Migrating From IBM DB2 to Adaptive Server Anywhere

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

Contents

Introduction **4**
General application porting issues 4

Special registers **5**

Constants **6**

Operators and predicates **7**

Data manipulation language **9**
ALLOCATE CURSOR 9
ASSOCIATE LOCATORS 9
CALL 10
CLOSE 10
COMMIT 10
CONNECT 10
DECLARE CURSOR 10
DECLARE GLOBAL TEMPORARY TABLE 10
DELETE (POSITIONED and SEARCHED) 11
DESCRIBE 11
DISCONNECT 11
EXPLAIN 11
FETCH 11
FREE LOCATOR 12
INSERT 12
LOCK TABLE 12
MERGE 12
RELEASE SAVEPOINT 13
ROLLBACK 13
SAVEPOINT 13
SELECT 13
SET CURRENT DEFAULT TRANSFORM GROUP 16
SET CURRENT DEGREE 16
SET CURRENT EXPLAIN MODE 16
SET CURRENT EXPLAIN SNAPSHOT 16
SET CURRENT ISOLATION 16
SET CURRENT LOCK TIMEOUT 16
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION . . . 17
SET CURRENT PACKAGE PATH 17
SET CURRENT PACKAGESET 17
SET CURRENT QUERY OPTIMIZATION 17
SET CURRENT REFRESH AGE 17
SET ENCRYPTION PASSWORD 17
SET INTEGRITY 17
SET PASSTHRU 17
SET PATH 18
SET SCHEMA 18
SET SERVER OPTION 18

SET SESSION AUTHORIZATION | SET SESSION USER 18
SET VARIABLE 18
SIGNAL 18
UPDATE 18
VALUES [INTO] 19

Compound statements **19**
BEGIN 19
CASE 20
FOR 20
FETCH 20
GET DIAGNOSTICS 20
GOTO 21
IF 21
ITERATE 21
LEAVE 21
LOOP 21
OPEN 21
REPEAT / UNTIL 21
RESIGNAL/SIGNAL 21
RETURN 22
WHILE 22

Functions, procedures, and triggers **22**
CREATE FUNCTION 23
CREATE PROCEDURE 24
CREATE TRIGGER 25

General error handling **25**

Built-in functions **26**

Data definition language **32**
ALTER TYPE ADD METHOD 33
COMMENT 33
CREATE DISTINCT TYPE 33
CREATE FUNCTION 33
CREATE INDEX 33
CREATE NICKNAME | ALIAS 34
CREATE PROCEDURE 34
CREATE SCHEMA 34
CREATE SEQUENCE 34
CREATE SERVER 34
CREATE TRANSFORM 34
CREATE TRIGGER 34
CREATE TABLE 35
CREATE TABLESPACE 37
CREATE TYPE 37
CREATE USER MAPPING 37
CREATE VIEW 38

CREATE WRAPPER . . . 38
DROP . . . 38
RENAME . . . 38
RENAME TABLESPACE . . . 38

Administration language **38**
CREATE BUFFERPOOL . . . 38
CREATE DATABASE PARTITION GROUP . . . 39
CREATE EVENT MONITOR | SET EVENT MONITOR STATE | FLUSH EVENT MONITOR . . . 39
FLUSH PACKAGE CACHE . . . 39
REFRESH TABLE . . . 39
GRANT/REVOKE . . . 39

Using the Migration wizard **41**
Creating an Adaptive Server Anywhere database . . . 42
Creating a data source for the DB2 database . . . 42
Migrating a DB2 database to Adaptive Server Anywhere . . . 43

Migrating an application from DB2 to Adaptive Server Anywhere **47**
Commenting on connections . . . 47
Porting CLI/ODBC applications . . . 47
Porting Java applications . . . 48
Porting PHP applications . . . 48
Porting Perl applications . . . 50
Porting .NET, ADO, or RDO applications . . . 50
Porting embedded SQL applications . . . 51

Legal Notice **55**
Contact Us . . . 55

Introduction

Migrating data from DB2 to Adaptive Server Anywhere can be a straightforward process if there are not a lot of DB2 extensions in use within your database and application. Adaptive Server Anywhere simplifies migration by including built-in tools that facilitate a smooth transition from DB2 UDB (and other RDBMSs) to Adaptive Server Anywhere.

