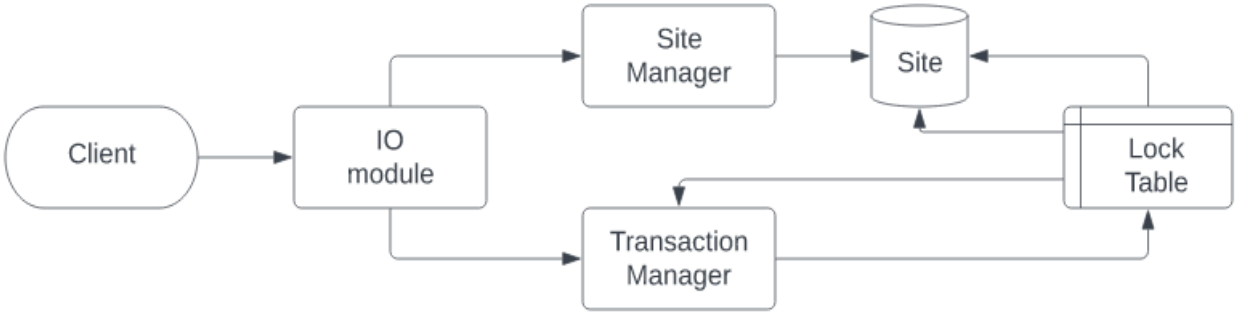
Advanced Database Systems, CSCI-GA.2434-001
New York University, Fall 2022

Distributed replicated concurrency control and recovery
Design Document

Shiv Sinha [srs9969@nyu.edu]
Abhishek Verma [av2783@nyu.edu]

