Datenbank-Programmierung

Kapitel 9: SQL-Skripte mit psql-Befehlen

Prof. Dr. Stefan Brass

Martin-Luther-Universität Halle-Wittenberg

Sommersemester 2022

http://www.informatik.uni-halle.de/~brass/dbp22/

Lernziele

Nach diesem Kapitel sollten Sie Folgendes können:

- Entscheiden, ob eine Aufgabe mit einem `psql`-Skript gut gelöst werden kann.

  Ist diese Technik zur Anwendungsprogrammierung für eine gegebene Aufgabe die beste Lösung? Das enthält natürlich eine Abwägung mit anderen Techniken, wie JDBC-Programmierung, die später vorgestellt werden.

- `psql`-Skripte schreiben

Inhalt

SQL-Skripte (1)

- Diese Vorlesung heißt ja „Datenbank-Programmierung“.
- Manche Aufgaben können durch Skripte implementiert werden, die von einer SQL-Kommandoschnittstelle wie `psql` ausgeführt werden.

  Ggf. noch mit einem Shellskript drum herum.

- So bekommt man zwar keine hübsche graphische Benutzerschnittstelle, aber die Methode eignet sich z.B. für
  - den wiederkehrenden Export von Daten in verschiedenen Formaten (z.B. CSV, JSON, XML, LaTeX),
  - einfache Berichte aus der Datenbank (Statistiken),
  - wiederkehrende administrative Aufgaben (z.B. Löschen/Archivieren alter Daten).

SQL-Skripte (2)

Erstes Beispiel:

- Man schreibt einen SQL-Befehl in eine Datei `lisa.sql`:

```
SELECT B.ANR, B.PUNKTE
FROM   STUDENTEN S, BEWERTUNGEN B
WHERE  S.VORNAME  = 'Lisa'
AND    S.NACHNAME = 'Weiss'
AND    B.SID = S.SID AND B.ATYP = 'H'
ORDER  BY B.ANR;
```

- In `psql` kann man die Datei ausführen mit:

```
\i lisa.sql
```

- Auf der UNIX/Linux-Kommandozeile mit:

```
psql -f lisa.sql
```

„`-f`“ wie „file“.

Inhalt

Variablen in SQL-Skripten (1)

- Angenommen, das obige Skript soll für einen beliebigen Studenten funktionieren, der über Vorname und Nachname ausgewählt wird.

  D.h. Vorname und Nachname sind Parameter/Eingabewerte des Skripts.

- `psql` hat dafür Variablen:

```
\set Vorname  'Lisa'
\set Nachname 'Weiss'
```

  Da es eine Reihe von vordefinierten Variablen gibt, die nur aus Großbuchstaben bestehen, wird empfohlen, dass Nutzer-Variablen mindestens einen Kleinbuchstaben enthalten. Variablennamen sind case-sensitiv.

- In allen folgenden Kommandos (SQL und psql) wird dann `:Vorname` durch `Lisa` ersetzt (ohne Anführungszeichen).

Variablen in SQL-Skripten (2)

- In der SQL-Anfrage sind Anführungszeichen `'...'` nötig, dafür schreibt man `:'Vorname'`.

  Dagegen würde `':Vorname'` nicht funktionieren, da der Doppelpunkt im Innern von String-Konstanten keine besondere Bedeutung hat. Wenn man den Variablenwert in `"..."` braucht (für Spaltennamen), geht das auch: `:"Var"`.

- Variablen können auch auf der Kommandozeile gesetzt werden (z.B. wichtig für Shell-Skripte):
  `psql -f hw.sql -v Vorname=Lisa -v Nachname=Weiss`

- Das `psql`-Skript kann auch eine Nutzer-Eingabe verlangen:
  `\prompt 'Bitte Vorname eingeben: ' Vorname`

- Ausgaben in `psql`-Skripten gehen mit `\echo`:
  `\echo :Vorname`

Variablen in SQL-Skripten (3)

- Hausaufgaben-Punkte für wählbaren Studenten:

```
-- Beispiel-Skript
\prompt 'Bitte Vornamen eingeben:  ' Vorname
\prompt 'Bitte Nachnamen eingeben: ' Nachname
\echo Hausaufgaben von :Vorname :Nachname

SELECT B.ANR, B.PUNKTE
FROM   STUDENTEN S, BEWERTUNGEN B
WHERE  S.VORNAME = :'Vorname'
AND    S.NACHNAME = :'Nachname'
AND    B.SID = S.SID
AND    B.ATYP = 'H' -- Hausaufgaben-Punkte
ORDER  BY B.ANR;
```

