Migrating a MySQL Database to Adaptive Server Anywhere 9

A whitepaper from iAnywhere Solutions, Inc., a subsidiary of Sybase, Inc.

Contents

Introduction

Migrating data from MySQL to Adaptive Server Anywhere can be a straightforward process if there are not a lot of MySQL extensions in use within your database and application. Adaptive Server Anywhere simplifies migration by including built-in tools that facilitate a smooth transition from MySQL (and other RDBMS's) to Adaptive Server Anywhere.

The first part of this document discusses in detail differences between Adaptive Server Anywhere and MySQL, including data type differences, feature differences, and syntax differences. Some of the features are unique to MySQL and can hinder migration. Approaches for how you might choose to deal with these issues are provided. The second part of this document includes a systematic explanation of how to migrate data from a MySQL database into an Adaptive Server Anywhere database using the Sybase Central Data Migration wizard. Finally, part three of the document supplies an example of how you might migrate an existing application running against MySQL to one that runs against Adaptive Server Anywhere.

Software version
This paper was written for SQL Anywhere Studio version 9.0.1 and later, and MySQL version 4.0 and later.

Differences between MySQL and Adaptive Server Anywhere

The following sections describe some of the differences between MySQL and Adaptive Server Anywhere that you may encounter during migration, along with some suggested solutions that can be used as starting points to resolve any issues that arise during migration. There are many ways to optimize your code with Adaptive Server Anywhere features that are missing from MySQL. Here are a few examples:

- Using subqueries can help simplify your code and limit the use of temporary tables. MySQL 4.1 supports subqueries, but only in a rudimentary form.
- Row-level locking avoids the need to lock entire tables for update and can reduce contention when you have multiple users accessing the database.
- Foreign key support is native in Adaptive Server Anywhere, allowing you to let the server perform joins instead of implementing them in your application code (when using MyIsam tables), as well as performing cascading operations.
- Stored procedures and triggers can greatly simplify your application code and enable you to keep your business logic independent of your application log. MySQL has only added stored procedure support in version 5, which is not yet in Alpha at the time of writing.
- Using views can make your database schema and rights management much easier and more efficient.

It is highly recommended that you review the Adaptive Server Anywhere documentation, as well as the developer resources, including samples and technical documents, available on the iAnywhere Solutions website at *www.ianywhere.com/developer* when moving to SQL Anywhere Studio.

Data types

In most cases, the MySQL data types can map directly to Adaptive Server Anywhere data types. The following table lists some examples:

MySQL data type	Equivalent Adaptive Server Anywhere data type	Notes
smallint unsigned	UNSIGNED SMALLINT	
smallint	TINYINT	
bool/boolean	TINYINT or BIT	
double(m, n)	DOUBLE(n)	The ‘m’ argument in MySQL is related to data formatting. See note below.

MySQL data type	Equivalent Adaptive Server Anywhere data type	Notes
float(m,n) unsigned	FLOAT(n)	The 'm' argument in MySQL is related to data formatting. See note below.
fixed	DECIMAL	
varchar(32) binary	BINARY(32)	
national char/varchar	CHAR/VARCHAR	
tiny/medium/long text	CHAR/VARCHAR/LONG VARCHAR	Adaptive Server Anywhere stores only what is required to hold the text values, so the extra requirement of specifying BLOB size via different BLOB types is not required.
tinyblob	BINARY(255)	
blob/mediumblob/ longblob	LONG BINARY	Adaptive Server Anywhere stores only what is required to hold the BLOB value, so the extra requirement of specifying BLOB size via different BLOB types is not required
year	DATE	

Note: In addition to the differences in the data types themselves, there is also a difference in the declaration of data types. MySQL provides an optional parameter for its numeric types that allows you to specify the display width of integer values (for example, an int(4) column would return the value '1' as '<s><s><s><s>1' where <s> is a space). The optional 'ZEROFILL' modifier on the type definition would replace the spaces in the previous example with 0's (for example, '1' is returned as '0001'). The merge of display format and data values in the type definition is not supported by Adaptive Server Anywhere. The cast and convert functions along with the various string manipulation functions are available to format data values when they are retrieved from the database.

The following data types differ from Adaptive Server Anywhere more substantially than by syntax:

MEDIUMINT MEDIUMINTs are 3-byte integer values. They can easily be simulated using an integer (4 bytes) in Adaptive Server Anywhere, or a smallint (2 bytes), depending on the expected range of values for the column.

Year Year is a 2 or 4 digit year value. The Adaptive Server Anywhere DATE datatype can be used to hold year values , but will use slightly more storage space. Date arithmetic and conversions can be performed using the Adaptive Server Anywhere built-in functions listed in the documentation under "Date and Time Functions" in the *Adaptive Server Anywhere SQL Reference*.

The following data types do not match exactly, and will require some work to

migrate them into Adaptive Server Anywhere:

NCHAR / NVARCHAR Prior to MySQL version 4.1, NCHAR is the same as CHAR, and Adaptive Server Anywhere can support this type with no changes required. However, as of MySQL version 4.1, an NCHAR value is stored in MySQL using the UTF8 character set. Adaptive Server Anywhere supports a variety of character sets, including UTF8. With a database created using the proper collation, the use of a special data type to store international values is not required. To learn more about the latest international character set support in Adaptive Server Anywhere, see the chapter “International Languages and Character Sets” in the *Adaptive Server Anywhere Database Administration Guide*.