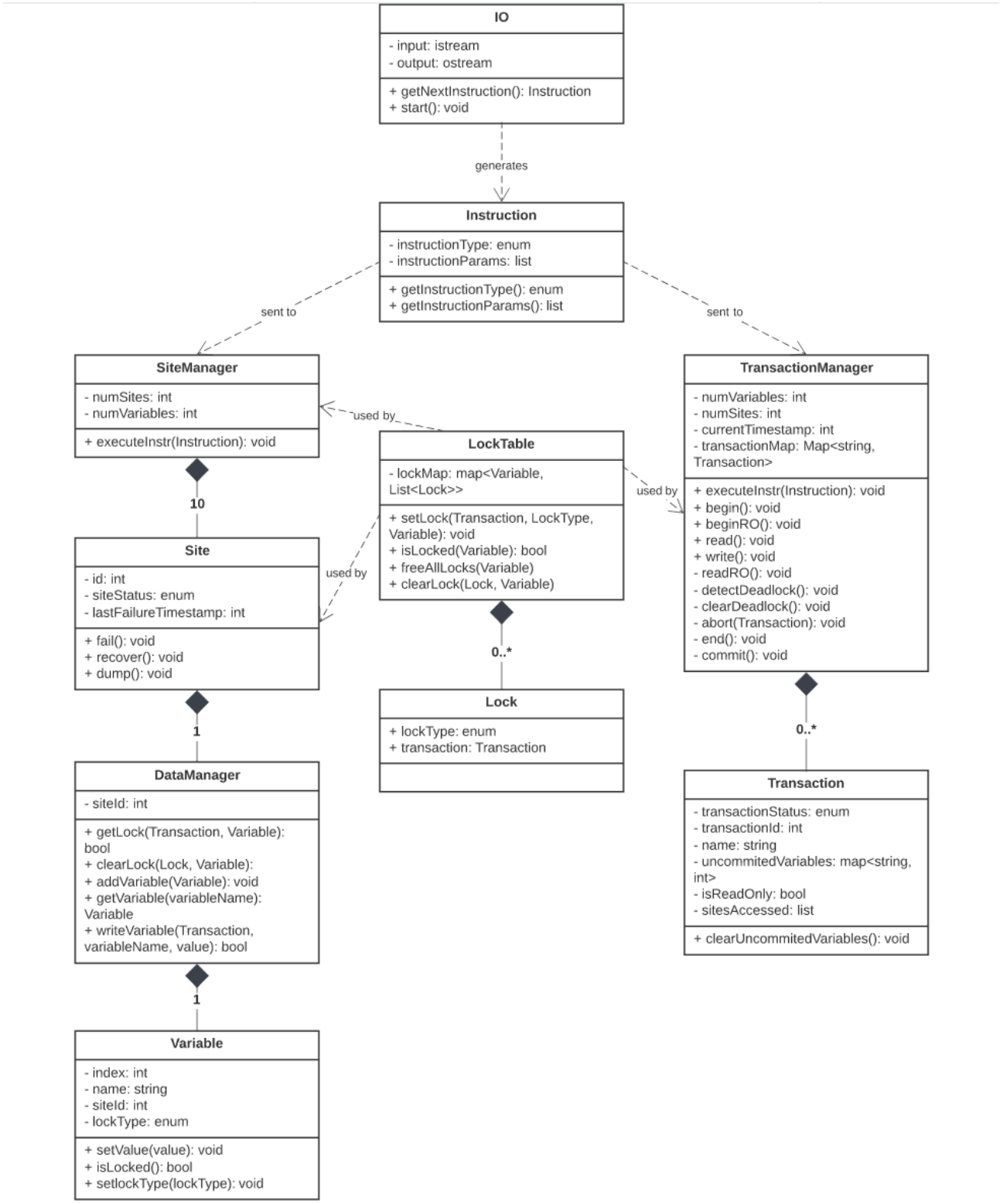
High-level Design

The following diagram presents a high-level design of different components of the distributed database. All these components are discussed in detail later.



Low-level design and UML diagram:

Here, we present a class diagram of different entities and their interaction. We describe the role of each of these classes along with additional information in the next section. The diagram follows UML format.



Components

Classes:

1. Main: The main class of the project is responsible for starting the site manager, transaction manager, and IO.
2. IO: IO operations, i.e., read instructions from stdin and forward them to SiteManager and TransactionManager.
3. SiteManager: The site manager manages sites and their interactions, such as failure and recovery, within the whole system. For any queries related to the site, including access to the lock table through the data manager, it is the central contact for the transaction manager. SiteManager provides all kinds of access to different sites and acts as proxy for them.
4. Site: As described in the project requirements, the site component illustrates a specific site. Users will have to navigate through SiteManager and then Site in order to access any variables that are present on this site. It contains a data manager that contains lock tables and variables.
5. DataManager: This program is local to each site and manages all variables and locks for the site
6. Lock: A lock placed by a transaction on a variable
7. LockTable: The LockTable holds information about locks present on variables. Its main attribute is lock_map which contains variables, keys, and values. It assists in setting, testing, and clearing locks.
8. TransactionManager: The TransactionManager class is responsible for handling transactions and deadlock detection and resolution. When a transaction commits, it calls the siteManager to clear locks.
9. Transaction: Transaction represents a running process that was started from an input file. A transaction can read and write to variables, as well as have a status assigned to it.
10. Variable: The variables class represents our sites' data, which is accessible by transactions.
11. Util: All the static utility functions are a part of this class.

Enums:

1. InstructionType: Type of instruction: READ, READ_ONLY, WRITE
2. LockStatus: Output returned when one tries to acquire a lock through siteManager: ALL_SITES_DOWN, NO_LOCK, GOT_LOCK, GOT_LOCK_RECOVERING
3. LockType: Types of locks: READ, WRITE
4. SiteStatus: Site status: UP, DOWN, RECOVERING
5. TransactionStatus: Status of a particular transaction: RUNNING, WAITING, BLOCKED, ABORTED, COMMITTED

Miscellaneous:

1. start.cpp: This file has main() function that starts the simulation (yes we are in a simulation :P)
2. All the function descriptions and comments can be found in the header files in the *include* directory.
3. Sample test inputs can be found in the *test* directory.