Variablen in SQL-Skripten (4)

Variable auf Anfrage-Ergebnis setzen:

- Mit `\gset` (seit Ver. 9.3) werden Variablen auf das Ergebnis der Anfrage im Puffer gesetzt (darf nur eine Zeile liefern):

```
SELECT SUM(B.PUNKTE) AS "Gesamt"
FROM   STUDENTEN S, BEWERTUNGEN B
WHERE  S.VORNAME = :'Vorname'
AND    S.NACHNAME = :'Nachname'
AND    B.SID = S.SID
AND    B.ATYP = 'H' -- Hier kein ;
\gset
\r
```

- Die Variable `Gesamt` enthält nun die Gesamtpunktzahl.

  Die Variablen heißen wie die Ergebnisspalten (besser `"..."` verwenden).

  „`\gset x_`“: Präfix „`x_`“ für Variablen. Falls 0 Zeilen: Variablen unverändert.

Inhalt

Ausgaben mit \echo:

- \echo druckt mehrere durch Leerplatz getrennte Argumente (Worte auf der Zeile) mit jeweils einem Leerzeichen getrennt.

  Unabhängig davon, wie viele Leerzeichen zwischen den Argumenten standen.

- Die Kontrolle über Leerplatz ist mit Stringkonstanten möglich:
  \echo 'X   Y'

  Dies gibt wirklich mehrere Leerzeichen zwischen X und Y aus.

  Die Anführungszeichen '...' werden nicht ausgegeben ("..." dagegen schon).

- Wenn man keinen Zeilenumbruch am Ende wünscht, kann man die Option -n als erstes Argument angeben:
  \echo -n 's'

  Dies gibt exakt den String *s* aus und nichts sonst. Z.B. „:“ direkt nach Namen:

```
\echo -n 'Hausaufgaben von' :Vorname :Nachname
\echo ':'
```

Ausgaben in Datei schreiben:

- `\o ⟨file⟩` („output destination“) leitet die Ausgabe der folgenden Anfragen in die Datei ⟨file⟩ um.

  Mit `\o |⟨cmd⟩` kann man die Ausgabe auch als Eingabe für ein beliebiges UNIX/Linux-Kommando verwenden („pipe“ wie in der Shell).

- `\qecho '⟨text⟩'` („query echo“) schreibt ⟨text⟩ dorthin, wohin auch Anfrage-Ergebnisse geschrieben werden (Datei).

  Dagegen schreibt der normale `\echo`-Befehl auch nach der Ausgabe-Umleitung mit `\o` weiter auf die Standard-Ausgabe (normalerweise den Bildschirm).
  Die Ausgabe kann von einem beliebigen Programm stammen: `\qecho ‘⟨prog⟩‘`
  (wenn man „backticks“ verwendet, wird der Linux-Befehl darin ausgeführt).

- Es ist möglich, in die Ausgabe-Datei SQL-Anweisungen zu schreiben, und diese anschließend mit `\i` auszuführen.

  Auf diese Art bekommt man auch eine Art von Schleifen.

Inhalt

Ausgabe-Formatierung (1)

- Das Standard-Ausgabeformat von Tabellen ist so:

```
sb=> select * from studenten;
 sid | vorname | nachname |         email
-----+---------+----------+-----------------------
 101 | Lisa    | Weiss    | weiss@acm.org
 102 | Michael | Grau     |
 103 | Daniel  | Sommer   | daniel@gmx.de
 104 | Iris    | Winter   | irisw@gmail.com
(4 rows)
```

- Mit Rot hervorgehoben ist der Rahmen der Tabelle (erste zwei Zeilen der Ausgabe und Zusammenfassung am Ende).
- Diesen kann man abstellen mit: `\pset tuples_only on`

Ausgabe-Formatierung (2)

- Normal werden die Datenwerte mit Leerzeichen aufgefüllt, so dass die Werte einer Spalte übereinander stehen.
  Das ist das Ausgabeformat „aligned“:

```
␣101␣|␣Lisa␣␣␣␣|␣Weiss␣␣␣␣|␣weiss@acm.org
␣102␣|␣Michael␣|␣Grau␣␣␣␣␣|␣
␣103␣|␣Daniel␣␣|␣Sommer␣␣␣|␣daniel@gmx.de
␣104␣|␣Iris␣␣␣␣|␣Winter␣␣␣|␣irisw@gmail.com
```

- Beim Format „unaligned“ werden keine extra Leerzeichen eingefügt (`\pset format unaligned`):

```
101|Lisa|Weiss|weiss@acm.org
102|Michael|Grau|
...
```

`\pset fieldsep '\t'` ersetzt den „|“ durch ein anderes Zeichen (z.B. TAB).

