Infinitas possibilidades de SQL

Você aprendeu que uma consulta SQL usa **SELECT**, **FROM** e **WHERE** para especificar os dados a serem retornados pela consulta. Neste texto, apresentamos informações mais detalhadas sobre formatação de consultas, uso de condições **WHERE**, seleção de todas as colunas em uma tabela, adição de comentários e uso de aliases. Tudo isso torna mais fácil entender (e escrever) consultas para colocar o **SQL** em ação. Na última seção do texto, apresentamos um exemplo do que um analista de dados faria para extrair dados de funcionários para um projeto.

Uso de letras maiúsculas e minúsculas, recuo e ponto e vírgula

É possível escrever consultas **SQL** em letras minúsculas, e não é preciso se preocupar com espaços extras entre as palavras. No entanto, usar letras maiúsculas e recuo pode ajudar a ler as informações com mais facilidade.

Mantenha suas dúvidas organizadas e ficará mais fácil revisá-las ou solucioná-las se você precisar verificá-las mais tarde.

```
SELECT
      field1
FROM
      table
WHERE
      field1 = condition;
```

Observe que a instrução **SQL** mostrada acima tem um ponto e vírgula no final. O ponto e vírgula encerra instruções, faz parte da norma **SQL-92** do **American National Standards Institute (ANSI)** e é uma sintaxe comum recomendada para adoção por todos os bancos de dados **SQL**. No entanto, nem todos os bancos de dados SQL adotaram ou aplicam o ponto e vírgula.

Portanto, é possível que você encontre algumas instruções SQL que não terminam com ponto e vírgula. Se uma instrução funciona sem ponto e vírgula, não há problema nenhum.

Condições WHERE

Na consulta mostrada acima, a cláusula **SELECT** identifica a coluna da qual você quer extrair dados por nome, **field1**, e a cláusula **FROM** identifica a tabela em que a coluna está localizada por nome, **table**. Por fim, a cláusula **WHERE** restringe sua consulta para que o banco de dados retorne apenas os dados com correspondência de valor exata ou os dados que correspondem a uma determinada condição que você quer satisfazer.
Por exemplo, se você estiver procurando um cliente específico com o sobrenome Chavez, a cláusula **WHERE** será:

WHERE field1 = 'Chavez'

No entanto, se você estiver buscando todos os clientes com um sobrenome que comece com as letras “Ch”, a cláusula WHERE será:

WHERE field1 LIKE ‘Ch%'

É possível concluir que a cláusula **LIKE** é muito eficiente, porque permite que você diga ao banco de dados para procurar um determinado padrão! O sinal de porcentagem (%) é usado como curinga para corresponder a um ou mais caracteres. No exemplo acima, **Chávez** e **Chen** seriam retornados. Observe que, em alguns bancos de dados, o asterisco (*) é usado como curinga em vez do sinal de porcentagem (%).

SELECT (selecionar) todas as colunas

É possível usar SELECT * ?

No exemplo, se você substituir **SELECT field1** por **SELECT ***, selecionará todas as colunas da tabela, em vez de somente a coluna field1. Do ponto de vista da sintaxe, é uma instrução SQL correta, mas você precisa usar o asterisco (*) com moderação e cuidado. Dependendo de quantas colunas uma tabela tem, é possível selecionar uma quantidade enorme de dados. Selecionar muitos dados pode fazer com que uma consulta seja executada lentamente.

Comentários

Algumas tabelas não são projetadas com convenções de nomenclatura descritivas suficientes. No exemplo, **field1** era a coluna do sobrenome de um cliente, mas você não o reconheceria pelo nome. Um nome melhor seria algo como **last_name**. Nesses casos, é possível colocar comentários ao lado do SQL para ajudar a lembrar o que o nome representa. Comentários são textos colocados entre certos caracteres, **/*** e ***/**, ou depois de dois travessões (**--**), conforme mostrado abaixo.

```
SELECT
      field1 /* this is the last name column */
FROM
      table -- this is the customer data table
WHERE
      field1 LIKE 'Ch%';
```

Comentários também podem ser incluídos fora ou dentro de uma instrução. É possível usar essa flexibilidade para fornecer uma descrição geral do que você vai fazer, observações passo a passo sobre como conseguir isso e por que você definiu parâmetros/condições diferentes.

```
-- This is an important query used later to join with the accounts table
      SELECT
            rowkey,  -- key used to join with account_id
            Info.date,  -- date is in string format YYYY-MM-DD HH:MM:SS
            Info.code  -- e.g., 'pub-###'

      FROM  Publishers
```

Quanto mais confortável você ficar com o SQL, mais fácil será ler e entender as consultas rapidamente. Ainda assim, nunca é demais ter comentários em uma consulta para lembrar o que você está tentando fazer. Isso também facilita a compreensão de outras pessoas sobre sua consulta, caso ela seja compartilhada. À medida que suas consultas ficam cada vez mais complexas, essa prática economiza muito tempo e energia para entender consultas complexas que você escreveu meses ou anos atrás.

Exemplo de uma consulta com comentários

Veja aqui um exemplo de como comentários podem ser escritos no BigQuery:

```
-- Pull basic information from the customer table
SELECT
      customer_id, --main ID used to join with customer_addresss
      first_name, --customer's first name from loyalty program
      last_name --customer's last name
FROM
      customer_data.customer_name
```

No exemplo acima, um comentário foi adicionado antes da instrução **SQL** para explicar o que a consulta faz. Além disso, um comentário foi incluído ao lado de cada um dos nomes das colunas para descrever a coluna e o uso dela. Geralmente, dois traços (--) são compatíveis. Portanto, é melhor usar e ser consistente nisso. É possível usar **#** no lugar de -- na consulta acima, mas # não é reconhecido em todas as versões de **SQL**. Por exemplo, o **MySQL** não reconhece esse caractere. Se o banco de dados que você está usando for compatível, também será possível colocar comentários entre **/*** e ***/**.

