Estratégia Concursos

Aula 01

Banco do Brasil (Escriturário - Agente de Tecnologia) Passo Estratégico de Tecnologia de Informação

Autor:
Fernando Pedrosa Lopes

12 de Julho de 2024

POSTGRESQL

Sumário

CONTEÚDO

PostgresSQL. Introdução e conceitos básicos. Histórico. Arquitetura. Tipos de Dados. Funções e operadores. Índices e otimização. Segurança. Backup e recuperação.

ANÁLISE ESTATÍSTICA

Inicialmente, convém destacar o percentual de incidência do assunto, dentro da disciplina **Bancos de Dados** em concursos/cargos similares. Quanto maior o percentual de cobrança de um dado assunto, maior sua importância.

Obs.: *um mesmo assunto pode ser classificado em mais de um tópico devido à multidisciplinaridade de conteúdo*.

Assunto	Relevância na disciplina em concursos similares
SQL	**21.6 %**
BI (Business Intelligence)	**9.0 %**
DW - Data Warehouse	**7.2 %**
SQL Server	**7.2 %**
Oracle	**6.3 %**
Banco de Dados Multidimensionais	**5.4 %**
Data Mining	**5.4 %**
Administração de banco de dados	**3.6 %**
Banco de Dados	**2.7 %**
Formas normais	**2.7 %**
ETL (Extract Transform Load)	**2.7 %**

Banco de Dados Relacionais	**2.7 %**
Arquitetura de Banco de Dados	**1.8 %**
SGBD - Sistema de Gerenciamento de Banco de Dados	**1.8 %**
OLAP (On-line Analytical Processing)	**1.8 %**
Segurança	**1.8 %**
MS-Access	**1.8 %**
Modelo relacional	**1.8 %**
Metadados e Metainformação	**1.8 %**
Álgebra relacional	**0.9 %**
Banco de Dados Paralelos e Distribuídos	**0.9 %**
Gerência de Transações	**0.9 %**
Modelagem de dados	**0.9 %**
Gatilhos (Triggers)	**0.9 %**
DER - Diagrama de Entidade e Relacionamento	**0.9 %**
Visão (View)	**0.9 %**
Banco de Dados Textuais	**0.9 %**
Índices	**0.9 %**
PostgreSQL	**0.9 %**
MySQL	**0.9 %**
Big Data	**0.9 %**

GLOSSÁRIO DE TERMOS

Faremos uma lista de termos que são relevantes ao entendimento do assunto desta aula. Caso tenha alguma dúvida durante a leitura, esta seção pode lhe ajudar a esclarecer.

PostgreSQL: Um sistema de gerenciamento de banco de dados relacional de código aberto, avançado e rico em recursos.

pgAdmin: Uma ferramenta gráfica de código aberto e multiplataforma para a administração e gestão de bancos de dados PostgreSQL.

Postmaster: Processo principal no PostgreSQL que gerencia conexões de cliente, inicia processos de servidor filho para lidar com essas conexões e realiza tarefas de manutenção do sistema.

WAL: Write-Ahead Logging. É a técnica usada pelo PostgreSQL para garantir a integridade dos dados registrando todas as modificações em um log antes de confirmar a transação que a causou.

MVCC: Multi-Version Concurrency Control. É o método que o PostgreSQL usa para lidar com a concorrência de dados, permitindo que várias transações leiam e escrevam sem interferir umas nas outras.

Savepoint: No PostgreSQL, um savepoint é um ponto na transação que você pode rolar para trás, sem precisar abortar toda a transação.

Cluster: Conjunto de bancos de dados que são gerenciados por uma única instância do servidor PostgreSQL.

Namespace: Em PostgreSQL, um namespace é um contêiner que contém um conjunto nomeado de objetos, como tabelas e funções. O termo é usado de maneira intercambiável com "schema".

PGDATA: Variável de ambiente que define o local do diretório de dados do PostgreSQL.

pg_catalog: O esquema pg_catalog é um esquema especial que contém as tabelas do sistema, que são tabelas construídas internamente pelo PostgreSQL para o seu funcionamento.

psql: A interface de linha de comando para o PostgreSQL.

ACL: Access Control List. Em PostgreSQL, ACLs são usadas para definir permissões em diferentes objetos do banco de dados.

RLS: Row Level Security. É uma funcionalidade do PostgreSQL que permite controlar quais usuários podem selecionar, inserir, atualizar ou excluir quais linhas de uma tabela baseada em políticas definidas.

pgAudit: Uma extensão do PostgreSQL que fornece capacidades de auditoria detalhadas para atividades de banco de dados.

pg_dump: Ferramenta de linha de comando usada para fazer backup de um banco de dados PostgreSQL.

pg_restore: Ferramenta de linha de comando usada para restaurar um banco de dados PostgreSQL a partir de um arquivo de backup criado pelo pg_dump.

ROTEIRO DE REVISÃO

A ideia desta seção é apresentar um roteiro para que você realize uma revisão completa do assunto e, ao mesmo tempo, destacar aspectos do conteúdo que merecem atenção.

Introdução

PostgreSQL é um Sistema de Gerenciamento de Banco de Dados (SGBD) de código aberto e objeto-relacional. É altamente personalizável, escalável e projetado para lidar com uma ampla gama de cargas de trabalho, desde pequenos aplicativos a grandes sistemas de internet com muitos usuários concorrentes.

PostgresSQL suporta transações ACID (Atomicidade, Consistência, Isolamento, Durabilidade), tem suporte robusto para integridade referencial (incluindo chaves estrangeiras) e também suporta muitos recursos avançados, como a capacidade de criar tipos de dados personalizados, funções armazenadas e procedimentos, bem como a capacidade de definir operadores para esses tipos.

A história do PostgreSQL começa em 1986, como parte do projeto POSTGRES na Universidade da Califórnia, Berkeley. Esse projeto, liderado pelo professor de Ciência da Computação Michael Stonebraker, visava superar as limitações dos SGBDs existentes ao tempo.

O POSTGRES introduziu muitos conceitos que se tornaram fundamentais para os SGBDs, incluindo a capacidade de definir tipos de dados e a possibilidade de usar linguagem de programação no banco de dados para definir esses tipos.

Em 1994, Andrew Yu e Jolly Chen adicionaram uma camada SQL ao POSTGRES, criando o Postgres95. Em 1996, a nova versão foi rebatizada para PostgreSQL para refletir a inclusão do SQL no projeto.

Desde então, o PostgreSQL tem sido mantido por um grupo de desenvolvedores de todo o mundo, conhecido como a PostgreSQL Global Development Group. Eles continuam lançando novas versões do PostgreSQL regularmente, adicionando novos recursos, melhorando o desempenho e aprimorando a segurança.

Ao longo dos anos, o PostgreSQL incorporou muitos recursos adicionais, tais como:

- A capacidade de agir como um banco de dados NoSQL, armazenando e consultando dados JSON e JSONB
- Suporte para dados geoespaciais por meio da extensão PostGIS
- Melhorias no particionamento de tabelas
- Melhorias na replicação lógica e na alta disponibilidade
- Inúmeras melhorias de desempenho, otimização de consultas e monitoramento

Atualmente, o PostgreSQL é usado por muitas grandes organizações, incluindo Apple, Cisco, Fujitsu, e o Federal Bureau of Investigation dos Estados Unidos, além de ser a espinha dorsal de muitos aplicativos web populares.

Características Gerais

PostgreSQL é conhecido por ser um dos sistemas de gerenciamento de banco de dados mais ricos em recursos e flexíveis disponíveis. Veja algumas de suas características mais importantes:

- **Completamente ACID**: O PostgreSQL é totalmente ACID (Atomicidade, Consistência, Isolamento e Durabilidade), o que significa que ele garante a integridade dos dados mesmo em caso de falha do sistema ou falha de energia.
- **Compatibilidade SQL**: O SGBD adere completamente ao padrão SQL e suporta muitas de suas funcionalidades avançadas. Ele também adiciona extensões não padrão, incluindo algumas que não estão disponíveis em outros SGBDs, como consultas recursivas e funções de janela.
- **Suporte extensivo a tipos de dados**: Suporta vários tipos de dados, incluindo tipos primitivos, tais como inteiro, numérico, booleano, caráter e de data e hora, além de tipos complexos, como array, hstore (para dados do tipo chave-valor), json e jsonb (para dados JSON), ltree (para dados de árvore) e intervalos.
- **Extensibilidade**: Uma das características mais poderosas do PostgreSQL é sua extensibilidade. Ele permite aos usuários criar seus próprios tipos de dados, operadores e funções. Além disso, o PostgreSQL suporta uma série de extensões que adicionam funcionalidades como suporte geoespacial (PostGIS), suporte a séries temporais (TimescaleDB), etc.

- **Segurança sólida**: Inclui uma série de recursos de segurança robustos. Suporta vários métodos de autenticação, incluindo certificados SSL e SCRAM-SHA-256. O PostgreSQL também suporta a criação de funções e procedimentos em linguagens seguras, como PL/pgSQL e PL/PythonU.
- **Recursos de otimização e desempenho**: O PostgreSQL tem um planejador de consultas avançado e um sistema de otimização de consultas robusto. Ele suporta vários tipos de índices, incluindo B-trees, hashes, GiST, SP-GiST, GIN e BRIN.
- **Suporte a transações**: O banco de dados suporta transações aninhadas (ou seja, transações dentro de transações) e possui um modelo de controle de concorrência multiversão (MVCC) sofisticado. Isso ajuda a minimizar bloqueios e mantém o banco utilizável em ambientes altamente concorrentes.
- **Replicação e alta disponibilidade**: Oferece várias opções para replicação e alta disponibilidade, incluindo replicação síncrona e assíncrona, bem várias soluções de terceiros para balanceamento de carga e failover automático.

