MaxDB Synchronization at United Drugs' Annual Convention

A report on the usage of the MaxDB Synchronization Manager
by Mark Thomas (United Drugs) and C.J. Collier (MySQL)

Table of Contents

Foreword

This document is a user report on the MaxDB Synchronization Manager. The article describes the use of the MaxDB Synchronization Manager during United Drugs Annual Convention from a technical standpoint. It gives an example of the magnitude of possible application of the MaxDB Synchronization Manager. Written for developers, the user report contains warnings on common pitfalls and gives hints how to circumvent the problems.

We decided to add a short article on the technical features of the Synchronization Manager to this document in the hope that it will help you understand the terms mentioned in the user report and the criticism made.

Contact Information

The MySQL AB Team is glad to answer your questions. Please use http://www.mysql.com/company/contact/ to contact us.

The MaxDB Synchronization Manager

By C.J. Collier, MySQL AB

The MaxDB Synchronization Manager is a replication tool introduced with MaxDB 7.6.00. The tool is available for download on http://dev.mysql.com/products/maxdb/ . It can by used to synchronize the contents of MaxDB database instances.

The Synchronization Manager can be used for more than simple replication. The development focus has been on an *intelligent* bi-directional synchronization. For this reason it features an administration GUI which can be used to define *how and which* data will be synchronized within a unit of databases participating in a synchronization setup.

The synchronization service has not been developed to complement the High-Avaiability features of MaxDB (Standby and HotStandby) or with scale-out in mind. Unlike MySQL replication the primary use is not to replicate the data of one Master to several Slaves as fast as possible for load-balancing purpose.

Application Scenarios

The Synchronization Manager allows you to use selections and projections when setting up the Synchronization. This qualifies the tool to be used in a wide area of applications. Selection means, that you can define rules to replicate only certain records of one database server to another. For example, your international organization could have multiple, local database servers and one master server in the headquarter. The server of the US subsidary might only contain the data of all US-customers, whereas the server of the European subsidary contains all european customers. Using the feature of selections you can forward all new records inserted to the server in the headquarter to the appropriate subsidaries.

FIRST_NAME	LAST_NAME	COUNTRY
C.J.	Collier	USA
Mark	Thomas	USA
Ulf	Wendel	Germany

Selection:
COUNTRY = USA

FIRST_NAME	LAST_NAME	COUNTRY
C.J.	Collier	USA
Mark	Thomas	USA

But you do not need an international organization to find a good example for the usefullness of this feature. Say for instance that you have several field sales people and a central database server in the headquarter. Every sales person is equipped with a notebook running a CRM solution based on MaxDB. During the visit of the customer, the sales person modifies the customer records. The changes made need to be uploaded from the notebook of the sales person to the main company database. The upload is done using the bi-directional synchronization feature. At the same time, selected leads that have been inserted into the master server during the day are downloaded to the notebook of the sales person. Using the feature of selections, only the leads of which the sales person is responsible are synchronized.

The second main feature that makes the synchronization of the MaxDB Synchronization Manager a sophisticated one is projection. Projection means, that only selected columns of a source table are made visible to the target. To continue the example of the sales engineers, certain information on a customer of a certain sales engineer might be available to other sales people. For example, the names and industrial areas might be visible to other sales people so that they can use the names to advertise their product but they cannot see the contact informations or sales notes.

FIRST_NAME	LAST_NAME	COUNTRY
C.J.	Collier	USA
Mark	Thomas	USA
Ulf	Wendel	Germany

Projection:
COUNTRY

COUNTRY
USA
USA
Germany

Also, in the area of Data Warehousing projections can be helpful. One major task in Data Warehouses is the extraction, load and transformation of data (ETL – extract, transfer, load)). Using the MaxDB Synchronization Manager you could extract only selected columns from your production systems (extract), synchronize them to your warehousing server (load) and transfer them to the target tables of the warehouse. For example, you could use projection to synchronize the data of the customer table of your production system to the country facts table of your warehousing server transferring only the values of the country column of the customer table.

Technical Requirements

Behind the scenes the solution is based on SQL trigger and Java technology. SQL trigger are the base of change notifications and Java is used to run the GUI and synchronization processes.

