**HAVING**: Só pode ser usado após o group by e somente com expressões de agregação

**ALL**: operador de comparação que só vai me trazer resultado se os valores que estão sendo comparados, atendam a comparação de todos os valores retornados

**ANY**: operador de comparação que só vai me retornar verdadeiro se pelo menos um dos valores selecionados atender a condição dos valores que estão nos parênteses.

**IN**: operador que vai me trazer os dados de cada elemento que estiver dentro do parênteses

Quando eu faço um UNION com valores nulos como por ex:

SELECT e.department\_id, TO\_NUMBER(null) location, e.hire\_date

FROM employees e

UNION

SELECT department\_id, location\_id , TO\_DATE(null)

FROM departments

Ele sempre vai trazer algum valor nulo, pois ele vai pesquisar na primeira tabela, e consequentemente a localização vai esta nula. Da mesma maneira funciona com a segunda tabela em que, por sua vez, a coluna de data vai estar nula.

Quando se cria uma view, e deseja adicionar uma linha a ela, você não poderá adicionar a mesma caso a view não tenha colunas que são NOT NULL na tabela base, pois o SGBD não poderá adicionar automaticamente algum valor para a coluna que está nula.

Isso ocorre porque quando você atualiza uma view, ela vai **adicionar automaticamente na tabela subjacente.**

NEXTVAL – Serve para gerar o próximo valor de uma sequence

CURRVAL- Server para exibir o valor gerado pela sequence a partir do NEXTVAL

Uma sequence é independente de tabela, então seu valor vai ser o mesmo para mais de uma tabela que estiver usando ela

Porém é recomendado que cada tabela tenha sua sequence.

Índice exclusivo: São índices que são criados automaticamente quando se cria constraint PRIMARY KEY e UNIQUE

Índice não exclusivo: São índices que são criados manualmente porém não são exclusivos

OBS: É possível criar índices exclusivos manualmente, através do comando CREATE UNIQUE INDEX nome\_index ON nome\_tabela(coluna\_tabela)

Porém não é muito recomendado, é melhor que seja criado um índice exclusivo através de uma constraint

**INSERT COMUM**: INSERT INTO nome\_tabela (nome\_coluna, nome\_coluna, nome\_coluna) VALUES (nome\_coluna, nome\_coluna, nome\_coluna)

**INSERT COMUM COM SUBCONSULTA** : INSERT INTO nome\_tabela (nome\_coluna, nome\_coluna, nome\_coluna)

SELECT ...

FROM ..

WHERE ...

**VALUES**: essa cláusula é usada quando eu quero inserir dados em uma tabela, no qual as colunas que contém os valores, precisam ser **ESPECIFICADAS** devido a **DUAS** condições:

1. Quando eu faço um INSERT comum, no qual eu preciso informar os valores de cada coluna como:

INSERT INTO nome\_tabela (nome\_coluna, nome\_coluna, nome\_coluna)

VALUES (nome\_coluna, nome\_coluna, nome\_coluna)

1. Quando eu faço um INSERT utilizando uma subconsulta com **MAIS** de **UMA** tabela e essas tabelas possuem colunas **DIFERENTES** , como:

INSERT ALL INTO sal\_hist\_dept

VALUES(EMP\_ID,HIREDATE,SALARY) INTO mgr\_hist\_dept

VALUES(EMP\_ID,MANAGER,SALARY)

SELECT employee\_id EMP\_ID, hire\_date HIREDATE, salary, manager\_id

MANAGER FROM employees WHERE department\_id = 90 ;

MERGE: é uma instrução que permite atualizar, inserir e excluir dados em uma única instrução

ROLLUP: Cria subtotal de groupby de forma HIERÁRQUICA, se eu tenho REGIAO, PRODUTO e CATEGORIA no GROUP BY ele vai ler da seguinte forma:

**REGIAO NORTE SUL**

**PRODUTO A B**

**CATEGORIA**

**Eletrônicos Vestuário**

O ROLLUP vai agrupar a partir do **PRODUTO**, porque tudo o que tinha que agrupar na camada de **CATEGORIA**, o GROUPBY já fez (GROUP BY ROLLUP (região, produto, categoria))

A partir do produto, ele agrupa **PRODUTOS** do **A**, INDEPENDENTE de categoria

A partir do produto, ele agrupa PRODUTOS do **B**, INDEPENDENTE de categoria

Em seguida, ele sobe mais uma camada e agora ele agrupa por REGIAO do **NORTE**, INDEPENDENTE de produto ou categoria

Em seguida, ele sobe mais uma camada e agora ele agrupa por REGIAO do **SUL**, INDEPENDENTE de produto ou categoria

Ao final, ele soma todos valores INDEPENDENTE de **REGIAO**, **PRODUTO** e **CATEGORIA**

**O 1 sempre vai estar na coluna que estiver com o valor nulo, pois é essa coluna que esta sendo feito o agrupamento de subtotal.**

**CUBE:** Cria subtotal de GROUPBY, de forma dimensional, ou seja, não importa se o produto não é da mesma região, ele vai agrupar com a semelhança da COLUNA e não de acordo com a semelhança da coluna E SE ESTA DENTRO DA COLUNA MAIS GENÉRICA:

**REGIAO PRODUTO CATEGORIA**

**NORTE SUL A B Eletrônicos Vestuário**

O CUBE vai agrupar nesse caso as categoria porque como não existe uma hierarquia, ele vai criar um subtotal para eletrônico e vestuário independente de PRODUTO e REGIAO

Para produto, ele também vai criar independente de região

Para região, ele vai criar independente de tudo

ROLLUP: Todas as combinações de forma hierárquica, ou seja, sempre respeitando o GROUP BY

CUBE: todas as combinações possíveis, independente do GROUP BY

Hierarquia:

Funcionalidade: consultar de forma hierárquica os dados a partir de um ponto de partida

O SQL vai procurar a partir do STARWITH na tabela e de acordo com a condição dada no CONECT BY PRIOR ele vai buscar todos os subordinados de alguém

Depois ele parte para o próximo valor do startwith