

---

## GUIDE TO CHOOSE BEST CACHING STRATEGY FOR YOUR APPLICATION

Introduction .....	2
Benefits .....	3
Performance.....	3
Consistency.....	3
Ease of use.....	3
Scalability.....	3
Stability .....	3
Reduces development effort.....	3
Feature Summary .....	4
Cache Granularity.....	4
Bi-directional Transaction Propagation .....	4
Transaction Propagation Modes.....	4
High Availability.....	5
Load Balancing .....	5
Recovery .....	5
Data Consistency.....	5
Platform Supported.....	6
Deployment Options.....	6
Active Result Set Cache.....	6
Transparent Caching with Gateway .....	7
Read Intensive Cache.....	7
Read/Write Intensive Cache .....	8
Write Intensive Cache.....	8
Multi Node Cache .....	9
Coherent Multi Node Cache.....	11
Conclusion .....	<b>Error! Bookmark not defined.</b>

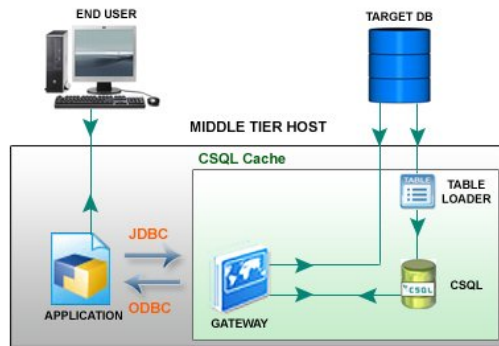
## Introduction

With the speed of business increasing, and the volume of information that enterprises must process growing as well, businesses in many industry domains transition to real time data management in order to stay competitive.

Though there is huge demand for speed, enterprises are reluctant to migrate their applications, as they do not want to give up the existing database systems they are using for many years that are proven stable in their environment.

Web and real time applications are mostly read intensive though some are read/write intensive. By employing caching for frequently accessed tables at the application tier, it shall reduce load on the backend databases and reduce network calls, resulting in very high throughput.

CSQL Cache is generic database caching platform to cache frequently accessed tables from your existing open source or commercial database management system (Oracle, MySQL, etc) close to application tier. It uses the fastest Main Memory Database (CSQL MMDB) designed for high performance and high volume data computing to cache the table and enables real time applications to provide faster and predictive response time with high throughput.



CSQL Caching uses transaction log based replication scheme using TCP/IP, to deliver high performance without compromising on the consistency of cached data and the original table in the existing database. CSQL provides application transparent caching using gateway module that takes care of providing unified interface to both cached and non-cached tables.

CSQL supports two transaction propagation modes for updateable cache tables, synchronous and asynchronous mode.

In case of synchronous mode, transaction commit return only after the transaction is committed at cache instance and target database, whereas in case of asynchronous mode, source site transaction commit return immediately after the transaction is committed at the cache. Caching process takes care of applying these transactions at target database.

CSQL Cache provides synchronous and asynchronous transaction propagation modes enabling applications to choose between high throughput and high consistency.

## Benefits

### Performance

CSQL Cache delivers **100 times faster** response time for cached tables as it uses fastest MMDB for caching and there is no network overhead involved for accessing cached tables.

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

### Consistency

Tables cached using CSQL are by default updateable. Transactions, which modify cache tables, are applied at the target database so as to keep the cache consistent with the target database. CSQL also supports bi-directional caching in which any direct updates on the target databases are automatically propagated to cache.

### Ease of use

CSQL Cache is simple to use and requires minimal setup to get the data cached. Requires no changes to application and minimal administration as CSQL has inbuilt self-healing mechanisms to tolerate faults.

Applications shall access cached tables at CSQL MMDB as well as non-cached tables at target database, which makes sure that no code changes are required in application (Checking whether the table is cached or not and divert queries to existing database or CSQL MMDB cache based on that).

### Scalability

Leverages multi core and processor architecture machines as it uses highly concurrent CSQL MMDB. For more information refer CSQL MMDB data sheet.

CSQL supports distributed caching in which multiple cache nodes shall be employed for load balancing, still preserving the cache coherence in all cache instances and target database. There is no limit on the total number of cache instances in the quorum.

### Stability

Enterprises do not take the risk of trying another database management system. CSQL works in conjunction with existing database and improves performance of 'hot' tables.

