UNIVERSIDADE FEDERAL DE ITAJUBÁ

Banco de Dados

Visões

Adaptado da profa Vanessa Cristina Oliveira de Souza

Visões

- Uma *view* é uma maneira alternativa de manipular dados de uma ou mais tabelas base
- Em uma *view* pode-se combinar dados de uma ou mais tabelas, inserir dados ou fazer outras operações DML

Visões

- Em terminologia SQL, é uma única tabela que é derivada de outras tabelas
- Uma *view* não necessariamente existe de forma física. Ela é considerada uma **tabela virtual**, ao contrário das **tabelas de base**, cujas tuplas sempre estão armazenadas fisicamente no banco de dados
- Assim, uma *view* pode ser considerada como uma tabela virtual ou uma consulta armazenada.

Visões

Num SGBD relacional, uma visão é uma tabela virtual que representa o resultado de alguma consulta ao banco de dados!

Onde se aplicam as Views?

- Controle de acesso
  - Restrição usuário x dados
  - Restrição usuário x domínio
- Modificação da apresentação dos dados
  - Associar vários domínios formando uma única entidade
  - Agregar informações

VIEW

Relação Cliente

CPF	Nome	Estado	Renda_Familiar
12345678910	Maria	MG	1000
12345678911	João	SP	2000
12345678912	Patrícia	MG	5000
12345678913	José	RJ	2500

Descrição dos Usuários do Sistema

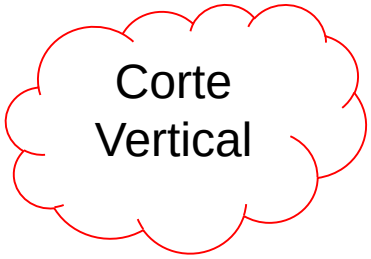
Usuário	Departamento
user1	Marketing
user2	Cobrança
user3	Gerente Regional

VIEW

Relação Cliente

Nome	Estado	Renda_Familiar
Maria	MG	1000
João	SP	2000
Patrícia	MG	5000
José	RJ	2500

Visão do user1



Descrição dos Usuários do Sistema

Usuário	Departamento
user1	Marketing
user2	Cobrança
user3	Gerente Regional

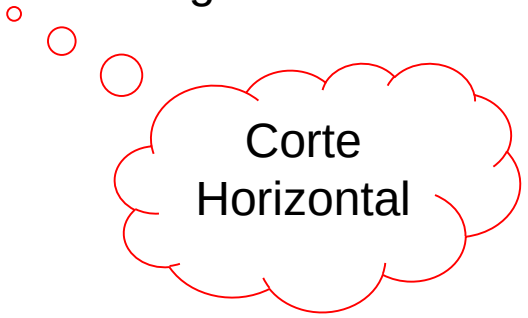
O marketing não tem acesso ao CPF do Cliente.
• Criar uma view somente com os dados que user1 pode acessar

VIEW

Relação Cliente

CPF	Nome	Estado	Renda_Familiar
12345678910	Maria	MG	1000
12345678912	Patrícia	MG	5000

Visão do user3
Gerente Regional de Minas Gerais



Descrição dos Usuários do Sistema

Usuário	Departamento
user1	Marketing
user2	Cobrança
user3	Gerente Regional

O gerente tem acesso à todos os dados da tabela cliente, mas apenas dos cliente da sua região.
• Criar uma view somente com os dados de MG

Sintaxe Básica

- CREATE VIEW <nome_da_view (alias)>
  AS SELECT <especificação_do_select>;

Alias: renomeação dos atributos que serão mostrados pela VIEW

- DROP VIEW <nome_da_view>;

Exemplo - BDEmpresa

funcionario(código,nome,identidade,cpf,email,coddept)
coddept referencia departamento(código)
projeto(código,nome,descrição,datainicial,datafim)

- Elabore uma visão que contenha todos os funcionários do departamento de código 100

create view fdept100 as

(Select * from funcionário where coddept =100)

- Elabore uma visão que contenha o cpf e o nome dos funcionários do departamento de código 100

create view fdept100 as

(Select cpf,nome from funcionário where coddept =100)

Consultas sobre Visões

- Supõe-se que uma *view* esteja sempre atualizada
- Se as tuplas nas tabelas da base sobre as quais a *view* é definida forem modificadas, a view precisa refletir essas mudanças
  - As mudanças aparecerão quando a *view* for consultada novamente
- Alterações nos dados são refletidos na *view*
- Alterações no esquema da tabela não são refletidos na *view*

Atualizações sobre Visões

- Algumas *views* são atualizáveis, ou seja, podem ser usadas para alterar, apagar ou inserir registros nas tabelas base
- Para uma *view* ser atualizável, precisa ter um relacionamento um pra um entre as tuplas da visão e as da tabela base

Atualizações sobre Visões

- Uma *view* com uma única tabela de definição é atualizável se seus atributos tiverem a chave primária da relação da base, bem como todos os atributos com a restrição NOT NULL que não tem valor default especificado
- As views definidas sobre múltiplas tabelas usando junções geralmente não são atualizáveis
- As views definidas usando funções de agrupamento e agregação não são atualizáveis

View materializada

- Os dados da visão são armazenados no banco
- Em alguns casos, a consulta base da view não é executada sempre que view é consultada.
  - Uma view materializada nem sempre é atualizada
  - Normalmente, são usadas para otimizar performance das consultas

View materializada

- Sintaxe no Postgres

CREATE MATERIALIZED VIEW telf (funcionario,telefone) AS

select f.nome,t.numtel from funcionário f join telefone t on f.código = t.codf)

