# Mimer SQL

# Linux Mobile Guide

Version 10.0

Mimer SQL, Linux Mobile Guide, Version 10.0, January 2011


Produced and published by Mimer Information Technology AB, Uppsala, Sweden.
P.O. Box 1713,
SE-751 47 Uppsala, Sweden.
Tel +46(0)18-780 92 00.
Fax +46(0)18-780 92 40.

Mimer SQL Web Sites:
http://developer.mimer.com
http://www.mimer.com

# Contents

# Chapter 1

# Introduction

Welcome to use Mimer SQL Mobile for Linux.

Mimer SQL is a Relational Database management System (RDBMS). This document describes how the Mimer SQL Mobile concept is used on Linux.

Mimer SQL can in general be described as self-tuned and powerful, and yet a very compact, database server. It possesses an extensive set of database functionality and is built on international SQL standards. With the Mimer SQL Mobile concept all functionality is remained intact while the runtime footprint is decreased. This is achieved by avoiding the runtime SQL compiling step using precompiled statements. A statement in this sense is a database object holding a predefined SQL statement. The statement can then be executed on the device. These statements can be static or dynamic. In the dynamic case, variables are used to couple the statement to the application. The database, including the statements, is then exported to the target device.

Some important aspects for the Mimer SQL Mobile product are the following:

**Small -** The Mimer SQL Mobile database product is optimized to be as small as possible. The database server also automatically compresses data on-the fly and savings of disk space are typically around 60%.

**Self-tuning -** Mimer SQL's self-tuning mechanisms reduce the number of configuration parameters needed and make your database virtually maintenance free. These mechanisms include automatic reorganization of your database that dramatically improves database query performance.

**Full functionality -** Mimer SQL Mobile supports transactions, stored procedures, triggers, referential integrity and multi-user concurrency. The complete support for SQL separates Mimer SQL Mobile from other small footprint databases on the market and enables you to reuse existing SQL knowledge. In addition, full Unicode 4 multi lingual support is provided.

**Standards -** Mimer SQL Mobile conforms to all major SQL standards.

**Fail-safe -** As Mimer SQL Mobile supports transactions, which means that your database is always consistent. Not even a power failure at a critical stage will damage your database. Once it is up and running again, your database automatically recovers to a consistent state.

## Understanding the Concept

The following steps should be carried out to get a Mimer SQL Mobile database up and running on Linux. The steps forms an overview of the concept and the involved components that will be described further later on in this document.

### 1 Install Mimer SQL Mobile for Linux

Unpack the delivered tar file and execute the miminstall command, or, install the distributed rpm file using the rpm installation procedures.

### 2 Create a database for development

Run the dbinstall program to create a Mimer SQL database. This is an ordinary Mimer SQL database, equal to a database created with the Mimer SQL Engine product.

### 3 Create and store precompiled statements

Create all the database objects for the application. Use the CREATE STATEMENT command to create and store the statements you need for your application logic. This enables you to access the database when it is exported for use in a minimized target environment. This is described further in the next section.

### 4 Verify the application in development environment

Verify the interaction between the application and the database. You can switch the ServerType configuration parameter for the database to indicate that you will enable the Mobile execution mode.

### 5 Export the database, including created statements

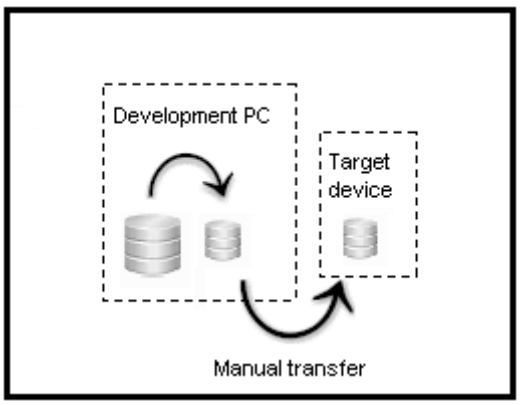
Use the mimpdacopy command to prepare and minimize the database for use on the target environment. The created database environment can be used immediately on the development x86 Linux computer, just like any other Mimer SQL database. Please note that execution in Engine mode is not possible for this database.

### 6 Install Mimer SQL Mobile software package for the target device

Install software on the target device. This device specific package is delivered within the Mimer SQL Mobile distribution package or, in some cases, as a separate archive.

### 7 Move exported databank files in place

To execute on a target environment, the exported databank files are moved manually to the device. See the following picture:




### 8 Start the Mimer SQL database server on the device

Let the Mimer SQL Mobile database server program operate the installed databank files.

### 9 Verify the application on the target platform

Verify the behavior on the target device.

# About Precompiled Statements

The Mimer SQL Mobile database server that runs on your target device has a small footprint, but yet all the powerful database functionality is enabled.

The difference between using a standard enterprise Mimer SQL Engine is that you need to create predefined statements when operating a Mimer SQL database server in Mobile mode. Such statements are created and stored in your Mimer SQL Engine development environment database on your desktop PC. When the database is exported, these statements give you access to the database on your device.

For example, to store the statement: `SELECT col1, col2 FROM tab`, you would use the SQL command called `CREATE STATEMENT` as follows:

```
CREATE STATEMENT select_stm SELECT col1, col2 FROM tab
```

The statement, named `select_stm`, is stored in the database. When you run your application it executes the following statement:

```
EXECUTE STATEMENT select_stm
```

(Please note that the `EXECUTE STATEMENT` part of the command is optional).

The same method is used for all types of data manipulation statements and calls to stored procedures.

The following are examples of precompiled SQL statements that show how to insert and update data, using both constants and parameter markers, and also how to call procedures:

```
CREATE STATEMENT insert_stm INSERT INTO tab(col1, col2)
VALUES('ABC', :col2_value);

CREATE STATEMENT update_stm UPDATE tab
SET col2=col2 + 10 WHERE col1 = :hw_col1;

CREATE STATEMENT call_stm CALL my_proc('ABC', 10, 20);
```

The statement named `insert_stm` inserts the string 'ABC' into col1. The second parameter, :col2_value, is a parameter marker and is represented by a host variable in an application program.

