MySQL-Unterabfragen

Colletta Hagert

Unister

Überblick

- Einführung
- Typen von Unterabfragen
  - Skalare Unterabfragen
  - Datensatzunterabfrage
  - Korrelierte Unterabfragen
  - Tabellen Unterabfragen
- Zusammenfassung

Eine Unterabfrage ist eine SELECT-Anweisung innerhalb einer anderen Anweisung.

Beispiel

SELECT * FROM t1 WHERE column1 = (SELECT column1 FROM t2);

SELECT * FROM t1 ... ist die äußere Abfrage (oder äußere Anweisung), und (SELECT column1 FROM t2) ist die Unterabfrage.

- Eine Unterabfrage muss immer in Klammern stehen.

Vorteile

- Unterabfragen bieten alternative Wege zur Durchführung von Operationen, die andernfalls komplexe Joins und Unions erfordern würden.

- Sie gestatten Abfragen, die strukturiert sind.

=> jeder Teil einer Anweisung kann isoliert werden

- Sind lesbarer als komplexe Joins oder Unions.

Typen von Unterabfragen

Unterabfragen definieren:

- Eine Unterabfrage kann die gleichen Klausen wie eine gewöhnliche SELECT-Anweisung enthalten:
    - DISTINCT,
    - GROUP BY,
    - ORDER BY,
    - LIMIT,
    - Joins,
    - Indexhinweise, UNION-Konstrukte, Kommentare, Funktionen usw.
- die äußere Anweisung einer Unterabfrage muss einen der folgenden Typen aufweisen muss: SELECT, INSERT, UPDATE, DELETE, SET oder DO.
- Derzeit ist es nicht Möglich eine Tabelle zu ändern und gleichzeitig in einer Unterabfrage eine Auswahl aus dieser Tabelle zu treffen.

Beschränkungen von Unterabfragen

Die Beschränkung gilt beispielsweise für Anweisungen der folgenden Form:

```
DELETE FROM tabelle WHERE ... (SELECT ... FROM tabelle ...);
UPDATE tabelle ... WHERE spalte = (SELECT ... FROM tabelle ...);
{INSERT|REPLACE} INTO tabelle (SELECT ... FROM tabelle ...);
```

- **Ausnahme**: Das obige Verbot gilt nicht, wenn Sie eine Unterabfrage für die modifizierte Tabelle in der FROM-Klausel verwenden.

Beispiel:

```
UPDATE tabelle ... WHERE spalte = (SELECT (SELECT ... FROM tabelle...)
    AS _tabelle ...);
```

Mögliche Rückgabewerte einer Unterabfrage sind:

- ein Skalar (einen einzelnen Wert)

=> Skalare Unterabfrage

- ein einzelner Datensatz

=> Datensatz Unterabfrage

- eine einzelne Spalte oder eine Tabelle (d. h. eine oder mehrere Spalten von einem oder mehreren Datensätzen)

=> Tabellen Unterabfrage

Skalare Unterabfragen

- Gibt einen Einzelwert zurück
- Kann fast überall dort benutzen werden, wo ein einzelner Spaltenwert, ein Funktionsergebnis oder ein Literal zulässig ist.
- eine solche Unterabfrage weist die Eigenschaften auf, die alle Operanden haben: einen Datentyp, eine Länge, eine Angabe dazu, ob sie NULL sein darf oder nicht usw.
- Fehlermeldung, wenn mehrere Spalten oder Zeilen zurück gegeben werden

Beispiel

```
SELECT Country.Name,
        100 * Country.Population / (SELECT SUM(Population) FROM
                                    Country)  AS pct_of_world_population
FROM Country;
```

Datensatzunterabfrage

- Gibt einen einzelnen Datensatz zurück

=> mehrere Spaltenwerte

- Spezielle Art von sog. Datensatzkonstruktoren

=> Ausdrücke wie **(1,2)** und **ROW(1,2)**. Diese Ausdrücke sind äquivalent

- Datensatzunterabfragen (und andere Datensatzkonstruktoren) können als Operanden für die folgenden Operatoren verwendet werden: =, <>, !=, <=>, <=, =>, <, >
- Fehlermeldung, wenn mehrere Zeilen zurück gegeben werden oder die Anzahl der Spalten der beiden Datensatz-konstruktoren nicht übereinstimmt.

Beispiel

SELECT ('London', 'GBR') = SELECT Name, Country FROM City WHERE ID = 456) AS IsLondon;

Der normale Anwendungsfall von Datensatzkonstruktoren sind Vergleiche mit Unterabfragen, die zwei oder mehr Spalten zurückgeben.

```
SELECT column1,column2,column3
    FROM t1
    WHERE (column1,column2,column3) IN
        (SELECT column1, column2, column3 FROM t2);
```

Diese Abfrage entspricht der Aufforderung „Suche alle Datensätze in Tabelle t1, die auch in Tabelle t2 vorhanden sind“.

Korrelierte Unterabfragen

Eine korrelierte Unterabfrage ist eine Unterabfrage, die eine Tabelle referenziert, die auch in der äußeren Abfrage erscheint.

