Administração de Banco de Dados

Aula 9

Prof. Marcos Alexandruk

Aula 9

Visões (Views)

- Visões regulares
- Visões materializadas
- Visões de objeto

Visões

- Uma visão (view) é uma representação lógica de uma ou mais tabelas.
- Uma visão deriva seus dados de tabelas denominadas tabelas base. (As tabelas base, na prática, podem ser tabelas ou outras visões.)
- As consultas em visões são realizadas da mesma maneira que as consultas em tabelas: basta incluir o nome da visão na cláusula WHERE.
- É possível realizar (com algumas restrições) operações DML (INSERT, UPDATE E DELETE) nas tabelas base através das visões
- Todas as operações executadas em uma visão afetam as tabelas base.
- As visões são também conhecidas como consultas armazenadas.
- O Oracle dispõe de vários tipos de visões. Exemplos:
  - Visões regulares
  - Visões materializadas

Visões

Visões regulares

- Uma visão regular armazena apenas sua definição ou consulta no dicionário de dados, não alocando, portanto, espaço em um segmento para armazenamento dos dados.
- Visões regulares podem seu utilizadas para ocultar a complexidade de determinadas consultas ao banco de dados ou para impor segurança.

Visões

Visões regulares

- **Exemplo: Visão FUNCIONARIO_VIEW criada a partir da tabela FUNCIONARIO.**

FUNCIONARIO			
MATRICULA	**NOME**	**DEPARTAMENTO**	**SALARIO**
1001	ANTONIO ALVES	ENGENHARIA	5300
1002	BEATRIZ BERNARDES	MARKETING	4800
1003	CLAUDIO CARCALHO	JURIDICO	5100

```
CREATE VIEW FUNCIONARIO_VIEW AS
SELECT MATRICULA, NOME, DEPARTAMENTO
FROM FUNCIONARIO;
```

FUNCIONARIO_VIEW		
MATRICULA	**NOME**	**DEPARTAMENTO**
1001	ANTONIO ALVES	ENGENHARIA
1002	BEATRIZ BERNARDES	MARKETING
1003	CLAUDIO CARCALHO	JURIDICO

Visões ajudam a melhorar a segurança no acesso aos dados.

Pode-se conceder acesso somente às visões e impedir a realização de consultas diretamente nas tabelas.

Visões regulares: READ ONLY

- Pode-se também criar uma visão com restrição **READ ONLY** (apenas leitura).

FUNCIONARIO			
MATRICULA	**NOME**	**DEPARTAMENTO**	**SALARIO**
1001	ANTONIO ALVES	ENGENHARIA	5300
1002	BEATRIZ BERNARDES	MARKETING	4800
1003	CLAUDIO CARCALHO	JURIDICO	5100

```
CREATE VIEW FUNCIONARIO_VIEW AS
SELECT MATRICULA, NOME, DEPARTAMENTO
FROM FUNCIONARIO
WITH READ ONLY CONSTRAINT FUNC_VIEW_READ_ONLY;
```

FUNCIONARIO_VIEW		
MATRICULA	**NOME**	**DEPARTAMENTO**
1001	ANTONIO ALVES	ENGENHARIA
1002	BEATRIZ BERNARDES	MARKETING
1003	CLAUDIO CARCALHO	JURIDICO

Visões regulares: UTILIZANDO APELIDOS PARA COLUNAS

FUNCIONARIO			
MATRICULA	NOME	DEPARTAMENTO	SALARIO
1001	ANTONIO ALVES	ENGENHARIA	5300
1002	BEATRIZ BERNARDES	MARKETING	4800
1003	CLAUDIO CARCALHO	JURIDICO	5100

```
CREATE VIEW FUNCIONARIO_VIEW (MAT_FUNC, NOME_FUNC, DEPT_FUNC) AS
SELECT MATRICULA, NOME, DEPARTAMENTO
FROM FUNCIONARIO;
```

ou:

```
CREATE VIEW FUNCIONARIO_VIEW AS
SELECT
MATRICULA AS MAT_FUNC,
NOME AS NOME_FUNC,
DEPARTAMENTO AS DEPT_FUNC
FROM FUNCIONARIO;
```

