



# Replication Guide

**CSQL**  
Main Memory Database Cache

# Table of Contents

<b>1. BEFORE YOU START.....</b>	<b>4</b>
1.1. WHAT IS CSQL.....	4
1.2. CSQL COMPONENTS.....	4
1.2.1. CSQL Main Memory Database	4
1.2.2. CSQL Cache	4
1.2.3. CSQL Replication	5
1.3. INFORMATION IN THIS MANUAL.....	5
1.4. TYPES OF USERS.....	5
1.5. CONVENTIONS.....	5
1.5.1. Typographic Rule	5
1.5.2. Syntax Annotation	6
1.6. ADDITIONAL DOCUMENTATION .....	7
1.6.1. On-Line Help	7
1.6.2. Documentation Notes, Release Notes	7
1.7. COMPLIANCE WITH INDUSTRY STANDARDS .....	7
1.7.1. SQL-92	7
1.7.2. ODBC Level 2	7
1.7.3. JDBC 3.0	8
1.8. CSQL SELECTIVE INFORMATION REFERENCES .....	8
1.8.1. Mailing lists	8
1.8.2. Forums	8
<b>2. INTRODUCING CSQL REPLICATION.....</b>	<b>9</b>
2.1. INTRODUCING REPLICATION CONCEPTS AND FEATURES.....	9
2.1.1. Data-Replication Types	9
2.1.1.1. Synchronous Data Replication	9
2.1.1.2. Asynchronous Data Replication	10
2.1.2. Data-Replication Capture Mechanism	11
2.1.2.1. Log-Based Data Capture	11
2.1.3. CSQL Replication Features	11
2.1.3.1. Replication Granularity	12
2.1.3.2. Transaction Propagation modes	12
2.1.3.3. High Performance	12
2.1.3.4. High Availability	13
2.1.3.5. Load Balancing	13
2.1.3.6. Recovery	13

2.1.3.7.	Consistent Information Delivery	14
2.1.3.8.	Well-Situated, Centralize Administration	14
2.1.3.9.	Platform Supported	15
2.1.4.	CSQL Replication Benefits	15
2.1.4.1.	Performance and Consistency	15
2.1.4.2.	Scalability	15
<b>3.</b>	<b><u>OVERVIEW OF REPLICATION</u></b>	<b>16</b>
<b>3.1.</b>	<b>HOW REPLICATION REPLICATES DATA</b>	<b>16</b>
3.1.1.	Catch Transactions	16
3.1.2.	Circulate Data for Replication	16
3.1.3.	Apply Replicated Data	16
<b>4.</b>	<b><u>DEPLOYMENT OPTIONS</u></b>	<b>17</b>
<b>4.1.</b>	<b>ACTIVE-STANDBY REPLICATION</b>	<b>17</b>
<b>4.2.</b>	<b>ACTIVE-ACTIVE REPLICATION</b>	<b>17</b>
<b>4.3.</b>	<b>LOAD BALANCING CLUSTER FOR READ</b>	<b>18</b>
<b>4.4.</b>	<b>BALANCING CLUSTER FOR READ/WRITE</b>	<b>19</b>
<b>4.5.</b>	<b>LOAD BALANCING CLUSTER WITH DISTRIBUTOR</b>	<b>20</b>
<b>4.6.</b>	<b>LOAD BALANCING CLUSTER WITH DISTRIBUTED DATA</b>	<b>21</b>
<b>5.</b>	<b><u>SETTING UP YOUR REPLICATION ENVIRONMENT</u></b>	<b>23</b>
<b>5.1.</b>	<b>OPERATIONAL RESTRICTIONS</b>	<b>23</b>
<b>5.2.</b>	<b>SQL STATEMENT USE</b>	<b>23</b>
<b>5.3.</b>	<b>DISTRIBUTED TRANSACTION</b>	<b>ERROR! BOOKMARK NOT DEFINED.</b>
5.3.1.	Logical-Log Records	<b>Error! Bookmark not defined.</b>
<b>5.4.</b>	<b>CONFIGURATION PARAMETER</b>	<b>23</b>
5.4.1.	Setting up parameters in csq1. conf file	25
<b>6.</b>	<b><u>ADMINISTERING AND MONITORING REPLICATION</u></b>	<b>28</b>
<b>6.1.</b>	<b>ACTIVE – STANDBY</b>	<b>28</b>
6.1.1.	Synchronous Replication	30
6.1.2.	Asynchronous Replication	31
<b>6.2.</b>	<b>ACTIVE – ACTIVE</b>	<b>32</b>
6.2.1.	Synchronous Replication	32
6.2.2.	Asynchronous Replication	33
<b>7.</b>	<b><u>TOOL FOR REPLICATION</u></b>	<b>34</b>

## 1. Before you start

This chapter gives brief introduction to what is CSQL, Information in this manual, and Conventions used in this manual.

### 1.1. What is CSQL

CSQL Main Memory Database is an easily accessible and powerful database management system, which includes the CSQL Main Memory Database, and Cache to MySQL, PostgreSQL, Sybase and Oracle disk based database management system.

It typically resides in the middle tier alongside applications and store data in main memory rather than disk, allowing it to be retrieved very quickly. It is used for applications where extremely fast response times are critical. Investment houses use CSQL to store current stock price information, which they need to quickly make trading decisions. Mobile operators use the software for billing, so that they can quickly determine whether a caller has enough prepaid credit to place a call.

CSQL could be used for a wide variety of applications ranging from single-user system to enterprise-wide multi-user installations with hundreds of concurrent connections.

### 1.2. CSQL components

This Introduction provides an overview of the three major components that make up CSQL. These components include:

- CSQL Main memory database
- CSQL Cache
- CSQL Replicator

#### 1.2.1. CSQL Main Memory Database

CSQL Main Memory Database derives much of its power from its unique architecture. At the core, the CSQL database storage engine provides the complete set of services – including data storage, concurrency management, transactions, and process management – needed to build efficient database management system. It provides standard interfaces for data manipulation that includes SQL, JDBC and ODBC.

#### 1.2.2. CSQL Cache

CSQL Cache option enables customers to significantly improve application response times and throughput. Based on the CSQL MMDB, CSQL Cache delivers a real-time, dynamic, updateable cache for frequently accessed data in MySQL, Postgres, Sybase and Oracle database.

For performance critical applications in industries such as Telecom, Process Control, Airline Reservation, Stock Market, HealthCare, etc.,

the CSQL Cache option delivers application response times in microseconds by bringing the frequently accessed data closer to the application, and by executing SQL requests in the CSQL MMDB.

We found that CSQL and its associated suite of products to be between 20 and 40 times faster than traditional database system.

Visit Benchmark result page at <http://www.csqldb.com> for more information.

### **1.2.3. CSQL Replication**

CSQL Replication offers complete data protection through automated, real-time, zero-latency fail over and recovery with high availability and load balancing cluster for CSQL MMDB. It provides both synchronous and asynchronous transaction propagation to peer instances. It enables scaling application throughput by sharing the read load across replicated CSQL instances.

