Introdução ao SQL

O que é SQL (Structured Query Language)?

- SQL é uma linguagem de programação usada para garantir e manipular bancos de dados relacionais. Porem aos usuários criar, modificar, deletar e extrair dados de bancos de dados, além de gerenciar estruturas de banco de dados e controlar o acesso aos dados.

- A SQL é crucial no mundo dos bancos de dados pois facilita a gestão e manipulação de dados em sistemas relacionais. Permite aos usuários executar consultas complexas de maneira eficiente, garantir a integridade dos dados e estabelecer controle de acesso seguro aos dados, sendo uma habilidade essencial para profissionais de TI e análise de dados.

Operações Básicas do SQL

```
CREATE TABLE jovem_aprendiz (
        id SERIAL PRIMARY KEY,
        nome VARCHAR (255) NOT NULL,
        idade INT NOT NULL,
        departamento VARCHAR (255)
);
```

```
INSERT INTO jovem_aprendiz
(nome, idade, departamento)
VALUES ('Ana Silva',18,'Recursos Humanos'),
       ('João Souza',19,'Financeiro'),
       ('Marta Oliveira',20,'Marketing');
```

```
SELECT * FROM jovem_aprendiz;
```

```
UPDATE jovem_aprendiz
SET departamento='TI'
WHERE nome='Ana Silva';
```

```
DELETE FROM jovem_aprendiz
WHERE nome='Marta Oliveira';
```

Funções e Operadores no SQL

O SQL fornece uma ampla gama de funções e operadores que ajudam na manipulação e análise de dados armazenados em bancos de dados relacionais. Abaixo estão algumas das categorias mais importantes.

Funções de Agregação:

Funções de agregação são usadas para computar valores derivados de colunas em um conjunto de linhas, fornecendo insights estatísticos úteis.

- COUNT: Conta o número de linhas.
- SUM: Soma os valores de uma coluna.
- AVG: Calcula a média dos valores de uma coluna.
- MAX e MIN: Retorna o maior e o maior valor de uma coluna, respectivamente.

- Exemplo:

SELECT COUNT(*) AS total, AVG(idade) AS media_idade FROM jovem_aprendiz;

Operadores Lógicos:

- Operadores lógicos são utilizados para comparar valores e fazer controle condicional dentro das queries.
- AND: Verifica se ambas as condições são verdadeiras.
- OR: Verifica se pelo menos uma das condições é verdadeira.
- NOT: Nega a condição fornecida.

- Exemplo:

SELECT * FROM jovem_aprendiz WHERE idade >=18 AND departamento = ‘Marketing’;

Operadores de Comparação

Operadores de comparação são usados para comparar valores.

- =: Igual a;
- != ou <>; Diferente de;
- <: Menor que;
- >: Maior que;
- <=: Menor ou igual a.
- >=: Maior ou igual a.

- Exemplo:
SELECT * FROM jovem_aprendiz WHERE idade >=20;

Funções de Data e Hora:

Funções de data e hora ajudam a manipular e formatar valores de data e hora.

- NOW(): Retorna a data e hora atuais;
- DATE_PART(): Extrai uma parte da data ou hora;
- AGE(): Calcula a idade baseada em datas.

- Exemplo:
SELECT NOW(), AGE(NOW(), data_nascimento) FROM jovem_aprendiz;

Postgresql

O POSTGRESQL É UM SISTEMA DE GERENCIAMENTO DE BANCO DE DADOS RELACIONAL DE CÓDIGO ABERTO, SUA ORIGEM REMONTA A 1986 COMO PARTE DO PROJETO POSTRES NA UNIVERSIDADE DA CALIFÓRNIA EM BERKELEY. MICHAEL STONEBRAKER, QUE TAMBÉM ESTAVA ENVOLVIDO NO DESENVOLVIMENTO INICIAL DO INGRES, LIDEROU ESTE PROJETO. O POSTRES TORNOU-SE POSTGRESQL EM 1996 COM A ADIÇÃO DE SUPORTE AO PADRÃO SQL.

