**Persistent Data Management**

Raw data storage will be provided by Akamai. In the system, data summaries will be kept. Since there are many clients, the system should support concurrent accesses for data. There will be some queries on the data attributes which might be complex depending on the clients. Also there can be a large set of data summaries. So, storing the data in a relational database will be beneficial. In this case, Block is a persistent object. Also the last modification times and versions of blockchains should be kept which makes BlockChain persistent. On the other hand, some clients will have accounts which are needed to reach authenticated data. Since there may be so many clients using that data, the account information will also be kept in a relational database.

**Access Control and Security**

Normally, the system does not require authentication, and since the architecture is de-centralized, there is not an access control on data. However in the case of authenticated data, system requires the clients to log in by using their usernames and passwords. There will also be encryption on authenticated data for avoiding third party access. Since there is only one actor in the system, the authentication of it is controlled by a capability list. They can query data from the chains and add new blocks as they want. They will also have access to functions of NetworkManager. For receiveData() and sendQuery() functions, authenticated data should be checked by the system.

|  |  |  |
| --- | --- | --- |
| Actor/Object | BlockChain | NetworkManager |
| Client | query(sqlPredicate)  addBlock(block) | connect(networkAddr)  sendData()  sendQuery(sqlPredicate)  receiveData() |