## **1.3. Information in this Manual**

This manual contains information to help you understand the concepts of data replication, configure your own replication system, administer and manage data replication throughout your enterprise.

The CSQL Replication feature provides a cost effective, efficient means to replicate data throughout your open-systems enterprise. CSQL Enterprise Replication is a client/server application that operates on Linux and Solaris for this release. This provides mechanisms to Synchronously or Asynchronously replicate data throughout your enterprise.

## **1.4. Types of Users**

This manual is for database server administrators.

This manual assumes that you have the following background:

- Basic knowledge required for Linux Operating System, and the utilities that your operating systems provides.
- Some experience working with relational databases or exposure to database concepts.
- Some experience with database server administration, operating system administration, or network administration.

## **1.5. Conventions**

The below typographic and syntax rules have been maintained throughout the guide.

### **1.5.1. Typographic Rule**

This manual uses the following typographic rule.

Table 1.1. Typographic Rule

<b>FORMAT</b>	<b>USED FOR</b>
Main Memory Database	This font is used for ordinary text.
<code>SELECT * FROM T1</code>	This font is used for SQL Statements and Program code.
UNIQUE	This font with uppercase letter indicates SQL keywords and data types.
csql.conf	This font indicates file name and path.
build.ksh	This font is used for command lines
SQLFetch()	This font is used for function names
Java.Sql.Statement	This font is used for interface, classes and header files names.
Test	This font is used for directory name.

### 1.5.2. Syntax Annotation

This manual uses the following syntax annotation conventions.

Table 1.2. Syntax Annotation

<b>FORMAT</b>	<b>USED FOR</b>
[ ]	This square bracket indicates that item in command line is optional
{ }	Curly braces indicates that one must choose one of the items separated by a vertical bar (   ) in a command line.
	A vertical bar separates arguments
...	An ellipsis indicates that arguments can be repeated several times
.	This indicates that continuation of previous lines of code
.	
.	
\$	This dollar sign indicates the Linux prompt.
#	The pound sign indicates the Linux root prompt.

## 1.6. Additional Documentation

The following sections describe the on-line documents that supplement the information in this manual. Please refer to these documents before you begin using your database server. They contain absolutely necessary information about installation, configuration and performance issues.

### 1.6.1. On-Line Help

The set of CSQL documentations is available on the CSQL web site:  
[http://www.csqldb.com/pro\\_documentation.html](http://www.csqldb.com/pro_documentation.html)

### 1.6.2. Documentation Notes, Release Notes

The following on-line documents appear in the installation package  
\$csql/docs directory.

File	Purpose
<a href="#">User Manual</a>	Describes concepts, components, basic usages, and configuration details of CSQL on Linux and Solaris platform
<a href="#">Programmer Guide</a>	This is for application developers who use CSQL drivers to access data in the database programmatically. It covers JDBC, ODBC and proprietary SQL interface.
<a href="#">Cache Guide</a>	Describes caching functionality of CSQL and configuration settings required to set up caching for MySQL, Postgres and Oracle DBMS.
<a href="#">Replication Guide</a>	This manual describes replication functionality of CSQL and configuration settings required to set up replication with CSQL MMDB or with CSQL Cache.

## 1.7. Compliance with Industry Standards

### 1.7.1. SQL-92

The American National Standards Institute (ANSI) has established a set of industry standards for SQL. CSQL SQL-based products are fully compliant with SQL-92 Entry Level, which is identical to ISO 9075:1992.

### 1.7.2. ODBC Level 2

The CSQL ODBC Driver supports ODBC version 2.5 and 3.0 Level 2 only. The Level 2 is all that is necessary to do standard operations and your application might want to limit up to level 2 ODBC calls. Refer *CSQL Programmers' Guide* to know in detail about ODBC APIs.

### 1.7.3. JDBC 3.0

The CSQL JDBC Driver supports JDBC 3.0 APIs. Its functionality includes parameters, selects, batch updates, programmatic inserts, updates, deletes.

## 1.8. CSQL Selective Information References

For providing support to CSQL users, below communication channels are dedicated for customer queries.

### 1.8.1. Mailing lists

The primary mechanism for CSQL communication is through its mailing lists. Anyone who uses this product shall participate in user mailing lists. You can search for the archive of past discussions before you post your question to the mailing lists in [http://sourceforge.net/mailarchive/forum.php?forum\\_name=csql-users](http://sourceforge.net/mailarchive/forum.php?forum_name=csql-users). The main channel for user support is [csql-users@lists.sourceforge.net](mailto:csql-users@lists.sourceforge.net) mailing list. As is usual with mailing lists, be prepared to wait for an answer.

Please summarize any off-list knowledge gained and post it for the benefit of all. For example, if a user asks a question and gets response, post that to the above-mentioned site also.

### 1.8.2. Forums

CSQL Users shall use forums for posting their queries for support in the below mentioned URL: [http://sourceforge.net/forum/forum.php?forum\\_id=562614](http://sourceforge.net/forum/forum.php?forum_id=562614).

Issues related to installation and implementation shall be posted here to get answers from CSQL technical team.

## 2. Introducing CSQL Replication

CSQL Replication prevents your business from losing significant revenue and damaging your business's reputation by ensuring that your mission-critical data is highly available. Sharing corporate data is an important aspect of day-to-day business operations. Data replication generates and manages multiple copies of data at two or more sites, which allows an enterprise to share corporate data throughout its organization.

### 2.1. Introducing Replication Concepts and Features

CSQL Replication plays a key role in the disaster recovery process, providing recovery of mission-critical data for applications at an alternate site. As the number of data-intensive and externally accessed applications increases, so does the need for high availability and greater amounts of data. This requires IT organizations to understand and classify each application's availability and recovery requirements.

Defining application requirements is an essential step in creating a successful business recovery strategy. Two variables to consider when developing a recovery strategy are acceptable levels of downtime and freshness of data. CSQL replication addresses the needs for continuous availability and data consistency.

This section introduces the following information:

- Data-replication types
- Data-catch mechanisms
- CSQL Enterprise Replication features

#### 2.1.1. Data-Replication Types

CSQL Provides two types of data replication: Synchronous and Asynchronous and the data which replicated is transactional level data that has been committed at the source site.

##### 2.1.1.1. Synchronous Data Replication

In case of synchronous mode, source instance (where transaction originates) transaction commits returns only after the transaction is committed at source instance and other destination instances in the quorum (group of CSQL instances taking part in the replication process). That means the data to be replicated is updated immediately when the source data is updated.

In Figure1-1, The transaction committed at Instance-1, will commit the data at Instance-2 and Instance-3.

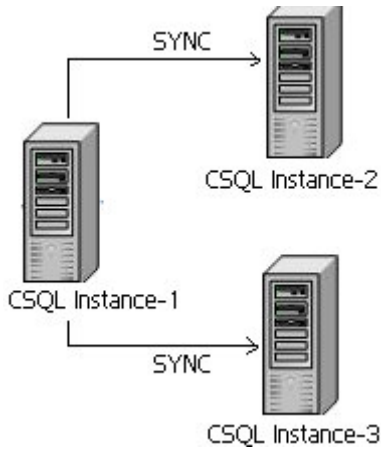


Fig 1-1. Synchronous Mode Replication

Synchronous data replication might be appropriate for applications that require immediate data synchronization. However, synchronous data replication requires that all hardware components and networks in the replication system be available at all times. It is difficult to guarantee the constant availability of all devices in a replication system, because hardware components and networks do fail at times. An alternative type of data replication is asynchronous data replication.

