Prac1 Distributed Databases

Introduction

Learning Objectives:

- Simulate horizontal and vertical fragmentation using centralized database. Understand how to update records on a distributed database with data replications.
- Apply semi-join algorithm to simulate data transmission cost reduction over computer networks. Understand why semi-join could be faster sometimes.

Part 1: Distributed DB Design

In part 1, we aim to simulate a distributed database using a standalone computer. To do so, we are acting as a global site, which possess the global conceptual schema, and data replication info. The global site deals with all the incoming queries. Then we create multiple databases in Postgresql, each of which represents a distributed local site. The data transferred among relations belonging to different user accounts corresponds to the data transferred over computer network among different sites. Given a global conceptual schema, we perform the following tasks.

Part 1.1 Database Fragmentation

1.1.1: Horizontal fragmentation

The Sales table is split into 10 fragments:

- Sales_h1: 1≤ SalesID ≤ 700000
- Sales_h2: 700000≤ SalesID < 1400000
- …
- Sales_h10: 6300000≤ SalesID≤ 7000000

To load fragments into database, connect to the Sales database and run script "sales_horizontal.sql".

```
SalesDB=# \i ./Part1/sales_horizontal.sql
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
TRUNCATE TABLE
INSERT 0 6715221
SalesDB=#
```

1.1.2: Vertical fragmentation

The Sales table is split into 2 fragments:

• Sales_v1: (SalesID, CustomerID)

- Sales_v2: (SalesID, SalesPersonID, ProductID, Quantity)

We provide you with the script "sales_v1.sql" which loads fragments into Sales_v1. **You need to complete the script "sales_v2.sql" by yourself.**

```
SalesDB=#
SalesDB=# \i ./Part1/sales_v1.sql
CREATE TABLE
INSERT 0 6715221
```

To load fragments into database, connect to the Sales database and run script "sales_v1.sql" and "sales_v2.sql".

Task: Count the total number of transactions that satisfy sales ID less than 500000 and processed by sales person 6. Compare the query plan of querying horizontal fragmented table, vertical fragmented table and the table without fragmentation.

Part 1.2 Database Replication

There are three types of replication strategies: Full Replication, Partial Replication, and No replication. In this practical, we simulate the distributed database systems using a single database in PostgreSQL. We represent distributed databases as **tables**, with each table containing all the fragmentation data in one actual distributed database.

You are asked to simulate these three strategies on a 'distributed' database that contains three local sites (DB1, 2 and 3). Each strategy requires 3 Tables for simulation, that is to say, you need to create 9 tables in total for this task (In PSQL, you should create 3 databases as 3 strategies, and 3 tables in each database as "distributed databases".).
Based on the product id, we partition the products table into 3 fragments. After creating the databases, you are asked to load data fragments into the database according to the replication rules.

1.2.1 Full replication

Each fragmentation will be a relation located on every site in the computer network (i.e., each site has a full copy of each fragment). You should create three sites (tables) to simulate the full replication in psql command line.

