WASHINGTON LUIZ PERONI

MACHINE LEARNING ENGINEERING

BIG DATA CLOUD PLATFORMS

AULA 02

SUMÁRIO

O QUE VEM POR AÍ?

Olá, pessoal! Boas-vindas à aula sobre “AWS Athena”, uma poderosa ferramenta de análise de dados. O Athena nos traz a praticidade de poder fazer queries SQL em nosso data Lake e analisar peta bytes de dados em tempo real. Também podemos fazer ETL entre tabelas e salvar o conteúdo diretamente no S3, utilizando seu comando “UNLOAD”. A AWS define o Athena da seguinte forma:

> “O Amazon Athena é um serviço de análise interativo e sem servidor criado em frameworks de código aberto, com suporte a formatos de tabela e arquivo abertos. O Athena fornece uma maneira simplificada e flexível de analisar petabytes de dados onde eles residem. Analise dados ou crie aplicações a partir de um data lake do Amazon Simple Storage Service (S3) e mais de 30 fontes de dados, incluindo fontes de dados on-premises ou outros sistemas em nuvem, usando SQL ou Python. O Athena é construído com mecanismos Trino e Presto de código aberto e frameworks Apache Spark, sem necessidade de provisionamento ou configuração.” (AWS, [s.d.], documento on-line)

Demostraremos alguns casos de uso em conjunto com outros serviços, como Glue e S3, para que vocês possam aplicar os conhecimentos adquiridos no mundo real.

HANDS ON

Demonstraremos na prática, dentro do console da AWS, o uso do Athena, mostrando desde a tela inicial e os conceitos básicos, até um caso de uso real de integração com Glue e S3, hoje o Data Lake está presente na maioria das empresas. Dividiremos o conteúdo nas seguintes etapas:

- AWS Athena via console, configuração inicial Workgroup.
- Visualização das Tabelas do Lake via AWS Athena.
- Queries SQL básicas (tabelas).
- Integração com AWS Glue, consumo e análise em tabelas.
- Queries SQL Complexas, Join entre tabelas.
- ETL com Athena, comando unload e integração com S3.

SAIBA MAIS

Histórico AWS Athena:

O Amazon Athena é um serviço de consulta interativa que permite analisar dados diretamente no Amazon S3 usando SQL padrão. Lançado pela AWS em 2016, o Athena simplifica a análise de grandes conjuntos de dados sem a necessidade de provisionar ou gerenciar infraestrutura. Ele utiliza o PrestoDB, um mecanismo de consulta distribuído de código aberto, para executar consultas SQL diretamente nos dados armazenados no S3.

Integração com S3 e Glue

O Athena integra-se perfeitamente com o Amazon S3 e o AWS Glue. Os dados no S3 são consultados diretamente, sem a necessidade de mover ou transformar os dados antes da análise. O AWS Glue pode ser utilizado para catalogar e organizar os metadados dos dados no S3, facilitando o entendimento e a descoberta de dados para os usuários do Athena. A integração com Glue também permite a criação de tabelas externas no Athena, proporcionando um esquema estruturado para os dados no S3.

Em resumo, o Athena fornece uma camada de consulta conveniente sobre os dados no S3, aproveitando a escalabilidade e a flexibilidade do PrestoDB, enquanto a integração com o Glue facilita a organização e catalogação dos dados armazenados no S3.

PrestoDB

O PrestoDB é um sistema de consultas distribuídas de código aberto, projetado para consultas analíticas em grande escala. Foi desenvolvido pelo Facebook e lançado como um projeto de código aberto em 2012. O PrestoDB foi projetado para fornecer consultas SQL rápidas e interativas em grandes conjuntos de dados distribuídos.

O projeto foi inicialmente liderado pelo Facebook, mas em 2013, a empresa transferiu o controle do projeto para a Fundação Presto Software, uma organização independente sem fins lucrativos. Desde então, o PrestoDB atraiu uma comunidade ativa de desenvolvedores e usuários, sendo adotado por várias empresas para análise de dados em escala.

O PrestoDB é conhecido por sua capacidade de consulta em tempo real, suporte a SQL e arquitetura distribuída, permitindo que consultas sejam distribuídas e processadas em vários nós. Essas características fazem do PrestoDB uma ferramenta popular para organizações que lidam com grandes volumes de dados e que exigem análises rápidas e eficientes.