### 2.1.1.2. Asynchronous Data Replication

In case of asynchronous mode, source site transaction commit will return immediately after the transaction is committed at the source site and before returning from commit, transaction logs are generated and sent to the Replicator. The Replicator process takes care of applying these transactions at destination sites in the quorum.

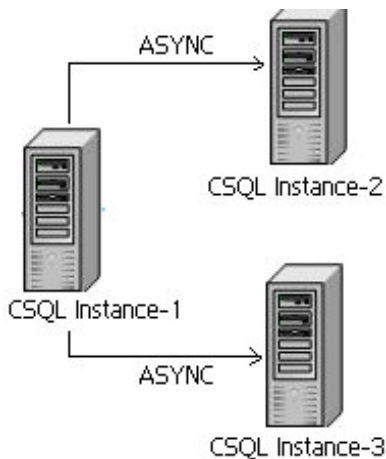


Fig 1-2. Asynchronous Data Replication

However, the data eventually synchronizes to the same value at all sites. The major benefit of this type of data replication is that if a particular database server fails, the

replication process can continue and the original transaction is not directly affected by replication.

Asynchronous replication is the most common type of data replication in open-system environments because it protects against the system and network failures.

### 2.1.2. Data-Replication Capture Mechanism

The most frequently used data-replication capture mechanism is log based.

#### 2.1.2.1. Log-Based Data Capture

Log-based data-capture systems operate as part of the normal database-logging process and add minimum overhead to the system.

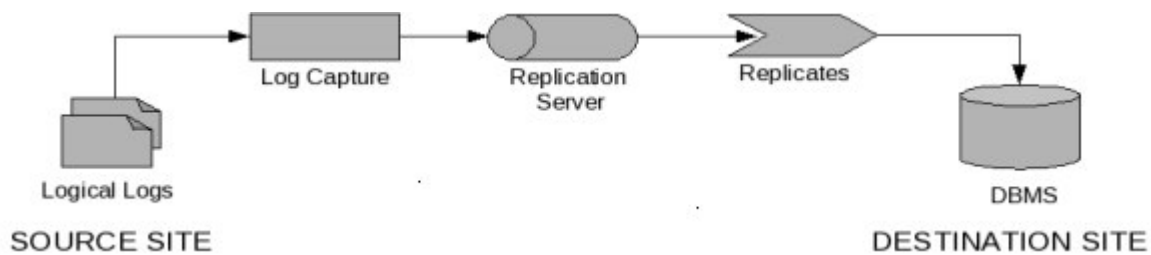


Fig 1-3. Log-Based Data Capture

The most efficient and cost-effective data replication is an asynchronous replication that uses a log-based transaction-capture mechanism.

### 2.1.3. CSQL Replication Features

CSQL Replication products use the following key elements to successfully replicate data throughout an open-system enterprise:

- Replication Granularity
- Transaction propagation modes
- High performance
- High availability
- Load Balancing
- Recovery
- Consistent information delivery
- Well-situated, centralized administration
- Flexible Architecture
- Platform Supported

### 2.1.3.1. Replication Granularity

CSQL supports replication at multiple granular levels. Database level replication, replicates all the tables in the database whereas table level replication allows applications to choose what tables to replicate in the quorum. [It also supports partial table replication in which only subset of the records are replicated between two sites.](#) Refer “Load Balancing Cluster with distributed data” for more information.

### 2.1.3.2. Transaction Propagation modes

Transactions are applied at destination sites in two modes namely, Synchronous and Asynchronous. The mode between two replicating sites need not be same. For example for two sites (site1 and site2 in replication quorum), site1 can propagate transactions to site2 in synchronous mode and site2 can propagate transactions to site1 in asynchronous mode or may not propagate transactions to site1.

In case of multi site cluster, application can choose the transaction propagation mode between each and every site in the quorum. Lets say you have four sites namely site1, site2, site3 and site4 in the quorum. Site1 and site2 can be connected using synchronous mode and site1 and site3 can be connected using asynchronous mode.

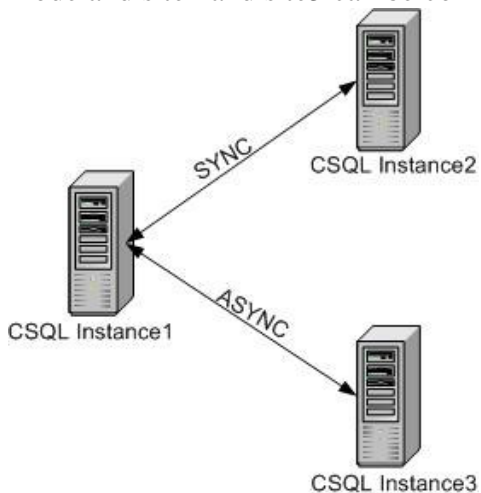


Fig 1-4. Both Sync and Async Mode Replication

### 2.1.3.3. High Performance

A replication system must not burden the source site, must use networks and all resources efficiently.

CSQL Replication uses an efficiently, log-based transaction-capture mechanism that is integrated in the database architecture.

Replication can operate in a LAN, WAN, or combined LAN/WAN configuration across TCP/IP network protocols. CSQL Replication uses the standard database server communication facility to establish network connectivity among the replicated database servers involved in replication.

#### **2.1.3.4. High Availability**

CSQL Provides sub second fail over as all transactions is applied at all the sites in the quorum. CSQL Replication provides continuous availability during planned maintenance operations such as software or hardware upgrade with no data loss.

A replication system must be reliable and must not expose business operations to computer system failures.

CSQL replication implemented asynchronous mode data replication. Asynchronous replication ensures that network and destination database server outages are tolerated. Replication also uses a delivery mechanism that stores data locally and then sends the data to the destination server in separate transactions.

In the event of a database server or network failure, the local database server can continue to service the local users, which provides a high degree of fault tolerance.

#### **2.1.3.5. Load Balancing**

CSQL Replication can be deployed as load balancing cluster with many sites to improve the throughput of the application. Transaction logs can be propagated by mediator components called ‘distributors’ whose job is to propagate transaction on behalf of the source site to the other entire destination site, reducing the load on the source site.

CSQL Replication supports intelligent load balancing techniques where subset of records is replicated rather than the complete table at every site. This mode is called as distributed replication, where each site contains records belonging to their zone and a centralized server, which contains the complete table for all zones. For more information refer “Load Balancing Cluster with distributed data” deployment option.

