ClustrixDB - High Level Architectural Overview

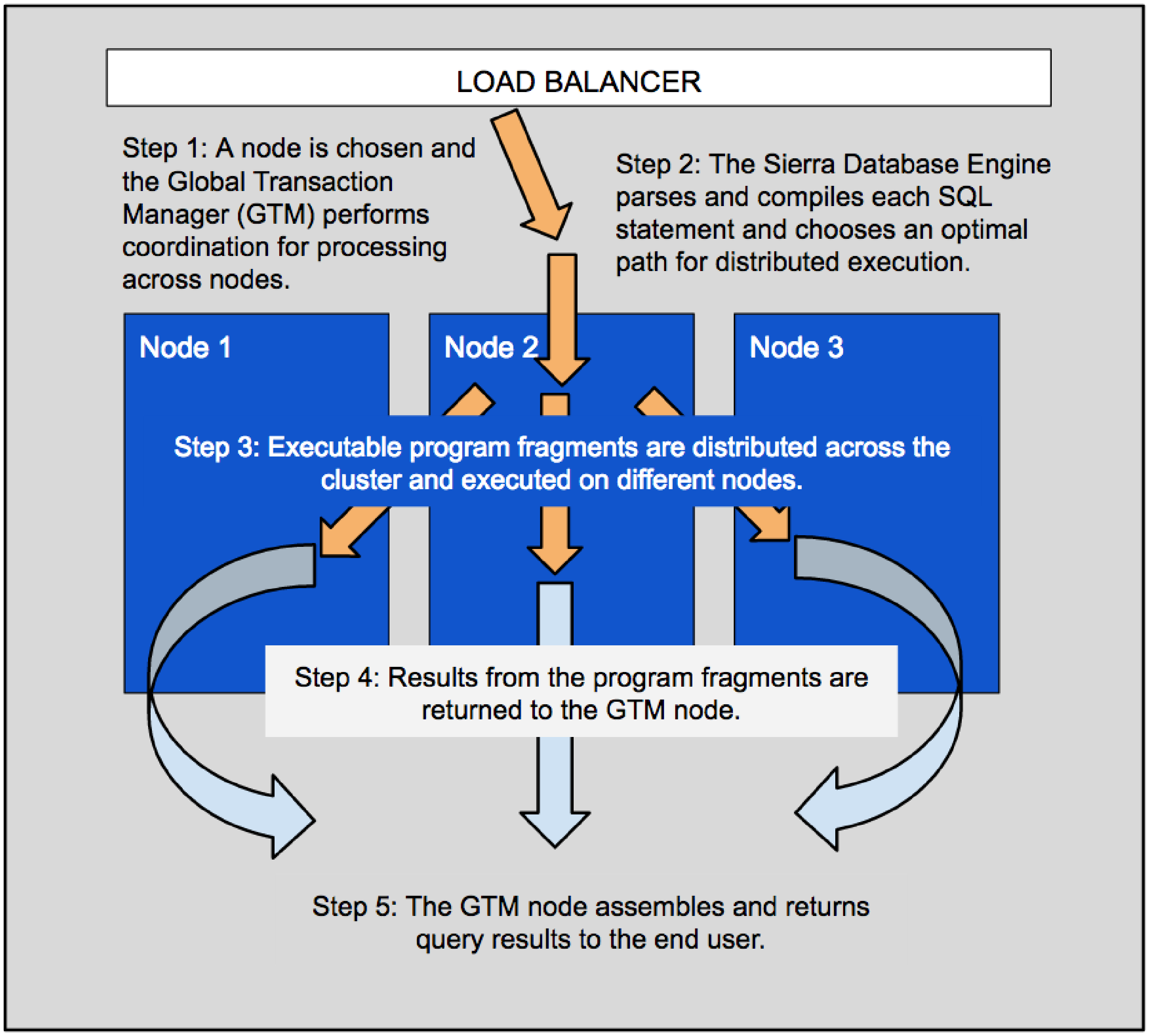
ClustrixDB is a clustered RDBMS that ensures ACID compliance for transaction processing while simultaneously providing easy scalability and fault tolerance. A ClustrixDB cluster is comprised of three or more nodes (networked homogeneous servers).

