



## **CSQL implementation with a Soft Switch (Asterisk) in a contact centre scenario**

**White-paper**

Version 1.0



**Disclaimer –**

Copyright @ LAKSHYA 2009

The information being shared in this whitepaper document is purely for technical and / or product reference. This document and any information enclosed within this document contains restricted and / or limited information and you must not disseminate, modify, copy/plagiarize or take any action in reliance upon it, unless permitted to by Lakshya Solutions Limited. None of the material in the document can be reproduced in any form.

LSL is in no way liable for any actions taken by the reader basing upon the contents of this document.

All rights reserved

All other brands, logos and products are trademarks or registered trademarks of their respective companies.

Compiled by Prabakaran T

Go to [www.csqldb.com](http://www.csqldb.com) for more information

Created on 16<sup>th</sup> October 2009

By Prabakaran T for LAKSHYA Solutions Ltd

India.

**Publishing History**

<b>Version number</b>	<b>Release Date</b>
1.0	16 <sup>th</sup> October 2009

## Table of Contents

Introduction .....	5
Typical Call Center.....	6
Typical IVR Applications .....	6
Why CSQL.....	6
Some key highlights.....	7
High Performance .....	7
Ease of use.....	7
Lower TCO .....	7
Deployment Modes .....	8
CSQL Stand Alone Database for Asterisk .....	8
Setting up CSQL for Real Time Queue.....	8
Creating Asterisk Queue Tables in CSQL.....	8
Asterisk Configuration.....	9
CSQL Main Memory Cache for MySQL, Postgres or Oracle DBMS.....	11

## Introduction

Contact / call center software which are based on the Open Source soft switch such as Asterisk use a dynamic real-time system to load objects that may change frequently such as SIP/IAX2 users and peers, queues and their members, from external database management systems. Had the same configurations been stored within the soft switch configuration files, then for every change in the configurations, the switch would require a restart for the changes to take effect, thus resulting in increased system downtimes, which severely disrupts processes on the floor and undermines performance statistics. Such system downtimes are the bane of call centers the world over and are to be avoided at any cost. Hence the facility to access external data sources through the “Real-time” library.

Such real time access to external data objects are also useful for up-scaling operations by clustering multiple soft-switches working with the same centralized information, or when you want to build external applications which modify information in the call flow and you want to do so without requiring a reload / re-start of the soft switch. This introduces a dependency on the performance of the database management system, as information required for the switch to handle the call flow is stored in an external source and has to be fetched in real time. This is where traditional disk resident database systems will never be able to deliver the desired real time performance levels, thereby slowing the performance of the switch resulting in an overall performance degrade of the call center and customer interaction quality and experience, which no call center can afford more so under the stringent customer facing telemarketing regulations.

CSQL is the fastest enterprise class open source main memory relational database management system. Traditional disk based database management systems are persistent but are incapable of dealing with dynamic data that change constantly and cannot cater to real time systems/applications that require responses in microseconds and not in milliseconds.

Telecom systems, such as soft-switches, that deal with real time data require faster response times and high throughput, were limiting themselves to generic databases foregoing incredible business benefits. With the speed of business increasing, and the volume of information/calls that applications must process growing as well, businesses in most industry domains are required to make the transition to real time data management, in order to stay competitive.

CSQL, by its inherent memory architecture manages database in memory, achieving dramatic gains in responsiveness and throughput, even compared to a fully cached disk resident database. This paper outlines the capabilities of CSQL and how it fits into the real time architecture and IVR applications based on Asterisk, as the soft-switch platform.

## Typical Call Center

Call centers work either in inbound or outbound mode. Inbound mode deals with receiving a call from a customer and then handling the call and resolving the issue of the customer. Outbound mode deals with the call center employee (also called agents) making a call to a customer in order to inform or sell a product or service.

Both inbound and outbound call centers; need to deal with queue management and call distribution algorithms. In the outbound scenario, for each campaign there will be one queue and agents get shuffled between campaigns based on the call load. In case of inbound, for each skill (For Example: handle credit card) there will be one queue and agents get shuffled between queues based on the load.

