

FACEC
F a c u l d a d e

Cont. Linguagem SQL

Professor: Yuri Ferreira

- Revisão aula anterior:
 - Exercícios de manipulação: **DELETE** e **UPDATE** e Subconsultas Aninhadas;
 - Visões;

➤ Conteúdo:

- Índices conceitos;
- Vantagens e Desvantagens;
- Índice Btree e hash;
- Triggers;

- **Visões:** Em alguns casos, não é desejável que todos os usuários vejam o **modelo lógico** inteiro (todas as relações reais armazenadas no banco de dados);
- Ex: “Considere uma pessoa que precisa conhecer o **nome** e o **departamento** de um funcionário, mas não o salário”; então em SQL:
 - `Select primeiro_nome, num_depto from funcionario;`
- Uma *view* fornece um mecanismo para **ocultar** determinados dados da exibição de determinados usuários;
- *View* é uma tabela virtual, diferentemente da tabela física;

- **Visões:** são definidas usando a instrução create view:
 - Create view nome_visao as <expressão select>
- Uma vez que a visão é definida, seu nome pode ser utilizado para consultar a tabela virtual;
- Não se cria uma nova tabela, mas na definição da visão é realizado o salvamento da expressão da consulta;

➤ Visões:

- Ex: Visão dos funcionários contendo somente o cpf, o nome e o nome do departamento;

```
create view func_deptos as
select f.primeiro_nome,
f.cpf, d.nome_departamento
from funcionario f
left join departamento d on
(f.num_depto=d.num_depto)
```

➤ Visões:

- Ex: Visão que lista todos funcionários e projetos trabalhados contendo o total de horas;

```
create view func_projetos_trabalhados as
select f.primeiro_nome, p.nome_projeto, t.horas
from trabalha_em t
left join funcionario f on
    (f.cpf=t.cpf_funcionario)
left join projeto p on
    (p.num_projeto=t.num_projeto)
```


➤ **Indices:**

- Índice de um banco de dados funciona como se fosse um índice de um livro ou dicionário;
- No dicionário as palavras estão em ordem alfabética, logo fica mais fácil encontrá-las;
- Utiliza-se para encontrar uma palavra a **Busca binária**;
- Índices são utilizados nos **atributos** de uma tabela;
- Geralmente em **chaves** e **campos** de consultas, visando melhorar a performance de busca;

➤ **Índices:**

- Muitas consultas fazem **referência** apenas a uma **pequena proporção** dos registros em uma tabela;
- É **ineficiente** para o sistema ler todos os registros para encontrar um registro com valor específico;

➤ **Indices:**

➤ **Vantagens:**

- O acesso aos dados é enormemente reduzido;
- Os índices são montados de uma forma que permite buscar apenas uma parte dos dados;
- O Índice mais comum e utilizado é tipo árvore binária (BTree);
 - Uma busca binária tem complexidade $O(\log N)$;
 - Melhora a performance principalmente para grandes volumes de dados (milhões/bilhões de linhas);
 - Ex: **1 bilhão** de linhas, em uma consulta utilizando índices, o resultado é alcançado com um pouco mais que **30** passos;

➤ **Índices:**

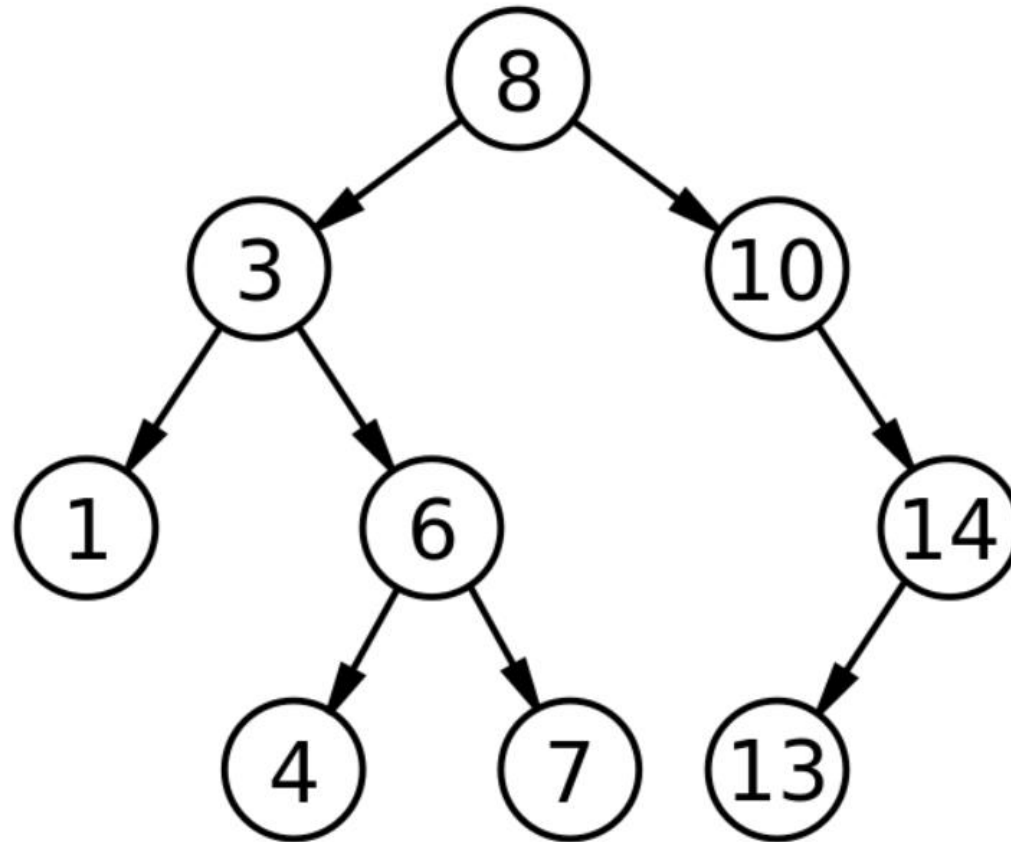
- Os índices utilizados com mais frequência tendem a ficar em **memória**;
- Permite o acesso aos dados ordenados, sem o custo de realizar ordenação;

