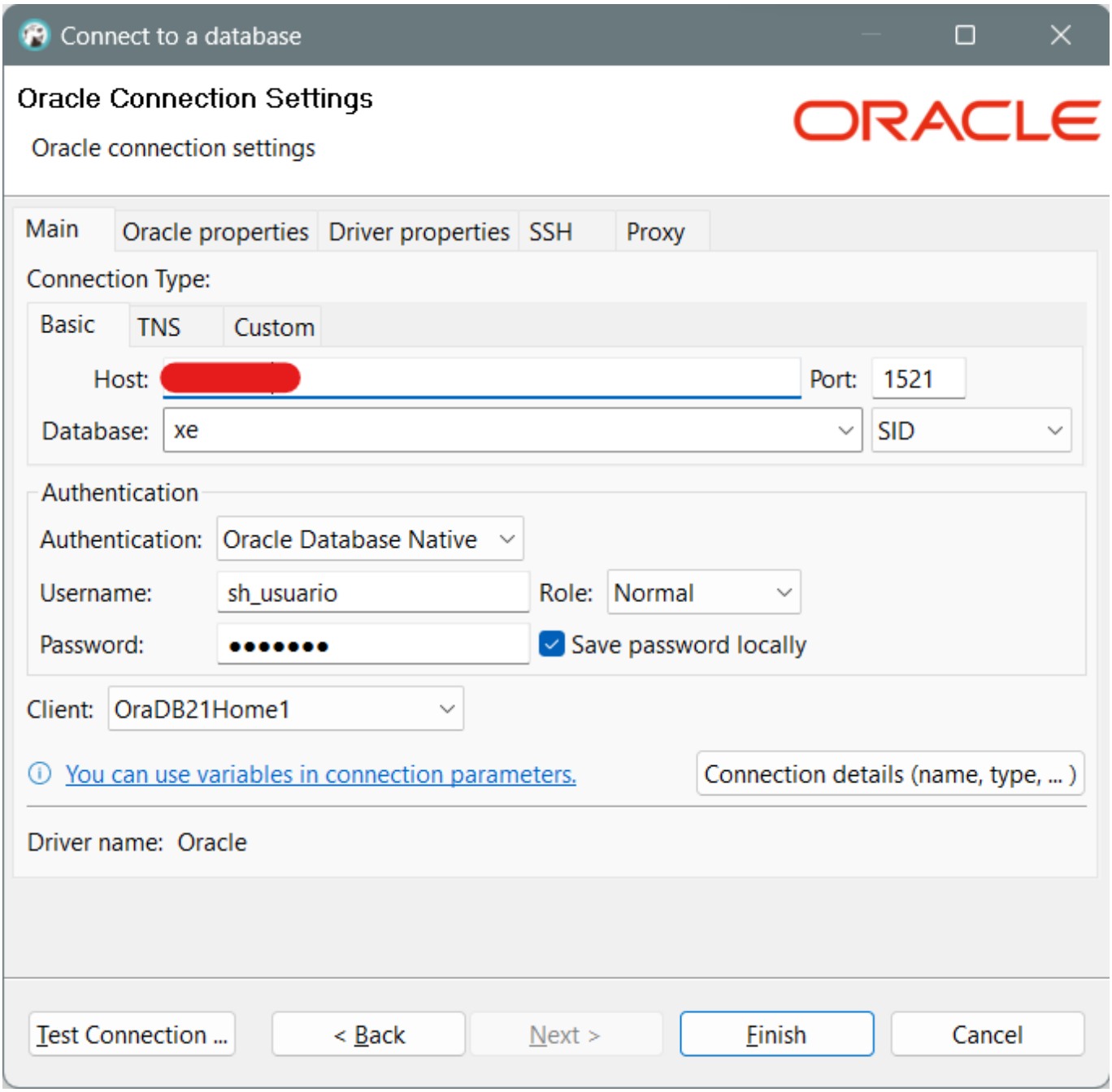
4. Garantir ao novo usuário as permissões de conexão ao banco de dados, rodando o comando `GRANT CREATE SESSION to sh_usuario;`

5. Garantir ao novo usuário permissão para selecionar dados das tabelas e inserir novas vendas e canais, rodando os comandos:

```
GRANT SELECT ON sh.CHANNELS TO sh_usuario;
GRANT SELECT ON sh.COSTS TO sh_usuario;
GRANT SELECT ON sh.COUNTRIES TO sh_usuario;
GRANT SELECT ON sh.CUSTOMERS TO sh_usuario;
```

```sql
GRANT SELECT ON sh.PRODUCTS TO sh_usuario;
GRANT SELECT ON sh.PROMOTIONS TO sh_usuario;
GRANT SELECT ON sh.SALES TO sh_usuario;
GRANT SELECT ON sh.SUPPLEMENTARY_DEMOGRAPHICS TO sh_usuario;
GRANT SELECT ON sh.TIMES TO sh_usuario;
GRANT SELECT ON sh.PRODUCTS_SALES TO sh_usuario;
GRANT SELECT ON sh.PROFITS TO sh_usuario;
GRANT SELECT ON sh.PRODUCTS_SALES_MV TO sh_usuario;

GRANT INSERT ON sh.sales to sh_usuario;
GRANT INSERT ON sh.channels to sh_usuario;
```

6. No cliente, abrir o DBeaver e criar uma nova conexão Oracle usando os seguintes parâmetros (instalando o *driver*) quando solicitado:
    - Username: sh_usuario
    - Password: usuario
    - Hostname: [IP DO SERVIDOR]
    - Port: 1521
    - SID: xe

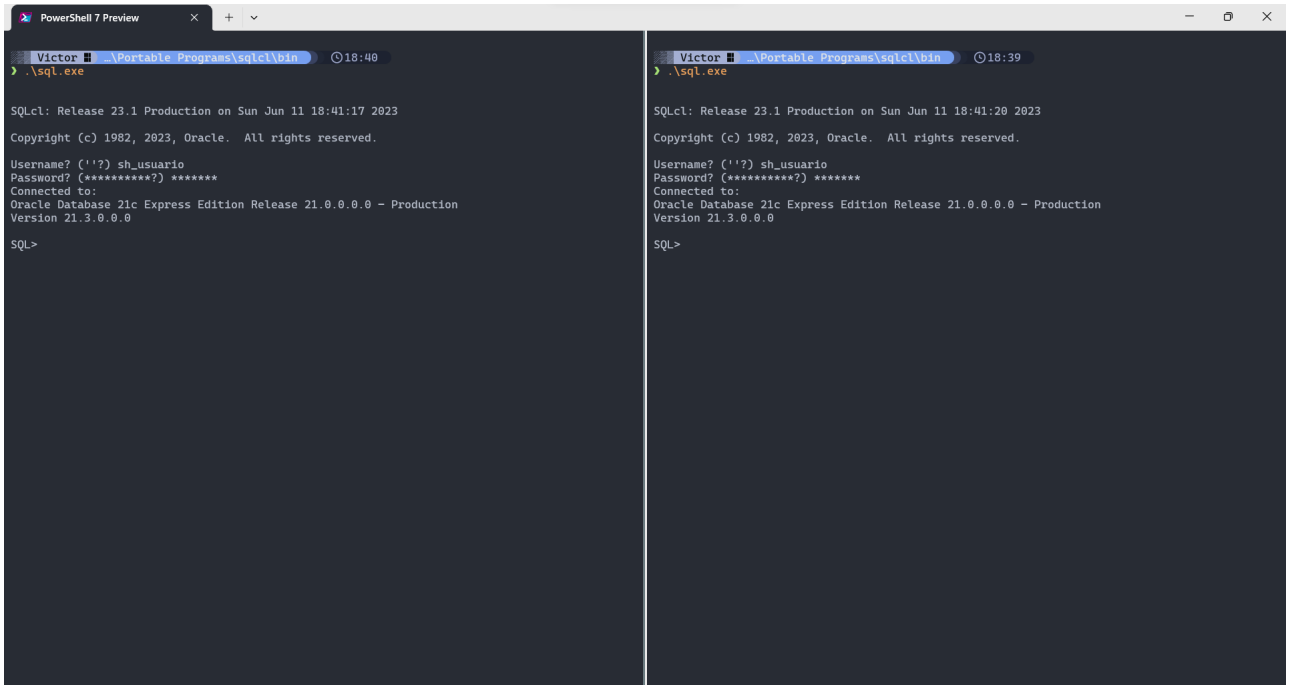


Feito isso, é possível se conectar ao banco com o usuário sh_usuario e selecionar dados das tabelas de sh, bem como inserir novos dados em sh.sales e sh.channels.