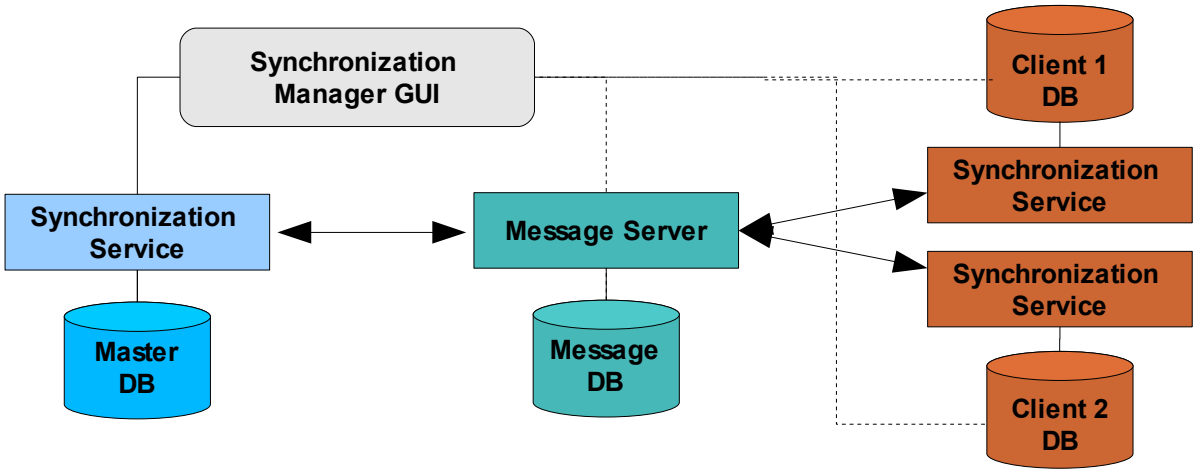
Beeing based on Java has some implications for Open Source users. One the one hand Open Source users of MaxDB profit from the about 250(!) test and development servers used by SAP to ensure the compatibility and quality of MaxDB with many operating systems and different generations of SAP applications using nightly builds, on the other hand the number Linux distributions that can be tested is still limited. MaxDB has been released on some 28 different platforms for Open Source users, please check the Platform Availability Matrix[1] for details. However, there are some not officially supported Linux distributions that come with other selections of libraries and other tools than needed by MaxDB. If you plan to use any such unsupported Linux distribution, you might be forced to fine-tune the out-of-the-box operating system installations and add or update some software packages of your operating system.

Database Landscape

A synchronization scenario consists of at least three database instances: a master database, a message database and at least one client database. You can run all database instances on the same server, but most usage scenarios will require more than only one server. On every database (master and clients) participating in a synchronization an additional synchronization service process is started. The synchronization services are communicating with their database instances and with the message server process which achives messages in the message database.

1 http://dev.mysql.com/doc/maxdb/pdf/pam_non_sap.pdf

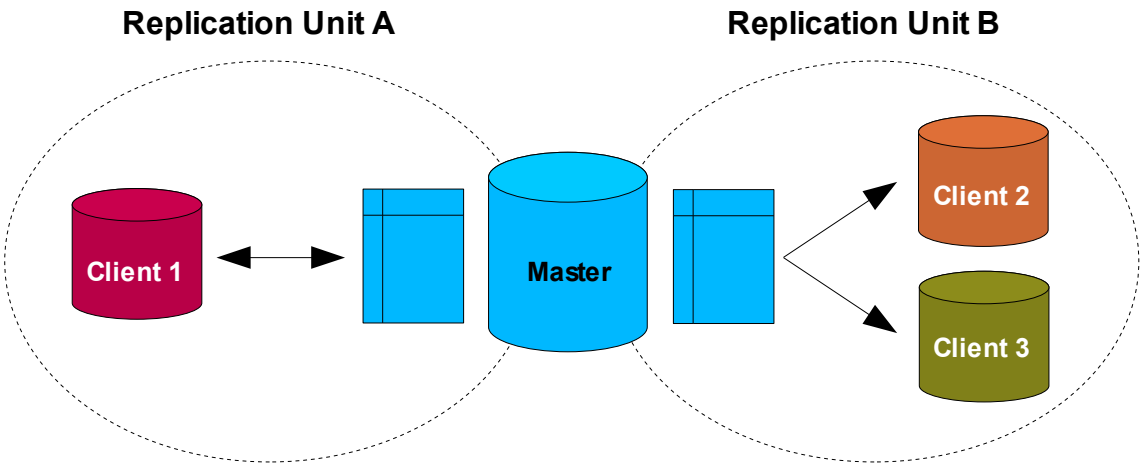
The Message Server and the Message Database are the "brain" of the system. Changes that are made to any of the databases cause message to be created and exchanged between the participants. For example, if you add a new record to the Master Database, the Synchronization Service of the Master Database gets informed by a SQL trigger and creates a new message in a "shadow table". The Synchronization Service is an extra java-based application that needs to be set up during the installation of the entire synchronization scenario.The change notification message created by the Synchronization Service is sent to the Message Server. Again, the Message Server is an extra java-based application. It uses a Message Database (a MaxDB database instance) to archive messages. According to the synchronization rules configured using the Synchronization Manager GUI, the Message Server transforms and forwards the messages it gets to the Synchronization Services running on the Client Databases.