Beispiel

```
SELECT Country.Name, (SELECT count(*)
                      FROM City
                      WHERE CountryCode = Country.Code) AS
                      CityCount
FROM Country
WHERE Region = 'Nordic Countries';
```

Im Beispiel referenziert die Unterabfrage eine Spalte, Code, von der Tabelle Country, die in der äußeren Abfrage vorkommt

Tabellen Unterabfragen

Tabellen Unterabfragen werden in zwei unterschiedlichen Kontexte verwendet:

1. In der FROM-Klausel einer SELECT-Anweisung

=> ersetzen eine Tabellename

Syntax

SELECT ... FROM (unterabfrage) [AS] name ...

- Unterabfragen in der FROM-Klausel können einen Skalar, eine Spalte, einen Datensatz oder eine Tabelle zurückgeben.
- Unterabfragen in der FROM-Klausel dürfen keine korrelierten Unterabfragen sein.

Beispiel

```
CREATE TABLE t1 (s1 INT, s2 CHAR(5), s3 FLOAT);
INSERT INTO t1 VALUES (1, '1', 1.0);
INSERT INTO t1 VALUES (2, '2', 2.0);

SELECT sb1, sb2, sb3
FROM (SELECT s1 AS sb1, s2 AS sb2, s3*2 AS sb3 FROM t1) AS sb
WHERE sb1 > 1;

// Ergebnis: 2, '2', 4.0.
```

2. Als rechte Seite Operand für die logische Operatoren IN, NOT EXISTS und EXISTS oder für die Vergelichsoperatoren =, !=, <>,<,>,<= oder >= (aber nicht <=>) sofern eine der Schlüsselwörter ANY, ALL oder SOME dem Vergleichsoperator folgt

Syntax I

WHERE operand vergleichs_operator **ANY** (unterabfrage)

WHERE operand **IN** (unterabfrage)

WHERE operand vergleichs_operator **SOME** (unterabfrage)

- Das Wort IN ist ein Alias für = ANY.
- NOT IN ist ein Alias für <> ALL, kein Alias für <> ANY.
- Das Wort SOME ist ein Alias für ANY.

Beispiele: Verwendung von ANY, IN SOME

Die folgenden beiden Anweisungen sind gleichwertig:

```
SELECT s1 FROM t1 WHERE s1 = ANY (SELECT s1 FROM t2);
SELECT s1 FROM t1 WHERE s1 IN (SELECT s1 FROM t2);
```

Die folgenden beiden Anweisungen sind gleichwertig:

```
SELECT s1 FROM t1 WHERE s1 <> ANY  (SELECT s1 FROM t2);
SELECT s1 FROM t1 WHERE s1 <> SOME (SELECT s1 FROM t2);
```

Syntax II

WHERE EXISTS (unterabfrage)

WHERE NOT EXISTS (unterabfrage)

Gibt eine Unterabfrage einen Datensatz zurück, ist eine EXISTS (Unterabfrage) TRUE und eine NOT EXISTS (Unterabfrage) FALSE.

Beispiel: Verwendung von EXISTS

```
SELECT *
FROM City
WHERE EXISTS (
        SELECT NULL
        FROM Country
        WHERE Capital = ID)
```

Zusammenfassung

- Eine Unterabfrage ist eine SELECT-Anweisung innerhalb einer anderen Anweisung.
- Kann die gleichen Klausen wie eine gewöhnliche SELECT-Anweisung enthalten
- Die äußere Anweisung einer Unterabfrage muss einen der folgenden Typen aufweisen muss: SELECT, INSERT, UPDATE, DELETE, SET oder DO.

Vorteile

- Unterabfragen bieten alternative Wege zur Durchführung von Operationen, die andernfalls komplexe Joins und Unions erfordern würden.
- Sie gestatten Abfragen, die strukturiert sind.
- Sind lesbarer als komplexe Joins oder Unions.

Typen von Unterabfragen

Skalare Unterabfrage

Der Rückgabewerte einer Unterabfrage ist ein Skalar (einen einzelnen Wert)

Datensatz Unterabfrage

Der Rückgabewerte einer Unterabfrage ist ein einzelner Datensatz

=> mehrere Spaltenwerte

Korrelierte Unterabfrage

Eine korrelierte Unterabfrage ist eine Unterabfrage, die eine Tabelle referenziert, die auch in der äußeren Abfrage erscheint.

Typen von Unterabfragen

Tabellen Unterabfrage

Der Rückgabewerte einer Unterabfrage ist eine einzelne Spalte oder eine Tabelle (d. h. eine oder mehrere Spalten von einem oder mehreren Datensätzen)

Werden in zwei unterschiedlichen Kontexte verwendet:

1. In der FROM-Klausel einer SELECT-Anweisung

2. Als rechte Seite Operand für die logische Operatoren IN, NOT EXISTS und EXISTS oder für die Vergelichsoperatoren =, !=, <>,<,>,<= oder >= (aber nicht <=>) sofern eine der Schlüsselwörter ANY, ALL oder SOME dem Vergleichsoperator folgt

Quellen

http://dev.mysql.com/doc/refman/5.1/de/subqueries.html