JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Laborprotokoll

DezSys02: Verteilte Datenbanken / Postgres

Systemtechnik Labor
5BHITT 2015/16, Gruppe X

Stefan Erceg

Version 1.0

Note:

Betreuer: Prof. Micheler

Begonnen am 09. Oktober 2015

Beendet am 22. Oktober 2015

Inhaltsverzeichnis

1 Einführung

Diese Übung zeigt, wie Fragmente einer verteilten Datenbank einer Station zugewiesen werden und wie diese Zuteilung in Postgres umgesetzt werden kann.

1.1 Ziele

Das Ziel dieser Übung ist die Allokation im Zuge des "Datenbankentwurfs einer verteilten Datenbank" zu üben und anhand eines DBMS zu vertiefen. Es werden mindestens zwei Datenbankinstanzen installiert, denen im Anschluss die einzelnen Fragmente zugeteilt werden. Diese Zuteilung soll fuer alle Fragmentierungsarten durchgeführt werden.
Die Funktionsweise und das Handling von verteilten Datenbanken soll im Anschluss mit SELECT Statements gezeigt werden.

1.2 Voraussetzungen

- Grundlagen von verteilten Datenbanken
- Laden der Demo Datenbank "Dell DVD Store"
- Fragmentierung der Demo Datenbank nach alle 3 Fragmentierungsarten
- SQL Kenntnisse
- Installation von mindestens zwei Postgres Datenbankservern (Version > 9.1)
- für manche Linux-Versionen: Installation eines zusätzlichen Pakets: postgresql-contrib-9.x
- Konfiguration Postgres Foreign Data Wrapper: http://www.postgresql.org/docs/9.4/static/postgres-fdw.html

1.3 Aufgabenstellung

Die Fragmente der Datenbank aus der Aufgabenstellung DezSys-01 sollen zwischen zwei Postgres Datenbankservern verteilt werden.

Folge den Instruktionen der Praesentation, um den Zugriff auf eine entfernte Postgres Instanz einzurichten. Als entfernte Einheit kann auch eine VM-Instanz verwendet werden.

Nach erfolgreicher Konfiguration sollen die Fragmente der 3 Fragmentierungsarten

- horizontal
- vertical
- combination

und jeweils einer der beiden Postgres Instanzen zugeordnet werden. Ein Fragment soll nach der Zuteilung nur auf einer Instanz vorhanden sein. Zur Unterscheidung, ob ein Fragment lokal oder remote vorhanden ist, sollen alle entfernte Ressourcen die Bezeichnung "remote" (Bsp. horizontal.remote_orders_q12) enthalten.

Im Anschluss soll zu jeder Fragmentierungsart von der lokalen Datenbank ein SELECT Statement zum Sammeln aller Daten entworfen werden. Dabei soll gezeigt werden, dass bei der Verteilung keine Datensätze verloren gegangen sind. Verwenden Sie dazu "SELECT count(*) FROM ...."

- Anzahl der Datensätze vor der Fragmentierung
- Anzahl der Datensätze der einzelnen Fragmente
- Anzahl der Datensätze aus dem SELECT statement, das alle Daten wieder zusammenfügt

Protokollieren Sie alle Arbeitsschritte, die SELECT Anweisungen und deren Resultate.

Bewertung: 16 Punkte
- Dokumentation der einzelnen Arbeitsschritte (4 Punkte)
- Zuteilung der Fragmente aller 3 Fragmentierungsarten (6 Punkte)
- SELECT statement zum Zusammenfügen und count(*) Resultat (6 Punkte)

2 Ergebnisse

2.1 Vorbereitungen beim Remote Server

Damit die Aufgabe durchgeführt werden kann, wurde die Debian 8 VM, welche bei der letzten Aufgabe verwendet wurde, geklont. Die geklonte VM stellt somit die Remote Postgres-Instanz dar.

In der Remote VM mussten dann in 2 Postgres-Konfigurationsfiles Änderungen vorgenommen werden, um dann später eine Verbindung vom lokalen zum remote Server zu gewährleisten:

- Zu Beginn wurde im File „postgresql.conf", welches sich im Verzeichnis „/etc/postgresql/9.4/main" befindet, die Zeile

  ```
  listen_addresses = 'localhost'
  ```

  auf

  ```
  listen_addresses = '*'
  ```

  umgeändert.

