Views

Views

- View é um resultado originado de uma consulta pré-definida.
- Essencialmente é um metadado que mapeia uma query para outra
- Pode ser considerada como uma tabela virtual
- Representa uma visão dos dados e não contém dados
- Também são chamadas de *named queries* ou *stored queries*

Views

- Dois objetivos principais das visões:
  - Simplificar consultas
  - Autorização de acesso (segurança)
- Visão: é um meio de prover ao usuário um “modelo personalizado” do banco de dados
- É uma relação que não armazena dados, composta dinamicamente por uma consulta que é previamente analisada e otimizada.

Views

- Um SGBD pode dar suporte a um grande número de visões sobre qualquer conjunto de relações
-
- O SGBD armazena a definição da visão, mas ela é instanciada quando uma consulta sobre ela for executada
  - Toda visão pode ser consultada mas nem toda visão pode ser atualizada
- Em SQL uma visão é definida como:

Create view nomeDaVisao <expressão de consulta>,

Onde <expressao de consulta> é qualquer expressão de consulta válida em SQL

Views

- Exemplo:

Projeto (codProj, tipo, descricao)
ProjetoEmpregado (codProj, codEmp, dataInicial, dataFinal)
Empregado (codEmp, nome, categoria, salario)

```
Create view vAltoEscalao as
  select codEmp, nome, salario
          from empregado
  where salario>10000
```

- Esta visão terá os atributos especificados na consulta

Views

- Exemplo:

```
Create view vAltoEscalao (a, b, c) as
  select codEmp, nome, salario
        from empregado
  where salario>10000
```

- Esta visão terá os atributos a, b, c, que serão instanciados com os respectivos valores recuperados pela consulta (codEmp, nome, salario)

Views

- Views com recursividade (Views sobre Views):

Projeto (codProj, tipo, descricao)
ProjetoEmpregado (codProj, codEmp, dataInicial, dataFinal)
Empregado (codEmp, nome, categoria, salario)
vAltoEscalao (codEmp, nome, salario)

```
Create view vProjetosAltoEscalao as
  select e.codEmp, a.nome, a.salario, p.descricao
          from vAltoEscalao a, Projeto p, ProjetoEmpregado pe, Empregado e
  where a.codEmp=pe.codEmp AND
       pe.codProj=p.codProj
```

Views

- Consultas SQL podem ser especificadas sobre a visão

```
select nome
        from vProjetosAltoEscalao
where descricao = “Projeto A”
```

- Uma visão está sempre atualizada:
    - Ao modificar tuplas nas tabelas envolvidas na visão, a visão vai automaticamente refletir as alterações
- A visão não é realizada quando é criada mas quando executamos uma consulta sobre ela

Views

- Exemplo

```
CREATE VIEW vSomenteBH
AS
SELECT p.Sobrenome, p.Nome, e.Cargo, end.Cidade,
uf.CodigoEstado
FROM RecursosHumanos.Empregado e
INNER JOIN Pessoa.Pesssoa p
ON p.EntidadeID = e.EntidadeID
   INNER JOIN Pessoa.EnderecoEntidade ent
   ON ende.EntidadeID = e.EntidadeID
   INNER JOIN Pessoa.Endereco end
   ON end.EnderecoID = ent.EnderecoID
   INNER JOIN Pessoa.UnidadeFederacao uf
   ON uf.UnidadeFederacaoID = end.UnidadeFederacaoID
WHERE end.Cidade = 'Belo Horizonte'
```

```
SELECT p.Sobrenome, p.Nome, e.Cargo, end.Cidade,
uf.CodigoEstado
FROM RecursosHumanos.Empregado e
INNER JOIN Pessoa.Pesssoa p
ON p.EntidadeID = e.EntidadeID
   INNER JOIN Pessoa.EnderecoEntidade ent
   ON ende.EntidadeID = e.EntidadeID
   INNER JOIN Pessoa.Endereco end
   ON end.EnderecoID = ent.EnderecoID
   INNER JOIN Pessoa.UnidadeFederacao uf
   ON uf.UnidadeFederacaoID = end.UnidadeFederacaoID
WHERE p.Sobrenome = 'Silva' AND end.Cidade = 'Belo Horizonte'
```

A consulta ficaria:

```
SELECT * FROM vSomenteBH WHERE Sobrenome = ´Silva´;
```

Views

- Exemplo:

  - Quando uma visão não é mais necessária podemos eliminá-la, usando o comando drop view

```
DROP VIEW nomeDaVisao
```

Views

- Vantagens das Views
  - Você pode usar o resultado de uma View em outras consultas diminuindo a complexidade
  - As consultas pré-definidas ficam armazenadas e você não precisa lembrar de como criá-las
  - Torna mais fácil entender a modelagem da aplicação, criando uma visão mais lógica
  - Facilita a troca do modelo físico sem o perigo de quebrar queries existentes
  - Permite otimizações por já ter conhecimento das queries utilizadas nas views

