Bazy danych
przed 2 wykładem

Andrzej Giniewicz

08.03.2024

Podczas pierwszego wykładu rozmawialiśmy o zasadzie pracy z serwerem SQL oraz o różnych dialektach języka. Teraz pora poznać prostą wersję najważniejszej instrukcji — `SELECT`.

1 Baza danych

Bazą danych SQL nazywamy zbiór tabel. Każda tabela ma określoną liczbę kolumn i wierszy (rekordów). Kolumny są charakteryzowane przez nazwy. Możemy myśleć o bazie danych jak o jednym dokumencie arkusza kalkulacyjnego, mającego potencjalnie wiele arkuszy. Pojedynczy arkusz będzie wtedy tabelą. Każda tabela może mieć kolumny innego typu i o innych nazwach. Informacja o typach i nazwach kolumn nazywa się schematem tabeli. Opis wszystkich tabel w bazie nazywa się schematem bazy danych.

2 Wybór rekordów z tabeli

Gdy połączymy się z bazą danych, możemy pobrać z niej dane zawarte w tabeli. Aby pobrać wszystkie kolumny i rekordy, użyjemy komendy

```
SELECT * FROM nazwa_tabeli;
```

Pamiętajmy jednak, że taka komenda pobierze całą tabelę przez Internet, jeśli zatem jest to duża tabela, nie powinniśmy wykonywać tego zapytania bez ograniczenia liczby wierszy, które mają się pojawić. Wyświetlenie pierwszych 10 wierszy często jest przydatne, aby zorientować się, co znajduje się w tabeli.

```
SELECT * FROM nazwa_tabeli LIMIT 10;
```

Oba fragmenty kodu, to tak zwane zapytania. Zapytanie powinno kończyć się średnikiem. Tradycyjnie, do pisania słów kluczowych SQL używamy wielkich liter. Komenda `SELECT` informuje serwer baz danych, że chcemy pobrać dane z bazy. Fragment zapytania `FROM nazwa_tabeli` mówi, z której tabeli pobieramy dane. Gdyby w nazwie tabeli była spacja, powinniśmy napisać `FROM ‘nazwa tabeli‘`, używając znaku odwrotnego apostrofu (na

większości klawiatur znajduje się razem z klawiszem tyldy). Gwiazdka oznacza „wszystkie kolumny". Więcej możliwości wyrażeń, jakie można wpisać w tym miejscu, poznamy niebawem. Fragment zapytania `LIMIT 10` mówi nam, że jesteśmy zainteresowani jedynie pierwszymi 10 wierszami. Możemy również wyświetlić 10 wierszy, pomijając przykładowo pierwszych 7.

```
SELECT * FROM nazwa_tabeli LIMIT 7, 10;
```

Jednym z technicznym powodów pojawienia się podziału forów internetowych na strony była chęć oszczędzenia zasobów bazy danych. Jeśli na stronie z listą tematów jest 25 wątków, to aby wyświetlić 13 stronę, użyjemy `LIMIT 300, 25`, ponieważ $(13-1)\cdot 25 = 12\cdot 25 = 300$, czyli pomijamy pierwszych 12 stron, co przekłada się na 300 rekordów.

Ogólna zasada budowy zapytań jest podobna. Piszemy `SELECT` i dodajemy różne dodatkowe elementy składni, mówiące serwerowi baz danych, jakiego rezultatu oczekujemy. SQL jest jednym z języków czwartej generacji, w którym informujemy komputer, co chcemy zrobić, a nie jak. Tym, jakie algorytmy zostaną użyte do osiągnięcia celu, nie musimy się przejmować, jeśli nie planujemy zajmować się agresywną optymalizacją.

3 Filtracja przypadków

Nie zawsze jesteśmy zainteresowani wyborem pierwszych kilku wierszy. Często interesuje nas wybranie wierszy spełniających pewien warunek. Warunek filtrujący wiersze dodajemy komendą `WHERE` pomiędzy nazwą tabeli a `LIMIT`. Możemy napisać

```
SELECT * FROM nazwa_tabeli WHERE warunek LIMIT 10;
```

lub

```
SELECT * FROM nazwa_tabeli WHERE warunek;
```

Do zapisu warunków możemy używać nazw kolumn. Na przykład możemy dodać do naszego kodu warunek `WHERE sex='M'` albo `WHERE age>18`. Popularne operatory znajdują się w tabeli poniżej.

Operator	Znaczenie
=	równy
<>	różny
<	mniejszy
>	większy
<=	mniejszy bądź równy
>=	większy bądź równy

Oprócz tego często stosujemy trzy operatory:

- `wartość BETWEEN a AND b`, co oznacza, że wartość jest w przedziale domkniętym od `a` do `b`,

- `wartość IN (a1, a2, a3, ...)`, co oznacza, że wartość jest jedną z wypisanych w nawiasie oraz
- `wartość LIKE 'wyrażenie'`, gdzie wyrażenie może być wyrażeniem regularnym, w którym % oznacza dowolny ciąg znaków, potencjalnie pusty, natomiast _ oznacza jeden dowolny znak.

Różne warunki możemy łączyć za pomocą wyrażeń logicznych `AND`, `OR` oraz `NOT`. Możliwe jest również nawiasowanie, które pomaga zapisać trudniejsze warunki, na przykład

```
(sex = 'M' AND age BETWEEN 18 AND 65) OR (sex = 'F' AND age BETWEEN 18 AND 60)
```

stanowi warunek opisujący wiek emerytalny.

W jednym zapytaniu może być co najwyżej jedno słowo kluczowe `WHERE`. W momencie, gdy zaczynamy dodawać bardziej złożone warunki, warto rozbić zapytanie na kilka linii. Spacja oraz znak nowej linii jest traktowany przez SQL w ten sam sposób.

