JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Laborprotkoll

Verteilte Datenbanken

Dezentrale Systeme
5YHITM 2016/17

Maximilian Seidl

Version 0.1
Begonnen am 6. Oktober 2016
Beendet am 6. Oktober 2016

Note:
Lehrer: T. Micheler

Inhaltsverzeichnis

1 Einführung

Diese Übung soll einen Einblick in das Themengebiet der „Verteilte Datenbanken" geben. Zunächst wird der verteilte Datenbankentwurf erläutert und in weiterer Folge wird der theoretische Ansatz anhand einer Beispieldatenbank geübt. Hier werden Fragmente einer verteilten Datenbank einer Station zugewiesen, danach erfolgt die Zuweisung zu Postgres-Instanzen.

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 durchgefuehrt 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)
- fuer manche Linux-Versionen: Installation eines zusaetzlichen Pakets: postgresql-contrib-9.x
- Konfiguration Postgres Foreign Data Wrapper: `http://www.postgresql.org/docs/9.4/static/postgres-fdw.html`

1.3 Aufgabenstellung

Unter Verwendung der Sample Database „Dell DVD Store" soll eine lokale Datenbank in eine verteilte Datenbank transferiert werden. Hierbei soll die Fragmentierung einer Datenbank durchgefuehrt werden. Basierend auf einer Tabelle/View des DVD Stores sollen folgende Fragmentierungsarten umgesetzt werden:

- horizontale Fragmentierung nach mindestens 2 Kriterien
- vertikale Fragmentierung
- kombinierte Fragmentierung

Die Fragmente sollen jeweils in einem Schema mit der Bezeichnung

- Schema horizontal
- Schema vertical
- Schema combination

in Tabellen mit sinnvollen Namen gespeichert werden. Die Fragmentierung soll unter selbst definierten Annahmen spezifiziert werden. Die Annahmen sollen im Protokoll festgehalten werden.

Die Fragmente der Datebank aus dem ersten Teil der Aufgabenstellung 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 Resourcen 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 Datensaetze verloren gegangen sind. Verwenden Sie dazu „SELECT count(*) FROM ...."

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

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 Zusammenfuegen und count(*) Resultat (6 Punkte)

2 Ergebnisse

2.1 Aufbau der Entwicklungsumgebung

2.1.1 Virtuelle Maschine

Wie in der Aufgabenstellung vorgesehen, habe ich mir zwei virtuelle Maschinen in Virtual Box aufgesetzt. Das verwendete Betriebssystem ist hierbei *ubuntu-server 14.04*, welches ich von der offiziellen Ubuntu-Seite heruntergeladen habe. Da das Aufsetzen einer VM hier nicht in die Dokumentation gehört, springe ich direkt in ein fertiges Betriebssystem.

2.1.2 Netzwerkkonfiguartion

Damit die Kommunikation zwischen den beiden virtuellen Maschinen und dem Host gewährleistet wird, müssen die Netzwerkkarten der Server auf **Netzwerkbrücke/Networkbridge** umgestellt werden. Des Weiteren muss der **"Promiscuous-Modus"** auf **"erlauben für alle VMs und den Host"** gestellt werden, damit eine fehlerfreie Übertragung möglich ist.

2.1.3 PostgreSQL Installation

Auf beiden VMs wird PostgreSQL aufgesetzt, um die Datenbank zu verteilen. Zunächst wird das benötigte Package in einem Packet-Manager heruntergeladen und installiert. Bei Ubuntu handelt es sich hierbei um *aptitude*.

```
1 sudp apt-get install postgreSQL
```

Listing 1: PostgreSQL Installation

2.1.4 Datenbankkonfiguration - User, Rechte

Bei der Erstkonfiguration von PostgreSQL wird ein default-User mit dem Namen *postgres* angelegt. Mit diesem Benutzer stellt man eine Verbindung zum PostgreSQL-Server her. Danach führt man folgende Schritte durch, um sich den User für diese Aufgabe mit den entsprechenden Rechten anzulegen.

```
1
  postgres=# CREATE ROLE ds2 WITH superuser login;
3 postgres=# ALTER ROLE ds2 PASSWORD 'ds2';

5 seidl@seidl:~$ psql -U ds2 -d ds2
```