Challenges of Distributed Systems

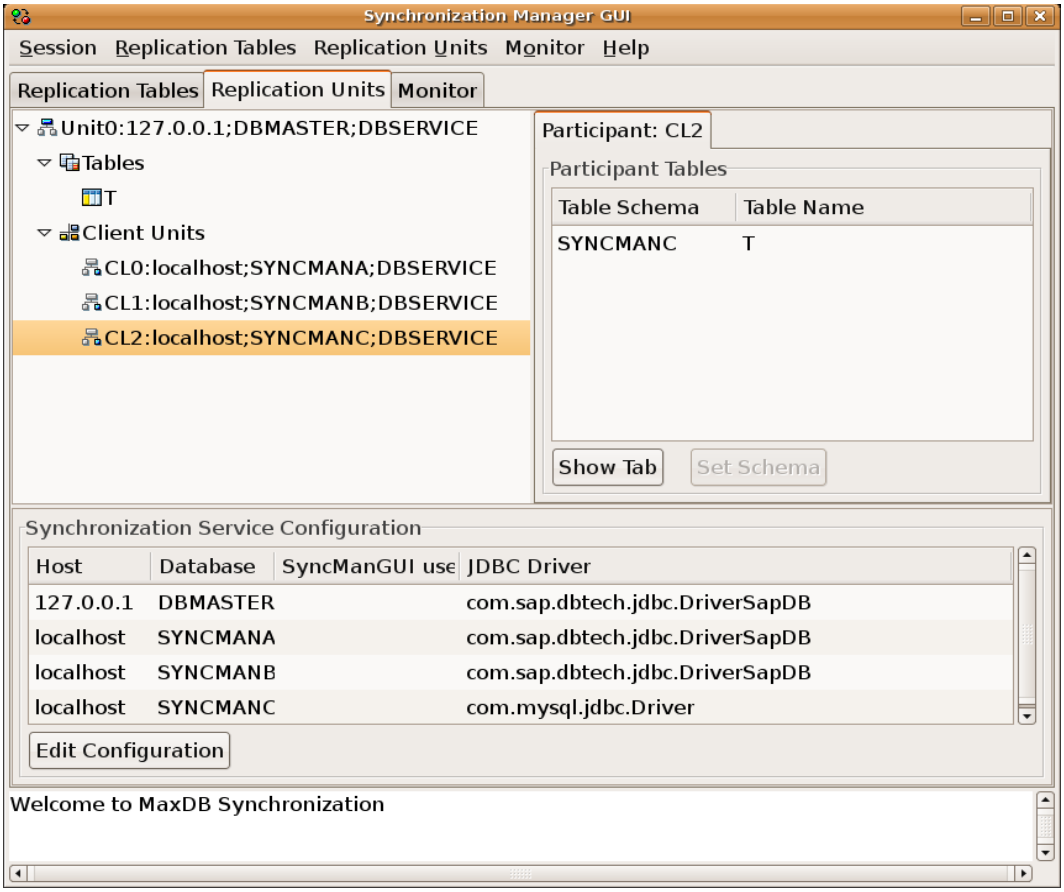
The rules of the synchronization are grouped in so called Replication Units. A Replication Unit is a description of which modfitication of certain tables on certain databases will be forwarded to other databases. A synchronization scenario can consist of as many Replication Units as you want.

