



# Leveraging the Benefits of SQLBase 11 in Database Applications

Unify Corporation  
Unify.com  
2007



---

## Table of Contents

Introduction .....	3
Multi-versioning Theory .....	3
The 'start of statement' Timestamp .....	3
Summary of Isolation Levels Prior to SQLBase 11 .....	4
Introducing the New Read Committed Isolation Level.....	4
Read Only Isolation Level Changes .....	6
Multiple Connections Using Different Isolation Levels .....	7
How a Read Committed Transaction Affects Other Operations ...	7
Team Developer and SqlSetIsolationLevel( ).....	8
Q&A.....	8

---

## Introduction

SQLBase 11 introduces a new Lock Manager and a new multi-versioning code which greatly improves performance of your application and database operations.

This paper will guide you through these new concepts as well as provide enough information for you to modify your applications to take full advantage of the SQLBase 11 performance benefits.

## Multi-versioning Theory

Multi-version concurrency control (MVCC) is a concurrency control method used by database management systems to provide concurrent access to database objects and consequently improve performance in a multi-user environment.

From a client's perspective, MVCC provides each connected user with a snapshot of the database. Any changes made by one user will not be seen by other users until the transaction has been committed.

MVCC database systems maintain several versions of an object so a transaction never has to wait for a lock on a database object. For instance, when a database item is updated, an old version of that item is kept. When a transaction tries to read that item, the appropriate version is chosen so serializability of the current transactions is maintained. When a transaction tries to write an item, a new version is created and the old version of the item is retained.

SQLBase 11 uses the concept of view serializability within its multi-versioning control algorithms.

## The 'start of statement' Timestamp

Multi-version concurrency control systems use timestamps to achieve serializability.

As discussed above, versions of modified items are kept by the database system so they can be used by read and write transactions. During a select operation, the transaction can use that version of the item and it will not get any locks. So the set of data that this transaction will 'see' depends on the timestamp used by the transaction's isolation level.

The SQLBase 11 'start of statement' timestamp represents the date and time that a database cursor is opened for select. In other words, it's the date and time when the cursor actually starts scanning for data.

.....

This means that when using the ‘start of statement’ timestamp, any uncommitted data or data committed after the cursor starts scanning will not be seen by the transaction. The only data that will be seen is really the one committed at the start of the select operation.

## Summary of Isolation Levels Prior to SQLBase 11

**Read Repeatability (RR)** is the default isolation level. RR provides a high level of data consistency at the expense of user concurrency. All S-locks (shared) placed on data as a result of a SELECT statement remain in place until the end of the transaction. This isolation level guarantees data consistency for the life of the transaction. Identical SELECT statements will return identical result sets since data cannot be changed by other transactions.

**Cursor Stability (CS)** provides a medium degree of data consistency along with a medium degree of user concurrency. When using CS, an S-lock is maintained on the current page you are processing. As you fetch rows after executing a query, the page of the current row is held with an S-lock. You are guaranteed stability at the cursor (cursor in this context is the current row of the result set). When the page changes as rows are fetched, the S-lock on the current page is released and an S-lock for the next page is acquired.

**Release Locks (RL)** provides a low degree of data consistency and a high degree of user concurrency. This isolation level holds locks only as long as needed to read data. RL places S-locks on data as it builds a result set. Once the result set is complete and the server is ready to return control to the client, all S-locks will have been released. As rows are fetched from the result set, an S-lock is briefly placed on the page of the row to place it in the input message buffer. Unlike with CS, this S-lock is immediately released.

**Read Only (RO)** provides both the highest degree of data consistency and the highest degree of user concurrency. RO can only be used for queries (select statements). No Data Manipulation Language (DML) or Data Definition Language (DDL) statements are permitted in the transaction when RO is used. To accommodate RO, SQLBase creates history files of the data as it is modified by other transactions. This isolation level uses no locks and is not blocked by locks. If an X-lock (exclusive) is on the page that contains rows applicable to a query, the history file provides the rows instead of the rows coming directly from the database.

## Introducing the New Read Committed Isolation Level

SQLBase 11 was re-designed to improve the read only (RO) isolation capabilities for selects and the release locks (RL) capabilities for insert and update operations.

.....

In this scenario, a new Read Committed (RC) isolation level was introduced and it provides 'row-level locking' like behavior to SQLBase.

