INFS3200 Sem 1, 2025

Assignment 1: Distributed Databases (10%)

Guidelines for Assignment Submission:

1. Submission Deadline: **April 4th, 2025, 15:00 (AEST).**
2. For questions requiring queries, kindly include both a screenshot of the query and a screenshot of the execution results.
3. When addressing discussion-based questions, provide comprehensive textual responses, supplemented with suitable visual aids if necessary.
4. The submission should be in PDF format, named as 'A1_s1234567.pdf' (replace '1234567' with your student ID). The submitted file should not exceed 10 MB.
5. All implementations and project components should be finalized within the UQZones environment. Evaluators might assess the assignment based on the established checkpoints within UQZones.
6. All assignment submissions must be submitted exclusively through the UQ Blackboard. Alternative methods of submission will not be acknowledged. Please be mindful that submissions via email will not be entertained under any circumstances.
7. It is imperative to adhere to the stipulated submission deadline to avoid penalties, as explained in the Educational Course Policies (ECP).
8. Ensuring the successful submission of your assignment within the designated timeline is your responsibility.

Introduction

In this assignment, you will be provided with four sql files that corresponding to four tables ('salaries', 'dept_emp', 'dept_manager', 'employees' and 'departments') in the 'EMP' database. Figure 1 shows the conceptual model of 'EMP' database, where red rectangular indicate primary keys, and blue squares indicate the foreign keys. The table 'titles' is stored in a remote server.

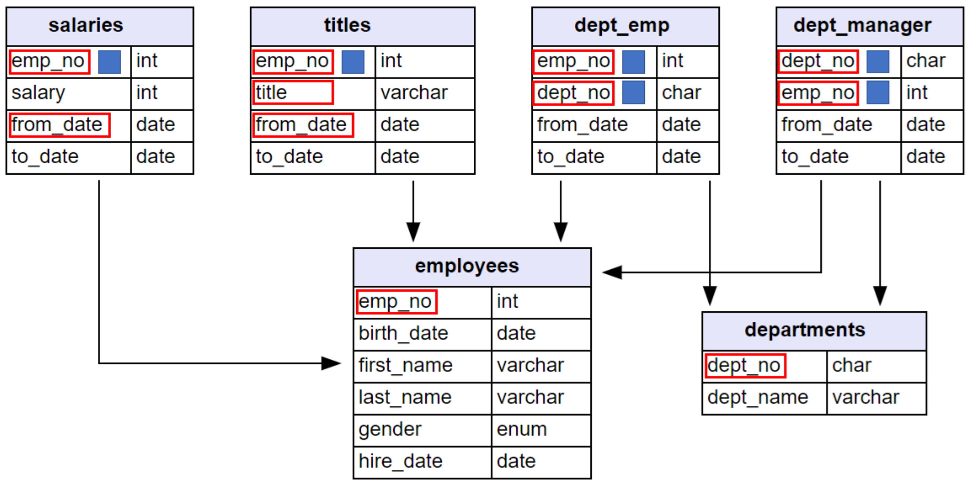

Figure1. The conceptual model of EMP database.

Task 1: Load Database (1 point)

Create a database named 'EMP_s1234567' (1234567 should be replaced by your studentID). Load the database with the provided sql files. You should provide some screenshots to list the database and tables, after creating all necessary tables and databases. For this task, only queries and screenshots for each question are required,. You don't need to provide explanations for sql queries.

- (0.5) Write sql queries to count the number of entries in table 'employees'.
- (0.5) Write sql queries to count the number of employees from the 'Marketing' department.

Task 2: Database Fragmentation (2.5 points)

- (1) Write sql queries to perform horizontal fragmentation on table 'salaries', based on the following rules:
  - 'from_date' before 1990-01-01
  - 'from_date' no earlier than 1990-01-01 and before 1992-01-01
  - 'from_date' no earlier than 1992-01-01 and before 1994-01-01
  - 'from_date' no earlier than 1994-01-01 and before 1996-01-01
  - 'from_date' no earlier than 1996-01-01 and before 1998-01-01
  - 'from_date' no earlier than 1998-01-01 and before 2000-01-01
  - 'from_date' no earlier than 2000-01-01

  Queries and screenshots for creating horizontal fragmentation are required.