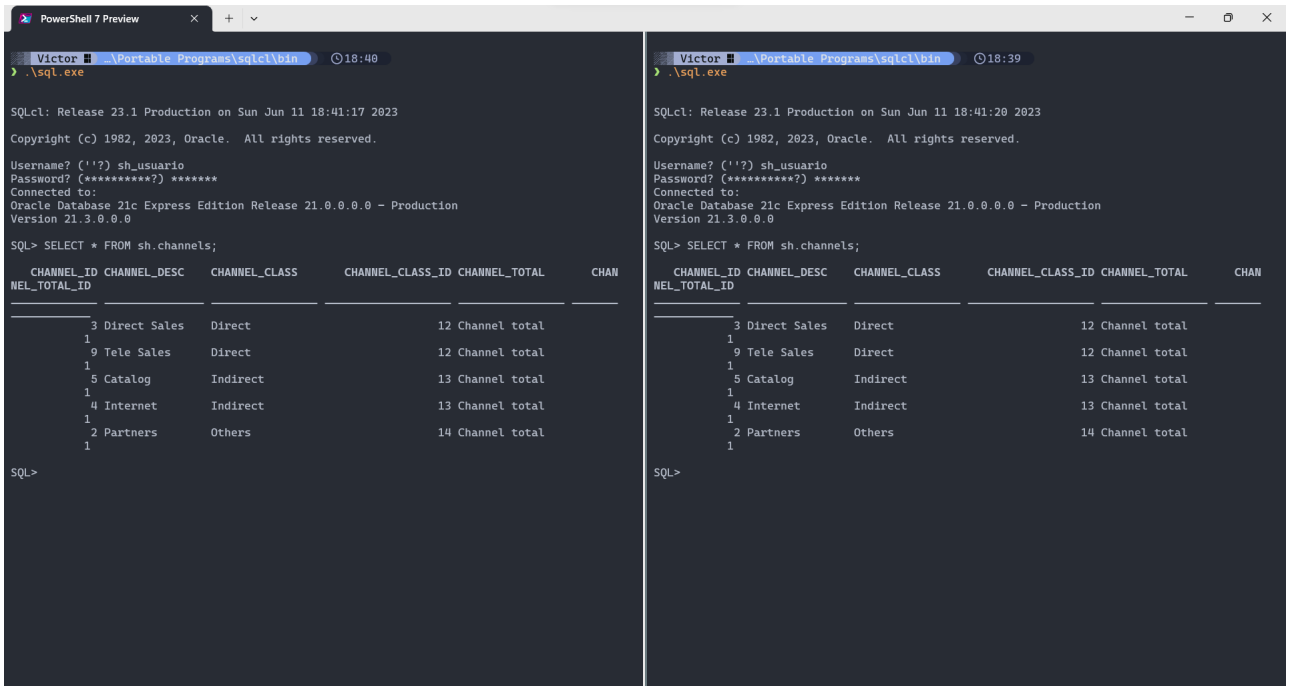
Testando as transações com múltiplos usuários

1. Se conectar ao banco com o usuário sh_usuário em duas instâncias diferentes (na mesma máquina em processos diferentes ou em máquinas separadas) usando o SQL Developer, DBeaver, SQL Plus, SQLcl

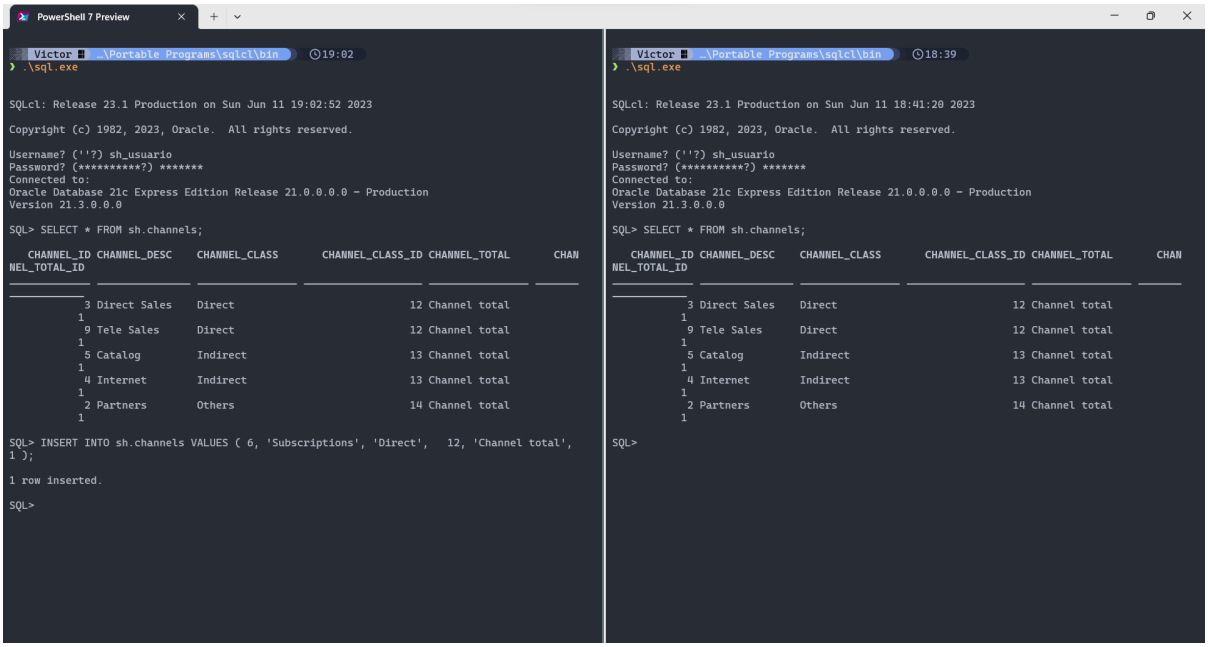
etc.



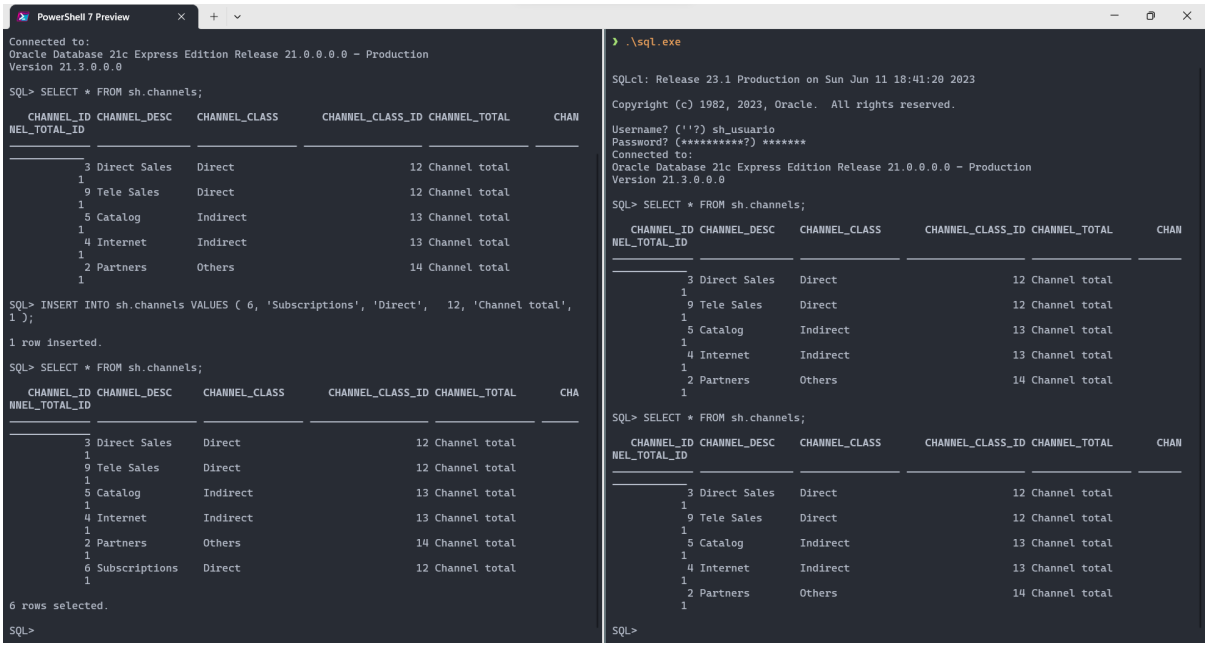
2. Para fins de comparação, executar (em ambas instâncias) o comando `SELECT * FROM sh.channels;`



