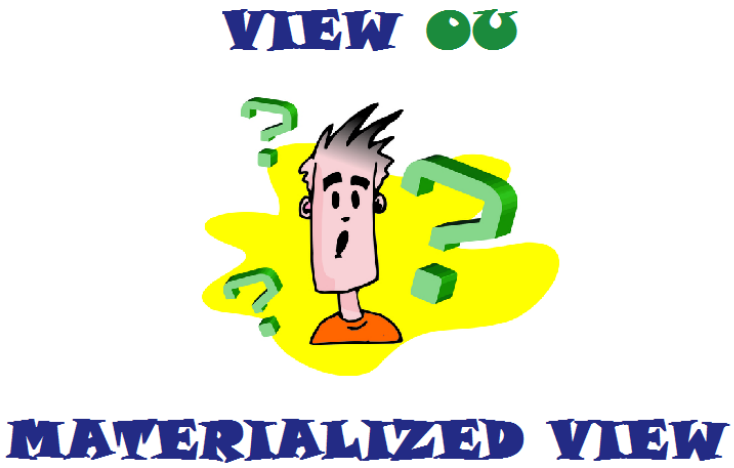
Qual a diferença entre View e Materialized View?

20 de agosto de 2015 Gustavo Furtado de Oliveira Alves Banco de dados 9 Comentários

Quando se trabalha com banco de dados podemos dizer que uma das tarefas mais importantes a ser realizada é a otimização do desempenho do banco de dados.

Dependendo da aplicação, uma determinada tabela deve ser otimizada para receber muitas consultas e poucas atualizações. Algumas vezes é o contrário, muitas atualizações na tabela, mas poucas consultas.

Neste post você vai entender um conceito que deve ser claro para você quando for decidir se pretende priorizar a escrita ou a leitura de uma tabela ou "visão". Você vai entender a diferença entre uma *VIEW* e uma *MATERIALIZED VIEW*.



Antes de falar sobre as diferenças entre esses dois tipos de objetos de banco de dados, é importante definir o que é uma *VIEW* ou *visão*.

O QUE É UMA *VIEW*

Uma *VIEW* (ou Visão) é uma consulta armazenada no banco de dados. Nós podemos, realizar consultas sobre uma *VIEW* como se fosse uma tabela. Muitas pessoas se referem às *VIEWs* como uma *tabela virtual*.

Uma das principais funções da VIEW é controlar a segurança do banco de dados. Geralmente se cria a VIEW (uma consulta armazenada no banco de dados) com os campos que determinado perfil de usuário pode acessar, e concede-se ao usuário acesso apenas a essa VIEW e não à(s) tabela(s) diretamente.

Artigos Recentes

Como instalar o MySQL no Windows (Passo a passo!)

Gustavo Furtado de Oliveira Alves

10 Comentários

Também utiliza-se VIEWs para apresentar informações mais organizadas para o usuário sem que ele precise elaborar uma consulta complexa. Esta já estaria pronta e armazenada no próprio banco de dados para uso.

Para entender o que é uma VIEW na prática, imagine as duas tabelas abaixo.

FUNCIONARIO

CPF	Nome	E-mail	Salário	ID_DEPARTAMENTO
123.456.789-10	João	joao@exemplo.com	R$ 4.500,00	1
111.222.333-44	Gustavo	gustavo@exemplo.com	R$ 2.500,00	1
222.333.444-55	Pedro	pedro@exemplo.com	R$ 3.500,00	2

DEPARTAMENTO

ID_DEPARTAMENTO	NOME
1	Financeiro
2	Recursos Humanos

Agora considere que um determinado usuário precisa de uma lista atualizada de Funcionários e seus respectivos departamentos. Por questões de segurança, não pode ser fornecido à este usuário informações de CPF, E-mail e Salário dos funcionários.

A melhor forma para se fazer isso é criar uma VIEW onde essas informações não apareçam e fornecer ao usuário acesso apenas a esta VIEW. Ou seja, o usuário só teria acesso visão conforme a imagem abaixo.

V_FUNCIONARIO_DEPARTAMENTO

FUNCIONARIO	DEPARTAMENTO
Gustavo	Financeiro
João	Financeiro
Pedro	Recursos Humanos

{.aligncenter .size-full .wp-image-2244 width="298" height="106"}

A consulta desta view poderia ser a seguinte:

Como instalar o GIT no Windows (Passo a passo!)

Gustavo Furtado de Oliveira Alves

5 Comentários

Como validar um CPF em JavaScript

Gustavo Furtado de Oliveira Alves

1 Comentário

O mínimo que você precisa saber sobre JSON para ser um bom programador!

Gustavo Furtado de Oliveira Alves

4 Comentários

Java, Python ou Javascript?

Gustavo Furtado de Oliveira Alves

6 Comentários

Exercício: Algorítmo Par ou Ímpar

Gustavo Furtado de Oliveira Alves

4 Comentários

Categorias

Banco de Dados (10)

{ Dicas de Programação } (12)

SELECT F.NOME AS FUNCIONARIO, D.NOME AS DEPARTAMENTO from FUNCIONARIO F INNER JOIN DEPARTAMENTO D ON D.ID_DEPARTAMENTO = F.ID_DEPARTAMENTO

Mas na realidade, a VIEW realiza uma consulta (query) em tempo de execução. Em uma VIEW simples essa consulta que é armazenada. Essa consulta pode ter condições próprias para restringir os dados que serão visualizados pelo usuário, tanto horizontal (colunas que serão apresentadas) quanto vertical (linhas que serão apresentadas).

Por exemplo, se no mesmo caso acima, além de não apresentar cpf, e-mail e salário, quiséssemos também retirar os funcionários do departamento "Recursos Humanos" da "visão" do usuário, bastaria colocar uma condição na cláusula "WHERE" da view (*where id_departamento <> 2*). O usuário nem saberia disso.

Acho que já deu pra entender o que é uma VIEW: **Uma consulta armazenada, uma tabela "virtual".**

Qualquer dúvida deixe um comentário no final do post.

Agora vamos ver ...

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 implica algumas coisas muito importantes que devem ser entendidas quando for decidir entre criar uma VIEW ou uma MATERIALIZED VIEW.

Primeiro, 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.

Dicionário de programador (6)

Iniciante em programação (28)

Tags

Basicamente a diferença no uso das duas é essa. A *view* realiza a consulta no momento que o usuário faz uma consulta nela e a *materialized view* realiza a consulta no momento em que uma das tabelas consultadas é atualizada.

Vejamos como seria na prática com o mesmo exemplo que utilizamos acima.

Se a *view* V_FUNCIONARIO_DEPARTAMENTO for materializada, sempre que a tabela departamento ou a tabela funcionário receber uma inclusão, alteração ou exclusão, a "consulta da view" também será executada e o resultado será armazenado.

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.

QUANDO USAR *VIEW* OU *MATERIALIZED VIEW*?

A decisão se a sua *view* deve ser simples ou materializada é tomada com base no tipo de utilização das tabelas usadas pela consulta da *view*. A decisão é simples. Você consulta mais na view do que altera os dados das tabelas? Os dados do seu banco de dados são alterados com frequência?

Em resumo, você deve 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.

Estamos entendidos? Quando for criar a próxima view você já sabe se ela deve ser materializada ou não.

Deixe um comentário se tiver dúvida.

Sobre Gustavo Furtado de Oliveira Alves