ClustrixDB utilizes a shared nothing architecture. Each node within a cluster can perform any read or write. To add capacity to your database, simply add more nodes.

The primary components of ClustrixDB that help achieve performance and scale are:

- The Global Transaction Manager (GTM), which coordinates the processing of a given transaction.
- The Rebalancer, which automatically distributes data across the cluster.
- The Sierra Database Engine, which determines an optimal query execution plan and then executes that plan on distributed data.

The diagram below shows how a typical query is processed by ClustrixDB.



Step 1: A node is chosen and the Global Transaction Manager (GTM) performs coordination for processing across nodes.

Step 2: The Sierra Database Engine parses and compiles each SQL statement and chooses an optimal path for distributed execution.

Step 3: Executable program fragments are distributed across the cluster and executed on different nodes.

Step 4: Results from the program fragments are returned to the GTM node.

Step 5: The GTM node assembles and returns query results to the end user.

Global Transaction Manager

Query processing within ClustrixDB begins when one node of the cluster is selected as the Global Transaction Manager (GTM), typically via a load balancer that distributes connections across the cluster. The GTM then manages all aspects of that transaction by directing each step of query execution, confirming that each step completes successfully, then collecting and finalizing the query results before returning them to the caller.

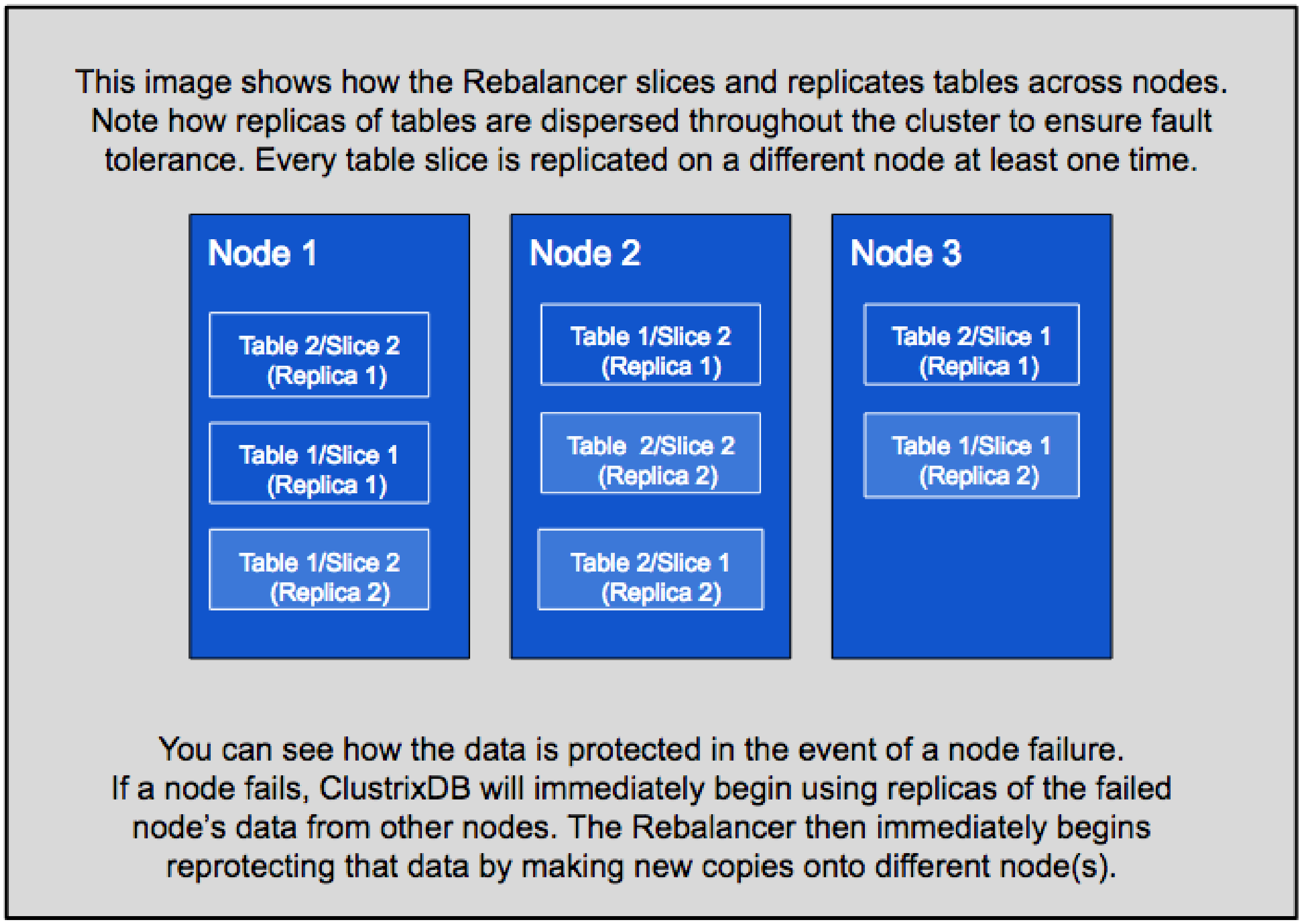
ClustrixDB compiles queries into executable query fragments that the GTM distributes to the appropriate node(s) for execution. As intermediate results become available, they are returned to the GTM. Once all query fragments have been executed successfully, the GTM finalizes the results and returns them to the client, application, or user.