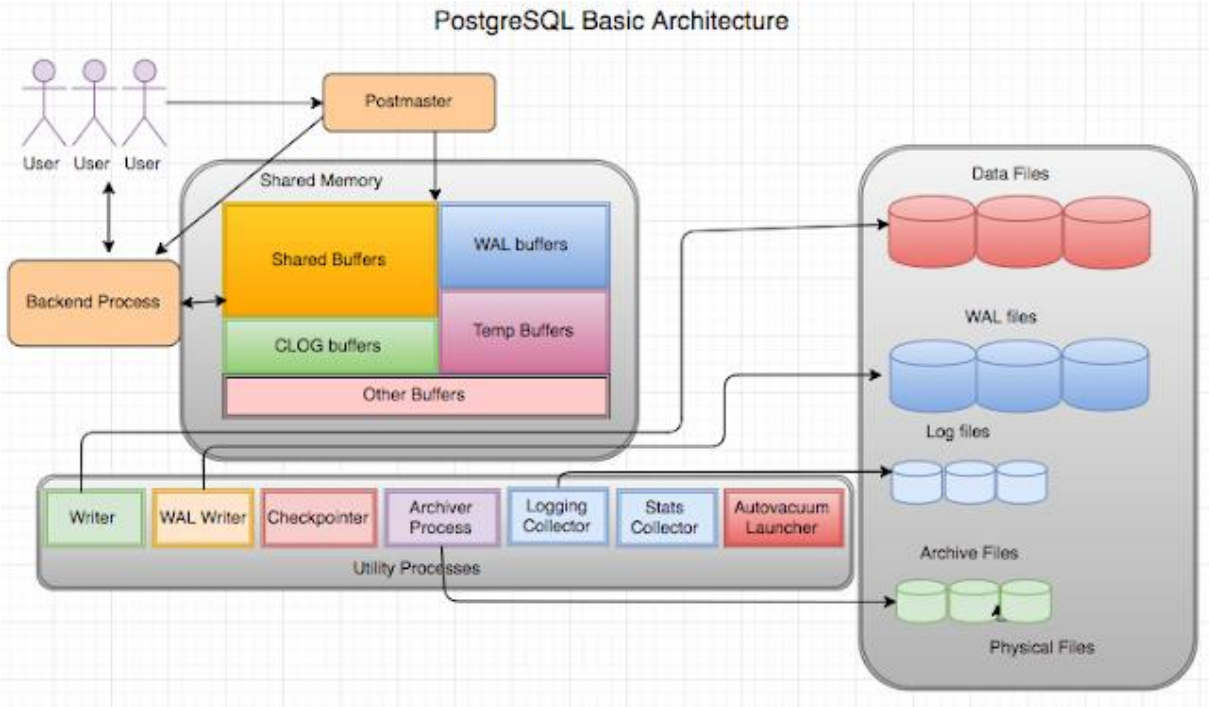
Instalação do PostgresSQL e pgAdmin

O PostgreSQL oferece várias formas de instalação e distribuição, cada uma adequada para diferentes necessidades e ambientes de trabalho. Algumas das formas mais comuns são:

- **Instalações Binárias**: A maneira mais simples e direta de instalar o PostgreSQL em sistemas Windows, MacOS e Linux. Os instaladores binários contêm uma versão pré-compilada do PostgreSQL que pode ser instalada diretamente. Os binários para Windows e MacOS incluem o pgAdmin, uma interface gráfica de usuário para gerenciamento de bancos de dados PostgreSQL.
- **Instalação a partir do código-fonte**: Para usuários mais avançados, o PostgreSQL pode ser compilado a partir do código-fonte. Isso oferece o máximo de flexibilidade, permitindo que você personalize sua instalação para se ajustar exatamente às suas necessidades.
- **Pacotes de distribuição**: Em muitas distribuições de Linux, o PostgreSQL pode ser instalado diretamente dos repositórios de pacotes da distribuição. Por exemplo, em sistemas baseados em Debian, como o Ubuntu, você pode instalar o PostgreSQL usando o gerenciador de pacotes **apt**. Em sistemas baseados em Red Hat, como o CentOS, você pode usar o **yum**.
- **Contêineres Docker**: Se você estiver trabalhando em um ambiente que faz uso intenso de contêineres, você pode usar a imagem oficial do PostgreSQL no Docker. Isso permite que você execute o PostgreSQL em um contêiner isolado, o que pode simplificar o gerenciamento de dependências e a configuração do ambiente.
- **Serviços em nuvem**: Vários provedores de nuvem, como Amazon RDS, Google Cloud SQL e Microsoft Azure, oferecem PostgreSQL como um serviço. Isso permite que você execute o PostgreSQL em um ambiente de nuvem gerenciado, onde tarefas como backup, replicação e escalonamento são gerenciadas pelo provedor de serviços em nuvem.

Arquitetura do PostgreSQL

A arquitetura do PostgreSQL é complexa e altamente modular, permitindo muita flexibilidade e extensibilidade. Vamos abordar seus componentes principais.



1. Processo de Backend: Quando um cliente se conecta ao PostgreSQL, um novo processo backend é iniciado. Cada processo backend é independente e possui sua própria memória privada.

2. Memória Compartilhada: Esta é a memória que pode ser acessada por todos os processos backend. Ela é usada para armazenar informações que devem ser compartilhadas entre os processos, como buffers de dados, bloqueios e informações de transações.

3. Processo Postmaster: É o processo principal que inicializa o sistema de banco de dados. Ele é responsável por aceitar conexões de clientes, iniciar novos processos de backend e gerenciar a comunicação entre os processos.

4. Arquivos de Dados: São onde o PostgreSQL armazena seus dados permanentemente. Cada tabela e índice tem seu próprio arquivo de dados.

5. WAL (Write-Ahead Logging): O PostgreSQL usa o Write-Ahead Logging para manter a integridade dos dados. Antes de qualquer alteração ser feita nos arquivos de dados, ela é registrada no log de transações (WAL). Isso garante que, em caso de falha do sistema, todas as transações podem ser recuperadas.

6. Vacuum: Este é um processo interno que limpa os dados que não são mais necessários. Quando as linhas são atualizadas ou excluídas, as versões antigas das linhas são mantidas para garantir que as transações em andamento ainda possam ver os dados antigos. O Vacuum remove essas linhas antigas para liberar espaço.

7. Buffer Cache: É uma área na memória compartilhada onde o PostgreSQL mantém cópias das páginas dos arquivos de dados. Isso permite que o PostgreSQL leia e escreva dados muito mais rapidamente do que se tivesse que acessar os arquivos de dados toda vez.

8. Executor: É a parte do PostgreSQL que executa as consultas. Ele toma um plano de consulta produzido pelo planejador e executa-o para retornar os resultados da consulta.

9. Planejador/Otimizador: Este componente recebe a consulta SQL, analisa-a e cria um plano de execução eficiente. Ele decide a ordem na qual as tabelas e linhas são acessadas, quais índices são usados, como as junções são executadas e assim por diante.

10. Parser: Este componente analisa a consulta SQL e a transforma em uma estrutura de dados interna chamada árvore de consulta.

Sistema de transações e controle de concorrência multiversionado (MVCC)

O PostgreSQL usa um modelo chamado Controle de Concorrência Multiversionado (MVCC) para gerenciar transações simultâneas. O MVCC permite que muitos usuários acessem o mesmo banco de dados ao mesmo tempo sem bloquear uns aos outros.

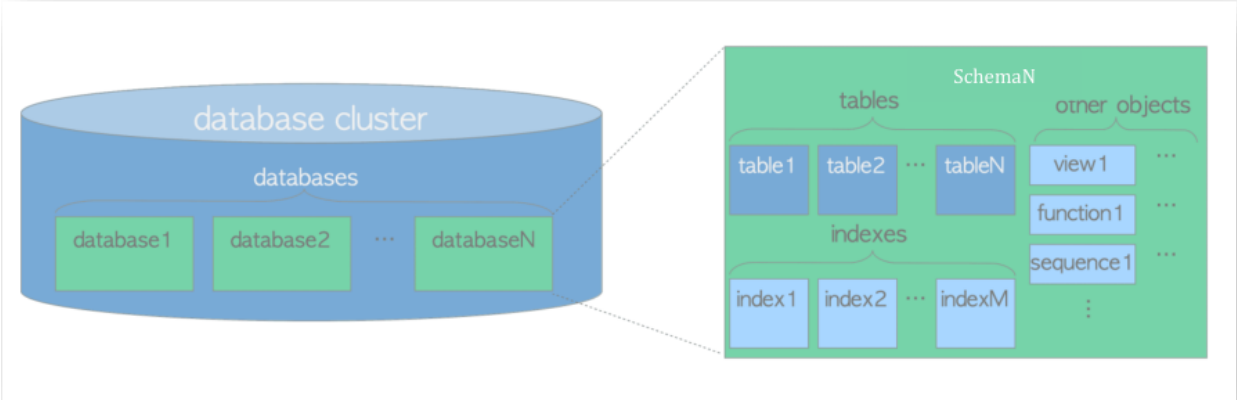
No MVCC, cada transação vê um "instantâneo" (*snapshot*) do banco de dados - uma visão do banco de dados no momento em que a transação começou. As transações podem fazer alterações no banco de dados sem afetar as visões de outras transações. Quando uma transação é concluída, suas alterações são "commitadas" - tornadas permanentes e visíveis para outras transações.