À medida que você desenvolve suas habilidades profissionalmente, dependendo do banco de dados **SQL** que usa, é possível escolher os símbolos de delimitação de comentários apropriados de sua preferência e mantê-los como um estilo consistente. À medida que suas perguntas se tornam cada vez mais complexas, a prática de adicionar comentários úteis ajudará a economizar muito tempo e energia para compreender as perguntas que podem ter sido escritas meses ou anos antes.

Aliases

Também é possível atribuir um novo nome, ou **alias**, aos nomes de colunas ou tabelas para facilitar o trabalho com elas (e evitar a necessidade de comentários). Isso é feito com uma cláusula **SQL AS**. No exemplo abaixo, o alias **last_name** foi atribuído a **field1,** e o alias **customers** foi atribuído a **table.** Esses aliases são válidos apenas pela duração da consulta. Um alias não altera o nome real de uma coluna ou tabela no banco de dados.

Exemplo de uma consulta com aliases

```
field1 AS last_name -- Alias to make my work easier
table AS customers -- Alias to make my work easier

SELECT
      last_name
FROM
      customers
WHERE
      last_name LIKE 'Ch%';
```

Como executar o SQL como analista de dados

Imagine que você é um analista de dados de uma pequena empresa e seu gerente pede alguns dados de funcionários. Você decide escrever uma consulta com SQL para conseguir o que precisa do banco de dados.
Você deseja extrair todas as colunas **empID**, **firstName**, **lastName**, **jobCode** e **salary**. Como você sabe que o banco de dados não é tão grande, em vez de inserir cada nome de coluna na cláusula **SELECT**, você usa **SELECT***. Isso selecionará todas as colunas da tabela Employee (Funcionário) na cláusula **FROM**.

```
SELECT
      *
FROM
      Employee
```

Agora é possível ser mais específico sobre os dados que quer na tabela Employee. Se você quer todos os dados sobre os funcionários no código de trabalho **SFI**, pode usar uma cláusula **WHERE** para filtrar os dados com base nesse requisito adicional.

Aqui, você usa:

```
SELECT
      *
FROM
      Employee
WHERE
      jobCode = 'SFI'
```

Uma parte dos dados resultantes retornados da consulta **SQL** pode ter a seguinte aparência:

empID	firstName	lastName	jobCode	salary
2	Homer	Simpson	SFI	15000
3	Marge	Simpson	SFI	30000
34	Bart	Simpson	SFI	25000
67	Lisa	Simpson	SFI	38000
88	Ned	Flanders	SFI	42000
76	Barney	Gumble	SFI	32000

Suponha que você observe uma grande faixa salarial para o código de trabalho **SFI**. Talvez você queira sinalizar todos os funcionários de todos os departamentos com salários menores para o seu gerente. Como os estagiários também estão incluídos na tabela e têm salários inferiores a US$ 30 mil, você quer ter certeza de que os resultados contêm apenas os funcionários em tempo integral com salários inferiores a US$ 30 mil. Em outras palavras, você quer excluir estagiários com o código de trabalho **INT** que também ganham menos de US$ 30 mil. A cláusula **AND** permite que você teste as duas condições.

Você cria uma consulta **SQL** semelhante à abaixo, em que <> significa "não é igual”:

```
SELECT
      *
FROM
      Employee
WHERE
      jobCode <> 'INT'
      AND salary <= 30000;
```

Os dados resultantes da consulta SQL podem ter a seguinte aparência (estagiários com o código de trabalho **INT** não são retornados):

empID	firstName	lastName	jobCode	salary
2	Homer	Simpson	SFI	15000
3	Marge	Simpson	SFI	30000
34	Bart	Simpson	SFI	25000
108	Edna	Krabappel	TUL	18000
99	Moe	Szyslak	ANA	28000

Com acesso rápido a esse tipo de dado usando SQL, é possível fornecer ao gerente toneladas de diferentes insights sobre os dados dos funcionários, incluindo se os salários são equitativos em toda a empresa. Felizmente, a consulta indica que apenas dois outros funcionários podem precisar de um ajuste de salário, e você mostra os resultados ao seu gerente.
Retirar os dados, analisá-los e implementar uma solução pode, em última análise, ajudar a melhorar a satisfação e a lealdade dos funcionários. Isso torna o SQL uma ferramenta muito útil.

Recursos para saber mais
Quem não assina pode acessar esses recursos gratuitamente, mas, se um site limitar o número de artigos gratuitos por mês e você já tiver atingido o limite, marque o recurso como favorito e volte a ele mais tarde.

- **Tutorial de SQL da W3Schools**

  : Se você quiser explorar um tutorial detalhado de SQL, este é o lugar perfeito para começar. Este tutorial inclui exemplos interativos que você pode editar, testar e recriar. Use-o como referência ou conclua todo o tutorial para praticar o uso de SQL. Clique no botão verde **Começar a aprender SQL agora** ou no botão **Próximo** para iniciar o tutorial.
- **Folha de referências sobre SQL**

  : Se você já está em um nível mais avançado, leia este artigo para ver a sintaxe SQL padrão usada no PostgreSQL. Quando terminar, você saberá muito mais sobre o SQL e conseguirá usá-lo para análise de negócios e outras tarefas.