The preceeding drawing shows two Replication Units. Replication Unit A illustrates a bi-directional synchronization of one table of the Master Database with the database of Client 1. Bi-directional means that both can modify the table and both witll retrieve modification messages from one another. Replication Unit B shows the uni-directional replication of one table of the Master Database to the databases of Client 2 and Client 3. In this second case modification messages are only sent from the Master Database to the Clients. The clients do not upload any modifications to the Master.



No matter if you define a uni-directional or bi-directional synchronization you must always be aware of a common pitfall of distributed systems: collisions. A classic area of collisions are primary keys or other unique columns. In the case of Replication Unit A and bi-directional synchronization it is obvious how it can happen that two records try to use the same value for a unique column. The Master and the Client could be using their own local database sequences to generate primary key values. At some point both sequences might return the same number, which results in both the Master and the Client trying to insert a record with the same primary key.

Such conflicts can even occur if you use uni-directional synchronization. For example an application running on the Client 2 database that belongs to the Replication Unit B could create a conflicting record. This conflict can only be solved on the client application level, whereas conflicts in a bi-directional setup can be detected and handled on the level of the Message Server.



Future Developments

We believe that the Synchronization Manager is a product with a huge market potential. Therefore we are eager to hear your feedback and learn from you what you need. One of the features we are working on is the integration of the MySQL Server into the synchronization process. The MySQL Server will be able to participate as a bi-drectional Slave in mixed landscapes with MaxDB databases in the near future.

Being a new development, the Synchronization Manager has been fully integrated into the development of the next version of MaxDB. Version 7.7 will come with a new eclipse based database tool suite which integrates various GUI tools into a larger tool suite. The tool suite will be shipped with a eclipse runtime and therefore no seperate downloads expect a j2ee.jar will be needed. This will further increase the number of (officially unsupported) linux distributions that can be used for out-of-the-box installations.

User report

By Mark Thomas, United Drugs

For the past four years, United Drugs' annual convention has required detailed data entry and retrieval processes for pharmacist, exhibitor, guest, speaker and staff registration. United Drugs uses a custom registration application developed in Microsoft Access which handles room bookings, payment processing, etc.

During the convention, laptops with the convention application installed are generally deployed at various locations to assist participants with registration and payment updates. The laptops are not networked, so changes made in one version would not be reflected in the other. The disparities have always needed to be reconciled later, using a labor-intensive, manual process. This year, we wanted to try something new.

Scenario

Implementing batch-oriented synchronization logic in the application itself would be too complicated to be economical. Synchronization at the database layer using an existing product, however, seemed a viable option. Since United Drugs has an existing investment in MaxDB, the decision was made to try using the MaxDB Synchronization Manager to solve the problem.

Architecture

Initially, the convention application was modified so that key MS Access tables (stored in the Access MDB file using Microsoft's Jet database engine) were replaced by ODBC links to a MaxDB database.

A new 7.6 instance was commissioned for this exercise on a spare PC running SuSE Linux 10[2]. A pair of Dell laptops running Windows XP was loaded with MaxDB 7.6 as well[3]. All three installations included the synchronization service. The copy of the convention application included on the laptops had the ODBC links modified to point to the local instances.

Every evening, a member of the IT staff at the convention connected both laptops to the hotel's high-speed Internet connection and connected to the United Drugs VPN. The sync service was used to synchronize all edits between the laptops and the master database.