Views

- Vantagens das Views
  - Views podem ter permissões – é possível proibir acesso à tabelas de acordo com a necessidade de controle, sobre o que e como o usuário pode acessar a informação
  - É possível adotar regras de negócio 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
  - Facilita o acesso de dados em bases legadas, permitindo melhor migração
  - Se a view for materializada pode ter ganho de desempenho para o acesso aos dados já recuperados

Views

- Desvantagens
  - Esconde a complexidade da query, ocultando o desempenho necessário para acessar determinada informação
  - Quando views usam outras views, pode-se estar fazendo consultas desnecessárias sem saber
  - Cria uma camada extra com mais objetos para administrar, aumentando a complexidade
  - Pode limitar exageradamente o que o usuário pode acessar impedindo certas tarefas.
  - 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 gatilho.

Views

- Visão Materializada

  - Uma visão materializada é um objeto de banco de dados que contém os resultados de uma consulta
  - O processo de criação de uma visão materializada é às vezes chamado de materialização .
  - Esta é uma forma de armazenar em cache os resultados de uma consulta
  - Os usuários de banco de dados geralmente usam visões materializadas por razões de desempenho, ou seja, como uma forma de otimização

Views

- Visão Materializada
  - Sempre que uma consulta ou uma atualização aborda a tabela virtual de uma visualização comum, o SGBD as converte em consultas ou atualizações nas tabelas base subjacentes.
  - Uma visão materializada adota uma abordagem diferente: o resultado da consulta é armazenado em cache como uma tabela concreta ("materializada")
  - Essa visão pode ser atualizada das tabelas base originais de tempos em tempos, permitindo um acesso muito mais eficiente, ao custo de armazenamento extra e de alguns dados potencialmente desatualizados.
  - Visões materializadas encontram uso especialmente em cenários de data warehousing , onde consultas frequentes das tabelas base reais podem ser caras.

Views

- Visão Materializada
  - Exemplo (Oracle):

```
CREATE MATERIALIZED VIEW MinhaView
REFRESH FAST START WITH SYSDATE
  NEXT SYSDATE + 1
    AS SELECT * FROM <nomeTabela>;
```

Autorização de Acesso

Objetivo

- Proteção contra acessos mal intencionados
- Controlar quais dados um usuário ou grupo de usuários pode ter acesso
- Controlar quais operações que um usuário ou grupo de usuários pode realizar sobre estes dados

Autorização de Acesso: Exemplos

- Exemplo Em um sistema bancário, um funcionário precisa saber os dados dos clientes, mas apenas os que tem conta poupança na agência **1899-6**

Cliente (#**codCli**, nome, endereco, codCidade)
Agencia (#**codAgencia**, nome, descricao)
Poupança (# **numConta**, # **codAgencia**, # **codCli, saldo**)

- Exemplo 2: um funcionário de uma empresa deve ter acesso aos nomes dos funcionários e aos seus projetos, mas não de ter acesso ao salário dos funcionários

Empregado (#**codEmp**, nome, categoria, salario)
Projeto (#**codProj**, tipo, descricao)
ProjetoEmpregado (# **codProj**, # **codEmp**, dataInicial, dataFinal)

Autorização de Acesso

- Um usuário do Banco de dados pode ter diversas formas de autorização a partes do BD:

- O DBA é o superusuário do BD, que pode tudo
  - Alguns privilégios são exclusivos dele, como *a recuperação* do BD, a configuração de parâmetros do SGBD, etc.
  - Concede/retira (revoga) privilégios de acesso aos outros usuários

Autorização de Acesso

Duas formas principais de acesso:

- ***Nível de conta* (usuário):** o administrador do BD pode dar permissões aos usuários para criar esquemas, modificar e criar tabelas e selecionar dados
  - O DBA estabelece permissões da conta, independente das relações do BD
- ***Nível de relação/visão*:** o DBA pode controlar o privilégio de acesso de cada usuário a relações ou visões específicas do BD
  - Definidas para SQL
  - Para cada usuário podem ser dadas permissões de leitura(seleção), modificação e referência

Autorização de Acesso

- **Autorização de leitura:** permite apenas recuperar dados de uma relação
  - ***Permissão select***
- **Autorização de modificação:** privilégios para ***insert, delete*** e ***update***
  - ***Para insert e update é possível restringir os atributos***
- **Autorização de referência:** uma conta (de usuário) pode fazer referência a uma relação ao especificar restrições de integridade
  - Pode ser por atributo

Autorização de Acesso em SQL

- Lista basica: alter, delete, index, insert, select e update

- Sintaxe:

grant <lista de privilégios> **on** <nome da relação ou visão> **to** <lista de usuários>

- Exemplo:

  grant select **on** cliente **to** U1, U2, U3

  grant update (saldo) **on** deposito **to** U1, U2

Autorização de Acesso: Exemplos

- Exemplo 1: Em um sistema bancário, um funcionário precisa saber os dados dos clientes, mas apenas os que tem conta poupança na agência **1899-6**