How does it work? When a SQLBase transaction modifies a page, it makes a copy of the page in cache. This way, any Read Only (RO) or Read Committed (RC) transaction may use that copy without getting any locks. For transactions that primarily read data, this dramatically improves concurrency.

When using the Read Committed (RC) isolation level, a snapshot of the data is taken at the time that a select statement starts (refer to 'start of statement' timestamp above). Therefore, any modification made to the data after the snapshot was taken will not be seen by the transaction that started the select statement.

As an example, run the following from two different SQLTalk sessions:

**Session1**

```
connect db sysadm/sysadm;  
set isolation rc;  
create table test (col1 int);  
commit;  
prepare select * from test;  
perform; <this sets the 'start of statement' timestamp>
```

**Session2**

```
connect db sysadm/sysadm;  
set isolation rc;  
insert into test values (1);  
commit;  
select * from test; <value 1 is shown>
```

**Go back to Session1**

```
fetch 10; <value 1 is not shown>
```

To solve this problem, re-execute the 'perform' statement from Session1 and the current data will be shown.

In the RC isolation level, write locks are released at the end of the transaction.

Note that unlike Read Only (RO) transactions, Read Committed (RC) transactions can both read and modify data. In addition, Read Only (RO) transactions can 'see' all the data that is selected as of the time the transaction started, which differs from the 'start of statement' timestamp in Read Committed (RC) transactions.

As in the Read Only (RO) mode, the Read Committed (RC) isolation level will also make use of a history file to maintain read committed transaction pages.

In this scenario, we can say that Read Committed (RC) transactions have Release Locks (RL)-like behavior for updates and Read Only (RO)-like behavior for selects.

## Read Only Isolation Level Changes

SQLBase's 'readonly' parameter for a database must be ON in order to support the Read Committed (RC) and Read Only (RO) isolation levels.

In the past, this parameter wasn't enabled by default. Because of performance improvements of read only transactions in SQLBase 11, the 'readonly' parameter is now enabled by default, if not otherwise overridden in the sql.ini for all databases on the server (readonly=0) or via SQLTalk's SET READ ONLY command for the current database.

The table below explains how the 'readonly' calculation works:

Last <b>SET READONLY</b> command	sql.ini setting		
	not set	0	1
DEFAULT	Enabled	Disabled	Enabled
ON	Enabled	Enabled	Enabled
OFF	Disabled	Disabled	Disabled

In order to execute the SET READONLY OFF command or SET READONLY DEFAULT (if the default is OFF), the requesting cursor cannot be running in Read Only (RO) isolation mode nor may transactions in that mode exist in the database.

Another change regarding the Read Only (RO) isolation level is that the "hisfilesize" parameter is now obsolete. It can still be set or retrieved but it will not affect the database operation. In SQLBase 11, the history file will grow or shrink as needed and it will be automatically deleted when the last user disconnects from the database.

For more information about the 'readonly' parameter, refer to the SQLBase Database Administrator's Guide, Keyword Descriptions section.

---

## Multiple Connections Using Different Isolation Levels

For each connection to SQLBase it is possible to have one set of cursors using a certain isolation level and another set of cursors using a different isolation level.

The advantage of this approach is that an application that makes extensive use of reporting capabilities (for which RO would be acceptable) and at the same time needs to rely on the most up-to-date information as in banking operations (where RL would be fine), that application could establish different connections to the database each using one of these isolation levels.

With SQLBase 11's Read Committed (RC) isolation level taking advantage of the read only improvements, Unify recommends that RC is used for all transactions. The reason is, as mentioned above, that read committed gives the best of both worlds: RO-like behavior for selects and RL-like behavior for inserts and updates.

Of course, this requires deep analysis of the application before making any changes. Please refer to the Q&A section below for Team Developer issues that may come up when using the new SQLBase 11 Read Committed mode.

## How a Read Committed Transaction Affects Other Operations

One of the caveats of the Read Committed (RC) mode is that different operations being executed in the same transaction will 'see' different data if modification took place after the statement had started scanning for data.

In other words, since a snapshot of the data is taken at the time that a statement starts, if a select operation takes place then another operation inserts a row in the table, the modified data will not be 'seen' by the cursor unless it is restarted.