The second statement, `update_stm` increases col2 by 10 wherever col1 matches the value retrieved using the parameter marker named :hw_col1.

The final statement, `call_stm` invokes the procedure named `my_proc` with 3 parameters.

From the application, a pre-compiled statement is executed just like any other SQL statement, using parameter markers to bind to application variables. As an example, from ODBC this is done with `SQLPrepare`/`SQLExecute` or `SQLExecDirect` and `SQLBindParameter`.

# Data Model Example With Statements

To fully understand the concept of Mimer SQL Mobile, the steps when developing the database environment for an example application will be described a little more in detail below.

We presume that a database is available, i.e. that **dbinstall** has been executed successfully. Next, we need a so called *user databank*. This will be the databank used to store the database objects that we will need for our project.

```
CREATE DATABANK db;
```

Let us imagine a simple GUI for the example application. It shows a selection list where a name can be chosen and the corresponding address is displayed. We must have a data model for the application. It is formed on two tables where one is storing names and the other is storing addresses. The tables are coupled together using the foreign key clause. The following SQL statements create the example tables:

```
CREATE TABLE person(
    pnr INT,
    name NCHAR VARYING(50),
    surname NCHAR VARYING(50),
    PRIMARY KEY(pnr));

CREATE SEQUENCE address_id_seq;

CREATE TABLE person_address(
    address_id INT DEFAULT NEXT_VALUE OF address_id_seq,
    pnr INT,
    phone NCHAR(10),
    zipcode NCHAR(10),
    address NCHAR VARYING(100),
    city NCHAR VARYING(50),
    PRIMARY KEY(address_id),
    FOREIGN KEY(pnr) REFERENCES person(pnr));
```

In addition, we need some statements to be used on the device. As an example, the following statements are created to insert persons and addresses and to retrieve the inserted data:

```
CREATE STATEMENT select_persons
    SELECT pnr, name, surname
    FROM person;

CREATE STATEMENT select_address
    SELECT address_id, phone, zipcode, address, city
    FROM person_address
    WHERE pnr=:pnr_value;

CREATE STATEMENT insert_person
   INSERT INTO person (pnr, name, surname)
   VALUES (:pnr, :name, :surname);

CREATE STATEMENT insert_address
    INSERT INTO person_address (pnr, phone, zipcode, address, city)
    VALUES (:pnr, :phone, :zipcode, :address, :city);
```

Now, the database objects needed for our example are completed, and it is possible to export the complete development database environment. When exported and installed on the device the statements created will be accessible immediately, providing database table insert and retrieval methods.

# What Happens When Exporting a Database?

When exporting and packaging Mimer databanks for deployment in a target environment, unnecessary system views, collations and system tables are removed.

The views eligible for removal are those in the INFORMATION_SCHEMA and MIMER schemas. Note that views are only removed if they are not accessed or required by statements created before deployment. Similarly, collations not used by the application schema are also removed. If you are familiar with Java programming, this operation resembles the obfuscation phase MIDP Java programmers frequently execute in the sense that unused code is omitted from the deployed product.

The views in the FIPS_DOCUMENTATION and INFO_SCHEM schemas are deprecated and they are never accessible on the mobile device.

The export is performed using the **mimpdacopy** command. When the operation is completed, a set of minimized databank files is created. In addition, and mainly for test purposes, necessary steps are taken to prepare a ready-to-start Mimer SQL Mobile database, with these databanks, on the current desktop development PC running Linux/x86.

For use on the target environment the exported databank files are manually copied there.

Please note that Mimer SQL Mobile for Linux only creates databases that can be used on backwards byte order (little endian) target environments running Mimer SQL v10.0.

# Chapter 2

# Getting Started on the Desktop

## Installation and Package Content

The distribution package Mimer SQL Mobile for Linux is installed as any Mimer SQL product for Linux, i.e. by using tar and the miminstall command, or by using RPM.

In fact, the product is an ordinary Mimer SQL Engine product with some additional tools and libraries specific for the Mimer SQL Mobile environment.

The following are the additional Mimer SQL Mobile components in the installation:

| | |
|---|---|
| `dbserver_mob` | Database server program for execution in Mobile mode (for Linux x86). Having this server program makes it easier to test and evaluate a Mimer SQL Mobile database configuration on the desktop PC. |
| `libminodbc.so` | Shared library holding a limited ODBC database API. |
| `midjdbc2.jar` | CLDC/MIDP database API for Java. |
| `mimexport` | Program used to remove (or add) data dictionary components that are not needed in a mobile runtime environment. Used by the mimpdacopy program. |
| `mimpdacopy` | Program used to perform all steps necessary for creating a mobile database from an existing standard Mimer SQL database. |
| `minjdbc3.jar` | CDC/FP database API for Java. |
| `minodbc.h` | Header file to be used when programming towards the limited ODBC database API (libminodbc.so). |

**Note:** In addition, as described in the next chapter, you will need a software package specifically ported for the target environment. These customized packages can be delivered within the Mimer SQL Mobile for Linux product, or it can be delivered separately

# Development Environment

As mentioned, a Mimer SQL Mobile product includes an environment that should be used as the development environment. This is where you should develop your Mimer SQL database and the database applications for mobile devices. These components are then transferred to the target environment.

Usually the developed application can be verified in a simulation environment on the workstation before it is used on the target hardware device.

In the development environment all the database objects are created, including the statements used to access data when operating on the device, as described earlier.

## Create a Database

Once you have installed Mimer SQL Mobile on your workstation, you are ready to create the database you will export to and use on your device.

Use the **dbinstall** command to create your initial database.

When successfully completed you have a Mimer SQL database server running on your Linux computer.

