

Mobilizing an Oracle 10g Database

A whitepaper from Sybase iAnywhere

CONTENTS

| | |
|--|----|
| Contents | 2 |
| Data Mobilization: An Overview | 3 |
| Prerequisites | 3 |
| Synchronization Environment | 4 |
| Consolidated Schema | 5 |
| Preparing the Consolidated Database | 7 |
| Connecting with MobiLink | 7 |
| Creating a Synchronization Model | 8 |
| Deploying the Synchronization Model | 12 |
| Synchronizing | 15 |
| Starting the Consolidated Database | 15 |
| Starting the MobiLink Server | 15 |
| Starting the Remote Database | 16 |
| Initializing the Remote ID | 16 |
| Synchronizing the Remote Client | 17 |
| Viewing remote client data in Sybase Central | 18 |
| Updating the Remote For Non-Query Actions | 19 |
| Ensuring Unique Primary Keys Across Remote Databases | 21 |
| Recreating the Synchronization Subscription | 22 |
| Conclusion | 24 |

DATA MOBILIZATION: AN OVERVIEW

Data mobilization provides a method for enterprises to bring their mission-critical information to personnel at the front lines of business. It enables employees at remote or branch offices to access and manipulate corporate data without requiring a constant network connection to headquarters. This significantly reduces connectivity costs and issues, and ensures that remote office workers are productive because they can always have access to the data.

MobiLink is a session-based synchronization system that allows two-way synchronization between a main database, called the consolidated database, and many remote databases. The consolidated database, which can be one of several ODBC-compliant databases, holds the master copy of all the data. Remote databases can be either SQL Anywhere or UltraLite databases. MobiLink synchronization is a component of SQL Anywhere, the comprehensive package for mobile data management from Sybase iAnywhere.

Synchronization typically begins when a MobiLink remote client opens a connection to a MobiLink synchronization server. During synchronization, the MobiLink client uploads database changes that were made to the remote database since the previous synchronization. On receiving this data, the MobiLink synchronization server updates the consolidated database, and then downloads changes on the consolidated database to the remote database.

MobiLink is proven to be a scalable, high-performance data synchronization technology placing iAnywhere as significant leaders in the mobile workspace. It functions on multiple platforms (including Windows, Mac OS X, Linux, PocketPC, Windows Mobile, Symbian and Palm) and works across any type of wired or wireless connection. In terms of development and database design there is the option of the intuitive and easy-to-use graphical interface of Sybase Central, or alternatively for more advanced developers there is the option to configure the SQL code directly. This allows the developer to concentrate on their code. A wide variety of development technologies are supported, including .NET and Java. Other features include monitoring and reporting, performance tuning and security.

For more information on each of the components in a MobiLink synchronization system, you should consult the product documentation. The section entitled "Introducing MobiLink Synchronization" in the book *MobiLink - Getting Started* gives an excellent overview. It is recommended that you read this section before proceeding with this whitepaper. SQL Anywhere documentation is available online at http://www.iAnywhere.com/developer/product_manuals/sqlanywhere/.

The rest of this document will walk through the mobilization of an Oracle database.

PREREQUISITES

- Oracle 10g
- SQL Anywhere 10.0.1
- MobiLink 10 (included in SQL Anywhere 10)

This demonstration uses the Order Entry (*OE*) and Human Relations (*HR*) sample schemas that are provided with the Oracle 10g installation. Information about this sample can be found in the Oracle documentation or can be viewed online at:

<http://www.oracle.com/technology/obe/obe1013jdev/common/OBEConnection.htm>

SYNCHRONIZATION ENVIRONMENT

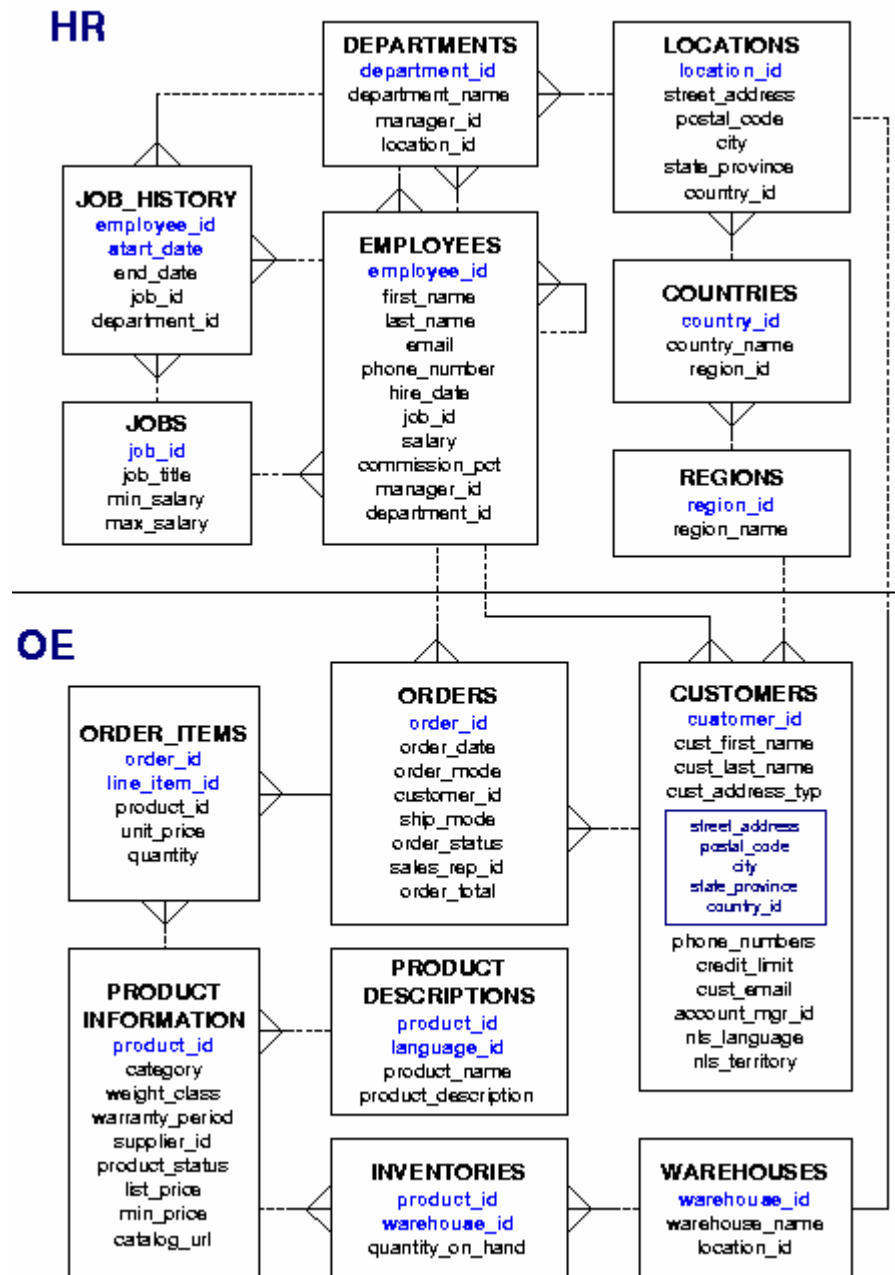
The purpose of this demonstration is to mobilize the data pertaining to a sales force team. The MobiLink synchronization server is capable of synchronizing with any type of SQL Anywhere remote client, such as SQL Anywhere or UltraLite. In this scenario, each remote client is a SQL Anywhere database running on a sales person's laptop or handheld device. The diagram below illustrates the synchronization environment.