#### **2.1.3.6. Recovery**

When a site goes down for any reason (planned maintenance, crash, or disaster), all the other sites in the quorum retains all the transactions at their local site till the site which went down resumes its operation. When the site comes back, it receives transaction updates from all the sites (transactions which happened when this site was down) and gets sync with all the other sites in quorum. This recovery is automatic and does not require any manual intervention.

And when a site goes down connected with its destination site using synchronous mode transaction propagation, the mode is automatically switched to asynchronous thereby avoiding transactions to fail. (Otherwise all transaction fails due to transaction execution failure at destination site, which is down). This makes sure that no transaction fail because of site failure. When the down site resumes its operations, the mode shall be switched back to synchronous.

### **2.1.3.7. Consistent Information Delivery**

A replication system must protect data integrity. CSQL Replication employs transaction consistent replication, if transaction fails at any destination site connected using synchronous propagation mode, the transactions fails at the source site also. This ensures that two parallel conflicting transactions (For example, deducting balance from account) running in two different sites to fail and ensure consistency of data across sites in the quorum.

For asynchronous mode propagation, transaction conflicts at destination site are detected and logged into file so that the administrator can resolve conflicts. Conflict resolution could be additive, subtractive or replaceable based on the data on which conflict occurred. For Example in case of conflicting deposit, conflict resolution is additive whereas for withdrawal, it is subtractive.

CSQL Replication provides tools to check for consistency between sites connected in asynchronous mode to assist application for maintaining data consistency across quorum. If update conflicts occur, CSQL Replication provides built-in automatic conflict detection and resolution.

### **2.1.3.8. Well-Situated, Centralize Administration**

Administrators must be able to easily manage all the distributed components of the replication system from a single point of control.

CSQL Replication system does not impose model or methodology restrictions on the enterprise. A replication system should allow replications based on specific business and application requirements.

#### **Ownership Models:**

CSQL Enterprise Replication supports the following replication models:

- Primary target  
Unidirectional updates from a source database server to many destination database servers, or from many destination servers to a source database server.
- Workflow  
Asynchronous updates that move from site to site.
- Update anywhere  
A peer-to-peer update capability that allows users function operates autonomously even if systems or networks are not available.

#### **Operating Modes:**

Replication provides expert operating modes to the user. The expert mode supports all ownership models. CSQL replication does not restrict the user for Operating Modes.

### **2.1.3.9. Platform Supported**

- Linux operating system on Intel x86 architecture
- Linux operating system on Intel x86\_64 architecture
- Solaris operating system on Intel x86 architecture
- Solaris operating system on Sparc 32 architecture
- Solaris operating system on Sparc 64 architecture

### **2.1.4. CSQL Replication Benefits**

CSQL Replication is simple to use and requires minimal setup to get the data replicated. Requires no changes to application and minimal administration as CSQL has inbuilt self-healing mechanisms to tolerate faults of individual sites in the quorum.

#### **2.1.4.1. Performance and Consistency**

Load balancing database operations on multiple servers improves throughput by multifold. Most of the real time applications are read intensive and by employing cluster with multiple CSQL instances, load can be distributed among all of them to improve the overall database throughput.

Transactions originating at source site can be applied on destination sites either Synchronously or Asynchronously. The former provides high consistency and the later provides high throughput. In case of asynchronous mode, the data will be available at the destination sites after few seconds.

#### **2.1.4.2. Scalability**

CSQL supports clusters up to 16 updateable CSQL instances, each propagating updates either in synchronous and asynchronous modes to other sites. There is no limit on the total number of read only CSQL instances in the quorum.

## 3. Overview of Replication

This chapter provides you with an overview of how Replication replicates data.

### 3.1. How Replication Replicates Data

This section introduces the processes that Replication uses to replicate data. Before you can replicate data, you must define Replication. For more information on how to define replication, see *Chapter 6. “Administering and Monitoring Replication.”* Replication mainly uses three major processes to replicate data.

- Catch transactions
- Circulate data for replication
- Apply replicated data

#### 3.1.1. Catch Transactions

CSQL Replication uses a log-based transaction-capture mechanism to capture committed transactions. A log-based transaction capture minimizes the effect on transaction performance.

#### 3.1.2. Circulate Data for Replication

CSQL Replication uses the message queue to ensure that all the logs reach the appropriate server, regardless of a network or system failure. In the event of a failure, the message queue stores the logs until the network or system is operational. The message queue replicates data efficiently with a minimum of data copying and data transfer.

#### 3.1.3. Apply Replicated Data

CSQL Replication uses a data-synchronization process to apply the replicated data to destination site. The destination database servers acknowledge receipt of a message when the message is safely stored. If the destination site fails to apply the log sent by the source then it could be because of connection failure or execution failure.

If it is the connection failure then the logs are sent after connection is reestablished, if it is execution failure then the entire transaction is written into the conflict resolution file.

Thus, this process ensures that transactions are applied at the destination site in the same order, as they were committed on the source database server.

## 4. Deployment Options

In this section, some of the significant models have been illustrated to keep you in touch with industry level deployments where recovery and fault-tolerance is the prime necessity.

### 4.1. Active-Standby Replication

In active –standby configuration, application performs database operations only on active instance. If the active CSQL instance goes down for any reason, then application shall access the standby CSQL instance for performing database operations. This mechanism is called as fail over. CSQL keeps data at standby instance same as active database instance so that application can fail over to standby instance in sub seconds.

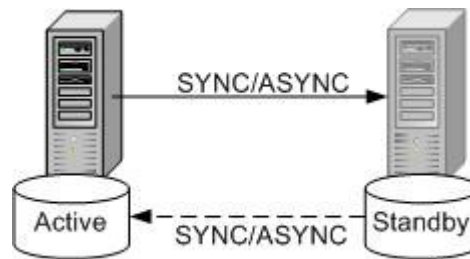


Fig 1-5. Active-Standby Replication

All the database operations such as INSERT, UPDATE, and DELETE on tables at active instance are applied transparently on standby database tables. CSQL can be configured either synchronously or asynchronously to apply these database operations on standby instance. For real time application, which require high throughput for modification operations, it makes sense to use asynchronous configuration. In asynchronous mode, execution returns immediately after it is performed at the active instance providing high throughput. Later CSQL replicator process makes sure that database operations at active instance are applied at standby instance without affecting the transactional behavior.

When the active sites resumes its operations, application can choose between either of the instances to make it active or standby based on its requirement.

### 4.2. Active-Active Replication

In active –active configuration, application performs database operations on both the active instances. If any CSQL instance goes down for any reason, then application accesses the other CSQL instance for performing database operations. CSQL keeps data at both active instances same to provide sub seconds fail over.

All the database operations such as INSERT, UPDATE, and DELETE on tables at CSQL instance1 are applied transparently on database tables of CSQL instance2.

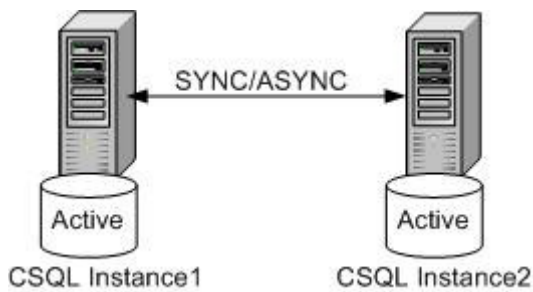


Fig 1-6. Active – Active Replication

If any CSQL instance goes down for any reason, the other instance preserves all the database operations that need to be applied (for updates which happen when the peer instance is down) and applies it when the instance which went down resume its operation.