Rebalancer

Distributed processing would not be possible nor necessary if ClustrixDB's data had not previously been distributed throughout the cluster. To accomplish this, ClustrixDB utilizes a patented data distribution methodology that is administered by its Rebalancer. The Rebalancer arranges data within the cluster to ensure that reads and writes are always balanced. It also guarantees that multiple copies (replicas) of data are maintained throughout the cluster to ensure fault tolerance. If a node is lost due to an unexpected failure, no data will be lost. The Rebalancer will automatically ensure that redundant replicas are created and maintained. It further accommodates the changing size of a cluster by re-balancing data to new nodes as they are added and moving data off nodes that are marked for removal, all while the database remains online.

The Rebalancer uses a consistent hashing algorithm to assign each table row to a given "slice" of that table and provides a map of all slices to every node. This allows ClustrixDB to quickly and easily ascertain the location of relevant data. The Rebalancer runs continuously in the background without interfering with ongoing production processing.

Although data is sliced and distributed to numerous nodes of a cluster, a database table will always appear as a single logical unit to an application. Clustrix uses a simple SQL interface and no special application programming is necessary to access data distributed throughout a ClustrixDB cluster.



This image shows how the Rebalancer slices and replicates tables across nodes. Note how replicas of tables are dispersed throughout the cluster to ensure fault tolerance. Every table slice is replicated on a different node at least one time.

Node 1
Table 2/Slice 2 (Replica 1)
Table 1/Slice 1 (Replica 1)
Table 1/Slice 2 (Replica 2)
Node 2
Table 1/Slice 2 (Replica 1)
Table 2/Slice 2 (Replica 2)
Table 2/Slice 1 (Replica 2)
Node 3
Table 2/Slice 1 (Replica 1)
Table 1/Slice 1 (Replica 2)

You can see how the data is protected in the event of a node failure. If a node fails, ClustrixDB will immediately begin using replicas of the failed node's data from other nodes. The Rebalancer then immediately begins reprotecting that data by making new copies onto different node(s).

Please read about ClustrixDB's Rebalancer for additional information.

