Bazy danych
przed 7 wykładem

Andrzej Giniewicz

19.04.2024

Dziś zajmiemy się podzapytaniami, czyli „zapytaniami w zapytaniach".

Podzapytanie to zapytanie `SELECT` znajdujące się wewnątrz innego zapytania. Podzapytanie nazywamy też zapytaniem wewnętrznym, a zapytanie, w którym jest umieszczone podzapytanie, zapytaniem zewnętrznym. Rozróżniamy cztery rodzaje podzapytań, zależnie od tego, jaka jest wartość zwracana przez zapytanie wewnętrzne.

1 Podzapytania skalarne

Podzapytanie skalarne (ang. scalar subquery), jest podzapytaniem, które zwraca jedną wartość, czyli jeden wiersz jednej kolumny. Może wystąpić w dowolnym miejscu, w którym SQL oczekuje literału. Na przykład możemy sprawdzić, które osoby zarabiają więcej niż średnią.

```
SELECT * FROM pracownicy
WHERE zarobki > (SELECT AVG(zarobki) FROM pracownicy);
```

Podzapytania powinniśmy zawsze otaczać nawiasami. Tutaj podzapytaniem jest

```
SELECT AVG(zarobki) FROM pracownicy
```

które zwraca średnie zarobki jako jedną liczbę. Podzapytanie zewnętrzne wykorzystuje tę liczbę i wybiera tych pracowników, którzy zarabiają powyżej średniej.

Może również być traktowane jako podzapytanie kolumnowe z jednym wierszem, wierszowe z jedną kolumną lub jako tabela pochodna z jednym wierszem i jedną kolumną. Wymienione tu typy zapytań opisane zostaną poniżej.

2 Podzapytanie kolumnowe

Podzapytanie kolumnowe (ang. column subquery), jest podzapytaniem, które zwraca jedną kolumnę wartości. Najczęściej występuje w zapytaniach z operatorami `ALL` i `ANY` w warunku `WHERE`. Jeśli w warunku po operatorze wstawimy `ALL` lub `ANY`, warunek będzie prawdziwy, jeśli jest prawdziwy dla wszystkich wartości z podzapytania w przypadku `ALL` oraz dla dowolnej wartości z podzapytania w przypadku `ANY`. Możemy na przykład zapisać

```
WHERE użytkownik = ANY (SELECT użytkownicy FROM ...)
```

co spowoduje wybór kolumy użytkowników podzapytaniem, po czym wartość warunku `WHERE` będzie prawdą, jeśli użytkownik z warunku pojawił się w dowolnym wierszu podzapytania. Operator `= ANY` możemy zapisać za pomocą prostszego synonimu `IN`

```
WHERE użytkownik IN (SELECT użytkownicy FROM ...)
```

Podobnie operator `<> ALL` możemy zapisać jako `NOT IN`. Przed `ALL` i `ANY` może pojawić się dowolny operator porównywania (<, >, <=, >=, =, <>).

Podzapytanie kolumnowe może być również traktowane jako tabela pochodna z jedną kolumną.

3 Podzapytanie wierszowe

Podzapytanie wierszowe (ang. row subquery), jest podzapytaniem, które zwraca jeden wiersz. Najczęściej występuje w warunku `WHERE` z operatorem porównywania (<, >, <=, >=, =, <>), gdzie po jego lewej stronie znajduje się więcej niż jedna kolumna.

```
WHERE (x, y) = (SELECT MAX(x), MAX(y) FROM ...)
```

jest logicznie równoważny do dwóch podzapytań

```
WHERE x = (SELECT MAX(x) FROM ...) AND y = (SELECT MAX(y) FROM ...)
```

jednakże skorzystanie z podzapytania wierszowego będzie w tym przypadku niemal dwukrotnie szybsze, ponieważ można policzyć maksimum obu kolumn przy jednokrotnym przejściu tabeli.

Podzapytanie wierszowe może być również traktowane jako tabela pochodna z jednym wierszem.

4 Tabele pochodne

Tabele pochodne (ang. derived table), są najbardziej ogólnym podzapytaniem, które może zwracać tabelę mającą więcej niż jeden wiersz i jedną kolumnę. Może wystąpić w klauzuli `FROM` lub z `WHERE` z warunkiem `EXISTS`.

