Einführung in Datenbanken

Kapitel 11: Sichten und CTEs in SQL

Prof. Dr. Stefan Brass

Martin-Luther-Universität Halle-Wittenberg

Wintersemester 2021/22

http://www.informatik.uni-halle.de/~brass/db21/

Lernziele

Nach diesem Kapitel sollten Sie Folgendes können:

- Unteranfragen unter `FROM` verwenden.
- Die Korrektheit von Anfragen mit Unteranfragen unter `FROM` beurteilen.
- Sichten (virtuelle Tabellen) mittels `CREATE VIEW`-Befehl anlegen.
- Den Nutzen von Sichten erläutern.
- Anfragen lesen und schreiben, die mit `WITH` Hilfstabellen einführen.
- Beurteilen, wann die Verwendung von Sichten bzw. `WITH` angemessen ist.

Inhalt

Beispiel-Datenbank

STUDENTEN

SID	VORNAME	NACHNAME	EMAIL
101	Lisa	Weiss	...
102	Michael	Grau	NULL
103	Daniel	Sommer	...
104	Iris	Winter	...

BEWERTUNGEN

SID	ATYP	ANR	PUNKTE
101	H	1	10
101	H	2	8
101	Z	1	12
102	H	1	9
102	H	2	9
102	Z	1	10
103	H	1	5
103	Z	1	7

AUFGABEN

ATYP	ANR	THEMA	MAXPT
H	1	ER	10
H	2	SQL	10
Z	1	SQL	14

Unteranfragen unter FROM (1)

- Da das Ergebnis einer SQL-Anfrage eine Tabelle ist, bietet sich an, dass man Unteranfragen an Stelle einer Tabelle in der FROM-Klausel schreiben kann.

  Das war in SQL-86 verboten, und SQL wurde damals oft kritisiert, „nicht orthogonale Konstrukte“ zu haben, die man nicht beliebig kombinieren kann.

- Hier wird der Verbund von STUDENTEN und BEWERTUNGEN (nur für Hausaufgaben) in einer Unteranfrage berechnet:

```
SELECT X.ANR, X.PUNKTE
FROM   (SELECT VORNAME, NACHNAME, ANR, PUNKTE
        FROM   STUDENTEN S, BEWERTUNGEN B
        WHERE  S.SID = B.SID AND ATYP = 'H') X
WHERE  VORNAME = 'Lisa' AND NACHNAME = 'Weiss'
```

  In der äußeren Anfrage beziehen sich VORNAME und NACHNAME auf X, innen natürlich auf S.

Unteranfragen unter FROM (2)

- Im obigen Beispiel verbessert die Unteranfrage leider nicht die Lesbarkeit der gesamten Anfrage.
- Unteranfragen unter FROM werden aber für für geschachtelte Aggregationen unbedingt benötigt, siehe Kapitel 13.
- Außerdem sind Unterabfragen unter FROM die Grundlage für Sichten (virtuelle Tabellen) und „Common Table Expressions (CTEs)“ (s.u.):
  - Man möchte große Anfragen stückweise aufbauen, so wie man Prozeduren in in der Programmierung einsetzt, um ein zu langes Programm sinnvoll zu strukturieren.
  - Mit Sichten kann man wiederverwendbare Bausteine für Anfragen definieren.

Unteranfragen unter FROM (3)

Syntaktische Feinheiten:

- SQL-92, PostgreSQL, SQL Server und DB2 fordern die Definition einer Tupelvariable für die Unteranfrage; in Oracle und Access ist das optional.
- SQL-92, PostgreSQL, SQL Server und DB2 (nicht Oracle8, Access) lassen folgende Umbenennung von Spalten zu:

```
FROM (...) X(AUFG_TYP, AUFG_NR, ...)
```

- In Oracle und Access können Spalten nur innerhalb der Unteranfrage umbenannt werden.

  Alle Systeme unterstützen die Spezifikation neuer Spaltennamen in der SELECT-Klausel, so dass dies eine portabele Möglichkeit ist.