Innym sposobem na wybranie tylko niektórych wartości, jest użycie słowa kluczowego `DISTINCT` tuż za `SELECT`. Zapytanie `SELECT DISTINCT` najpierw wykonuje operacje filtrowania za pomocą `WHERE`, a następnie usuwa zduplikowane wiersze.

4 Porządkowanie wyników

Kolejną komendą, która może się pojawić, jest porządkowanie wyników. Robimy to komendą `ORDER BY`, po której może pojawić się jedno lub więcej wyrażeń oddzielonych przecinkami. Porządkowanie następuje od lewej do prawej, czyli zgodnie z porządkiem leksykograficznym. Możemy na przykład napisać `ORDER BY age`, co posortuje osoby po wieku w kolejności rosnącej. Jest to równoważne napisaniu `ORDER BY age ASC` od angielskiego *ascending*. Odwrotną kolejność zadajemy, pisząc `ORDER BY age DESC`. Jeśli chcemy, aby osoby w tym samym wieku były posortowane według nazwisk rosnąco, napiszemy na przykład

```
ORDER BY age DESC, name
```

Operacja sortowania odbywa się tylko na wynikach spełniających warunek z `WHERE`, nie na całej tabeli, więc jeśli odfiltrowaliśmy dużo wierszy, będzie stosunkowo szybka. Jeśli używamy `SELECT DISTINCT`, porządkowane będą jedynie wiersze unikatowe. Częstym zabiegiem, aby sprawdzić wszystkie dane najstarszego mężczyzny, jest posortowanie danych i wybranie jednego wiersza.

```
SELECT * FROM nazwa_tabeli WHERE sex='M' ORDER BY age DESC LIMIT 1;
```

Powyższe zapytanie najpierw odfiltruje z tabeli tylko wiersze spełniające warunek `sex='M'`, następnie posortuje wyniki malejąco po kolumnie `age` i zwróci pierwszy wiersz.

5 Wybieranie i tworzenie nowych kolumn

Wybieranie kolumn nazywa się inaczej projekcją. Do tej pory wszystkie przypadki wybierały pełen zbiór kolumn za pomocą gwiazdki. Jeśli chcemy, możemy też wybrać tylko kilka kolumn. Robimy to, wypisując je oddzielone przecinkami, przykładowo

```
SELECT age, sex FROM nazwa_tabeli;
```

Możemy też dodać tak zwane aliasy, aby w wyniku pojawiła się inna nazwa niż oryginalnie.

```
SELECT age AS wiek, sex FROM nazwa_tabeli;
```

Maksymalne ograniczenie kolumn oraz przypadków do tylko takich, których potrzebujemy, jest bardzo istotne, ponieważ zmniejsza ilość przesyłanych przez sieć danych i przyśpiesza czas zapytania.

Możemy również wykonać przekształcenia lub obliczenia, aby utworzyć nowe kolumny. Zwykle nowe kolumny mają podane aliasy, ponieważ nazwa zawierająca wyrażenie arytmetyczne, może być trudniejsza w użytkowaniu.

```
SELECT x, y, x+y AS z FROM nazwa_tabeli;
```

Powyższe zapytanie zwróci wynik mający trzy kolumny: `x`, `y` oraz `z`, przy czym `z` jest wyliczone z dwóch pozostałych jako suma. Lista możliwych funkcji i operacji na liczbach znajduje się w dokumentacji pod adresem `https://mariadb.com/kb/en/numeric-functions/`, natomiast opis funkcji dla napisów pod adresem `https://mariadb.com/kb/en/string-functions/`. Na stronie `https://mariadb.com/kb/en/control-flow-functions/` znajdziemy funkcje takie, jak `IF` oraz `CASE`. Funkcje dla innych typów, takich jak daty, będą omówione na dalszych wykładach. Wszystkie te funkcje działają osobno dla każdego przypadku.

6 Funkcje agregujące

Funkcje agregujące, w przeciwieństwie do funkcji działających na przypadkach, działają na całej kolumnie i zwracają jeden wiersz. Przykładami często używanych funkcji agregujących są `SUM`, `AVG` oraz `COUNT`. Przykładowo możemy sprawdzić średnie zarobki.

```
SELECT AVG(zarobki) AS wynik FROM tabela;
```

Średnie zarobki osób w wieku poniżej 35 lat sprawdzimy natomiast zapytaniem

```
SELECT AVG(zarobki) AS wynik FROM tabela WHERE age<35;
```

Warto zaznaczyć, że `DISTINCT` działa już na wyniku działania funkcji. Jeśli w zapytaniu jest funkcja agregująca, rezultat będzie miał tylko jeden wiersz, zatem będzie on unikatowy niezależnie od sytuacji. Dla funkcji `COUNT` wyznaczającej liczbę przypadków, istnieje wariant zliczający liczbę różnych przypadków. Używamy tego samego słowa kluczowego `DISTINCT`, jednakże umieszczamy je nie po `SELECT`, tylko w nawiasie funkcji `COUNT`.

```
SELECT COUNT(DISTINCT name) FROM tabela;
```

zwróci liczbę różnych nazwisk w danych.

Jeśli w zapytaniu używamy funkcji agregujących, nie powinniśmy używać kolumn, które nie są stałe, ponieważ w wyniku pojawi się pewna wartość kolumny — nie wiadomo, która. W zapytaniu

```
SELECT COUNT(DISTINCT name), firstname FROM tabela;
```

obok liczby różnych nazwisk, pojawi się jedno imię. To, które imię to będzie, zależy od implementacji silnika baz danych.

Dokumentacji funkcji agregujących dostępnych w MariaDB znajduje się na stronie `https://mariadb.com/kb/en/aggregate-functions/`.