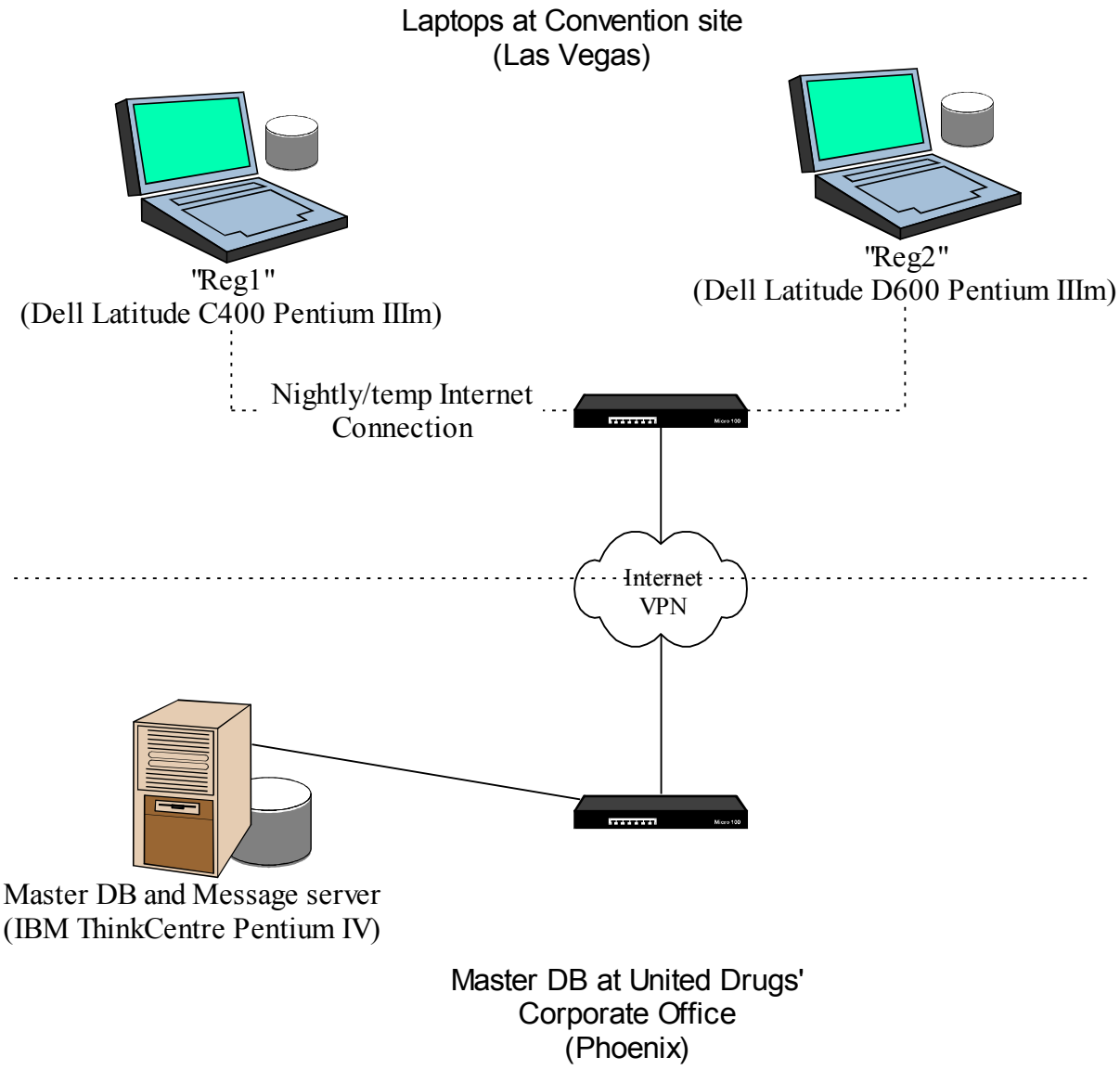
Installation and Configuration

The first phase involved the installation and configuration of the synchronization service and message service. Since our primary database is still using MaxDB 7.5.0.31 and the Synchronization Manager requires 7.6, we could not integrate this application with the rest of our data.

A new database was commissioned on a spare PC running SuSE Linux 10 which served as the master Synchronization partner.

2 IBM ThinkCentre A50p: Pentium IV, 2.66 GHz, 60GB IDE hard drive, 1GB RAM. A bit overpowered for a database this size, but the point is that we used desktop hardware that we had lying around at the time.

3 The first laptop is a Dell Latitude C400 Pentium IIIm, 1.2GHz with 256MB RAM and 30GB hard drive. The second is a Dell Latitude D600 Pentium IIIm, 1.4GHz, 512MB RAM, 20GB hard drive.