In the example environment, if you decided to install it during the dbinstall session, there are a number of precompiled statements ready to study and execute. For example, the DbVisualizer tool can be used to review these statements. Use the MIMER_STORE user ident with password *GoodiesRUs*. The **mimexampledb** command can be used if the option was omitted.

Another option provided by the dbinstall command gives the possibility to create an "initial example development environment" which can serve our purposes here as a starting point when creating an application. A database ident named DEVUSER with password 'devuser' is created. The mimdevenv command can be used if the option was omitted.

## Export the Databank Files

When you have prepared your database, it is time to export the database for use on a target device or in an emulation environment. At this stage you have created the database environment with database objects and initial data, including precompiled statements to access the database.

The export operation is performed using the mimpdacopy command.

### mimpdacopy command

The mimpdacopy command usage is as follows:

```
Usage: mimpdacopy -t type [-s|-m][-o output_directory] [-p sysadm_password]
[-n] [database]
Options: -n                    No output to screen.
         -o output_directory   Directory to store converted databanks.
         -p sysadm_password    SYSADM password.
         -m                    multi-user mode.
         -s                    single-user mode.
         -t type               Target server type, 'mobile' or 'micro'.
Info: Used to prepare and minimize a database for usage on a mobile target.
```

If the `-o` option is omitted a new directory is created directly beneath the home directory of the target database.

If the `-p` option is omitted the SYSADM password will be asked for at a command prompt.

If the database argument is omitted the database defined in the MIMER_DATABASE environment variable will be used (or, default in the sqlhosts file).

In the following example the database called `mimdb` is exported using a default scenario. The home directory for the mimdb database is `/usr/local/MimerSQL/mimdb`. To be able to compare databank file sizes, there is a listing of the original databank files made in the beginning, and in the end of the session a corresponding list of the exported files is shown:

```
# ls -l /usr/local/MimerSQL/mimdb
totalt 16072
-rw-------  1 root root 2048000 15 maj 16.05 db.dbf
-rw-------  1 root root 2048000 15 maj 16.05 logdb.dbf
-rw-------  1 root root     556 14 maj 17.31 mimer.log
-rw-r--r--  1 root root    1653 14 maj 09.58 multidefs
-rw-------  1 root root 2048000 14 maj 11.05 sqldb.dbf
-rw-------  1 root root 8192000 15 maj 16.05 sysdb100.dbf
-rw-------  1 root root 2048000 15 maj 16.05 transdb.dbf
# mimpdacopy -t mobile mimdb
Locating output directory ... done
Verifying output directory ... done
Creating output directory ... done
* Output directory: /usr/local/MimerSQL/mimdb/mimdb_mobile
* Database: mimdb (Running)
* Target: mobile
* Target database: mimdb_mobile
* Mode: MULTI
SYSADM password: ?????
Locating databank files ... done
Registering mobile database mimdb_mobile ... done
Doing backup for export ... done
Exporting database ... done
Doing backup for export to minimize ... done
Moving exported databanks in place ... done
Creating multidefs file for mobile ... done
Operation completed!
Exported databanks located in: /usr/local/MimerSQL/mimdb/mimdb_mobile
# ls -l /usr/local/MimerSQL/mimdb/mimdb_mobile
totalt 920
-rw-------  1 root root  18432 15 maj 16.05 db.dbf
-rw-------  1 root root   6144 15 maj 16.05 logdb.dbf
-rw-r--r--  1 root root   1653 15 maj 16.05 multidefs
-rw-------  1 root root   4096 15 maj 16.05 sqldb.dbf
-rw-------  1 root root 870400 15 maj 16.05 sysdb100.dbf
-rw-------  1 root root   4096 15 maj 16.05 transdb.dbf
#
```

## Starting a Mobile Database Server on x86

As can be seen in the example above, a sub directory named `mimdb_mobile` is created beneath the origin database home directory. This directory holds the exported databank files. In addition, a corresponding multidefs file is created where the ServerType automatically is set to Mobile. In fact, the new database is registered and ready to start using the mimcontrol command:

```
# mimcontrol -s mimdb_mobile
2009-05-15 16:28:49.78   <Information>
====================================
Mimer SQL 10.0.5E Beta Test May  8 2009
Mimer SQL server for database MIMDB_MOBILE STARTED at /usr/local/MimerSQL/
mimdb/mimdb_mobile

#
```

The database mimdb_mobile is now up and running and ready to access. When listing processes you can see that the database server program operating this database is called dbserver_mob. As you may know the corresponding program for an ordinary Engine server is dbserver.

## Import Using the mimexport Command

It should be mentioned that it is possible to recreate all meta data for an exported mobile database so that it can be fully used within the development environment again. This is achieved by using the --import option to the mimexport command:

```
# mimexport -?
Usage: mimexport -e | -i [-s] -t type [database]
       -e, --export            Export to target environment
       -i, --import            Import from target environment
       -s, --silent            Silent operation
       -t type, --target=type  Target environment type, 'mobile' or 'micro'
       database                Database to operate on (if omitted current
directory is assumed)
#
```

# Executing Example on x86

By using the mobile server in the development environment (Linux, x86) you can easily verify that your applications work well with a Mimer SQL Mobile database server.

Please note that when accessing a database operated by the dbserver_mob database server program, only prepared statements can be used.

In the following session we use the example described earlier in this document. The session is a complete example covering the creation of database objects, including statements, simple test execution in development database, export of databank files and execution on exported mobile database.

