Trabalhando com Views no Oracle

Veja neste artigo como criar, alterar e excluir views no banco de dados Oracle, facilitando a realização de algumas consultas de estrutura complexa.

Figura 1: Logomarca do Oracle

View

Podemos definir uma view como uma consulta predefinida baseada em uma ou mais tabelas. As view´s podem receber consulta assim como uma tabela, para isso deve-se passar o nome da(s) view(s) na cláusula FROM dento da consulta. As view's oferecem algumas vantagens e estão divididas em 2 tipos, View Simples e Complexa.

Algumas Vantagens de Usar View:

- Para restringir acesso a dados;

View Simpes - Uma view simples recupera linhas de uma única tabela base, não contém funções grupo e pode aceitar operações DML(Linguagem de Manipulação de Dados).**View Complexa** - Uma View complexa recupera linhas de várias tabelas, contém funções de grupo e nem sempre permite operações DML.

Abaixo vemos a sintaxe Padrão para criação de uma view, o que está entre colchetes([]) é opcional.

Listagem 1: Sintaxe padrão de criação de view

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW Nome_Da_View
  [(alias[, alias]...)]
 AS subquery
[WITH CHECK OPTION [CONSTRAINT nome_Constraint]]
[WITH READ ONLY [CONSTRAINT nome_Constraint]];
```

- **OR REPLACE** – Significa que a view deverá ser alterada, caso ela já exista.
- **FORCE** – Força a criação da view mesmo que as tabelas de base não existam.
- **NOFORCE** – Não permite a criação da view se as tabelas de base não existirem,

esse já o padrão, ou seja ,se no FORCE não for declarado o NOFORCE é embutido de forma implícita.

- **NOME_DA_VIEW** – É nome da visão.
- **ALIAS** – É o apelido de uma expressão na subconsulta. Deve haver o mesmo número de apelidos do que expressções na subconsulta.
- **SUBQUERY** – É a subconsulta que recupera as linhas das tabelas de base. Se você estiver usando alias(apelidos), pode usá-los na lista após a instrução SELECT.
- **WITH CHECK OPTION** – Significa que somente as linhas que seriam recuperadas na subconsulta podem ser inseridas, atualizadas ou removidas. Se você não usar essa cláusula, as linhas não são verificadas.
- **NOME_CONSTRAINT** – É o nome que será atribuído à restrição WITH CHECK OPTION ou WITH READ ONLY.
- **WITH READ ONLY** – Significa que só podem ser lidas as linhas da tabela base.

Nota: para nossos exemplos vamos trabalhar sobre o esquema HR que vem como padrão na instalação do Oracle. Para criação de view's o usuário precisa ter o privilégio CREATE VIEW liberado.

CRIANDO VIEW SIMPLES

Listagem 2: Criando a View Emp_dep

```
CREATE VIEW emp_dep
 AS
  SELECT employee_id,Last_name,salary
 FROM
   employees;
```

A view emp_dep criada foi baseada na tabela EMPLOYEES, essa view não possui

qualquer restrição.

Listagem 3: Criando a View Emp_dep_const

```
CREATE VIEW emp_dep_const
 AS
  SELECT employee_id,Last_name,salary
 FROM
   employees
 WITH READ ONLY constraint apenas_leitura;
```

A view emp_dep_const criada foi baseada na tabela EMPLOYEES, porém ao contrário da view emp_dep essa view só aceita leitura, nenhuma outra operação como delete, insert e etc.

Listagem 4: Criando a View Emp_dep_option_Const

```
CREATE VIEW emp_dep_option_Const
 AS
  SELECT employee_id,Last_name,salary
 FROM
   employees
 WHERE department_id = 20
 WITH CHECK OPTION CONSTRAINT emp_20;
```

A view emp_dep_option_Const criada foi baseada na tabela EMPLOYEES, essa view possui a cláusula WITH CHECK OPTION, onde foi criada a constraint emp_20. Essa constraint se baseia no que está na cláusula WHERE, que nesse caso só irá recuperar as linhas onde o department_id seja igual a 20. Com a constraint emp_20 significa que as operações DML só poderão ser aplicadas onde o departamento for 20.

Listagem 5: Criando a View Emp_dep_apelido