Each sales person in this scenario is a remote synchronization client. They each have a local SQL Anywhere database that is synchronized with a corporate Oracle database. A salesperson can download a subset of the corporate data to their laptop or handheld device, and manipulate that data from the remote database.

CONSOLIDATED SCHEMA

The consolidated database schema encapsulates information about employees, orders, customers and products. The diagram below outlines the relationships between the different elements of the schema¹.



We are primarily interested in the *OE* schema, however we need to refer to the *HR* schema to get information about each individual sales person from the EMPLOYEES table. Each entity above

¹ Oracle Database Sample Schemas Documentation
http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14198/diagrams.htm

represents a table that exists in a consolidated Oracle database. Here is a brief description of each object.

- CUSTOMERS – customers whose information is kept on record
- INVENTORIES – how much of each product is stored in each warehouse
- ORDER_ITEMS – list of products included in each order
- ORDERS – record of a sale between a salesperson and a customer on a specific date
- PRODUCT_DESCRIPTION – descriptions of each product in different languages
- PRODUCT_INFORMATION – a record of each product in the system

This walk-through outlines how to make a subset of this data available locally to each person on the sales team.

The first step is to design a remote schema. It is unnecessary and inefficient for each salesperson to have a copy of the entire consolidated database. The remote schema will be designed such that it only contains information relevant to one particular salesperson. To achieve this, it will be designed as a subset of the consolidated database in the following way:

- CUSTOMERS – all rows
- INVENTORIES – no rows
- ORDER_ITEMS – filter by sales_rep_id
- ORDERS – filter by sales_rep_id
- PRODUCT_DESCRIPTION – no rows
- PRODUCT_INFORMATION – all rows

Each salesperson needs to keep records of all customers and products, so that any product can be sold to any customer. We will assume a salesperson always speaks the same language as the customer they are selling to, so we do not need any rows from the product description table. Each salesperson needs information about orders and order items, but not ones related to other salespeople, this is achieved by filtering rows based on salesperson identifier.

Note

Above we demonstrate designing a remote schema by excluding tables and taking a subset of rows from other tables. We can also take a subset of columns from a table if certain columns are not required on the remote devices.

The next step is to choose the synchronization direction of each table. What is important to consider here is what information does a remote need to read, and what information does a remote need to create. In our example, a specific salesperson will need a list of products and customers, but will never enter a new product into the system. We are making the restriction that products and customers will always enter the system from the consolidated database at headquarters. However, a salesperson needs to be able to record new orders on a regular basis. These factors lead to the following decisions about each table:

- CUSTOMERS – download to remote only
- ORDER_ITEMS – download and upload
- ORDER – download and upload
- PRODUCT_INFORMATION – download to remote only

PREPARING THE CONSOLIDATED DATABASE

The following statements are designed to prepare the Oracle database for use with MobiLink. Use the Oracle SQL Plus application to connect to the *OE* schema and execute these queries (be sure to connect as SYS):

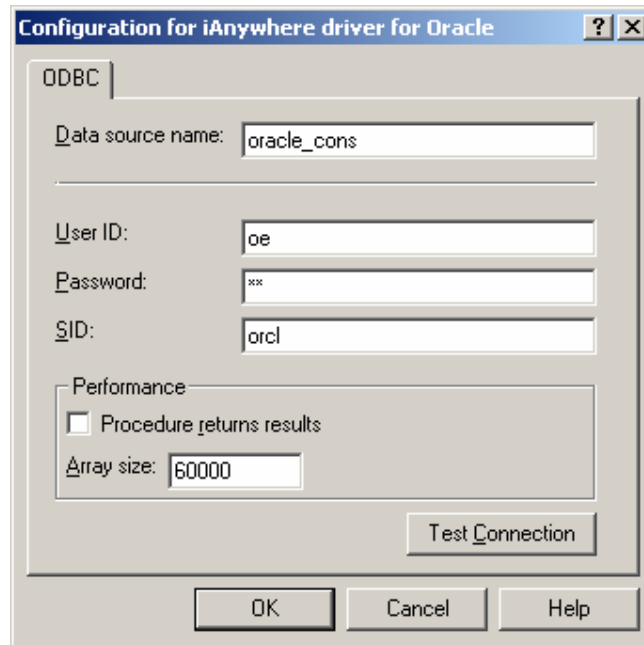
```
ALTER TABLE customers DROP COLUMN cust_address;  
ALTER TABLE customers DROP COLUMN phone_numbers;  
ALTER TABLE customers DROP COLUMN cust_geo_location;  
ALTER TABLE product_information DROP COLUMN warranty_period;  
  
GRANT CREATE ANY TRIGGER TO oe;
```

Certain columns are dropped because they were created as user-defined types. We could find out how to translate these user-defined types into types that SQL Anywhere will recognize, but doing so is not relevant to this paper. Next, we grant permission to user *OE* to create triggers because MobiLink needs to create a few triggers using *OE*'s credentials.

After running the DDL commands listed above, the MobiLink server should have no trouble connecting to the database and setting it up for synchronization with any number of remote devices.

CONNECTING WITH MOBILINK

1. Verify that your Oracle service is running and that the *OE* database is configured properly.
2. Create an ODBC data source that points to *OE*. You should use the iAnywhere driver for Oracle² that comes with SQL Anywhere 10.0.1. The configuration should look similar to the image below.

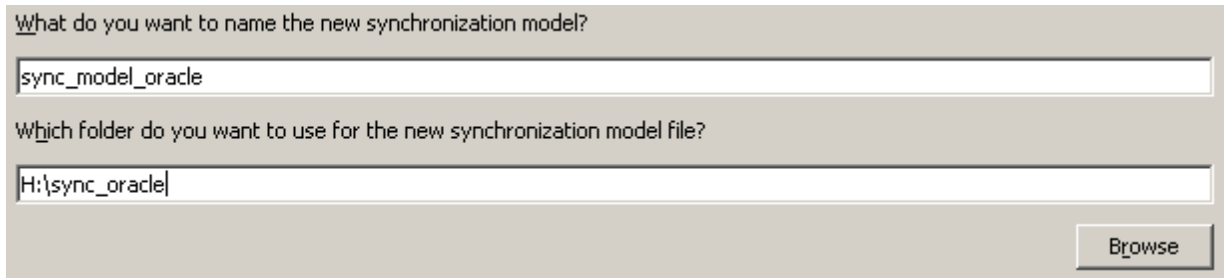


² For more information about recommended ODBC drivers for MobiLink, see <http://www.sybase.com/detail?id=1051962>.

After configuring your ODBC data source, you can use the MobiLink plug-in to connect to the Oracle database and create a synchronization model.

CREATING A SYNCHRONIZATION MODEL

1. Start Sybase Central by choosing Programs | SQL Anywhere 10 | Sybase Central.
2. From the Tools menu, choose MobiLink 10 | Set Up MobiLink Synchronization. The Create Synchronization Model wizard appears.
3. Enter the name and location of your new model, and then click Next.