ENUM An ENUM value is a string object whose value must be chosen from a list of supplied values enumerated in the column definition when a table is created. The enumerated values can also be inserted/retrieved by their index position in the ENUM definition. The index value 0 is reserved for the empty string. The ENUM datatype is represented in Adaptive Server Anywhere by a TINYINT column. To accomplish the same behavior as MySQL ENUM, there are a few options, but changes to the client application will almost certainly be required. Some options you have are:

- altering the client side application to remove the need for the ENUM values
- translating the ENUM values on the client side
- adding some logic to the server side to attempt to mimic the MySQL behavior of ENUM values by using stored procedures, triggers, computed columns, views and/or a mapping table for the ENUM types

For example, a view could be created on the table containing the enum fields to allow for the return of the values as a string, while a regular SELECT could be used to return them as a number. Here is an example of a view that could be used:

```
CREATE TABLE enumtbl( pkey INTEGER NOT NULL PRIMARY KEY, enumval TINYINT );

CREATE VIEW v_enumtable AS
   SELECT pkey,
      CASE enumval
         WHEN 0 THEN ''
         WHEN 1 THEN 'val1'
         WHEN 2 THEN 'val2'
         WHEN 3 THEN 'val3'
      ELSE NULL
      END
   FROM enumtbl;
```

Then a query may look something like this:

```
SELECT pkey, enumval FROM v_enumtable;
```

Alternatively, a mapping table could be created for the ENUM values and whenever you retrieve data from enumtbl, a join can be made to the mapping table containing the ENUM strings.

```
CREATE TABLE enummap( enumval TINYINT NOT NULL PRIMARY KEY,  enumstr CHAR(16) );
```

Then a query may look something like this:

```
SELECT pkey, enumstr FROM enumtbl, enummap
WHERE enumtbl.enumval = enummap.enumval;
```

An insert on the table can be done directly if you are using the index values of the ENUM; otherwise, a stored procedure could be used to insert a row into any table containing an ENUM. The stored procedure would contain the logic to decode the ENUM values. Following is a sample stored procedure implementation to deal with an ENUM column equivalent in Adaptive Server Anywhere (using the same table definition as above):

```
CREATE PROCEDURE sp_insert_enumval( IN pkeyval int, IN enum CHAR(16) )
BEGIN
   DECLARE enum_map TINYINT;

   IF enum IS NOT NULL THEN
      CASE enum
         WHEN '' THEN SET enum_map = 0
         WHEN 'val1' THEN SET enum_map = 1
         WHEN 'val2' THEN SET enum_map = 2
         WHEN 'val3' THEN SET enum_map = 3
      ELSE SET enum_map = 0
      END CASE
   END IF;

   INSERT INTO enumtbl VALUES( pkeyval, enum_map );
END
```

SET A SET value is a string object whose value must be chosen from a list of values supplied when the column is defined. It is different from the ENUM type in that 0 or more values from the list may be combined to create a valid value for the column. Each value in the set is assigned a binary value and data can be assigned or retrieved by using a number representing the combination of values to be set (for example, specifying a value of 9 would insert the first and the fourth value from the set into the column). Depending on how many values are in the set (64 is the maximum), anything from a tinyint to a bigint is required to map a SET value from MySQL to Adaptive Server Anywhere. To achieve the same behavior as MySQL, methods similar to those demonstrated above with the ENUM data type can be used.

MySQL function mappings to Adaptive Server Anywhere

Many of the functions in both MySQL and Adaptive Server Anywhere have the same name. Most MySQL functions that have different names have an equivalent Adaptive Server Anywhere version. MySQL contains a few built-in functions that do not exist in Adaptive Server Anywhere. Most of these functions can be created in Adaptive Server Anywhere as user-defined functions that perform the same activity. If you give these functions the same name in the Adaptive Server Anywhere database, you will not need to modify the existing client application's SQL statements. Here are a some examples of how Adaptive Server Anywhere user-defined functions can supply the same functionality as their MySQL built-in counterparts:

```
CREATE FUNCTION FROM_UNIXTIME( IN fromdt bigint default 0,
  IN fmt varchar(32) default 'Mmm dd, yyyy hh:mm:ss' )
RETURNS datetime
BEGIN
  RETURN( dateformat( dateadd( second, fromdt, '1970/01/01 00:00:00' ), fmt ) )
END;
CREATE FUNCTION SEC_TO_TIME( IN sec bigint default 0 )
RETURNS time
BEGIN
  return( dateadd( second, sec, '1970/01/01 00:00:00' ) )
END;
```

The following sections detail many of the MySQL functions along with their Adaptive Server Anywhere equivalents. The list is extensive, but not exhaustive, as the list of functions in both Adaptive Server Anywhere and MySQL changes with each release.

String functions

MySQL function	Adaptive Server Anywhere function	Notes
IFNULL(a, b)	IFNULL(a, b, a) or ISNULL(a, b)	The Adaptive Server Anywhere IFNULL function behaves slightly differently from the MySQL version.
IF(cond, a, b)	IF cond THEN a ELSE b END IF	Adaptive Server Anywhere supports the IF statement in both procedural logic, as well as embedded, as an expression in a select list.
CONCAT(a, b, ...)	STRING(a, b, ...)	In MySQL, if any of a, b, ... is NULL, the return value is NULL, while in Adaptive Server Anywhere, if any of a, b, ... is NULL, it is treated as an empty string for the purposes of concatenation.
CONCAT_WS(sep, str1, str2, ...)	STRING(str1, sep, str2, sep, ...)	See comment for Concat () function above.
CONV(N, frombase, tobase)	INTOHEX(N), HEXTOINT(N)	Adaptive Server Anywhere only provides functions that allow you to convert to and from hexadecimal. Other conversions would have to be manually implemented using a UDF.
HEX(arg)	INTOHEX(srg) if arg is numeric, HEXTOINT(arg) if arg is string	
CHAR(N,...)	CHAR(N)	The Adaptive Server Anywhere CHAR() function only supports one argument.

