New installation View : Step 2

installation negative test : Step 6

MYSQL:

Synchronization --

http://dev.mysql.com/doc/workbench/en/wb-database-synchronization.html

possible to synchronize a model in MySQL Workbench with a live database. By default, the synchronization process will change the live database to be the same as the model, but this is configurable during the synchronization process.

which database system there are ?

jdbc and persistence compatible

2pc works in database

how cluster mode working for them

consistency

think for validating how caching can be dealt between client and server

1. **MYSQL Feature:**

Speed, platform independence, foreign key constraints, transaction, replication, full text search, sql compatibility, client-server architecture

1. **Synchronization, Consistency, 2pc :**

We are using MySql rdbms for providing consistent synchronization through two phase commit. Mysql provides server side support for Distributed Transactions (XA transactions). On the client side, there are no special requirements. The XA interface to a MySQL server consists of SQL statements that begin with the XA keyword. MySQL client programs must be able to send SQL statements and to understand the semantics of the XA statement interface. They do not need be linked against a recent client library. Older client libraries also will work.

XA supports distributed transactions, that is, the ability to permit multiple separate transactional resources to participate in a global transaction. Transactional resources often are RDBMSs but may be other kinds of resources. A global transaction involves several actions that are transactional in themselves, but that all must either complete successfully as a group, or all be rolled back as a group. In essence, this extends ACID properties “up a level” so that multiple ACID transactions can be executed in concert as components of a global operation that also has ACID properties.

Applications that use global transactions involve one or more Resource Managers and a Transaction Manager. A Resource Manager (RM) provides access to transactional resources. A database server is one kind of resource manager. It must be possible to either commit or roll back transactions managed by the RM. A Transaction Manager (TM) coordinates the transactions that are part of a global transaction. It communicates with the RMs that handle each of these transactions. The individual transactions within a global transaction are “branches” of the global transaction. Global transactions and their branches are identified by a naming scheme described later. The MySQL implementation of XA MySQL enables a MySQL server to act as a Resource Manager that handles XA transactions within a global transaction. A client program that connects to the MySQL server acts as the Transaction Manager.

The process for executing a global transaction uses two-phase commit (2PC). This takes place after the actions performed by the branches of the global transaction have been executed.

1. In the first phase, all branches are prepared. That is, they are told by the TM to get ready to commit. Typically, this means each RM that manages a branch records the actions for the branch in stable storage. The branches indicate whether they are able to do this, and these results are used for the second phase.

2. In the second phase, the TM tells the RMs whether to commit or roll back. If all branches indicated when they were prepared that they will be able to commit, all branches are told to commit. If any branch indicated when it was prepared that it will not be able to commit, all branches are told to roll back.

In some cases, a global transaction might use one-phase commit (1PC). For example, when a Transaction Manager finds that a global transaction consists of only one transactional resource (that is, a single branch), that resource can be told to prepare and commit at the same time.

These support are available for InnoDB storage engine. InnoDB is a high-reliability and high-performance storage engine for MySQL. InnoDB has been designed for CPU efficiency and maximum performance when processing large data volumes. Its design follows the ACID model, with transactions featuring commit, rollback, and crash-recovery capabilities to protect user data. Row-level locking (without escalation to coarser granularity locks) and Oracle-style consistent reads increase multi-user concurrency and performance. InnoDB tables arrange your data on disk to optimize common queries based on primary keys. Each InnoDB table has a primary key index called the clustered index that organizes the data to minimize I/O for primary key lookups. To maintain data integrity, InnoDB also supports FOREIGN KEY referential-integrity constraints. It is also possible to mix InnoDB tables with tables from other MySQL storage engines, even within the same statement. For example, you can use a join operation to combine data from InnoDB and MEMORY tables in a single query.

1. **Adding new node or cluster:-**

http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster-online-add-node.html

http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster-online-add-node-basics.html

https://dev.mysql.com/doc/mysql-cluster-excerpt/5.1/en/mysql-cluster-online-add-node-example.html

Steps:

1. Edit the cluster configuration config.ini file, adding new [ndbd] sections corresponding to the nodes to be added. In the case where the cluster uses multiple management servers, these changes need to be made to all config.ini files used by the management servers.
2. Perform a rolling restart of all MySQL Cluster management servers. (**using –reload and –initial**)
3. Perform a rolling restart of all existing MySQL Cluster data nodes. It is not necessary (or usually even desirable) to use **--initial** when restarting the existing data nodes.
4. perform a rolling restart of any SQL or API nodes connected to the MySQL Cluster.
5. Start the new data nodes. The new data nodes may be started in any order. They can also be started concurrently, as long as they are started after the rolling restarts of all existing data nodes have been completed, and before proceeding to the next step
6. Execute one or more CREATE NODEGROUP commands in the MySQL Cluster management client to create the new node group or node groups to which the new data nodes will belong.
7. Redistribute the cluster's data among all data nodes (including the new ones) by issuing an [ALTER ONLINE TABLE ... REORGANIZE PARTITION](http://dev.mysql.com/doc/refman/5.1/en/alter-table.html) statement in the [mysql](http://dev.mysql.com/doc/refman/5.1/en/mysql.html" \o "4.5.1 mysql — The MySQL Command-Line Tool) client for each [NDBCLUSTER](http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster.html) table..
8. Reclaim the space freed on the “old” nodes by issuing, for each [NDBCLUSTER](http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster.html) table, an [OPTIMIZE TABLE](http://dev.mysql.com/doc/refman/5.1/en/optimize-table.html)statement in the [mysql](http://dev.mysql.com/doc/refman/5.1/en/mysql.html" \o "4.5.1 mysql — The MySQL Command-Line Tool) client.

**Deleting Node:**

http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster-mgm-client-commands.html

1. **Limitation:-**

http://dev.mysql.com/doc/refman/5.0/en/mysql-cluster-limitations-exclusive-to-cluster.html

**Machine architecture.** All machines used in the cluster must have the same architecture**.**

**Adding and dropping of data nodes**: Online adding or dropping of data nodes is not currently possible. In such cases, the entire cluster must be restarted.

**Backup and restore between architectures.** It is also not possible to perform a Cluster backup and restore between different architectures.

**Online schema changes**. It is not possible to make online schema changes such as those accomplished using ALTER TABLE or CREATE INDEX, as the NDB Cluster engine does not support autodiscovery of such changes.

Many database systems offer the possibility of defining custom data types. MySQL does not support such functionality, nor is any currently planned.

1. **Client/Server Architecture:** MySQL is a client/server system. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they query data, save changes, etc. The clients can run on the same computer as the server or on another computer (communication via a local network or the Internet). Almost all of the familiar large database systems (Oracle, Microsoft SQL Server, etc.) are client/server systems. These are in contrast to the file-server systems, which include Microsoft Access, dBase and FoxPro. The decisive drawback to file-server systems is that when run over a network, they become extremely inefficient as the number of users grows.
2. **MySQL Alternatives are** ☹ . <http://searchitchannel.techtarget.com/feature/MySQL-alternatives>