What do you want to name the new synchronization model?

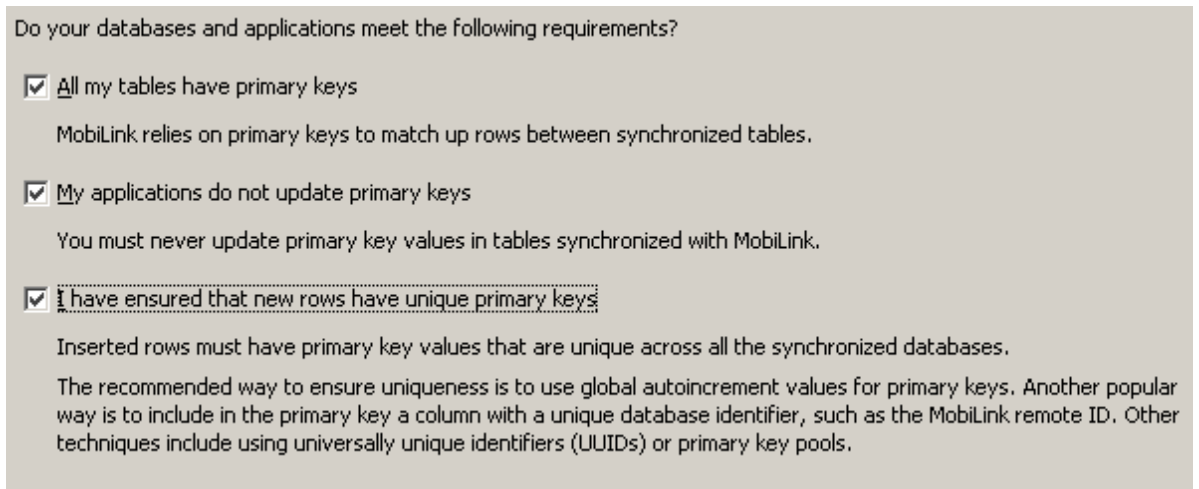
sync_model_oracle

Which folder do you want to use for the new synchronization model file?

H:\sync_oracle

Browse

4. The next page verifies that the remote database schema is ready for synchronization. You must select all of the options on this page before you can proceed. Click Next.



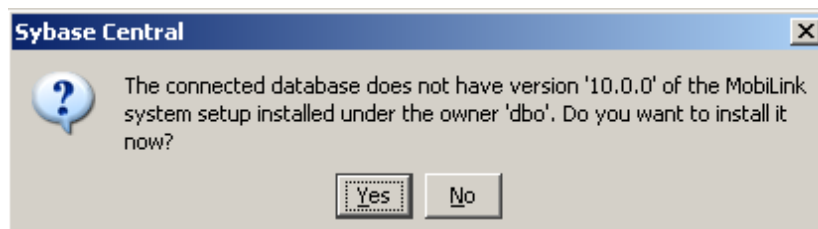
Do your databases and applications meet the following requirements?

All my tables have primary keys
MobiLink relies on primary keys to match up rows between synchronized tables.

My applications do not update primary keys
You must never update primary key values in tables synchronized with MobiLink.

I have ensured that new rows have unique primary keys
Inserted rows must have primary key values that are unique across all the synchronized databases.
The recommended way to ensure uniqueness is to use global autoincrement values for primary keys. Another popular way is to include in the primary key a column with a unique database identifier, such as the MobiLink remote ID. Other techniques include using universally unique identifiers (UUIDs) or primary key pools.

5. Choose the consolidated database. Type the name of the data source you created that points to *oe*, the user ID, and the password. Click OK.
6. If this is the first time your database has been used by MobiLink, the following dialog appears.



Sybase Central

The connected database does not have version '10.0.0' of the MobiLink system setup installed under the owner 'dbo'. Do you want to install it now?

Yes No

Click Yes. MobiLink automatically creates all the tables necessary for the database to be used in a synchronization environment. This does not alter your existing schema; it only adds MobiLink system tables and procedures.

7. Once MobiLink is finished making the necessary modifications, click Next.
8. Choose the remote schema. Since you have not created a remote schema yet, have MobiLink create one based on the consolidated schema by selecting No, Create a New Remote Database Schema. Click Next.

The screenshot shows a dialog box with two radio button options. The first option, "No, create a new remote database schema", is selected. Below it, text reads "You will choose tables from the consolidated schema to create the new remote database schema." The second option, "Yes, use an existing remote database", is unselected. Below it, text reads "Connect to an existing SQL Anywhere Server or SQL Anywhere UltraLite database to obtain the remote schema." There is a button labeled "Choose a remote database,..." and a list of labels: "Name:", "User:", "Product:", and "Version:".

9. Select the checkboxes for the following tables to include them in the remote schema, and then click Next.
 - customers
 - orders
 - order_items
 - product_information
10. Select Timestamp-based Download. Choosing timestamp-based downloads minimizes the amount of data that is transferred because only data that has been updated since the last download is transmitted. Click Next.

The screenshot shows a dialog box with three radio button options. The first option, "Timestamp-based download", is selected. Below it, text reads "Only download rows that have changed since the last download." The second option, "Snapshot download", is unselected. Below it, text reads "Download all rows every synchronization, even if they have been previously downloaded." The third option, "Custom download logic", is unselected. Below it, text reads "You will write your own download_cursor and download_delete_cursor scripts instead of having them generated automatically. You can write them in the Events editor."

11. Choose how to implement timestamp-based downloads. On this page, select Use Shadow Tables to Hold Timestamp Columns. Using shadow tables is often preferred because it only adds triggers to existing tables, as opposed to adding columns. Choosing the first option allows MobiLink to add a column to every synchronized table in the consolidated schema. Click Next.

The download-by-timestamp option requires a timestamp column for each consolidated table to track when rows are modified.

Timestamp column name:

Use timestamp columns in synchronized tables

The column will be added to each synchronized consolidated table if it does not already exist, and it will be excluded from synchronization. Use this option to avoid joining with a shadow table when selecting the data to download.

Use shadow tables to hold timestamp columns

A shadow table will be created for each synchronized consolidated database table. Use a shadow table if you do not want to modify the consolidated schema.

12. Configure how to propagate record deletions to remote devices. Select Yes to indicate that you want the remotes to download deletes. Select Use Shadow Tables to Record Deletions. MobiLink creates shadow tables on the consolidated database to implement this. Click Next.

Do you want data deleted on the consolidated database to be deleted on the remote databases?

Yes No

Since you are not using snapshot download, you need to record which rows have been deleted.

How do you want to record row deletions?

Use shadow tables to record deletions

A shadow table will be created to hold the primary key and time of deletion for each deleted row.

Timestamp column name:

Use logical deletes

Instead of deleting rows in the consolidated database, your applications will use a one-character column to indicate if the row has been deleted. This column will be added if not already present.

13. Specify that the remotes download only a subset of data from the consolidated database.

Do you want all remote databases to download the same data?

Yes, download the same data to each remote

No, download different subsets to different remote databases

Although the data will be partitioned by remote database, you will specify how later by using custom logic for each table in Model mode. Click Next.