- Site1_S1234567
- Site2_S1234567
- Site3_S1234567

To load data into database FULL_S1234567, first, you need to create database FULL_S1234567. Then, you need to load the data fragmentation (.csv format) from the folder "…/P2/Part1/". Repeat the same process for other sites.

```
SalesDB=# create database "FULL_uqbyuan3";
CREATE DATABASE
SalesDB=#
SalesDB=# \c "FULL_uqbyuan3";
You are now connected to database "FULL_uqbyuan3" as user "uqbyuan3".
FULL_uqbyuan3=#
```

```
FULL_uqbyuan3=# create table site1_uqbyuan3 (ProductID serial Primary key, Name varchar(50) NOT NULL, Price numeric(19, 4) DEFAULT NULL);
CREATE TABLE
FULL_uqbyuan3=# create table site2_uqbyuan3 (ProductID serial Primary key, Name varchar(50) NOT NULL, Price numeric(19, 4) DEFAULT NULL);
CREATE TABLE
FULL_uqbyuan3=# create table site3_uqbyuan3 (ProductID serial Primary key, Name varchar(50) NOT NULL, Price numeric(19, 4) DEFAULT NULL);
CREATE TABLE
FULL_uqbyuan3=#
```

```
FULL_uqbyuan3=# \COPY site1_uqbyuan3 from './Part1/Product_PA1.csv' delimiter ',' CSV HEADER;
COPY 199
FULL_uqbyuan3=# \COPY site1_uqbyuan3 from './Part1/Product_PA2.csv' delimiter ',' CSV HEADER;
COPY 200
FULL_uqbyuan3=# \COPY site1_uqbyuan3 from './Part1/Product_PA3.csv' delimiter ',' CSV HEADER;
COPY 105
FULL_uqbyuan3=# \COPY site2_uqbyuan3 from './Part1/Product_PA1.csv' delimiter ',' CSV HEADER;
COPY 199
FULL_uqbyuan3=# \COPY site2_uqbyuan3 from './Part1/Product_PA2.csv' delimiter ',' CSV HEADER;
COPY 200
FULL_uqbyuan3=# \COPY site2_uqbyuan3 from './Part1/Product_PA3.csv' delimiter ',' CSV HEADER;
COPY 105
FULL_uqbyuan3=# \COPY site3_uqbyuan3 from './Part1/Product_PA1.csv' delimiter ',' CSV HEADER;
COPY 199
FULL_uqbyuan3=# \COPY site3_uqbyuan3 from './Part1/Product_PA2.csv' delimiter ',' CSV HEADER;
COPY 200
FULL_uqbyuan3=# \COPY site3_uqbyuan3 from './Part1/Product_PA3.csv' delimiter ',' CSV HEADER;
COPY 105
FULL_uqbyuan3=#
```

1.2.2 Partial replication

Each fragment will be a relation located on some of the sites in the computer network (i.e., more than one site may have a copy of this fragment, but not all of them. Specifically, three sites should be created in database PA_S1234567:

- Site1_S1234567, which contains:
  - Product_PA1
  - Product_PA2
- Site2_S1234567, which contains:
  - Product_PA1
  - Product_PA3
- Site 3_S1234567, which contains:
  - Product_PA2
  - Product_PA3

```
SalesDB=# create database "PA_uqbyuan3";

CREATE DATABASE
SalesDB=#
SalesDB=#
SalesDB=# \c PA_uqbyuan3
You are now connected to database "PA_uqbyuan3" as user "uqbyuan3".
PA_uqbyuan3=#
PA_uqbyuan3=#
```

```
PA_uqbyuan3=#
PA_uqbyuan3=# create table site1_uqbyuan3 (ProductID serial Primary key, Name varchar(50) NOT NULL, Price numeric(19, 4) DEFAULT NULL);
CREATE TABLE
PA_uqbyuan3=# create table site2_uqbyuan3 (ProductID serial Primary key, Name varchar(50) NOT NULL, Price numeric(19, 4) DEFAULT NULL);
CREATE TABLE
PA_uqbyuan3=# create table site3_uqbyuan3 (ProductID serial Primary key, Name varchar(50) NOT NULL, Price numeric(19, 4) DEFAULT NULL);
CREATE TABLE
PA_uqbyuan3=#
```

```
PA_uqbyuan3=#
PA_uqbyuan3=# \COPY site1_uqbyuan3 from './Part1/Product_PA1.csv' delimiter ',' CSV HEADER;
COPY 199
PA_uqbyuan3=# \COPY site1_uqbyuan3 from './Part1/Product_PA2.csv' delimiter ',' CSV HEADER;
COPY 200
PA_uqbyuan3=# \COPY site2_uqbyuan3 from './Part1/Product_PA1.csv' delimiter ',' CSV HEADER;
COPY 199
PA_uqbyuan3=# \COPY site2_uqbyuan3 from './Part1/Product_PA3.csv' delimiter ',' CSV HEADER;
COPY 105
PA_uqbyuan3=# \COPY site3_uqbyuan3 from './Part1/Product_PA2.csv' delimiter ',' CSV HEADER;
COPY 200
PA_uqbyuan3=# \COPY site3_uqbyuan3 from './Part1/Product_PA3.csv' delimiter ',' CSV HEADER;
COPY 105
PA_uqbyuan3=#
```

