Serialisability

INSTRUCTIONS

1. Download the ZIP file "`project4.zip`" from Luminus
   "`Files > Projects > Project 4: Serialisability`".
2. Submit one ZIP file, `<student number>.zip` (for example: "`A012345L.zip`"), containing the following files to the folder "Project 4: Submission" in Luminus
   "`Files > Projects > Project 4: Serialisability`" by **Friday 3 April 2020 at 18:30**.
   - A text file named `synthesis.txt`.
   - Six python source code files named `db_connect.py`, `create_account.py`, `run_sums.py`, `run_exchanges.py`, `run_all_exchanges.py`, and `run_experiments.py`.
   - Four csv files named `S-correctness.csv`, `S-time.csv`, `P-correctness.csv` and `P-time.csv`.
3. Past this deadline and before **Friday 10 April 2020, 18:30**, you may submit to the "`Project 4: Late Submission`" folder (penalties apply).
4. Do not modify any of the template files unless otherwise indicated.

The goal of this project is to run an experiment that illustrates the value of serializability in PostgreSQL. The database contains a single `account` table, with a `balance` as one of its columns. Two types of transactions are executed concurrently: `Sum`, which calculates the sum of all account balances, and `Exchange`, which exchanges the balances of two arbitrary accounts. The experiment consists in running those transactions in parallel at different isolation levels and assess both correctness and performance. You should read and understand how the transaction isolation levels work in PostgreSQL [1].

Use Python 3 and SQLAlchemy library[2] to answer the questions.

Run the command `pip install psycopg2-binary sqlalchemy` to install the SQLAlchemy library.

[1] `https://www.postgresql.org/docs/10/sql-set-transaction.html`
[2] `https://docs.sqlalchemy.org/en/latest/core/tutorial.html`

The ZIP file "`project4.zip`" contains the following files:

- Six python source code files named `db_connect.py`, `create_account.py`, `run_sums.py`, `run_all_exchanges.py`, and `run_experiments.py`.
- A text file named `synthesis.txt`.
- Four csv files named `S-correctness.csv`, `S-time.csv`, `P-correctness.csv` and `P-time.csv`.

Run the PostgreSQL database and create a database called `cs4221_p4`.

The Python file `db_connect.py` connects to the database. It has a `get_conn` function that returns a connection to the PostgreSQL database server. Open the file `db_connect.py` and make sure that the database connection information (e.g. username, password) is correct.

The Python file `create_account.py` creates an `account` table that stores an account number (integer) as primary key, a branch number (integer) and a balance (float) by using the connection from the `db_connect.py` file. It populates the `account` table with 100,000 records where the account number is numbered from 1 to 100,000, the branch number is randomized in the [1, 20] interval and the balance is randomized uniformly in the [0, 100,000) interval rounded to 2 decimal places. Run the file `create_account.py` file to populate the database.

Question 1 [2 marks]

The `Sum` transaction calculates the sum of all `balances` over the `account` table.

The template file is a Python file that takes S and I as inputs, where S is the number of sums and I is the isolation level (`'READ UNCOMMITTED'`, `'READ COMMITTED'`, `'REPEATABLE READ'`, `'SERIALIZABLE'`). The function `sum_balance` takes a session as input and execute a query that calculates the sum of all `balances` over the `account` table. You should complete this function in Python using SQLAlchemy library functions for raw SQL queries. Please refer to the SQLAlchemy documentation.

The function `S_sums` runs the `sum_balance` function S times in sequence with isolation level I and returns all sum values upon completion. This function has been written for you.

The completed Python file `run_sums.py` should print all the sums.

Submit the completed Python file `run_sums.py`.

Question 2 [2 marks]

The `Exchange` transaction swaps the balance of two accounts. More specifically, it proceeds in five steps:

1. it reads the balance from a first account A_1 (picked at random) into a variable V_1,
2. it reads the balance from a second account A_2 (picked at random) into a variable V_2,
3. it writes the value V_1 as the new balance of the account A_2,
4. it writes the value V_2 as the new balance of the account A_1.
5. commit the changes.

Using `run_sums.py` as a template, write a similar Python file `run_exchanges.py` that takes E and I as inputs, where E is the number of exchanges in a subprocess and I is the isolation level (`'READ UNCOMMITTED'`, `'READ COMMITTED'`, `'REPEATABLE READ'`, `'SERIALIZABLE'`), measures and outputs the overall execution time.

Create a function `exchange` that takes a session as an input and execute the `Exchange` transaction in five steps defined previously. The generation of the two random account numbers is done in Python. This function is similar to the `sum_balance` function in `run_sums.py`.

Create a function `E_exchanges` that runs the `exchange` function E times in sequence, with isolation level I and a 0.1 ms pause in-between transactions (use the `time.sleep` function). This function is similar to the `S_sums` function in `run_sums.py`.

Submit the Python file `run_exchanges.py`.

Question 3 [6 marks]

The Python file `run_all_exchanges.py` takes E, P and I as inputs, where E is the number of exchanges in a subprocess, P is the number of subprocesses and I is the isolation level (`'READ UNCOMMITTED'`, `'READ COMMITTED'`, `'REPEATABLE READ'`, `'SERIALIZABLE'`). The code runs P `run_exchanges.py` subprocesses concurrently. The code measures and returns the average execution time of all subprocesses. This code in this file has been written for you.

The Python file `run_all_experiments.py` takes S, E, P and I as inputs, where S is the number of sums, E is the number of exchanges in a subprocess, P is the number of subprocesses and I is the isolation level (`'READ UNCOMMITTED'`, `'READ COMMITTED'`, `'REPEATABLE READ'`, `'SERIALIZABLE'`). It calculates the sum of all `balances` over the `account` table once. This value is called the *true sum*. It runs the `run_sums.py` and `run_all_exchanges.py` codes concurrently with their respective parameters. It calculates `correctness` which is the number of correct sums (from the output of `run_sums.py` with respect to the *true sum*) divided by S (number of sums). This code in this file has been written for you.

Feel free to or not to modify the Python file `run_all_experiments.py` to script the processing of the answers to the questions below.

(a) Set the number of exchanges $E = 200$ and the number of subprocesses $P = 100$. Vary the number of sums, S, from 100 to 1000 in increments of 100 (i.e. $S = \{100, 200, \ldots, 1000\}$). For each value of S, repeat the experiment 20 times. For each isolation level I and each value of S, report the mean and the standard deviation (over the 20 experiments) of the correctness in the template file `S-correctness.csv` and of the average execution time (over P subprocesses) in the template file `S-time.csv`.

(b) Set the variables for the number of sums, S, equals to the S with the lowest average accuracy over all isolation levels from the previous experiment (if there are more than one, choose the smallest S). The total number of exchanges must remain constant over all experiments, $E \times P = 2000$. Every time you change the number of subprocesses, P, you must adjust the number of exchanges, E, to the nearest integer. Vary P from 1 to 100 in increments of 10 (i.e. $P = \{1, 10, 20, \ldots, 100\}$). For each value of P, repeat the experiment 20 times. For each isolation level I and each value of P, report the mean and the standard deviation (over the 20 experiments) of the correctness in the template file `P-correctness.csv` and of the average execution time (over P subprocesses) in the template file `P-time.csv`.

Synthesise concisely the observations that you can make from looking at your results in `S-correctness.csv`, `S-time.csv`, `P-correctness.csv` and `P-time.csv`, respectively. Namely, describe what you observe, formulate hypotheses that explain your observations and check them with the documentation. Report your synthesis for the four tables in the space indicated, respectively, in the file `synthesis.txt`.

– END OF PAPER –