09

Para saber mais: Ordem de execução do SQL

Agora que estamos acrescentando alguma complexidade às queries que o Sequelize vai passar para o SQL, é interessante relembrar que existe uma ordem de execução para os operadores e cláusulas.

No caso de queries de `SELECT`, a ordem lógica é a seguinte:

`FROM`: pega as tabelas onde estão os dados

`WHERE`: filtra os dados

`GROUP BY`: agrega os dados

`HAVING`: filtra os dados agregados

`SELECT`: retorna os resultados

`ORDER BY`: ordena os resultados

`LIMIT`: limita a quantidade de resultados

Ou seja, cada query começa encontrando os dados, filtrando e ordenando. Essa ordem pode fazer com que certos resultados sejam ou não acessíveis em dado momento. Por exemplo, a cláusula `WHERE` é executada antes de `GROUP BY`, então não podemos depender de dados retornados pelo `GROUP BY` para então passar `WHERE`.

Porém, os DBMS (*Database Management Systems*) como MySQL, PostgreSQL, MSSQL, entre outras, utilizam *database engines*, ou algo como "motores de database" numa tradução mais literal, para executar as queries. Esses *engines*, na prática, reorganizam a ordem lógica acima para otimizar as queries e deixá-las mais rápidas e com melhor performance, **enquanto essa reorganização não modificar os resultados da query**. Os *database engines* também fazem algumas verificações para garantir que a query faça sentido como um todo antes de fazer essa reorganização e rodar qualquer coisa.

Assim, embora exista uma ordem lógica na execução de uma query `SELECT`, e seja uma boa prática nos basearmos nela, na prática não temos realmente como saber qual é a ordem que será efetivamente utilizada, pois isso vai depender de como cada *engine* vai calcular a forma mais otimizada de execução da query.

A Julia Evans tem um artigo (https://jvns.ca/blog/2019/10/03/sql-queries-don-t-start-with-select/) (em inglês) que explica esta questão de forma bastante visual.