Comparative study between ACID and BASE database model

Sumedha Banthia, Anand Iyer and Aditya Mukherjee

***Abstract*— In this paper we are analysing the differences between the ACID and the BASE properties in the field of Database management system and using those differences comping up with a conclusion about which model should be used in what field.**

We will be taking an example of MYSQL and Apache Cassandra, which will help the user understand the comparison better and is a much more comprehensive manner

I. INTRODUCTION

Deciding on the right database management system (DBMS) can be a difficult task. The number of available options is huge. In this paper we will be comparing the two most popular database transaction models,i.e ACID and BASE.

Here our paper will be divided into three section. In the first section we will be describing the ACID transaction model using the example of MYSQL. In the second part of our discussion we will be using Apache Cassandra to describe our BASE transaction model and then in the last part we will be comparing them both and come up with a conclusion.

Our Aim in this paper ,is to make the reader clear about the two main transaction models, Their uses and their advantages and then come up with clear conclusion on which model should be used where and why.

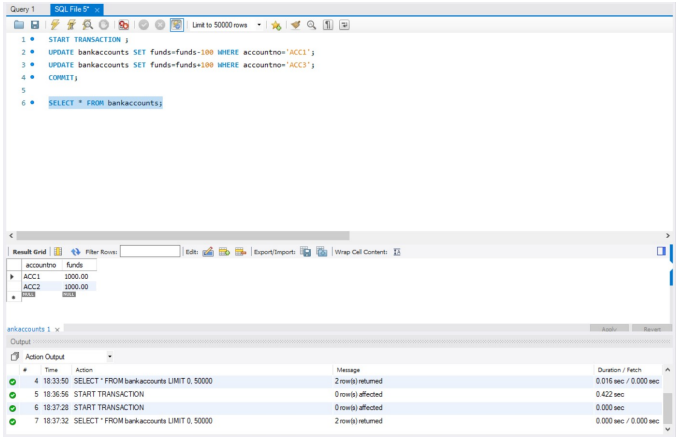
II. ACID TRANSACTION MODEL

The ACID Database bases itself around consistency for transactions. It keeps all the database in a an ordered manner and ensures all transactions are performed in a smooth manner . This makes it a good fit for online transaction business or online analytical business. This Transaction model is mainly found in the SQL databases.

A. Atomicity

The Atomicity Property of a Transaction in SQL Server ensures that either all the DML Statements (i.e. insert, update, delete) inside a transaction are completed successfully or all of them are rolled back.

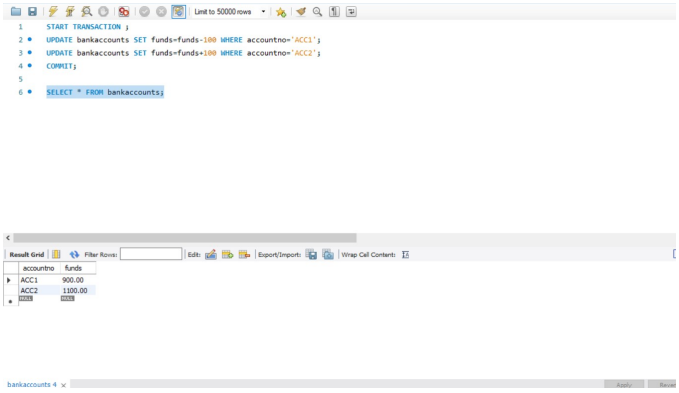
Lets take our example here, we have two bank accounts here in the transaction and one of them is correct and other one is not , and hence none of the changes take place and the table remains as the old one



B. Consistency

The Consistency Property of a Transaction in SQL Server ensures that the database data is in a consistent state before the transaction started and also left the data in a consistent state after the transaction is completed. If the transaction violates the rules then it should be rolled back.

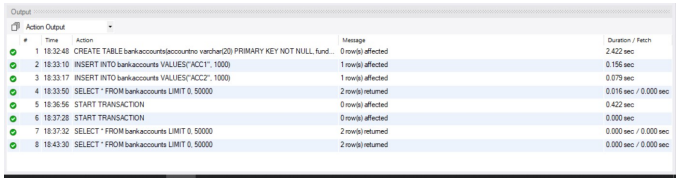
Let's take our example here, We have two bank accounts here and after the transaction we have a consistent table where the money transaction has taken place successfully



C. Isolation

The Isolation Property of a Transaction in SQL Server ensures that the intermediate state of a transaction is invisible to other transactions. The Data modifications made by one transaction must be isolated from the data modifications made by all other transactions. Most databases use locking to maintain transaction isolation. Here that means we can't do

two DML commands at the same time to the same database, each of the commands are isolated against each other.Here you can see that the commands take place on after another and not parallel due to the isolation level in MYSQL



D. Durability

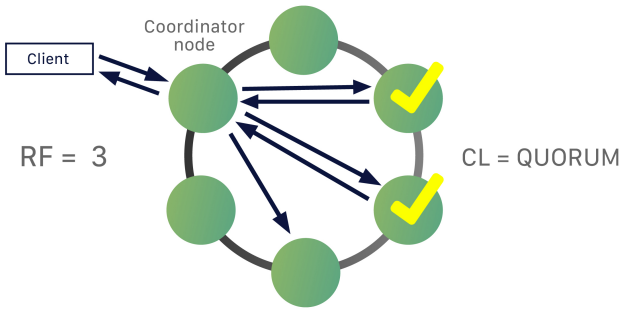
The Durability Property of a Transaction in SQL Server ensures that once the transaction is successfully completed, then the changes it made to the database will be permanent. Even if there is a system failure or power failure or any abnormal changes, it should safeguard the committed data.