Queue is the virtual line in which customers (callers) wait for an agent to be available to take their call. One of the major concerns in call center is the wait-time of the customer, as no one enjoys waiting in the line.

Queue management and distribution management, being the core of a call center platform, should perform faster in order to reduce the wait-time of the customer and effectively utilize the agent time. Typically Asterisk queries the database for every incoming call to get queue information and associated agent status for managing and distributing calls as per the agent status. During periods of high call volumes, simultaneous queries to the database residing in another host, will significantly degrade response times, thereby resulting in poor management of call queues and longer wait times for the caller.

## Typical IVR Applications

IVR applications, which are typically designed for self-service, handle incoming calls automatically by interacting with database systems (known as database dips), TTS and ASR systems. During high call loads, data access response times degrade, thus resulting in higher wait times and leading to higher abandoned calls and customer dissatisfaction.

Asterisk, by default, provides MySQL Dial Plan Application plug in and built in func\_odbc module for interacting with the database. Now MySQL requires the connect/disconnect for every call resulting in lower throughputs, high network bandwidth usage and increased CPU usage. In this scenario, by deploying the lightweight ODBC compliant database CSQL, IVR applications can provide faster responses to callers even during peak loads.

## Why CSQL

Traditional databases such as Oracle, MySQL, Postgres, etc are client/server based and are disk resident, hence by using them to store queue information in a centralized database host, for example, which is connected to the asterisk machine (soft-switch) over a network will obviously deliver poor response times owing to network overhead and load on the database server.

By employing a lightweight (takes less CPU time, reduced disk I/O activity allowing asterisk to use disks efficiently) embedded (running in the switch (host) itself) main memory database management system such as CSQL, response times will drastically improve to few microseconds rather than few milliseconds or seconds without disrupting the asterisk processing at the same host. Hence CSQL empowers the soft-switch to provide a more dynamic contact center that delivers drastically improved quality of customer service, quickly and effortlessly.

This disruptive performance benefit delivered through a CSQL empowered call center switch is the need of the hour and will become the default requirement in call centers in the days to come.

### **Some key highlights**

#### **High Performance**

Modern applications are transaction-intensive, demanding the responsiveness of in-memory data caching to support increased transaction volumes with minimal latency. Wisconsin benchmarking of CSQL on commodity hardware delivers 30 times faster response times than a leading database management system.

#### **Ease of use**

CSQL comes as a pre-installed package (2 MB size) that follow download, extract and use strategy. It requires no DBA and completely eliminates performance tuning. Through its standard interface support such as ODBC and JDBC, reduces the learning curve for developers to use CSQL in their applications.

It provides a transparent pass-through mechanism, in case the application wishes to use the existing (or primary) database for normal queries and use CSQL only for frequently accessed queries.

#### **Lower TCO**

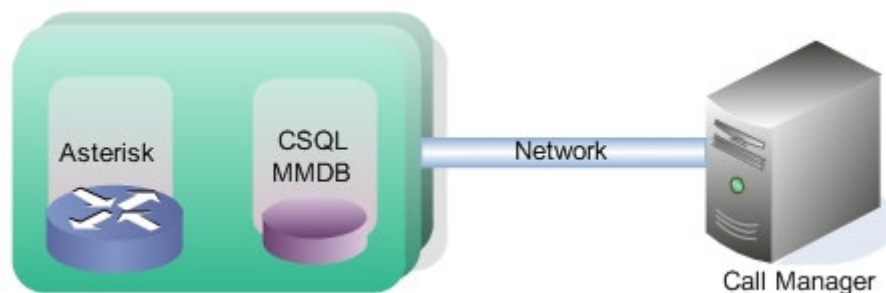
Organizations of all types and sizes are under pressure to boost efficiency and cut costs. Traditional data management solutions are expensive to maintain, manage, and scale. CSQL - a high performance main memory relational database management system was designed to lower database TCO. CSQL's innovative architecture helps organizations dramatically reduce both capital and operating expenses, while improving performance, data availability and recovery.