### Reduces development effort

Through its standard interface support (ODBC and JDBC), it reduces the learning curve for developers to adopt CSQL in their applications. It requires no DBA and completely eliminates performance tuning.

By its inherent transparent caching mechanism, applications shall access cached tables at CSQL MMDB as well as non-cached tables at target database, which makes sure that

no code changes are required in application. This reduces huge development and testing effort.

## Feature Summary

### Cache Granularity

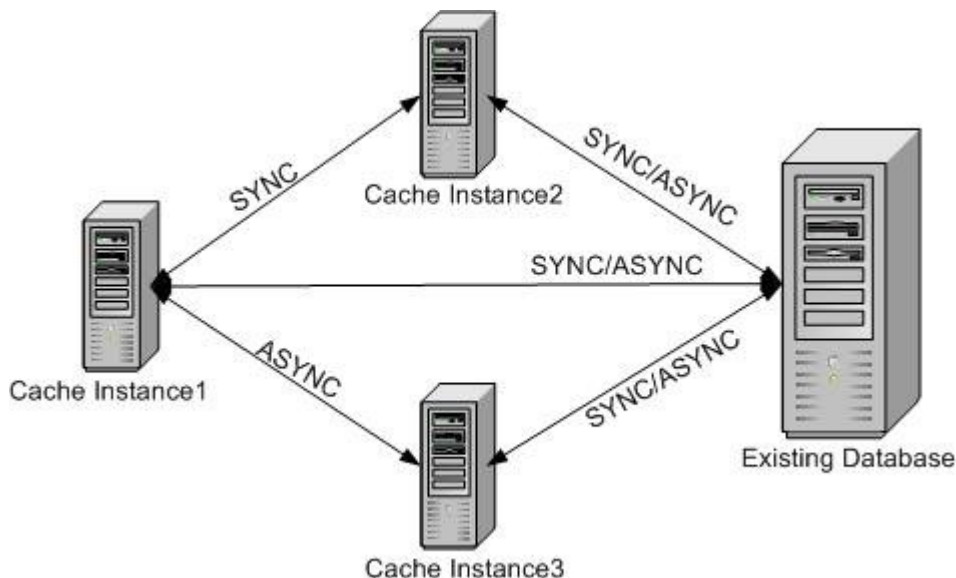
CSQL supports updateable caching at multiple granular levels. Table level caching, caches all the records from the original table whereas partial table caching allows applications to choose what fields and what records needs to be cached. This is useful in cases where the original table is too big that it cannot be fit in the available memory.

### Bi-directional Transaction Propagation

When there are direct updates on the target database (application updates the original table in target database), CSQL cache server automatically receives updates from target database and applies them on cached table periodically to keep the cache consistent with the original table.

### Transaction Propagation Modes

Transactions that affect cached tables are applied at original tables in target database in two modes namely, Synchronous and Asynchronous. In case of multi node caching, these transactions are also applied in all the other nodes, which cache that table. The mode between two cache instances need not be same. For example for two cache instances (instance1 and instance2), instance1 can propagate transactions to instance2 in synchronous mode and instance2 can propagate transactions to instance1 in asynchronous mode or not to propagate transactions to instance1.



In case of multi node caching, application can choose the transaction propagation mode between each and every instance in the quorum. Lets say you have three instances namely instance1, instance2 and instance3 in the quorum. Instance1 and Instance2 can be connected using synchronous mode and instance1 and instance3 can be connected using asynchronous mode. Again the mode at which these three instances propagate transactions to target database could either synchronous or asynchronous. This gives flexibility to applications to choose the right configuration based on their throughput and data consistency requirements.

## High Availability

CSQL provides transparent sub-second fail over to target database in case of cache instance failure. During this time application will see reduction in throughput, as all transactions will be directly handled by the target database. Once the cache instance comes back, applications regain high throughput.

## Load Balancing

CSQL supports distributed caching in which multiple cache nodes shall be employed for load balancing, still preserving the cache coherence in all cache instances and target database. This improves throughput of the application significantly.

## Recovery

When a cache go down for any reason (planned maintenance, crash, or disaster), all the other instances in the cluster including target database, retains all the transactions at their local instance till the instance which went down resumes its operation. When the cache instance comes back, it receives transaction updates from all the other instances (transactions which happened when this instance was down) and automatically gets synchronized with all the other instances in quorum. This recovery is automatic and does not require any manual intervention.