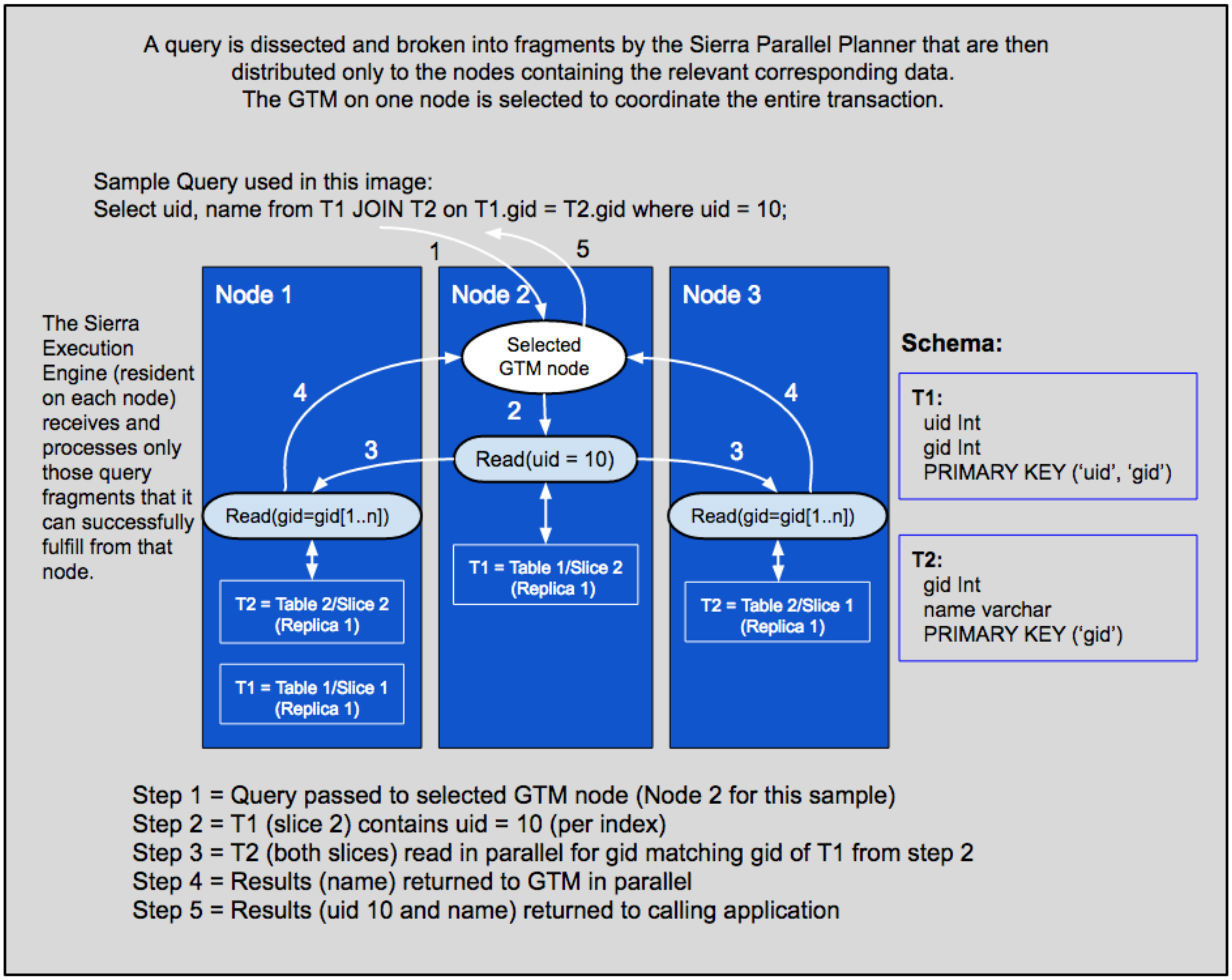
Sierra Database Engine

Sierra is the SQL engine of ClustrixDB that handles query planning and execution. It has been specifically designed to work in a distributed, shared nothing environment while facilitating access to distributed data as efficiently as possible. The Sierra Database Engine consists of two parts:

- The Sierra Parallel Planner determines an optimal execution plan for a SQL statement.
- The Sierra Distributed Execution Engine executes query fragments based on those plans and provides intermediate results.