Cliente (**codCli**, nome, endereco, codCidade)
Agencia (**codAgencia**, nome, descricao)
Poupança (**numConta**, # **codAgencia**, # **codCli, saldo**)

- **Solução: usar views**

```
Create view vClientePoupancaAgencia as
  select c.codCli, c.nome, p.numConta
     from cliente c, poupanca p
  where c.codCli=p.cod_cli and
                          p.codAgencia="1899-6"
```

- **grant** select **on** vClientePoupancaAgencia **to** U10

Autorização de Acesso: Exemplos

- Exemplo 2: um funcionário de uma empresa deve ter acesso aos nomes dos funcionários e aos seus projetos, mas não de ter acesso ao salário dos funcionários

```
Create view vEmpregadoProjeto as
   select e.codEmp, e.nome, p.tipo, p.descricao, j.dataInicial, j.dataFinal
   from empregado e, projeto p, projetoEmpregado j
   where e.codEmp=j.codEmp and j.codProj=p.codProj
```

- **grant** select **on** vEmpregadoProjeto **to** U50

Empregado (**codEmp**, nome, categoria, salario)
Projeto (#**codProj**, tipo, descricao)
ProjetoEmpregado (# **codProj**, # **codEmp**, dataInicial, dataFinal)

Autorização de Acesso

- Roles (Papéis): são interessantes quando um grupo de usuários tem as mesmas restrições de acesso:

ex: vários caixas de um banco (várias agências) tem permissão para creditar e debitar valores na conta dos clientes. Ao invés de dar permissão de inclusão, alteração e exclusão para cada um dos caixas que tem este direito, cria-se o papel ***movimentacao***

Create role movimentacao

E dá-se a permissão ao papel

*Grant insert, update, delete **ON TABELA** to movimentacao*

Vincula todos os usuarios ao papel

GRANT movimentacao TO A, B, C

Exemplo Completo

- Suponha que o DBA crie 4 contas U1, U2, U3 e U4.

1) Somente U1 deve criar relações no banco (privilégio de conta)

```
grant createTab to U1
```

2) com essa autorização o usuário U1 pode criar relações e terá TODOS os privilégios sobre elas

- Suponha que U1 criou as relações

Empregado (**codEmp**, nome, categoria, salario)
Projeto (#**codProj**, tipo, descricao)
ProjetoEmpregado (# **codProj**, # **codEmp**, dataInicial, dataFinal)

Views

3) Suponha que U1 quer dar ao usuário U2 permissão para incluir e remover tuplas em **Empregado** e **Projeto**

```
grant INSERT, DELETE ON EMPREGADO,PROJETO TO U2
```

4) Suponha que U1 quer dar ao usuário U3 permissão para recuperar tuplas em **Empregado** e seja capaz de PROPAGAR este privilégio

```
grant SELECT ON EMPREGADO TO U3 WITH GRANT OPTION
```

Empregado (**codEmp**, nome, categoria, salario)
Projeto (#**codProj**, tipo, descricao)

4) Agora U3 pode conceder privilégio de seleção para U4 sobre a relação Empregado

grant SELECT **ON** EMPREGADO **TO** U4

- Obs: U4 não pode propagar este privilégio

5) Suponha que U1 queira revogar a permissão de U3

REVOKE SELECT **ON** EMPEGADO **FROM** U3

Empregado (**codEmp**, nome, categoria, salario)
Projeto (#**codProj**, tipo, descricao)

Views

6)Suponha que U1 queira dar ao usuário U3 permissão apenas para consultar **empregados** que trabalhem no projeto PUC.

```
Create view vEmpregadoProjeto as
  select e.codEmp, e.nome, e.categoria, e.salario
     from empregado e, projeto p, projetoEmpregado j
  where e.codEmp=j.codEmp and j.codProj=p.codProj and
        p.descricao="PUC"
```

- **grant** SELECT **ON** vEMPREGADOProjeto **TO** U3

Empregado (**codEmp**, nome, categoria, salario)
Projeto (**codProj**, tipo, descricao)
ProjetoEmpregado (# **codProj**, # **codEmp**, dataInicial, dataFinal)

Views

grant all to {listaUsuários | public}

Exercícios

- Dado o esquema abaixo:

Médico (CRM, nome, idade, cidade, especialidade, #*númeroA*)
Paciente (RG, nome, idade, cidade, doença)
Funcionário (RG, nome, idade, cidade, salário)
Consulta (#*CRM*, #*RG*, data, hora)
Ambulatório (númeroA, andar, capacidade)

- Escreva uma visão que contenha o nome do medico, o nome do paciente e a data da consulta
- Escreva uma visão que contenha o nome dos médicos que não atendem em nenhum ambulatório
- Escreva uma visão que recupere os funcionários que nunca consultaram