3. Após evidenciadas as mesmas tuplas, iniciar uma nova transação, inserindo um novo canal em uma das instâncias:
    - Rodar o comando SQL `INSERT INTO sh.channels VALUES ( 6, 'Subscriptions', 'Direct', 12, 'Channel total', 1 );`

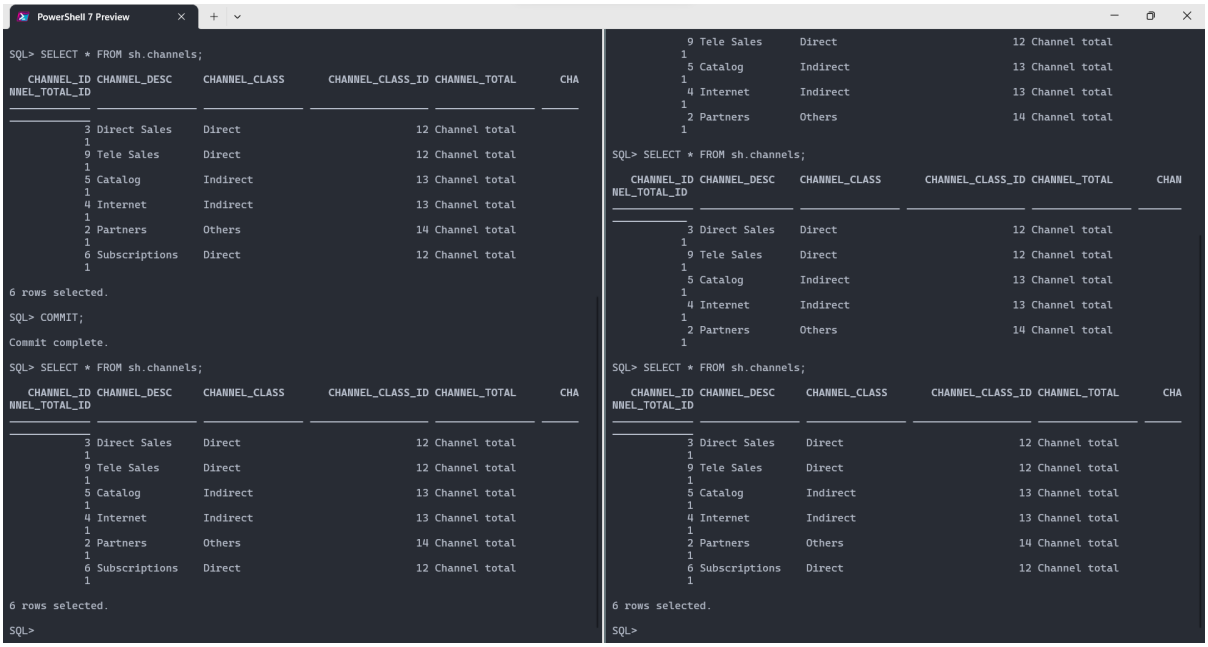
- 

4. Mais uma vez, rodar o comando `SELECT` do passo 2 em ambas instâncias

- 

  - Apenas o usuário que iniciou a transação vê os resultados do `INSERT`, evidenciando a propriedade de isolação das transações.

5. Tornar permantentes as alterações no banco, com o comando `COMMIT;` (no usuário que iniciou a transação de inserção) e executar novamente os comandos `SELECT` nas duas instâncias



- Após a execução do `COMMIT`, os dados são mantidos permanentes (propriedade de durabilidade) e podem ser vistos nas duas instâncias.

6. Visões (*views*)

Visões no Oracle DB funcionam de forma semelhante ao Postgres e desempenham basicamente o mesmo papel: são representações lógicas, de uma ou mais tabelas, ou seja, basicamente uma consulta armazenada. A visão obtém os dados a partir das tabelas bases, que podem ser tabelas ou até mesmos outras visões. De modo geral, as visões possibilitam "moldar" a apresentação de determinado dado de diferentes maneiras, para diferentes usuários. Além disso, garantem um nível a mais de segurança, pois restringem o acesso direto a determinadas tuplas de uma tabela, e também contam com a vantagem de poder esconder a complexidade de uma consulta. No Oracle, uma visão pode ser criada com o comando `CREATE VIEW nome AS` (onde "nome" é um nome qualquer), seguido de uma consulta (`SELECT`). Assim, toda vez que a visão é buscada, o resultado de seu `SELECT` é retornado.

Exemplo de *view*

Para o esquema sh já discutido, a seguinte visão retorna uma lista de todas as vendas cadastradas, bem como informações do custo de cada produto (e qual foi o custo total de cada venda), ou seja, auxilia na visualização dos lucros obtidos com cada venda:

```
CREATE OR REPLACE VIEW profits
 AS SELECT
  s.channel_id,
  s.cust_id,
  s.prod_id,
  s.promo_id,
  s.time_id,
  c.unit_cost,
  c.unit_price,
  s.amount_sold,
  s.quantity_sold,
```

```
  c.unit_cost * s.quantity_sold TOTAL_COST
 FROM
  costs c, sales s
 WHERE c.prod_id = s.prod_id
   AND c.time_id = s.time_id
   AND c.channel_id = s.channel_id
   AND c.promo_id = s.promo_id;
```

Assim, essas informações de lucros podem ser retornadas com o comando `SELECT * FROM profits;`

Worksheet | Query Builder

```
SELECT * FROM profits;
```

Query Result

SQL | Fetched 50 rows in 0.206 seconds

	CHANNEL_ID	CUST_ID	PROD_ID	PROMO_ID	TIME_ID	UNIT_COST	UNIT_PRICE	AMOUNT_SOLD	QUANTITY_SOLD	TOTAL_COST
1	3	13869	131	999	04-JAN-19	16.52	26.96	26.96	1	16.52
2	3	14124	131	999	04-JAN-19	16.52	26.96	26.96	1	16.52
3	3	26630	131	999	04-JAN-19	16.52	26.96	26.96	1	16.52
4	3	357	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
5	3	477	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
6	3	720	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
7	3	868	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
8	3	1842	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
9	3	1945	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
10	3	2146	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
11	3	2272	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
12	3	3547	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
13	3	3705	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
14	3	3796	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
15	3	3819	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
16	3	5184	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
17	3	5298	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
18	3	5310	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
19	3	5437	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
20	3	5588	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
21	3	5730	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
22	3	7178	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
23	3	7691	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
24	3	7849	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
25	3	8351	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52
26	3	8488	131	999	15-JAN-19	16.52	26.59	26.59	1	16.52

Outro exemplo de *view*

Para o esquema sh já discutido, a seguinte visão retorna uma lista de todos os produtos (seu nome e id), e quantas unidades do mesmo foram vendidas no total:

```
CREATE OR REPLACE VIEW products_sales
    AS
    SELECT     s.prod_id,
               p.prod_name,
               SUM(s.quantity_sold) AS units_sold
    FROM       sales s INNER JOIN products p
        ON s.prod_id = p.prod_id
```

```
GROUP BY p.prod_name, s.prod_id
ORDER BY s.prod_id;
```

Assim, essas informações de vendas podem ser retornadas com o comando `SELECT * FROM products_sales;`