Se duas transações tentarem modificar a mesma linha ao mesmo tempo, o PostgreSQL usa um sistema de "timestamping" para determinar qual transação chegou primeiro. A transação que chega por último terá que esperar ou será cancelada, dependendo do nível de isolamento de transação definido.

O PostgreSQL também suporta transações aninhadas através de um recurso chamado **Savepoints**. Os Savepoints permitem que você comece uma transação dentro de outra transação, que pode ser revertida sem afetar a transação externa.

Terminologia

Antes de avançarmos, os termos abaixo são importantes para entender a arquitetura do PostgreSQL. Aqui está uma revisão básica:

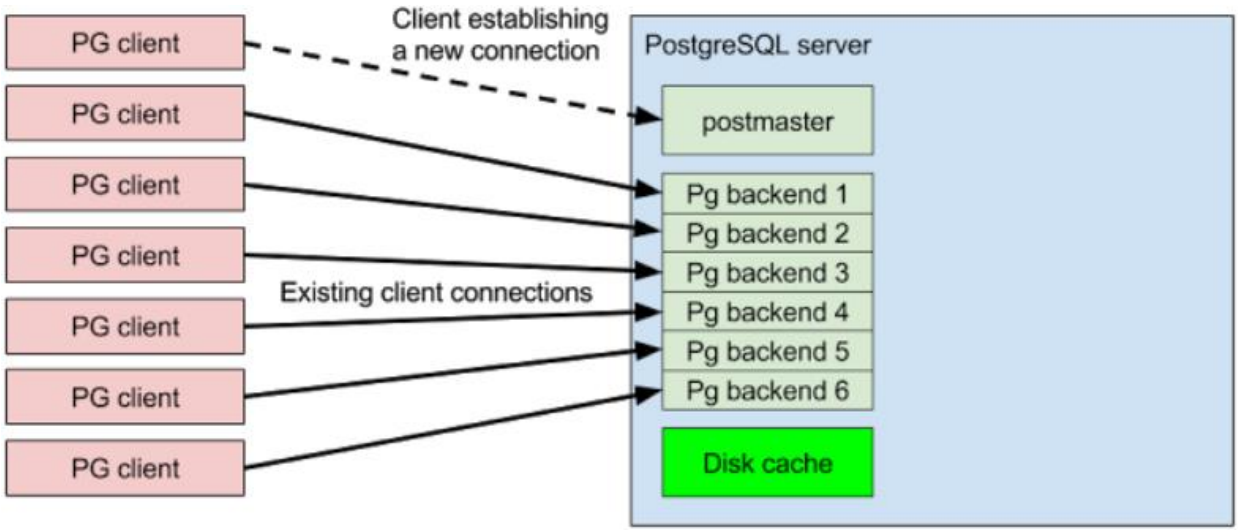


Cluster: No contexto do PostgreSQL, um cluster não se refere ao agrupamento de vários servidores, como o termo é comumente usado. Em vez disso, um cluster PostgreSQL é uma coleção de bancos de dados que são gerenciados por uma única instância do servidor PostgreSQL. Ele inclui um ou mais bancos de dados, bem como tabelas de sistema que o PostgreSQL usa para gerenciar o cluster.

Banco de dados: Um banco de dados no PostgreSQL é um espaço de nomes que contém tabelas, visões, índices e outros objetos de banco de dados. Cada banco de dados pertence a um único cluster e é isolado dos outros bancos de dados no cluster. Ou seja, você não pode acessar objetos de um banco de dados a partir de outro banco de dados.

Namespace: No contexto do PostgreSQL, um "namespace" é um espaço para nomeação que é usado para isolar e organizar objetos de banco de dados, como tabelas, índices e funções. Em termos mais simples, é um contêiner que agrupa objetos relacionados, evitando conflitos de nomes. Namespaces são geralmente sinônimo de "schema". Um schema é uma coleção de objetos de banco de dados que pertencem a um banco de dados específico. Cada banco de dados pode ter vários schemas, e eles ajudam a organizar e separar os objetos dentro de um banco de dados.

Postmaster: Postmaster é o processo principal que inicializa o sistema de banco de dados PostgreSQL. Ele é responsável por aceitar conexões de clientes, iniciar novos processos de backend e gerenciar a comunicação entre os processos. Se o Postmaster falhar, todos os processos de backend serão encerrados e o sistema PostgreSQL será desligado.



PGDATA: PGDATA é uma variável de ambiente que especifica o local no sistema de arquivos onde os arquivos de dados do PostgreSQL estão armazenados. A pasta PGDATA contém todos os arquivos que o PostgreSQL usa para gerenciar seus bancos de dados, incluindo arquivos de dados, logs de transações, arquivos de configuração e arquivos de log.

WALs (Write-Ahead Logs): O PostgreSQL usa o Write-Ahead Logging (WAL) para manter a integridade dos dados. Antes de qualquer alteração ser feita nos arquivos de dados, ela é registrada no log de transações (WAL). Isso garante que, em caso de falha do sistema, todas as transações podem ser recuperadas. Os WALs são armazenados no diretório pg_wal dentro da pasta PGDATA.

Catálogo do Sistema

O catálogo do sistema no PostgreSQL é um conjunto de tabelas de sistema que mantém informações sobre o banco de dados e seu conteúdo. Essencialmente, é o "banco de dados sobre o banco de dados".

Ele contém metadados sobre todos os objetos no banco de dados, incluindo tabelas, índices, colunas, tipos de dados, usuários, permissões e muito mais. O PostgreSQL usa esses metadados para saber como acessar e manipular os dados.

Os catálogos do sistema estão organizados em schemas especiais. O mais importante deles é o schema **pg_catalog**, que é automaticamente incluído no caminho de pesquisa de todos os usuários e contém as tabelas de sistema mais importantes.

Alguns exemplos de tabelas no **pg_catalog** incluem:

- **pg_class**: contém informações sobre todas as tabelas, índices e sequências.
- **pg_attribute**: contém informações sobre todas as colunas em todas as tabelas.
- **pg_index**: contém informações sobre todos os índices.
- **pg_type**: contém informações sobre todos os tipos de dados.
- **pg_namespace**: contém informações sobre todos os schemas (ou namespaces).
- **pg_roles**: contém informações sobre todos os papéis de usuário.

Por fim, vale ressaltar que enquanto você pode consultar as tabelas do catálogo do sistema, você normalmente não deve tentar alterá-las diretamente. Em vez disso, você deve usar os comandos SQL apropriados para alterar os objetos do banco de dados, que então atualizarão as tabelas do catálogo do sistema como necessário.

Autenticação

A autenticação em PostgreSQL é a primeira etapa na determinação dos direitos de acesso de um cliente a um banco de dados. Quando um cliente tenta estabelecer uma conexão, o PostgreSQL verifica a autenticidade do cliente usando um de vários métodos de autenticação.'

Veja os métodos:

Trust: Esse método permite que a conexão seja feita sem qualquer autenticação. O cliente que solicitou a conexão é confiável e a conexão é automaticamente concedida.

Reject: Este método rejeita automaticamente qualquer tentativa de conexão, independentemente das credenciais.

Scram-sha-256: É um método de autenticação baseado em senha, mais seguro que o md5. Ele usa o método SCRAM-SHA-256, definido no RFC 7677, para criptografar senhas. A partir da versão 10, o PostgreSQL oferece suporte a este método.

MD5: Este método também é baseado em senha. No entanto, ao contrário do método de senha, as senhas não são enviadas como texto simples. Em vez disso, o PostgreSQL criptografa a senha usando o algoritmo MD5.

Password: Este método permite que a conexão seja estabelecida através do fornecimento de uma senha, que é enviada em texto simples. Por causa disso, é menos seguro e deve ser usado apenas em conexões seguras ou confiáveis.

GSS: Este método usa o Generic Security Services Application Program Interface (GSSAPI) para autenticação. Este é um método baseado em Kerberos e só está disponível se o PostgreSQL foi construído com suporte a GSSAPI.

SSPI: Este é um método baseado no Windows que usa a Interface de Programação de Aplicativos de Segurança do Provedor de Suporte de Segurança (SSPI) para autenticação.

Ident: Este método de autenticação funciona verificando se o nome de usuário do sistema operacional do cliente corresponde ao nome de usuário do PostgreSQL solicitado. Este método está disponível apenas para conexões TCP/IP e Unix local.

Peer: Este método é semelhante ao Ident, mas está disponível apenas para conexões locais de Unix. Ele usa o ID de usuário do cliente no sistema operacional para autenticação.

LDAP: Este método permite ao PostgreSQL delegar a autenticação para um servidor LDAP. A senha do usuário é enviada ao servidor LDAP em texto simples, a menos que uma conexão segura (LDAPS) seja configurada.

RADIUS: Com este método, o PostgreSQL envia a senha do usuário para um servidor RADIUS para autenticação.

Cert: Este método usa autenticação SSL certificada pelo cliente. O cliente deve apresentar um certificado SSL válido que seja aceito pelo servidor PostgreSQL.

