Controle			
Versão	**Data**	**Autor**	**Notas da Revisão**
1.0	15/09/2024	Ronaldo Costa Miranda	

Equipe		Período de:	**/**/**	Até:	**/**/**

Objetivos deste documento

Direcionar o uso da Package PROCESSAR_XML.

Cenário:

Desafio Técnico em PL/SQL:

Processamento de Pedidos com XML e Consulta de Itens com SYS_REFCURSOR

Descrição:

Criar um Package em PL/SQL que processa um XML no formato CLOB.

Contendo:

Um pedido e uma lista de produtos, e também forneça uma consulta para retornar todos os itens de um pedido.

O package deve garantir que não haja produtos duplicados no mesmo pedido.

Requisitos Funcionais:

Criar os objetos de Banco de dados Oracle conforme ordem e scripts.

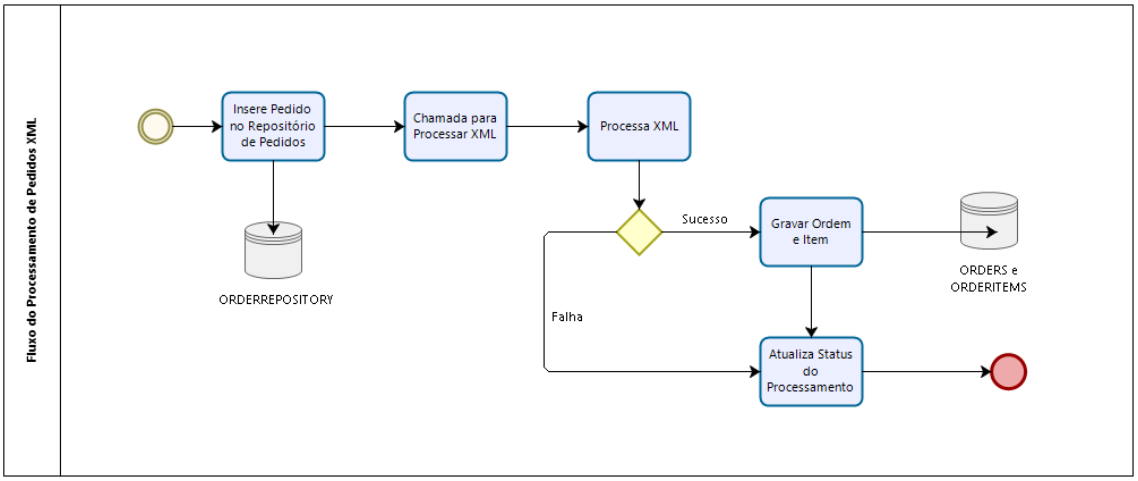
Criação da estrutura de tabelas necessárias para gravação no Repositório e Ordens e Itens após o processamento com sucesso.

Os arquivos seguem disponíveis no repositório GitHub em conformidade com as diretrizes do desafio. Seguem os nomes dos arquivos:
CriacaoTabelas.sql
Package.sql

Arquivos com os Inserts para testes
InsertXMLComItens.sql
InsertXMLComItensDuplicados.sql
InsertXMLSemItens.sql

Arquivos de blocos anônimos para testes
BlocoAnonimoProcessarPedido.sql
BlocoAnonimoConsultaItensPedido.sql

Descrição do fluxo: Processar Pedido



CLOB
O tipo CLOB é ideal para armazenar grandes volumes de dados XML, e Oracle fornece várias ferramentas para manipulação eficiente de documentos XML armazenados nesse formato
É um tipo de dado externo do Oracle. O tipo de dado externo CLOB armazena dados de tamanho fixo ou variável no formato de caracteres. Um CLOB pode armazenar até 4 gigabytes de dados de caracteres.

XML
A Extensible Markup Language (XML) permite definir e armazenar dados de maneira compartilhável. A XML oferece suporte ao intercâmbio de informações entre sistemas de computador, como sites, bancos de dados e aplicações de terceiros

Uso do SYS_REFCURSOR
O **SYS_REFCURSOR** é uma ferramenta poderosa no Oracle PL/SQL, usada para lidar com consultas dinâmicas.