As CSQL maintains redo log records and checkpoint for cached tables, recovery becomes faster as it retrieves only the incremental changes from the target database and other instances in the cluster.

## Data Consistency

CSQL employs transaction consistent caching, if transaction fails at target database connected using synchronous propagation mode fails, the transactions fails at the cache also. This ensures that two parallel conflicting transactions (For example deducting balance from account) running in cache and original table to fail and ensure consistency of data between cache and the original table.

For asynchronous mode propagation, transaction conflicts are detected and logged into file so that the administrator can resolve conflicts. Conflict resolution could be additive, subtractive or replacing 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 Cache provides tools to check for consistency between cache and original table connected in asynchronous mode to assist application for ensuring and maintaining data consistency between cache and original table.

## Interface support

CSQL cache provides JDBC and ODBC driver. Existing applications require changes to DSN name in case of ODBC driver and database URL in case of JDBC to adapt caching. It also provides PHP-ODBC interface for applications written in PHP.

## Supported Database Platforms

CSQL supports any database management system, which provides standard ODBC driver. CSQL caching is tested with Oracle, MySQL and Postgres. Testing for other DBMS are underway based on the priorities. If you are using any other database, and want to cache tables with CSQL Cache, send us a mail to [technical@csqldb.com](mailto:technical@csqldb.com) to get it tested and certified at priority.

## Platform Supported

Linux – x86

Linux – x86\_64 and Solaris are under development

## Deployment Options

### Active Result Set Cache

In result set based caching strategy adopted by applications, one of the major drawbacks is changes to the source table will not be reflected in the cached result set. This limits the strategy to be used only for tables, which are never updated. In CSQL active result set caching mode, applications are allowed to update result set either directly in CSQL MMDB or on the target database in which case it is automatically pushed to cached table at CSQL MMDB. Another advantage of CSQL Cache is that it also allows querying on the cached result set (as it is stored in MMDB) using SQL queries.

### ACTIVE RESULTSET CACHING

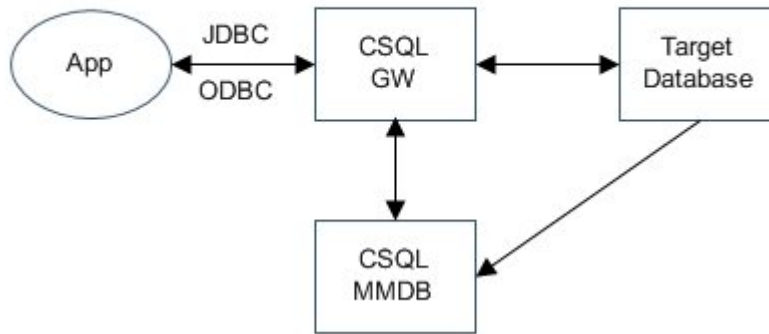


In this architecture, there will be one CSQL MMDB and target database. The application talks to CSQL cache through the standard JDBC or ODBC interface. The tables to be cached say “customer\_info” (from telephone billing database) is loaded from the target database into CSQL MMDB. If this table is updated in the target database by some other backend application, the updated/inserted record is automatically pushed to CSQL cache by employing bi-directional mode. This mode is suited for developing new applications and requires code changes for existing applications.

## Transparent Caching with Gateway

In this model, CSQL gateway module provides a unified interface to applications through the standard JDBC or ODBC interface. Existing application just changes the DSN name for ODBC application or connection URL for JDBC applications to employ transparent caching.

### GATEWAY CACHE



Frequently accessed tables are cached from target database into CSQL MMDB and gateway component enables access/modifications to cached tables in CSQL MMDB and non-cached tables, which reside only in the target database.