# Create Databank, Tables and Statements

```
# bsql mimdb

Mimer SQL Command Line Utility  Version 10.0.5E Beta Test
Copyright (C) Mimer Information Technology AB. All rights reserved.

Username: SYSADM
Password:
SQL>create databank db of 1000 pages;
SQL>CREATE TABLE person(
SQL&    pnr INT,
SQL&    name NCHAR VARYING(50),
SQL&    surname NCHAR VARYING(50),
SQL&    PRIMARY KEY(pnr));
SQL>CREATE SEQUENCE address_id_seq;
SQL>
SQL>CREATE TABLE person_address(
SQL&    address_id INT DEFAULT NEXT_VALUE OF address_id_seq,
SQL&    pnr INT,
SQL&    phone NCHAR(10),
SQL&    zipcode NCHAR(10),
SQL&    address NCHAR VARYING(100),
SQL&    city NCHAR VARYING(50),
SQL&    PRIMARY KEY(address_id),
SQL&    FOREIGN KEY(pnr) REFERENCES person(pnr));
SQL>
SQL>CREATE STATEMENT select_persons
SQL&    SELECT pnr, name, surname
SQL&    FROM person;
SQL>
SQL>CREATE STATEMENT select_address
SQL&    SELECT address_id, phone, zipcode, address, city
SQL&    FROM person_address
SQL&    WHERE pnr=:pnr_value;
SQL>
SQL>CREATE STATEMENT insert_person
SQL&    INSERT INTO person (pnr, name, surname)
SQL&    VALUES (:pnr, :name, :surname);
SQL>
SQL>CREATE STATEMENT insert_address
SQL&    INSERT INTO person_address (pnr, phone, zipcode, address, city)
SQL&    VALUES (:pnr, :phone, :zipcode, :address, :city);
SQL>
```

## Execute Statements on Development Database

```
SQL>insert_person;
PNR:1234567890
NAME:Alan
SURNAME:Pearson
SQL>insert_address;
PNR:1234567890
PHONE:0812345678
ZIPCODE:17744
ADDRESS:Main Street 10
CITY:Trenton
SQL>select_persons;
        PNR NAME
SURNAME
================================================
 1234567890 Alan
Pearson
===

                  1 row found

SQL>select_address;
PNR_VALUE:1234567890
 ADDRESS_ID PHONE      ZIPCODE
ADDRESS
CITY
================================================================================
          1 0812345678 17744
Main Street 10
Trenton
===

                  1 row found

SQL>exit;
```

## Export Development Database

```
# mimpdacopy -t mobile mimdb
Locating output directory ... done
Verifying output directory ... done
Saving existing output directory ... done
Creating output directory ... done
* Output directory: /usr/local/MimerSQL/mimdb/mimdb_mobile
* Database: mimdb (Running)
* Target: mobile
* Target database: mimdb_mobile
* Mode: MULTI
SYSADM password:
Locating databank files ... done
Doing backup for export ... done
Exporting database ... done
Doing backup for export to minimize ... done
Moving exported databanks in place ... done
Creating multidefs file for mobile ... done
Operation completed!
Exported databanks located in: /usr/local/MimerSQL/mimdb/mimdb_mobile
```

The databank files created here are those that should be used when executing in mobile mode on the target device. They should be manually moved there.

# Start Mobile Database and Execute Statements

```
# mimcontrol -s mimdb_mobile
2009-05-18 14:36:38.10   <Information>
====================================
Mimer SQL 10.0.5E Beta Test May  8 2009
Mimer SQL server for database MIMDB_MOBILE STARTED at /usr/local/MimerSQL/
mimdb/mimdb_mobile

# bsql mimdb_mobile

Mimer SQL Command Line Utility  Version 10.0.5E Beta Test
Copyright (C) Mimer Information Technology AB. All rights reserved.

Username: SYSADM
Password:
SQL>insert_person;
PNR:1234567891
NAME:Ben
SURNAME:Johansen
SQL>insert_address;
PNR:1234567891
PHONE:0912345678
ZIPCODE:76577
ADDRESS:Bond Street 12
CITY:Trenton
SQL>select_persons;
        PNR NAME
SURNAME
================================================
 1234567890 Alan
Pearson
===
 1234567891 Ben
Johansen
===

                  2 rows found

SQL>select_address;
PNR_VALUE:1234567891
 ADDRESS_ID PHONE      ZIPCODE
ADDRESS
CITY
================================================================================
          2 0912345678 76577
Bond Street 12
Trenton
===

                  1 row found

SQL>exit;
#
```

In addition to the example above it is possible to let your development database act as a mobile database server, being operated by the mobile database server program. This way the database server on the desktop will emulate a Mimer SQL database server. To achieve this simply change the **ServerType** parameter in the *multidefs* file for the database. Please note, that the server must be restarted before the setting becomes active.

# Chapter 3

# Getting Started on the Device

## The Device Environment

The customized Mimer SQL Mobile software package for the target platform is either delivered with Mimer SQL Mobile for Linux, or as a separate distribution package.

This delivery of Mimer SQL for a specific device environment is usually a subset of an ordinary Mimer SQL Engine distribution for Linux, including only the most necessary parts for execution on a device. Being a subset, the Mimer documentation set found at the Mimer SQL Developer Site (http://developer.mimer.se/documentation) can be used for most issues.

Of course, there are areas, like for example installation and administration, which are very different in this environment. Usually there are more manual steps involved than for an enterprise installation. The intention is that these dissimilarities should be described in this document.

### Delivered Items for the Device

When the delivered device software package is unpacked, a tree structure including the following directories are obtained:

- bin- Location for executable files
- lib - Location for libraries
- misc - Location for various sorts of files
- opt - Location for various optional files and files that are not needed in a runtime production environment. For example, when the database API is chosen, copy the corresponding database API library/driver to the lib-directory mentioned above.

As you will see below there are a lot of files delivered, although a system in runtime only may need a few. This is to get the flexibility to customize the environment according to local preferences. For example, it is possible to run a standard Mimer SQL Engine subset on the device.