- Ebenfalls wurde das File „pg_hba.conf", welches sich im Verzeichnis „/etc/postgresql/9.4/main" befindet, folgendermaßen bearbeitet:

```
# TYPE  DATABASE        USER            ADDRESS                 
local   all             all                                     peer
host    all             all             127.0.0.1/32            md5
host    all             all             ::1/128                 md5
host    ds2             ds2             samenet                 md5
```

Nach den Änderungen in den beiden Files musste man den Postgres-Server mittels `sudo service postgresql restart` neustarten.

2.2 Vorbereitungen beim Local Server

Beim Local Server habe ich mich zu Beginn mit dem Befehl

```
sudo psql –U ds2 –W ds2
```

als User „ds2” in PostgreSQL angemeldet.

Folgende Schritte wurden danach durchgeführt:

- Eine neue Extension wurde in PostgreSQL erstellt:

  ```
  CREATE EXTENSION postgres_fdw;
  ```

- Der Remote Server wurde mit seiner IP-Adresse, seinem Port und dem Datenbanknamen von der Datenbank, welche wir aufteilen wollen, definiert:

  ```
  CREATE SERVER foreign_server
  FOREIGN DATA WRAPPER postgres_fdw
  OPTIONS
  (host '192.168.164.130', port '5432', dbname 'ds2');
  ```

- Damit der User „ds2” zum Remote Server gemappt werden kann, wird folgender CREATE-Befehl durchgeführt:

  ```
  CREATE USER MAPPING FOR ds2
  SERVER foreign_server
  OPTIONS (user 'ds2', password 'ds2');
  ```

2.3 Definieren der Foreign Tabellen

Zum Erstellen der Foreign Tabellen wurden bestimmten Tabelle von jeder Fragmentierungsart aus den dump-Files, die bei der letzten DezSys-Übung erstellt wurden, herausgenommen und diese dem Remote Server zugewiesen.

2.3.1 horizontale Fragmentierung

Hier wurde die Tabelle „customersover30fromrest" dem Remote Server zugewiesen:

```
CREATE FOREIGN TABLE horizontal.remote_customersover30fromrest (
    customerid integer,
    firstname character varying(50),
    lastname character varying(50),
    address1 character varying(50),
    address2 character varying(50),
    city character varying(50),
    state character varying(50),
    zip character varying(9),
    country character varying(50),
    region smallint,
    email character varying(50),
    phone character varying(50),
    creditcardtype integer,
    creditcard character varying(50),
    creditcardexpiration character varying(50),
    username character varying(50),
    password character varying(50),
    age smallint,
    income integer,
    gender character varying(1)
)
SERVER foreign _server
OPTIONS
(schema_name 'horizontal', table_name 'customersover30fromrest');
```

2.3.2 vertikale Fragmentierung

Hier wurde die Tabelle „ordersgeneralinfo” dem Remote Server zugewiesen:

```
CREATE FOREIGN TABLE vertical.remote_ordersgeneralinfo (
    orderid integer,
    orderdate date,
    customerid integer,
    totalamount numeric
)
SERVER foreign_server
OPTIONS
(schema_name 'vertical', table_name 'ordersgeneralinfo');
```

2.3.3 kombinierte Fragmentierung

Hier wurde die Tabelle „femalecustomers” dem Remote Server zugewiesen:

```
CREATE FOREIGN TABLE combination.remote_femalecustomers (
    customerid integer,
    firstname character varying(50),
    lastname character varying(50),
    address1 character varying(50),
    address2 character varying(50),
    city character varying(50),
    state character varying(50),
    zip character varying(9),
    country character varying(50),
    region smallint,
    email character varying(50),
    phone character varying(50),
    username character varying(50),
    password character varying(50),
    age smallint,
    income integer,
    gender character varying(1)
)
SERVER foreign_server
OPTIONS
(schema_name 'combination', table_name 'femalecustomers');
```

2.4 Durchführung der SELECT-Befehle

2.4.1 horizontale Fragmentierung

Bei der horizontalen Fragmentierungsart besitze ich 4 einzelne Fragmentierungen. Es wurde zu Beginn bei jeder Fragmentierung überprüft, wie viele Datensätze sie jeweils besitzt:

```
ds2=# SELECT COUNT(*) FROM horizontal.remote_customersover30fromrest;
 count
-------
  8207
(1 row)
```

```
ds2=# SELECT COUNT(*) FROM horizontal.customersover30fromus;
 count
-------
  8156
(1 row)
```

```
ds2=# SELECT COUNT(*) FROM horizontal.customersunder30fromrest;
 count
-------
  1793
(1 row)
```

```
ds2=# SELECT COUNT(*) FROM horizontal.customersunder30fromus;
 count
-------
  1844
(1 row)
```