Método	Descrição
Trust	**Conexão sem autenticação**
Reject	**Rejeita todas as conexões**
Scram-sha-256	**Autenticação baseada em senha com criptografia SCRAM-SHA-256**
MD5	**Autenticação baseada em senha com criptografia MD5**
Password	**Autenticação baseada em senha com senha enviada em texto claro**
GSS	**Autenticação baseada no GSSAPI (método Kerberos)**
SSPI	**Autenticação baseada em Windows SSPI**

Ident	**Autenticação baseada na correspondência do nome de usuário do SO com o do PostgreSQL**
Peer	**Autenticação baseada na correspondência do ID de usuário do Unix**
LDAP	**Autenticação delegada para um servidor LDAP**
RADIUS	**Autenticação delegada para um servidor RADIUS**
Cert	**Autenticação baseada em certificado SSL**

Permissões e Roles

No PostgreSQL, a segurança dos dados é administrada através de um sistema de permissões e roles (papéis). Esta estrutura de controle de acesso permite especificar quem pode fazer o que e com que objetos de banco de dados.

Um **role** no PostgreSQL é um conceito semelhante a um usuário ou grupo em outros sistemas de banco de dados. Cada role tem um nome e pode ter atributos e privilégios associados a ela.

Existem dois tipos de roles:

1. **Role regular:** Age como um usuário que pode se conectar ao banco de dados, possui privilégios e pode ter objetos pertencentes a ele.
2. **Role de grupo:** Não pode se conectar ou possuir objetos, mas outros roles podem ser membros deste. É útil para gerenciar privilégios para um conjunto de roles.

As roles podem ter vários atributos, incluindo, mas não se limitando a, SUPERUSER, CREATEDB e CREATEROLE, que concedem ao role privilégios de superusuário, a capacidade de criar bancos de dados e a capacidade de criar roles, respectivamente.

Permissões são privilégios concedidos a roles. As permissões definem o que uma determinada role pode fazer com um objeto de banco de dados, como uma tabela ou função. Elas são concedidas usando o comando GRANT e revogadas usando o comando REVOKE.

Por exemplo, se você quiser dar a um usuário a permissão para selecionar, inserir, atualizar e excluir linhas em uma tabela, você usaria um comando como este:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON my_table TO my_role;
```

Para revogar essas permissões, você usaria um comando como este:

```
REVOKE SELECT, INSERT, UPDATE, DELETE ON my_table FROM my_role;
```

Existem muitos outros privilégios que você pode conceder ou revogar, incluindo o privilégio de executar funções, acessar tipos de dados, etc.

Tipos de Dados

O PostgreSQL suporta muitos tipos de dados padrão e personalizados. Veja um resumo geral dos tipos de dados mais comuns e importantes:

1. Numérico:

- **INTEGER**: Um número inteiro com um tamanho de 4 bytes.
- **BIGINT**: Um número inteiro com um tamanho de 8 bytes.
- **REAL**: Um número de ponto flutuante com precisão de 6 dígitos decimais.
- **DOUBLE PRECISION**: Um número de ponto flutuante com precisão de 15 dígitos decimais.
- **NUMERIC** ou **DECIMAL**: Um número de precisão exata com uma precisão e escala variável.

Tipo de Dado	Tamanho de Armazenamento	Descrição
smallint	**2 bytes**	**Número inteiro pequeno, -32768 até 32767**
integer	**4 bytes**	**Número inteiro normal, -2147483648 até 2147483647**
bigint	**8 bytes**	**Número inteiro grande, -9223372036854775808 até 9223372036854775807**
decimal	**Variável**	**Número decimal exato, até 131072 dígitos antes do ponto**

		e 16383 dígitos depois do ponto
numeric	**Variável**	**Número decimal exato, até 131072 dígitos antes do ponto e 16383 dígitos depois do ponto**
real	**4 bytes**	**Número decimal aproximado (precisão de ponto flutuante), com precisão de 6 dígitos decimais**
double precision	**8 bytes**	**Número decimal aproximado (precisão de ponto flutuante), com precisão de 15 dígitos decimais**
smallserial	**2 bytes**	**Número inteiro pequeno autoincrementado, 1 até 32767**
serial	**4 bytes**	**Número inteiro normal autoincrementado, 1 até 2147483647**
bigserial	**8 bytes**	**Número inteiro grande autoincrementado, 1 até 9223372036854775807**

2. Binário:

- **BYTEA**: Armazena valores binários como uma sequência de bytes.

3. Booleano:

- **BOOLEAN**: Representa um valor verdadeiro ou falso.

4. Caracteres:

- **CHAR(n)**: Uma cadeia de caracteres de comprimento fixo.
- **VARCHAR(n)**: Uma cadeia de caracteres de comprimento variável.
- **TEXT**: Uma cadeia de caracteres de comprimento variável sem limite de tamanho.

5. Temporal: PostgreSQL possui vários tipos de dados para representar datas e horas:

- **DATE**: Representa uma data (ano, mês, dia).
- **TIME**: Representa uma hora do dia.
- **TIMESTAMP**: Representa uma data e hora.
- **INTERVAL**: Representa uma duração.

6. Enumerado:

- **ENUM**: Este é um tipo de dado definido pelo usuário que representa um conjunto estático de valores.

7. Geométrico: PostgreSQL suporta vários tipos de dados geométricos para representar pontos, linhas e polígonos.

8. JSON:

- **JSON** e **JSONB**: PostgreSQL também suporta tipos de dados para armazenar e manipular JSON. **JSONB** é um formato binário que permite indexação.

9. Arrays:

- PostgreSQL suporta arrays de qualquer tipo de dados, incluindo tipos de dados definidos pelo usuário.

10. UUID:

- **UUID**: Para armazenar Universal Unique Identifiers.

11. Hstore:

- **hstore**: Um tipo de dado para armazenar pares chave-valor.

Funções e Operadores

O PostgreSQL suporta várias funções e operadores que permitem realizar operações complexas nos dados. Eles são categorizados em vários tipos:

1. Funções e operadores matemáticos:

Operações matemáticas básicas, como adição (+), subtração (-), multiplicação (*), divisão (/) e módulo (%), entre outras. Além disso, o PostgreSQL suporta várias funções matemáticas, como **abs**, **round**, **ceil**, **floor**, **sqrt**, **log**, **exp**, **sin**, **cos**, **tan**, e muitos outros.

Função	Descrição
abs(x)	**Retorna o valor absoluto de x**
cbrt(x)	**Retorna a raiz cúbica de x**
ceil(x) ou ceiling (x)	**Retorna o menor número inteiro maior ou igual a x**
degrees(x)	**Converte x de radianos para graus**
div(y,x)	**Divide y por x e retorna um inteiro**
exp(x)	**Retorna o número de Euler (e) elevado à potência x**
floor(x)	**Retorna o maior número inteiro menor ou igual a x**
ln(x)	**Retorna o logaritmo natural de x**
log(x)	**Retorna o logaritmo de base 10 de x**
mod(y, x)	**Retorna o resto da divisão de y por x**
pi()	**Retorna o valor de Pi**
power(a, b)	**Retorna a elevado à potência b**
radians(x)	**Converte x de graus para radianos**
round(x)	**Arredonda x para o número inteiro mais próximo**
sqrt(x)	**Retorna a raiz quadrada de x**
trunc(x)	**Retorna x truncado para um número inteiro**

2. Funções e operadores de string:

Os operadores de string incluem operações como concatenação (||), além de funções como **length**, **substring**, **trim**, **lower**, **upper**, **initcap**, **concat**, **to_char**, **regexp_replace**, entre outros.

Função	Descrição

char_length(string) ou character_length(string)	**Retorna o número de caracteres na string**
lower(string)	**Converte todos os caracteres em string para minúsculas**
upper(string)	**Converte todos os caracteres em string para maiúsculas**
initcap(string)	**Converte a primeira letra de cada palavra em string para maiúscula**
position(substring in string)	**Retorna a posição da primeira ocorrência de substring em string**
length(string)	**Retorna o número de caracteres em string**
replace(string, from, to)	**Substitui todas as ocorrências de from por to em string**
concat(string1, string2, ...)	**Concatena string1, string2, etc.**
concat_ws(separator, string1, string2, ...)	**Concatena string1, string2, etc. com um separador**
split_part(string, delimiter, field)	**Divide string em partes pelo delimiter e retorna a parte número field**
to_char(number, format)	**Converte number para string usando o format especificado**
regexp_replace(source, pattern, replacement [, flags])	**Substitui substrings que correspondem a pattern com replacement em source**

3. Funções e operadores de data e hora:

O PostgreSQL suporta uma variedade de funções e operadores de data e hora, como **current_date**, **current_time**, **now**, **extract**, **date_part**, **age**, além de operadores como **interval**, e muitos outros.

Função	Descrição
now()	Retorna a data e a hora atuais
current_date	Retorna a data atual
current_time	Retorna a hora atual
age(timestamp)	Retorna a diferença entre a data atual e o timestamp fornecido
date_part('field', timestamp)	Extrai o campo 'field' (como 'year', 'month', 'day', 'hour', 'minute', 'second') do timestamp
extract('field' from timestamp)	Mesma função que date_part
date_trunc('field', timestamp)	Trunca o timestamp para o campo 'field' especificado
interval 'quantity unit'	Cria um intervalo de tempo especificado pela quantidade e unidade (como '3 hours', '2 days')
timestamp 'date_string'	Cria um timestamp a partir da string de data fornecida
to_char(timestamp, format)	Converte um timestamp para uma string de acordo com o formato especificado
to_timestamp(text, format)	Converte uma string para um timestamp de acordo com o formato especificado

4. :

O PostgreSQL suporta todos os operadores de comparação padrão, como **=**, **<>** (ou **!=**), **<**, **>**, **<=**, **>=**. Ele também suporta operadores especiais para tipos de dados específicos, como **LIKE** e **ILIKE** para strings, **IS NULL** e **IS NOT NULL** para verificar valores nulos, entre outros.

Operador	Descrição
=	Verifica se dois valores são iguais
<> ou !=	Verifica se dois valores são diferentes
<	Verifica se o valor à esquerda é menor que o valor à direita

>	Verifica se o valor à esquerda é maior que o valor à direita
<=	Verifica se o valor à esquerda é menor ou igual ao valor à direita
>=	Verifica se o valor à esquerda é maior ou igual ao valor à direita
IS NULL	Verifica se o valor é nulo
IS NOT NULL	Verifica se o valor não é nulo
IS TRUE	Verifica se o valor é verdadeiro
IS NOT TRUE	Verifica se o valor não é verdadeiro
IS FALSE	Verifica se o valor é falso
IS NOT FALSE	Verifica se o valor não é falso
LIKE	Compara strings usando caracteres curinga (% para qualquer número de caracteres, _ para um único caractere)
NOT LIKE	Negativa do LIKE
ILIKE	Versão que não distingue maiúsculas de minúsculas do LIKE
NOT ILIKE	Negativa do ILIKE
BETWEEN	Verifica se o valor está dentro de um intervalo
NOT BETWEEN	Verifica se o valor está fora de um intervalo
IN	Verifica se o valor está dentro de um conjunto de valores
NOT IN	Verifica se o valor está fora de um conjunto de valores

5. Funções e operadores condicionais e lógicos:

Os operadores condicionais incluem **CASE**, **COALESCE**, **NULLIF**, enquanto os operadores lógicos incluem **AND**, **OR**, **NOT**.

Operador ou Função	Descrição
AND	Retorna verdadeiro se ambas as expressões forem verdadeiras
OR	Retorna verdadeiro se qualquer uma das expressões for verdadeira
NOT	Inverte o valor de verdade da expressão
CASE WHEN condition THEN result [WHEN ...] [ELSE result] END	Realiza operações condicionais em uma consulta SQL
COALESCE(value1, value2, ...)	Retorna o primeiro valor não nulo na lista
NULLIF(value1, value2)	Retorna nulo se value1 e value2 forem iguais, caso contrário, retorna value1
GREATEST(value1, value2, ...)	Retorna o maior valor na lista
LEAST(value1, value2, ...)	Retorna o menor valor na lista

6. Funções e operadores de agregação:

O PostgreSQL fornece funções de agregação para operações como somatório (**sum**), média (**avg**), mínimo (**min**), máximo (**max**), contagem (**count**), entre outras. Ele também suporta agregações de string, como **string_agg**, e funções de agregação estatística, como **stddev** e **variance**.

Função	Descrição
avg(expression)	Retorna a média de uma expressão numérica
count(expression)	Retorna o número de linhas que a expressão corresponde
count(*)	Retorna o número total de linhas recuperadas
max(expression)	Retorna o valor máximo de uma expressão
min(expression)	Retorna o valor mínimo de uma expressão

sum(expression)	**Retorna a soma de uma expressão numérica**
array_agg(expression)	**Retorna uma matriz de valores que a expressão corresponde**
string_agg(expression, delimiter)	**Retorna uma string que concatena os valores da expressão com um delimitador**
bool_and(expression)	**Retorna verdadeiro se todas as expressões são verdadeiras**
bool_or(expression)	**Retorna verdadeiro se ao menos uma expressão é verdadeira**
json_agg(expression)	**Retorna um JSON que inclui todos os valores da expressão**

7. Funções e operadores de array:

Os operadores de array no PostgreSQL incluem operações como concatenação de arrays (||), busca de elementos (**[]**), e muitos outros. As funções incluem **array_length**, **array_prepend**, **array_append**, entre outros.

Operador ou Função	Descrição
array[valor1, valor2, ...]	**Cria uma matriz com valor1, valor2, etc.**
ARRAY[valor1, valor2, ...]	**Cria uma matriz com valor1, valor2, etc. (equivalente ao anterior)**
valor[índice]	**Acessa o elemento da matriz no índice especificado**
`valor1	
array_append(array, valor)	**Anexa valor ao final de array**
array_prepend(valor, array)	**Anexa valor ao início de array**
array_dims(array)	**Retorna uma string que indica as dimensões do array**
array_length(array, dimensão)	**Retorna o comprimento da array na dimensão especificada**

array_cat(array1, array2)	**Concatena array1 e array2**
array_remove(array, valor)	**Remove todas as ocorrências de valor em array**
array_replace(array, valor1, valor2)	**Substitui todas as ocorrências de valor1 em array por valor2**
unnest(array)	**Expande uma array para um conjunto de linhas**

8. Funções de sistema e de gestão de banco de dados:

Existem também várias funções de sistema e de gestão de banco de dados disponíveis, que permitem, por exemplo, obter informações sobre as configurações do sistema, gerenciar transações, ou obter estatísticas sobre o desempenho do banco de dados.

Terminal psql

O **psql** é um terminal interativo para trabalhar com o PostgreSQL. Ele fornece uma interface em linha de comando para o PostgreSQL que permite ao usuário executar consultas interativas e de script e exibir os resultados em vários formatos.

Veja alguns dos principais recursos do **psql**:

1. Comandos SQL: Você pode digitar qualquer comando SQL e ver os resultados. Isso inclui comandos para criação de tabelas, inserção de dados, atualização de dados, exclusão de dados e muito mais.

2. Comandos do psql: Além dos comandos SQL, o psql também possui seus próprios comandos internos. Esses comandos começam com uma barra invertida (\) e permitem fazer coisas como listar todas as tabelas no banco de dados (**\dt**), listar todos os bancos de dados (**\l**) e conectar a um banco de dados diferente (**\c**).

3. Execução de scripts SQL: Você pode usar o psql para executar scripts SQL armazenados em arquivos. Por exemplo, você pode usar o comando **\i** seguido pelo nome do arquivo para executar um script SQL de um arquivo.

4. Formatação de saída: O psql permite que você controle a formatação dos resultados das consultas. Você pode escolher entre várias opções de formatação, incluindo alinhamento de colunas, bordas de tabelas e muito mais.

5. Ajuda: Se você não tem certeza de como usar um comando específico, o psql oferece uma função de ajuda útil. Basta digitar **\h** seguido pelo nome do comando para ver informações sobre como usar esse comando.

6. Edição de consultas: O psql também possui recursos de edição de linha de comando avançados, incluindo histórico de comandos e edição de linha de comando completa, graças ao suporte à biblioteca readline do GNU.

A tabela abaixo mostra os comandos psql mais utilizados:

Comando	Descrição
\q	**Sai do psql.**
\h	**Fornece ajuda sobre comandos SQL.**
\?	**Lista todos os comandos psql.**
\l	**Lista todos os bancos de dados disponíveis.**
\c _database_	**Conecta-se a outro banco de dados.**
\dt	**Lista todas as tabelas na base de dados atual.**
\d _table_	**Descreve uma tabela.**
\du	**Lista todos os usuários e roles.**
\e	**Abre um editor para escrever uma consulta SQL complexa.**
\s	**Exibe o histórico de comandos.**
\i _file_	**Executa um script SQL de um arquivo.**
\timing	**Liga ou desliga a exibição do tempo de execução dos comandos.**

Comandos DDL e DML (peculiaridades)

No contexto do PostgreSQL, os comandos DDL incluem **CREATE**, **ALTER** e **DROP**, usados para definir ou alterar a estrutura de objetos de banco de dados. Os comandos DML incluem **SELECT**, **INSERT**, **UPDATE** e **DELETE**, usados para manipular os dados dentro dos objetos de banco de dados.

Veja algumas peculiaridades e diferenças do PostgreSQL em relação ao SQL padrão:

1. DDL:

- **Suporte ao tipo de dados SERIAL**: No PostgreSQL, você pode usar o tipo de dados **SERIAL** ao criar uma tabela para gerar automaticamente um número de sequência único para uma coluna.
- **IF EXISTS, IF NOT EXISTS em comandos DDL**: O PostgreSQL permite o uso de **IF EXISTS** e **IF NOT EXISTS** em comandos DDL, como **DROP TABLE IF EXISTS** ou **CREATE TABLE IF NOT EXISTS**. Essa é uma extensão do PostgreSQL e não é suportada em alguns outros sistemas de banco de dados.

2. DML:

- **RETURNING em INSERT, UPDATE e DELETE**: O PostgreSQL permite o uso da cláusula **RETURNING** com **INSERT**, **UPDATE** e **DELETE**, que pode retornar os valores das linhas modificadas.
- **CTEs (Common Table Expressions) com UPDATE e DELETE**: Você pode usar CTEs com **UPDATE** e **DELETE**, o que não é possível em alguns outros sistemas de banco de dados.
- **Suporte para ARRAY**: O PostgreSQL permite a manipulação de arrays como um tipo de dados nativo. Isso significa que você pode usar funções DML em arrays, como adicionar um elemento a um array, remover um elemento, ordenar um array e muito mais.

Por último, é importante notar que o PostgreSQL é um sistema de gerenciamento de banco de dados relacional que adere completamente ao padrão SQL. Isso significa que a maioria dos comandos SQL que você está acostumado a usar em outros sistemas de banco de dados funcionará da mesma forma no PostgreSQL.

Índices e Otimização no PostgreSQL

O PostgreSQL fornece vários métodos de índice, cada um com suas próprias características de desempenho e de armazenamento. Veja os tipos:

- **B-tree**: Esse é o método de indexação padrão. É útil para tipos de dados que possuem uma ordem total bem definida e para operações que precisam desta ordem, como igualdade e comparações de intervalo.
- **Hash**: O índice hash é útil para consultas de igualdade. No entanto, a partir do PostgreSQL 10, os índices hash se tornaram WAL-logados e replicados, então eles não são mais "inseguros" como eram considerados antes.
- **GiST (Generalized Search Tree)**: GiST é um método de indexação genérico que pode ser usado para vários tipos de consultas além de comparações simples, como pesquisas de sobreposição para intervalos e tipos geométricos.
- **SP-GiST (Space Partitioned GiST)**: SP-GiST é útil para tipos de dados que podem ser particionados em um espaço não balanceado, como tipos de dados geométricos, prefixos de texto ou números de telefone.
- **GIN (Generalized Inverted Index)**: O GIN é útil para tipos de dados que possuem vários valores por linha, como arrays e tipos de texto de corpo inteiro.
- **BRIN (Block Range INdexes)**: Os índices BRIN são úteis para tipos de dados que têm uma correlação de ordem linear com a posição física da linha na tabela, como campos de data/hora em registros que são inseridos em ordem cronológica.

Otimização de consultas e planos de execução

PostgreSQL possui um otimizador de consultas sofisticado que tenta escolher o plano de execução mais eficiente para cada consulta com base em estatísticas coletadas sobre a distribuição dos dados nas tabelas. Isso inclui a escolha de que índices usar, em que ordem juntar as tabelas, se deve usar uma junção de hash em vez de uma junção de loop aninhado, etc.

Você pode visualizar o plano de execução de uma consulta usando o comando **EXPLAIN**. Ele mostra detalhes como o custo estimado, o número estimado de linhas e os métodos de junção usados. Com **EXPLAIN ANALYZE**, o PostgreSQL executará a consulta e retornará informações de tempo de execução reais, além do plano.

Por exemplo, vamos supor que temos uma tabela chamada **employees** com os campos **id**, **first_name**, **last_name**, **email**, **hire_date**, e **salary**. E vamos assumir que um índice B-tree foi criado no campo **salary**. Agora, queremos verificar o plano de execução para a seguinte consulta SQL:

```
SELECT * FROM employees WHERE salary > 50000;
```

Você pode usar o comando **EXPLAIN** da seguinte maneira:

```
EXPLAIN SELECT * FROM employees WHERE salary > 50000;
```

Um exemplo de saída pode ser:

```
Seq Scan on employees  (cost=0.00..18334.00 rows=520 width=86)
Filter: (salary > 50000)
```

Significa que:

- **Seq Scan on employees**: O PostgreSQL decidiu fazer uma varredura sequencial na tabela **employees**. Ou seja, está passando por todas as linhas da tabela.
- **cost=0.00..18334.00**: Esta é uma estimativa do custo para executar esta consulta, dada em unidades de "custo de disco". O primeiro número é o custo inicial antes de qualquer linha ser retornada. O segundo número é o custo total para recuperar todas as linhas.
- **rows=520**: Esta é uma estimativa do número de linhas que a consulta irá retornar.
- **width=86**: Esta é uma estimativa da largura média (em bytes) de cada linha retornada.
- **Filter: (salary > 50000)**: Esta é a condição que está sendo aplicada para filtrar as linhas.

Se o PostgreSQL decidisse usar o índice no campo **salary**, a saída do **EXPLAIN** poderia parecer diferente, talvez algo como:

```
Index Scan using employees_salary_idx on employees  (cost=0.29..8.30 rows=1
width=86)