Dependencies

The Synchronization Manager is dependent on Java and requires SQL triggers for all bi-directional partner databases. Therefore, recent a recent version of the Sun Java runtime must be installed. The Sun Java runtime that ships with SuSE Linux appears to be incompatible with the syncmangui, so it was necessary to uninstall that version before proceeding. We used the download links and version info in the Collier HOWTO[4] to obtain the j2ee.jar, java runtime, and eclipse installation files that we used. Two of the eclipse libraries appeared to be missing, and had to be obtained via the #maxdb IRC channel and installed separately. It took about 3 full days of tinkering to get the syncmangui, syncservice, and messageservice to run.

4 Installing MySQL MaxDB Synchronization Manager, http://dev.mysql.com/tech-resources/articles/syncman/index.html

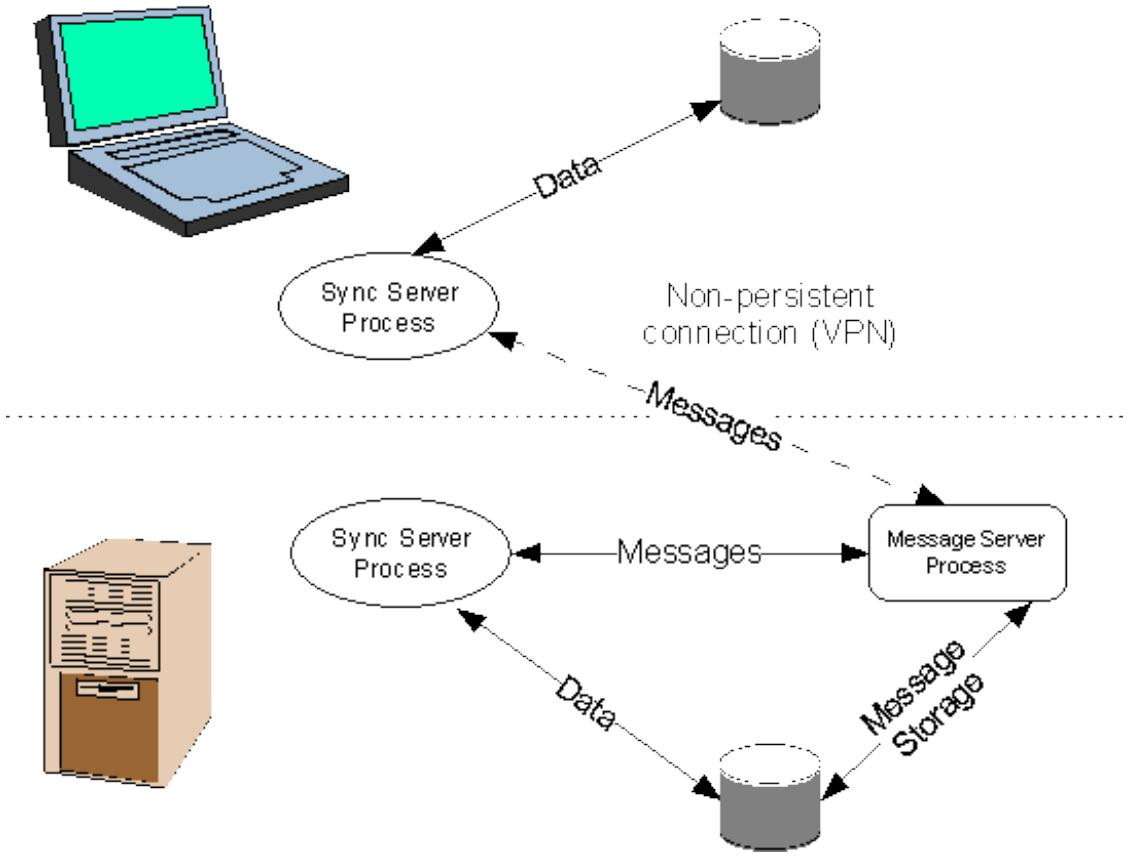
Database structures

The database structures were converted from Jet tables into data definition language (DDL) by hand, then created in the MaxDB master database (designated CONVENT) utilizing the MaxDB DBLoader. The tables were divided into two groups: replication tables and “static”, non-replicated tables.