Sierra Parallel Planner

Sierra Parallel Planner is a cost-based optimizer that uses probability statistics, data volumes, indices, and overhead of query operators to determine the most efficient query plan. A key differentiating feature of ClustrixDB's planner is that it determines this plan while taking into account the distribution of the data across the cluster.



A query is dissected and broken into fragments by the Sierra Parallel Planner that are then distributed only to the nodes containing the relevant corresponding data.
The GTM on one node is selected to coordinate the entire transaction.

Sample Query used in this image:
Select uid, name from T1 JOIN T2 on T1.gid = T2.gid where uid = 10;

The Sierra Execution Engine (resident on each node) receives and processes only those query fragments that it can successfully fulfill from that node.

1
5
Node 1
Node 2
Node 3
Selected GTM node
2
4
3
Read(uid = 10)
Read(gid=gid[1..n])
T2 = Table 2/Slice 2 (Replica 1)
T1 = Table 1/Slice 1 (Replica 1)
T1 = Table 1/Slice 2 (Replica 1)
T2 = Table 2/Slice 1 (Replica 1)
Schema:
T1:
uid Int
gid Int
PRIMARY KEY ('uid', 'gid')
T2:
gid Int
name varchar
PRIMARY KEY ('gid')

Step 1 = Query passed to selected GTM node (Node 2 for this sample)
Step 2 = T1 (slice 2) contains uid = 10 (per index)
Step 3 = T2 (both slices) read in parallel for gid matching gid of T1 from step 2
Step 4 = Results (name) returned to GTM in parallel
Step 5 = Results (uid 10 and name) returned to calling application