View materializada

- Atualização da Visão depende do SGBD
  - Sempre que a visão for consultada
  - Explicitamente através de um comando
  - Periodicamente
- Inserção, remoção e alteração de dados
  - Depende do SGBD, no POSTGRES não é permitido

View materializada

- DROP MATERIALIZED VIEW
- REFRESH MATERIALIZED VIEW
  - Atualização da view de acordo com a consulta definida na criação
- WITH NO DATA
  - A view é criada sem dados

Exemplo

- Crie um banco de dados universidade e Implemente as tabelas abaixo (todos os atributos são obrigatórios)
- Insira duas tuplas em cada tabela
- Insira na tabela professor, no mínimo, uma tupla com titulação = ‘doutor’

professor(matricula,nome,titulacao,coddept)

disciplina(codigo,nome,descricao)

turma(matriculap, codigod,semestre)

Exemplo

- Dadas as tabelas **professor, disciplina e turma** do banco **universidade**, criar uma visão para mostrar a matrícula e o nome de todos os professores com titulação = ‘doutor’

professor(matricula,nome,titulacao,coddept)

disciplina(codigo,nome,descricao)

turma(matriculap, codigod,semestre)

Exemplo

- Dadas as tabelas **professor, disciplina e turma** do banco **universidade**, criar uma visão para mostrar a matrícula e o nome de todos os professores doutores

Create view doutores as

select p.matricula,p.nome from professor p

where titulacao = ‘doutor’

Exemplo

- Dadas as tabelas **professor, disciplina e turma** do banco **universidade**, criar uma visão para mostrar o nome dos professores com o código das disciplinas que ele ministra

```
Create view profdisc (professor,disciplina)
as    select p.nome, t.codigod
      from professor p join turma t
      on p.matricula = t.matriculap
```

Consultas sobre Visões

- Sobre uma *view* podem ser realizadas consultas:
  - SELECT * FROM doutores;
  - SELECT * FROM doutores WHERE nome LIKE ‘Maria%’;
  - SELECT p.professor, d.nome, FROM profdisc p JOIN disciplina d ON d.codigo = p.disciplina
  - SELECT disciplina, COUNT(professor)

    FROM profdisc GROUP BY (disciplina)

Consultas sobre Visões

- Insira um professor doutor na tabela
- Execute um select * na view doutores
- Verifique que o novo registro inserido na tabela aparece na view
- Execute um select * na view profdisc
- Veja que o novo registro da tabela professor também consta na view

Atualizações sobre Visões

- Tente inserir uma tupla na view doutores e verifique se isso é possível
- Insira uma coluna na tabela professor que deverá armazenar o telefone (coluna não é obrigatória)
- Crie uma view com nome dadosp contendo os dados dos professores com exceção do telefone
- Tente inserir uma tupla na view e verifique se isso é possível

Exercícios

- Crie uma visão materializada que armazene o código do departamento e quantidade de professor
  - Execute um select na view e verifique o resultado
  - Insira um registro na tabela professor e repita o select na view
  - A quantidade de professores foi atualizada?
  - Execute um refresh na view e faça novamente a consulta
- Crie uma visão materializada que armazene o código, o nome e a descrição das disciplinas com a opção with no data
  - Execute um select * na view. Essa operação retornou algum dado?
  - Atualize a view (refresh materialized view nome_da_view)
  - Execute o select * novamente.
  - Tente inserir uma tupla na tabela disciplina através da nova view

Exercício

- Elabore uma view (nomef) contendo o código e o nome do funcionário
- Compare o select * from nomef e select * from funcionario. Os dados retornados foram idênticos?
- Tente inserir um registro em funcionário através da view nomef (insert into nomef values (11,'Paula'). A inserção foi realizada? Qual a justificativa? Você deve considerar como sua tabela funcionário foi criada.

Exercício

- Adicione uma coluna na tabela funcionário (especialidade, não obrigatória, varchar(100)
- Elabore uma view (dadosf) contendo os todos os campos obrigatórios da tabela funcionário
- Tente inserir uma tupla na tabela funcionário usando a view dadosf (insert info dadosf values ....)
- A inserção foi realizada? A criação da coluna especialdiade influenciou na inserção? Qual a justificativa?

Exercício

- Insira uma coluna na tabela funcionário pra armazena o salário do funcionário. Preencha a coluna de todas as linhas da tabela (ao menos 1 linha com salário < 10000 e um com salário > 10000).
- Elabore uma view dadosf10000 com todos os campos obrigatórios da tabela funcionário e o salário considerando os funcionários com salario maior que 10000.
- Insira duas tuplas na tabela funcionário através da view dadosf10000. Uma tupla deve ter o salário > 10000 e a outra deve ter o salário < 10000 (insert into dadosf10000 values ....)
- A inserção foi realizada? Faça um select * from dadosf10000, a tupla com valor < 10000 inserida através da view dadosf10000 aparece no resultado do select?

Exercício

- No POSTGRES, a partir da versão 9.4, é possível usar a cláusula WITH CHECK OPTION (SQL Padrão)
- Quando esta cláusula é acrescentada no final da definição da view, caso a atualização esteja relacionada a tuplas que não pertencem a view, a atualização é rejeitada. Neste caso, a inserção da tupla com salário < 10000 do exercício anterior seria rejeitada
- Visões temporárias (create temp view .....)
  - A view só existe quando a seção é finalizada

Exercício

- Elabore uma visão materializada com o nome do departamento e a quantidade de funcionários.
- Faça um select * na visão criada
- Insira um novo funcionário, verifique se a inserção foi propagada para a visão criada (a quantidade de funcionários do departamento relacionado ao funcionário inserido foi atualizada?)
- Atualize a view criada
  - Execute o comando refresh materialized view ...
- Faça um select * na visão criada novamente e verifique se os dados da view foram atualizados