The replication group consisted of four tables which would be assigned to “In/Out” replication groups in all databases. The second group contained an additional 3-4 tables from our primary MaxDB database (UNITEDDB) that are critical to the application. Up-to-date copies of this data were needed by the convention application, but this application would not be permitted to change the contents of these tables, so they would be copied by DBLoader to the CONVENT database and assigned to an “Out” replication group.

We were unable to complete the setup of the tables in the second (“Out”) group, because one of the tables consisted of rows that were too large for the syncservice to handle.

The synchronization of this group was abandoned, and static snapshots were copied into Jet tables instead.



Synchronization setup

Our initial tests of the synchronization scenario we wished to use proved easy to set up, once all of the software was installed and working. We discovered that the process was relatively fragile and prone to corruption by operator error and once broken, was impossible to repair. Several times during the testing we had to drop all of the shadow and message tables and reconfigure synchronization from scratch. This also meant reinitializing the client units.

Deployment

Each of the laptops was given its own installation of MaxDB, including the DB engine and synchronization service binaries. The laptop database instances were configured by hand, and the table schemas were then deployed using the DBLoader on the master DB computer. A copy of the convention application was deployed to each laptop, and the ODBC links to the replicated tables were pointed to "localhost". The laptops were then populated with test data, synchronization was enabled, and the laptops were given to the users to test.

Here, we discovered two issues we had overlooked: First, the MaxDB synchronization process requires that all databases have static IP addresses which do not change after their client units are "activated". [5].

This was a problem, since both the convention hotel's network and our own VPN assign dynamic IP addresses from a pool. We were able to work around the problem by creating two additional 1-address pools on the VPN gateway and configuring each laptop to gain access through a separate pool. Thus, the addresses were fixed to a "static" value—at least as far as the message server was concerned. This added a degree of complexity to the final "client unit activation", since the laptops had to be outside our company LAN, accessing the VPN, to be "activated". This was ultimately not difficult to achieve.

The second issue had more serious implications. The developers of the convention application have become accustomed to using MaxDB's "default serial" for the assignment of primary key values. The problem is: adding "default serial" to a column causes the synchronization service to terminate abnormally. We needed a new keying strategy.

The application was quickly modified to generate its own sequential key values and assign them to new records. At this point, the code for the three copies of the convention application (LAN or internal use and each laptop) was forked. Each copy was configured to assign keys from a different range, so that they would not assign conflicting keys. It was a crude solution which wouldn't have worked indefinitely, but allowed us to function for the duration of the convention.

At this point, the client tables were populated, synchronization was tested again, backups were made of each of the databases, and the laptops were packaged for transportation to Las Vegas.

Results

The synchronization server performed as expected at the convention. The laptops were easily kept in sync with each other, and with the master database, with minimal effort. Although the setup proved to be more difficult than anticipated, we consider the project successful. Thanks to C.J. Collier and the SAP team for their assistance.

Issues summary

- The Sun Java runtime that ships with SuSE Linux appears to be incompatible with the syncmangui, so a different version must be obtained.
- The row size of one of our central "legacy" tables is too large for the synchronization service to handle. This can be ameliorated somewhat by tinkering with the _PACKET_SIZE database parameter.
- Sync process is relatively fragile and prone to corruption. For example, activating or deactivating a synchronization unit while the syncservice process was running will likely corrupt the

5 *Note from the SAP Development Team*: Actually, static IPs are not required, we recommend that domain/host names are used. Unfortunately by setting defaults to '127.0.0.1' instead of 'localhost', it was suggested that IPs are needed. This was done, since some linux flavors resolve 'localhost' to an IPv6 IP (notably SuSE) . Such IPs are not understood by MaxDB's x_server and (at the time by many other applications as well)We will set the default back to 'localhost' and address the IPv6 problem in the documentation.

synchronization capability beyond repair, so that all clients will have to be reinitialized from scratch.

- The message server requires that the client systems have static IP addresses. This makes the whole process somewhat impractical for mobile use.
- Assignment of unique key values cannot utilize default serial or sequences. When using synchronization, the generation and management of unique key values has to be moved from the database to the application layer.