Resumo da Palestra

Aluna: Samila

A palestra contou com a participação de Lucas e Gabriel, que compartilharam experiências valiosas sobre carreira e fundamentos de bancos de dados.

Lucas

Lucas apresentou sua trajetória acadêmica e profissional, destacando os principais passos desde a graduação até sua posição atual no Databricks. Ele enfatizou a importância de participar de laboratórios durante a graduação, ressaltando que essa vivência prática é um diferencial competitivo no mercado de trabalho. Lucas também abordou o papel fundamental da mentoria, sugerindo que buscar um mentor é crucial para discutir planos futuros de carreira e receber orientações personalizadas. Ele mencionou o curso de criatividade de Murilo Gun, destacando que a criatividade é a base da inovação e pode ser desenvolvida por meio de treinamentos específicos.

Gabriel

Gabriel focou em conceitos essenciais de bancos de dados relacionais. Ele explicou o significado do acrônimo ACID, que representa os princípios de Atomicidade, Consistência, Isolamento e Durabilidade, fundamentais para garantir a integridade das transações em bancos de dados. Gabriel esclareceu que o termo "banco de dados relacional" deriva da álgebra relacional, base teórica desse modelo.

Foram detalhados diversos tipos de joins:

- **Inner Join**: Retorna apenas as linhas que possuem correspondência em ambas as tabelas. Ou seja, apenas os registros que atendem à condição de junção aparecem no resultado.
- **Left Join** (ou Left Outer Join): Retorna todas as linhas da tabela à esquerda e as linhas correspondentes da tabela à direita. Se não houver correspondência, os campos da tabela à direita vêm como `NULL`.
- **Right Join** (ou Right Outer Join): Retorna todas as linhas da tabela à direita e as linhas correspondentes da tabela à esquerda. Se não houver correspondência, os campos da tabela à esquerda vêm como `NULL`.
- **Full Outer Join**: Retorna todas as linhas quando há correspondência em uma das tabelas. Ou seja, combina os resultados do Left e do Right Join, preenchendo com `NULL` onde não houver correspondência.
- **Cross Join**: Realiza o produto cartesiano entre as tabelas, retornando todas as combinações possíveis entre as linhas das duas tabelas. Não utiliza condição de junção.

- **Self Join**: É um join de uma tabela com ela mesma, útil para comparar registros dentro da mesma tabela.
- **Left Anti Join**: Retorna apenas as linhas da tabela à esquerda que não possuem correspondência na tabela à direita.
- **Right Anti Join**: Retorna apenas as linhas da tabela à direita que não possuem correspondência na tabela à esquerda.
- **Anti Outer Join**: Retorna as linhas que não possuem correspondência em nenhuma das tabelas, ou seja, registros exclusivos de cada tabela.

Esses diferentes tipos de joins permitem realizar análises e consultas complexas, extraindo informações relevantes a partir de múltiplas fontes de dados.

Gabriel também abordou o uso de CTEs (*Common Table Expressions*) e explicou a diferença entre a ordem de escrita e a ordem de execução lógica das queries. Destacou que subqueries podem ser utilizadas tanto no `FROM` quanto no `WHERE`, e que, em muitos casos, subqueries são mais otimizadas do que joins, sendo preferível utilizá-las quando possível.

Além disso, foram discutidas as *window functions*, funções de deslocamento (*offset*) e funções de posição (*first*, *last*, *nth*), que permitem análises avançadas sobre conjuntos de dados.

Otimização de Consultas

Foram apresentadas dicas para otimizar consultas em bancos de dados, como o uso de índices e o comando `EXPLAIN` para analisar planos de execução. Recomenda-se evitar o uso de `SELECT *` e aplicar filtros o mais cedo possível nas queries, utilizando cláusulas `WHERE` para restringir o volume de dados processados.

Modelagem de Dados

A palestra abordou conceitos de modelagem, destacando a normalização e as diferenças entre ambientes OLTP (Online Transaction Processing), que são mais rápidos para gravação, e OLAP (Online Analytical Processing), otimizados para leitura. Também foi mencionado o conceito de OBT (One Big Table).

Arquitetura de Dados

Por fim, foi apresentada a evolução das arquiteturas de dados ao longo do tempo:

- **Data Warehouse** (1984)
- **Data Lake** (2010)
- **Modern Data Warehouse** (2011)
- **Data Lakehouse** (2020)

- **Data Mesh** (2019)

Cada arquitetura foi contextualizada em relação às necessidades e desafios de armazenamento, processamento e análise de grandes volumes de dados.

Conclusão: A palestra proporcionou uma visão abrangente sobre carreira, fundamentos e práticas avançadas em bancos de dados, além de apresentar tendências em arquitetura de dados, contribuindo para a formação e atualização dos participantes.