If CSQL instances are connected using synchronous mode, when one instance goes down for some reason, it automatically changes to asynchronous configuration to preserve update logs for the other instance. Once this transition happens, application can reconnect and resume operations on it. The configuration can be changed to synchronous after the instance, which went down resumes its operation.

### 4.3. Load Balancing Cluster for Read

Most of the real time applications and high traffic web sites are read intensive. These applications shall distribute the query load to multiple instances to improve the throughput. In the above configuration, one instance is used for performing both read and write operations and other instances (CSQL Instance2 and CSQL Instance3) are used to distribute the load for read operations.

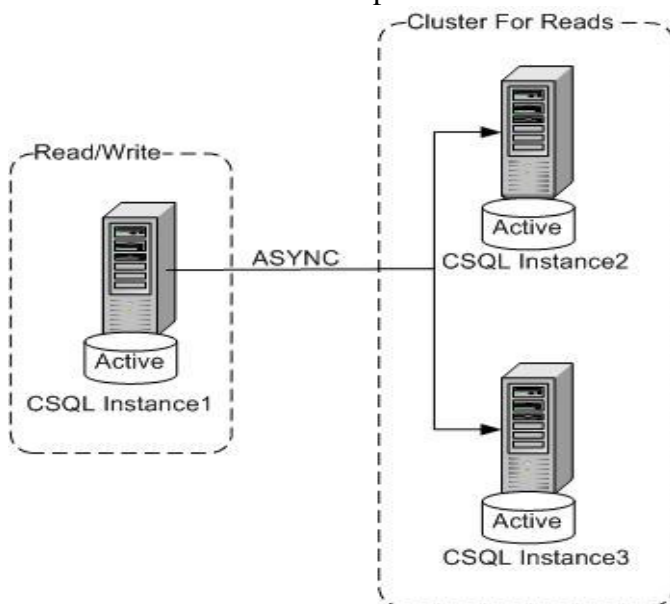


Fig 1-7. Load Balancing for Read

This deployment model is useful if your application has real time transaction requirement as well as near real time reporting requirements for different components. For example in SMS gateway application, the component which initiates the SMS sending process and stores the information in database requires real time performance and another component (reporting) which provides statistics and reports on the SMS application requires near real time performance. The former requires real time performance as it directly affects the total number of SMS sent / sec and the later require near real time performance where reporting web interface should display results within few seconds.

#### 4.4. Balancing Cluster for Read/Write

This deployment option suites application which has mixed read /write load where performance improvement is required for both read and read/write transactions. This is achieved by employing two clusters; one, which allows both the operations and another, which allows, only read operations. Read/Write cluster can be configured either in synchronous or asynchronous mode based on the application requirement for freshness of the data.

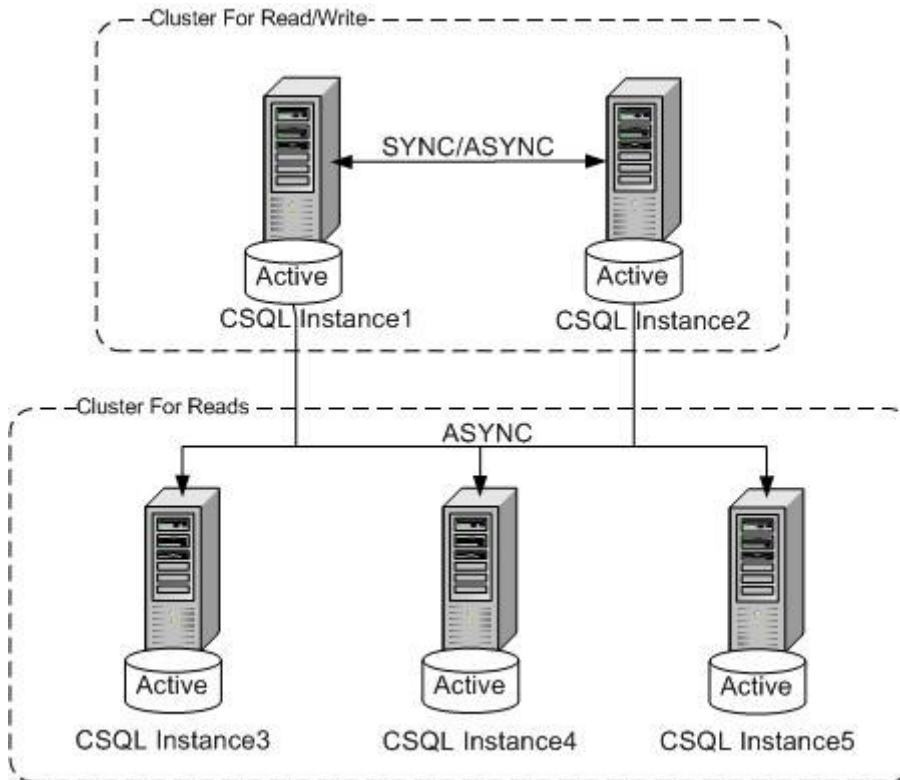


Fig 1-8. Balancing Cluster for Read/Write

This deployment option suites application which has mixed read /write load where performance improvement is required for both read and read/write transactions. This is achieved by employing two clusters; one, which allows both the operations and another, which allows, only read operations. Read/Write cluster can be configured either in synchronous or asynchronous mode based on the application requirement for freshness of the data.

For example online charging (pre-paid billing) application does read/write transaction to retrieve and update the credit balance during the call for every minute (based on the pulse rate). This requires real time performance as the charging is done online during the call itself. These types of applications can employ the above deployment model for increasing their throughput

#### 4.5. Load Balancing Cluster with Distributor

This deployment model is well suited for application, which are Write intensive. Transaction logs when propagated to other sites, (CSQL Instance3 to CSQL Instance5) in quorum takes some CPU time, which may affect the throughput of the CSQL instance1 when there are large number of destination instances in the “Read cluster”.