Ausgabe-Formatierung (3)

- Wenn die Tupel breiter als eine Terminal-Zeile sind, kann man "`\pset format wrapped`" probieren. Spaltenwerte werden dann teils über mehrere Zeilen verteilt.

  Das wird durch einen Punkt am Ende jeder Zeile eines aufgespaltenen Wertes und einen Punkt am Anfang der nächsten Zeile symbolisiert. Die Spalten bleiben aber mindestens so breit wie der Spalten-Name.

- Wenn viel zu breit: `\pset expanded on`

```
* Record 1
sid      101
vorname  Lisa
nachname Weiss
email    weiss@acm.org
* Record 2
...      ...
```

Ausgabe-Formatierung (4)

- `\pset format html` liefert Ausgaben im HTML-Format.

  Damit könnte man Anfrage-Ergebnisse in einfache Webseiten einbauen, ggf. sogar mit dem Skript die ganze Webseite generieren.

- `\pset format latex` liefert Ausgaben im LaTeX-Format.

  Damit kann man die Tabelle z.B. leicht in Abschlussarbeiten einbauen.

- `\pset footer off` stellt „(*n* `rows`)“ am Ende ab.

  Dagegen würde `tuples_only` zusätzlich auch den Kopf entfernen.

- Man kann natürlich auch SQL-Anfragen schreiben, die LaTeX oder HTML generieren.

  Dazu würde man Strings mit `||` konkatenieren und ggf. mehrere Teile mit `UNION ALL` zusammenfügen. Spalten für die Ausgabe-Sortierung müssen beim `UNION` noch vorhanden sein, aber man kann es in eine Unteranfrage/`WITH` stecken, und dann nach einer Spalte sortieren, die nicht im `SELECT` auftaucht.

Ausgabe-Formatierung (5)

- Angenommen, man hat eine SQL-Anfrage, die nur eine Ausgabespalte hat, und man möchte genau die erzeugten Daten in eine Datei schreiben:

```
\pset tuples_only on
\pset format unaligned
\o ⟨File⟩
SELECT ...;
-- Bei Bedarf alles zurückschalten:
\o
\pset format aligned
\pset tuples_only off
```

`unaligned` ist auch bei einer Spalte wichtig: Sonst werden Zeilenumbrüche in den Daten mit „+“ markiert (und Leerzeichen vor/nach jedem Wert gedruckt).

Für einige häufige Optionen gibt es auch kurze Spezialbefehle, z.B. schaltet `\t` die Option `tuples_only` immer um zwischen `on` und `off`.

Inhalt

Ausgabe-Formatierung (1)

- Es soll eine Ausgabe-Datei der folgenden Form erstellt werden:

```
Hausaufgaben-Ergebnisse:

=======================================
Grau, Michael:
      1:  9.0
      2:  9.0
  Summe: 18.0
=======================================
Sommer, Daniel:
      1:  5.0
  Summe:  5.0
=======================================
...
```

Ausgabe-Formatierung (2)

- Zum Ausprobieren:
  [https://www.informatik.uni-halle.de/~brass/dbp22/sql/script.sql]
- `script.sql` (Anfang):

```
-- Kein Tabellen-Rahmen:
\pset tuples_only on
\pset format unaligned
-- Ausgabe in Datei "ha_punkte.txt":

\o ha_punkte.txt

-- Ueberschrift und Leerzeile:
SELECT 'Hausaufgaben-Ergebnisse:';
SELECT '';
```

Ausgabe-Formatierung (3)

- `script.sql` (Teil 2/6):

```
-- Ausgabestuecke fuer jeden Studenten:
SELECT AUSGABE FROM (
-- Trennstrich:
SELECT NACHNAME, VORNAME,
       0 AS STUECK_NR, 0 AS ANR,
       '==============================' ||
       '=============================='
       AS AUSGABE
FROM STUDENTEN
UNION ALL
```

Am Ende wird nur die Spalte `AUSGABE` gedruckt. Die anderen Spalten dienen der Sortierung der Ausgabe-Stücke. Haupt-Sortierkriterium sind Nachname und Vorname des Studierenden, danach kommt die Stück-Nummer. Die Aufgaben-Nummer ist nur innerhalb eines Stücks nötig.