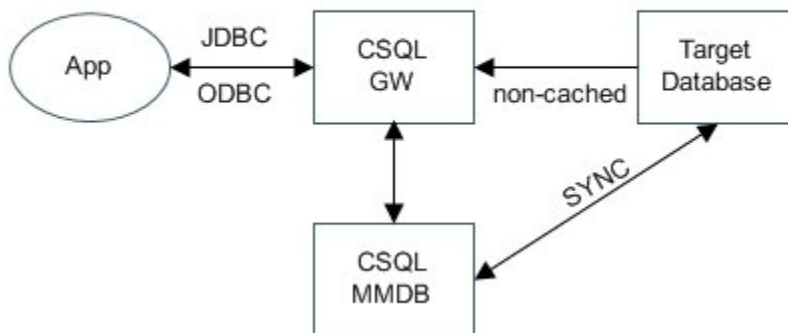
If the application sends a select query, requesting some records from the table "customer\_info", the gateway checks whether the table resides in the CSQL cache. If it is so, it returns records from CSQL. If the table is not found in the cache, the gateway returns the records from the target database. If our application inserts/updates any record in the cached table, that operation is effected in both cache and the target database by the gateway.

This gateway cache can be configured in three ways. They are read intensive cache, read/write intensive cache and write intensive cache.

### Read Intensive Cache

This deployment mode is useful when the reads are more than writes on cached tables. In flight booking management system, flight schedules are read-only and are not updated frequently. They shall be cached in CSQL cache, so that applications accessing these timetables will get ultra-fast response.

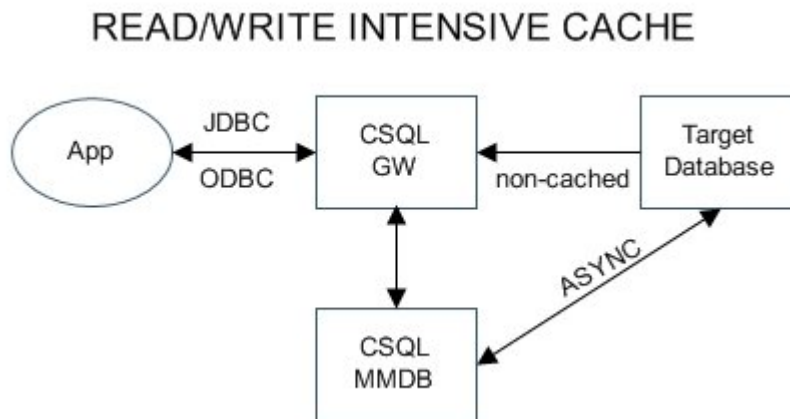
### READ INTENSIVE CACHE



Application connected to gateway fetches records from either CSQL MMDB (in case of cached table) or from target database (in case of non-cached table). If application does updates (which happens very rarely as the application is read intensive) in the cached table, CSQL MMDB will do the same operation on target database before the commit operation returns. So in this architecture, reads are ultra fast and writes are comparatively slow as it involves updating both cache and target database before commit returns.

## Read/Write Intensive Cache

This deployment is suitable for applications, which have equal probability for read and write operations to database.



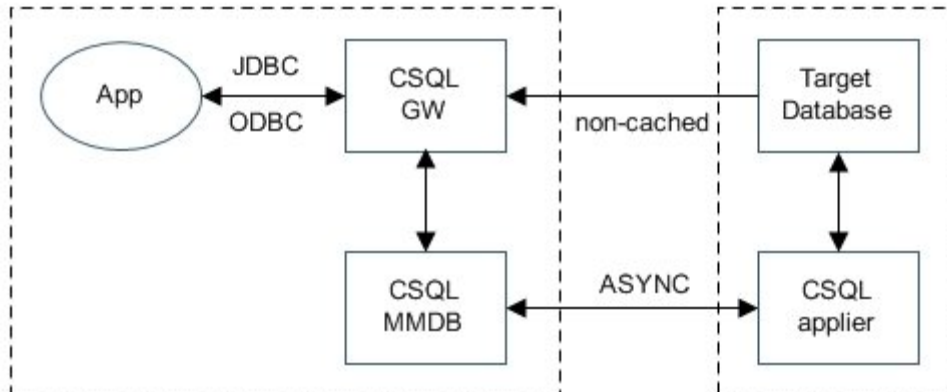
Here everything is same as the read intensive cache except the update propagation mode between CSQL MMDB and target database. If a transaction tries to update the cached table, CSQL MMDB carries out the operation in its cached table and then just informs another internal process (applier) to perform the same operation in the target database. Commit operation returns immediately after updating the cache, thereby delivering high throughput for modification operations.

## Write Intensive Cache

CSQL MMDB performs write operations faster than disk-based databases. If you have performance bottleneck on write operations, then applications can use this deployment option to improve throughput for write intensive transactions.