This document discusses in detail differences between Adaptive Server Anywhere and DB2, including data type differences, feature differences, and syntax differences. This discussion does not include semantic differences between Adaptive Server Anywhere and DB2 that may impact your migration. Approaches for how you might choose to deal with these issues are provided where possible. This document also includes a systematic explanation of how to migrate data from a DB2 database into an Adaptive Server Anywhere database using the Sybase Central Migrate Database wizard. Finally, this document supplies an example of how you might migrate an existing application running against DB2 to one that runs against Adaptive Server Anywhere.

This paper was written for SQL Anywhere Studio version 9.0.2 and later, and IBM DB2 UDB version 8.2 and later.

General application porting issues

If your application relies heavily on the following, there may be significant work to do as Adaptive Server Anywhere does not support these extensions or significant differences exist.

- XA transactions (non-MS DTC)
- Mixing ESQL and CLI in an application
- Data change tables
- Structured data types
- Materialized views (ASTs)
- LOBs (indicators are not supported in Adaptive Server Anywhere)
- Datalink data types
- Writing DB2 utilities with C/C++ code

The default command termination delimiter with DB2 scripts is @, while Adaptive Server Anywhere uses **;** or **go**. Interactive SQL can be changed to use @ by specifying the **–d** @ option when starting Interactive SQL or by changing the value of the command_delimiter option.

Data types

This section covers basic SQL data types.

☞ For detailed information about the recommended data type mappings between DB2 UDB and Adaptive Server Anywhere, see “Data type conversions: DB2” at *http://www.ianywhere.com/developer/product_manuals/-sqlanywhere/0902/en/html/dbugen9/00000654.htm*.

Note that DB2 DATALINK, and structured types cannot be mapped to Adaptive Server Anywhere.

The default timestamp format in DB2 is *YYYY-MM-DD-HH-MM-SS.nnnnn*, while in Adaptive Server Anywhere the default timestamp format is *YYYY-MM-DD HH-MM-SS.nnn*. You can execute the following SQL statement in Adaptive Server Anywhere to change the timestamp format:

```
SET OPTION PUBLIC.TIMESTAMP_FORMAT="YYYY-MM-DD-HH-MM-SS.nnnnn"
```

The default date format in DB2 is *MM-DD-YYYY*, while in Adaptive Server Anywhere it is *YYYY-MM-DD*. Use the following SQL statement to change the date the date format in Adaptive Server Anywhere:

```
SET OPTION PUBLIC.DATE_FORMAT="MM-DD-YYYY"
```

While IBM DB2 and Adaptive Server Anywhere have the same default means to insert date information, it is still noteworthy that the Adaptive Server Anywhere DATE_ORDER option allows you to control the order of date parts. The default is Year, Month, Day (‘YMD’). To change the order to Month, Day, Year, use the following command:

```
SET OPTION PUBLIC.DATE_ORDER='MDY'
```

Special registers

Adaptive Server Anywhere registers are not updateable.

DB2	Adaptive Server Anywhere equivalent	Comments
CURRENT CLIENT_-ACCTNG	AppInfo	
CURRENT CLIENT_-APPLNAME	Current user	
CURRENT CLIENT_-USERID	AppInfo	
CURRENT CLIENT_-WRKSTNNAME	SELECT nodeaddr FROM sa_conn_-info( ) WHERE number= @@spid	You can get an IP address, but not a name.
CURRENT DATE	CURRENT DATE	

DB2	Adaptive Server Anywhere equivalent	Comments
CURRENT DBPARTITION-NUM	N/A	
CURRENT DEFAULT TRANSFORM GROUP	N/A	
CURRENT DEGREE	N/A	
CURRENT EXPLAIN MODE	N/A	
CURRENT EXPLAIN SNAPSHOT	N/A	
CURRENT ISOLATION	SELECT CONNECTION_-PROPERTY('isolation_level')	
CURRENT LOCK TIME-OUT	N/A	
CURRENT MAINTAINED TABLE TYPES FOR OPTI-MIZATION	N/A	
CURRENT PACKAGE PATH	N/A	
CURRENT PATH	N/A	
CURRENT QUERY OPTI-MIZATION	SELECT CONNECTION_-PROPERTY('optimization_level')	
CURRENT REFRESH AGE	N/A	
CURRENT SCHEMA	CURRENT USER	
CURRENT SERVER	CURRENT DATABASE	
CURRENT TIME	CURRENT TIME	
CURRENT TIMESTAMP	CURRENT TIMESTAMP	
CURRENT TIMEZONE	SELECT CONNECTION_-PROPERTY('TimeZoneAdjustment')	
CURRENT USER	CURRENT USER	
SESSION_USER	CURRENT USER	
SYSTEM_USER	CURRENT USER	
USER	CURRENT USER	

Constants

String and numeric constants are consistent between DB2 and Adaptive Server Anywhere.