The following programs are delivered:

| | |
|---|---|
| `bin/dbserver_mob` | The database server program for execution in Mimer SQL Mobile mode. The only program that is actually needed on the device at runtime. |
| `bin/dumper` | A script that is automatically executed to collect system environment details in case of a database server failure. |
| `misc/license.txt` | License agreement notice. |
| `misc/mimerkey.mcfg` | Default license key. |
| `misc/mimrelease.info` | Basic information generated when creating the package. |
| `opt/bin/bsql` | Optional tool. Command line tool for ad-hoc SQL queries and batch execution. |
| `opt/bin/dbc` | Optional tool. Used to verify Mimer SQL databank files. |
| `opt/bin/dbfiles` | Optional tool. Used to list databank file names as stored in the data dictionary. |
| `opt/bin/dbopen` | Optional tool. Used to open all Mimer SQL databanks in a system (mainly to trig possible implicit databank verifications before the system is accessed). |
| `opt/bin/dbserver` | Optional program. The database server program in Mimer SQL Engine mode (including an SQL compiler). |
| `opt/bin/esql` | Optional tool. Embedded SQL Pre-processor for SQL statements embedded in the C/C++ language. |
| `opt/bin/exload` | Optional program. Used to load the Mimer SQL example database if using a Mimer SQL Engine database. (For details on the example database, please see the developer site article http://developer.mimer.se/features/feature_19.htm). |
| `opt/bin/mimcontrol` | Optional tool. Used to control the database server program (dbserver_mob or dbserver). |
| `opt/bin/mimhosts` | Installation tool. Used to install and manage the Mimer SQL database registration file. |
| `opt/bin/miminfo` | Optional tool. Used to monitor the database server program. |
| `opt/bin/mimlicense` | Installation tool. Used to install and manage the Mimer SQL License key file. |
| `opt/bin/mimload` | Optional tool. Used to load or unload database contents if using a Mimer SQL Engine database. |

| | |
|---|---|
| opt/bin/mimpath | Optional tool. Used to locate the database home path for a given database. |
| opt/bin/mimsysconf | Optional tool. Used to display some Linux system configuration parameters. |
| opt/bin/mimversion | Optional tool. Used simply to display the Mimer SQL version identification. |
| opt/bin/sdbgen | Optional tool. Used to generate initial system databank files when running in Mimer SQL Engine mode, i.e. with SQL compiler. |
| opt/lib/libmimcomm.so | Optional database API. Link library used when using JDBC over the local communication interface. |
| opt/lib/libmimdbi.so | Optional database API. Link library for Embedded SQL based applications. |
| opt/lib/libmimdbs.so | Optional library. Link library that is used automatically when accessing a database in single-user mode, i.e. when the database server program is not running for a database. |
| opt/lib/libmimodbc4.so | Optional database API. Link library for ODBC based applications when the ODBC client is presuming the SQLWCHAR datatype being 4 bytes. |
| opt/lib/libminodbc.so | Optional database API. Link library for ODBC base applications using a limited set of API calls. |
| opt/lib/midjdbc2.jar | Optional database API. CLDC/MIDP JDBC database API for Java. |
| opt/lib/mimjdbc3.jar | Optional database API. Standard JDBC database API for Java. |
| opt/lib/minjdbc3.jar | Optional database API. CDC/FP JDBC database API for Java. |
| opt/misc/singledefs | Optional configuration file. If accessing the database in single-user mode, this file can be used to adjust parameter values. |
| opt/misc/sqlhosts.dat | Optional template. This file can be used as a template when creating the Mimer SQL registration file. |

# Software Setup

In the examples used in this session the distribution is presumed to be `MimerMobileSQL10.0.5E` installed in `/opt`.

Please note, for a proper setup at every login the environment variable settings performed below should be done in local or global user login profiles like `/etc/profile`, as desired.

To prepare for execution on the device, do the following:

1. See to that the PATH environment variable includes a reference to the directory named "bin" of the distribution:

   ```
   # export PATH="$PATH:/opt/MimerMobileSQL-10.0.5E/bin"
   ```

   During development it is recommended that the installation `opt/bin/` directory is added as well. This directory holds programs that usually are not needed in runtime. To save space the opt-directory tree is expected to be deleted in a production runtime environment:

   ```
   # export PATH="$PATH:/opt/MimerMobileSQL-10.0.5E/opt/bin"
   ```

2. Setup LD_LIBRARY_PATH to include a reference to the directory named `lib` of the distribution (installed in the `/opt` directory in the following example):

   ```
   # export LD_LIBRARY_PATH="/opt/MimerMobileSQL-10.0.5E/lib"
   ```

   During development it is recommended that the installation opt/lib/-directory is added as well. This directory holds libraries/drivers that usually are not needed in runtime. To save space the opt-directory tree is expected to be deleted in a production runtime environment:

   ```
   # export LD_LIBRARY_PATH="/opt/MimerMobileSQL-10.0.5E/opt/lib"
   ```

3. Usually a license key is needed, which in that case should be installed using the mimlicense command. Install the license key in /etc/mimerkey using the following command from the directory named "misc" of the distribution:

   ```
   # export MIMER_KEYFILE=/opt/mimerkey
   # mimlicense -f /opt/MimerMobileSQL-10.0.5E/misc/mimerkey.mcfg
   ```

   The license key is by default installed in the /etc/mimerkey file, but to override this behavior a filename can be assigned in the MIMER_KEYFILE environment variable.

# Executing Example on Target Platform

## Initiate the Database Server

When the device software is in place it is time to put the database environment together. In this example the database name is `mimdb_mobile`.

1. Create a database home directory.

   ```
   # mkdir /opt/mimdb_mobile
   ```

2. Transfer the exported databank files from the desktop environment to this newly created directory.

3. The database and the related home directory should be registered in `/etc/sqlhosts` (or in a file pointed out by the MIMER_SQLHOSTS environment variable). This can be done using the mimhosts program.

