Banco de Dados II

Views

Profa.: Márcia Sampaio Lima

EST - UEA

View

- Conceito:
  - É um objeto que pertence a um BD.
  - Definida baseada em declarações SELECT.
  - É um resultado originado de uma consulta pré-definida.
  - Retorna uma visualização de dados oriundos de uma ou mais tabelas.

View

- Conceito:
  - Na essência é um ***metadado*** que mapeia uma ***query para outra***, por isto pode ser considerado como uma tabela virtual.
  - Representa uma **visão** de dados e não contém dados.
  - Com ela temos a ilusão de estar vendo uma tabela que não existe.

View

- Também são chamadas de *named queries* ou *stored queries*.
- Cuidado:
  - *Query* (consulta) X resultado
  - View é a *query* mesmo.

"É basicamente um texto de um SELECT. Pense que ela é uma *query* que você armazena em uma variável e ao invés de escrever toda a *query* de novo sempre que for necessária usa a variável"

View

- “Virtual Tables":
  - Como tambem podem ser chamadas das views
- “Based Tables"
  - Tabelas usadas na formação das views.
- Podem ser formadas a partir de tabelas ou de outras views.

View

- Criando uma View:

```
CREATE VIEW nome_da_view AS

SELECT * FROM nome_tabela;
```

View

- **Verificando se a view foi criada:**

```
SHOW TABLES;
```

View

- Funcionamento:
  - São geradas sempre que forem necessárias.
  - Sua carga ocorrerá conforme a necessidade e otimizações do BD.
  - Existe a ilusão de que uma atualização da tabela real envolvida reflita na *view*, afinal a *view* é gerada no momento que ela é necessária.

View

- Funcionamento:
  - São tabelas virtuais dinâmicas, que só existem no momento em que precisa delas.
  - Somente a *query* para sua geração é pré-definida.
  - A "tabela virtual" que é criada, não é uma tabela física criada em memória com todos os dados.
  - A *view* é apenas uma forma de traduzir uma *query* por outra *query*.

View

- Exemplo:

```
CREATE VIEW vw_alunos AS

SELECT nome, edereco FROM Alunos;
```

View

- **Alterando uma view:**

```
ALTER VIEW nome_da_view AS

SELECT * FROM nome_outra_tabela;
```

View

- **Excluindo uma view:**

```
DROP VIEW nome_da_view;
```

View

- Uma *View* é uma transformação em cima de uma ou mais tabelas que se comporta como uma tabela (tabela virtual).

- Normalmente é somente-leitura (não é possível nem inserir, nem alterar, nem excluir linhas dela).

View

- Exemplo:

```
CREATE VIEW MinhaView AS

SELECT COLUNA2, COLUNA4, COLUNA6

FROM TABELA WHERE COLUNA1 = 1;
```

- Para acionar a view:
  - SELECT * FROM MinhaView;

View

- Exemplo:

```
CREATE VIEW MinhaView2 AS

SELECT CONCAT(NOME, ' ', SOBRENOME) AS
NOME_COMPLETO, COLUNA2, COLUNA3, COLUNA4

FROM TABELA_COM_NOME;
```

View

- **Vantagens:**
  - **Economizar tempo com retrabalho;**
  - **Velocidade de acesso às informações;**
    Uma vez compilada, o seu recordset (conjunto de dados) é armazenado em uma tabela temporária (virtual).
  - **Mascarar complexidade do banco de dados;**

View

- **Vantagens**
  - **Simplifica o gerenciamento de permissão de usuários;**

    Ex.: Em vez de conceder permissão para que os usuários contem tabelas base, os proprietários de bancos de dados podem conceder permissões para que os usuários consultem dados somente através de views. Isso também protege as alterações na estrutura das tabelas

View

- **Vantagens**
  - **Organizar dados a serem exportados para outros aplicativos;**

    Ex.: Você pode criar uma view baseada em uma consulta complexa, que associe até 32 tabelas e depois exportar dados para outro aplicativo para análise adicional.

View

- **Vantagens**
  - Pode ser usada este resultado em outras consultas diminuindo a complexidade.
  - As Views pré-definidas ficam armazenadas e não precisa lembrar de como criá-las. Não confundir com resultados.
  - Regras de negócio podem ser adotadas nas *views*. A maneira como você vai acessar os dados está pré-definida. Isto é útil para formatar dados, ajudar ferramentas externas e facilitar o acesso via APIs.

View

- Exemplo

```
CREATE VIEW dbo.SeattleOnly  AS
SELECT p.LastName, p.FirstName, e.JobTitle, a.City,
sp.StateProvinceCode
FROM HumanResources.Employee e INNER JOIN Person.Person p
          ON p.BusinessEntityID = e.BusinessEntityID
         INNER JOIN Person.BusinessEntityAddress bea
              ON bea.BusinessEntityID = e.BusinessEntityID
                INNER JOIN Person.Address a
                ON a.AddressID = bea.AddressID
                   INNER JOIN Person.StateProvince sp
                    ON sp.StateProvinceID = a.StateProvinceID
WHERE a.City = 'Seattle'
```

Views

- **Desvantagens**
  - Esconde uma complexidade da *query* podendo enganar o desenvolvedor quanto à performance necessária para acessar determinada informação. E pode ser pior quando *views* usam outras *views*. Em alguns casos você pode estar fazendo consultas desnecessárias sem saber de forma muito intensiva.

Views

- **Desvantagens**
  - Cria uma camada extra. Mais objetos para administrar. Algumas pessoas consideram isto um aumento de complexidade.
  - Pode limitar exageradamente o que o usuário pode acessar impedindo certas tarefas.

Views

- **Desvantagens**
  - Se a *view* for materializada fará com que alterações nas tabelas reais envolvidas sejam mais lentas afinal são mais tabelas para atualizar. Este tipo de *view* funciona de forma semelhante a um Trigger.

View

- **O QUE É UMA *MATERIALIZED VIEW***
  - **Visão Materializada** é uma view, só que neste caso, o que é armazenado não é a consulta e sim o resultado dela.
  - Isso nos leva a uma pergunta:
    - Quando devo criar uma VIEW ou uma MATERIALIZED VIEW?

View

- **O QUE É UMA *MATERIALIZED VIEW***
  - Uma MATERIALIZED VIEW é uma tabela real no banco de dados que é atualizada SEMPRE que ocorrer uma atualização em alguma tabela usada pela sua consulta.
  - Por este motivo, no momento em que o usuário faz uma consulta nesta visão materializada o resultado será mais rápido que se ela não fosse materializada.

View

- **O QUE É UMA *MATERIALIZED VIEW***
  - Diferença:
    - A *view* realiza a consulta no momento que o usuário faz uma consulta
    - A *materialized view* realiza a consulta no momento em que uma das tabelas consultadas é atualizada.

View

- ***MATERIALIZED VIEW***

  - Embora a consulta na *view* fique mais rápida com o pre-processamento da consulta interna, o processo de escrita no banco de dados fica mais lento, pois é necessário executar a consulta interna da *materialized view* toda vez que um dado sofrer alteração.

View

- Quando usar:
  - Usar uma *visão materializada* quando **o desempenho das buscas na view é mais importante que o desempenho da escrita nas tabelas que ela utiliza**.
  - Mas se uma tabela utilizada pela *view* tem muita alteração de dados, talvez seja mais interessante que a *view* não seja materializada

View

- Exemplo prático

```
CREATE VIEW vw_nf  As

SELECT i.notaFiscal, i.livro, l.nome

FROM itenscompra i INNER JOIN livros l

ON i.livro = l.idLivro
```

- Executando:
  - Select * from vw_NF;