Project Description

The course project on *Transactions, Concurrency and Recovery* is a venue for students to practice building a parallel database system that supports concurrent multi-user access. Students will again use the CBMS dataset from the **combined** provinces of Palawan and Marinduque to design a three-node parallel database system where they perform concurrent transactions, and simulate crash and recovery techniques. The final output will be a software that will be demonstrated during the indicated schedule to the respective ADVANDB teachers, and a test script showing the results of performing the different test cases described below on your software.

> "A distributed database management system (DDBMS) ensures that changes, additions, and deletions performed on the data at any given location are automatically reflected in the data stored at all the other locations. Therefore, every user always sees data that is consistent with the data seen by all the other users."

Methodology

Students are to form teams with **3 - 4 members, subject to certain grouping constraints as discussed by your teacher**.

To proceed with this project, each team should conduct the following:

Step 1. Build a Parallel Database System

Create three (3) nodes – with three (3) separate computers that are connected and can communicate with each other. Each node should have its own database with the following contents:

- Node 1: Central Office – repository of ALL data from all regions (in this case, the complete dataset of Palawan and Marinduque).
- Node 2: Palawan Branch – repository of the dataset from the province of Palawan.
- Node 3: Marinduque Branch – respository of the dataset from the province of Marinduque.

Step 2. Local Concurrency Control and Recovery

2.1 Create an application to simulate **local** concurrency control for each node. The application should be able to show two or more local transactions concurrently executing with each other and they all leave the database in a consistent state. Three cases to simulate:

a. Case #1: All transactions are reading.

b. Case #2: At least one transaction is writing (update / deletion) and the others are reading.

c. Case #3: All transactions are writing (update / deletion).

Notes:

- For each case, set the isolation level to *read uncommitted*, *read committed*, *read repeatable* and *serializable*.
- Updates in either Node 2 or Node 3 should be replicated in the central node. Any updates in the central node should be replicated in either Node 2 or Node 3.
- Reusing query statements from MCO1 and MCO2 is highly encouraged.

2.2 Extend the application to simulate crash and recovery using multiple concurrent transactions. Show how the system recovers from failures. Two cases to simulate:

a. Case #1: At least one transaction fails due to software failure.
b. Case #2: The entire node fails due to other types of failure, e.g., loss of power, network disruption, hardware failure.

Step 3. Global Concurrency Control and Recovery

3.1 Extend the previous application to show three or more **global** transactions, at least one in each node, concurrently executing with each other and they all leave the database in a consistent state. Three cases to simulate:

a. Case #1: All transactions are reading.
b. Case #2: At least one transaction is writing (update / deletion) and the others are reading.
c. Case #3: All transactions are writing (update / deletion).

Notes:

- For each case, set the isolation level to *read uncommitted*, *read committed*, *read repeatable* and *serializable*.
- Updates in either Node 2 or Node 3 should be replicated in the central node. Any updates in the central node should be replicated in either Node 2 or Node 3.

3.2 Extend the application to simulate **global** crash and recovery. Show how the system recovers from failures when global transactions are executing. Two cases to simulate:

a. Case #1: Either Node 2 or Node 3 fails.
b. Case #2: The central node fails.

Step 4. Evaluation

Write a test script to cover all the test cases described in Steps 2 and 3. Execute each of your test case at least 3 times and log the results. Remember that testing should be efficient and effective.

Final Deliverables

The following final deliverables are required to be submitted:

1. Source Code
2. Test Script with results (follow SPSWENG / SOFTENG / INTROSE format)

- Softcopy of the deliverables must be submitted at **12:01am of April 8** (for all classes). Follow the file naming convention: **ADVANDB_<section>_<lastnames of members>.<ext>**
- Printed copies of the deliverables (if required) must be submitted during the date of the presentation.
- Late submissions will receive 10 points deduction per day (including weekends and holidays) and 0 points for the presentation. No late submissions will be accepted after April 11, 2016.
- The actual schedule for the presentation of software and test script will be provided by your respective teacher.
- **Plagiarized works will automatically be given a grade of 0.0 for the course.**