➤ **Desvantagens:**

- Aumenta o tempo de escrita e atualização no banco, pois tem que rodar algoritmos de ordenação;
- E aumenta o espaço ocupado pelo banco, pois fisicamente ficam armazenadas os índices em estruturas como se fosse tabelas;

➤ **Indices:**

➤ **Árvore Binária:**



➤ **Indices:**

➤ **Hash:**

➤ Índices Hash tem complexidade $O(n)$;

➤ Eles são utilizados junto com uma função Hash que identifica a chave específica de um registro;


➤ Indices:

➤ Instrução para criação de índices em uma tabela:

➤ Create index <nome> on <tabela>(atributo);

➤ Ex:

```
create index cpf_func_idx  
on funcionario(cpf);
```



```
select * from funcionario  
where cpf='91875163905'
```

```
CREATE UNIQUE INDEX funcionario_cpf_idx  
ON funcionario USING btree (cpf);
```

➤ Triggers:

- São regras ativas que executam **ações** de forma automática, quando ocorre certos **eventos** no banco de dados;
- São utilizados para impor restrições complexas da aplicação, por exemplo:
 - Monitorar a media dos alunos sempre que uma nota nova for inserida;
 - Verificar o pré-requisito do curso antes do aluno ser matriculado em uma disciplina;
- Outras aplicações incluem manutenção automática de dados derivados;

➤ Triggers:

➤ Exemplos:

- Pode ser utilizado para replicar em uma tabela log as alterações efetuadas em outras tabelas;
- Suponhamos um atributo **salário_total** na tabela **departamento**, que é derivado da soma dos salários de todos os **funcionários**;
 - Para manutenção deste atributo pode ser utilizado uma **trigger**;

➤ Triggers:

- Eventos que podem causar uma mudança no **sal_total**, são:
 - 1º Inserir (uma ou mais) linhas de novos funcionários;
 - 2º Alterar o salário de um ou mais funcionários existentes;
 - 3º Alterar a designação dos funcionários existentes de um departamento para outro.
 - 4º Excluir linhas de funcionários;
- A **ação** para os **eventos** 1,2,4 é atualizar automaticamente o valor de **sal_total** do departamento, refletindo a inserção, atualização ou exclusão;
- Para o **evento** 3, duas **ações** são necessárias: atualizar o **sal_total** do antigo e do novo departamento;

➤ Triggers:

- Cada SGDB implementa as triggers de uma forma;
- Estrutura do commando no PostgreSQL:

```
CREATE TRIGGER nome_trigger [BEFORE | AFTER | INSTEAD OF]  
Evento  
ON nome_tabela  
[  
    -- Lógica da Trigger ...  
];
```

- Propriedades: NEW e OLD;

➤ Triggers:

➤ Exercício:

- 1) Criar uma coluna na tabela departamento responsável por armazenar a soma total dos salários de todos funcionários do departamento;
- 2) Criar uma Trigger na tabela funcionário que ao inserir um novo funcionário seja atualizado o total dos salários do departamento, na tabela departamento, considerando este novo cadastro;

➤ Referências:

- SILBERSCHATZ, A.; KORTH, F.; SUDARSHA, S. Database System Concepts. 6. ed. Nova York: MC Graw Hill, 2011.
- ELMASRI, R.; NAVATHE B. Sistemas de banco de dados. 6. Ed. São Paulo, SP: Pearson Addison-Wesley, 2011.