escola britânica de artes criativas & tecnologia

Módulo | SQL Avançado

Caderno de **Aula**
Professor Mariane Neiva

Tópicos

1. Subqueries;
2. Agregações por particionamento
3. Visões;

Aulas

Nessa aula, usaremos a seguinte tabela:

```
CREATE TABLE transacoes (
  id_cliente INT,
  id_transacao INT,
  data_compra DATE,
  valor FLOAT,
  id_loja varchar(25)
);
```

Também temos os seguintes valores inseridos na tabela

```
INSERT INTO transacoes VALUES (1,768805383,'2021-06-10',50.74,'magalu');
INSERT INTO transacoes VALUES (2,768805399,'2021-06-13',30.90,'giraffas');
INSERT INTO transacoes VALUES (3,818770008,'2021-06-05',110.00,'postoshell');
INSERT INTO transacoes VALUES (1,76856563,'2021-07-10',2000.90,'magalu');
INSERT INTO transacoes VALUES (1,767573759,'2021-06-20',15.70,'subway');
INSERT INTO transacoes VALUES (3,818575758,'2021-06-25',2.99,'seveneleven');
INSERT INTO transacoes VALUES (4,764545534,'2021-07-11',50.74,'extra');
INSERT INTO transacoes VALUES (5,76766789,'2021-08-02',10.00,'subway');
INSERT INTO transacoes VALUES (3,8154567758,'2021-08-15',1100.00,'shopee');
```

Como resultado da função SELECT, temos a seguinte tabela:

id_cliente	id_transacao	data_compra	valor	id_loja
1	768805383	2021-06-10	50.74	magalu
2	768805399	2021-06-13	30.90	giraffas
3	818770008	2021-06-05	110.00	postoshell
1	76856563	2021-07-10	2000.90	magalu
1	767573759	2021-06-20	15.70	subway
3	818575758	2021-06-25	2.99	seveneleven
4	764545534	2021-07-10	50.74	extra
5	76766789	2021-08-92	10.00	subway
3	8154567758	2021-08-15	1100.00	shopee

Além disso, também utilizaremos a tabela:

```
CREATE TABLE cliente (
  id_cliente INT,
  nome varchar(25),
  data_compra DATE,
  valor_compra float,
  loja_cadastro varchar(25)
);
```

Teremos os seguintes dados na tabela:

```
INSERT INTO cliente VALUES (5,'jose', '2020-07-01',500.43,'magalu');
INSERT INTO cliente VALUES (1,'maria','2019-03-02',150.70,'subway');
INSERT INTO cliente VALUES (2,'valentina','2020-01-09',210.99,'postoshell');
```

```
INSERT INTO cliente VALUES (4,'joana','2019-05-11',1300.50,'magalu');
INSERT INTO cliente VALUES (6,'fernando','2020-03-02',86.55,'seveneleven');
```

Como resultado temos:

id_cliente	nome	data_compra	valor_compra	loja_cadastro
5	jose	2020-07-01	500.43	cea
1	maria	2019-03-02	150.70	riachuelo
2	valentina	2020-01-09	210.99	zara
4	joana	2019-05-11	1300.50	pontofrio

1. Subqueries

Podemos resumir o uso de subqueries quando utilizamos um comando dentro do outro.

Para utilizar, basta inserir um comando

- dentro de um SELECT, INSERT, UPDATE ou DELETE
- junto com um operador = , >, <=, >= e LIKE.
- aliado a um WHERE, HAVING e FROM

A query *externa* é chamada de **main query** e a interna é chamada de **subquery**.

- a **subquery** deve ser utilizada com parenteses
- a **subquery** em geral é executada primeiro!
- é como se a **subquery** fosse um comando condicional associado
- **subquery** não pode ser associado com ORDER BY

Exemplo teórico:

```
SELECT <nome_coluna>
FROM <nome_tabela>
WHERE <nome_coluna> <expressao> <operador>
    ( SELECT <coluna>  from <tabela>   WHERE ... );
```

Exemplo prático:

```
SELECT id_loja, id_cliente, id_transacao from transacoes
WHERE id_loja IN
(SELECT cliente.loja_cadastro from cliente where cliente.valor_compra > 160 )
```

> **O que ele executa?**

Primeiro a **subquery** é executada.

O resultado de:

```
SELECT cliente.loja_cadastro from cliente where cliente.valor_compra > 160
```

loja_cadastro
magalu
postoshell
magalu

Ou seja, vamos considerar apenas essas lojas na seleção da **main query**.

É como se traduzissemos para:

```
SELECT id_loja, id_cliente, id_transacao from transacoes
WHERE id_loja IN
('magalu','postoshell')
```

Resultado:

id_loja	id_cliente	id_transacao
magalu	1	768805383
postoshell	3	818770008
magalu	1	76856563

Use a criatividade para gerar novas queries com subqueries!

2. Agregações por particionamento

Esse é um aspecto do AWS Athena utilizado para organizar e gerar as queries de maneira mais eficiente no framework.

É uma organização hierarquica onde cada *pasta* contem *subpastas* com o rótulo e valores.

> Por que utilizar? Para economizar dados carregados no AWS Athena, aumentando a performance reduzindo custos.

> Como fazer?
>
> - No S3, crie uma pasta no AWS com o nome do seu dataset.
> - Vamos supor que queremos separar as lojas na nossa partição.
> - Para isso criamos subpastas:
>   - transacoes_partition/id_loja=magalu
>   - transacoes_partition/id_loja=giraffas
>   - transacoes_partition/id_loja=postoshell

> - transacoes_partition/id_loja=subway
> - transacoes_partition/id_loja=seveneleven
> - transacoes_partition/id_loja=extra
> - transacoes_partition/id_loja=shopee

Dentro de cada uma das subpastas, colocamos apenas aquelas informações referentes a id_loja dedicada.

A geração da partição é indicada na hora da **CRIAÇÃO** da tabela com o comando **PARTITIONED by** id_loja (no exemplo).

Depois da criação, é necessário carregas as partições como o comando:

```
CREATE EXTERNAL TABLE transacoes_part(
  id_cliente BIGINT,
  id_transacoes BIGINT,
  valor DOUBLE)
  PARTITIONED BY (id_loja string)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
WITH SERDEPROPERTIES (
  'serialization.format' = ',',
  'field.delim' = ','
)
LOCATION 's3://transacoes-partition/'


MSCK REPAIR TABLE transacoes_part
```

Você pode verificar pela contagem de linhas na tabela completa:

```
select count(*) from transacoes_part
```

A partir disso, nós podemos seguir com os comandos de SELECT que aprendemos nos modulos do curso.

> Mais detalhes na aula prática!

3. Visões

Uma visão é uma tabela virtual, não fisica.

Isso significa que toda vez a view é referenciada, ela é também criada.

> Porque utilizar? Ao invés de fazer um SELECT várias vezes, você pode criar uma visão cujos dados são o resultado desse SELECT, ou quando:
>
> - você quer um subconjunto dos seus dados com frequência
> - você combina múltiplas tabelas
> - você quer simplificar as chamadas de queries

Funções para a visão:

nome	descrição
CREATE VIEW	cria a visão a partir de um select
DESCRIBE	mostra a lista de colunas da visão e atributos
DROP VIEW	deleta a visão
SHOW CREATE VIEW	mostra a query que criou a visao
SHOW VIEWS	mostra as visões disponíveis em uma base de dados
SHOW COLUMNS	lista as colunas em uma visão

Mais detalhes aqui

> Pratique!