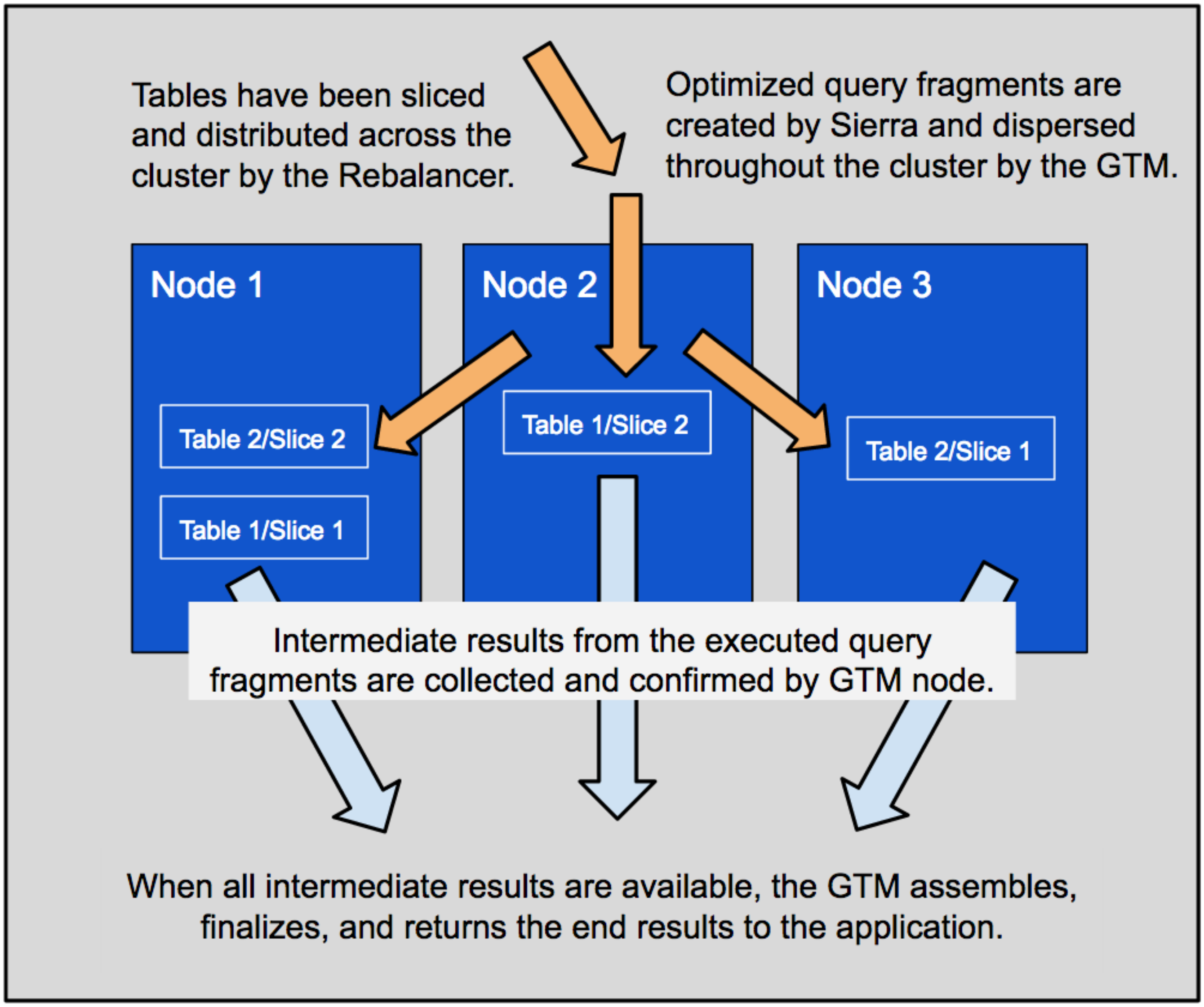
For additional samples that demonstrate how this query fragmentation works, please see A New Approach to Scale-Out RDMS.

Sierra Distributed Execution Engine

Once the Sierra Planner determines the optimal plan for a query, it is compiled into machine-executable query fragments. Those compiled query fragments are then executed across different nodes in the cluster, providing efficiency and increased concurrency of execution. Once execution on each node is complete, the results are returned to the GTM node, which then combines the partial results and returns the final result set to the user.

Selected aggregate processing is also distributed. When advantageous, computations are fragmented and distributed the same as other queries except that partial aggregates (SUM, MAX, MIN, AVG, etc.) are calculated on each node's distributed data first. The intermediate results are then coalesced by the GTM to produce the final result.

Let's look one final time at the query distribution in ClustrixDB - this time with the distributed data in place.


Tables have been sliced and distributed across the cluster by the Rebalancer.
Optimized query fragments are created by Sierra and dispersed throughout the cluster by the GTM.
Node 1
Node 2
Node 3
Table 2/Slice 2
Table 1/Slice 1
Table 1/Slice 2
Table 2/Slice 1
Intermediate results from the executed query fragments are collected and confirmed by GTM node.
When all intermediate results are available, the GTM assembles, finalizes, and returns the end results to the application.

Conclusion

ClustrixDB uses automatic data distribution, a sophisticated query planner, and a distributed execution model to provide scalability and concurrency in an ACID compliant RDBMS. To accomplish this, ClustrixDB utilizes many of the same techniques used by other Massively Parallel Processing (MPP) databases: It uses Paxos for distributed transaction resolution, and Multi-Version Concurrency Control (MVCC) to prevent transaction conflicts. With the aid of the major components outlined above, ClustrixDB provides this distributed execution with a simple SQL interface while also providing scalability, efficiency, and fault tolerance.