|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **Hope Foundation’s,**  **Finolex Academy of Management and Technology, Ratnagiri** | | | | | | | | | |
| **Department of Information Technology** | | | | | | | | | |
| Subject name: OLAP LAB | | | | | | | | Subject Code: ITL503 | | | |
| Class | | TE IT | | Semester – V (CBCGS) | | | | Academic year: 2018-19 | | | |
| Name of Student | |  | | | | | **QUIZ Score :** | | | | |
| Roll No | |  | | | Assignment/Experiment No. | | | | | 01 | |
| Title**: Implementation of Two Phase locking protocol.** | | | | | | | | | | | |
|  | | | | | | | | | | | |
| 1. **Course objectives applicable:**   **LOB1**- Understand the stages and properties of transactions and the concepts involved with Serializability. | | | | | | | | | | | |
| 1. **Course outcomes applicable:**   **LO1**- Apply the knowledge of Concurrency control management to simulate Two Phase Locking protocol. | | | | | | | | | | | |
| **3. Learning Objectives:**   * To be able to apply the knowledge of Concurrency control management. * To be able to understand the working of Two Phase Commit protocol. | | | | | | | | | | | |
| 1. **Practical applications of the assignment/experiment:**  * Wherever the distributed database management system is utilized, the 2 PC or 3 PC protocols are implemented to achieve atomicity. E.g. Banking system. | | | | | | | | | | | |
| **5. Prerequisites**: Working of Client Server communication, Transaction properties. | | | | | | | | | | | |
| **6. Hardware Requirements**:   1. PC with 4GB RAM, 500GB HDD,   **7. Software Requirements:**  1. Programming language , Java | | | | | | | | | | | |
|  | | | | | | | | | | | |
| **8. Quiz Questions (if any): (Online Exam will be taken separately batch wise, attach the certificate/ Marks obtained)**   1. What is transaction? 2. List all the properties of transaction. 3. What is atomicity? | | | | | | | | | | | |
|  | | | | | | | | | | | |
| **9. Experiment/Assignment Evaluation:** | | | | | | | | | | | |
| **Sr. No.** | **Parameters** | | | | | | | | **Marks obtained** | | **Out of** |
| **1** | Technical Understanding (Assessment may be done based on Q & A **or** any other relevant method.) Teacher should mention the other method used - | | | | | | | |  | | 6 |
| **2** | Neatness/presentation | | | | | | | |  | | 2 |
| **3** | Punctuality | | | | | | | |  | | 2 |
| **Date of performance (DOP)** | | |  | | | **Total marks obtained** | | |  | | **10** |
| **Date of checking (DOC)** | | |  | | | **Signature of teacher** | | | | | |

**Theory: <HANDWRITTEN>**

During normal execution, each site maintains a log, and the actions of a sub transaction are logged at the site where it executes. A commit protocol is followed to ensure that all sub transactions of a given transaction either commit or abort uniformly. The transaction manager at the site where the transaction originated is called the coordinatorfor the transaction; transaction managers at sites where its sub transactions execute are called subordinates**.** When the user decides to commit a transaction, the commit command is sent to the coordinator for the transaction. This initiates the 2PC protocol:

1. The coordinator sends a *prepare* message to each subordinate.

2. When a subordinate receives a *prepare* message, it decides whether to abort or commit its sub transaction. It force-writes an abort or preparelog record, and *then* sends a *no* or *yes* message to the coordinator.

3. If the coordinator receives *yes* messages from all subordinates, it force-writes a commit log record and then sends a *commit* message to all subordinates. If it receives even one *no* message, or does not receive any response from some subordinate for a specified time-out interval, it force-writes an abort log record, and then sends an *abort* message to all subordinates.

4. When a subordinate receives an *abort* message, it force-writes an abort log record, sends an *ack* message to the coordinator, and aborts the sub transaction. When a subordinate receives a *commit* message, it force-writes a commit log record, sends an *ack* message to the coordinator, and commits the sub transaction.

5. After the coordinator has received *ack* messages from all subordinates, it writes an end log record for the transaction.

The name *Two-Phase Commit* reflects the fact that two rounds of messages are exchanged: First a voting phase, then a termination phase, both initiated by the coordinator. The basic principle is that any of the transaction managers involved (including the coordinator) can unilaterally abort a transaction, whereas there must be unanimity to commit a transaction.

**Algorithm:-**

**At coordinator**

1. start
2. Create ServerSocket object and accept all subordinates request and connect with them.
3. Send a *prepare* message to each subordinate.
4. Receives *yes or no* messages from all subordinates.
5. If *yes* messages from all subordinates are received, write a commit log record and then send a *commit* message to all subordinates.
6. If *no* messages from all subordinates or even one are received, write an abort log record and then send an *abort* message to all subordinates.
7. Received *ack* messages from all subordinates, write an end log record for the transaction.
8. Stop.

**At subordinate**

1. start
2. Create Socket object and connect to coordinate.
3. Receives *prepare* messages from coordinate.
4. Send a *yes or no* message to coordinate.
5. Receives commit or abortmessage from coordinate.
6. Send a ack to coordinate.
7. Stop.

**Observations: <HANDWRITTEN>**

* Revised the socket programming concept of java.
* The result cannot be easily reconstructed.
* To understand the protocol is not very complex.

**Results:**

**Learning Outcomes Achieved <HANDWRITTEN>**

1. This program has achieved the objective of
2. The programs was coded in
3. It was proved that

**Conclusion: <HANDWRITTEN>**

**Practical implementation of ‘atomicity’ is implemented and understood.**

**References** :

1. https://www.javatpoint.com/mysql-tutorial

2. Advance Database Management system by Rini Chakrabarti.