Tarefa

Luiz Roberto Rodrigues Nobre

1. Pesquise quais são os tipos de dados suportados pelo PostgreSQL e quantidade de memória necessária para armazenar cada um.

De acordo com a documentação oficial do PostgreSQL, há mais de 20 tipos de dados que podem ser inseridos, mas o que nos interessa são os seguintes tipos:

- Numeric Types
- Monetary Types
- Character Types
- Date/Time Types
- Boolean Type
- Enumerated Types
- UUID Type

Tabela de Numeric Types

Nome dos numéricos	Tamanho em Bytes	Observações
smallint	2	
integer	4	
bigint	8	
decimal	Variável	Precisão especificada pelo usuário
numeric	Variável	Precisão especificada pelo usuário
real	4	
double precision	8	
smallserial	2	
serial	4	
bigserial	8	

Monetary Types

O tipo `monetary` ocupa 8 bytes, com uma precisão fracionária fixa (0.00, por exemplo), mas também pode ser definida na configuração do banco `lc_monetary`.

Character Types

Nome	Tamanho em bytes	Observações

Nome	Tamanho em bytes	Observações
character varying, varchar	Tamanho variável com limite	
character, char, bpchar	Tamanho fixo	Caso não seja totalmente ocupado, é completado com espaços
text	Tamanho variável ilimitado	

Date/Time Types

Nome	Tamanho em bytes	Observações
timestamp	8	Possui tempo e data sem fuso horário
timestamp	8	Possui tempo e data com fuso horário
date	4	Apenas data
time	8	Apenas tempo
interval	16	Intervalo de tempo

Boolean Type

O `boolean` possui apenas 1 byte de tamanho, e salva os valores `TRUE`, `YES`, `ON`, `1` (verdadeiro) e `FALSE`, `NO`, `OFF`, `0` (falso).

UUID Type

`UUID` significa *Universally Unique Identifier*. Tem o tamanho de 16 bytes (128 bits) e é um identificador único gerado por algoritmo.

---

2. Quais são os tipos de arquivos suportados pelo PostgreSQL?

Heap

É o formato padrão de armazenamento de tabelas no PostgreSQL. Cada tabela é guardada em arquivos segmentados no diretório `PGDATA/base/(diretório de armazenamento)`, com blocos de 8KB.

Sequencial

O PostgreSQL armazena os dados na ordem em que são inseridos. O comando `Seq Scan` permite a varredura sequencial dos registros.

Hashing

Usado através de índices do tipo `hash`, criando uma estrutura de armazenamento chamada *bucket* para valores com o mesmo hash. Melhora a performance de buscas exatas.

Clustering

Permite organizar fisicamente os blocos da tabela conforme uma chave de ordenação, através do comando `CLUSTER`. Não é mantido automaticamente.

---

3. Documentação inicial do Projeto Físico da "Seguradora"

Cliente

Identificador	Cliente
Descrição	Tabela de armazenamento de todos os clientes da seguradora
Organização	Tabela Heap
Ordenação	numero

- Atributos

Nome	Chave	Tipo de dados	Tamanho	Restrições	Bytes
numero	Primária	Integer	-	Not Null, Sem repetição	4
nome	-	Varchar	100	Not Null	100

Logradouro

Identificador	Logradouro
Descrição	Tabela de armazenamento de logradouros vinculados aos clientes
Organização	Tabela Heap
Ordenação	numero

- Atributos

Nome	Chave	Tipo de dados	Tamanho	Restrições	Bytes
numero	Primária	Integer	-	Not Null, Sem repetição	4
data	-	Date	-	Pode ser nula	4
cliente_numero	Estrangeira	Integer	-	Chave estrangeira, Not Null	4

Carro

Identificador	Carro

Identificador	Carro
Descrição	Tabela de armazenamento de carros cadastrados
Organização	Tabela Heap
Ordenação	numero

- Atributos

Nome	Chave	Tipo de dados	Tamanho	Restrições	Bytes
numero	Primária	Integer	-	Not Null, Sem repetição	4
nome	-	Varchar	100	Not Null	100
modelo	-	Varchar	100	Pode ser nulo	100
logradouro_numero	Estrangeira	Integer	-	Chave estrangeira, Única	4

Acidente

Identificador	Acidente
Descrição	Tabela de armazenamento de todos os acidentes registrados
Organização	Tabela Heap
Ordenação	id_acidente

- Atributos

Nome	Chave	Tipo de dados	Tamanho	Restrições	Bytes
id_acidente	Primária	Integer	-	Not Null, Auto Increment	4
data	-	Date	-	Pode ser nula	4
hora	-	Time	-	Pode ser nula	8
local	-	Varchar	200	Pode ser nula	200
carro_numero	Estrangeira	Integer	-	Chave estrangeira, Not Null	4

4. Script SQL da criação das tabelas

```sql
-- tabela Cliente
CREATE TABLE Cliente (
    numero INT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL
);

-- tabela Logradouro
CREATE TABLE Logradouro (
```

```
    numero INT PRIMARY KEY,
    data DATE,
    cliente_numero INT NOT NULL,
    FOREIGN KEY (cliente_numero) REFERENCES Cliente(numero)
);

-- tabela Carro
CREATE TABLE Carro (
    numero INT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    modelo VARCHAR(100),
    logradouro_numero INT UNIQUE,
    FOREIGN KEY (logradouro_numero) REFERENCES Logradouro(numero)
);

-- tabela Acidente
CREATE TABLE Acidente (
    id_acidente INT PRIMARY KEY AUTO_INCREMENT,
    data DATE,
    hora TIME,
    local VARCHAR(200),
    carro_numero INT,
    FOREIGN KEY (carro_numero) REFERENCES Carro(numero)
);
```