Closed-code vendors usually charge a single-perpetual license for each instance and additional annual maintenance and support. CSQL's pricing strategy is based on annual subscriptions that covers desired support levels and does not entail any initial perpetual l

icense fee or charges of any kind.

## Deployment Modes

### CSQL Stand Alone Database for Asterisk



In Stand-alone mode, CSQL Main memory database runs as an embedded database on the soft-switch machine. Durability mode of CSQL should be turned “on” (in the csql.conf – for detailed configurations please get in touch with us at [technical@csqldb.com](mailto:technical@csqldb.com)), to ensure durability of data after shutdown or crash. The Asterisk ODBC resource should be configured to use CSQL ODBC driver through unixODBC driver Manager.

For More information on setting up unixODBC for CSQL, refer

<http://csqocache.wordpress.com/2009/10/12/configuring-csql-for-unixodbc/>

Queue information shall be updated using either JDBC / ODBC application residing in some other host connected with network. These updates will be available to Asterisk instantaneously.

### Setting up CSQL for Real Time Queue

Refer below blog for installing and configuring CSQL for odbc access

<http://csqocache.wordpress.com/2009/10/12/configuring-csql-for-unixodbc/>

### Creating Asterisk Queue Tables in CSQL

Run the below stated SQL statements using csql tool or isql tool to create the necessary tables

```
CREATE TABLE ast_queues (name CHAR (128) NOT NULL ,
musiconhold CHAR (128), announce CHAR (128), context CHAR
(128), timeout INT , monitor_join TINYINT , monitor_format
CHAR (128), queue_youarenext CHAR (128), queue_thereare
CHAR (128), queue_callswaiting CHAR (128), queue_holdtime
CHAR (128), queue_minutes CHAR (128), queue_seconds CHAR
(128), queue_lessthan CHAR (128), queue_thankyou CHAR
(128), queue_reporthold CHAR (128), announce_frequency INT
```

```
, announce_round_seconds INT , announce_holdtime CHAR
(128), retry INT , wrapuptime INT , maxlen INT ,
servicelevel INT , strategy CHAR (128), joinempty CHAR
(128), leavewhenempty CHAR (128), eventmemberstatus TINYINT
, eventwhencalled TINYINT , reportholdtime TINYINT ,
memberdelay INT , weight INT , timeoutrestart TINYINT ,
ringinuse TINYINT , setinterfacevar TINYINT );
```

```
CREATE INDEX ast_queues_idx1_Primary on ast_queues ( name )
HASH UNIQUE;
```

```
CREATE TABLE ast_queue_member (uniqueid INT NOT NULL
AUTO_INCREMENT , membername CHAR (40), queue_name CHAR
(128), interface CHAR (128), penalty INT , paused INT );
CREATE INDEX ast_queue_member_idx on ast_queue_member (
queue_name ) HASH ;
```

```
INSERT INTO ast_queues VALUES( 'csqltest_queue', 'default',
NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL, NULL, NULL, NULL, NULL, NULL, NULL, 1, NULL,
NULL, 'leastrecent', 'yes', NULL, NULL, NULL, NULL, NULL,
NULL, NULL, 0, NULL);
```

```
INSERT INTO ast_queue_member VALUES(1, '<agent name>',
'csqltest_queue', 'SIP/<Agent name>', 1, 0);
```

**Note:** replace “agent name” with the name of the agent.

## Asterisk Configuration

This section describes all the configuration changes required for Asterisk to work with CSQL and to retrieve or update data in CSQL main memory database or cache.

### Add following lines to “extconfig.conf”

```
queues => odbc, asteriskcsql, ast_queues
queue_members => odbc, asteriskcsql, ast_queue_member
```

### Add following lines to “res\_odbc.conf”

```
[asteriskcsql]
enabled => yes
dsn => mycsql
username => root
password => manager
pre-connect => yes
pooling => yes
```

**Add following lines to “sip.conf”**

```
[csqluser1]
type = peer
host = dynamic
dtmfmode = rfc2833
username = csqluser1
secret = csql123
qualify = yes
canreinvite = no
reinvite = no
callerid = csqluser1
context = mycontext
insecure = no
[csqluser2]
type = peer
host = dynamic
dtmfmode = rfc2833
username = csqluser2
secret = csql123
qualify = yes
canreinvite = no
reinvite = no
callerid = csqluser2
context = mycontext
insecure = no
```

**Add following lines in “extensions.conf” for “mycontext”**

```
exten = 222,1,Queue(csqltest_queue)
exten = 222,n,Hangup
```

Register “csqluser1” and “csqluser2” sip users and call extension “222” from “csqluser2”. Check asterisk log if there is any error.