Ausgabe-Formatierung (4)

- `script.sql` (Teil 3/6):

```
-- Name, Vorname:
SELECT NACHNAME, VORNAME,
       1 AS STUECK_NR, 0 AS ANR,
       NACHNAME || ', ' || VORNAME || ':'
       AS AUSGABE
FROM STUDENTEN
UNION ALL
```

Durch die `STUECK_NR` 1 wird diese Ausgabezeile hinter der Ausgabezeile mit der `STUECK_NR` 0 eingeordnet (in einer Art „Schleife“ über allen Studenten, die ein höher priorisiertes Sortier-Kriterium sind).

Ausgabe-Formatierung (5)

- `script.sql` (Teil 4/6):

```
-- Einzelergebnisse (Punkte pro Aufgabe):
SELECT NACHNAME, VORNAME, 2 AS STUECK_NR, ANR,
       '    ' || TO_CHAR(ANR, '90') || ':' ||
           TO_CHAR(PUNKTE, '90.9')
       AS AUSGABE
FROM STUDENTEN S, BEWERTUNGEN B
WHERE S.SID = B.SID AND B.ATYP = 'H'
UNION ALL
```

Mit `TO_CHAR` wird eine Ausgabe-Formatierung erreicht. Das zweite Argument ist das Format. Für jede Ziffer `9` im Format wird eine Dezimalziffer des Ergebnisses ausgegeben. Bei der Ziffer `0` im Format werden auch führende Nullen gedruckt.
[https://www.postgresql.org/docs/current/functions-formatting.html]

Ausgabe-Formatierung (6)

- `script.sql` (Teil 5/6):

```
-- Summe aller Hausaufgaben-Punkte:
SELECT NACHNAME, VORNAME,
       3 AS STUECK_NR, 0 AS ANR,
       '  Summe:' ||
       TO_CHAR(COALESCE(SUM(PUNKTE),0.0), '90.9')
       AS AUSGABE
FROM STUDENTEN S LEFT JOIN BEWERTUNGEN B
     ON S.SID = B.SID AND B.ATYP = 'H'
GROUP BY S.NACHNAME, S.VORNAME
) X
ORDER BY NACHNAME, VORNAME, STUECK_NR, ANR;
```

Die Tupelvariable (hier `X` für die Unteranfrage ist syntaktisch nötig.

Ausgabe-Formatierung (7)

- `script.sql` (Schluss):

```
-- Wieder Standard-Ausgabe (Terminal):
\o
-- Normale Ausgabe von Tabellen (mit Rahmen):
\pset format aligned
\pset tuples_only off
```

Wenn das Skript mit „`psql -f script.sql`“ ausgeführt wird, ist die Rücksetzung der Ausgabeformatierung nicht nötig. Wenn es aber mit „`\i script.sql`“ in `psql` ausgeführt wird, ist es natürlich nützlich, wenn der normale Ausgabemodus auch hinterher wieder gilt.

Literatur/Quellen

- Wikipedia: PostgreSQL (englisch, deutsch)
  [https://en.wikipedia.org/wiki/PostgreSQL]
  [https://de.wikipedia.org/wiki/PostgreSQL]
- Homepage des Projekts
  [https://www.postgresql.org/]
- Deutsche PostgreSQL Homepage
  [http://www.postgres.de/]
- PostgreSQL Dokumentation: `psql`
  [https://www.postgresql.org/docs/9.5/app-psql.html]
- PostgreSQL Tutorial:
  [http://www.postgresqltutorial.com/]
  PSQL Commands: [http://www.postgresqltutorial.com/psql-commands/]
- pgAdmin (GUI zur Administration, auch SQL-Zugang)
  [https://www.pgadmin.org/]
- PostgreSQL Documentation 11: Chapter 37: The Information Schema
  [https://www.postgresql.org/docs/current/information-schema.html]