Jeśli przykładowo chcemy policzyć liczbę wierszy zwracanych przez bardziej skomplikowane zapytanie, możemy napisać

```
SELECT COUNT(*) FROM (SELECT ...) AS t;
```

Podzapytanie w nawiasie może być dowolnie złożone, ale ostatecznie zwraca wynik jako pewną tabelę. Tabelę tę na potrzeby zapytania zewnętrznego nazywamy `t` (ponieważ każda tabela w MySQL i MariaDB musi mieć jakąś nazwę) i wykonujemy na niej zapytanie `SELECT COUNT(*) FROM ...`, które liczy liczbę wierszy.

Podzapytania zwracające tabele pochodne mogą wystąpić również z łączeniem tabel, wszędzie tam, gdzie oczekujemy tabeli. Jeśli użyjemy tabeli pochodnej z operatorem `EXISTS`, warunek będzie prawdziwy, jeśli tabela pochodna ma choć jeden wiersz.

```
WHERE EXISTS (SELECT * FROM pracownicy WHERE zarobki>10000)
```

Jeśli choć jedna osoba w zapytaniu wewnętrznym ma zarobki powyżej 10000, podzapytanie będzie miało przynajmniej jeden wiersz, więc warunek będzie spełniony.

5 Podzapytania skorelowane

Podzapytania mogą być niezależne od zapytań zewnętrznych lub skorelowane. Zapytanie nazywamy skorelowanym, jeśli wynik jego działania jest różny dla poszczególnych wierszy. Przykładem takiego zapytania może być następujący fragment.

```
SELECT wiek, zarobki FROM pracownicy AS t1
WHERE zarobki > (SELECT AVG(zarobki) FROM pracownicy AS t2
                 WHERE t2.wiek = t1.wiek);
```

Przeanalizujmy kod. Zacznijmy od zapytania wewnętrznego. Zapytanie to liczy średnie zarobki osób w wieku `t1.wiek`. Wiek nie jest tutaj stały, tylko pochodzi z tabeli z zewnętrznego zapytania — decydując dla każdej osoby z osobna, czy zarabia powyżej średniej, musimy już wiedzieć, ile ma lat, ponieważ dla każdego wieku wyznaczyliśmy osobną średnią.

Taki rodzaj zapytań najczęściej wykonuje się dużo wolniej. Jeśli zapytanie nie jest skorelowane, SQL liczy jego wartość i potem dla każdego wiersza używa wyniku. Jeśli zapytanie jest skorelowane, dla każdego wiersza konieczne jest uruchomienie całego podzapytania od nowa. Z tego powodu, jeśli podzapytanie i zapytanie zewnętrzne dotyczą tej samej tabeli, złożoność zmienia się z $\Theta(n)$ do $\Theta(n^2)$, co przy ogromnych tabelach może spowodować bardzo duże czasy wykonania. Warto pomyśleć wtedy, czy potrzebujemy podzapytań skorelowanych, ponieważ często da się je zapisać za pomocą połączenia dwóch tabel, co może okazać się wydajniejsze.

Nie zawsze zapytania skorelowane są mniej wydajne. Niekiedy, jeśli podzapytanie skorelowane jest małe i wykonuje się szybko, może się okazać, że będzie wydajniejsze niż alternatywne rozwiązania. Zawsze warto rozważyć rozmiary tabel, którymi dysponujemy i pomyśleć o złożoności zapytań.

Niekiedy zapytanie, które nie jest skorelowane, jest konwertowane na takie przez pomyłkę optymalizatora. W takich sytuacjach pozornie niewinne zapytanie, szczególnie często stosowane ze słowem kluczowym `IN` w warunku `WHERE`, gdy zarazem zapytanie zewnętrzne i podzapytanie korzystają z tej samej tabeli, może znacząco spowolnić program. Jednym z rozwiązań problemu jest wtedy opakowanie podzapytania w kolejne podzapytanie i zamiast

```
WHERE zmienna IN (podzapytanie)
```

napisać

```
WHERE zmienna IN (SELECT * FROM (podzapytanie) AS t)
```

ten zabieg zwany opakowaniem podzapytania, wymusza na optymalizatorze potraktowanie tabeli jako niezależnej od tabeli zewnętrznej, ponieważ operator `IN` nie pracuje już na tej samej tabeli, co zapytanie zewnętrzne, tylko na tabeli `t`, która tak się składa, jest identyczna co do wartości do podzapytania na tabeli zewnętrznej.