Visões

- **Visões facilitam a realização de consultas complexas baseadas em duas ou mais tabelas.**

```
--------------------    -----------------
TABELA: CLIENTE         TABELA: PEDIDO
--------------------    -----------------
CODCLI   NOME      UF   NR  VALOR  CODCLI
--------------------    -----------------
1001     FULANO    SP   1   4800   1002
1002     BELTRANO  RJ   2   3600   1003
1003     CICRANO   SP   3   5500   1001
```

```
SELECT C.UF, SUM(P.VALOR)
FROM CLIENTE C
INNER JOIN PEDIDO P
ON C.CODCLI = P.CODCLI
GROUP BY C.UF;
```

```
-------------------------------
VISAO: CLIENTE_PEDIDO_VIEW
-------------------------------
CODCLI   NOME      UF   NR  VALOR
-------------------------------
1001     FULANO    SP   1   4800
1002     BELTRANO  RJ   2   3600
1003     CICRANO   SP   3   5500
```

```
SELECT UF, SUM(VALOR)
FROM CLIENTE_PEDIDO_VIEW
GROUP BY C.UF;
```

Visões

- **Inserir ou atualizar dados em visões criadas a partir de duas ou mais tabelas base:**

```
--------------------    -----------------
TABELA: CLIENTE         TABELA: PEDIDO
--------------------    -----------------
CODCLI   NOME      UF   NR  VALOR  CODCLI
--------------------    -----------------
1001     FULANO    SP   1   4800   1002
1002     BELTRANO  RJ   2   3600   1003
1003     CICRANO   SP   3   5500   1001
```

```
-------------------------------
VISAO: CLIENTE_PEDIDO_VIEW
-------------------------------
CODCLI   NOME      UF   NR  VALOR
-------------------------------
1001     FULANO    SP   1   4800
1002     BELTRANO  RJ   2   3600
1003     CICRANO   SP   3   5500
```

```
CREATE VIEW CLIENTE_PEDIDO_VIEW AS
SELECT C.CODCLI, C.NOME, C.UF,
P.NR, P.VALOR
FROM CLIENTE C
INNER JOIN PEDIDO P
ON C.CODCLI = P.CODCLI;
```

Visões

- **Criar um trigger instead of para inserir dados nas tabelas CLIENTE e PEDIDO através da visão CLIENTE_PEDIDO_VIEW:**

```
CREATE OR REPLACE TRIGGER CLI_PED_INSERT
INSTEAD OF INSERT ON CLIENTE_PEDIDO_VIEW
REFERENCING NEW AS N
FOR EACH ROW
DECLARE
   rowcnt number;
BEGIN
   SELECT COUNT(*) INTO rowcnt FROM CLIENTE WHERE CODCLI = :N.CODCLI;
    IF rowcnt = 0  THEN
       INSERT INTO CLIENTE (CODCLI,NOME,UF) VALUES (:N.CODCLI, :N.NOME, :N.UF);
   ELSE
      UPDATE CLIENTE SET CLIENTE.NOME = :N.NOME, CLIENTE.UF = :N.UF
         WHERE CLIENTE.CODCLI = :N.CODCLI;
    END IF;
   SELECT COUNT(*) INTO rowcnt FROM PEDIDO WHERE NR = :N.NR;
   IF rowcnt = 0  THEN
       INSERT INTO PEDIDO (NR,VALOR,CODCLI) VALUES (:N.NR, :N.VALOR, :N.CODCLI);
    ELSE
      UPDATE PEDIDO SET PEDIDO.VALOR = :N.VALOR, PEDIDO.CODCLI = :N.CODCLI
         WHERE PEDIDO.NR = :N.NR;
   END IF;
END;
/
```

Visões

Materialized Views (Visões Materializadas)

- Uma visão materializada armazena sua definição ou consulta no dicionário de dados, porém, diferentemente de das visões regulares, aloca espaço em um segmento para armazenamento dos dados. Este tipo de visão pode, por exemplo, replicar uma cópia somente leitura da tabela base para outro banco de dados.

- Visões materializadas utilizam um log de visão materializada associado às tabelas base para realizar atualizações incrementais. Caso não se utilize este recurso, será preciso realizar uma atualização completa quando for necessária a atualização dos dados na visão materializada.