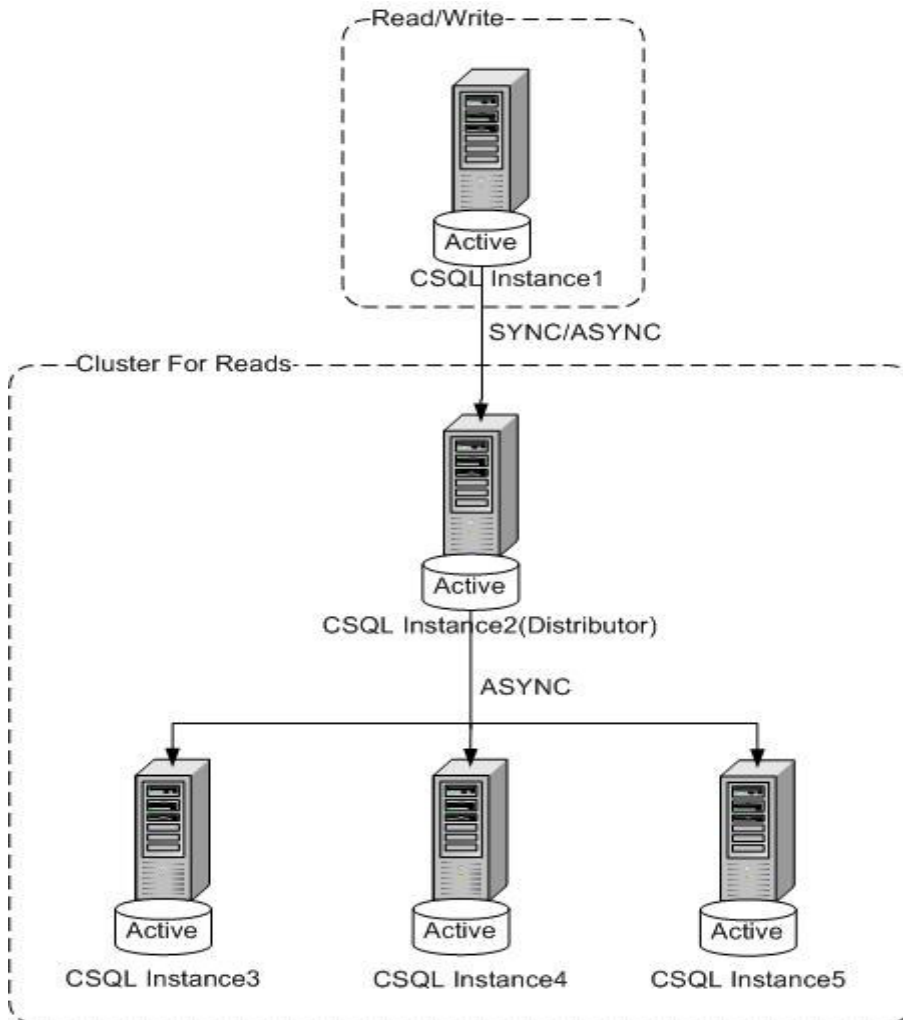


Fig 1-9. Load Balancing Cluster with Distributor

To overcome this and provide high throughput for application accessing CSQL Instance1, the application can use another instance (called Distributor), CSQL Instance2 to propagate transaction to all the sites in the “Read Cluster” on behalf of the source site. Multiple distributors can also be employed to scale the number of CSQL instances in the “Read cluster”.

#### 4.6. Load Balancing Cluster with Distributed data

This deployment model is well suited for application, which are distributed in nature but requires centralized management like banks and Visitor location Register (Telecom IN application). Instead of replicate the complete centralized database to all instances, application shall replicate data which belong to that particular zone or region it serves.

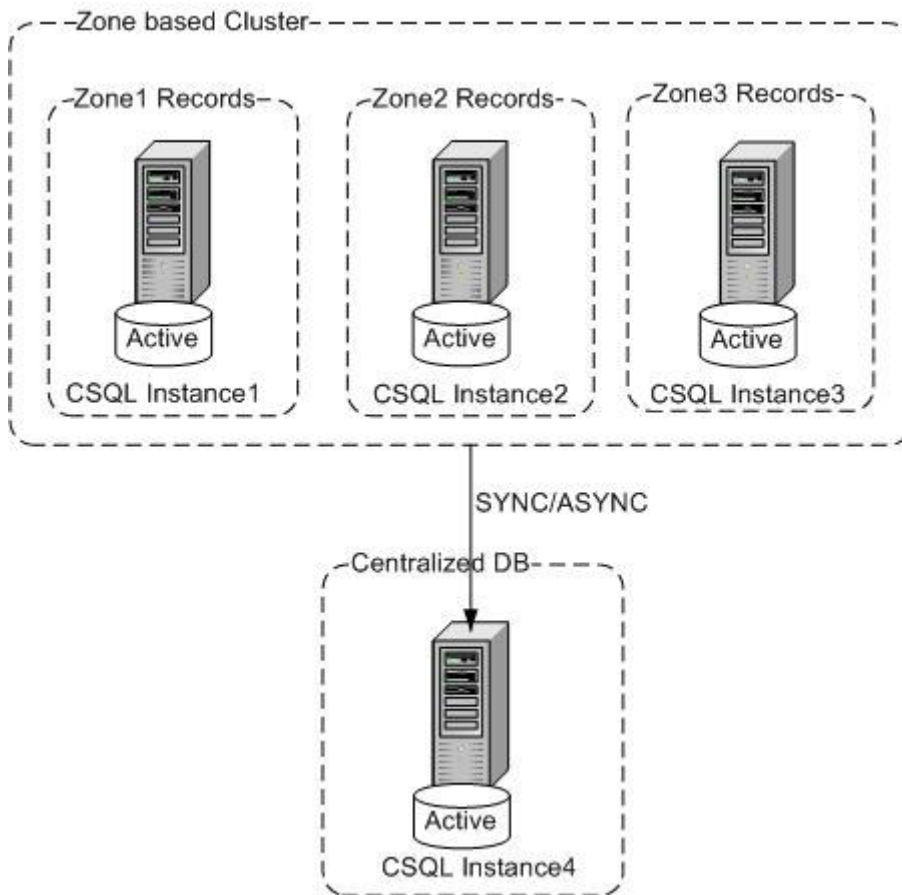


Fig 1-10. Load Balancing Cluster with Distributed Data

Telecom service providers divide the entire region they service into multiple zones (For example, state-wise) and keep records pertaining to that zone in the local data center rather than the remote centralized data center. This improves the performance of the application by multifold as it does not go over WAN for each database access.

In the above diagram, CSQL Instance1 stores only records for zone1 and replicates all the transactions that affect zone1 records to centralized database either in synchronous and

asynchronous mode. CSQL Instance2 stores only records for zone2 and replicates all the transactions that affect zone2 records to centralized database. The centralized database contains records up to date data for all the zones. This makes data management lot easier.

DBMS available in the market provide poor support for this configuration and puts more burdens on the application to replicate and manage data.

## 5. Setting Up Your Replication Environment

This chapter discusses the various operational restrictions, capacity requirements, and system changes you need to make based on the Replication system you select.

### 5.1. Operational Restrictions

Replication imposes the following operational Restrictions:

- Replication is supported between CSQL instances
- A CSQL server instance can participate in only one quorum.

### 5.2. SQL Statement Use

Once you have defined a table to Replication, you cannot execute any operation that changes the table and its schema definitions. You cannot use below SQL statement against the table.

- DROP TABLE

The following additional restrictions also apply to tables defined for replication:

- Do not remove the primary key constraint.
- Do not add, or drop columns.
- Do not modify the primary key column.

The following SQL statements are permitted:

- ADD INDEX
- DROP INDEX

### 5.3. Configuration Parameter

The database server configuration file (`csql.conf`) under CSQL root directory includes eleven configuration parameters that affect the behavior of Replication.