III. BASE Transaction Model

A. Basically Available

Basically available means that the data is not persistent like the Durability aspect of the ACID model , but the model will be available for most of the time .

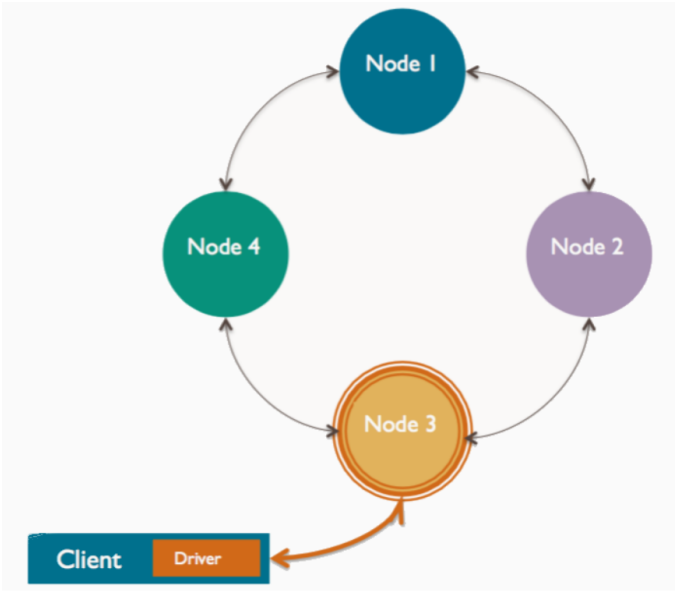
In Cassandra database is stored as replicas in different nodes and clusters. These nodes are different from each other and thus when one nodes dose not work the other node takes its place and thus the user doesn't face difficulties there. But there data are replicas and not in the same form and thus consistency of data will be different



B. Soft State

Soft State means that when a user doesn't maintains data , that will go away. Stated another way, the information will expire unless it is refreshed.

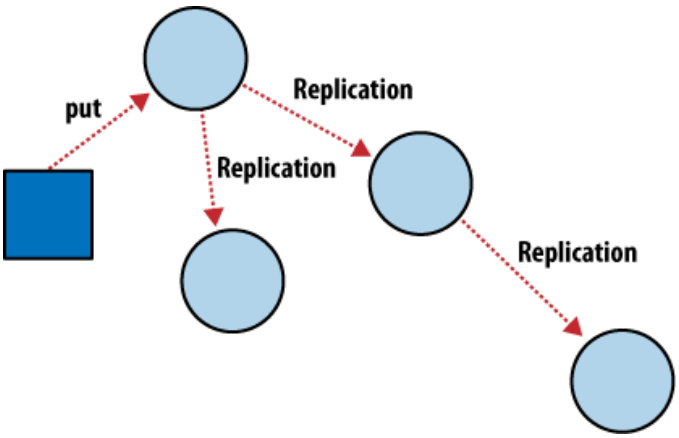
In Cassandra the soft state can be observed in two ways, one is where it mentioned Time To Live to some values and if it is not accessed than the value is discarded. And the Second one is the gossip protocol within Cassandra; a new node can determine the state of the cluster from the gossip messages it receives, and this cluster state must be constantly refreshed to detect unresponsive nodes.



C. Eventual consistency

The fact that BASE does not enforce immediate consistency does not mean that it never achieves it. However, until it does, data reads are still possible (even though they might not reflect the reality)

In Apache Cassandra , the data is stored in different replicas.These replicas eventually becomes consistent to other replicas as the time goes on. Divergent versions of the same data may exist temporarily but they are eventually reconciled to a consistent state. Eventual consistency is a trade off to achieve high availability and it involves some read and write latencies



Serial No	Criteria	ACID	BASE
01	Simplicity	Simple	Complex
02	Focus	Commits	Best Attempt
03	Maintenance	High	Low
04	Consistency Of Data	Strong	Weak/Loose
05	Concurrency Scheme	Nested Transactions	Close To Answer
06	Scaling	Vertical	Horizontal
07	Implementation	Easy To Implement	Difficult To Implement
08	Upgrade	Harder To Upgrade	Easy To Upgrade
09	Type of Database	Robust	Simple
10	Type of Code	Simple	Harder
11	Time Required For Completion	Less Time	More Time
12	Examples	Oracle, MySQL, SQL Server, etc.	DynamoDB, Cassandra, CouchDB, SimpleDB etc.

IV. Conclusion

The ACID model is geared towards consistency of transactions while BASE has a concentrated focus on high availability. This makes ACID better suited to deal with use cases such as analytical or transaction processing while BASE model is better suited towards systems that require more flexibility.FOr the user to use BASE he should be very careful about his data consistency and also be careful about his schema before hand , and in ACID the automatic and smoothness in transaction takes care of all the aforementioned conditions In essence, there is no apparent better model when the two are compared, it is more a question of the developer's use case and requirements.

References

[1] https://phoenixnap.com/kb/acid-vs-base
[2] https://cassandra.apache.org/doc/latest/
[3] https://www.geeksforgeeks.org/acid-model-vs-base-model-for-database/G
[4] https://medium.com/geekculture/acid-vs-base-in-databases-1bcad774da26
[5] https://www.ijser.org/researchpaper/A-Comparative-Study-of-ACID-and-BASE-in-Database-Transaction-Processing.pdf