# White Paper

## WRITE INTENSIVE CACHE

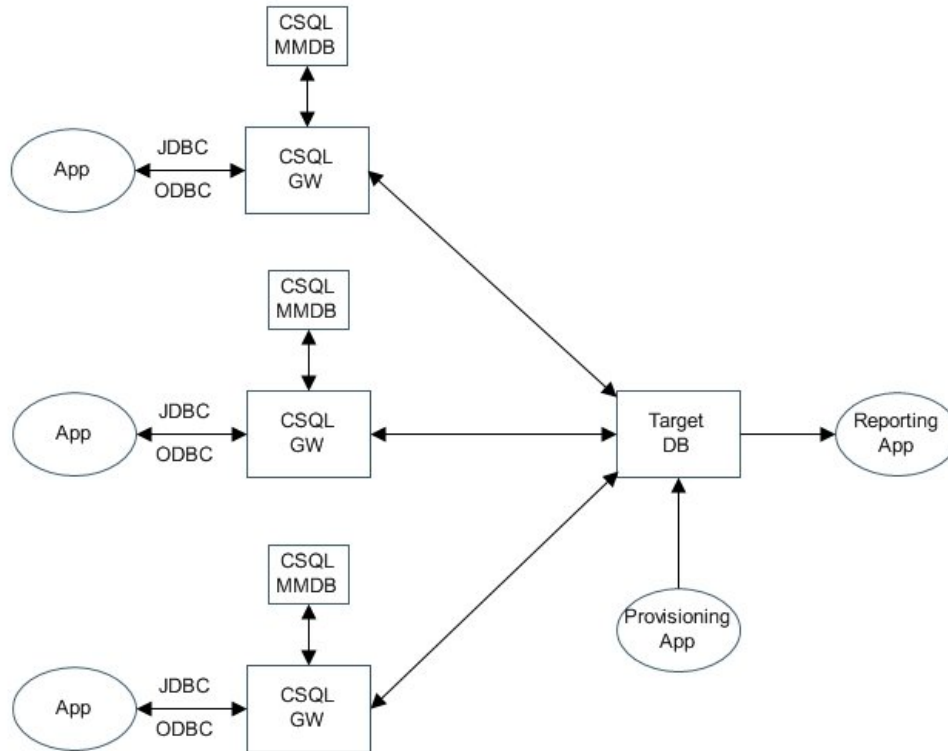


Here the application, Gateway and CSQL MMDB lies in one machine and another process 'applicier' runs in the host where target database is running. Updates on cached tables are propagated to another applicier process, which runs in target database host and that process takes care of executing the operations on behalf of CSQL MMDB. This reduces the computing cycles on the host where application runs, as it does not perform execution at the host where application resides.

### Multi Node Cache

Applications can scale up by employing multiple cache instances to cache tables. This architecture consists of many CSQL gateways, each having its own CSQL cache. All the gateways are connected to a single target database. Applications shall be connected to any one of the gateways through the standard JDBC/ODBC interface.

