Nomes: Edson, Layla e Gustavo Nunes.

BANCO DE DADOS II

Criação de Índices (de variados tipos) no **PostgreSQL (com ênfase na criação de índices com múltiplas colunas em B-Trees e índices de árvore R).**

1- Criação de Índices (de variados tipos) no PostgreSQL (com ênfase na criação de índices com múltiplas colunas em B-Trees e índices de árvore R).

R: Tipo de criação de índice de árvore B-Trees:
O postgree utiliza como padrão a criação de index em árvores B(sendo o B significando “balanced”), se ajustando em situações mais comuns, tendo a ideia de que ambos os lados da árvore sejam mais ou menos os mesmos.
Exemplo:

Criamos uma tabela simples de paciente:

```
create table paciente(
   idpaciente serial primary key,
   nomepaciente varchar(60)
);
```

Criamos um index onde utilizamos o nome do paciente como estrutura:

```
create index nomePaciente on paciente(nomepaciente);
```

Podemos criar índices em múltiplas colunas:

Criamos uma outra tabela:

```
create table usuario(
   idusuario serial primary key,
   nomeusuario varchar(60),
   cpf integer,
   cidade varchar(40)
);

insert into usuario(nomeusuario, cpf, cidade) values ('gustavo', 212, 'colatina');
```

Criamos um índice mas agora utilizando como base o nome e o cpf do usuario:

```
create index nomeUSUARIOCpf on usuario(nomeusuario, cpf);
```

Índice Hash:

Os índices hash são utilizados para lidar com comparações de igualdade simples.

Exemplo utilizando a mesma tabela criada acima:

create index idXNome on usuario USING hash(nomeusuario);

Índice GIN:

Os índices GIN são úteis para mapear vários valores em uma linha, diferindo dos índices B-Tree que são otimizados para quando uma linha possui um valor único de chave, também sendo preferíveis quando utilizados para retornar um valor de tipo texto.
Exemplo:
Create index buscaCpf on usuario using gin(cpf);

Índice R-Tree:

```
create table usuario(
   idusuario serial primary key,
   nomeusuario varchar(60),
   cpf integer,
   cidade varchar(40)
);

insert into usuario(nomeusuario, cpf, cidade) values ('gustavo', 212, 'colatina');

create index nomeUSUARIOCpf on usuario(nomeusuario, cpf);

drop table paciente;

create index idXNome on usuario USING hash(nomeusuario);

drop index idXNome;

create index rtree on usuario using rtree (cpf); /*o rtree nao possui muitas vatanges em relacao ao metodo gist, então o postgresql automaticamente converte para o metodo gist*/
```

Índice GIST:

O índice gist gera muitas perdas porque cada documento é representado no índice por uma assinatura de tamanho fixo. A assinatura é gerada pelo hash de cada palavra em um único bit em uma string de n bits, com todos esses bits combinados em OR para produzir uma assinatura de documento de n bits. Quando duas palavras se misturam à mesma posição de bit, haverá uma correspondência falsa. Se todas as palavras na consulta tiverem correspondências (real ou falsa), a linha da tabela deverá ser recuperada para verificar se a correspondência está correta.

create index busca_cidade on usuario using gist(cidade);

1. Explique as possíveis **vantagens** e **desvantagens** de **índices B-Tree com múltiplas colunas**.

Vantagens: B-Trees tendem a ser mais econômicas; A rotina de inserção e exclusão são rápidas; Todos os nós folha sempre ficam no mesmo nível; Os índices BTree mantém-se atualizados com informações lógicas com todas as informações gravadas no banco de dados o que permite ao banco de dados distinguir dados que possuem algum nexo entre si dos que não possuem, dividindo-os em "nós" lógicos e "folhas" que serão apreciados pelo algoritmo BTree.

Desvantagens: É necessário manter as chaves num nó em ordem linear; Se o banco de dados for pequeno o índice B-Tree obterá uma menor performance; O processo de exclusão pode ser demorado devido a necessidade de re-alocar os "nós" novamente.

2. A **estrutura** de uma **árvore-R**.

Árvores R são árvores de estruturas de dados que são similares as árvores B, mas que são usadas para métodos de acesso no espaço com o fim de indexar informação multidimensional, por exemplo, as coordenadas (X, Y) de uma posição geográfica. Um

exemplo de uso comum das árvores R seria: "Encontre todos os museus que estão até no máximo 1,5 km da minha posição geográfica atual".

A estrutura de dados divide o espaço com aninhamento hierarquico, e possivelmente sobreposição de retangulos de limite mínimos (também chamados de caixas limitantes - bounding boxes).

Cada nodo de uma árvore R tem um número variável de entradas(até um limite máximo pré-definido). Cada entrada dentro de um nodo. que não é folha, armazena dois tipos de dados: uma maneira de identificar um nodo filho, e a caixa limitante de todas as entradas dentro deste nodo filho.

3. O **algoritmo** de uma **árvore-R**.

Os algoritmos de inserção e remoção usam caixas limitantes a partir dos nodos para assegurar que todos os elementos próximos estão localizados no mesmo nodo folha (Em particular, um novo elemento será adicionado ao nodo folha que requer o menor aumento espacial na sua caixa limitante). Cada entrada dentro de um nodo folha armazena também dois pedaços de informação: uma maneira de identificar o elemento de dados real(o que, alternadamente, pode ser colocado diretamente no nodo), e a caixa limitante do elemento de dados.

Analogamente, os algoritmos de busca(por exemplo: intersecção, contém, mais próximo) usam caixas limitantes para decidir se irão buscar ou não dentro de um nodo filho. Desta forma, a maioria dos nodos na árvore jamais são alcançados durante uma busca. Como árvores B, isto torna as árvores R mais indicadas para banco de dados, onde nodos podem ser paginados na memória do computador quando preciso.

Algoritmos diferentes podem ser usados para dividir os nodos quando eles se tornam cheios demais, resultando em subtipos de árvores R: quadricas e lineares.

Árvores R não garantem historicamente boa performance no pior caso, mas geralmente demonstram boa performance na aplicação prática com dados reais. Contudo, um novo algoritmo foi publicado em 2004 que definiu a prioridade de uma árvore R, e que

assume ser tão eficiente quanto os mais eficientes métodos atuais e assume-se ter ótima performance nos testes de pior caso.

4. Quando um índice baseado em árvore-R é útil?

São úteis para métodos de acesso no espaço com o fim de indexar informação multi-dimensional.