```
CREATE VIEW emp_dep_apelido(Cod_Func,Nome_Func,Salario)
```

```
AS
 SELECT employee_id,Last_name,salary
FROM
  employees;
```

A view emp_dep_apelido criada foi baseada na tabela EMPLOYEES, e nesse caso foram usados apelidos para as colunas que irão compor a view, dessa forma na hora de aplicar operações DML sobre essa view, pode-se utilizar os apelidos usado em sua criação(Cod_Func, Nome_Func, Salario). O número de apelidos deve corresponder ao numero de colunas envolvidas na subconsulta.

Listagem 6: Criando a View Emp_dep_apelido_as

```
CREATE VIEW emp_dep_apelido_as
 AS
  SELECT
  employee_id as Cod_Func,
  Last_name as Nome,
  salary as Salario
 FROM
   employees;
```

A view emp_dep_apelido_as criada acima também foi baseada na tabela EMPLOYEES, e nesse caso foram usados apelidos para as colunas que irão compor a view, dessa forma na hora de aplicar operações DML sobre essa view pode-se utilizar os apelidos usados em sua criação(Cod_Func, Nome_Func, Salario). Nessa view os apelidos para as colunas foram passados direto na subconsulta, mas o resultado é mesmo da emp_dep_apelido.

Listagem 7: Criando a View V_Grupo

```
CREATE FORCE VIEW V_Grupo
 AS
  SELECT grupo_id,descricao
```

```
 FROM
   Grupo;
```

A view V_Grupo criada foi baseada na tabela GRUPO, porém a tabela grupo não existe no banco, então, para que fosse possível a criação dessa View sem uma tabela base foi usada a opção FORCE.

CRIANDO VIEW COMPLEXA

Listagem 8: Criando a View Emp_e_dep

```
CREATE VIEW emp_e_dep
 AS
  SELECT employee_id,Last_name,department_name
 FROM
   employees E,
   departments D
 WHERE
   D.DEPARTMENT_ID = E.EMPLOYEE_ID;
```

A view emp_e_dep criada foi baseada em duas tabelas, EMPLOYEES e DEPATMENTS. Nesse momento deixa de ser uma view simples e passa a ser uma view complexa, pois está sendo aplicada uma subconsulta sobre duas tabelas.

Listagem 9: Criando a View Dep_mediasal

```
CREATE VIEW dep_mediasal
 AS
SELECT
 department_id as Cod_Departamento,
 AVG(salary)  as Media_Salario
FROM     employees
GROUP BY department_id;
```

A view dep_mediasal acima foi baseada em uma tabela: EMPLOYEES. Nesse caso, mesmo sendo utilizada apenas uma tabela, foi usada a função de agrupamento AVG e os registros estão sendo agrupados através do Group By.

ALTERANDO UMA VIEW

Para alterar uma View você precisará usar OR REPLACE que é a opção para substituir uma view já existente.

Listagem 10: Alterando a View Emp_dep

```
CREATE OR REPLACE VIEW emp_dep
 AS
  SELECT employee_id,Last_name,email,salary
 FROM
   employees;
```

A view emp_dep está sendo alterada com o OR REPLACE, nesse caso está sendo inserida a coluna EMAIL que não estava na criação da View no primeiro momento. A emp_dep será substituída já com a nova coluna que foi incluída na subconsulta.

Listagem 11: Alterando a View Emp_e_dep

```
CREATE OR REPLACE VIEW emp_e_dep
 AS
  SELECT Last_name,department_name
 FROM
   employees E,
   departments D
 WHERE
   D.DEPARTMENT_ID = E.EMPLOYEE_ID;
```

A view emp_e_dep está sendo alterada com o OR REPLACE, nesse caso está sendo

excluída a coluna EMPLOYEE_ID que estava na criação da View no primeiro momento. A emp_e_dep será substituída com a coluna EMPLOYEE_ID já removida da subconsulta.

REMOVENDO UMA VIEW

Quando uma view não é mais necessária você pode remover do mesmo modo que são removidos outros objetos do Oracle.

Listagem 12: Removendo a View Emp_e_dep

```
Drop view emp_e_dep;
```

Nota: um dica pra criação de uma view é criar e testar antes a subconsulta e após isso aplicar a sintaxe de criação de view.

Finalizamos por aqui esse artigo, até o próximo.

Abraço.

Eliézio Mesquita

Formado em Sistema de Informação, pela faculdade Estácio. Atualmente trabalha na empresa Office Sistemas(www.officesistemas.com.br) como DBA Oracle e como Instrutor de Oracle na Ka Solution(www.kasolution.com.br). elieziomesquita [...]