Características Princípais

- Código Aberto;
- Suporta propriedades ACID (atomicity, consistency, isolation, durability) Garantindo transações seguras e confiáveis
- Oferece suporte a definiçõa de tipos de dados customizados, permitindo que os usuários definam seus próprios tipos de dados;
- É ALTAMENTE EXTENSÍVEL, PERMITINDO QUE OS USUÁRTOS ADICIOMEM NOVAS FUNÇÕES, OPERADORES, TIPOS DE DADOS E MUITO MAIS.
- INDEXAÇÃO E OTIMIZAÇÃO DE CONSULTAS: - POSSUI UM PLANEJADOR DE CONSULTAS SOFISTICADO E SUPORTE A INDEXAÇÃO AVANÇADA, O QUE AJUDA A OTIMIZAR O DESEMPENHO DAS CONSULTAS,

- SUPORTE A LINGUAGENS DE PROCEDIMENTOS: - OFERECE SUPORTE A VÁRIAS LINGUAGENS DE PROCEDIMENTOS, INCLUINDO PL/PGSOL, PL/PYTHON, PL/PERL E PL/TCL.

- CONFORMIDADE COM SQL: - OFERECE ALTA CONFORMIDADE COM OS PADRÕES SOL, FACILITANDO A MIGRAÇÃO DE OUTROS SISTEMAS DE BANCO DE DADOS RELACIONAL.

- RECURSOS DE SEGURANÇA: - INCLUI VÁRIOS RECURSOS DE SEGURANCA, COMO CONTROLE DE ACESSO BASEADO EM FUNÇÃO, CRIPTOGRAFIA DE DADOS E AUTENTICAÇÃO DE CERTIFICADO SSL.

- SUPORTE A OBJETOS E JSON: - ALEM DE SER UM ROBNS, POSSUI SUPORTE PARA ARMAZENAMENTO E MANIPULAÇÃO DE OBJETOS E DADOS JSON, PERMITINDO FLEXIBILIDADE NA GESTÃO DE DADOS,

- PARTICIONAMENTO DE TABELA: - OFERECE SUPORTE A PARTICIONAMENTO DE TABELAS, AJUDANDO NA ORGANIZAÇÃO DE GRANDES CONJUNTOS DE DADOS E MELHORANDO O DESEMPENHO,

- REPLICAÇAO E BACKUP: - SUPORTE A REPLICAÇÃO SÍNCRONA E ASSÍNCRONA, ALÉM DE ROBUSTAS SOLUÇÕES DE BACKUP E RECUPERAÇÃO.

TIPOS DE DADOS ESPECIAIS EM POSTGRESQL

- JSON -> JSON (JavaScript Object Notation). Mantém o formato original do texto JSON

- JSONB -> Uma versão binárla do tipo JSON. Permite indexação e é mais eficiente em operações de busca e atualização. Embora as inserções sejam um pouco mais lentas do que o tipo JSON

- HSTORE • tipo de dado "chave-valor". É átil para armazenar coleções de pares chave-valor em uma única coluna PostgreSQL

- Arays -› você armazene arrays de qualquer tipo de dado base, incluindo números, strings e outros tipos de dados

- UUID -> Usado para armazenar identificadores universais únicos.

- Range Types -› Representam uma fala de valores.

- Geometric Types -> Permitem representar pontos, linhas, polígonos, etc., para operações geométricas.

- Network Address -› Especializado no armazenamento de endereços IP e CIDR.

- Bit Strings -> Usados para armazenar strings de bits.

ÍNDICES E PERFORMANCE

Índices são estruturas de dados que permitem uma busca mais rápida por registros em uma tabela, funcionando como uma espécie de "mapa" que aponta diretamente para os locais onde os dados estão armazenados

- Melhoria na Velocidade de Consulta;
- Eficiência em Operações de JOIN;
- Ordenação e Agrupamento;
- Restrições de Unicidade;
- Manutenção de Índices.

Imagine que você tem uma planilha Excel enorme com muitas colunas e linhas, repleta de dados variados. Agora, suponha que você precise encontrar informações específicas nessa planilha. Uma abordagem seria percorrer cada linha e cada coluna até encontrar o que procura, mas isso seria muito demorado e ineficiente.

Agora, pense na função de "filtro" no Excel, onde você pode escolher uma coluna específica e aplicar um filtro para mostrar apenas as linhas que correspondem a certo critério. Ao fazer isso, você elimina uma grande quantidade de dados irrelevantes e foca apenas nos dados que são Importantes para sua busca, agilizando significativamente o processo de encontrar o que precisa.