O PrestoDB é frequentemente comparado a sistemas de bancos de dados tradicionais, como o MySQL ou o PostgreSQL. Vamos explorar algumas diferenças, exemplos práticos, vantagens e desvantagens.

Diferenças entre PrestoDB e sistemas de banco de dados comuns

Arquitetura distribuída: o PrestoDB é projetado para ambientes distribuídos, permitindo que consultas sejam processadas em paralelo em vários nós. Isso contrasta com os bancos de dados tradicionais, que muitas vezes operam em um único nó.

Consulta em tempo real: o PrestoDB é otimizado para consultas analíticas em tempo real, fornecendo respostas rápidas a consultas complexas. Bancos de dados tradicionais podem não ser tão eficientes para análises em larga escala.

Principais vantagens do PrestoDB

Desempenho em escala: oferece desempenho eficiente para consultas em grandes conjuntos de dados distribuídos.

Suporte a SQL completo: suporta a maioria das instruções SQL padrão, facilitando a transição para usuários familiarizados com SQL.

Integração com diversas fontes de dados: pode consultar dados de várias fontes, como Hadoop, Amazon S3, MySQL etc., facilitando a integração com ambientes heterogêneos.

Principais desvantagens do PrestoDB

Consistência fraca: o PrestoDB pode operar com consistência eventual, o que pode ser uma desvantagem em casos em que a consistência forte é crucial.

Complexidade de configuração: configurar e otimizar um ambiente distribuído pode ser complexo, exigindo conhecimento técnico.

SQL (Structured Query Language)

O SQL ou Linguagem de Consulta Estruturada é uma linguagem de programação específica para gerenciar e manipular bancos de dados relacionais. Ele foi desenvolvido pela IBM nos anos 1970 e padronizado pela primeira vez em 1986. Desde então, tornou-se uma linguagem fundamental para interação com sistemas de gerenciamento de banco de dados relacionais (SGBDR).

Principais vantagens do SQL

Facilidade de uso: SQL é uma linguagem de fácil compreensão, com uma sintaxe declarativa que permite aos usuários expressar consultas de maneira intuitiva.

Independência de dados: os comandos SQL são projetados para serem independentes do tipo de banco de dados, oferecendo portabilidade e flexibilidade na manipulação de dados.

Segurança e controle de acesso: SQL fornece recursos robustos para controlar o acesso aos dados, garantindo a segurança e a integridade das informações armazenadas.

Consistência de dados: SQL garante a consistência dos dados por meio de restrições de integridade, como chaves primárias e estrangeiras, o que contribui para a qualidade e confiabilidade dos dados armazenados.

Suporte a transações: o SQL oferece suporte a transações, permitindo a execução de operações complexas de maneira atômica, consistente, isolada e durável (ACID), o que é crucial para a integridade dos dados em aplicações críticas.

Comunidade ativa e recursos on-line: a vasta comunidade de usuários do SQL e a abundância de recursos on-line, como fóruns e tutoriais, facilitam o aprendizado e resolução de problemas.

Principais desvantagens do SQL:

Complexidade em operações avançadas: algumas operações mais avançadas podem se tornar complexas de expressar eficientemente em SQL, levando a consultas extensas ou dificuldade na otimização do desempenho.

Limitações de desempenho em grandes volumes de dados: em situações de grandes volumes de dados, consultas SQL complexas podem impactar o desempenho do sistema, exigindo otimizações cuidadosas.

Escalabilidade horizontal limitada: em sistemas distribuídos ou de grande escala, a escalabilidade horizontal (acréscimo de servidores) pode ser desafiador implementar efetivamente usando apenas SQL.

Dificuldade em representar modelos hierárquicos ou de rede: modelos de dados hierárquicos ou de rede podem ser desafiadores para representar eficientemente em SQL, pois a linguagem foi originalmente projetada para lidar com modelos relacionais.

Necessidade de conhecimento profundo para otimização: otimizar consultas SQL para melhorar o desempenho pode exigir um conhecimento profundo do sistema de gerenciamento de banco de dados específico, o que pode ser uma barreira para usuários iniciantes.