   ```
   # export MIMER_SQLHOSTS=/opt/sqlhosts
   # mimhosts -l mimdb_mobile /opt/mimdb_mobile
   ```

An alternative is to use the provided template file and update it manually. Copy the template database registry file sqlhosts.dat file, found in the `misc` directory of the distribution to `/etc`.

```
# cp /opt/MimerMobileSQL-10.0.5E/misc/sqlhosts.dat /etc/sqlhosts
```

When using the template, the file needs to be manually updated. A simple sqlhosts file may look as the follows, where the database mimdb_mobile is registered as a local database, and also as the default database:

```
DEFAULT:
mimdb_mobile
LOCAL:
mimdb_mobile    /opt/mimdb_mobile
REMOTE:
```

4. If a default database is defined in the *sqlhosts* file it may not be necessary to perform this step, but it is still a recommendation to set the MIMER_DATABASE environment variable to hold the database name:

   ```
   # export MIMER_DATABASE=mimmersql_mobile
   ```

5. Generate the default configuration file (multidefs) for the database server by using the following command:

   ```
   # mimcontrol -g mimdb_mobile
   ```

6. Update the following parameters in the multidefs file:
   - Decrease the number of RequestThreads to 3.
   - Decrease the number of BackgroundThreads to 2.
   - Set the ServerType parameter to 1, indicating execution in Mobile mode (starting the dbserver_mob server program).

7. Complete the database directory by copying the dumper file from the bin directory of the installation. In case of an unexpected termination of the database server process, the *.dumper.sh* script will be executed to give additional details on the current process environment.

   ```
   # cp ./bin/dumper /opt/mimdb_mobile/.dumper.sh
   ```

# Executing on Target Environment

Now the database server can be started using the mimcontrol command (presumed to be located using PATH). Verify that the **ServerType** parameter of the multidefs file is set to 1 indicating that the mobile database server program, **dbserver_mob**, should be started:

```
# mimcontrol -s mimdb_mobile
```

When a mobile database server is stopped and restarted the databank files are decreased in size. I.e. databanks shrink when they go offline, i.e. released space is returned to the file system manager.

Now the statements in the database are ready to use from an application or from a Mimer SQL interpreter like BSQL or DbVisualizer. To list statements in a mobile environment the predefined statement information_schema.enum_statements can be used.

To stop the database server program, use the following command:

```
# mimcontrol -t mimdb_mobile
```

For further details on the **mimcontrol** program, please see the *System Management Handbook* part of the general Mimer SQL documentation set found at the Mimer SQL Developer Site (http://developer.mimer.se/documentation).

## Network Access

If the database on the target platform should be accessible over the TCP/IP network the **TCPPort** parameter in the *multidefs* file should be changed from `inetd` to a specific port number. The port number 1360 is recommended since it is registered for Mimer SQL. As mentioned earlier, whenever the *multidefs* file is updated the database server must be restarted for new parameter values to take effect.

## Avoiding Use of mimcontrol

It is possible to start the database server without using the **mimcontrol** program by executing the following commands:

```
# cd  /opt/mimdb_mobile
# dbserver_mob mimdb_mobile >> mimer.log 2>&1 &
```

To stop the database server without using mimcontrol, the SIGTERM signal is sent to the dbserver_mob process (where 1962 is the process identification number, PID, below):

```
# kill -TERM 1962
```

## Using Engine Environment on the Device

As hinted above, if resources are enough and footprint is acceptable, it is possible to run Mimer SQL Engine on the target device (using the **dbserver** database server program). This means that the entire range of SQL commands are supported on the target platform, including ALTER, CREATE and so on. It is also possible to compile new SQL statements.

The server type can be specified in the *multidefs* file via the **ServerType** parameter. For execution in Engine mode this parameter should be set to 0.

Note that you should use standard databank files in this case, i.e. not exported ones. Standard databanks can be generated on the device by using the sdbgen command, or by copying the origin databank files from the desktop environment. The sdbgen command is executed as follows, where `SYSPW` is used as the SYSADM password:

```
# sdbgen -p SYSPW mimdb
```

The sdbgen program can be executed without any arguments and will in that case prompt for information needed.

Prepare the database in the same way as described above for Mobile use.

When the preparations are completed the database server is ready to use. Executing in this mode enables the possibility to design, create and populate the database directly on the device. This can, for example, be used with the provided **exload** command that creates the Mimer SQL example database.

Other examples needed may be copied from the desktop development environment.

# Chapter 4

# Application Development

## Supported Database API's

Mimer SQL applications in a Mobile environment can use common Mimer SQL database API's like Embedded SQL, Standard ODBC and Java Standard Edition (SE). In addition, there are several different approaches available to use more minimized and reduced interfaces that may be more suitable in the current target environments.

As size is of great consequence on any handheld device, a general recommendation when developing you application is that you should minimize the size of the database by removing objects that are not needed.

**Note:** The database API drivers and libraries are located in the opt/lib-directory of the Mimer SQL software installation on the device. The recommendation is that the needed driver/library is copied to the lib-directory. This way the opt-directory tree can be removed for a production environment.

### Java Programming

The Mimer SQL Mobile product includes several Java/JDBC drivers for access to Mimer SQL databases. Which one to use depends upon the Java environment used on the target platform.

For details on Mimer JDBC, see the *Mimer JDBC Driver Guide* found at the Mimer SQL Developer Site (http://developer.mimer.se/documentation).

#### Java/JDBC using the Mimer CLDC/MIDP driver

Java applications using J2ME must be developed using a MIDP development environment. For this purpose any MIDP development tool may be used, the Sun Java Wireless Toolkit (WTK) is often used and can be recommended.

The Mimer MIDP driver, *midjdbc2.jar*, should be installed into the Wireless Toolkit manually. The MIDP driver is installed simply by copying the *midjdbc2.jar* file from the Mimer SQL installation directory to the `{wtklib}\apps\lib` directory where `{wtklib}` is the installation directory of the J2ME Wireless Toolkit. Depending upon the scope of the installation, i.e. if the driver should be globally installed or local to a user or a project, there are various installation options. For more information on how to install, configure and use the J2ME Sun Java Wireless Toolkit, see http://developer.mimer.com/howto/howto_43.htm.