14. Choose No Conflict Detection. For most applications you want to select either row or column-based conflict detection to ensure consistency throughout remotes, but for the purposes of this tutorial, you can select no conflict detection. Click Next.

Which type of conflict detection do you want?

No conflict detection
Always apply uploaded updates without checking for conflicts (best performance).

Row-based conflict detection
A conflict is detected if the row has been updated on both the remote and the consolidated databases since the last synchronization.

Column-based conflict detection
A conflict is detected only if the same column has been updated for the row in both the remote and consolidated databases. Otherwise only the uploaded column updates will be applied.

Note: If a table has BLOB columns then row-based conflict detection will be used.

15. Name your publication and script version, and then click Next. The publication is the object that specifies what data is to be synchronized. The script defines how data that is uploaded from remotes should be applied to the consolidated database. You can use different script versions for different applications, allowing you to maintain a single MobiLink server while synchronizing to entirely different applications. Click Finish.

The remote tables that you have chosen to synchronize will be grouped into a publication, and the synchronization logic scripts will be grouped into a script version.

What do you want to name the publication?

sync_model_oracle_publication

What do you want to name the script version?

sync_model_oracle_script

16. You should now be in the Model mode view and should be looking at the following screen:

| Table Mappings | | | | | |
|----------------|--------|--------------------------|------|--------------------------|------------|
| | Status | Remote Table ▲ | Dir. | Consolidated Table | Dnld. Type |
| 1 | | CUSTOMERS (OE) | ↔ | CUSTOMERS (OE) | Timestamp |
| 2 | | ORDER_ITEMS (OE) | ↔ | ORDER_ITEMS (OE) | Timestamp |
| 3 | | ORDERS (OE) | ↔ | ORDERS (OE) | Timestamp |
| 4 | | PRODUCT_INFORMATION (OE) | ↔ | PRODUCT_INFORMATION (OE) | Timestamp |

17. The Dir. column specifies the direction the data is passed (download-only, upload-only or bi-directional). Set the directions are follows:
- ORDERS and ORDER_ITEMS are bi-directional
 - The remaining tables are download only
18. Set up the row filtering procedure that you skipped in step 13. In a MobiLink environment, each remote database is given a remote ID. You can use this remote ID to filter rows. You do this in the download_cursor script for each synchronized table. The download cursor script specifies what columns and rows are downloaded from each table to the remote database. To filter rows by remote ID, you add a restriction to the WHERE clause of the download cursor.

1. For the ORDERS table, change the value in the Download Subset column to Custom.

- Click the Download Subset tab at the bottom of the screen.
- Fill in the text boxes on this tab as follows:

Details for ORDERS (OE)

Tables to add to the download cursor's FROM clause:

SQL expression to use in the download cursor's WHERE clause:

```
"OE"."ORDERS"."SALES_REP_ID" = {ml s.remote_id}
```

This ensures that you only download information about one salesperson, namely, the rep that has an identifier that equals the remote ID for the database. Later you must set up the remote database with a remote ID equal to a particular sales rep's identifier.

- Save the synchronization model. The synchronization model is complete and ready to be deployed.

DEPLOYING THE SYNCHRONIZATION MODEL

- In the left pane, right-click the synchronization model you just created and choose Deploy from the popup menu.
- Select all of the options, instructing MobiLink to deploy each piece of the synchronization environment. Click Next.

Specify the deployment details for one or more of the following:

Consolidated database

Remote database and synchronization client

Mobilink server

- MobiLink generates a .sql file with the commands it needs to execute on the consolidated database. Specify the location of the .sql file, and choose to connect to the consolidated database and have MobiLink run the file against it right away. Click Next when connected.

How do you want to deploy to the consolidated database?

Save changes to the following SQL file (and create a command file in the same folder):

H:\sync_oracle\sync_model_oracle\consolidated\sync_model_oracle_consolidated.sql

Connect to the consolidated database to directly apply the changes

Which database do you want to use as your consolidated database?

Choose a consolidated database...

Name: oracle_cons
 User: OE
 Product: Oracle
 Version: 10.02.0000 Oracle 10.2.0.1.0

4. Choose New SQL Anywhere Database to specify the type of remote database being deployed. Click Next.

New SQL Anywhere database

This will create a new, full-featured database for Windows (including Windows CE) or UNIX platforms, using the remote schema from the synchronization model and default database creation options.

New UltraLite database

This will create a new, compact database for Windows CE, Windows XP, Palm OS or Symbian OS, using the remote schema from the synchronization model and default database creation options.

Existing SQL Anywhere or UltraLite database

Choose this option if you already have a SQL Anywhere or UltraLite remote database. If the database has no tables, then deployment will add the remote schema from the synchronization model.

5. MobiLink generates another .sql file with the commands to set up the remote database with all schema and synchronization information. The wizard can also create a new database file right now and initialize it. If you choose not to do this, you must generate a new remote database, and then run the .sql file against it. Select both checkboxes, and specify locations for the SQL file and the remote database file. Click Next.

Make a command file and a SQL file with commands to create the database

This will create files that you can run later to create a remote database. You must specify where you want to save the SQL file. The command file will be created in the same folder.

SQL file:

Create a remote SQL Anywhere database

This will create a remote SQL Anywhere database with default database creation options. You must specify where you want to save the main database file. A transaction log file will also be created in the same folder.

SQL Anywhere database file:

6. Specify a MobiLink user name and password. The MobiLink user is stored in the MobiLink server and is used to authenticate the user during synchronization.

What user name do you want to use for connecting to the MobiLink server?

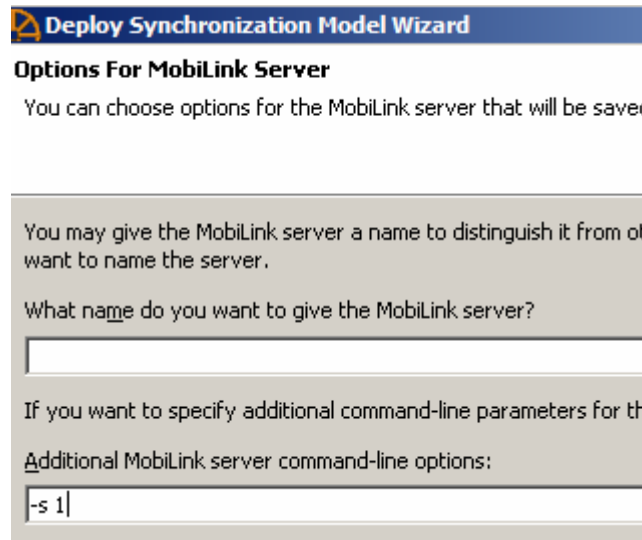
What password do you want to use?

Note: The password is not masked above because it will be stored as plain text in generated cor database if you are using a SQL Anywhere remote database). It can be empty if you do not want protection with default MobiLink authentication.

Register this user in the consolidated database for MobiLink authentication

7. Continue with the defaults until you reach the Options for MobiLink Server page. You will need to specify a server option to preserve compatibility with the Order Entry database.

The sample Oracle consolidated database includes a row-level trigger, `INSERT_ORD_LINE`, which fires before an insert into the `Order_Items` table. The trigger assigns the appropriate `LINE_ITEM_ID`. However, by default, MobiLink synchronizations will send up to 10 row updates, inserts, or deletes in a single batch, causing an error with the row-level trigger. To maintain full compatibility with the existing `INSERT_ORD_LINE` trigger, the MobiLink server needs to be instructed to synchronize a single row update per block. Type `'-s 1'` without the quotes in the command-line options field.