- SITE\_ID
- CSQL\_SQL\_SERVER
- PORT
- REPLICATION
- REPLICATION\_SITES
- NETWORK\_CONFIG\_FILE
- CONFL\_RESOL\_FILE
- MSG\_KEY

- ID\_SHM\_KEY
- ASYNC\_MSGMAX
- MAX\_QUEUE\_LOGS

The above parameters could be divided in two parts one for Replication and another for Network.

**Replication Configuration parameters:**

- SITE\_ID
- REPLICATION
- REPLICATION\_SITES
- NETWORK\_CONFIG\_FILE
- CONFL\_RESOL\_FILE
- MSG\_KEY
- ID\_SHM\_KEY
- ASYNC\_MSGMAX
- MAX\_QUEUE\_LOGS

**Network Configuration Parameters:**

- CSQL\_SQL\_SERVER
- PORT

CSQL requires some system parameters that need to be set before starting CSQL server. These configuration variables are defined in a file called `csql.conf`. Some of the parameters mentioned in this file may have to be tweaked based on the requirements.

The lines starting with `#` are ignored as comments and the rest are treated as configuration variables. These variable values are read from this file during server start up.

These configuration variables are divided logically into following classes.

- Server section variables
- Client section variables
- Cache section variables
- Replication section variables

It is very important to note that for Server section parameters, the value should be the same for the server process and all the CSQL client processes, which connects to it. Otherwise, behavior is undefined. For Server and Client section variables and additional

information about configuration parameters refer to the, [Section 3.5.2.2 in CSQL User's Manual](#).

### 5.3.1. Setting up parameters in csq1.conf file

The configuration file, csq1.conf has eleven parameters that need to be set up for replication. Some of the parameters are present in Cache Section and most are present in SQL Network Server Section in csq1.conf file.

- **Set the REPLICATION parameter to true to enable replication**

```
[ REPLICATION = true ]
```

This is a Boolean parameter and should be set to true to enable replication across site.

- **Set the SITE\_ID parameter respect to the database server**

```
[ SITE_ID=1 ]
```

This is an integer parameter and it identifies a CSQL instance in a quorum. Set this to a unique value for all the nodes in the quorum. In example, this parameter will be set to 1 for CSQL Instance-1 and will be set to 2 for CSQL Instance-2 and so on.

- **Set the CSQL\_SQL\_SERVER configuration parameter to true**

```
[ CACHE_SQL_SERVER=true ]
```

This is a Boolean parameter and should be set to true when replication is turned on.

- **Set the PORT parameter for network access**

```
[ PORT = 5678 ]
```

This is an integer parameter to set the port for network access, by default it is set to the number 5678.

- **REPLICATION\_SITES according to the sites go long**

```
[ REPLICATION_SITES = 16 ]
```

This is an integer parameter to specify the maximum no of sites present in the quorum. Default value is 16.

- **NETWORK\_CONFIG\_FILE parameter to '/tmp/csq1/csqlnw.conf'**

```
[ NETWORK_CONFIG_FILE=/tmp/csql/csqlnw.conf ]
```

This is a string parameter, which contains the complete path to the file, which holds the complete information about the peer site information in the quorum.

- **CONFL\_RESO\_FILE parameter**

```
[ CONFL_RESO_FILE=/tmp/csql/conflResoFile.txt ]
```

This is a string parameter, which contains the complete path to the file and also holds the complete information about the transactional conflicts that have been generated because of execution failures at the destination site in case of asynchronous updates.

- **MSG\_KEY and ID\_SHM\_KEY configuration parameter**

```
[ MSG_KEY = 2525 ]  
[ ID_SHM_KEY = 1947 ]
```

These are integer parameters and are used for creating message queue for MsgQueue server. It is used when your replication mode is Async only. This is an internal parameter and user need not modify this value.

- **ASYNC\_MSGMAX parameter**

```
[ ASYNC_MSGMAX = 8192 ]
```

This parameter should be set when asynchronous cache updates or asynchronous replication feature is required. By default this parameter is 8192 bytes(8kb). This corresponds to kernel parameter of message queue 'kernel.msgmax', which is the maximum size of a message in a message queue.

By default Linux has 8192 bytes for this parameter and it is suggested to verify for the same in the user environment. Privileged user can modify this parameter using sysctl tool as follows

```
#!/sbin/sysctl -w kernel.msgmax=<SIZE_TO_SET>
```

- **MAX\_QUEUE\_LOGS Parameter**

```
[ MAX_QUEUE_LOGS = 100 ]
```

This parameter is set for maximum number of messages that CSQL Message Queue hold it internally incase of peer site goes down. If this queue gets full,

then all updates for down site are removed from the queue and it is expected to re subscribe all the replicated tables when it comes back.

After setting up the environment variables we can set our sites to replicate data from one site to another.

## 6. Administering and Monitoring Replication

This chapter describes how to administer and monitor CSQL Replication using replication tools and utilities.

These are the two tools, which are involved in replication process.

- csql
- repltable

This two simple deployment options are used for the demonstration purpose:

- Active - Standby
- Active – Active

In below examples, two deployment scenarios have been represented, first one is Active-Standby and the second one is Active-Active on two CSQL Instances for the sake of easiness. For each deployment scenario both synchronous and asynchronous mode is used one after another.

### 6.1. Active – Standby

This Active-Standby deployment means the Source site is always used as Active site and the Destination site is always used as Standby. The two sites are in two different machines and are connected through the network. If Source Site goes down application can connect to the Destination Site and continue the operation.

The one way replication mode is used for this deployment and they can be,

- Synchronous
- Asynchronous

The two instances of CSQL should work in two different machines, for that we have to configure the replication and set up the network file which contains the information about the peer site.

Based on this deployment, CSQL Instance-1 is considered as Source site or Active Site, CSQL Instance –2 is considered as Destination Site or Standby Site. The Replication configuration is made accordingly both in csql. conf file and csqlnw. conf file.

#### **Configure csql. conf file for CSQL Instance – 1**

The Replication configuration parameters in the csql. conf file should look like the below contents.

```
DURABILITY=ON
SITE_ID=1
```

```
CSQL_SQL_SERVER=true
PORT=5678
REPLICATION=true
REPLICATION_SITES=16
NETWORK_CONFIG_FILE=/tmp/csql/csqlnw.conf
CONFL_RESO_FILE=/tmp/csql/conflResoFile.txt
MSG_KEY=2525
ID_SHM_KEY=1947
ASYNC_MSGMAX=8192
MAX_QUEUE_LOGS=100
```

### **Configure csql.conf file for CSQL Instance – 2**

The Replication configuration parameters in the csql.conf file should look like the below contents.

```
DURABILITY=ON
SITE_ID=2
CSQL_SQL_SERVER=true
PORT=5678
REPLICATION=true
REPLICATION_SITES=16
NETWORK_CONFIG_FILE=/tmp/csql/csqlnw.conf
CONFL_RESO_FILE=/tmp/csql/conflResoFile.txt
MSG_KEY=2525
ID_SHM_KEY=1947
ASYNC_MSGMAX=8192
MAX_QUEUE_LOGS=100
```

In the above Configuration parameters, SITE\_ID is the only parameter which is changed. The next step is configuring csqlnw.conf file.

