ENGENHARIA DE COMPUTAÇÃO

ATIVIDADE EM SALA

DISCIPLINA: BDAE6 – Banco de Dados 2 **Prof(a): Helen Freitas**	**DATA:** 05/08/2020
Nome: Vinicius de Souza Santos Prontuário: BI3008061	

1.0 OBJETIVO:

Pesquisar sobre:

1. Definição de Sistema de Gerenciamento de Banco de Dados
2. Sopa de letrinhas: DDL, DML, DQL, DTL, DCL
3. Quais tipos de dados são aceitos pelo Banco de Dados Oracle
4. Inner (outer) Join x = (ver sintaxe no oracle)

2.0 Desenvolvimento

2.1 Definição de Sistema de Gerenciamento de Banco de Dados:

Um Sistema de Gerenciamento de Banco de Dados (SGBD) – do inglês Data Base Management System (DBMS) – é o conjunto de programas de computador (softwares) responsáveis pelo gerenciamento de uma base de dados. Seu principal objetivo é retirar da aplicação cliente a responsabilidade de gerenciar o acesso, a manipulação e a organização dos dados. O SGBD disponibiliza uma interface para que seus clientes possam incluir, alterar ou consultar dados previamente armazenados. Em bancos de dados relacionais a interface é constituída pelas APIs (Application Programming Interface) ou drivers do SGBD, que executam comandos na linguagem SQL (Structured Query Language).

2.2 Sopa de letrinhas: DDL, DML, DQL, DTL, DCL

2.2.1 DDL: Dias da Data Líquida / DML: Data Manipulation Language

Apesar da linguagem SQL ser uma única linguagem, ela é dividida em tipos de acordo com a funcionalidade dos comandos.

DDL e DML são tipos de linguagem SQL.

A DDL, Data Definition Language ou LInguagem de Definição de Dados, apesar do nome não interage com os dados e sim com os objetos do banco.

São comandos desse tipo o CREATE, o ALTER e o DROP.

Já a DML, Data Manipulation Language, ou Linguagem de Manipulação de Dados. interage diretamente com os dados dentro das tabelas.

São comandos do DML o INSERT, UPDATE e DELETE.

2.2.2 DQL: Data Query Language

representando a principal instrução SQL, o SELECT, comando que realiza consultas no SGBD (Sistema de Gerenciamento de Banco de Dados).

2.2.3 DTL: Data Transaction Language

Os comandos DTL são responsáveis por gerenciar diferentes transações ocorridas dentro de um banco de dados. Ele é dividido em 3 comandos:

BEGIN TRAN (OU BEGIN TRANSACTION) – Marca o começo de uma transação no banco de dados que pode ser completada ou não.

COMMIT – Envia todos os dados da transação permanentemente para o banco de dados.

ROLLBACK – Desfaz as alterações feitas na transação realizada.

Comandos como Insert, Update e Delete, são processos de transações de dados.

2.2.4 DCL: Data Control Language

são comandos responsáveis pelo controle dos dados:

GRANT – Concede permissões ao usuário de realizar operações

REVOKE – Retira (ou restringe) permissões do usuário de executar operações

2.3 Quais tipos de dados são aceitos pelo Banco de Dados Oracle

Os tipos de dados Oracle e os tipos de dados do Microsoft SQL Server nem sempre são correspondências exatas. Onde possível, o tipo de dados correspondente é selecionado automaticamente ao publicar uma tabela de Oracle. Em casos em que o mapeamento de um único tipo de dados não é claro, mapeamentos alternativos de tipo de dados são fornecidos. Para obter informações sobre como selecionar mapeamentos alternativos, consulte "Especificando Mapeamentos Alternativos de Tipos de Dados", mais adiante neste tópico.

A tabela a seguir mostra como os tipos de dados são mapeados por padrão entre o Oracle e o SQL Server quando os dados são movidos de um Publicador Oracle para o Distribuidor do SQL Server. A coluna de Alternativas indica se mapeamentos alternativos estão disponíveis.

Tipo de dados de Oracle	Tipo de dados do SQL Server
BFILE	VARBINARY(MAX)
BLOB	VARBINARY(MAX)
CHAR([1-2000])	CHAR([1-2000])
CLOB	VARCHAR(MAX)
DATE	DATETIME
FLOAT	FLOAT
FLOAT([1-53])	FLOAT([1-53])
FLOAT([54-126])	FLOAT
INT	NUMERIC(38)
INTERVAL	DATETIME
LONG	VARCHAR(MAX)
LONG RAW	IMAGE
NCHAR([1-1000])	NCHAR([1-1000])
NCLOB	NVARCHAR(MAX)
NUMBER	FLOAT
NUMBER([1-38])	NUMERIC([1-38])
NUMBER([0-38],[1-38])	NUMERIC([0-38],[1-38])
NVARCHAR2 ([1-2000])	NVARCHAR([1-2000])
RAW ([1-2000])	VARBINARY([1-2000])
real	FLOAT
ROWID	CHAR(18)
timestamp	DATETIME
TIMESTAMP(0-7)	DATETIME
TIMESTAMP(8-9)	DATETIME
TIMESTAMP(0-7) WITH TIME ZONE	VARCHAR(37)
TIMESTAMP(8-9) WITH TIME ZONE	VARCHAR(37)
TIMESTAMP(0-7) WITH LOCAL TIME ZONE	VARCHAR(37)
TIMESTAMP(8-9) WITH LOCAL TIME ZONE	VARCHAR(37)
UROWID	CHAR(18)
VARCHAR2([1-4000])	VARCHAR([1-4000])

2.4 Inner (outer) Join x = (ver sintaxe no oracle):

O *Join* ou *Inner Join* é uma consulta que combina registros de duas ou mais tabelas. A condição de junção compara duas colunas de diferentes tabelas, combinando pares de registros em cada tabela, sendo avaliada como verdadeiro se a combinação for encontrada em ambas as tabelas. A **Figura 1** representa este conceito enquanto que a **Listagem 1** indica como ele pode ser implementado.