8. At this point, you can complete the wizard by clicking Finish. Your synchronization model will be deployed with the following default options:
 - TCP/IP communication stream.
 - MobiLink server address of `'localhost'`. You will need to change this if your MobiLink server is running on a different machine.
 - No information, except warnings and errors, will be displayed in the MobiLink dialogs. All other messages will be outputted to a log file.

You are finished with the deployment wizard. All your files are initialized and can be found in the folder you specified earlier. Your consolidated database is fully configured for synchronization with many remote clients, and you have successfully deployed one remote client. If you want to deploy other remote clients, you can run this wizard again, making sure to create a new MobiLink user and opt out of deploying the consolidated database and MobiLink server. Since these have already been deployed, all you need to do is deploy other remote synchronization clients.

SYNCHRONIZING

STARTING THE CONSOLIDATED DATABASE

1. Ensure that your Oracle service is running.

The MobiLink server must interact with the consolidated database (*oe*) to synchronize the remote clients.

STARTING THE MOBILINK SERVER

Note

By default, MobiLink uses the snapshot/READ COMMITTED isolation level for upload and download. For the MobiLink server to be able to make the most effective use of snapshot isolation, the Oracle account used by the MobiLink server must have permission for the V_\$TRANSACTION Oracle system view. If it does not, a warning is issued and rows may be missed on download. Only SYS can grant this access. The Oracle syntax for granting this access is:

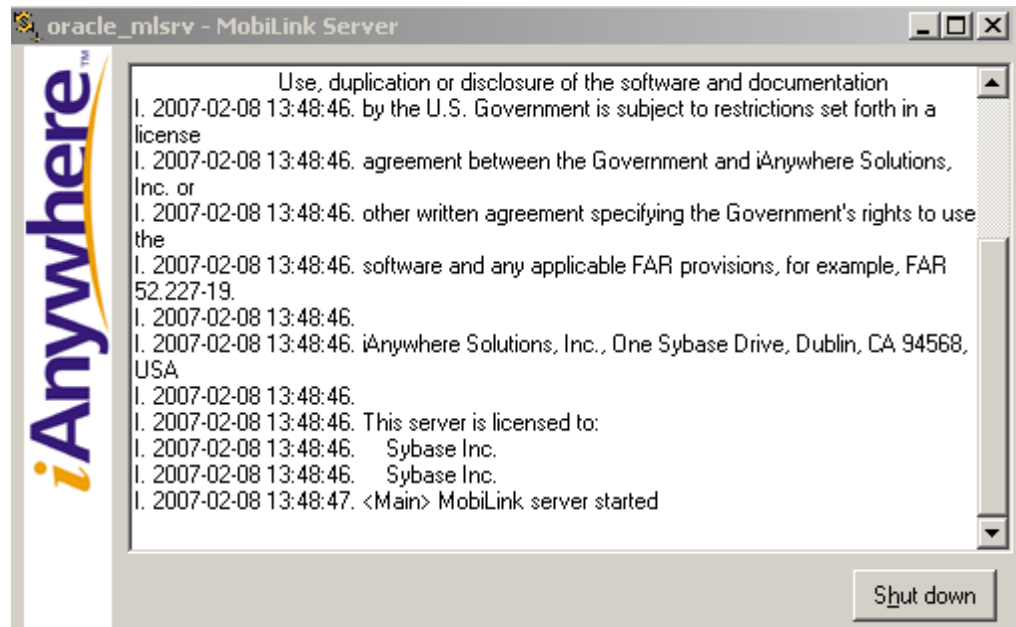
```
GRANT SELECT ON SYS.V_$TRANSACTION TO oe;
```

1. At a command prompt, navigate to the folder where you created the synchronization model. (This is the root directory you chose in the first step of the synchronization model wizard.) If you used the suggested directory names, you should have the following directories located in the root directory: `sync_model_oracle\mlsrv`.
2. Run the following command from the `mlsrv` directory:

```
sync_model_oracle_mlsrv.bat "dsn=oracle_cons;uid=your-oracle_login;pwd=your-oracle-pwd;"
```

- **sync_model_oracle_mlsrv.bat** is the command file created to start the MobiLink server.
- **dsn** is your ODBC data source name.
- **uid** is the user name you use to connect to the consolidated database.
- **pwd** is the password you use to connect to the consolidated database.

If this command runs successfully, the MobiLink server starts and a window similar to the following appears:



If the MobiLink server fails to start, check your consolidated database connection information.

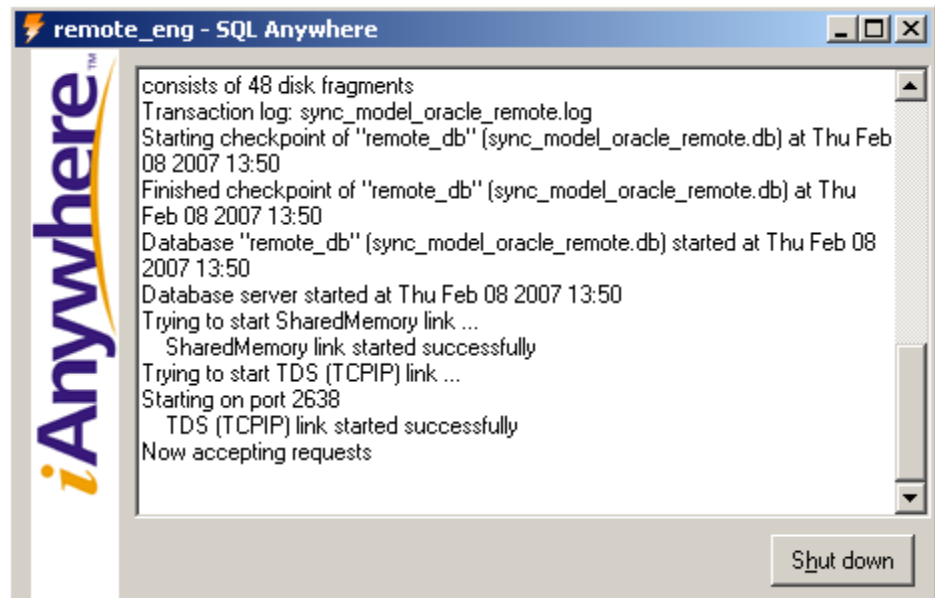
STARTING THE REMOTE DATABASE

1. At a command prompt, navigate to the directory where the deployment wizard created your remote database. If you used the suggested directory names, you should have the following directories located in the root directory: `sync_model_oracle\remote`.
2. Start your remote SQL Anywhere database with the following command:

```
dbeng10 -n remote_eng sync_model_oracle_remote.db -n remote_db
```

- **dbeng10** is the database server used to start the SQL Anywhere database
- **remote_eng** is the database server name.
- **sync_model_oracle_remote.db** is the database file that is started on remote_eng.
- **remote_db** is the name of the database on remote_eng.