8207 + 8156 + 1793 + 1844 ergibt 20 000 Datensätze, daher sollten in der customers-Tabelle, die fragmentiert wurde, und beim SELECT-Statement, das alle Daten wieder zusammenführt, ebenfalls **20 000 Datensätze** herauskommen:

```
ds2=# SELECT COUNT(*) FROM customers;
 count
-------
 20000
(1 row)
```

```
ds2=# SELECT (SELECT COUNT(*) FROM horizontal.remote_customersover30fromrest) +
ds2-# (SELECT COUNT(*) FROM horizontal.customersover30fromus) +
ds2-# (SELECT COUNT(*) FROM horizontal.customersunder30fromrest) +
ds2-# (SELECT COUNT(*) FROM horizontal.customersunder30fromus)
ds2-# AS horizontal_result;
 horizontal_result
-------------------
             20000
(1 row)
```

2.4.2 vertikale Fragmentierung

Bei der vertikalen Fragmentierungsart besitze ich 2 einzelne Fragmentierungen. Es wurde zu Beginn bei jeder Fragmentierung überprüft, wie viele Datensätze sie jeweils besitzt:

```
ds2=# SELECT COUNT(*) FROM vertical.remote_ordersgeneralinfo;
 count
-------
 12000
(1 row)
```

```
ds2=# SELECT COUNT(*) FROM vertical.ordersadditionalinfo;
 count
-------
 12000
(1 row)
```

Da bei der vertikalen Fragmentierung die Spalten (= Attribute) fragmentiert werden, sollten in der orders-Tabelle, die fragmentiert wurde, ebenfalls **12 000 Datensätze** herauskommen:

```
ds2=# SELECT COUNT(*) FROM orders;
 count
-------
 12000
(1 row)
```

Beim SELECT-Statement, das alle Daten wieder zusammenführt, sollte nun die doppelte Anzahl an Datensätzen, nämlich **24 000**, existieren, da die Tabelle in 2 Fragmente aufgeteilt wurde:

```
ds2=# SELECT (SELECT COUNT(*) FROM vertical.remote_ordersgeneralinfo) +
ds2-# (SELECT COUNT(*) FROM vertical.ordersadditionalinfo)
ds2-# AS vertical_result;
 vertical_result
-----------------
           24000
(1 row)
```

2.4.3 kombinierte Fragmentierung

Bei der kombinierten Fragmentierungsart besitze ich 3 einzelne Fragmentierungen. Es wurde zu Beginn bei jeder Fragmentierung überprüft, wie viele Datensätze sie jeweils besitzt:

```
ds2=# SELECT COUNT(*) FROM combination.remote_femalecustomers;
 count
-------
  9955
(1 row)
```

```
ds2=# SELECT COUNT(*) FROM combination.malecustomers;
 count
-------
 10045
(1 row)
```

```
ds2=# SELECT COUNT(*) FROM combination.customerscreditcarddetails;
 count
-------
 20000
(1 row)
```

Die customers-Tabelle, welche hier fragmentiert wurde, sollte **20 000 Datensätze** besitzen, da die Tabelle „customerscreditcarddetails" vertikal fragmentiert wurde und bei dieser ebenfalls 20 000 existieren:

```
ds2=# SELECT COUNT(*) FROM customers;
 count
-------
 20000
(1 row)
```

Beim SELECT-Statement, das alle Daten wieder zusammenführt, sollte nun die doppelte Anzahl an Datensätzen, nämlich **40 000**, existieren, da die Tabelle einmal horizontal und einmal vertikal fragmentiert wurde:

```
ds2=# SELECT (SELECT COUNT(*) FROM combination.remote_femalecustomers) +
ds2-# (SELECT COUNT(*) FROM combination.malecustomers) +
ds2-# (SELECT COUNT(*) FROM combination.customerscreditcarddetails)
ds2-# AS combination_result;
 combination_result
--------------------
              40000
(1 row)
```