Manutenção de schema pode ser rigorosa: alterações frequentes no esquema do banco de dados podem ser complicadas de gerenciar, especialmente em sistemas de produção, devido à necessidade de garantir a consistência dos dados existentes.

Exemplos de SQL no AWS Athena no dia a dia

Consulta simples:

Objetivo: verificar os primeiros registros de uma tabela.

SQL:

SELECT * FROM my_table LIMIT 10;

Descrição: essa consulta retorna os primeiros 10 registros da tabela my_table, fornecendo uma rápida visualização dos dados.

Consulta com filtros e ordenação:

Objetivo: obter dados específicos e ordená-los.

SQL:

SELECT product_name, price FROM products WHERE category = 'Electronics' ORDER BY price DESC LIMIT 5;

Descrição: filtra produtos na categoria 'Electronics', ordena pelo preço em ordem decrescente e retorna os 5 primeiros resultados.

Consulta com junções:

Objetivo: combinar dados de duas tabelas relacionadas.

SQL:

SELECT orders.order_id, customers.customer_name FROM orders JOIN customers ON orders.customer_id = customers.customer_id WHERE orders.order_date >= '2024-01-01';

Descrição: realiza uma junção entre as tabelas orders e customers, trazendo os nomes dos clientes que fizeram pedidos após 1 de janeiro de 2024.

Consulta com agregações:

Objetivo: calcular estatísticas sobre os dados.

SQL:

SELECT category, COUNT(*) as total_products, AVG(price) as avg_price FROM products GROUP BY category HAVING total_products > 50 ORDER BY avg_price DESC;

Descrição: agrupa os produtos por categoria, calcula a contagem total e a média de preço, exibindo apenas categorias com mais de 50 produtos, ordenadas por preço médio decrescente.

Consulta com subconsultas:

Objetivo: utilizar resultados de uma consulta dentro de outra.

SQL:

SELECT customer_name FROM customers WHERE customer_id IN (SELECT DISTINCT customer_id FROM orders);

Descrição: retorna nomes de clientes que fizeram pelo menos um pedido, utilizando uma subconsulta para obter os IDs de clientes distintos na tabela orders.

Esses exemplos abrangem situações comuns de uso do Athena para realizar consultas analíticas e extrair informações valiosas de dados armazenados no Amazon S3. Adaptar esses exemplos conforme a estrutura específica dos seus dados e consultas desejadas é fundamental para otimizar o uso do Athena no seu dia a dia.

O QUE VOCÊ VIU NESTA AULA?

Nessa aula vimos como o AWS Athena nos ajuda a analisar grandes volumes dados do nosso data Lake no S3. Em conjunto com outros serviços, como o Glue, ele nos permite ETL e análises em tempo real ou sob demanda, nos possibilitando tirar todo o proveito do processamento paralelo sem servidor do serviço, essencial para a era do Big Data.

Revise esse material quantas vezes forem necessárias, pratique SQL em casa e em caso de dúvidas, entre em contato conosco. Bons estudos!

REFERÊNCIAS

AWS. **Amazon Athena.** [s.d.]. Disponível em: <https://aws.amazon.com/pt/athena/>. Acesso em: 22 mar. 2024.

CONNOLLY, T; BEGG, C. **Database Systems:** A Practical Approach to Design, Implementation, and Management. [S.l.]: Pearson, 2014.

DATE, C. J. **SQL and Relational Theory:** How to Write Accurate SQL Code. Sebastopol: O'Reilly Media, 2011.

ELMASRI, R; NAVATHE, S. B. **Fundamentals of Database Systems**. [S.l.]: Pearson, 2016.

GARCIA-MOLINA, H.; ULLMAN, J. D.; WIDOM, J. **Database Systems:** The Complete Book. [S.l.]: Pearson, 2008.

MELTON, J.; SIMON, A. R. **Understanding SQL's Stored Procedures:** A Complete Guide to SQL/PSM. Burlington: Morgan Kaufmann, 2004.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Database System Concepts**. Nova Iorque: McGraw-Hill, 2010.

PALAVRAS-CHAVE

Palavras-chave: Big Data. ETL. Data Lake. Banco de Dados. BI. Tomada de Decisão. Engenharia de Dados. AWS ATHENA. PrestoDB. SQL.