INSTITUTO FEDERAL
RIO GRANDE DO SUL

Visões, Transações e Índices

TIAGO G MORAES

Roteiro

- Visões
  - Visões materializadas
- Transações
  - Controle de transação
- Índices

Visões

- Conceito:
  - Um "olhar" específico sobre os dados
  - São consultas armazenadas no banco de dados
- Esse "olhar" ou consulta pode:
  - ser realizado muitas vezes
  - ou ser muito custoso
  - Omitir informações por segurança, como salário
- Sendo assim o resultado dessa consulta pode ser armazenado!
  - **Em uma *View***

Visões

- Sintaxe
  - Criação

```
CREATE VIEW <nome_view> [(<col1>,<col2>,...)] AS
(SELECT ...)
```

  - Excluir o objeto *view*:

```
DROP VIEW <nome_view>
```

- Podem ser usadas como relações (outra tabela)
  - Já que o retorno de uma consulta é uma tabela
- Exemplo:

```
CREATE VIEW View_empregado AS
(SELECT nome,cpf,datanascimento) FROM empregado;

SELECT * FROM View_empregado;
```


Visões

- O que acontece se a tabela empregado é atualizada (delete, update ou insert)?
  - Como fica a *view*?
- Política de atualização
  - A tabela resultado nem fica de fato salva no SGBD
  - Sempre são reavaliadas quando forem utilizadas
    - Uma consulta vira duas consultas: recalcular a *view* e a consulta propriamente dita
- Visões materializadas:
  - São efetivamente salvas no banco de dados
  - Cada SGBD possui uma política de atualização

Visões - materializadas

- Criar o objeto *view* materializada:

```
CREATE MATERIALIZED VIEW <nome_view>
(<col1>,<col2>,...) AS (SELECT ...)
```

- Excluir o objeto *view* materializada:

```
DROP MATERIALIZED VIEW  <nome_view>
```

- Comparação entre *views* e *views* materializadas
  - Caso a *view* tenha mais consultas que atualizações:
    - *view* materializada é mais eficiente
  - Caso a *view* tenha mais atualizações que consultas:
    - *view* normal é mais eficiente

Visões

- Atualizações (delete, update e insert) em visões
  - Uma atualização na view deve refletir em uma atualização nas tabelas que geraram a view
  - Isso pode causar problemas: falta de valores essenciais
    - Pois violam restrições de integridade
  - Em geral os SGBD's não permitem tais operações...
- Para realizar uma atualização, normalmente uma view deve:
  - Ter apenas **uma tabela** na cláusula FROM
  - Colunas **sem** expressões ou **distinct**
  - Atributos não listados **não podem ser NOT NULL**
  - **Sem** cláusula **group by**

Índices

- Conceito
  - A intuição é a mesma de um índice de um livro:
    - serve para irmos direto na página do capítulo desejado
    - ao invés de procuramos página a página
  - É uma estrutura de dados complementar vinculada a uma coluna da tabela
  - Serve para tornar mais eficiente uma busca por um elemento da coluna
  - É útil para campos que aparecem muitas vezes nas condições de busca:

```
SELECT * FROM empregado WHERE cpf='01256489750'
```


Índices

- Sintaxe:
  - Criação do índice

```
CREATE INDEX <nome_indice> ON <nome_tab>(<nome_col>)
```

  - Apagar o índice:

```
DROP INDEX <nome_indice>
```

- Tipos:
  - Variam quanto a estrutura de dados que implementam:
    - Hash (busca por chave)
    - **B-Trees (árvores binárias) → padrão postgres**
    - R-Trees (tipo árvores B, mas para informação multidimensional)
    - GiST (árvores de busca genérica), ...

Índices

- Prós:
  - Tunning de banco de dados
  - Melhora a performance de consultas
- Contras:
  - Aumento da estrutura do banco
  - Mais dados sendo gravados
  - Diminui a performance de atualizações (índice necessita também ser atualizado)
- No postgres:
  - Por padrão é criado um índice para todo campo chave (PK ou UNIQUE)

Transações

- Conceito
  - É uma ou mais consultas que devem ser consideradas atômicas:
    - Ou seja, ou todas instruções são realizadas ou nenhuma
  - Exemplo:
    - Transferência bancária: 2 *updates* (um decréscimo e um acréscimo)
    - Quando apenas 1 *update* é realizado isso representa um estado inconsistente do Banco de dados
- É necessário definir o começo e o fim de uma transação

Transações

- Começo de uma transação:
  - **BEGIN TRANSACTION;**
- Volta ao estado antes do começo da transação:
  - **ROLLBACK WORK;**
- Termina e efetiva as mudanças:
  - **COMMIT WORK;**
- Se um erro ocorre o banco garante um ROLLBACK para evitar dados inconsistentes