  Index Cond: (salary > 50000)
```

Neste caso, "Index Scan" indica que um índice está sendo usado, e "Index Cond" mostra a condição que está sendo aplicada ao índice.

Ferramentas de monitoramento e diagnóstico

O PostgreSQL fornece várias ferramentas de monitoramento e diagnóstico. Algumas delas são:

- **Logs do sistema**: Os logs do PostgreSQL podem fornecer informações valiosas sobre o desempenho do sistema, erros, conexões de clientes e muito mais.
- **Views de estatísticas**: O PostgreSQL mantém várias views de estatísticas que contêm informações sobre o desempenho do sistema, como **pg_stat_activity** (mostra a atividade atual do servidor), **pg_stat_user_tables** (estatísticas por tabela), **pg_stat_user_indexes** (estatísticas por índice), e muitas outras.

- **pg_stat_statements**: Esta é uma extensão que fornece uma maneira de rastrear as estatísticas de execução de todos os comandos SQL executados por um servidor.
- **Ferramentas de terceiros**: Existem várias ferramentas de monitoramento de terceiros disponíveis para o PostgreSQL, como o pgAdmin, o Postgres Enterprise Manager e o pgBadger, cada uma com seus próprios conjuntos de recursos.

Segurança no PostgreSQL

As políticas de segurança no PostgreSQL podem ser implementadas no nível do objeto de banco de dados, como tabelas e funções, por meio do controle de permissões. Além disso, o PostgreSQL também oferece suporte a políticas de linha de segurança (RLS – Row Level Security), que permitem configurar "políticas" de tabela para restringir, em um nível de linha, se um usuário pode selecionar, inserir, atualizar ou excluir uma linha.

ACL (Access Control Lists)

ACL, ou Lista de Controle de Acesso, é uma lista de permissões associadas a um objeto do banco de dados. Cada objeto possui uma ACL que especifica quais funções do PostgreSQL (basicamente usuários ou grupos de usuários) têm o direito de executar quais tipos de operações naquele objeto.

Por exemplo, você pode ter uma tabela em que um usuário tem permissão para selecionar e inserir dados, enquanto outro usuário tem permissão para selecionar, inserir, atualizar e excluir dados. Essas permissões são armazenadas na ACL da tabela.

Para visualizar as ACLs para tabelas em um banco de dados, você pode consultar a coluna **relacl** na tabela **pg_class**.

RLS (Row-Level Security)

RLS, ou Segurança em Nível de Linha, é um recurso do PostgreSQL que permite implementar políticas de segurança em nível de linha em suas tabelas. Em vez de conceder ou negar permissões em todo o objeto de banco de dados (como uma tabela), a RLS permite que você especifique políticas que podem permitir ou negar o acesso a determinadas linhas dentro dessa tabela com base em qual usuário está realizando a consulta.

Isso é particularmente útil em situações em que você deseja que diferentes usuários tenham acesso a diferentes subconjuntos de dados dentro da mesma tabela. Por exemplo, você pode

ter uma tabela de "pedidos" em que cada usuário deve ser capaz de ver apenas os pedidos que eles próprios fizeram.

As políticas RLS podem ser configuradas para permitir ou negar acesso em operações de seleção, inserção, atualização e exclusão, e as políticas podem avaliar qualquer expressão válida do PostgreSQL para determinar se o acesso deve ser concedido ou não.

Auditoria

A auditoria é outro aspecto importante da segurança, pois permite rastrear atividades de usuários e sistemas para detectar atividades suspeitas e ajudar na solução de problemas. O PostgreSQL oferece vários métodos para auditoria, como logs de consultas, logs de conexão/disconexão, etc. Além disso, há a extensão **pgAudit** que fornece capacidades de auditoria detalhadas para o PostgreSQL.

Veja alguns exemplos de como podemos fazer auditoria no PostgreSQL:

1. **Logs do Sistema PostgreSQL:** O PostgreSQL permite configurar a geração de logs do sistema para registrar diferentes tipos de eventos. Inclui tentativas de conexão, desconexões, instruções executadas, erros ocorridos e muito mais. A geração de log é altamente configurável, permitindo que você defina o nível de detalhamento desejado.
2. **Extensão pgAudit:** A extensão pgAudit fornece capacidades de auditoria detalhadas para o PostgreSQL. Ela fornece uma forma de registrar instruções SQL de classes específicas, tais como leitura, gravação, função, papel e ddl. É uma ferramenta poderosa para implementar auditoria detalhada e estrita.
3. **Triggers:** Os triggers podem ser usados para criar um log de auditoria personalizado no PostgreSQL. Isso poderia envolver a criação de uma tabela de log e a implementação de triggers para registrar certos eventos nessa tabela.

Cada método tem suas próprias vantagens e desvantagens. O uso de logs do sistema ou a extensão pgAudit pode fornecer uma visão mais ampla das atividades, enquanto o uso de triggers pode permitir um controle mais granular e específico do que é registrado. O melhor método depende das necessidades específicas de auditoria.

É importante notar que a auditoria pode ter um impacto na performance do sistema, por isso, deve ser implementada com cuidado. Além disso, os dados de auditoria devem ser protegidos e gerenciados adequadamente, pois podem conter informações sensíveis.

Backup e Recuperação

A proteção dos dados em um sistema de banco de dados é de vital importância para a continuidade dos negócios. O PostgreSQL oferece várias opções de backup e recuperação que podem ser selecionadas com base nas necessidades específicas do sistema.

Backup Lógico

Um backup lógico do PostgreSQL envolve a criação de um arquivo que contém comandos SQL que, quando executados, recriarão o banco de dados do zero. Este backup inclui apenas os dados e a estrutura do banco de dados (tabelas, índices, etc.) e não os arquivos físicos do sistema de arquivos.

As ferramentas **pg_dump** e **pg_dumpall** são comumente usadas para criar backups lógicos no PostgreSQL. **pg_dump** é usado para fazer o backup de um único banco de dados, enquanto **pg_dumpall** é usado para fazer o backup de todo o cluster do PostgreSQL.

Backup Físico

Um backup físico no PostgreSQL é uma cópia dos arquivos de dados no sistema de arquivos. Este backup inclui todo o cluster de bancos de dados e pode ser usado para uma recuperação mais rápida do sistema. O PostgreSQL suporta dois tipos de backup físico: backup base (ou completo) e backup incremental.

Um backup base é uma cópia dos arquivos de dados no momento do backup. Um backup incremental, por outro lado, é uma cópia das alterações feitas nos dados desde o último backup.

pg_dump e pg_restore

pg_dump é uma ferramenta de utilidade que é usada para fazer backup lógico de um banco de dados PostgreSQL. **pg_dump** cria um arquivo que contém comandos SQL para recriar o banco de dados original. O arquivo de backup pode ser em formato de texto simples, ou em formatos binários personalizados ou tar, que permitem salvar blobs (objetos grandes binários).

pg_restore é a ferramenta de utilidade usada para restaurar um banco de dados PostgreSQL a partir de um arquivo de backup criado usando **pg_dump** no formato personalizado ou tar.

pg_restore permite que você realize a restauração em paralelo para acelerar o processo, e também permite que você selecione quais objetos do banco de dados restaurar.

APOSTA ESTRATÉGICA

A ideia desta seção é apresentar os pontos do conteúdo que mais possuem chances de serem cobrados em prova, considerando o histórico de questões da banca em provas de nível semelhante à nossa, bem como as inovações no conteúdo, na legislação e nos entendimentos doutrinários e jurisprudenciais[1].

O controle de concorrência multiversionado (MVCC) no PostgreSQL representa uma abordagem sofisticada para gerenciar acessos simultâneos a um banco de dados. Ao invés de bloquear recursos, o MVCC permite que várias transações ocorram ao mesmo tempo, atribuindo a cada transação uma "visão" do banco de dados no momento em que começou. Isso significa que as transações podem proceder sem esperar que outras terminem, melhorando significativamente o desempenho em ambientes com muitos usuários e minimizando os conflitos de dados. Ao permitir que as transações vejam uma versão consistente do banco de dados, o MVCC evita a necessidade de bloqueios de leitura, o que facilita operações em um sistema altamente concorrente, mantendo ao mesmo tempo a integridade dos dados.

QUESTÕES ESTRATÉGICAS

Nesta seção, apresentamos e comentamos uma amostra de questões objetivas selecionadas estrategicamente: são questões com nível de dificuldade semelhante ao que você deve esperar para a sua prova e que, em conjunto, abordam os principais pontos do assunto.

A ideia, aqui, não é que você fixe o conteúdo por meio de uma bateria extensa de questões, mas que você faça uma boa revisão global do assunto a partir de, relativamente, poucas questões.

---

[1] Vale deixar claro que nem sempre será possível realizar uma aposta estratégica para um determinado assunto, considerando que às vezes não é viável identificar os pontos mais prováveis de serem cobrados a partir de critérios objetivos ou minimamente razoáveis.

1. **(FCC / TRT23-MT - 2016)** São vários os tipos de dados numéricos no PostgreSQL. O tipo:

a) smallint tem tamanho de armazenamento de 1 byte, que permite armazenar a faixa de valores inteiros de −128 a 127.

b) bigint é a escolha usual para números inteiros, pois oferece o melhor equilíbrio entre faixa de valores, tamanho de armazenamento e desempenho.

c) integer tem tamanho de armazenamento de 4 bytes e pode armazenar valores na faixa de −32768 a 32767.

d) numeric pode armazenar números com precisão variável de, no máximo, 100 dígitos.

e) serial é um tipo conveniente para definir colunas identificadoras únicas, semelhante à propriedade autoincremento.

Comentários:

(a) Errado, tem o tamanho de dois bytes; (b) Errado, o **bigint** é utilizado quando o range do **integer** é insuficiente – ele é mais utilizado para representar inteiros gigantescos, logo não é a escolha usual para números inteiros; (c) Errado, ele armazena um range de -2147483648 to +2147483647; (d) Errado, é até 131072 dígitos antes do ponto decimal e até 16383 dígitos após o ponto decimal; (e) Correto, trata-se de um inteiro de quatro bytes autoincrementável usado para definir colunas identificadoras.

Gabarito: Letra E

2. **(FCC / TRE-PB - 2015)** No PostgreSQL, o tipo de dados numérico considerado meramente uma notação conveniente para definir colunas identificadoras únicas, semelhante à propriedade auto incremento em alguns Sistemas Gerenciadores de Banco de Dados, é o tipo:

a) serial.

b) smallint.

c) byte.

d) bit.

e) blob.

Comentários:

serial	Inteiro de quatro bytes autoincrementável.
smallint	Inteiro de dois bytes com sinal.
bit	Cadeia de bits de tamanho fixo.

Os tipos **serial** e **smallserial** não são tipos verdadeiros – são meramente uma conveniência de notação para criar colunas de identificador exclusivo (semelhante ao tipo **AUTO_INCREMENT** suportada por alguns outros bancos de dados). Não existem os tipos **byte** e **blob**.

Gabarito: Letra A

3. **(FCC / TRT 13PB - 2014)** O SQL define dois tipos primários para caracteres: A e B. Estes tipos podem armazenar cadeias de caracteres com comprimento de até n caracteres, onde n é um número inteiro positivo. A tentativa de armazenar uma cadeia de caracteres mais longa em uma coluna de um destes tipos resulta em erro, a não ser que os caracteres excedentes sejam todos espaços; neste caso, a cadeia de caracteres será truncada em seu comprimento máximo. Se a cadeia de caracteres a ser armazenada for mais curta que o comprimento declarado, os valores do tipo B são completados com espaços; os valores do tipo A simplesmente armazenam a cadeia de caracteres mais curta.

Os tipos de dados destinados a armazenar cadeias de caracteres no PostgreSQL 8.0, descritos como A e B no texto acima são, respectivamente,

a) char(n) e character varying(n).

b) character varying(n) e character(n).