Worksheet | Query Builder

```
SELECT * FROM products_sales
```

Query Result

SQL | Fetched 50 rows in 0.251 seconds

	PROD_ID	PROD_NAME	UNITS_SOLD
1	13	Soccer Goal - Portable	6002
2	14	Pitching Machine and Batting Cage Combo	6010
3	15	Right-Handed Graphite Shaft Iron Set	5766
4	16	Sonic Core Graphite Racquet	6929
5	17	Soccer Goal - Official	6160
6	18	Lithium Electric Golf Caddy	9591
7	19	Cricket Bat Bag	10430
8	20	Match Used Autograph Racquet	10903
9	21	Speed Trainer Bats and Training Program	5205
10	22	MLB Official Game Baseball w/ Display Case	3441
11	23	Plastic Cricket Bat	19642
12	24	Endurance Coolcore 1/2 Zip Pullover	20948
13	25	5 Point Batting Tee	19557
14	26	Pro Style Batting Tee	15950
15	27	Bucket of 24 Synthetic Baseballs	12163
16	28	English Willow Cricket Bat	16796
17	29	Limited Edition Racquet	7197
18	30	Linseed Oil	29282
19	31	Fiber Tape	23108
20	32	Bucket of 24 Leather Baseballs	11253
21	33	Baseball Is Life Cap	22768
22	34	11.5" Youth Triple Stripe Series Glove	13043
23	35	Catchers Mitt	16494
24	36	12" Premium Series Glove	13008
25	37	12.75" Premium Series Glove	17430
26	38	11" Youth Field Master Glove	9523

Acesso de dados em visões no Oracle DB

Quando uma visão é referenciada em uma declaração SQL, o Oracle executa os seguintes passos:

1. Une a query (quando possível) com a visão (que na prática é outra query).
2. Analisa a instrução unida acima, em uma área SQL compartilhada.
3. Por fim, executa a declaração SQL.

Exemplo dos passos (retirado da documentação da Oracle)

The following example illustrates data access when a view is queried. Assume that you create `employees_view` based on the `employees` and `departments` tables:

```
CREATE VIEW employees_view AS
  SELECT employee_id, last_name, salary, location_id
  FROM   employees JOIN departments USING (department_id)
  WHERE  department_id = 10;
```

A user executes the following query of `employees_view`:

```
SELECT last_name
FROM   employees_view
WHERE  employee_id = 200;
```

Oracle Database merges the view and the user query to construct the following query, which it then executes to retrieve the data:

```
SELECT last_name
FROM   employees, departments
WHERE  employees.department_id = departments.department_id
AND    departments.department_id = 10
AND    employees.employee_id = 200;
```

7. Visões materializadas (*materialized views*)

Uma visão materializada é o resultado de uma query que foi armazenado ou "materializado" como um objeto de um schema. Algumas características:

- Elas contêm os dados armazenados e consomem espaço de armazenamento real.
- Precisam ser atualizadas ou "*refreshed*" quando algo na tabela principal mudar.
- Melhoram a performance de uma execução SQL quando usadas para operações de reescrita (*query rewrite*). No Oracle, as visões materializadas são criadas de forma semelhante às visões, usando o comando `CREATE MATERIALIZED VIEW nome AS` (onde "nome" é um nome qualquer), seguido de

uma consulta (`SELECT`). Assim, o resultado desse `SELECT` é armazenado em disco, em outra tabela, e quando é efetuada uma busca na visão materializada, o Oracle retorna as tuplas dessa tabela.

Sintaxe de uma visão materializada

```
CREATE MATERIALIZED VIEW view-name
BUILD [IMMEDIATE | DEFERRED]
REFRESH [FAST | COMPLETE | FORCE ]
ON [COMMIT | DEMAND ]
[[ENABLE | DISABLE] QUERY REWRITE]
AS
SELECT ...;
```

Exemplo de *materialized view*

Para o esquema sh já discutido, a seguinte visão materializada retorna uma lista de todos os produtos (seu nome e id), e quantas unidades do mesmo foram vendidas no total. A consulta é exatamente igual à visão `products_sales`, porém é uma visão materializada, que armazena esses resultados em uma tabela:

```
CREATE MATERIALIZED VIEW products_sales_mv
    AS
    SELECT    s.prod_id,
              p.prod_name,
              SUM(s.quantity_sold) AS units_sold
    FROM      sales s INNER JOIN products p
        ON s.prod_id = p.prod_id
    GROUP BY p.prod_name, s.prod_id
    ORDER BY s.prod_id;
```

Assim, essas informações de vendas podem ser retornadas com o comando `SELECT * FROM products_sales_mv;`

Worksheet | Query Builder

```
SELECT * FROM products_sales_mv;
```

Query Result

SQL | Fetched 50 rows in 0.018 seconds

	PROD_ID	PROD_NAME	UNITS_SOLD
1	13	Soccer Goal - Portable	6002
2	14	Pitching Machine and Batting Cage Combo	6010
3	15	Right-Handed Graphite Shaft Iron Set	5766
4	16	Sonic Core Graphite Racquet	6929
5	17	Soccer Goal - Official	6160
6	18	Lithium Electric Golf Caddy	9591
7	19	Cricket Bat Bag	10430
8	20	Match Used Autograph Racquet	10903
9	21	Speed Trainer Bats and Training Program	5205
10	22	MLB Official Game Baseball w/ Display Case	3441
11	23	Plastic Cricket Bat	19642
12	24	Endurance Coolcore 1/2 Zip Pullover	20948
13	25	5 Point Batting Tee	19557
14	26	Pro Style Batting Tee	15950
15	27	Bucket of 24 Synthetic Baseballs	12163
16	28	English Willow Cricket Bat	16796
17	29	Limited Edition Racquet	7197
18	30	Linseed Oil	29282
19	31	Fiber Tape	23108
20	32	Bucket of 24 Leather Baseballs	11253
21	33	Baseball Is Life Cap	22768
22	34	11.5" Youth Triple Stripe Series Glove	13043
23	35	Catchers Mitt	16494
24	36	12" Premium Series Glove	13008
25	37	12.75" Premium Series Glove	17430
26	38	11" Youth Field Master Glove	9523

Métodos de *Refresh* para Visões Materializadas

Visões materializadas precisam ser "recarregadas" de tempos em tempos para manter seus dados consistentes no banco. Para isso, no Oracle, existem dois métodos principais:

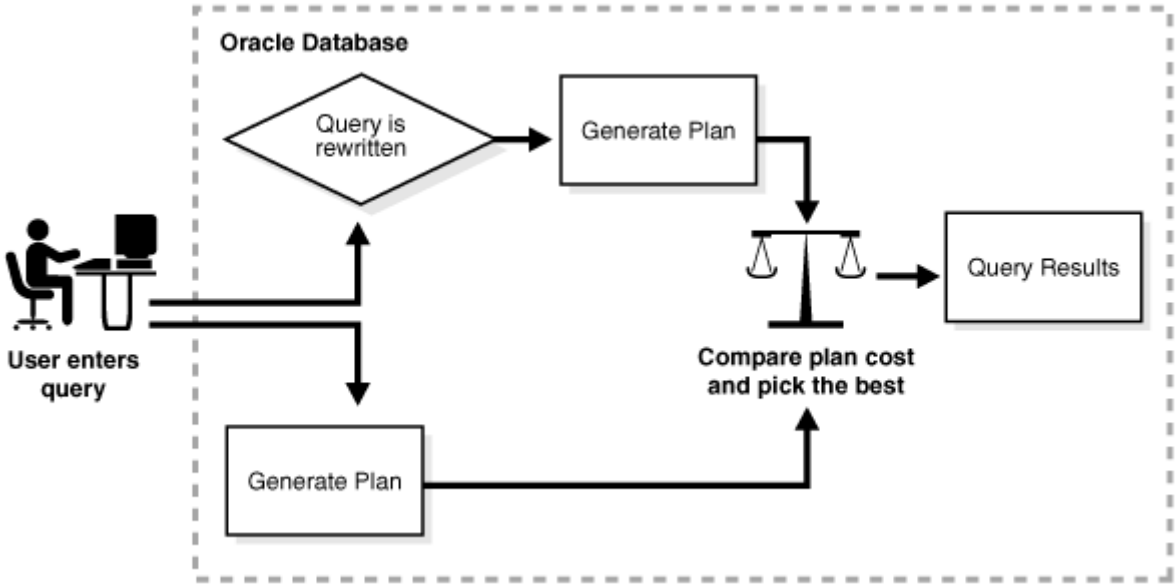
- ***Complete Refresh:*** Executada ao criar uma visão materializada. Reconstrói a visão materializada por completo. Esse método pode ser lento, especialmente caso o banco de dados precise ler e processar

imensas quantidades de dados. Pode ser executado sob demanda com o comando `BEGIN DBMS_SNAPSHOT.REFRESH( '"SCHEMA"."MATERIALIZED_VIEW"','C'); end;` (onde "SCHEMA" e "MATERIALIZED_VIEW" são, respectivamente, os nomes do esquema e da visão materializada).