MySQL function	Adaptive Server Anywhere function	Notes
STRCMP(expr1, expr2)	COMPARE(expr1, expr2)	
LENGTH(str)	BYTE_LENGTH(str)	The Adaptive Server Anywhere LENGTH() function returns the number of characters in str, not necessarily the byte length.
OCTET_LENGTH(str)	BYTE_LENGTH(str)	
CHARACTER_LENGTH(str)	CHAR_LENGTH(str)	
BIT_LENGTH(str)	BYTE_LENGTH(str) * 8	
LOCATE(substr, str[, pos])	LOCATE(str, substr[,pos])	The order of the arguments differs in Adaptive Server Anywhere.
POSITION(substr IN str)	LOCATE(str, substr)	
INSTR(str, substr)	LOCATE(str, substr)	
SUBSTRING(str FROM pos[FOR len])	SUBSTRING(str, pos[, len])	
MID(str, pos, len)	SUBSTRING(str, pos, len)	
TRIM(str)	TRIM(str)	Adaptive Server Anywhere does not support the other forms of the MySQL TRIM function.
INSERT(str, pos, len, newstr)	STUFF(str, pos, len, newstr)	
ELT(N, str1, str2, ...)	ARGN(N, str1, str2, ...)	
CONVERT(expr USING trans_name)	CSCONVERT(expr, trans_name)	trans_name may differ for equivalent character sets in Adaptive Server Anywhere and MySQL. See the documentation for details.

Numeric functions

MySQL function	Adaptive Server Anywhere function	Notes
CEIL(x)	CEILING(x)	
ROUND(x)	ROUND(x, 0)	

MySQL function	Adaptive Server Anywhere function	Notes
x DIV y	FLOOR(x/y)	
LN(x)	LOG(x)	
LOG(x, y)	LOG(x) / LOG(b)	
LOG2(x)	LOG(x) / LOG(2)	
POW(x, y)	POWER(x, y)	
ATAN(x, y)	ATAN2(x, y)	

Date and time functions

MySQL function	Adaptive Server Anywhere function	Notes
TIME(expr)	CAST(expr as TIME)	The CONVERT function could also be used.
TIMESTAMP(expr)	DATETIME(expr)	
DAYOFWEEK(expr)	DOW(expr)	
WEEKDAY(expr)	MOD(DOW(expr) - 1, 7)	
DAYOFMONTH(expr)	DAY(expr)	
WEEKOFYEAR(expr)	DATEPART(week, expr)	
PERIODADD(expr, N)	DATEFORMAT(DATEADD(month, 2, expr || '/01'), 'YYYYMM')	
PERIOD_DIFF(P1, P2)	DATEDIFF(month, P2||'/01', P1||'/01')	
ADDDATE(date, numdays)	DATEADD(day, numdays, date)	
SUBDATE(date, numdays)	DATEADD(day, -numdays, date)	
EXTRACT(type FROM date)	DATEPART(type, date)	The 'type' argument differs between Adaptive Server Anywhere and MySQL and will have to be adjusted accordingly.
TO_DAYS(date)	DAYS(date) - 58	Adaptive Server Anywhere measures differences in dates from '0000/02/29' instead of '0000/01/01'.

MySQL function	Adaptive Server Anywhere function	Notes
MAKEDATE(year, dayofyear)	YMD(year, 0, dayofyear)	
UNIX_TIMESTAMP(date)	DATEDIFF(second, '1979/01/01', date)	

Syntax mappings

Most of the syntax features of MySQL are available in Adaptive Server Anywhere, but occasionally the syntax for accessing those features is different. The following chart details many of these statements along with their Adaptive Server Anywhere equivalents.

☞ For specific examples of the Adaptive Server Anywhere syntax listed below, see "SQL Statements" in *Adaptive Server Anywhere SQL Reference*.

Operators

MySQL has several operators used to compare two or more arbitrary expressions and evaluate boolean expressions. The following is a list of those expressions, along with the Adaptive Server Anywhere equivalent if applicable.

MySQL operator	Adaptive Server Anywhere operator	Notes
!=	<>	
<=>	(expr1 = expr2 OR ((expr1 IS NULL) AND (expr2 IS NULL)))	The <=> operator represents equality, including NULL values (NULL=NULL is true).
ISNULL(expr)	IS NULL expr	
INTERVAL(N, N1, N2, ...)	none built in	A user defined function could easily be used to achieve the same function. For example: if (N < N1) then 0 elseif(N < N2) then 1 elseif ...
!	NOT	
&&	AND	
||	OR	
a XOR b	((a AND (NOT b)) OR ((NOT a) AND b))	The Adaptive Server Anywhere expression is complex for large numbers of XOR arguments, so an alternative method is recommended (dependant on the specific application scenario) to migrate these expressions.