### **Configure csqlnw.conf file :**

The network configuration file should contain the information about both the CSQL Instances and it should contain the below specified information. The file path has to be mentioned in the NETWORK\_CONFIG\_FILE parameter in csql.conf file.

#### **File Format:**

```
ID : IP_address : Port : Repl_Mode
```

### 6.1.1. Synchronous Replication

In case of synchronous mode, source site transaction commits return only after the transaction is committed at source instance and other destination instances in the quorum.

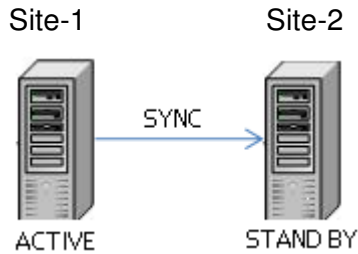


Fig 1-11. Synchronous Replication

#### Information in csqlnw.conf file at Instance-1

```
2 : 192.168.1.111 : 5678 : SYNC
```

The contents in csqlnw.conf file provide information about CSQL Instance-2, which is required by CSQL Instance-1 to connect and replicates data.

#### Steps to replicate table :

Follow the below steps to replicate data from Instance-1 to Instance-2.

Before begin with operations, Start the CSQL Server in two hosts with all necessary configuration parameters in csq1.conf file. Refer [CSQL User Manual to start the server](#).

#### **1. Start the CSQL Server at both the Instances.**

The below contents shows that all the required servers are running and it should matched at your both the Sites except Server's PID value.

```
$csqlserver
Sys_DB [Size=0001MB]
User_DB [Size=0010MB]
Network Server Started [PID=21640]
Replication Server Started [PID=21641]
Database Server Started...
```

## 2. Publish the table at Instance-1 using “repltable” tool

Lets assume that the “emp” table is present at the Instance-1 with some records. Use `repltable` tool to publish the table. *Refer the Section 8., for repltable too with all the options.*

```
$repltable -t emp
```

After executing this command entry will be made into the `csqtable.conf` file.

## 3. Subscribe the table at Instance-2

By executing the below command the table is replicated (Subscribed) at Instance-2 and the entry is made in `cachetable.conf` file.

```
$repltable -t emp
```

Now the table ‘t1’ is present in Source Site as well as in Destination Site. Insert some records at Instance-1 and it will be replicated to Instance-2.

## 6.1.2. Asynchronous Replication

In case of asynchronous mode, source site transaction commit return immediately after the transaction is committed at the source site. Replication process takes care of applying these transactions at destination sites in the quorum. This mode gives high throughput.

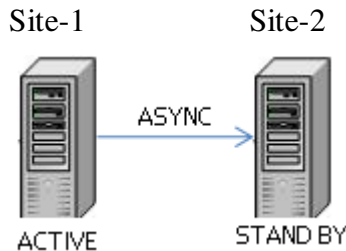


Fig 1-12. Asynchronous Replication.

To test this mode you can keep the above configurations, which have been made in `csql.conf` file both for Source Site and Destination Site. The change, which needs to be made that, is in `csqlnw.conf` file only.

### **Information in `csqlnw.conf` file at Instance-1 :**

```
2 : 192.168.1.138 : 5678 : ASYNC
```

The contents in `csqlnw.conf` file provide information about CSQL Instance-2, which is required by CSQL Instance-1 to connect and replicates data.

Follow the above section “Steps to Replicate table” to carry out replication.

For any reasons, if Active Site goes down, the application should connect to the Standby Site.

## 6.2. Active – Active

This Active-Active deployment means both the Source Site and Destination site is active and take part in replication. Application can access any of the sites at any point of time and the data should be consistent in both the database. The two sites are in two different machines and could connect through the network.

The two-way replication mode is used for this deployment and these are,

- Synchronous
- Asynchronous

For configuration in `csql.conf` file, follow the 6.1.Active-Standby deployment option for two CSQL Instances. The only change that needs to take part is in `csqlnw.conf` file only for both the Instances at both sites.

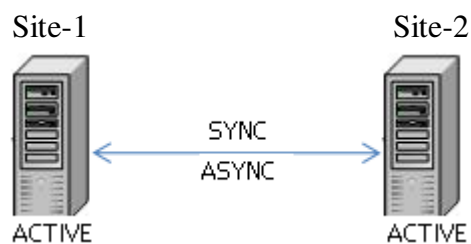


Fig 1-13. Replication in Sync/Async in Active-Active Deployment option

Follow the 6.1.Active-StandBy Section to configure `csql.conf` file, which will be same for this mode also. The `csqlnw.conf` file only needs to be changed according to the Sync or Async mode.

### 6.2.1. Synchronous Replication

In case of synchronous mode, source site transaction commits return only after the transaction is committed at source instance and other destination instances in the quorum.

#### **Information in the `csqlnw.conf` file at Instance-1 and Instance-2:**

```
1 : 192.168.1.111 : 5678 : SYNC
2 : 192.168.1.111 : 5678 : SYNC
```

The first line out of the above two in `csqlnw.conf` file provides information about CSQL Instance-1, which is required by CSQL Instance-2 to connect and replicates data. And the

second line provides the information about CSQL Instance-2, which is required by CSQL-Instance-1.

Follow the section “Steps to Replicate Table”, which have been defined in the Section 6.1.2.Synchronous Replication for Active-Stand By scenario to publish and subscribe table at both the site.

## 6.2.2. Asynchronous Replication

In case of asynchronous mode, source site transaction commit return immediately after the transaction is committed at the source site. Replication process takes care of applying these transactions at destination sites in the quorum.

### Information in the *csqInw.conf* file :

```
1 : 192.168.1.111 : 5678 : ASYNC
2 : 192.168.1.111 : 5678 : ASYNC
```

After setting up the ASYNC mode replication, user can publish the table at Source Site and subscribe the table at Destination Site. Follow the above mentioned section “Steps to Replicate Table” to do it.

In Active-Active deployment scenario, if any one of the site goes down then data will available in other site and application can connect to it and when it comes up it connects to its peer sites and back it its normal position.

**Note:** In case of Asynchronous active-active configuration, when updates happen on the same record in both the sites simultaneously, conflicts can arise or data at both sites may converge. Administrator should check for any conflicts and verify consistency of data and fix them manually.

## 7. Tools for Replication

The `repltable` tool is used to replicate table between sites. To do so, the source site needs to publish the table so that other sites can subscribe it.

### Example:

```
$repltable -t <TableName>
```

### Syntax:

```
repltable  
  
[-U username]  
[-P password]  
-t Table Name  
[-u ]  
[?]
```

#### **-U username**

This is a optional argument and its required for authentication.

#### **-P password**

This is a optional argument also required for authentication.

#### **-t tablename**

The specified table name will be replicated with this `-t` option.

#### **-u**

To keep the table unpublished and as a normal table.