- ***Incremental Refresh / Fast Refresh:*** Processa apenas as mudanças para dados existentes. Ele elimina a necessidade de "reconstruir" a visão materializada do começo. Pode ser executado sob demanda com o comando `BEGIN DBMS_SNAPSHOT.REFRESH( '"SCHEMA"."MATERIALIZED_VIEW"','F'); end;` (onde "SCHEMA" e "MATERIALIZED_VIEW" são, respectivamente, os nomes do esquema e da visão materializada). É possível atualizar visões materializadas por demanda ou em intervalos regulares de tempo. Além disso, é possível criar uma configuração de modo que a cada *commit* de transação de suas tabelas base, a visão seja atualizada.

Query Rewrite

Essa opção, habilitada com o comando `ENABLE QUERY REWRITE` ao criar uma visão materializada, transforma uma *user request* escrita em termos da tabela principal, localizada no banco, em uma semanticamente equivalente que inclui visões materializadas. Ou seja, é o mecanismo que permite ao Oracle automaticamente reescrever as consultas em termos de visões materializadas já existentes. É sabido que quando o banco possui quantidades gigantescas de dados, realizar uma operação de junção, ou agregação é extremamente custoso em questão de tempo e processamento. Como visões materializadas possuem essas operações pré-computadas, uma query rewrite pode rapidamente atender demandas de outras queries usando visões materializadas. A figura abaixo (retirada da documentação do Oracle) mostra o Oracle DB gerando um plano de execução para ambas as *queries*, reescrita e de usuário, e no fim comparando qual tem menor custo para obter os resultados desejados:



8. Desempenho

A análise de desempenho no Oracle é de extrema importância, pois o desempenho eficiente de um banco de dados Oracle é essencial para garantir a eficácia e a produtividade das aplicações e dos processos de negócio. Ela permite otimizar consultas, identificar gargalos, melhorar a escalabilidade, monitorar o desempenho e garantir a satisfação do usuário. Tornando possível a tomada de medidas proativas para melhorar o desempenho, maximizar a eficiência e obter o máximo valor do seu banco de dados Oracle.

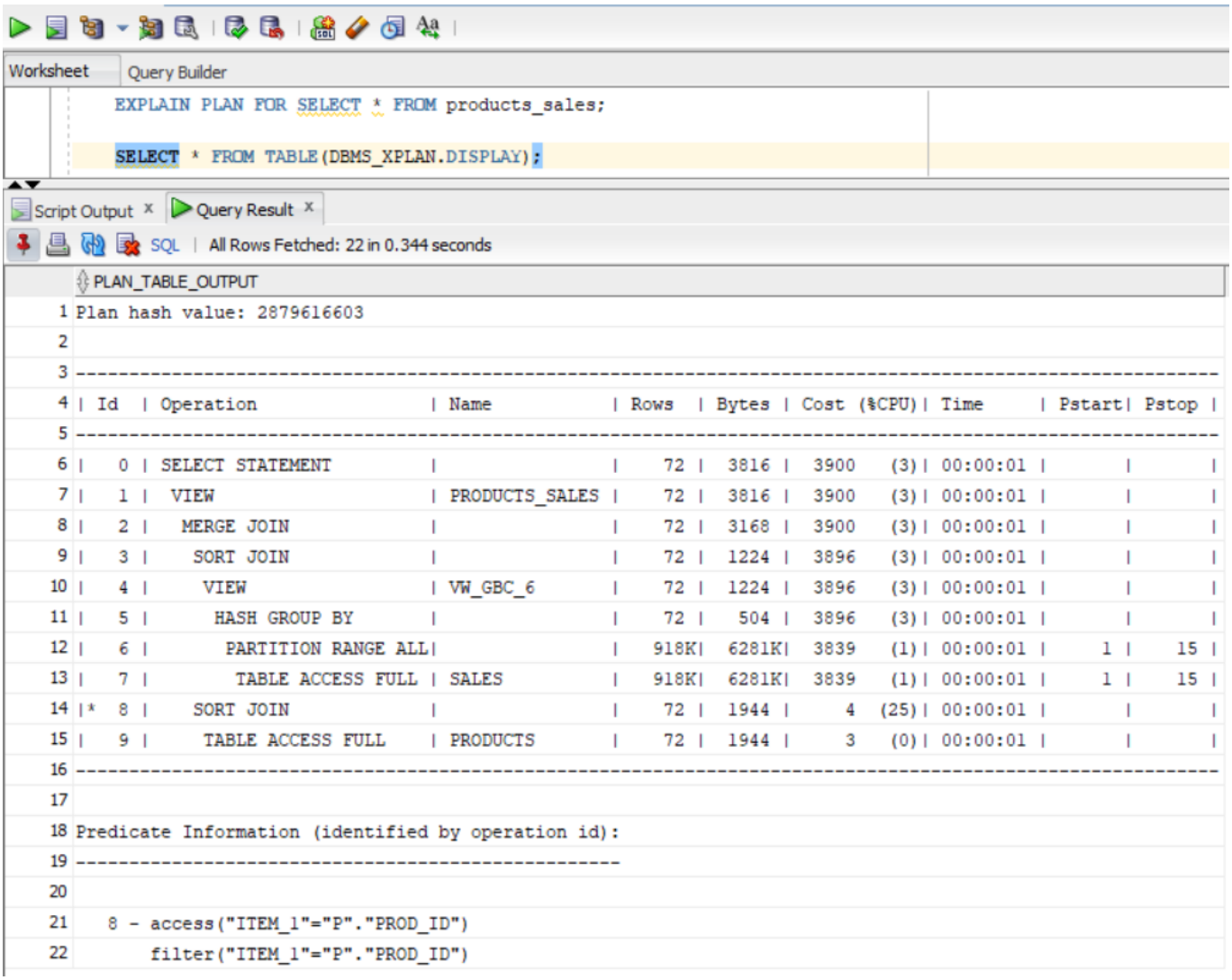
Plano/Relatório de Consulta

Ao rodar o comando `EXPLAIN PLAN`, o plano de execução de uma consulta é salvo na tabela `DBMS_XPLAN.DISPLAY`, cujos resultados podem ser vizualizados (usando o comando `SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);`. Esse comando obtêm a sequência de operações que o otimizador de consultas do Oracle planeja executar para processar a consulta. O `EXPLAIN PLAN` permite avaliar a eficiência da consulta e identificar possíveis problemas de desempenho.