Data Manipulation Language

MySQL statement	Adaptive Server Anywhere equivalent	Notes
INSERT ... ON DUPLICATE KEY UPDATE	INSERT ... ON EXISTING UPDATE	Adaptive Server Anywhere also offers the options ERROR and SKIP for existing rows.
SELECT ... INTO OUTFILE	UNLOAD SELECT ... DBISQL OUTPUT TO	
SELECT/UPDATE/DELETE ... LIMIT	FIRST or TOP n	
DEFAULT '0' NOT NULL auto_increment	NOT NULL DEFAULT AUTOINCREMENT	
NOT NULL auto_increment	NOT NULL DEFAULT AUTOINCREMENT	
LIMIT offset, numRows	TOP numRows START AT offset	
Insert IGNORE	INSERT ... ON EXISTING SKIP	
Replace ...	INSERT ... ON EXISTING UPDATE	
FROM_DAYS()	DAYS()	
TO_DAYS()	DATEADD(day, ...)	
WEEKDAY()	DOW()	
GROUP_CONCAT	LIST	
STD	STDDEV	
CHARACTER_LENGTH	LENGTH() function	
Position()	LOCATE() function	
LOCALTIME, LOCALTIMESTAMP	NOW() built in function	
DECODE	CASE statement	
INSERT INTO . . . DEFAULT VALUES.	INSERT INTO . . . VALUES(DEFAULT)	
LOAD DATA INFILE	LOAD TABLE	

Miscellaneous syntax

The following is a miscellaneous list of compatibility items that do not fit into the aforementioned categories. It also includes mappings between functions that are not exactly the same, but are designed to provide the same functionality.

MySQL syntax	Adaptive Server Anywhere syntax	Notes
VERSION()	@@version global variable	
mysql_insert_id()	@@identity global variable	
LAST_INSERT_ID variable	@@identity global variable	
mysql_affected_rows()	@@rowcount global variable	
ANALYZE TABLE	sa_table_page_usage, sa_table_fragmentation	Adaptive Server Anywhere also offers access to other properties via the **property()** function.
OPTIMIZE TABLE	CREATE STATISTICS	Adaptive Server Anywhere has a self-tuning optimizer that automatically maintains statistics, so statistics do not need to be updated manually.
CHECK TABLE	sa_validate() procedure	
USE *database-name*		There is no equivalent in Adaptive Server Anywhere. Each database running on a server requires its own connection.
LOCK TABLES (*name*) WRITE	LOCK TABLE *table-name* IN EXCLUSIVE MODE	Adaptive Server Anywhere supports row-level locking, so table locks are generally not required.
UNLOCK TABLES	COMMIT	A COMMIT releases all locks, unless a cursor is opened using the WITH HOLD clause.
Create table(KEY...)	CREATE TABLE ... CREATE INDEX	Adaptive Server Anywhere requires two statements.
DO	CALL	

MySQL syntax	Adaptive Server Anywhere syntax	Notes
FLUSH/RESET	sa_flush_cache sa_flush_statistics	Most of the other flushable elements in MySQL are automatically managed by Adaptive Server Anywhere and do not need to be flushed.
expr1 SOUNDS LIKE expr2	SOUNDEX(expr1) = SOUNDEX(expr2)	
REGEX/RLIKE	SIMILAR	SIMILAR works differently from the mysql REGEX syntax, but performs the same function. It may suit the needs where the MySQL REGEXP expression is being used.
BINARY str	CAST str AS BINARY	
CURDATE() | CURRENT_DATE()	CURRENT DATE	
CURTIME() | CURRENT_TIME()	CURRENT TIME	
SYSDATE() | LOCALTIME() | CURRENT_TIMESTAMP()	CURRENT TIMESTAMP	
UTC_DATE()	CURRENT UTC TIMESTAMP	
DATABASE()	CURRENT DATABASE	
LOAD_FILE(file)	xp_read_file(file)	In Adaptive Server Anywhere, the contents of file are returned as a long binary field, while in MySQL they are returned as a string.
CONNECTION_ID()	CONNECTION_PROPERTY('Number')	

Other migration issues

The following is a list of miscellaneous notes to keep in mind while migrating from MySQL to Adaptive Server Anywhere.

- The identifiers in MySQL are optionally enclosed with the back quote (‘), while Adaptive Server Anywhere uses the double quote (**"**) or alternatively, square brackets (**[ ]**).

- Some words are keywords in Adaptive Server Anywhere and not in MySQL, such as **comment** and **session**. These keywords must be enclosed in double quotes in order to be used with Adaptive Server Anywhere. Alternatively, you can use the Adaptive Server Anywhere NON_KEYWORDS option to change the list of recognized keywords.

  ☞ For information about the NON_KEYWORDS option, see "NON_KEYWORDS option [compatibility]" in *Adaptive Server Anywhere Database Administration Guide*.

- The minimum timestamp value in Adaptive Server Anywhere is '0001-01-01 00:00:00', while it is '0000-00-00 00:00:00' in MySQL.

- Timestamps in MySQL have the format of *YYYY-MM-DD hh:mm:ss*. Adaptive Server Anywhere includes fractions of a second as part of the timestamp value. The TIMESTAMP_FORMAT option allows you to specify the exact format used to return datetime values.

  ☞ For information about the TIMESTAMP_FORMAT option, see "TIMESTAMP_FORMAT option [compatibility]" in *Adaptive Server Anywhere Database Administration Guide*.

- While MySQL allows the use of single or double quotes around string literals, by default single quotes must be used to enclose string values in Adaptive Server Anywhere. As previously mentioned, by default, double quotes signify the use of a database object identifier. This behavior can be changed by setting the QUOTED_IDENTIFIER option in the database.

  ☞ For information about the QUOTED_IDENTIFIER option, see "QUOTED_IDENTIFIER option [compatibility]" in *Adaptive Server Anywhere Database Administration Guide*.