1.2.3 No replication

Each fragment will be a relation located on only one site in the computer network. You should create three sites in database NO_S1234567:

- Site1_S1234567, which only contains:
  - Product_PA1
- Site2_S1234567, which only contains:
  - Product_PA2
- Site3_S1234567, which only contains:
  - Product_PA3

```
SalesDB=# create database "NO_uqbyuan3";

CREATE DATABASE
SalesDB=#
SalesDB=#
SalesDB=# \c NO_uqbyuan3
You are now connected to database "NO_uqbyuan3" as user "uqbyuan3".
NO_uqbyuan3=#
```

```
NO_uqbyuan3=# create table site1_uqbyuan3 (ProductID serial Primary key, Name varchar(50) NOT NULL, Price numeric(19, 4) DEFAULT NULL);
CREATE TABLE
NO_uqbyuan3=# create table site2_uqbyuan3 (ProductID serial Primary key, Name varchar(50) NOT NULL, Price numeric(19, 4) DEFAULT NULL);
CREATE TABLE
NO_uqbyuan3=# create table site3_uqbyuan3 (ProductID serial Primary key, Name varchar(50) NOT NULL, Price numeric(19, 4) DEFAULT NULL);
CREATE TABLE
NO_uqbyuan3=#
```

```
NO_uqbyuan3=#
NO_uqbyuan3=# \COPY site1_uqbyuan3 from './Part1/Product_PA1.csv' delimiter ',' CSV HEADER;
COPY 199
NO_uqbyuan3=# \COPY site2_uqbyuan3 from './Part1/Product_PA2.csv' delimiter ',' CSV HEADER;
COPY 200
NO_uqbyuan3=# \COPY site3_uqbyuan3 from './Part1/Product_PA3.csv' delimiter ',' CSV HEADER;
COPY 105
NO_uqbyuan3=#
```

Task: Given the update query below, write a set of SQL queries, which applies this update to the system under each replication strategy, respectively. Each of your update transaction should guarantee consistency between copies and should not perform update to sites which are not possible to have the record. **Hint**: Three sets of SQL queries (or three transactions) in total for three different replication strategies.

- **Query**: Change the **price** of a record (ProductID=233) to 360.

Part 2: Distributed Query Processing with Postgresql

In part 2, we still simulate a distributed database using this centralised Postgresql database. However, the distributed sits are now organised in a peer-to-peer architecture, which means the queries can be issued at any of the local sites and receive answer from it.

To start with, create a new user with read-only privileges on sales database

1. Go to terminal and run 'psql'
2. CREATE USER readonly_user WITH PASSWORD 'infs3200';
3. \c SalesDB
4. GRANT CONNECT ON DATABASE "SalesDB" TO readonly_user;
5. GRANT USAGE ON SCHEMA public TO readonly_user;
6. GRANT SELECT ON TABLE customers TO readonly_user;

Sales database has a read-only user.

Now, create a new database called 'tute2' and connect to it. The external database 'tute2' wishes to fetch data from the sales database.

The postgres_fdw module in Postgresql provides the foreign-data wrapper (FDW), which can be used to access data stored in external PostgreSQL servers. To prepare for remote access using FDW:

- Install the postgres_fdw extension using CREATE EXTENSION.
- Create a foreign server object, using CREATE SERVER, to represent each remote database you want to connect to. Specify connection information, except user and password, as options of the server object.
- Create a user mapping, using CREATE USER MAPPING, for each database user you want to allow to access each foreign server. Specify the remote user name and password to use as user and password options of the user mapping.
- Create a foreign table, using CREATE FOREIGN TABLE or IMPORT FOREIGN SCHEMA, for each remote table you want to access. The columns of the foreign table must match the referenced remote table.

The script for creating FDW to connect customer table of sales database is provided in "…/P2/Part2/create_fdw.sql". Note: you need to modify "YourUserName" in the "create_fdw.sql" file.

Task: For each transaction that satisfy salesID larger than 5000000 and customer ID equals to 6, show the customer first name, last name and product id. Hint: You need to create another foreign table to achieve this.