The format of hexadecimal constants in IBM DB2 that need to be addressed are x'FFFF', which map to 0xFFFF in Adaptive Server Anywhere. This difference may affect your scripts, but will likely not affect applications.

Operators and predicates

Arithmetic error behavior can be controlled with the DFT_SQLMATHWARN setting in DB2. With Adaptive Server Anywhere, you can set the divide_by_zero option to obtain similar error handling.

Unlike DB2, Adaptive Server Anywhere does not support user-defined SQL operands.

DB2	Adaptive Server Anywhere equivalent	Comments
ALL, ANY, SOME predicates	ALL, ANY, SOME predicates	
BETWEEN predicate	BETWEEN predicate	
CASE	CASE	
CAST	CAST	The SCOPE clause is not supported as it relates to structured types.
CONCAT	||	
DATE TIME arithmetic	Supported, but there are differences (see Comments).	For addition and subtraction of duration, DB2 interprets casting integers to identify whether the value is a DATE duration, TIME duration, or TIMESTAMP duration. Adaptive Server Anywhere supports adding and subtracting days to TIMESTAMPs and DATEs. It is recommended that you use the DATEADD function for specific functionality. For date/time subtraction, Adaptive Server Anywhere supports DATE – DATE and TIMESTAMP – TIMESTAMP. TIME – TIME is not supported. DATEDIFF is recommended for this type of functionality.
EXISTS predicate	EXISTS predicate	
IN predicate	IN predicate	☞ See Quantified predicates below.

DB2	Adaptive Server Anywhere equivalent	Comments
IS [NOT] NULL	IS [NOT] NULL	
LIKE predicate	LIKE	The ESCAPE CLAUSE is not supported in Adaptive Server Anywhere. When your search pattern includes trailing blanks, Adaptive Server Anywhere matches the pattern only to values that contain blanks—it does not blank-pad strings.

Quantified predicates	Adaptive Server Anywhere does not support row value constructs (ANSI feature F641).	Many queries using this feature can be reworded. Consider the following query:
		`SELECT * FROM TA`
		`  WHERE ( TA.x, TA.y )`
		`  IN ( SELECT TB.x, TB.y`
		`  FROM TB`
		`  WHERE z = 'abc' )`
		This query can be reworded to the following, which has similar semantics:
		`SELECT * FROM TA`
		`  WHERE EXISTS (`
		`  SELECT * FROM TB`
		`  WHERE TB.x = TA.x`
		`  AND TB.y = TA.y AND TB.z`
		`  = 'abc' )`

= Equal to > Greater than < Less than >= Greater than or equal to <= Less than or equal to != Not equal to <> Not equal to !> Not greater than !< Not less than	= Equal to > Greater than < Less than >= Greater than or equal to <= Less than or equal to != Not equal to <> Not equal to !> Not greater than !< Not less than	
Dereferencing operator		Adaptive Server Anywhere does not support structured types.
Scalar full-select	Supported.	

DB2	Adaptive Server Anywhere equivalent	Comments
SELECTIVITY	(CONDITION *selectivitypercentage*)	Adaptive Server Anywhere uses the term user estimates instead of SELECTIVITY. ☞ For more information, see "Explicit selectivity estimates" at *http://www.ianywhere.com/developer/product_manuals/sqlanywhere/0902/en/html/dbrfen9/00000040.htm*
Sequence references		Adaptive Server Anywhere does not support sequences. Use GLOBAL AUTOINCREMENT or AUTOINCREMENT and @@identity instead of sequences.
TREAT		Adaptive Server Anywhere does not support structured types.
TYPE predicate		Adaptive Server Anywhere does not support structured types.
VALUES(X)		Using SELECT X is recommended.