c) character(n) e char(n).

d) varchar(n) e text(n).

e) character varying(n) e varchar(n).

Comentários:

(a) **char(n)** e **character varying(n)**.

(b) **character varying(n)** e **character(n)**.

(c) **character(n)** e **char(n)**.

(d) **varchar(n)** e **text(n)**.

(e) **character varying(n)** e **varchar(n)**.

Gabarito: Letra B

4. **(FCC / TRF1 - 2014)** O PostgreSQL disponibiliza para os usuários um amplo conjunto de tipos de dados nativos. Dentre os tipos para data e hora estão: time, timestamp, date e:

a) bigdate.

b) datetime.

c) datetimeoffset.

d) smalldatetime.

e) interval.

Comentários:

A letra E representa um intervalo de tempo. Os outros tipos não existem.

Gabarito: Letra E

5. **(FCC / TCE-RS – 2014)** O sistema gerenciador de bancos de dados PostgreSQL 9.1 admite diversos tipos de dados. Dentre eles, o tipo de dados:

a) *integer* requer 8 bytes para seu armazenamento.

b) *boolean* admite até 3 valores distintos.

c) money requer 4 bytes para seu armazenamento.

d) para armazenamento de endereços IPv4 não está disponível.

e) *smallint* requer 2 bytes para seu armazenamento.

Comentários:

integer	**Inteiro de quatro bytes com sinal.**
boolean	**Booleano lógico (true/false).**
money	**Montante de dinheiro com 8 bytes.**
smallint	**Inteiro de dois bytes com sinal.**
cidr	**Endereço de rede IPv4 ou IPv6.**

(a) Errado, requer 4 bytes; (b) Errado, admite apenas dois valores distintos; (c) Errado, requer 8 bytes; (d) Errado, está disponível por meio do tipo cidr; (e) Correto.

Gabarito: Letra E

QUESTIONÁRIO DE REVISÃO E APERFEIÇOAMENTO

A ideia do questionário é elevar o nível da sua compreensão no assunto e, ao mesmo tempo, proporcionar uma outra forma de revisão de pontos importantes do conteúdo, a partir de perguntas que exigem respostas subjetivas.

São questões um pouco mais desafiadoras, porque a redação de seu enunciado não ajuda na sua resolução, como ocorre nas clássicas questões objetivas.

O objetivo é que você realize uma auto explicação mental de alguns pontos do conteúdo, para consolidar melhor o que aprendeu ;)

Além disso, as questões objetivas, em regra, abordam pontos isolados de um dado assunto. Assim, ao resolver várias questões objetivas, o candidato acaba memorizando pontos isolados do conteúdo, mas muitas vezes acaba não entendendo como esses pontos se conectam.

Assim, no questionário, buscaremos trazer também situações que ajudem você a conectar melhor os diversos pontos do conteúdo, na medida do possível.

É importante frisar que não estamos adentrando em um nível de profundidade maior que o exigido na sua prova, mas apenas permitindo que você compreenda melhor o assunto de modo a facilitar a resolução de questões objetivas típicas de concursos, ok?

Nosso compromisso é proporcionar a você uma revisão de alto nível!

Vamos ao nosso questionário:

Perguntas

1. O que é PostgreSQL e quais são algumas de suas principais características?
2. O que é o Postmaster no PostgreSQL?
3. O que é Write-Ahead Logging (WAL) e por que é importante?
4. Explique o que é um Savepoint em PostgreSQL.
5. O que é o PGDATA?
6. O que é o pg_catalog?
7. Explique o que é ACL no PostgreSQL.
8. O que é RLS no PostgreSQL e para que serve?
9. Qual é a diferença entre backup lógico e backup físico no PostgreSQL?
10. Quais são as funções das ferramentas pg_dump e pg_restore?
11. Qual é a função da ferramenta psql?
12. O que é o controle de concorrência multiversionado (MVCC) no PostgreSQL?
13. Como o PostgreSQL lida com a autenticação de usuários?
14. Quais são alguns dos principais tipos de dados suportados pelo PostgreSQL?
15. Quais são algumas das características de segurança disponíveis no PostgreSQL?
16. Quais são os diferentes tipos de índices suportados pelo PostgreSQL?
17. Como o comando EXPLAIN pode ajudar na otimização de consultas?
18. O que é o Namespace no contexto do PostgreSQL?
19. Qual é a diferença entre as funções e os operadores no PostgreSQL?
20. O que é pgAudit e por que é útil?

Perguntas e Respostas

1. O que é PostgreSQL e quais são algumas de suas principais características?

   Resposta: PostgreSQL é um sistema de gerenciamento de banco de dados relacional de código aberto, avançado e rico em recursos. Suas características incluem suporte para transações ACID, extensibilidade, integridade referencial, controle de concorrência multiversionado (MVCC), entre outras.

2. O que é o Postmaster no PostgreSQL?

Resposta: O Postmaster é o processo principal no PostgreSQL que gerencia conexões de cliente, inicia processos de servidor filho para lidar com essas conexões e realiza tarefas de manutenção do sistema.

3. O que é Write-Ahead Logging (WAL) e por que é importante?

   Resposta: WAL é a técnica usada pelo PostgreSQL para garantir a integridade dos dados, registrando todas as modificações em um log antes de confirmar a transação que a causou.

4. Explique o que é um Savepoint em PostgreSQL.

   Resposta: Um savepoint em PostgreSQL é um ponto na transação que você pode rolar para trás, sem precisar abortar toda a transação.

5. O que é o PGDATA?

   Resposta: PGDATA é uma variável de ambiente que define o local do diretório de dados do PostgreSQL.

6. O que é o pg_catalog?

   Resposta: O pg_catalog é um esquema especial que contém as tabelas do sistema, que são tabelas construídas internamente pelo PostgreSQL para o seu funcionamento.

7. Explique o que é ACL no PostgreSQL.

   Resposta: ACL (Access Control List) em PostgreSQL são usadas para definir permissões em diferentes objetos do banco de dados.

8. O que é RLS no PostgreSQL e para que serve?

   Resposta: RLS (Row Level Security) é uma funcionalidade do PostgreSQL que permite controlar quais usuários podem selecionar, inserir, atualizar ou excluir quais linhas de uma tabela baseada em políticas definidas.

9. Qual é a diferença entre backup lógico e backup físico no PostgreSQL?

   Resposta: O backup lógico em PostgreSQL envolve a exportação dos dados do banco de dados em um formato que pode ser lido por SQL, enquanto o backup físico envolve copiar os arquivos físicos que compõem o banco de dados.

10. Quais são as funções das ferramentas pg_dump e pg_restore?

Resposta: Pg_dump é uma ferramenta de linha de comando usada para fazer backup de um banco de dados PostgreSQL, enquanto pg_restore é usada para restaurar um banco de dados PostgreSQL a partir de um arquivo de backup criado pelo pg_dump.

11. Qual é a função da ferramenta psql?

    Resposta: Psql é a interface de linha de comando para o PostgreSQL. É usado para interagir com o servidor e executar operações de banco de dados.

12. O que é o controle de concorrência multiversionado (MVCC) no PostgreSQL?

    Resposta: MVCC é o método que o PostgreSQL usa para lidar com a concorrência de dados, permitindo que várias transações leiam e escrevam sem interferir umas nas outras.

13. Como o PostgreSQL lida com a autenticação de usuários?

    Resposta: PostgreSQL oferece vários métodos de autenticação, incluindo senha, md5, scram-sha-256, ident, e cert. Cada método tem suas próprias vantagens e pode ser escolhido de acordo com as necessidades de segurança.

14. Quais são alguns dos principais tipos de dados suportados pelo PostgreSQL?

    Resposta: PostgreSQL suporta uma ampla gama de tipos de dados, incluindo numéricos, strings, booleanos, datas e horas, e tipos geométricos, além de permitir a definição de tipos personalizados.

15. Quais são algumas das características de segurança disponíveis no PostgreSQL?

    Resposta: PostgreSQL oferece vários recursos de segurança, incluindo controle de acesso baseado em função, segurança em nível de linha (RLS), criptografia de dados, autenticação forte e auditoria detalhada.

16. Quais são os diferentes tipos de índices suportados pelo PostgreSQL?

    Resposta: PostgreSQL suporta vários tipos de índices, incluindo B-tree, Hash, GiST, SP-GiST, GIN, e BRIN. Cada tipo de índice é útil para diferentes tipos de consultas e tipos de dados.

17. Como o comando EXPLAIN pode ajudar na otimização de consultas?

    Resposta: O comando EXPLAIN no PostgreSQL fornece um plano de execução de uma consulta, que mostra como o sistema de banco de dados pretende executar a consulta. Isso pode ajudar a entender por que uma consulta pode estar lenta e onde otimizar.

18. O que é o Namespace no contexto do PostgreSQL?

    Resposta: Em PostgreSQL, um namespace é um contêiner que contém um conjunto nomeado de objetos, como tabelas e funções. O termo é usado de maneira intercambiável com "schema".

19. Qual é a diferença entre as funções e os operadores no PostgreSQL?

    Resposta: As funções no PostgreSQL são rotinas que aceitam parâmetros, executam operações e retornam o resultado da operação como um valor. Por outro lado, os operadores são símbolos especiais que representam operações específicas entre valores, como adição, subtração, multiplicação, divisão, etc.

20. O que é pgAudit e por que é útil?

    Resposta: PgAudit é uma extensão do PostgreSQL que fornece capacidades de auditoria detalhadas para atividades de banco de dados. Isso é útil para fins de conformidade e para entender e analisar atividades no banco de dados.

LISTA DE QUESTÕES ESTRATÉGICAS

1. **(FCC / TRT23-MT - 2016)** São vários os tipos de dados numéricos no PostgreSQL. O tipo:

   a) smallint tem tamanho de armazenamento de 1 byte, que permite armazenar a faixa de valores inteiros de −128 a 127.

   b) bigint é a escolha usual para números inteiros, pois oferece o melhor equilíbrio entre faixa de valores, tamanho de armazenamento e desempenho.

   c) integer tem tamanho de armazenamento de 4 bytes e pode armazenar valores na faixa de −32768 a 32767.

   d) numeric pode armazenar números com precisão variável de, no máximo, 100 dígitos.

   e) serial é um tipo conveniente para definir colunas identificadoras únicas, semelhante à propriedade autoincremento.

2. **(FCC / TRE-PB - 2015)** No PostgreSQL, o tipo de dados numérico considerado meramente uma notação conveniente para definir colunas identificadoras únicas, semelhante à propriedade auto incremento em alguns Sistemas Gerenciadores de Banco de Dados, é o tipo:

a) serial.

b) smallint.

c) byte.

d) bit.

e) blob.

3. **(FCC / TRT 13PB - 2014)** O SQL define dois tipos primários para caracteres: A e B. Estes tipos podem armazenar cadeias de caracteres com comprimento de até n caracteres, onde n é um número inteiro positivo. A tentativa de armazenar uma cadeia de caracteres mais longa em uma coluna de um destes tipos resulta em erro, a não ser que os caracteres excedentes sejam todos espaços; neste caso, a cadeia de caracteres será truncada em seu comprimento máximo. Se a cadeia de caracteres a ser armazenada for mais curta que o comprimento declarado, os valores do tipo B são completados com espaços; os valores do tipo A simplesmente armazenam a cadeia de caracteres mais curta.

   Os tipos de dados destinados a armazenar cadeias de caracteres no PostgreSQL 8.0, descritos como A e B no texto acima são, respectivamente,

   a) char(n) e character varying(n).

   b) character varying(n) e character(n).

   c) character(n) e char(n).

   d) varchar(n) e text(n).

   e) character varying(n) e varchar(n).

4. **(FCC / TRF1 - 2014)** O PostgreSQL disponibiliza para os usuários um amplo conjunto de tipos de dados nativos. Dentre os tipos para data e hora estão: time, timestamp, date e:

   a) bigdate.

   b) datetime.

   c) datetimeoffset.

d) smalldatetime.

e) interval.

5. **(FCC / TCE-RS – 2014)** O sistema gerenciador de bancos de dados PostgreSQL 9.1 admite diversos tipos de dados. Dentre eles, o tipo de dados:

a) *integer* requer 8 bytes para seu armazenamento.

b) *boolean* admite até 3 valores distintos.

c) money requer 4 bytes para seu armazenamento.

d) para armazenamento de endereços IPv4 não está disponível.

e) *smallint* requer 2 bytes para seu armazenamento.

Gabaritos

1. E
2. A
3. B
4. E
5. E

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.