Unteranfragen unter FROM (4)

Syntaktische Feinheiten, Forts.:

- Innerhalb der Unteranfrage kann man nicht auf andere Tupelvariablen zugreifen, die in der gleichen FROM-Klausel definiert werden:

```
SELECT S.VORNAME, S.NACHNAME, X.ANR, X.PUNKTE
FROM   STUDENTEN S,
       (SELECT B.ANR, B.PUNKTE
        FROM   BEWERTUNGEN B
        WHERE  B.ATYP = 'H'
        AND    B.SID = S.SID) X
```

Falsch!

Die Unteranfragen der gleichen FROM-Klausel müssen also parallel auswertbar sein, und können nicht von einander abhängen. Man dürfte in den Unteranfragen aber auf Tupelvariablen zugreifen, die weiter außen deklariert sind.

Unteranfragen unter FROM (5)

Lateral Join:

- Mit dem Schlüsselwort „LATERAL“ („seitlich“, „quer“) vor der Unteranfrage wird es legal, auf links davon definierte Tupelvariablen zuzugreifen:

```
SELECT S.VORNAME, S.NACHNAME, X.ANR, X.PUNKTE
FROM   STUDENTEN S,
       LATERAL (SELECT B.ANR, B.PUNKTE
                FROM   BEWERTUNGEN B
                WHERE  B.ATYP = 'H'
                AND    B.SID = S.SID) X
```

- PostgreSQL unterstützt das seit Version 9.3.

  Insgesamt ist es im Moment noch nicht übermäßig portabel. DB2 unterstützt es seit Version 9.1, Oracle seit Version 12cR1, MySQL seit Version 8.0.14.
  [https://modern-sql.com/slides/ModernSQL-2019-05-08.pdf]

Inhalt

Sichten (1)

- Eine Sichtdeklaration speichert eine Anfrage unter einem Namen in der Datenbank:

```
CREATE VIEW HA_PUNKTE AS
   SELECT   VORNAME, NACHNAME, ANR, PUNKTE
   FROM     STUDENTEN S, BEWERTUNGEN B
   WHERE    S.SID=B.SID AND ATYP = 'H'
```

- Sichten können in Anfragen wie normale Tabellen verwendet werden:

```
SELECT ANR, PUNKTE
FROM   HA_PUNKTE
WHERE  VORNAME = 'Lisa' AND NACHNAME = 'Weiss'
```

- Eine Sicht ist eine Abkürzung für eine Unteranfrage.

Sichten (2)

- Wird eine Sicht in einer Anfrage verwendet, so ersetzt das DBMS nur den Sichtnamen durch die Anfrage, für die er steht (man bekommt so Unteranfragen unter `FROM`).

  Sichten existieren schon in SQL-86. Da aber SQL-86 Unteranfragen unter `FROM` nicht enthielt, gab es komplexe Restriktionen zur Anwendung der Sichten.

- Durch Verwendung von Sichten kann man komplexe Anfragen Schritt für Schritt aufbauen.

  Man überlege aber, ob es das Verständnis wirklich fördert. Sind die einzelnen Schritte allzu klein, oder ist die Bedeutung der jeweiligen Sichten nicht klar, wäre vielleicht eine „monolithische“ Anfrage einfacher.

- Sichten in SQL entsprechen Methoden/Prozeduren in Java: Man bekommt benannten, wiederverwendbaren Code.

Sichten (3)

- Sichten können auch angewendet werden, um Zugriffsrechte einzuschränken.

  So können Zeilen und Spalten ausgeblendet werden, oder nur aggregierte Daten angezeigt werden. Es ist möglich, dass ein Nutzer der Datenbank zwar Leserechte für eine Sicht hat, aber nicht für die zugrundeliegende Tabelle.

- Wenn man betonen will, dass eine Tabelle keine Sicht („virtuelle Tabelle“) ist, spricht man auch von „Basistabelle“.
- Bei einfachen Sichten können auch Updates über die Sicht möglich sein:
  - Update-Befehle wie z.B. `INSERT` werden von der Sicht auf die zugrundeliegende Basistabelle übersetzt.

    Sichten sind „virtuelle Tabellen“ und oft muss der Nutzer gar nicht wissen, dass es nur eine Sicht ist. Zugriffsrechte und „View Updates“ werden in der Fortsetzungs-Vorlesung im Sommersemester behandelt.