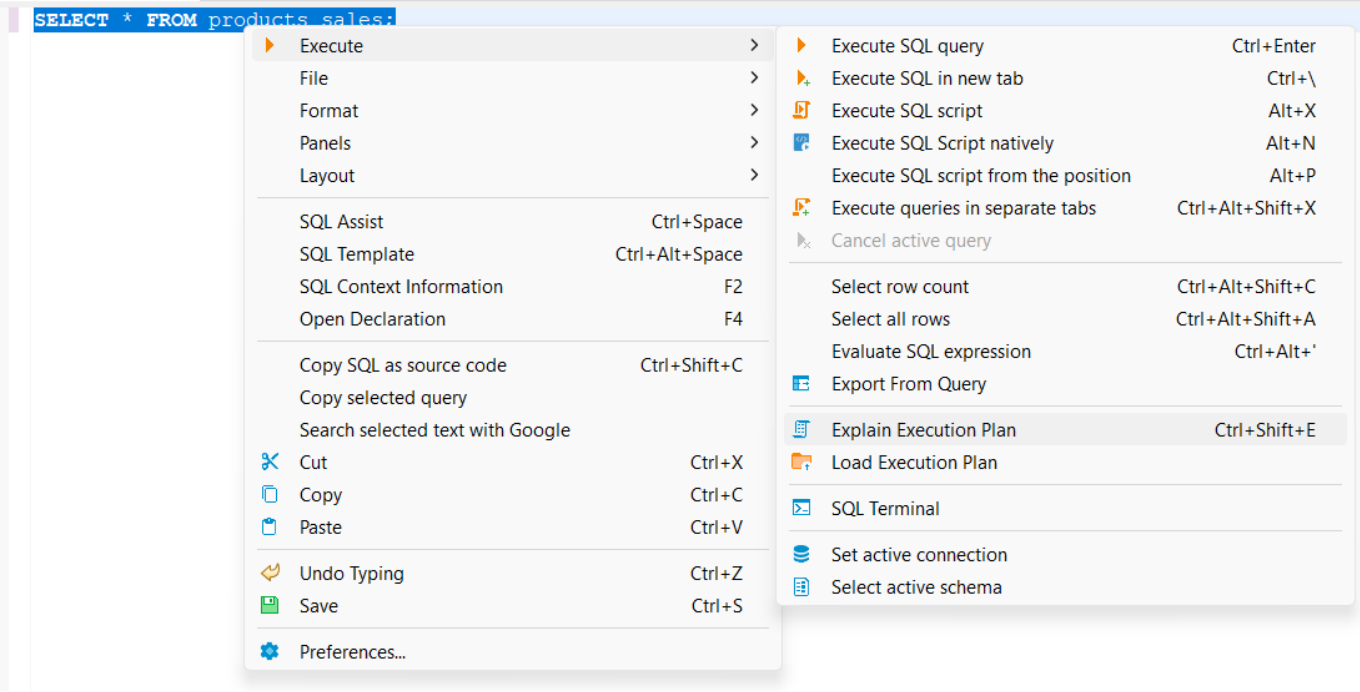
Exemplo (usando a visão criada anteriormente)

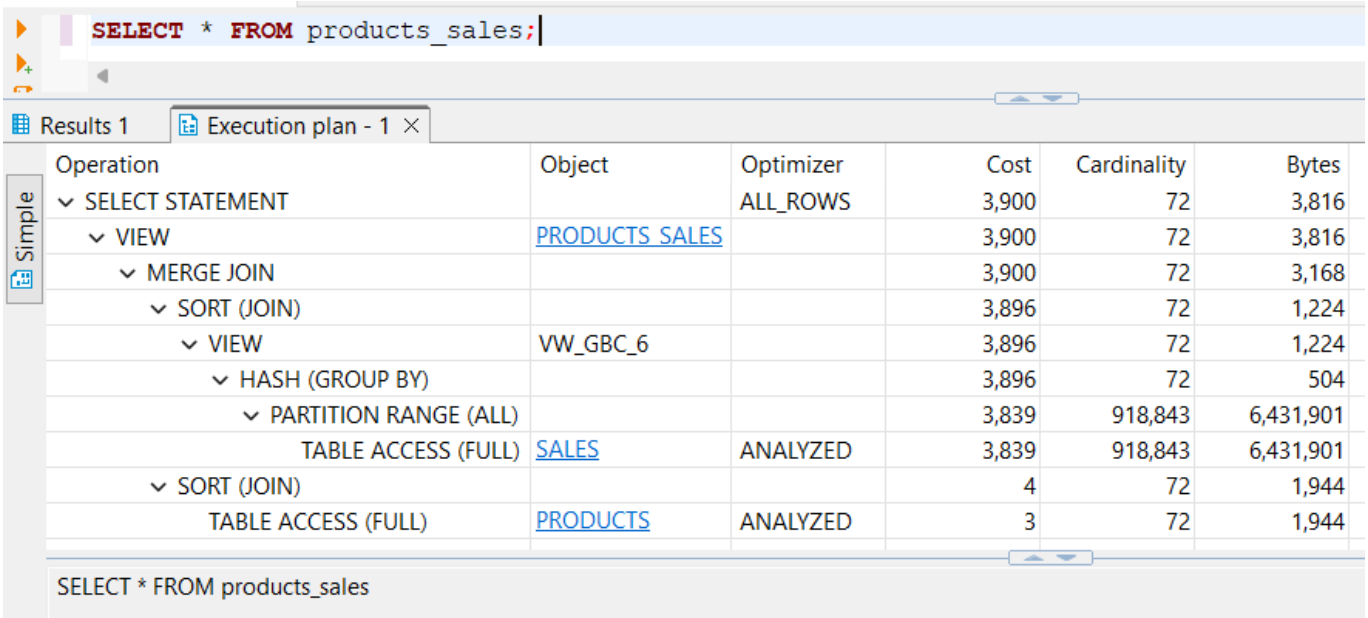
```
EXPLAIN PLAN FOR SELECT * FROM products_sales;

SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

Assim, o seguinte plano de consulta é mostrado:



Usando o DBeaver, também é possível selecionar a consulta desejada, clicar com o botão direito sobre ela, e selecionar *Execute* > *Explain Execution Plan* para obter o plano de execução de uma consulta:





Índices

Os índices são usados no banco de dados Oracle para melhorar o desempenho das consultas e acelerar a recuperação de dados. Eles são estruturas de dados adicionais que armazenam valores de colunas específicas em uma tabela e fornecem um caminho rápido para localizar registros com base nesses valores. No entanto, é importante lembrar que a criação de índices também tem algumas considerações. Os índices ocupam espaço em disco e podem afetar o desempenho durante as operações de modificação de dados. Portanto, é necessário encontrar um equilíbrio entre a criação de índices para melhorar o desempenho das consultas e evitar o excesso de índices, que podem ter um impacto negativo no desempenho geral do banco de dados. Os principais tipos de índices no banco de dados Oracle são:

Índice de Chave Única

O índice de chave única garante que os valores da coluna indexada sejam exclusivos, sem permitir duplicatas. Ele é criado automaticamente quando você define uma restrição de chave primária ou chave única em uma

tabela. Não há necessidade de criar explicitamente um índice de chave única, pois ele é criado automaticamente quando a restrição é definida.

Índice de Chave Externa

O índice de chave externa é criado automaticamente quando você define uma restrição de chave estrangeira em uma tabela. Ele garante a integridade referencial entre duas tabelas. Assim como o índice de chave única, não é necessário criar explicitamente um índice de chave externa.

Índice B-tree (padrão)

O índice B-tree é o tipo de índice mais comumente usado no Oracle. Ele é criado automaticamente quando você cria uma restrição de chave primária ou chave estrangeira em uma tabela. No entanto, você também pode criar um índice B-tree explicitamente usando o seguinte comando:

```
CREATE INDEX index_name ON table_name (column_name);
```

Índice Bitmap

O índice Bitmap é útil para colunas com baixa cardinalidade, ou seja, com um número limitado de valores distintos. Ele usa uma estrutura bitmap para representar os valores dos registros. Para criar um índice bitmap, você pode usar o seguinte comando:

```
CREATE BITMAP INDEX index_name ON table_name (column_name);
```

Índice Hash

O índice hash é adequado para consultas de igualdade rápida, onde os valores da chave de pesquisa são distribuídos uniformemente. Para criar um índice hash em Oracle, você pode usar o seguinte comando:

```
CREATE INDEX index_name
ON table_name (column_name)
INDEXTYPE IS HASH;
```

Índice de Função

O índice de função permite criar um índice em uma expressão ou função aplicada a uma coluna. Isso é útil quando você precisa indexar os resultados de uma função para melhorar o desempenho das consultas. Para criar um índice de função, você pode usar o seguinte comando:

```
CREATE INDEX index_name ON table_name (function_name(column_name));
```

Para excluir um índice no Oracle, você pode usar o comando DROP INDEX. A sintaxe para excluir um índice é a seguinte:

```
DROP INDEX index_name;
```

Parâmetros de Configuração

Uso de *hints*

É possível usar *hints* (sugestões) nas consultas para especificar se um índice deve ser usado ou ignorado pelo otimizador de consultas. Os *hints* fornecem instruções diretas ao otimizador sobre como executar a consulta. Aqui estão alguns exemplos de *hints* relacionados a índices:

- `INDEX(table_name index_name)`: Força o uso específico de um índice em uma tabela.
- `NO_INDEX(table_name index_name)`: Instrui o otimizador a ignorar um índice específico em uma tabela.

Uso de parâmetros de sessão

É possível definir os parâmetros de sessão `OPTIMIZER_USE_INVISIBLE_INDEXES` e `OPTIMIZER_IGNORE_HINTS` para controlar o uso de índices em consultas. Esses parâmetros afetam todas as consultas executadas na sessão. `OPTIMIZER_USE_INVISIBLE_INDEXES`: Se definido como "TRUE", permite que o otimizador use índices invisíveis em consultas. `OPTIMIZER_IGNORE_HINTS`: Se definido como "TRUE", instrui o otimizador a ignorar todos os hints nas consultas.

Desativação temporária de índices

É possível desativar temporariamente um índice usando a cláusula `ALTER INDEX` com a opção `UNUSABLE`. Isso fará com que o Oracle trate o índice como não utilizável, e o otimizador não o considerará durante a execução das consultas. No entanto, a estrutura do índice ainda é mantida no banco de dados. `ALTER INDEX index_name UNUSABLE;` É possível reativar um índice previamente desativado usando a cláusula `ALTER INDEX` com a opção `REBUILD`. Isso reconstruirá o índice e o tornará utilizável novamente pelo otimizador: `ALTER INDEX index_name REBUILD;`

Exemplo: Criando um Índice

Exemplificando criação de índice, plano de consulta e parâmetros de configuração na amostra escolhida.

Criando índice bitmap na coluna prod_status da tabela products (criado por padrão ao rodar o script):

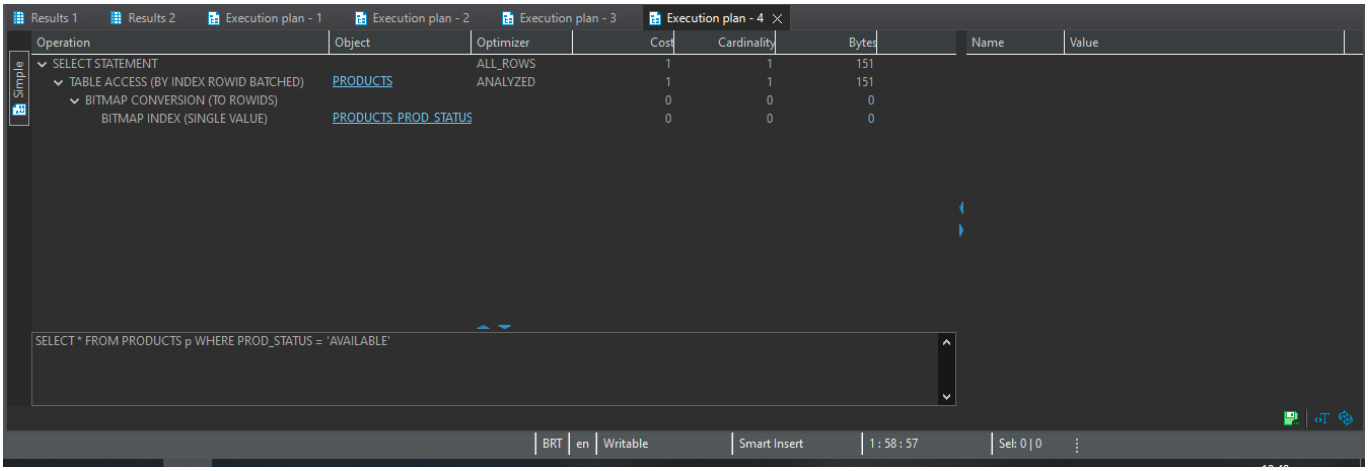
```
CREATE BITMAP INDEX products_prod_status_bix
   ON products(prod_status);
```

Plano de consulta para a seleção dos produtos que tem o status = 'AVAILABLE':

```
```

```
EXPLAIN PLAN FOR SELECT prod_name, prod_status FROM products WHERE prod_status =
'AVAILABLE';
```

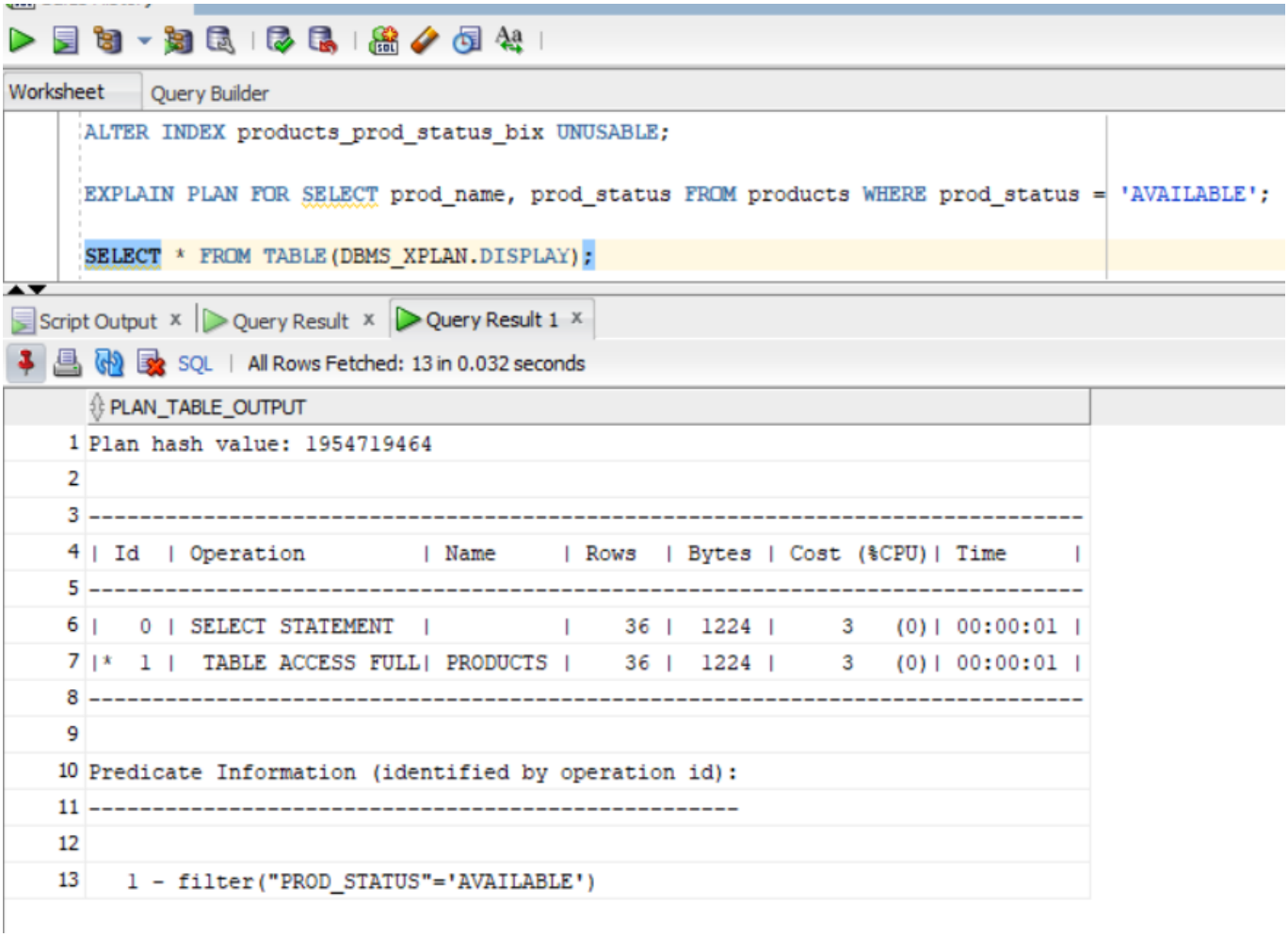
Resultado do plano de consulta com índice 'products_prod_status_bix' ativado:



Desativando o índice bitmap criado:

```
ALTER INDEX products_prod_status_bix UNUSABLE;
```

Resultado do plano de consulta com índice 'products_prod_status_bix' desativado:



Reativando o índice bitmap products_prod_status_bix:

```
ALTER INDEX products_prod_status_bix REBUILD;
```

Estatísticas

Para melhorar o desempenho de um banco de dados Oracle, é essencial manter as estatísticas atualizadas regularmente. As estatísticas são informações sobre a distribuição de dados nas tabelas e índices do banco de dados, e elas são usadas pelo otimizador de consultas para determinar o plano de execução mais eficiente. É importante lembrar que a atualização das estatísticas deve ser realizada com cautela e planejamento adequado, especialmente em ambientes de produção. Recomenda-se realizar testes e análises de desempenho antes e depois da atualização das estatísticas para avaliar os impactos no desempenho do banco de dados.

Atualização Automática

O Oracle possui um recurso de atualização automática de estatísticas chamado "Automated Maintenance Tasks". Esse recurso pode ser configurado para executar automaticamente a coleta de estatísticas em tabelas e índices em intervalos regulares. Para ativar a atualização automática, você precisa habilitar e configurar o "Automatic Optimizer Statistics Collection". Isso pode ser feito executando os seguintes comandos:

```
BEGIN
  DBMS_AUTO_TASK_ADMIN.ENABLE(
    client_name     => 'auto optimizer stats collection',
    operation       => NULL,
    window_name     => NULL);
END;
/
```

Procedimento DBMS_STATS.GATHER_DATABASE_STATS

O procedimento `DBMS_STATS.GATHER_DATABASE_STATS` permite coletar estatísticas para todas as tabelas e índices em todo o banco de dados. Ele pode ser executado usando o seguinte comando:

```
BEGIN
  DBMS_STATS.GATHER_DATABASE_STATS(
    options => 'GATHER AUTO');
END;
/
```

Procedimento DBMS_STATS.GATHER_TABLE_STATS

É possível usar o procedimento `DBMS_STATS.GATHER_TABLE_STATS` para coletar estatísticas em tabelas e índices específicos. Isso permite que você tenha um controle mais granular sobre quais objetos deseja

atualizar as estatísticas. O seu uso é muito similar ao do procedimento `DBMS_STATS.GATHER_DATABASE_STATS`.

Comparação de Desempenho: Visão X Visão Materializada

Usando os conceitos relatório de plano de execução, é possível comparar o custo da visão e da visão materializada que foram criadas.

Exemplo

Para exibir o plano de execução da visão products_sales, é possível rodar o seguinte comando:

```
EXPLAIN PLAN FOR SELECT * FROM products_sales;
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

Que retorna o seguinte resultado:

```
Plan hash value: 2879616603

-------------------------------------------------------------------------------------
-------------------------
| Id  | Operation                | Name           | Rows  | Bytes | Cost (%CPU)|
Time     | Pstart| Pstop |
-------------------------------------------------------------------------------------
-------------------------
|   0 | SELECT STATEMENT         |                |    72 |  3816 |  3900   (3)|
00:00:01 |       |       |
|   1 |  VIEW                    | PRODUCTS_SALES |    72 |  3816 |  3900   (3)|
00:00:01 |       |       |
|   2 |   MERGE JOIN             |                |    72 |  3168 |  3900   (3)|
00:00:01 |       |       |
|   3 |    SORT JOIN             |                |    72 |  1224 |  3896   (3)|
00:00:01 |       |       |
|   4 |     VIEW                 | VW_GBC_6       |    72 |  1224 |  3896   (3)|
00:00:01 |       |       |
|   5 |      HASH GROUP BY       |                |    72 |   504 |  3896   (3)|
00:00:01 |       |       |
|   6 |       PARTITION RANGE ALL|                |   918K|  6281K|  3839   (1)|
00:00:01 |     1 |    15 |
|   7 |        TABLE ACCESS FULL | SALES          |   918K|  6281K|  3839   (1)|
00:00:01 |     1 |    15 |
|*  8 |    SORT JOIN             |                |    72 |  1944 |     4  (25)|
00:00:01 |       |       |
|   9 |     TABLE ACCESS FULL    | PRODUCTS       |    72 |  1944 |     3   (0)|
00:00:01 |       |       |
-------------------------------------------------------------------------------------
-------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
```

```
   8 - access("ITEM_1"="P"."PROD_ID")
       filter("ITEM_1"="P"."PROD_ID")
```

Já para a visão materializada (que retorna a mesma consulta) products_sales_mv, o comando a ser executado para analisar o plano de execução é:

```
EXPLAIN PLAN FOR SELECT * FROM products_sales_mv;
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

Que por sua vez retorna:

```
Plan hash value: 3783523653

-------------------------------------------------------------------------------
--------
| Id  | Operation            | Name              | Rows  | Bytes | Cost (%CPU)|
Time     |
-------------------------------------------------------------------------------
--------
|   0 | SELECT STATEMENT     |                   |    72 |  2304 |     3   (0)|
00:00:01 |
|   1 |  MAT_VIEW ACCESS FULL| PRODUCTS_SALES_MV |    72 |  2304 |     3   (0)|
00:00:01 |
-------------------------------------------------------------------------------
--------
```

Assim, fica claro que a diferença entre ambas é bem grande. O custo da visão chegou a 3900, com um uso máximo de CPU de 25%. Enquanto isso, o custo máximo da visão materializada foi de apenas 3, sem uso significativo de CPU.

Conclusão

O OracleDB é um banco de dados relacional que surgiu no final da década de 70 e desde seu surgimento veio evoluindo e se tornando um dos SGBD (Sistema Gerenciador de Banco de Dados), que graças as suas qualidades - as quais pode-se citar segurança, escalonamento e desempenho – se tornou dentro do mercado um dos banco de dados mais conhecidos e utilizados atualmente. A partir da análise e estudo sobre os temas visões, visões materializada, desempenho e transações foi possível visualizar que o OracleDB disponibiliza seus serviços de forma completa e abrangente. Com recursos avançados de visões e visões materializadas, o OracleDB permite uma melhor organização e simplificação do acesso aos dados, melhorando a eficiência e a facilidade de uso. Além disto o OracleDB procura disponibilizar diversos métodos para utilizar suas soluções, como por exemplo em desempenho onde com conhecimento sobre o produto você pode definir suas prioridades no uso de índice. Esses atributos combinados fazem do OracleDB uma escolha sólida para organizações que exigem um sistema de gerenciamento de banco de dados confiável, seguro e de alto desempenho.

Referências

- SRINIVAS, Apoorva. **Installation of the Sample Schemas**. Oracle Help Center. Disponível em: https://docs.oracle.com/en/database/oracle/oracle-database/21/comsc/installing-sample-schemas.html. Acesso em: 18 jun. 2023.
- KUSH, Frederick; HERMANN BAER, Mark Bauer; POTINENI, Padmaja. **Refreshing Materialized Views**. Oracle Help Center. Disponível em: https://docs.oracle.com/en/database/oracle/oracle-database/12.2/dwhsg/refreshing-materialized-views.html. Acesso em: 18 jun. 2023.
- ASHDOWN, Lance; KEESLING, Donna; KYTE, Tom. **Database Concepts**. Oracle Help Center. Disponível em: https://docs.oracle.com/en/database/oracle/oracle-database/21/cncpt/. Acesso em: 18 jun. 2023.
- **Oracle Database Transactions| Oracle Transactions | COMMIT, ROLLBACK, SAVEPOINT | Oracle Tutorial**. [s.l.: s.n.], 2022. Disponível em: https://www.youtube.com/watch?v=T7V8fgwtRB4. Acesso em: 18 jun. 2023.
- **Artigo da SQL Magazine 36 - Índices no Oracle - Parte I**. DevMedia. Disponível em: https://www.devmedia.com.br/artigo-da-sql-magazine-36-indices-no-oracle-parte-i/6835. Acesso em: 18 jun. 2023.
- **Materialized Views in Oracle. oracle-base.com**. Disponível em: https://oracle-base.com/articles/misc/materialized-views. Acesso em: 19 jun. 2023.