**What is claimed is:**

1. A distributed computing system comprising:

a primary cluster and a secondary cluster, each cluster maintaining a plurality of shards,

said primary cluster operable to:

receive a set of transactions, each transaction comprising corresponding updates to at least one shard in said primary cluster, followed by a commit command to indicate end of updates in the transaction;

process each transaction of said set of transactions by:

storing each update as a corresponding provisional record in a respective shard of said primary cluster;

processing said commit command for the transaction at a corresponding commit time; and

applying said commit command, after said processing, to each of said respective shards, by converting each provisional record of the transaction to a corresponding regular record in the corresponding shard, each regular record being stored associated with the corresponding commit time for the transaction, and

asynchronously propagate the updates for the shards to said secondary cluster, along with an apply safe time for each shard in said primary cluster,

wherein said apply safe time propagated for a shard in said primary cluster indicates a time instance such that all transactions related to the shard and having commit time less than said time instance have been applied in the shard;

said secondary cluster operable to:

store each of the asynchronously received updates in a corresponding shard of said secondary cluster;

determine a cluster safe time as the minimal value of all the apply safe times corresponding to all of said plurality of shards in said primary cluster;

receive a read request directed to said secondary cluster for data stored in said secondary cluster;

identify a set of regular records having corresponding commit times less than or equal to said cluster safe time; and

provide said set of regular records as a response to said read request.

2. The distributed computing system of claim 1, wherein no transactions related to said shard are committed by a first time instance, wherein said apply safe time is set to any value up to and including said first time instance.

3. The distributed computing system of claim 1, wherein a first subset of transactions related to said shard are committed but not applied by a second time instance, wherein said apply safe time is set to a value which is less than or equal to a minimum of commit times of said first subset of transactions.

4. The distributed computing system of claim 1, wherein all transactions committed by a third time instance related to said shard are applied by said third time instance, wherein said apply safe time is set to any value up to and including said third time instance.

5. The distributed computing system of claim 1, wherein said secondary cluster sends a plurality of get-changes requests corresponding to respective shards of said primary cluster, and wherein each shard in said primary cluster transmits the corresponding updates in response to the respective get-changes request asynchronously,

wherein the corresponding updates includes provisional records, and regular records with corresponding commit times.

6. The distributed computing system of claim 5, wherein in response to a get-changes request directed to a shard in said primary cluster, wherein a shard leader in said primary cluster is operable to:

identify a leader safe time;

determine one or more transactions to be resolved based on the provisional records stored in said shard;

send get-transaction-status requests to one or more transaction coordinators;

receive responses to said get-transaction-status requests indicating statuses of whether said one or more transactions are still in progress, committed or aborted;

generate one or more apply actions to convert the provisional records of committed transactions to corresponding regular records;

compute a current end of WAL (write-ahead log) location; and

set said apply safe time for said shard as being equal to said leader safe time if all records until said current end of WAL location have been sent to said secondary cluster.

7 (Currently Amended): The distributed computing system of claim 1, where each of said primary cluster and said secondary cluster implements a corresponding distributed database,

wherein each of said plurality of shards corresponds to one of a plurality of tablets in said distributed database, wherein each tablet stores data corresponding to a unique set of keys.

8. A method for providing transactional consistency in a secondary cluster, said method being performed in said primary cluster, said method comprising:

receiving a set of transactions, each transaction comprising corresponding updates to at least one shard in said primary cluster, followed by a commit command to indicate end of updates in the transaction;

processing each transaction of said set of transactions by:

storing each update as a corresponding provisional record in a respective shard of said primary cluster;

processing said commit command for the transaction at a corresponding commit time; and

applying said commit command, after said processing, to each of said respective shards, by converting each provisional record of the transaction to a corresponding regular record in the corresponding shard, each regular record being stored associated with the corresponding commit time for the transaction, and

asynchronously propagating the updates for the shards to said secondary cluster, along with an apply safe time for each shard in said primary cluster,

wherein said apply safe time propagated for a shard in said primary cluster indicates a time instance such that all transactions related to the shard and having commit time less than said time instance have been applied in the shard.

9. The method of claim 8, wherein no transactions related to said shard are committed by a first time instance, wherein said apply safe time is set to any value up to and including said first time instance.

10. The method of claim 8, wherein a first subset of transactions related to said shard are committed but not applied by a second time instance, wherein said apply safe time is set to a value which is less than or equal to a minimum of commit times of said first subset of transactions.

11. The method of claim 8, wherein all transactions committed by a third time instance related to said shard are applied by said third time instance, wherein said apply safe time is set to any value up to and including said third time instance.

12. The method of claim 8, further comprising:

receiving, from said secondary cluster, a plurality of get-changes requests corresponding to respective shards of said primary cluster; and

transmitting asynchronously, from each shard in said primary cluster, the corresponding updates in response to the respective get-changes request,

wherein the corresponding updates includes provisional records and regular records with corresponding commit times.

13. The method of claim 12, wherein in response to a get-changes request directed to a shard in said primary cluster, said method further comprises:

retrieving a leader safe time from a leader node in said primary cluster;

determining one or more transactions to be resolved based on said leader safe time and the provisional records stored in said shard;

sending resolve intent requests to shards participating in said one or more transactions;

receiving responses to said resolve intent requests indicating statuses of whether the provisional records corresponding to said one or more transactions already committed, have been converted to regular records; and

computing said apply safe time for said shard as being equal to said leader safe time if said statuses indicate that the conversion is successful.

14. The method of claim 12, wherein in response to a get-changes request directed to a shard in said primary cluster, said method further comprising:

identifying, by a shard leader in said primary cluster, a leader safe time;

determining, by said shard leader, one or more transactions to be resolved based on the provisional records stored in said shard;

sending, by said shard leader, get-transaction-status requests to one or more transaction coordinators;

receiving, by said shard leader, responses to said get-transaction-status requests indicating statuses of whether said one or more transactions are still in progress, committed or aborted;

generating, by said shard leader, one or more apply actions to convert the provisional records of committed transactions to corresponding regular records;

computing, by said shard leader, a current end of WAL (write-ahead log) location; and

setting, by said shard leader, said apply safe time for said shard as being equal to said leader safe time if all records until said current end of WAL location have been sent to said secondary cluster.

15. A non-transitory machine-readable medium storing one or more sequences of instructions providing transactional consistency in a secondary cluster, wherein execution of said one or more instructions by one or more processors contained in a node of said secondary cluster causes said node to perform the actions of:

receiving asynchronously from a primary cluster, updates to a plurality of shards maintained in a primary cluster, along with an apply safe time for each shard in said primary cluster,

wherein said apply safe time propagated for a shard in said primary cluster indicates a time instance such that all transactions related to the shard and having commit time less than said time instance have been applied in the shard;

storing each of the asynchronously received updates in a corresponding shard of said secondary cluster;

determining a cluster safe time as the minimal value of all the apply safe times corresponding to all of said plurality of shards in said primary cluster;

receiving a read request directed to said secondary cluster for data stored in said secondary cluster;

identifying a set of regular records having corresponding commit times less than or equal to said cluster safe time; and

providing said set of regular records as a response to said read request.