Project Description

The course project on *Transactions, Concurrency and Recovery* is a venue for students to practice building a distributed database system that supports concurrent multi-user access. Students will again use the World Development Indicators (WDI) dataset to design a three-node distributed 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 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

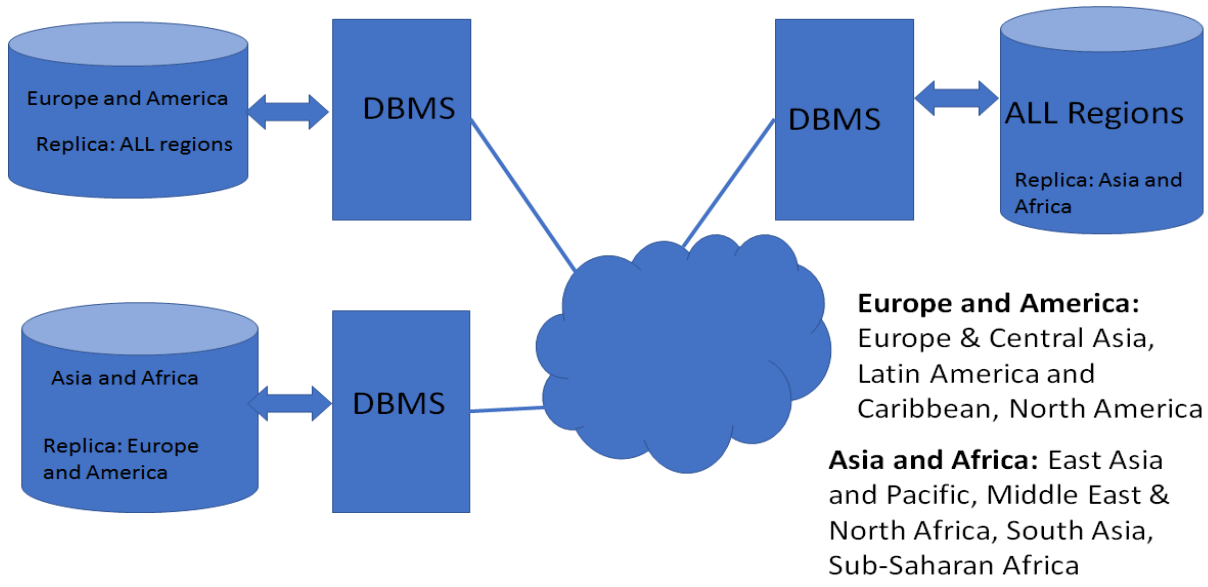
Students are to form teams with **2-3 members**

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

Step 1. Build a Distributed Database System

NOTE: Reusing query statements from MCO2 is highly encouraged.

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 (see Figure 1):



- Node 1: Main site: repository of ALL regions (schema and data) ; it also contain a replica (copy) of Asia and Africa data (schema and data)
- Node 2: repository of (data and schema) Europe and America (i.e. Europe & Central Asia, Latin America and Caribbean, North America); it also contains a replica (copy) of All regions (data and schema)

- Node 3: repository of (data and schema) Asia and Africa (i. e. East Asia and Pacific, Middle East & North Africa, South Asia, Sub-Saharan Africa); it also contains a replica copy of Europe and America

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

Notes:

- Updates in Node 2 should be replicated in Node 1 and its replica
- Updates in Node 3 should be replicated in Node 1 and its replica
- Updates in Node 1 should be replicated in Node 2, Node 3 and its replica

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: query that involves reading and summarizing data in at least 2 nodes (SELECT)
b. Case #2: At least one transaction is writing (insert/ update / deletion) and the others are reading.
c. Case #3: All transactions are writing (insert/update / deletion).

Notes:

- Updates in Node 2 should be replicated in Node 1 and its replica
- Updates in Node 3 should be replicated in Node 1 and its replica
- Updates in Node 1 should be replicated in Node 2, Node 3 and its replica

3.2 Extend the application to simulate **global** crash and recovery. Show how the system recovers from failures when global transactions are executing. Also show that the because of replication, the system has higher availability (not easily vulnerable to failures) Two cases to simulate:

a. Case #1: Either Node 2 or Node 3 fails (or both).
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.

Grading Scheme:

- **System (85%) : Correctness, Complexity, Completeness, others**
- **Test scripts (15%): readability, organization, completeness**

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 (uploaded in the group dropbox folders) at **23:59 on December 10** Create a separate folder labeled **"MCO3"** in the dropbox group folder and upload all deliverables (technical report (pdf/MSWord), source code of the system, SQL queries, inserts, deletes, updates (.txt), ppt presentation slides) into the folder. Whenever applicable, label all your files using the following convention:

  MCO3Fname1-FName2-FName3.ext

- Printed copies of the deliverables (Test script) must be submitted during demo **Dec. 11**
- Late submissions will receive 10 points deduction per day and 0 points for the presentation. No late submissions will be accepted after Dec. 10
- Prepare a Power point presentation to demonstrate your Distributed DB application, test methodology and results. The actual schedule for class presentation will on **Dec. 11 (tentative).**

- **Plagiarized works will automatically be given a grade of 0.0 for the course.**