Listing 2: User, Rechte anlegen

Als nächstes muss noch das Konfigurations-File so bearbeitet werden, dass die Datenbank Zugriffe von allen Seiten annimmt. Hierbei werden die Files **/etc/postgresql/9.5/main/postgresql.conf** und **pg-hba.conf** bearbeitet. Im folgenden Listing werden die einzutragenden Kommandos dargestellt:

```
1 host    ds2     ds2     samenet     trust

3 listen_addresses= '*'
```

Listing 3: PostgreSQL Konfiguration

2.2 PostgreSQL Remote-Server Konfiguration

Jetzt muss nur noch der Remote-Server richtig den Local-Server als **foreign-server** eingetragen werden. Dies erfolgt mit folgenden Kommandos in der psql-Konsole:

```
1   CREATE EXTENSION postgres_fdw;
    CREATE SERVER foreign_server
3   FOREIGN DATA WRAPPER postgres_fdw
    OPTIONS
5       (host '10.0.106.18', port '5432', dbname 'ds2');
    CREATE USER MAPPING FOR ds2
7   SERVER foreign_server
    OPTIONS(user 'ds2', password 'ds2');
```

Listing 4: Remote-Server Konfiguration

2.3 Horizontale Fragmentierung

Für die horizontale Fragmentierung habe ich mir bei der Tabelle **customers** überlegt, mit folgenden Trennungen:

- gender = 'F' - Bedeutet alle Einträge welche dem weiblichen Geschlecht angehören
- age $<$ 21 - und jünger als 21 sind.

Damit die Fragmentierung möglich ist, muss man eine **FOREIGN TABLE** am Remote-Server erstellen. Diese Tabellen holen sich dann die Daten von dem Local-Server und trägt diese in die Tabelle ein. Mit einem simplen SELECT-Statement kann man das ganze überprüfen:

```
SELECT COUNT(*) FROM horizontal.remote_customers_f_under_21;
```

Listing 5: Horizontale Fragmentierung

2.4 Vertikale Fragmentierung

Für die vertikale Fragmentierung erfolgt ebenfalls eine Trennung der Tabelle **customers**, aber diesmal werden die Spalten getrennt. Die Tabelle wird unterteilt in **payment-info** und **non-payment-info**. Zu beachten bei der vertikalen Fragmentierung ist das Eintragen der PRIMARY KEYS in **beide Tabellen** (sowhol FOREIGN als auch local).

```
1 INSERT INTO vertical.customers_payment_info
  SELECT customerid, creditcardtype, creditcard, creditcardexpiration, age, income FROM customers;
```

Listing 6: Vertikale Fragmentierung

2.5 Kombinierte Fragmentierung

Die kombinierte Fragmentierung ist die Mischung einer horizontalen, sowie einer vertikalen Fragmentierung. Ich habe mir hierfür folgende Unterteilung überlegt:

- Vertikale Fragmentierung

  Vertrauliche Daten z.B.: Kreditkarte, Passwort

- Horizontale Fragmentierung

  Alle Kunden aus den US werden getrennt von den Kunden außerhalb.

3 SELECT-Resultate

Horizontale Fragmentierung

```
  SELECT COUNT(*) FROM horizontal.customers_f_under_21;
2 count
  -------
4 429
  (1 row)
6
  SELECT COUNT(*) FROM horizontal.remote.customners_f_under_21;
8 count
  -------
10 429
  (1 row)
```

Listing 7: SELECT-horizontal

kombinierte Fragmentierung

```
1 SELECT * FROM combined.customers_us_name_phone
  NATURAL JOIN
3 combined.remote_customer_non_us_name_phone
  UNION
5 SELECT * FROM combined.customer_non_us_name_phone
  NATURAL JOIN
7 combined.remote_customer_us_name_phone;
```

Listing 8: SELECT-combined

Als Resultat bekommt man wieder eine vollständige **customer**-Tabelle.

3.1 Zeitaufwand

Zeitaufwand	Task
4h	Schule Labor, Fragmentierungsmethoden ueberlegt
3h	Entwicklungsumgebung und Serverkonfiguration + PostgreSQL
5h	Umsetzung der Fragmentierungsvarianten

Tabelle 1: Zeitaufzeichnung

Listings

Tabellenverzeichnis