Migrating a MySQL database to an Adaptive Server Anywhere database

Migrating data from MySQL to Adaptive Server Anywhere is a straightforward process, with minor issues occurring only if you are using the MySQL-specific data types mentioned previously. Data migration can be accomplished using the Data Migration wizard that is a part of Sybase Central. Alternatively, a more customized migration can be done using the sa_migrate set of stored procedures in Adaptive Server Anywhere. The mysqldump utility, coupled with the Adaptive Server Anywhere LOAD TABLE statement, could also be used to migrate the data. Note that if the MySQL SET or ENUM data types are used in the MySQL database, you may have some additional considerations when migrating your MySQL database to Adaptive Server Anywhere.

☞ For information about these data types and differences from Adaptive Server Anywhere, see "Data types" on page 3.

Requirements

- This document assumes you have a MySQL database running on any of the supported platforms and Adaptive Server Anywhere 9.0.1 installed on any of the supported Windows platforms.
- If you have not created a MySQL database, you can create a few tables in the MySQL test database to walk through the migration steps.
- The MySQL ODBC 3.5.1 (or later) driver must also be installed on the machine running the Adaptive Server Anywhere database.

Creating an Adaptive Server Anywhere database

You must first create an Adaptive Server Anywhere database to migrate the MySQL database to. The following steps explain how to create a new database using Sybase Central.

1. Start Sybase Central. From the Start menu, choose Programs ➤ SQL Anywhere 9 ➤ Sybase Central.
2. Create a new Adaptive Server Anywhere 9 database.
   - In the left pane of Sybase Central, select Adaptive Server Anywhere 9.
   - In the right pane, click the Utilities tab.
   - Double-click Create Database.
     The Create Database wizard appears.
   - Follow the instructions in the wizard to create a new database.

Creating a data source for the MySQL database

The migration process requires an ODBC connection to the source database. Therefore, you need to create an ODBC data source (DSN) for the MySQL database.

1. Download and install the **MySQL ODBC 3.51 driver** if you have not already done so.

   The most recent driver is located at *http://www.mysql.com/downloads/api-myodbc.html*.

2. Start Sybase Central. From the Start menu, choose Programs ➤ SQL Anywhere 9 ➤ Sybase Central.

3. In the left pane of Sybase Central, select Adaptive Server Anywhere 9 and then click the Utilities tab in the right pane.

4. Double-click Open ODBC Administrator.

   The ODBC Data Source Administrator dialog appears.

5. Click Add.

   The Create New Data Source wizard appears.

6. Select the MySQL ODBC 3.51 Driver from the list of available drivers and then click Finish.

   The MySQL ODBC 3.51 Driver - DSN Configuration dialog appears.

7. Type a name for the data source in the Data Source Name field. For example, name the data source **MySQL migrate**.

8. Type the appropriate values in any other fields required for your MySQL database.

9. Click the Test Data Source button to ensure you have configured the data source correctly.

10. Click OK.

Migrating the MySQL database to Adaptive Server Anywhere

In order to migrate to the new Adaptive Server Anywhere database, you must first connect to the Adaptive Server Anywhere database. The following instructions explain how to connect using the database file location.

❖ **To connect to the Adaptive Server Anywhere database**

1. In the left pane of Sybase Central, select Adaptive Server Anywhere 9 and then from the File menu, choose Connect.

   The Connect dialog appears.

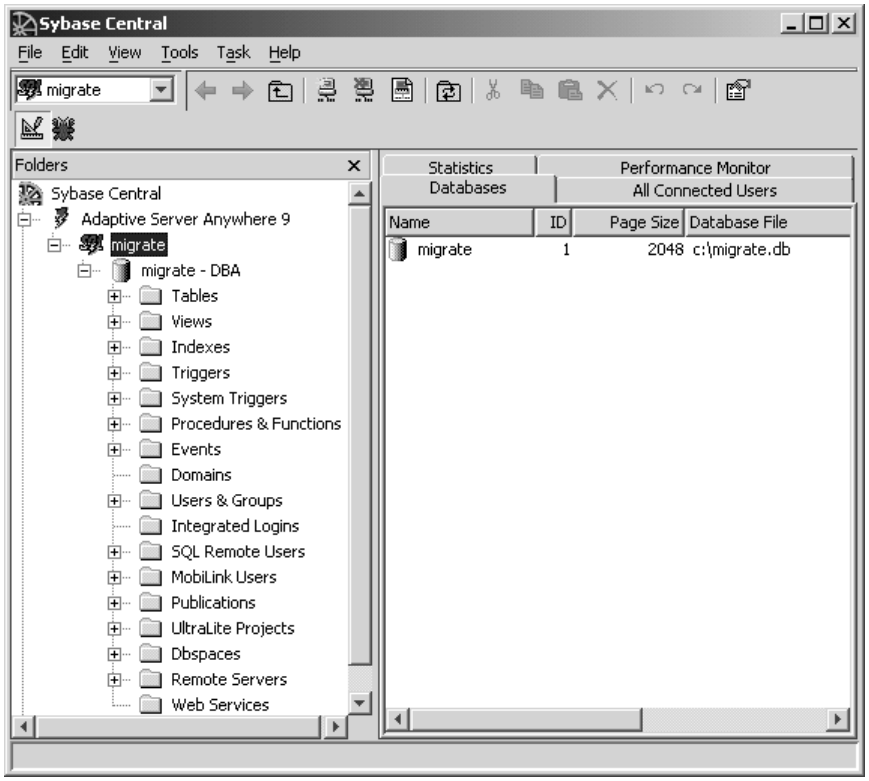
2. On the Identification tab, type a valid User ID and password for your database. By default, all Adaptive Server Anywhere databases contain a **DBA** user ID with the password **SQL**.
3. On the Database tab, click the Browse button and select the Adaptive Server Anywhere database file you created.
4. Click OK.