If this command runs successfully, a SQL Anywhere database server named remote_eng starts and loads the database called remote_db. You should see a window that is similar to the following:



INITIALIZING THE REMOTE ID

In the remote schema, each remote database represents one salesperson. The synchronization scripts you wrote included logic that instructed the MobiLink server to download a subset of data based on the remote ID of the remote database. You must set the database's remote ID to the value of a valid salesperson identifier.

It is important to complete this step before the first synchronization because when the remote device synchronizes for the first time, it downloads all information related to the chosen salesperson.

1. Choose a valid sales rep identifier.

For a list of valid sales rep identifiers, use the Oracle SQL*Plus application to execute the statement:

```
SELECT COUNT(SALES_REP_ID), SALES_REP_ID
FROM OE.ORDERS GROUP BY SALES_REP_ID;
```

For example:

```
SQL> select count(sales_rep_id), sales_rep_id from oe.orders group by sales_rep_id;
COUNT(SALES_REP_ID) SALES_REP_ID
-----
5 153
0
5 155
13 161
12 163
10 154
7 158
5 156
7 159
6 160

10 rows selected.
```

In our example, we will have the remote database represent the salesperson with a SALES_REP_ID of 154. Therefore, we need to set the database's remote identifier to a value of '154'.

2. At a command prompt, execute the following command:

```
dbisql -c "eng=remote_eng;dbn=remote_db;uid=DBA;pwd=sql" "SET OPTION
PUBLIC.ml_remote_id='154'"
```

- **dbisql** is the application used to execute SQL commands against a SQL Anywhere database.
- **eng** sets the database server name to remote_eng.
- **dbn** sets the database name to remote_db.
- **uid** sets the user name DBA used to connect to your remote database.
- **pwd** sets the password sql used to connect to your remote database.
- **SET OPTION PUBLIC.ml_remote_id='154'** is the SQL command used to set the remote ID to a value of 154.

SYNCHRONIZING THE REMOTE CLIENT

Now you are ready to synchronize the remote client for the first time. This is done with the MobiLink client program dbmlsync. Dbmlsync connects to the remote database, authenticates itself with the MobiLink server, and performs all of the uploads and downloads necessary to synchronize the remote and consolidated databases based on a publication in the remote database.

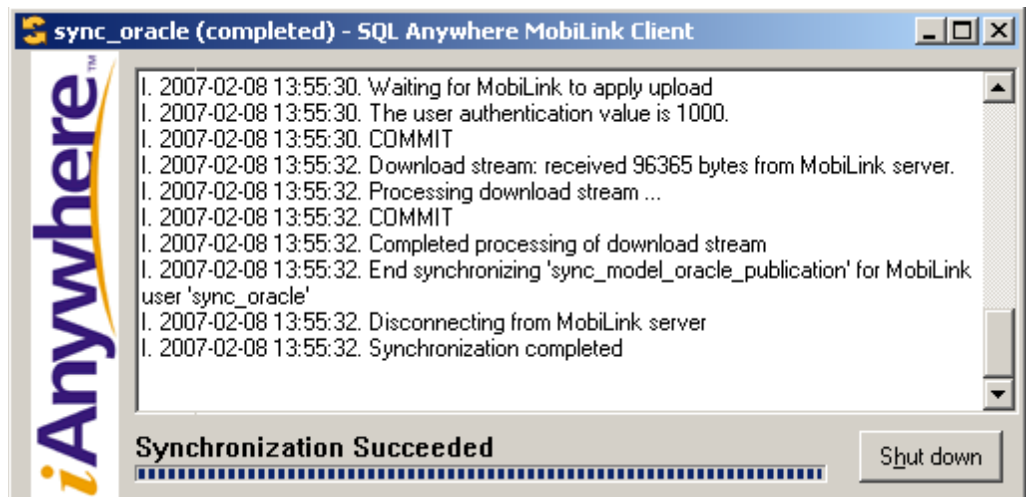
1. At a command prompt, execute the following command:

```
dbmlsync -c "eng=remote_eng;dbn=remote_db;uid=DBA;pwd=sql;" -n  
sync_model_oracle_publication -u sync_mluser -mp pass_mluser
```

- **dbmlsync** is the synchronization application.
- **eng=remote_eng** specifies the name of the remote database server.
- **dbn=remote_db** specifies the name of the remote database.
- **uid** is the user name used to connect to the remote database.
- **pwd** is the password used to connect to the remote database.
- **sync_model_oracle_publication** is the publication on the remote device that will be used to perform the synchronization.
- **sync_mluser** is the user name used to authenticate with the MobiLink server.
- **pass_mluser** is the password used to authenticate with the MobiLink server.

If you are running the dbmlsync application on a different computer from your MobiLink server, you must pass in arguments that specify the location of the MobiLink server.

2. If this command runs successfully, the dbmlsync application populates the remote database with a subset of information from the consolidated database. You should see a screen similar to the following:



If synchronization fails, you can start by checking the connection information you pass to the dbmlsync application, as well as the MobiLink user name and password. Failing that, check the publication name you used, and ensure the consolidated database and MobiLink server are running. You can also examine the contents of the synchronization logs (server and client).

VIEWING REMOTE CLIENT DATA IN SYBASE CENTRAL

After successfully synchronizing the remote client to the consolidated database through the MobiLink server, the remote data should be populated with information relevant to one sales rep. You can verify this in Sybase Central using the SQL Anywhere 10 plug-in.

1. Open Sybase Central and use SQL Anywhere 10 to make a new connection.

If you haven't stopped the databases that were started in the above sections, your connection information should look like this:

- **User name** DBA
- **Password** sql
- **Server name** remote_eng
- **Database name** remote_db

Once you are connected, you can navigate to the remote tables to view your data.

2. Choose any table and then click the Data tab. The ORDERS table should look like the following:

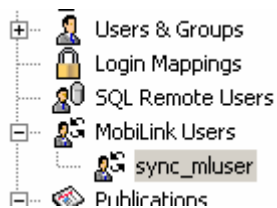
| | ORDER_ID | ORDER_DATE | ORDER_MODE | CUSTOMER_ID | SALES_REP_ID |
|----|----------|-------------------------|------------|-------------|--------------|
| 1 | 2397 | 1999-11-19 17:41:54.696 | direct | 102 | 154 |
| 2 | 2402 | 1999-07-02 04:34:44.665 | direct | 161 | 154 |
| 3 | 2403 | 1999-07-01 17:49:13.615 | direct | 162 | 154 |
| 4 | 2409 | 1999-06-29 10:53:41.984 | direct | 167 | 154 |
| 5 | 2429 | 1999-11-10 06:49:25.526 | direct | 117 | 154 |
| 6 | 2438 | 1999-09-01 10:53:26.934 | direct | 104 | 154 |
| 7 | 2442 | 1990-07-27 13:22:59.662 | direct | 107 | 154 |
| 8 | 2443 | 1990-07-27 14:34:16.562 | direct | 108 | 154 |
| 9 | 2451 | 1999-12-17 20:03:52.562 | direct | 148 | 154 |
| 10 | 2454 | 1999-10-02 18:49:34.678 | direct | 103 | 154 |

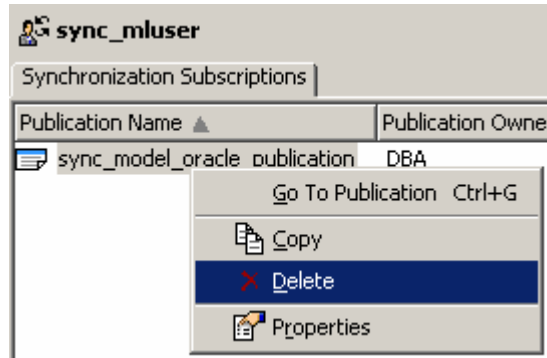
Notice that all the records in this table are for a sales rep with an identifier of 154. This particular salesperson is not concerned with the sales information of other representatives. Now this representative's database takes up less space, and requires less time to synchronize. Since the remote database size is kept to a minimum, frequently performed operations, such as entering a new order, can run to completion faster and more efficiently.