## VERTICAL MULTINODE CACHE



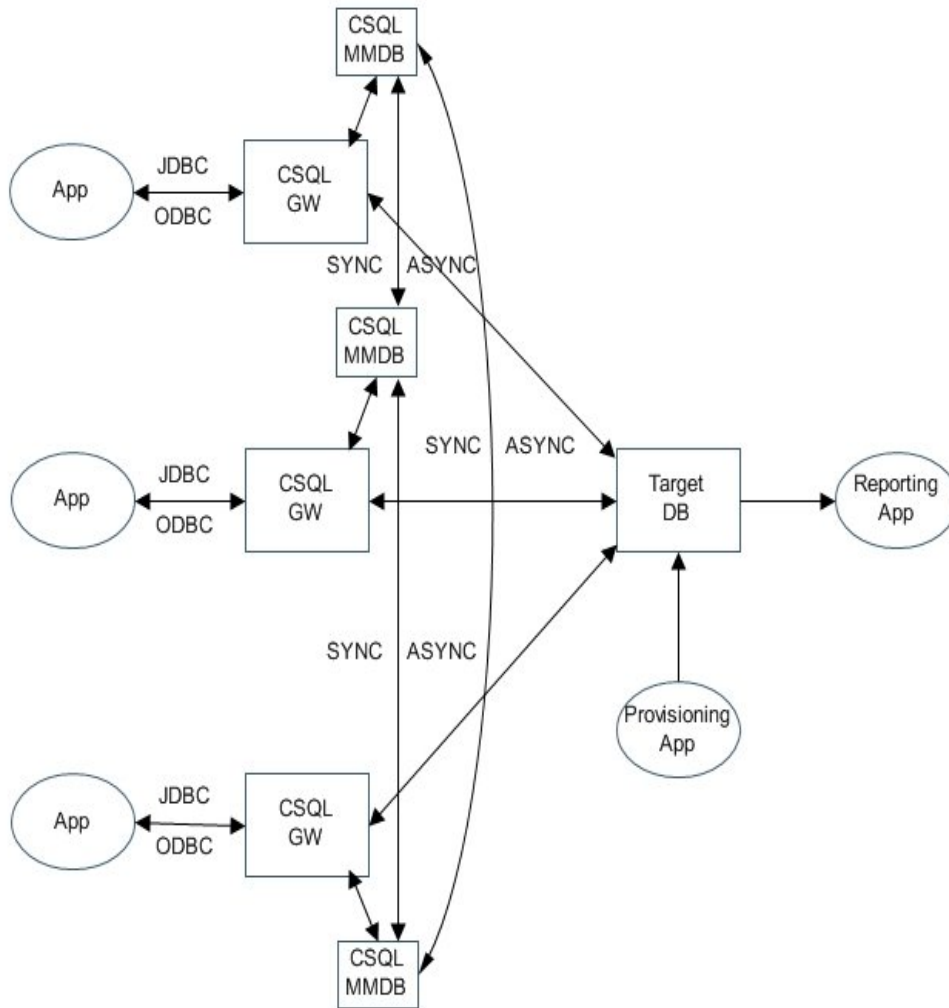
In this configuration, each CSQL MMDB shall cache complete table at all nodes or a subset of records in each node. For example, for caching 'customer\_info' table using this architecture, all the zone 1 customer details will be cached in the first CSQL cache, the second one caches the zone 2 customer details and the last one caches the zone 3 customer details. In this way, we can add any number of nodes, thus improving the performance of the entire system. All the gateways are then connected to a centralized server in which the original 'customer\_info' table resides. In case of telecom domain, each zone (Bangalore, Chennai, Delhi, Mumbai etc., ) contains CSQL cache with only their zone customer details and all these servers are connected to a centralized database server(target database) which contains information about all the customers in India.

By enabling bi-directional caching at all cache nodes, updates on any of the cache instances will be propagated to other cache nodes automatically. In the above deployment scenario, provisioning application and the reporting application talk directly to the target database. The provisioning application inserts/updates the original table in target database, which is then pushed into the corresponding CSQL MMDB only. For example, new customer added to Chennai zone will be available only in the Chennai Zone CSQL Instance.

## Coherent Multi Node Cache

This deployment option is useful for application with stringent cache consistency requirements. When update operation is performed on one cache node, it should be reflected in all the other cache nodes as well before it returns to the application. This can be used for banking applications to cache account table for accessing balance amount before performing any withdrawal. When the withdrawal transaction completes, the balance amount should be updated in all the other cache nodes as well.

### COHERENT MULTINODE CACHE



Each cache server can communicate to the other caches instances either synchronously or asynchronously. In synchronous communication, when application updates the cached table, the updates are applied to all the cache instances before returning to the application. In asynchronous communication, when application updates the table, the updates are applied to CSQL MMDB and returns to the application. Another process “applier” takes care of applying these updates on other cache instances asynchronously thereby improving the throughput of updates.

## Summary

- Easy to use
- High performance and throughput
- Multiple Cache Granularity
- Updateable cache tables
- Bi-directional updates
- Synchronous and Asynchronous transaction propagation
- High Availability
- Transparent fail over to target database
- Load Balancing with many cache nodes
- Automatic Recovery
- Data consistency across all cache nodes
- No application or data migration
- Standard ODBC and JDBC interface support
- Reduces TCO and support costs



For more information, visit Product web site at <http://www.csqldb.com>  
Company web site at <http://www.lakshyasolutions.com>

LAKSHYA Solutions Limited, 1st Floor, #73&74 Margosa Road, ABMGP Building, 17th Cross, Malleshwaram, Bangalore – 560055, Karnataka, India. Call - +91.80.4149.0561

Copyright @ LAKSHYA 2008

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.