   The Adaptive Server Anywhere database server starts automatically.

The next step is to tell Sybase Central where to find the MySQL database. This is done by creating a remote server.

❖ **To create a remote server**

1. In the left pane of Sybase Central, expand your database server and database icons. In the example below, the database **migrate** is running on a database server that is also named **migrate**.



2. In Sybase Central, select the Remote Servers folder in the left pane.
3. From the File menu, choose New ➤ Remote Server.

   The Remote Server Creation wizard appears.
4. Follow the instructions in the wizard to create a remote server that connects to your MySQL database.

- On the first page of the wizard, type a name for the remote server, for example, **MySQL migrate**, and then click Next.
- Choose **Generic** as the type of remote server. Click Next.
- Select the Open Database Connectivity (ODBC) option and type the name of the ODBC data source for your MySQL database in the connection information field. For example, if you named your ODBC data source MySQL migrate when you created it, type **MySQL migrate** in the connection information field.

5. Click Finish.

   The new remote server appears in Sybase Central.

If the remote server does not define a user that is the same as the user ID you are connected to the Adaptive Server Anywhere database with, you must create an external login for your current user. For example, if you connected to the Adaptive Server Anywhere database with user ID **DBA**, and your MySQL database does not contain a user ID DBA, then you must create an external login.

❖ **To create an external login**

1. In the left pane of Sybase Central, open the Remote Servers folder and then select your remote server.
2. In the right pane, click the External Logins tab.
3. From the File menu, choose New ➤ External Login.

   The External Login Creation wizard appears.
4. Select the user you are currently connected as from the list of users.
5. Type the name of a user in the MySQL database in the Login Name field. Type the password for this user in the Password and Confirm Password fields. Click Finish.

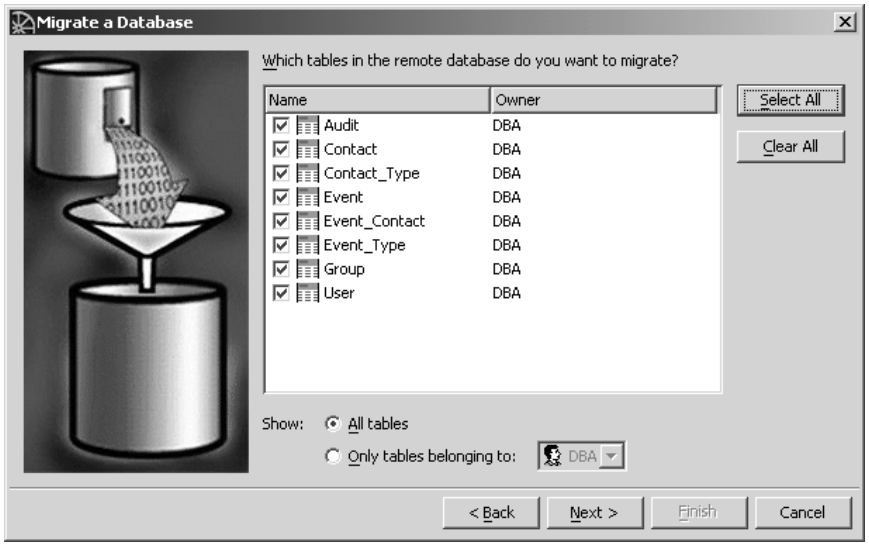
Now you are ready to migrate your MySQL database: Adaptive Server Anywhere is running, connected, and able to communicate to the MySQL database via ODBC. The next step is to use the Migration Wizard to perform the migration.

❖ **To migrate the MySQL database**

1. In the left pane of Sybase Central, select your Adaptive Server Anywhere database.
2. From the File menu, choose Migrate Database.

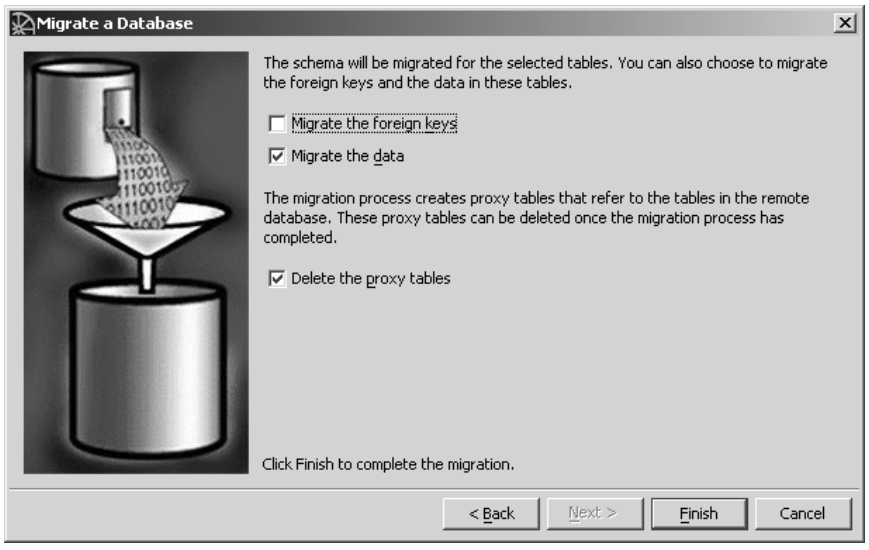
   The Database Migration wizard appears.
3. Click Next on the introductory page.
4. Select the current database and then click Next.
5. Select the MySQL remote server you created, for example, **MySQL migrate**, and then click Next.