UPDATING THE REMOTE FOR NON-QUERY ACTIONS

Now that the required data from the remote database has been populated with the appropriate data, some changes to the schema are necessary to enable non-query actions, such as INSERT, UPDATE and DELETE to work as expected.

The remote database schema cannot be modified while it is a part of a synchronization definition. Therefore, the first step is to delete the synchronization publication subscription for the remote database. This can be done either through SQL commands or graphically, using Sybase Central.



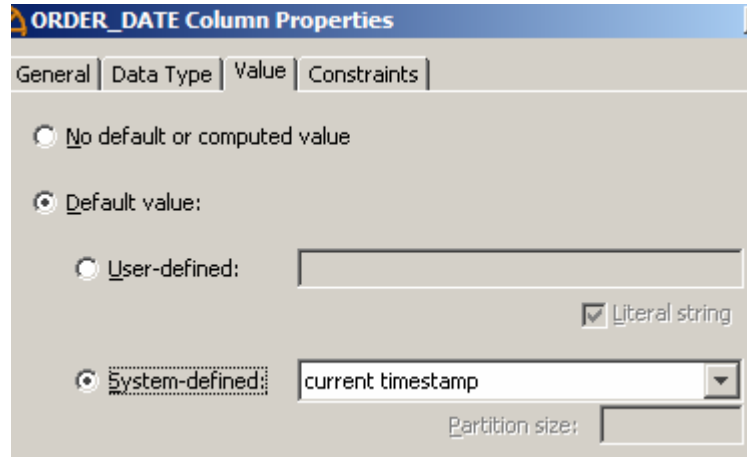


Oracle Database 10g uses the 'Numeric' data type to identify all numeric data. The MobiLink synchronization wizard converts these numeric data types into the 'Decimal' data type, which preserves the size and precision of the Numeric type. However, for certain behaviors in SQL Anywhere, such as autoincrement which requires an integer data type, this will not be acceptable. Therefore, it is necessary to change some of the column types from their default type. The following column types should be changed from Decimal to Integer, both for functionality as well as clarity.

- Customers.CUSTOMER_ID
- Order_Items.ORDER_ID
- Order_Items.LINE_ITEM_ID
- Order_Items.PRODUCT_ID
- Order_Items.QUANTITY
- Orders.ORDER_ID
- Orders.CUSTOMER_ID
- Orders.ORDER_STATUS
- Orders.SALES_REP_ID
- Orders.PROMOTION_ID
- Product_Information.PRODUCT_ID

| ORDER_ITEMS (OE) | | | | |
|------------------|-------------------------------------|--------------|------|-----------|
| Columns | | | | |
| | PKey | Name | ID ▲ | Data Type |
| 1 | <input checked="" type="checkbox"/> | ORDER_ID | 1 | integer |
| 2 | <input checked="" type="checkbox"/> | LINE_ITEM_ID | 2 | decimal |
| 3 | <input type="checkbox"/> | PRODUCT_ID | 3 | double |
| 4 | <input type="checkbox"/> | UNIT_PRICE | 4 | float |
| 5 | <input type="checkbox"/> | QUANTITY | 5 | integer |

To automatically insert the correct time stamp with a new order, the ORDERS.ORDER_DATE column's value should be changed to the current timestamp value.



The next step is to take advantage of SQL Anywhere's built in functionality to ensure primary key uniqueness. The changes to the column data types allow you to do this.

ENSURING UNIQUE PRIMARY KEYS ACROSS REMOTE DATABASES

One of the primary fundamentals of distributed database design is how to maintain unique primary keys. There are several ways to achieve unique keys across remotes and SQL Anywhere offers global autoincrementing fields as one possibility. This approach, as well as some others, is discussed in detail in our documentation. Please refer to the 'Synchronization Techniques' chapter located in the 'MobiLink – Server Administration' book as a starting point for a general understanding of synchronization techniques. Our documentation can be accessed online from the following URL:

http://www.iAnywhere.com/developer/product_manuals/sqlanywhere/

For a more detailed look at how to maintain a large number of remotes with unique keys and how to easily deploy new remotes against an Oracle consolidated database (and others), please consult the following whitepaper:

<http://www.sybase.com/detail?id=1055953>

Here is a summary of the contents of the above whitepaper:

- An introduction to the workings of global autoincrement
- How to setup Oracle sequences and make them emulate global autoincrementing fields
- How to include global autoincrementing fields in remote schema definitions
- How to quickly deploy new remotes with unique keys

In the 'Primary Keys in a Distributed Database Environment' whitepaper above, a general overview of maintaining primary keys across the consolidated Oracle and remote databases was presented. For the purposes of the Order Entry demo application, a more specific implementation of global autoincrement on the remote database follows, to give a concrete use-case.

The consolidated Order Entry database includes an existing sequence for the Orders.ORDER_ID field. In order to give the remote databases some room for their primary keys, reduce the maximum value of the sequence to something more manageable, such as 1,000,000. For the purpose of the

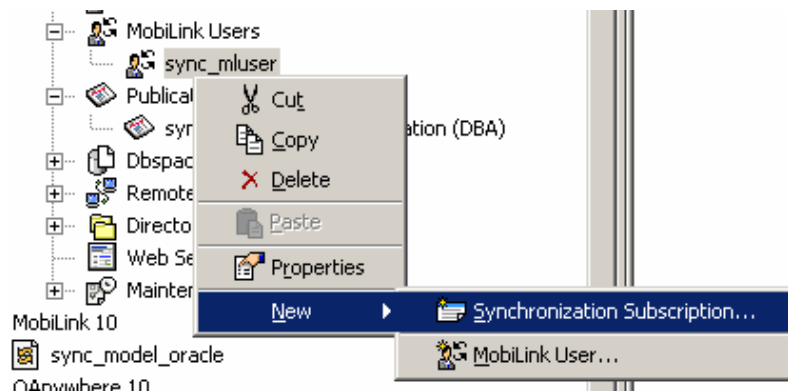
demo, assume the remotes may enter up to 100,000 orders. This means with a global autoincrement partition size of 100,000, the smallest database identifier that will not interfere with the sequence on the consolidated is 10. The consolidated can now enter up to 1,000,000 orders and the first remote database, with database identifier 10, can enter orders from 1,000,001 to 1,100,000.

Of course it is always preferable to create partition sizes that are too large as opposed to too small on both the consolidated and remote databases. Keep the level of potential data growth in mind when creating the partition sizes. The values used in this example were for simplicity and may not be appropriate for other databases and circumstances.