Bloco PL/SQL Anônimo
Não possui um nome, e é utilizado para executar código sem a necessidade de criá-lo permanentemente no banco de dados (como seria o caso de uma procedure ou função).

Package - Um package é uma coleção de objetos PL/SQL agrupados logicamente sob o nome de pacote.

A package foi criada com o nome PROCESSAR_XML.
Dentro dela estão agrupados dois objetos: a Procedure PROCESSAR_PEDIDO e a Function RETORNAR_ITENS_PEDIDO.

A Procedure PROCEDURE PROCESSAR_PEDIDO recebe dois parâmetros:
(P_XML_PEDIDO CLOB, p_status OUT NUMBER)

Os dados que serão processados por essa Procedure são oriundos da tabela ORDERREPOSITORY.

Para cumprimento dos requisitos do desafio PLSQL iniciei pela criação das tabelas:
Tabela ORDERREPOSITORY – Esse objeto tem a finalidade de armazenar as inserções realizadas por um sistema qualquer que tenha a função de disparar os XML's. Poderia ser mais prático e apenas criar a Package realizando chamadas por intermédio de um bloco anônimo de forma direta e unitária, alcançaria o mesmo efeito, porém pensei em um sistema de controle de registros usando algumas colunas auxiliares como STATUSPROCESSAMENTO para usar o mesmo XML várias vezes podendo reprocessa-los, ao invés de manter os inserts descritos, passivos de alguma confusão na edição.

A coluna STATUSPROCESSAMENTO tem a ideia de controlar se o registro já foi processado ou não, seguindo a seguinte legenda:

STATUSPROCESSAMENTO = 0 [Não processado]
STATUSPROCESSAMENTO = 1 [Processado com sucesso]
STATUSPROCESSAMENTO = 3 [Processado com erro]

Paralelamente a coluna de DATE_PROCESSAMENTO será inserida assim que o processamento ocorrer, independente do resultado do STATUSPROCESSAMENTO.

A coluna DESCRIPTIONERROR pode ser nula e recebe a mensagem de erro tratada dentro da Package de acordo com as situações mapeadas e retornadas pela variável p_status enviada (com valor default de criação do tipo Number) e retornada pela Procedure, com valor atualizado internamente na procedure.

A gravação do pedido e itens, exige a atomicidade da transação de inserção, ou seja, que os itens sejam gravados apenas quando o pedido também estiver persistido. Por esse motivo, na Procedure PROCESSAR_PEDIDO o Commit está ao final de todo o procedimento.

Criei um tratamento de exceção geral para eventualidades na execução com o controle de uma variável auxiliar p_status para controlar os pontos conhecidos de tratamento de validações.

p_status = Inicializa 0 e se todo o procedimento de inserção ocorrer corretamente
p_status = 1 (receberá o valor 1)
p_status = 2 (quando o XML não tiver itens)
p_status = 3 (quando o XML tiver itens duplicados)

A princípio havia inserido o levantamento de exceção do tipo RAISE_APPLICATION_ERROR, entretanto, tratamento das exceções com esse tipo de recurso, impediria o controle das chamadas em Loop, que realizei através de bloco anônimo BlocoAnonimoProcessarPedido.sql, que envio como parte do procedimento para execução da Package.

Em resumo p_status pode ser incrementado, internamente com mais validações, entretanto, quando retornar, para a chamada do bloco anônimo (que poderia inclusive ser um JOB para automatizar) sempre atualizará o STATUSPROCESSAMENTO = 2, o que indica alguma validação conhecida e erro no processamento, com mensagem gravada em DESCRIPTIONERROR.