6. Click the Select All button and then click Next to migrate all the MySQL tables to the Adaptive Server Anywhere database.



7. Select the Adaptive Server Anywhere database user you wish to own the tables. Click Next.

8. Select the options you wish to migrate. Because of limitations of the MySQL ODBC driver, foreign keys cannot be migrated. Clear the Migrate the Foreign Keys option to avoid errors later.



9. Click Finish to start the migration.

   The Migrating Database window appears. You can close this window when the status changes to **Completed**.

Tweaking the new Adaptive Server Anywhere database

Now that you have migrated the MySQL schema and data to the Adaptive Server Anywhere database, you can start enjoying the benefits Adaptive Server Anywhere brings. One immediate benefit is transactional support.

Here are a few easy tweaks that will make your existing schema even better.

Since not all MySQL tables support referential integrity, your MySQL schema may not have foreign keys. Even for InnoDB users, the MySQL ODBC driver does not support inspecting foreign key relationships; therefore the new Adaptive Server Anywhere database will not contain any foreign keys.

❖ **To add referential integrity support**

1. List the foreign keys in the MySQL database by issuing the following SQL statement against the MySQL database:

   ```
   SHOW TABLE STATUS FROM database_name
   ```

   Alternatively, `SHOW CREATE TABLE table_name` will also reveal any foreign key relationships.

   The referential constraints are listed under the comment column for each table in the form:

   ```
   (column_name) REFER ref_db_name/ref_table_name(ref_column_name)
   ```

2. Specify referential integrity constraints:
   - You can use Sybase Central to add the foreign keys to your database
   - Alternatively, for each of the foreign keys, issue the following SQL statement against the Adaptive Server Anywhere database (using the Interactive SQL utility (dbisql)):

     ```
     ALTER TABLE "table_name"
        ADD FOREIGN KEY "foreign_key_name" ("column_name")
        REFERENCES "ref_table_name" ("ref_column_name");
     ```

     With the new foreign key constraints in place, the Adaptive Server Anywhere database will check for referential integrity automatically and greatly improve data integrity.

Properly placed indexes improve database performance significantly, while poorly placed ones hinder performance with equal significance. SQL Anywhere Studio 9 introduced the Index Consultant that inspects database usage and workload and recommends changes to the indexing structure as needed. MySQL dictates that foreign key columns must have indexes explicitly defined, but this is not the case with Adaptive Server Anywhere. Also, for each primary key, MySQL creates a primary index that is redundant in Adaptive Server Anywhere. The Index Consultant will likely recommend removing the redundant indexes that are copied from the MySQL database during the migration process. The Index Consultant can prove to be a useful tool to boost the performance of the migrated Adaptive Server Anywhere database even further after migration has been completed.

☞ For information about optimizing your schema, refer to your SQL Anywhere Studio documentation and the iAnywhere developer resources available online at *www.ianywhere.com/developer*.

Migrating applications from MySQL to Adaptive Server Anywhere

Application migration from MySQL to Adaptive Server Anywhere depends on the interface used to access your MySQL application. The following are some of the more popular interfaces that should require only minimal work to migrate:

- **ODBC** Both Adaptive Server Anywhere and MySQL support the ODBC 3.51 API specification. Generally, migration of these applications will involve changing the ODBC data source to point to Adaptive Server Anywhere instead of MySQL. There may be some specific differences in terms of the implementation of certain API functions, but given the maturity of the ODBC specification, these should be minor.
- **JDBC** MySQL has a type 4 JDBC driver (100% Java implementation). To migrate to the Adaptive Server Anywhere equivalent, the Sybase jConnect driver should be used. However, to achieve the maximum performance benefits of Adaptive Server Anywhere, it is recommended that you use the iAnywhere JDBC driver. The iAnywhere JDBC driver is a type 2 JDBC driver. The Adaptive Server Anywhere JDBC drivers support all of the core elements of the JDBC 2.0 specification and some of the optional ones.
- **Perl** For information about migrating Perl applications, see “Migrating a Perl application from MySQL to Adaptive Server Anywhere” on page 22.
- **PHP** For information about migrating PHP applications, see “Migrating a PHP application from MySQL to Adaptive Server Anywhere” on page 23.

Applications written using the other interfaces supported by MySQL will require more work to migrate as there is no support for these drivers in Adaptive Server Anywhere. This includes the MySQL C/C++ API and the Python, Tcl, and Eiffel access drivers. In some cases, a third-party driver may be found that allows you to bridge to ODBC or natively access Adaptive Server Anywhere.

Migrating a Perl application from MySQL to Adaptive Server Anywhere

Migration of Perl applications from MySQL to Adaptive Server Anywhere is very simple. You have the option of using ODBC to connect using the DBD::ODBC driver or using the native Adaptive Server Anywhere driver (called DBD::ASAny) that ships with Adaptive Server Anywhere.

If you are already using the DBD::ODBC driver, application migration is simply a matter of changing your connection string to refer to Adaptive Server Anywhere. Once that is complete, there may be some minor tweaks required to deal with the differences between Adaptive Server Anywhere and MySQL as discussed in previous sections of this paper, but minimal work is required to complete the migration.