Essa funcionalidade de filtragem no Excel é semelhante à maneira como os Índices funcionam em um banco de dados. Um Índice em um banco de dados é como um filtro que é aplicado a uma coluna específica (ou conjunto de colunas) para acelerar as buscas. Ao invés de percorrer todas as linhas de uma tabela, o banco de dados pode usar o índice para ir diretamente às linhas que satisfazem a consulta, economizando tempo e recursos computacionais.

Pontos de reflexão

Join

"Join" é um termo comumente utilizado em bancos de dados e linguagens de consulta, como SQL (Structured Query Language), para combinar dados de duas ou mais tabelas com base em uma condição relacionada entre elas. Quando você utiliza a cláusula "JOIN" em uma consulta SQL, você está instruindo o banco de dados a combinar registros de diferentes tabelas com base em uma chave comum, criando assim um conjunto de resultados mais completo e significativo.

Existem vários tipos de joins, incluindo:

- **INNER JOIN:** Retorna apenas os registros que têm correspondência em ambas as tabelas. Ou seja, ele retorna apenas as linhas que têm chaves correspondentes em ambas as tabelas.

Exemplo de SQL:

```
SELECT tabela1.coluna, tabela2.coluna
FROM tabela1
INNER JOIN tabela2
ON tabela1.chave = tabela2.chave;
```

- **LEFT (OUTER) JOIN:** Retorna todos os registros da tabela à esquerda (a primeira tabela mencionada) e os registros correspondentes da tabela à direita. Se não houver correspondência, os resultados da tabela à direita conterão valores nulos.

Exemplo de SQL:

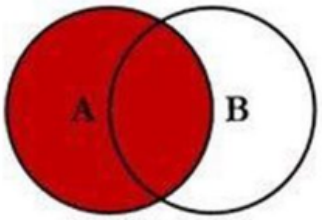
```
SELECT tabela1.coluna, tabela2.coluna
FROM tabela1
LEFT JOIN tabela2
ON tabela1.chave = tabela2.chave;
```

- **RIGHT (OUTER) JOIN:** Retorna todos os registros da tabela à direita (a segunda tabela mencionada) e os registros correspondentes da tabela à esquerda. Se não houver correspondência, os resultados da tabela à esquerda conterão valores nulos. Alguns bancos de dados também suportam FULL OUTER JOIN, que retorna registros quando há uma correspondência em uma das tabelas.

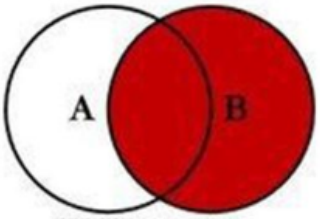
Exemplo de SQL (RIGHT JOIN):

```
SELECT tabela1.coluna, tabela2.coluna
FROM tabela1
RIGHT JOIN tabela2
ON tabela1.chave = tabela2.chave;
```

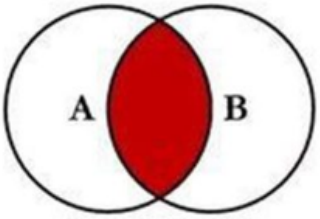
SQL JOINS



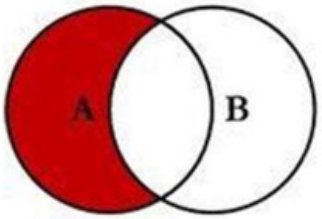

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



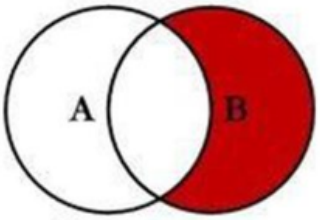

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



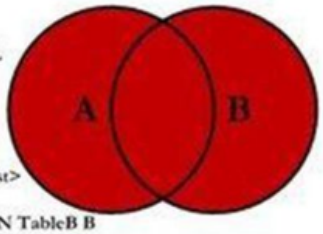

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



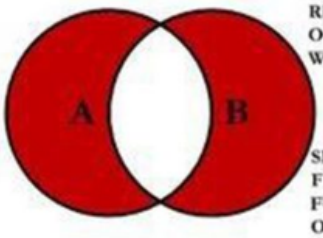

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```