### Java/JDBC using the Mimer CDC/FP driver

The Mimer SQL product includes a JDBC driver suitable for J2ME/CDC environments. There is an official document specifying what such a driver should support. The actual specification is called *JDBC for CDC/FP Optional Package*, see the official Sun web site (http://java.sun.com/javame/reference/apis/jsr169). The driver can be found in the Mimer installation directory, with the name *minjdbc3.jar*.

### Standard Java/JDBC

For a standard Java/JDBC based application, the *mimjdbc3.jar* driver should be used.

## Programming with ODBC

When working with ODBC as database API in this environment we recommend avoiding use of an ODBC Driver Manager (if there are not specific needs for using one). Instead we suggest that the application should be linked directly towards link libraries provided within this product.

The major advantage of using an ODBC Driver Manager is that you can get a transparent runtime layer towards several different kinds of database management systems. In these constrained environments where Mimer SQL Mobile is to be used it is not likely that more than one database is used, thus involving an ODBC Driver Manager complicates the software pile and adds on to the resource exploitation.

For details on using Mimer with ODBC, see the *Mimer SQL Programmer's Manual*. This manual is part of the Mimer SQL documentation set found at the Mimer SQL Developer Site (http://developer.mimer.se/documentation).

### ODBC using the minodbc driver

The *libminodbc* library contains a limited version of the standard ODBC specification. It is recommended that the application should be linked directly towards this API, not going through an ODBC Driver Manager (which also goes along with the fact that mobile target operating systems do not usually have an ODBC Driver Manager).

The provided *minodbc.h* file is a merge of *sql.h*, *sqlext.h*, *sqltypes.h* and *sqlucode.h*. It is recommended that application developers should include *minodbc.h* file (and not the other header files mentioned). This is not a requirement, but since *minodbc.h* reflects the functionality available in the Mobile ODBC driver it is convenient.

The limitations of this API compared to the standard ODBC library is that the ANSI interface is not supported. In addition, the following routines are **not** supported:

- SQLBrowseConnectW
- SQLColumnPrivilegesW
- SQLColumnsW

- SQLDataSourcesW
- SQLDriversW
- SQLErrorW
- SQLForeignKeysW
- SQLGetDescRecW
- SQLGetTypeInfoW
- SQLNativeSqlW
- SQLPrimaryKeysW
- SQLProcedureColumnsW
- SQLProceduresW
- SQLSetConnectOptionW
- SQLSetDescFieldW
- SQLSpecialColumnsW
- SQLStatisticsW
- SQLTablePrivilegesW
- SQLTablesW
- Standard ODBC

For a standard ODBC based application, the *libmimodbc4* library should be used. When linking directly to this library, the application should include the *mimcli.h* header file, which then takes care of including other needed header files in a proper way.

# Embedded SQL in C/C++

If using SQL embedded in C or C++ code, the esql pre-processor program should be used to prepare the code for compilation.

The *libmimdbi* library contains the database API used for embedded SQL.

For details on using embedded SQL with Mimer, see the *Mimer SQL Programmer's Manual*. This manual is part of the Mimer SQL documentation set found at the Mimer SQL Developer Site (http://developer.mimer.se/documentation).

# Chapter 5

# Summary Using the Mimer SQL Example Database

## Installing the Example Database

This example session is based on the Mimer SQL Example Database that should be installed on the desktop development Mimer environment.

Usually when installing the Mimer SQL Engine product, the example database is created by default, or it may be offered as an installation option. If it is installed at a later stage, the **mimexampledb** (or **exload**) command is recommended.

## Export the Database For Mobile Use

Now, export the database that includes the example database for use in a Mimer SQL mobile environment. This is done as described in *Export the Databank Files* on page *8*, using the **mimpdacopy** command.

## Start a Mobile Database Server

Start a Mimer SQL Mobile database server, using the created mobile databank files, in either of the following scenarios:

1 **Directly in the Mimer SQL development environment (x86)**

Simply start the database recently created by mimpdacopy using the **mimdbserver** (or **mimcontrol**) command, as described in the section *Starting a Mobile Database Server on x86* on page *10*.

**2 On the target mobile device using the Mimer SQL Mobile target device environment (ARM)**

Move the database directory recently created by mimpdacopy to the target device. The Mimer SQL target device environment is presumed being installed according to the instructions given in the *Software Setup* on page *17*. Initiate and start the database as described in the sections *Initiate the Database Server* on page *18*, and *Executing on Target Environment* on page *19*.

# Execute Predefined Statements

When accessing databank files that have been exported for mobile use, predefined SQL statements are used. In the Mimer SQL Example Database there are several predefined SQL statements defined, which will be described in the example sessions below.

First, there is a general statement called **information_schema.enum_statements** that can be used to list all predefined statements available. The session is taken from a session using the Mimer BSQL tool from the MIMER_STORE account, usually having the password *GoodiesRUs*:

```
SQL>information_schema.enum_statements;
STATEMENT_SCHEMA
STATEMENT_NAME
STATEMENT_TYPE
STATEMENT
================================================================================
INFORMATION_SCHEMA

ENUM_STATEMENTS

            57



===
INFORMATION_SCHEMA

STATEMENT_DEFINITION

            57



===
Continue ?g
STATEMENT_SCHEMA
STATEMENT_NAME
STATEMENT_TYPE
STATEMENT
================================================================================
INFORMATION_SCHEMA

STATEMENT_ATTRIBUTES

            57



===
```

```
INFORMATION_SCHEMA

SET_DATABANK_OPTION

            77



===
INFORMATION_SCHEMA

RESET_DATABANK_OPTION

            77



===
MIMER_STORE

PRODUCT_DETAILS_STMT

            57
-


===
MIMER_STORE

BARCODE_STMT

            85
CALL barcode(:ean)


===
MIMER_STORE

COMING_SOON_STMT

            85
CALL coming_soon(:category)


===
MIMER_STORE

VALIDATE_EAN_CODE_STMT

            77
SET :ean = validate_ean_code(:ean)


===
MIMER_STORE

MUSIC_DETAILS_STMT

            57
-


===
```

```
MIMER_STORE

MUSIC_SEARCH_STMT

            85
CALL mimer_store_music.Search(:title, :recorded_by, :max_rows)


===
MIMER_STORE

MUSIC_TITLE_DETAILS_STMT

            85
CALL mimer_store_music.TitleDetails(:item_id)


===
MIMER_STORE

MUSIC_TRACK_DETAILS_STMT

            85
CALL mimer_store_music.TrackDetails(:item_id)


===
MIMER_STORE

BOOK_DETAILS_STMT

            57
-


===
MIMER_STORE

BOOK_SEARCH_STMT

            85
CALL mimer_store_book.search(:title, :author)


===
MIMER_STORE

BOOK_TITLE_DETAILS_STMT

            85
CALL mimer_store_book.title_details(:item_id)


===

                16 rows found
```