- (0.5) Calculate the average employee salary in between 1996-06-30 and 1996-12-31 of the attribute "from_date" on the fragmented 'salaries' table. Show the query explanation. (Queries and screenshots are required. You also need to show the screenshot of the query plan and give descriptions about how PSQL database system executes and optimizes the queries.)
- (1) Write sql queries to perform vertical fragmentation on table 'employee', based on the following rules:
  - 'emp_no', 'first_name', 'last_name', and 'hire_date' should be stored in table 'employees_public'
  - 'emp_no', 'birth_date', and 'gender' should be stored in table 'employees_ confidential'
  - The 'employees_ confidential' table should be stored in a new database called 'EMP_Confidential'.
    - *Hint: you can export data to a sql file.*

Task 3: Database Replication (2.5 points)

Consider the table 'employee' is sliced into 10 fragmentations based on the value of 'emp_no'. This task does not involve specific operations, queries are not required. Instead, you can offer clear and thorough explanations. If you feel simple examples can better assist explanation, you may give some simple examples to support explanation.

- (1.5) Consider you have 5 different server sites. How to design full replication/ partial replication/ no replication? What could be the pros and cons for each fragmentation strategy?
- (1) Consider your master server has the fragmentation allocation schema. Based on your design, what is the process to update a record with a specific 'emp_no'?

Task 4: Access foreign data with FDW (4 points)

The table 'titles' is stored in a remote database that requires you to use Foreign Data Wrappers (FDW) to access the data. You are provided with the following login details:

- User Name: sharedb
- Password: Y3Y7FdqDSM9.3d47XUWg
- Host: infs3200-sharedb.zones.eait.uq.edu.au
- Port: 5432
- Database Name: sharedb

- (1) Write sql queries to establish a FDW to the external DB. Queries/commands should be provided.
- (1) For each unique 'title' in table 'titles', calculate the averaged current salary. You should query from the foreign table and provide the queries and screenshots of the outputs.
    - *Hint: current staff has 'to_date' that equals to '9999-01-01'*
- (1) Consider that the current employee table is vertically fragmented and stored on different databases. Perform semi join from 'EMP' database to select the last name and first name of employees that 'birth_date' is no earlier than 1970-01-01 and before 1975-01-01. You should create another foreign table mapping to the necessary table and access the vertical fragmentation table through FDW. Queries, screenshots and brief descriptions are required.
    - *Note: 'birth_date' information can only be accessed from 'EMP_Confidential' database through FDW. Direct read from the original employee table will receive 0 mark for this question.*

- (1) Compared with inner join, will semi join incur higher transmission cost under this scenario? You should respectively show the steps for semi-join and inner-join with queries and **transmission cost** (not the join cost). Queries, screenshots of the query plans, and explanations about how the joins are performed and why one join strategy has more transmission cost than the other should be included in the submission.
    - *Note: Screenshots are required for the comparison.*

Task 4.3 and 4.4 Appendix

Databases to be used: In these questions, we assume the database that contains "employees_public" is the local site, and the database "EMP_Confidential" that is created in task 2 and contains "employees_confidential" is the remote site. In this case, you don't have to use "sharedb", so you don't need to modify sharedb or access it. If you want to access employees_confidential table from the employee_pulic database, how can you access it? (Hint: use FDW)

Semi-join analysis: we are simulating distributed databases within a centrialize database system using psql, and we don't need to execute semi-join. Instead, we aim to simulate its behavior. Below is a simple example outlining the steps involved in semi-join emulation, and let's consider the tutorial question:

"Assume R is at site 1 and S is at site 2, and a query R $\bowtie$ S is issued at site 2. List the steps for a query processing strategy using semi-join, and check if the semi-join is a beneficial option in this case (ignore local processing cost).

Relation R has schema R(A, B), and relation S has schema S(B,C,D), where attribute B is the foreign key."

In this case the local site is the site where the query is issued, which is site 2, and the remote site is site 1.

Step 1: The first step is to transmit foreign keys/primary keys from local site to remote site to prepare semi join:

```
SELECT B FROM S;
```

Step 2: Once the foreign keys are transmitted to the remote site 1, semi-join is performed at remote site:

```
SELECT R.A, R.B
FROM R, (SELECT B FROM S) FS
WHERE R.B = FS.B;
```

Step 3: The semi-join result from the previous step will be sent back to the local site, and perform final join:

```
SELECT FR.A, FR.B, SL.C, SL.D -- Should be the final attributes returned to user
FROM S SL, (SELECT R.A, R.B FROM R, (SELECT B FROM S) FS WHERE R.B = FS.B) FR
WHERE SL.B = FR.B;
```

You should perform similar steps in task 4.3, but note that you may add some filtering conditions to optimize the query and reduce transmission cost at remote and local sites.

The procedure of inner-join in task 4.4 is directly sending all necessary attributes from remote site to the local site. However, you should be aware of which step(s) have transmission cost for semi-join and inner-join?