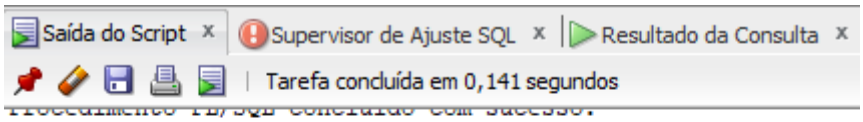
A Package garante a validação da duplicidade de itens (requisito explicito do desafio) e também de um XML que não tenha itens. Evidentemente são validações simples e certamente existem outras a serem feitas, porém ao equilibrar o tempo para a entrega e a velocidade em que programo, percebo que a melhor estratégia é manter com essas validações nesse momento.

Descrição do fluxo: RETORNAR_ITENS_PEDIDO

Completando os requisitos do desafio, criei uma consulta na package com a Function RETORNAR_ITENS_PEDIDO utilizando o recurso SYS_REFCURSOR.

Essa function recebe como parâmetro o (p_order_id NUMBER), ou seja, o pedido. E lista todos os itens dos pedidos que foram processados.

Para essa visualização de resultados, a procedure anterior, deverá ter êxito em alguma gravação.
Na implementação utilizei o recurso DBMS_OUTPUT.PUT_LINE que exibe no Client os resultados, como abaixo.




```
Item ID: 1, Produto: Produto A, Quantidade: 2, Preço: 19.99
Item ID: 2, Produto: Produto B, Quantidade: 3, Preço: 12.99


Procedimento PL/SQL concluído com sucesso.
```

Para isso, é necessário habilitar o recurso SERVEROUTPUT caso não esteja habilitado.
--HABILITA O SERVEROUTPUT DO SQLDEVELOPER
SET SERVEROUTPUT ON;

O modelo acima está formatado segundo a chamada do bloco anônimo BlocoAnonimoConsultaItensPedido.sql, que seguirá com a entrega.

Outra forma, pragmática de visualizar esse resultado é executando um select simples diretamente na tabela de itens, ou relacionando-a com a tabela de Ordens.

```
17  SELECT * FROM ORDERITEMS ITM
18  INNER JOIN ORDERS ORD ON ORD.ORDERID = ITM.ORDERID
19  WHERE ITM.ORDERID = :IDPEDIDO;
```

Resultado da Consulta

SQL | Todas as Linhas Extraídas: 2 em 0,044 segundos

	ORDERITEMID	ORDERID	PRODUCTNAME	QUANTITY	PRICE	ORDERID_1	CUSTOMERNAME	ORDERDATE
1	1	1	Produto A	2	19.99	1	Ronaldo	14/09/24 00:00:00
2	2	1	Produto B	3	12.99	1	Ronaldo	14/09/24 00:00:00

Pontos de Observações:

DBMS_XMLDOM é um pacote mais avançado para manipulação do modelo de objetos XML no Oracle. Particularmente eu não conhecia esse recurso do Oracle. Tentei utilizá-lo sem sucesso, porque exigia privilégios que eu não tinha na instalação do SGBD e na máquina que utilizei.
Então, diante disso, tive que procurar outro recurso e encontrei o XMLTABLE, que também nunca havia utilizado, pois as experiências que tive com mapeamento de arquivos XML, foram utilizando linguagem de alto nível (C#). A experiência foi interessante.

XMLTABLE mapeia o resultado de uma avaliação XQuery em linhas e colunas relacionais. Você pode consultar o resultado retornado pela função como uma tabela relacional virtual usando SQL.

Manipular o XML

Tive Dificuldades em utilizar os pacotes Oracle, em alguns pontos específicos, por exemplo: não conseguia navegar na árvore XML para pegar o valor do ID da Ordem no nó ancestral. Isso me fez perder muito tempo.
Posteriormente, percebi que a dificuldade em realizar esse procedimento era em razão de o pedido ainda não ter sido persistido em banco, o que era um requisito que provoquei propositalmente, visto que precisaria garantir a atomicidade da transação.
Por fim, consegui ter êxito, guardando o valor o ID em uma variável para usar posteriormente.

	Assinatura	**Data**
Analista de Desenvolvimento	Ronaldo Costa Miranda	15/09/2024