From a Team Developer perspective, a `SqlPrepare( )` command will not start the timestamp but a `SqlExecute( )` command will. If your application prepares and executes select statements but only later in the process it will fetch through data, you might want to make sure that the data being fetched is the current data stored in the database. There are several ways of achieving it depending on the way your application is build. Making use of ROWID validation is one of the ways.

---

## Team Developer and SqlSetIsolationLevel( )

Team Developer (TD) uses the SQLBase API to set SQLBase isolation levels. Therefore, TD developers can continue to use the SqlSetIsolationLevel( ) function to set the new Read Committed (RC) isolation from the application by simply passing the RC parameter as following:

```
SqlSetIsolationLevel(hSql,'RC')
```

## Q&A

This section will help you find answers to most-commonly asked questions when migrating database applications that benefit from the SQLBase 11 new features.

**Q: I'm receiving error '9283 – Incorrect version of library' when connecting to SQLBase 11 from a Team Developer application. How do I solve it?**

*A: Make sure that the SQLBase 11 directory is set in the PATH prior to Team Developer's directory.*

**Q: When I try to connect to SQLBase 11 from my Team Developer application the error '10444 – SQL Error 10444, not found in ERROR.SQL file' shows up.**

*A: This error occurs when an incorrect version of the SQLBase API is in use. Applications connecting to SQLBase 11 need to use the API from version 11 as well. If you have files 'sqlws32.dll' and 'sqlbapw.dll' from an older SQLBase version in the application directory, you'll need to rename or delete them in order to connect.*

**Q: I'm using the function SqlSetIsolationLevel to enable the new Read Committed (RC) isolation level from my Team Developer application. When this function is executed I receive error '227 – Invalid Isolation Level RC'. Why?**

*A: Because your application is using a version of the SQLBase API older than SQLBase 11. Make sure that the files 'sqlws32.dll' and 'sqlbapw.dll' in use by your application show version 11.0.0.*

**Q: Can I set different isolation levels for different cursors of the same connection?**

*A: No. Different cursors of the same connection must use the same isolation level. For instance, if you open a SQLBase 11 connection from Team Developer,*

---

*oper and set the isolation level to 'RC', all cursors of that connection will also use 'RC'.*

**Q: Can I have multiple connections to SQLBase 11 using different isolation levels?**

*A: Yes. For instance, you can have one connection to SQLBase 11 using 'RO' isolation level and another connection using 'RC'.*

**Q: From my Team Developer application I try to set the SQLBase 11 isolation level to 'RC'. At this point I receive error '258 – Read only mode is disabled for this database'. How can I fix it?**

*A: In order to use the new SQLBase 11 'RC' isolation level, database 'readonly' mode must be enabled. This is the default when SQLBase 11 is installed but may have been changed from the sql.ini for all databases (readonly=0) or via SQLTalk for the current database. Make sure that 'readonly' mode is enabled by checking those two places.*

---

## About Unify

Unify is a global provider of software development technology and solutions that helps IT customers make SOA a reality and a success. Unify's productive and reliable development tools, migration solutions and databases enable organizations to deliver business essential applications in an SOA. Composer for Lotus Notes offers a complete migration solution for Lotus Notes business applications.

Unify's award-winning NXJ Developer enables IT teams to be extremely productive, learn new technologies fast and deliver SOA-based applications on time and on budget. The Team Developer, SQLBase, DataServer, ACCELL and VISION product families enable cross-platform rapid development on Java/J2EE, Linux or .Net. In its nearly thirty-years in software development, Unify has delivered emerging technologies to its thousands of IT customers and ISV, VAR and distributor partners. Unify is headquartered in Sacramento, Calif., and can be reached at (916) 928-6400 or by visiting [www.unify.com](http://www.unify.com).

Unify Corporation  
2101 Arena Blvd., Suite 100  
Sacramento, CA 95834  
USA

Phone: 1.916.928.6400

Toll Free: 1.800.468.6439

Fax: 1.916.928.6404

United Kingdom: +44 (0)1753 245 510

France: +33 (0)1 34 58 28 30

COPYRIGHT © 2007. UNIFY CORPORATION. All rights reserved.

Unify, the Unify logo, SQLBase and Team Developer are registered trademarks of Unify Corporation. Composer is a trademark of Unify Corporation. Java and J2EE are the trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All other company or product names are trademarks of their respective owners.