Below, the predefined statements defined by the Mimer SQL Example Database are shown when executed from the Mimer BSQL command line tool, again, using the MIMER_STORE account:

```
SQL>PRODUCT_DETAILS_STMT;
PRODUCT
PRODUCER                                          FORMAT                    PRICE
 STOCK REORDER_LEVEL RELEASE_DATE             EAN_CODE STATUS PRODUCT_SEARCH
    ITEM_ID CATEGORY_ID  PRODUCT_ID DISPLAY_ORDER    IMAGE_ID
================================================================================
'Murder in the Cathedral'
HarperCollins                                     Paperback                 5.99
    17             4 1998-02-16          9780006498643 A      693953
      60646           2       30619            20           -
===
'Reave the Just' and Other Tales
Voyager                                           Paperback                 6.99
    14             3 1999-10-04          9780006511717 A      971433
      60647           2       30620            20           -
===
100 Anos
EMI International                                 Audio CD                  9.98
    16             5 1990-12-20            77774238724 A      064000
      60001           1       30001            20           -
===


Continue ?n


SQL>BARCODE_STMT;
EAN:606949030124
TITLE
CREATOR                                           FORMAT                    PRICE
    ITEM_ID
================================================================================
Greatest Hits
2Pac                                              Audio CD                  24.98
      60219
===

                 1 row found

SQL>COMING_SOON_STMT;
CATEGORY:music
PRODUCT
PRODUCER                                          FORMAT
RELEASE_DATE     PRICE     ITEM_ID
====================================================================
Greatest Hits
EMI Int'l                                         Audio CD
21/08/2001       14.98       60221
===
Seven Day Weekend
Sanctuary / Trojan                                Audio CD
11/09/2001       17.98       60484
===
Seven Day Weekend
Sanctuary / Trojan                                Audio CD
11/09/2001       17.98       60484
===
Seven Day Weekend
Sanctuary / Trojan                                Audio CD
11/09/2001       17.98       60484
===
```

```
Continue ?n


SQL>VALIDATE_EAN_CODE_STMT;
EAN:606949030124
                EAN
====================
        606949030124

                 1 row found

SQL>MUSIC_DETAILS_STMT;
ARTIST
PRODUCT
PRODUCER                                          FORMAT                    PRICE
 STOCK REORDER_LEVEL RELEASE_DATE             EAN_CODE STATUS PRODUCT_SEARCH
CATEGORY_ID  PRODUCT_ID DISPLAY_ORDER    IMAGE_ID     ITEM_ID ARTIST_SEARCH
  ARTIST_ID
================================================================================
(EC) Nudes
Vanishing Point
Recommended Records                               Audio CD                16.98
    11             6 1995-03-29           752725001922 A      764657
          1       30592            20           -       60618 056340
      50618
===
2Pac
Greatest Hits
Interscope Records                                Audio CD                24.98
     8             5 1998-11-24           606949030124 A      593435
          1       30206            20           -       60219 750000
      50219
===


Continue ?n


SQL>MUSIC_SEARCH_STMT;
TITLE:greatest
RECORDED_BY:2pac
MAX_ROWS:10
TITLE
ARTIST                                            FORMAT                    PRICE
    ITEM_ID   ARTIST_ID MATCH_LEVEL
================================================================================
Greatest Hits
2Pac                                              Audio CD                24.98
      60219       50219 ****
===

                 1 row found

SQL>MUSIC_TITLE_DETAILS_STMT;
ITEM_ID:60219
TITLE
ARTIST
LABEL                                             FORMAT
RELEASE_DATE     PRICE PLAY_TIME    IMAGE_ID     ITEM_ID
======================================================================
Greatest Hits
2Pac
Interscope Records                                Audio CD
24/11/1998       24.98                    -       60219
===

                 1 row found
```

```
SQL>MUSIC_TRACK_DETAILS_STMT;
ITEM_ID:60219
TRACK_NO TITLE                                          LENGTH   SAMPLE_ID
======== ============================================== ====== ===========
       1 Keep Ya Head Up                                                 -
       2 2 of Amerikaz Most Wanted                                       -
       3 Temptations                                                     -
       4 God Bless the Dead                                              -
       5 Hail Mary                                                       -
       6 Me Against the World                                            -
       7 How Do U Want It                                                -
       8 So Many Tears                                                   -
       9 Unconditional Love                                              -
      10 Trapped                                                         -
      11 Life Goes On                                                    -
      12 Hit 'Em Up                                                      -

                12 rows found

SQL>BOOK_DETAILS_STMT;

MIMER/DB internal error -19016
         Function not supported OPCODE=211

SQL>BOOK_SEARCH_STMT;
TITLE:best
AUTHOR:a

MIMER/DB internal error -19016
         Function not supported OPCODE=211

SQL>BOOK_TITLE_DETAILS_STMT;
ITEM_ID:61301

MIMER/DB internal error -19016
         Function not supported OPCODE=211
```

The final three executions above show a scenario giving an error code, *Function not supported OPCODE=211*, indicating that functionality has been used in the underlying database objects that is not supported in this particular Mimer SQL Mobile customization. In this case it is the REPLACE function that is not supported.

# Index