Some MySQL-specific methods can be migrated to Adaptive Server Anywhere equivalents by using queries or standard DBD functionality. For example:

MySQL	Adaptive Server Anywhere	Comment
mysql_insertid	SELECT @@identity	
is_blob, is_num, is_not_null, length, name, table, type	NAME, TYPE, SCALE, PRECISION, NULLABLE	All of these property items are DBI standard elements
is_key, is_pri_key	SELECT . . . FROM syscolumn WHERE ...	Detection of indexes/keys can be done by looking at the table and column definitions in the system tables

Migrating a PHP application from MySQL to Adaptive Server Anywhere

Migrating a PHP application from MySQL to Adaptive Server Anywhere is simple. You have the option of using ODBC to connect to Adaptive Server Anywhere or using an Adaptive Server Anywhere-PHP module provided by iAnywhere Solutions (available for download from the iAnywhere Developer website at *www.ianywhere.com/developer*).

Windows users may prefer to migrate to the ODBC API. Setting up a DSN in Windows for use with ODBC is simple. In addition, the Windows binary for PHP already has built-in ODBC support.

Linux users, on the other hand, may find the PHP module more convenient to set up. Adaptive Server Anywhere support can be compiled into PHP using the `-with sqlanywhere[=path_to_asa]` flag when calling the configure script. Details about the module can be found at *http://www.sybase.com/detail?id=1019698*.

If the PHP application is already using ODBC to connect to the MySQL database, then there is no need to change the function calls. You can skip the section below and go directly to "PHP migration notes" on page 25.

Function mapping

The MySQL, ODBC, and Adaptive Server Anywhere APIs are very similar. It is often possible to map one function directly to another. Sometimes, when a function has no equivalent counterpart, you must be creative and come up with alternative code that achieves the same result. In certain cases, you may be better off rewriting small portions of the code to take advantage of advanced features provided by Adaptive Server Anywhere. For example, with transaction support, the application can efficiently maintain atomicticity and easily ensure data integrity.

The following table lists some commonly used MySQL functions and their ODBC and Adaptive Server Anywhere equivalents.

MySQL	Adaptive Server Anywhere (ODBC)	Adaptive Server Anywhere (PHP module)
mysql_close	odbc_close	sqlanywhere_disconnect
mysql_connect	odbc_connect	sqlanywhere_connect
mysql_errno	odbc_error	See "mysql_errno" on page 24
mysql_error	odbc_errormsg	None
mysql_escape_string	See "mysql_escape_string" on page 24	See "mysql_escape_string" on page 24
mysql_fetch_row	odbc_fetch_row	sqlanywhere_fetch_row
mysql_insert_id	See "mysql_insert_id" on page 24	See "mysql_insert_id" on page 24
mysql_num_fields	odbc_num_fields	sqlanywhere_num_fields
mysql_num_rows	odbc_num_rows	sqlanywhere_num_rows
mysql_query	odbc_exec	sqlanywhere_query
mysql_select_db	None	None

mysql_connect

Connecting via ODBC is straight forward. The odbc_connect function takes, at a minimum, the DSN, user name, and password.

Connecting via the PHP module requires an Adaptive Server Anywhere connection string. Usually this can be done by the following function call:

```
sqlanywhere_connect("uid=DBA;pwd=SQL;eng=eng_name")
```

mysql_errno

This function returns the error number of the previous query. The same result can be obtained by issuing the following SQL statement:

```
SELECT @@error
```

mysql_escape_string

Neither ODBC nor the PHP module provides a way to escape a SQL string. However, this can be easily done by replacing each single quote with two single quotes.

mysql_insert_id

This function returns the last inserted ID of an autoincrement column. The same result can be obtained by issuing the following SQL statement:

```
SELECT @@identity
```

As you can see, many MySQL functions translate directly into ODBC and Adaptive Server Anywhere calls. For the remaining functions, simple alternatives can be found. As with any migration job, there are, unfortunately, differences between MySQL and Adaptive Server Anywhere that require more attention. These points are discussed in the following section.

PHP migration notes

There are subtle differences in the way SQL strings are treated by the various database vendors. For example, timestamps in MySQL have the format ***YYYY-MM-DD hh:mm:ss***, while Adaptive Server Anywhere supports timestamps with fractions of a second. The strtotime function in PHP fails to recognize Adaptive Server Anywhere timestamps. Extra work must be done to remove the fractional second portion of the Adaptive Server Anywhere timestamp.

Adaptive Server Anywhere via ODBC also provides support for transactions and prepared statements. The odbc_commit and odbc_rollback functions terminate a transaction as you would expect. One point to notice is that PHP defaults to autocommit, meaning every statement is committed as soon as it is successfully executed. The odbc_autocommit function is used to set the autocommit behavior to enable the use of larger transactions. Prepared statements are useful if the same queries, possibly with different parameters, are to be executed many times. This can help increase efficiency as each dynamic SQL statement is built within the engine once only. The odbc_prepare and odbc_execute functions are used to execute prepared statements.

To summarize, migrating your PHP application from MySQL to Adaptive Server Anywhere involves migrating the database, changing MySQL function calls to Adaptive Server Anywhere calls, and tweaking the schema and SQL statements to resolve any differences between the databases. Typically, some performance gains can be achieved by utilizing advanced features available in Adaptive Server Anywhere.

Legal Notice

Contact Us

iAnywhere Solutions Worldwide Headquarters One Sybase Drive, Dublin, CA, 94568 USA

Phone 1-800-801-2069 (in US and Canada)

Fax 1-519-747-4971

World Wide Web *http://www.ianywhere.com*

E-mail contact.us@ianywhere.com