**Related Information**

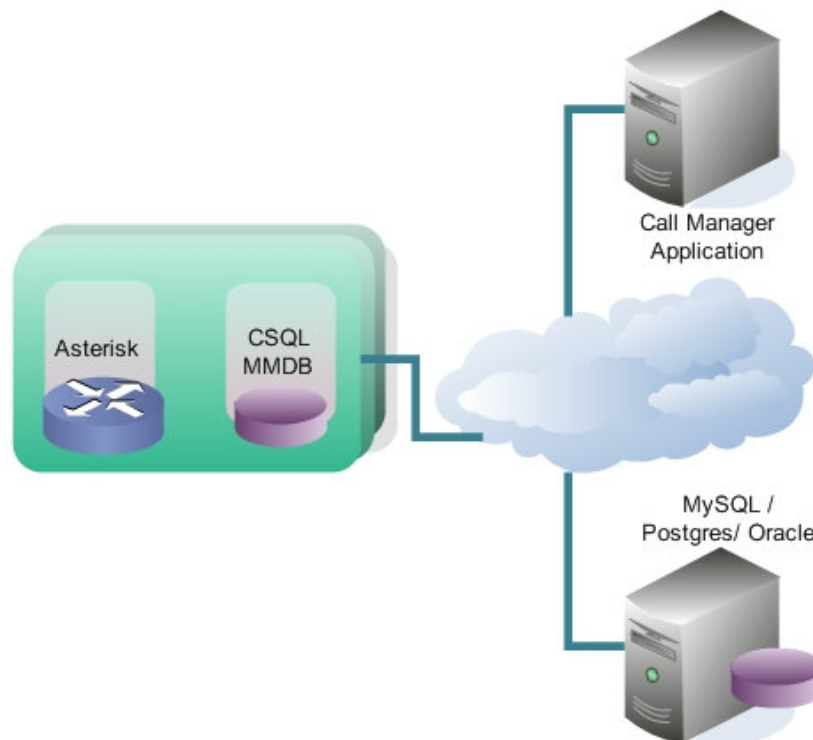
Refer below blog `func_odbc`:

[http://csqlcache.wordpress.com/2009/10/14/using-csql-for-asterisk-func\\_odbc/](http://csqlcache.wordpress.com/2009/10/14/using-csql-for-asterisk-func_odbc/)

SIP Users:

<http://csqlcache.wordpress.com/2009/10/15/storing-asterisk-real-time-sip-users-in-csql-main-memory-database/>

### CSQL Main Memory Cache for MySQL, Postgres or Oracle DBMS



In this deployment mode, existing database from MySQL / Postgres / Oracle (termed as target database in this document) is cached in CSQL Main memory database to provide real time access. Updates on CSQL Cache are automatically propagated to target database and vice versa. Service Provisioning (such as adding new queue, adding members, etc), can be done by the Call Manager Application on the target database. CSQL Bidirectional caching, ensures that modification on target database automatically reflects in the cache and are available for asterisk.

Refer Bi-directional caching section of CSQL Cache Guide to setup bi-directional caching for any target database. <http://www.csqldb.com/prod-Docummentation.html>

LAKSHYA Solutions Limited, 1st Floor, #73&74 Margosa Road, ABMGP Building, 17th Cross, Malleshwaram, Bangalore – 560055, India.

The information being shared in this document is purely for technical and / or product reference. This document and any information enclosed within this document contains restricted and / or limited information and you must not disseminate, modify, copy/plagiarize or take any action in reliance upon it, unless permitted to by Lakshya Solutions Limited. None of the material in the document can be reproduced in any form.