RECREATING THE SYNCHRONIZATION SUBSCRIPTION

The next step is to recreate the subscription publication that was deleted prior to modifying the database schema. Again, this can be accomplished by directly executing SQL against the database or using Sybase Central. The following steps are for Sybase Central.

1. In Sybase Central, navigate to MobiLink Users and right click on sync_mluser. Create a new synchronization subscription.



2. Select the desired publication and click Finish to add a subscription.

A new subscription has now been created for the MobiLink user. Types have been updated and SQL Anywhere's global autoincrement feature has been enabled to ensure primary key uniqueness.

The above three sections detailing changes to the remote database schema have been rolled into a single SQL script which can be executed against the remote database. This is an alternative method, if the user does not wish to make changes through Sybase Central.

```

/* Set the remote database ID to 10 */

SET OPTION "PUBLIC"."global_database_id" = '10';

/* Drop the subscription to sync_model_oracle_publication so schema changes
can be made */

IF EXISTS (
    SELECT 1
    FROM SYS.SYSSYNCSUBSCRIPTIONS

```

```

        WHERE site_name = 'sync_mluser' AND
publication_name='sync_model_oracle_publication'
) THEN
    DROP SYNCHRONIZATION SUBSCRIPTION TO "sync_model_oracle_publication"
FOR "sync_mluser";
END IF;

```

```

/*

```

The following are table columns which are to have their data type changed from the default decimal to integer.

```

Customers.CUSTOMER_ID
Order_Items.ORDER_ID
Order_Items.LINE_ITEM_ID
Order_Items.PRODUCT_ID
Order_Items.QUANTITY
Orders.ORDER_ID
Orders.CUSTOMER_ID
Orders.ORDER_STATUS
Orders.SALES_REP_ID
Orders.PROMOTION_ID
Product_Information.PRODUCT_ID

```

```

*/

```

```

ALTER TABLE "OE"."CUSTOMERS" DROP PRIMARY KEY;
ALTER TABLE "OE"."CUSTOMERS" ALTER "CUSTOMER_ID" integer;
ALTER TABLE "OE"."CUSTOMERS" ADD PRIMARY KEY ( "CUSTOMER_ID" ASC );
ALTER PRIMARY KEY ON "OE"."CUSTOMERS" RENAME TO "CUSTOMERS";

```

```

ALTER TABLE "OE"."ORDER_ITEMS" DROP PRIMARY KEY;
ALTER TABLE "OE"."ORDER_ITEMS" ALTER "ORDER_ID" integer;
ALTER TABLE "OE"."ORDER_ITEMS" ALTER "LINE_ITEM_ID" integer;
ALTER TABLE "OE"."ORDER_ITEMS" ADD PRIMARY KEY ( "ORDER_ID" ASC,
"LINE_ITEM_ID" ASC );
ALTER TABLE "OE"."ORDER_ITEMS" ALTER "PRODUCT_ID" integer;
ALTER TABLE "OE"."ORDER_ITEMS" ALTER "QUANTITY" integer;
ALTER PRIMARY KEY ON "OE"."ORDER_ITEMS" RENAME TO "ORDER_ITEMS";

```

```

ALTER TABLE "OE"."ORDERS" DROP PRIMARY KEY;
ALTER TABLE "OE"."ORDERS" ALTER "ORDER_ID" integer;
ALTER TABLE "OE"."ORDERS" ADD PRIMARY KEY ( "ORDER_ID" ASC );
ALTER TABLE "OE"."ORDERS" ALTER "ORDER_ID" SET DEFAULT global autoincrement(
100000 );
ALTER TABLE "OE"."ORDERS" ALTER "ORDER_DATE" SET DEFAULT current timestamp;
ALTER TABLE "OE"."ORDERS" ALTER "CUSTOMER_ID" integer;
ALTER TABLE "OE"."ORDERS" ALTER "ORDER_STATUS" integer;
ALTER TABLE "OE"."ORDERS" ALTER "SALES_REP_ID" integer;
ALTER TABLE "OE"."ORDERS" ALTER "PROMOTION_ID" integer;
ALTER PRIMARY KEY ON "OE"."ORDERS" RENAME TO "ORDERS";

```

```

ALTER TABLE "OE"."PRODUCT_INFORMATION" DROP PRIMARY KEY;
ALTER TABLE "OE"."PRODUCT_INFORMATION" ALTER "PRODUCT_ID" integer;
ALTER TABLE "OE"."PRODUCT_INFORMATION" ADD PRIMARY KEY ( "PRODUCT_ID" ASC );

```

```
ALTER PRIMARY KEY ON "OE"."PRODUCT_INFORMATION" RENAME TO
"PRODUCT_INFORMATION";
```

```
/* Recreate the subscription to the synchronization publication once the
schema changes have been made */
```

```
IF NOT EXISTS (
    SELECT 1
    FROM SYS.SYSSYNCSUBSCRIPTIONS
    WHERE site_name = 'sync_mluser' AND
publication_name='sync_model_oracle_publication'
) THEN
    CREATE SYNCHRONIZATION SUBSCRIPTION TO
"sync_model_oracle_publication" FOR "sync_mluser"
        TYPE tcpip ADDRESS 'host=localhost;port=2439;' OPTIONS
ScriptVersion='sync_model_oracle_script',MobiLinkPwd='pass_mluser';
END IF;

COMMIT;
```

CONCLUSION

Upon completion of this demonstration, you have successfully mobilized an Oracle database. It is now set up with one remote synchronization client, and is capable of deploying many more with little time or effort.

To summarize, you completed the following tasks to configure the synchronization environment:

- Designed a remote schema based on a subset of the consolidated schema. You did this by choosing each remote to represent a particular salesperson. The remote database was given a remote ID equal to that rep's identifier.
- Prepared the consolidated database.
- Ensured that each synchronized table had a primary key.
- Created an ODBC data source. The ODBC data source pointed to the Oracle database so that MobiLink could access it.
- Made the necessary modifications to the consolidated database to make it compatible with MobiLink and the SQL Anywhere remote database.
- Created a synchronization model.
- Defined how data was uploaded and downloaded to and from the consolidated database.
- Deployed the synchronization model.
- Set up the MobiLink server and server options to account for consolidated database requirements.
- Created a remote database.
- Generated command files to allow for simple initialization.
- Started the remote database.
- Started a new SQL Anywhere database server running the remote database.
- Set the remote database's remote ID.
- Synchronized the remote database to the consolidated database.
- Viewed data in the remote database.

- Verified that data was in fact a subset of the consolidated database, all pertinent to the particular salesperson whose rep identifier you used.
- Removed the publication subscription to allow modifications to the remote database schema
- Made modifications to the remote database to ensure compatibility with consolidated database when inserting, updating or deleting rows.
- Recreated the publication subscription

COPYRIGHT © 2007 IANYWHERE SOLUTIONS, INC. ALL RIGHTS RESERVED. SYBASE, AFARIA, SQL ANYWHERE, ADAPTIVE SERVER ANYWHERE, MOBILINK, ULTRALITE, AND M-BUSINESS ANYWHERE ARE TRADEMARKS OF SYBASE, INC. ALL OTHER TRADEMARKS ARE PROPERTY OF THEIR RESPECTIVE OWNERS.