Inhalt

Lokale Sichten: WITH (1)

- Man kann eine Sicht auch nur für eine Anfrage definieren („common table expression“, „CTE“):

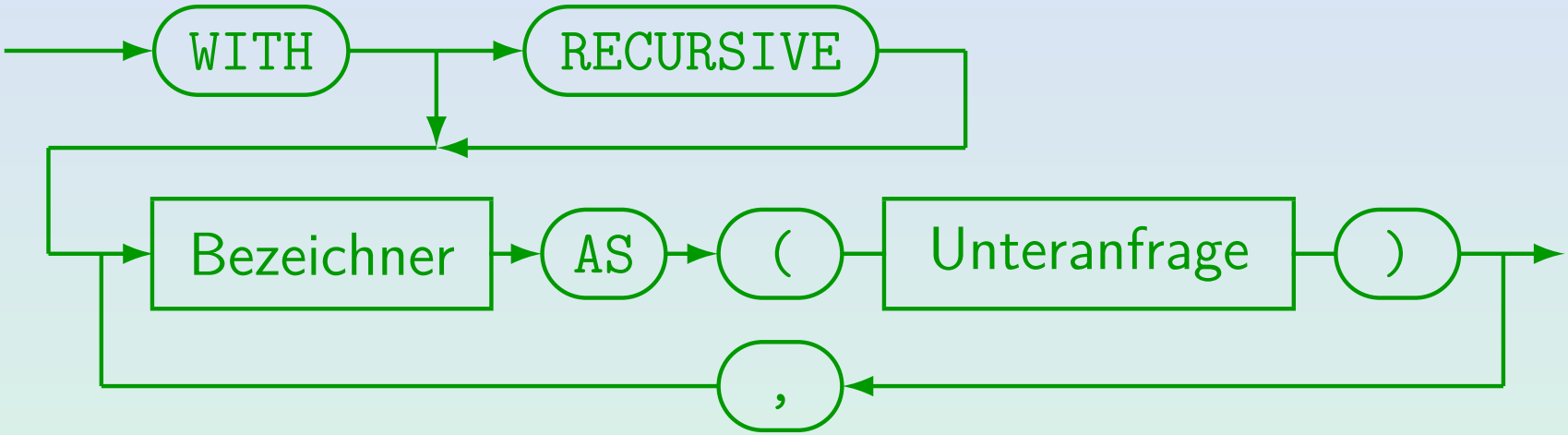
```
WITH   HA_PUNKTE AS
       (SELECT VORNAME, NACHNAME, ANR, PUNKTE
        FROM   STUDENTEN S, BEWERTUNGEN B
        WHERE  S.SID=B.SID AND ATYP = 'H')

SELECT ANR, PUNKTE
FROM   HA_PUNKTE
WHERE  VORNAME='Michael' AND NACHNAME='Grau'
```

- Dies ist besonders nützlich, wenn man diese Unteranfrage mehrfach benötigt.

Lokale Sichten: WITH (2)

Lokale Sicht-Definition (vor SELECT-Anfrage):



- Lokale Sichtdefinitionen mit „WITH“ wurden in SQL-99 eingeführt.
- Bei Oracle heisst es „subquery_factoring_clause“.
- Bei Microsoft SQL Server heisst es „common_table_expression“ (CTE).
  Dort kann man nach dem Sichtnamen in Klammern noch Spaltennamen angeben.
  „RECURSIVE“ entfällt auch bei rekursiven Definitionen.
- Siehe auch: [Wikipedia: Hierarchical and Recursive Queries in SQL].

Lokale Sichten: WITH (3)

Übertreiben Sie es nicht!

- Bei der Klausur-Korrektur ist mir aufgefallen, dass teils exzessiv Hilfstabellen mit WITH genutzt werden.
  - Z.B. insgesamt 25 Zeilen mit 4 Hilfstabellen.
  - Die Musterlösung hatte 15 Zeilen mit einer Hilfstabelle.
- Es entspricht privaten Hilfsmethoden in Java, die nur ein Mal aufgerufen werden.

  Natürlich kann man in Java Hilfsmethoden auch mehrfach aufrufen, so wie man in SQL WITH-Hilfstabellen auch mehrfach in der Anfrage benutzen kann. Dann würde die Schwelle für eine Abspaltung wesentlich niedriger sein.
  - Wie kompliziert sollte der Rumpf mindestens sein?
  - Oder wie relevant und natürlich das benannte Konzept?