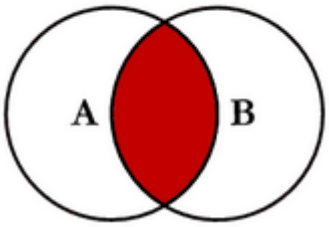

Figura 1. Representação do join.

Listagem 1. Implementação do join.

```
select

a.id,

b.data

from

Tabela_A a,

Tabela_B b,

where

a.id = b.id

select

a.id,

b.data

from

Tabela_A a INNER OUTER JOIN Tabela_B b

on (a.id = b.id)
```

Já o *Outer Join* estende o resultado do *inner join* podendo retornar todos os registros de ambas as tabelas mesmo se a combinação for avaliada como falso, ou seja, combinando registros

mesmo com desigualdades. Ele é subdividido em *right outer join*, *left outer join* e *full outer join*, como vemos abaixo.

O *Right outer join* foi pensado em uma junção entre uma tabela A e uma tabela B de forma a retornar todos os registros da tabela B do universo de dados em questão. Sendo assim, todos os registros do universo de dados em questão da tabela B serão selecionados e quando a combinação da junção for falso, valores classificados como *null* serão listados para os campos selecionados da tabela A. A **Figura 2** representa este conceito enquanto que a **Listagem 2** indica como ele pode ser implementado.

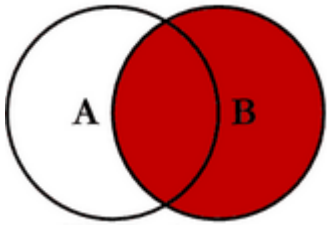

Figura 2. Representação do right outer join.

Listagem 2. Implementação do right outer join.

```
select

a.id,

b.data

from

Tabela_A a,

Tabela_B b,

where

a.id = b.id(+)

select

a.id,

b.data

from

Tabela_A a RIGHT OUTER JOIN Tabela_B b

on (a.id = b.id)
```

Já o *Left outer join* foi definido pensando em uma junção entre uma tabela A e uma tabela B, para retornar todos os registros da tabela A do universo de dados em questão. Sendo assim,

todos os registros do universo de dados em questão da tabela A serão selecionados e quando a combinação da junção for falso, valores classificados como *null* serão listados para os campos selecionados da tabela B. A **Figura 3** representa este conceito enquanto que a **Listagem 3** indica como ele pode ser implementado.

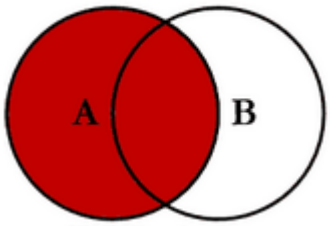

Figura 3. Representação do left outer join.

Listagem 3. Implementação do left outer join.

```
select

a.id,

b.data

from

Tabela_A a,

Tabela_B b,

where

a.id(+) = b.id

select

a.id,

b.data

from Tabela_A a LEFT OUTER JOIN Tabela_B b on (a.id = b.id)
```

Por fim, o *Full outer join* foi definido pensando em uma junção entre uma tabela A e uma tabela B, para retornar todos os registros do universo de dados em questão de ambas as tabelas. Sendo assim, todos os registros do universo de dados em questão de ambas as tabelas serão selecionados e quando a combinação da junção for falso, valores classificados como *null* serão listados para os campos selecionados da tabela em questão. A **Figura 4** representa este conceito enquanto que a **Listagem 4** indica como ele pode ser implementado.

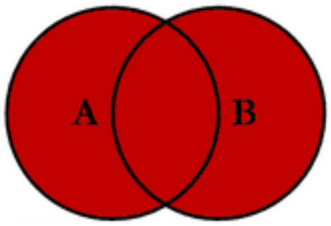

Figura 4. Representação do full outer join.

Listagem 4. Implementação do full outer join.

```
select

a.id,

b.data

from Tabela_A a FULL OUTER JOIN Tabela_B b on (a.id = b.id)
```

As definições são bem simples e aos olhos de um desenvolvedor de sistemas é uma solução eficaz. Mas aí está o início de um possível problema. Como sabemos, a tendência de uma base de dados é o seu crescimento com o passar do tempo, de acordo com suas operações DML (inserts, updates e deletes). Se esse recurso não for cuidadosamente, poderá acarretar em problemas de construções SQL para consultas, conforme exemplos:

· Produto Cartesiano: não é bem um tipo de junção, mas resulta no cruzamento de cada registro da tabela A com todos os registros da tabela B;

· *Table access full*: é uma consequência do produto cartesiano. Esse evento ocorre quando não há restrição de dados na tabela em questão ou quando não há índices suficientes para suporte à consulta, gerando um acesso físico diretamente nos arquivos de dados onde os dados da tabela estão armazenados;

· Seleção de dados desnecessários.

Quando existe *outer joins* em consultas, o otimizador de consulta é deixado de lado tornando-a complexa e não performática, pois também anula a utilização dos índices para colunas em questão, como destacado anteriormente. O otimizador é o principal componente para a execução de um comando de seleção, sendo responsável pela identificação da melhor forma de resolvê-lo.

X Vinicius Santos

Assinado de forma digital por Vinicius Santos
DN: cn=Vinicius Santos, o=Instituto Federal de Sao Paulo, ou=Birigui, email=vinicius.souza1@aluno.ifsp.edu.br, c=BR
Versão do Adobe Acrobat: 2018.011.20038

Vinicius de Souza Santos
Estudante de BDAE6