Materialized Views - Exemplos:

Criação de uma tabela denominada ALUNO:

```
---------------
TABELA: ALUNO
---------------
RA       NOME
---------------
1001    ANONIO
1002    BEATRIZ
1003    CLAUDIO
```

Criar um MATERIALIZED VIEW LOG (log de visão materializada):

```
CREATE MATERIALIZED VIEW LOG ON ALUNO;
```

Os logs de visões materializadas são usados para sincronização entre a tabela base (master table) e a visão.
Antes da criação da visão materializada, a master table deve ser associada a um materialized view log.

Materialized Views - Exemplo 1:

CRIAÇÃO DE UMA VISÃO MATERIALIZADA QUE SERÁ ATUALIZADA **MANUALMENTE**:

```
CREATE MATERIALIZED VIEW ALUNO_VIEW_1
-- popular a view no momento de sua criação
BUILD IMMEDIATE
-- selecionar os dados da tabela base
AS SELECT * FROM ALUNO;
```

CONSULTAR A VIEW ALUNO_VIEW_1:

```
SELECT * FROM ALUNO_VIEW_1;
```

INSERIR UMA NOVA LINHA NA TABELA ALUNO:

```
INSERT INTO ALUNO (RA, NOME) VALUES (4,'DANIELA');
COMMIT;
```

UTILIZAR A PROCEDURES DBMS_MVIEW.REFRESH PARA ATUALIZAR A VIEW:

```
CALL DBMS_MVIEW.REFRESH ('ALUNO_VIEW_1','F');
-- F: FAST (INCREMENTAL) | C: COMPLETE (COMPLETA) | ?: FORCE (INCREMENTAL
-- SE POSSÍVEL, CASO NÃO SEJA, REALIZA UMA ATUALIZAÇÃO COMPLETA)
```

CONSULTAR A VIEW ALUNO_VIEW_1:

```
SELECT * FROM ALUNO_VIEW_1;
```

Materialized Views - Exemplo 2:

CRIAÇÃO DE UMA VISÃO MATERIALIZADA QUE SERÁ ATUALIZADA **AUTOMATICAMENTE**:

```
CREATE MATERIALIZED VIEW ALUNO_VIEW_2
-- popular a view no momento de sua criação
BUILD IMMEDIATE
-- atualizar de forma incremental a cada 1 segundo
REFRESH FORCE START WITH SYSDATE NEXT SYSDATE + 1/86400
-- selecionar os dados da tabela base
AS SELECT * FROM ALUNO;
```

CONSULTAR A VIEW ALUNO_VIEW_2:

```
SELECT * FROM ALUNO_VIEW_1;
```

INSERIR UMA NOVA LINHA NA TABELA ALUNO:

```
INSERT INTO ALUNO (RA, NOME) VALUES (5,'ERNESTO');
COMMIT;
```

CONSULTAR A VIEW ALUNO_VIEW_2:

```
SELECT * FROM ALUNO_VIEW_2;
```

Visões

Visões de objeto

- Uma visão de objeto é uma tabela virtual de objeto. Cada linha na visão é um objeto, que é uma instância de um tipo de objeto. Um tipo de objeto é um tipo de dados definido pelo usuário.
- As visões de objeto permitem que as aplicações orientadas a objetos vejam os dados como uma coleção de objetos que têm atributos e métodos, facilitando a migração de um ambiente puramente relacional para um ambiente orientado a objetos.
- É possível utilizar as visões de objeto para recuperar, atualizar, inserir e excluir dados relacionais como se estes fossem armazenados como um tipo de objeto. Pode-se também definir visões com colunas que são tipos de dados de objetos, tais como objetos e coleções (tabelas aninhadas e varrays).

Referências Bibliográficas

- https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_8004.htm
- https://docs.oracle.com/cd/B10501_01/server.920/a96567/repmview.htm
- https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_6002.htm
- http://aprendaplsql.com/sql/diferenca-entre-views-e-materialized-views-oracle/
- https://docs.oracle.com/cd/A58617_01/server.804/a58227/ch_objvw.htm