Data manipulation language

Applications rely heavily on the SQL language a database supports. You will find a large overlap of supported syntax between Adaptive Server Anywhere and DB2 databases. Note that reliance on data change tables, structured data types, LOB indicators, LOB files, DATALINKS, administrative clauses, and data federation clauses may introduce additional migration considerations. If your application does not rely on any of these DB2 extensions, your migration will be much easier.

ALLOCATE CURSOR

Not required.

ASSOCIATE LOCATORS

LOCATORS variables are not supported in Adaptive Server Anywhere.

CALL

Calling SQL and Java procedures is supported in Adaptive Server Anywhere.

CLOSE

While DB2 supports this statement interactively, Adaptive Server Anywhere supports this statement in embedded SQL and procedural language. The WITH RELEASE clause is not required and the locks are released implicitly.

COMMIT

Both standard and SAVEPOINT syntax is supported.

CONNECT

The USING *password* clause must be replaced with IDENTIFIED BY *password*. The lock-block clause is not supported in Adaptive Server Anywhere and must be removed. The lock-block clause controls whether other users can use the system. To obtain single-user mode in Adaptive Server Anywhere, you can start the database server with the –b option or use the sa_server_option stored procedure. Adaptive Server Anywhere does not support the NEW and CONFIRM clauses of the authorization block clause that change passwords. Password management logic must be changed to use the GRANT CONNECT statement.

DECLARE CURSOR

WITH HOLD is not supported in the DECLARE CURSOR statement, but it is supported in the OPEN statement.

```
WITH RETURN TO CALLER
WITH RETURN TO CLIENT
```

The return is determined dynamically by the database server where the result set is returned. In Adaptive Server Anywhere, these clauses are not required.

☞ For more information, see item 3.e in "BEGIN" on page 19.

DECLARE GLOBAL TEMPORARY TABLE

Adaptive Server Anywhere supports DECLARE LOCAL TEMPORARY TABLE. Adaptive Server Anywhere temporary tables do not support to specify storage options such as IN TABLESPACE.

☞ For information about differences in TABLE definitions, see "CREATE TABLE" on page 35.

DELETE (POSITIONED and SEARCHED)

Unless the application is using TYPED TABLES or data change tables, the DELETE statement will migrate to Adaptive Server Anywhere. The FROM ONLY clause applies to TYPED TABLES, which Adaptive Server Anywhere does not support. The include and assignment clauses apply to data change tables (or intermediate result set), which Adaptive Server Anywhere does not support. If you use data change tables in your queries, you may want to consider using triggers and temporary tables for updates/deletes to obtain a desired flow control as a workaround to data change tables. The WITH clause needs to change. The WITH clause can be mapped to the following:

DB2	Adaptive Server Anywhere
WITH RR	WITH SERIALIZABLE
WITH RS	WITH REPEATABLEREAD
WITH CS	WITH READCOMMITED
WITH UR	WITH READUNCOMMITED

DESCRIBE

Supported in embedded SQL with Adaptive Server Anywhere.

☞ For more information, see "Porting embedded SQL applications" on page 51.

DISCONNECT

With Adaptive Server Anywhere, you specify a connection name, rather than a server name, in the syntax. If no name is specified, then the current connection is disconnected. The keyword ALL is not supported in Adaptive Server Anywhere.

EXPLAIN

Adaptive Server Anywhere does not support EXPLAIN in an interactive environment, but rather in embedded SQL. In most cases you would not include the EXPLAIN statement in your application because Adaptive Server Anywhere has a different approach to performance tuning than IBM DB2.

☞ For information about performance tuning in Adaptive Server Anywhere, see "How the optimizer works" at *http://www.ianywhere.com/developer/product_manuals/sqlanywhere/0902/en/html/dbugen9/00000391.htm* and PLAN function at *http://www.ianywhere.com/developer/product_manuals/sqlanywhere/0902/en/html/dbrfen9/00000247.htm*.

FETCH

There is no difference in the syntax, however there are semantic differences, and

Adaptive Server Anywhere does not support intent locks, FETCH FOR UPDATE queries, or write locks.

FREE LOCATOR

LOCATORS variables are not supported in Adaptive Server Anywhere.

INSERT

The Adaptive Server Anywhere INSERT statement differs in the following two ways from the DB2 INSERT statement:

- The Adaptive Server Anywhere INSERT statement does not support inserting into a full SELECT statement. It does, however, support inserting into a table/view from a SELECT statement.
- The Adaptive Server Anywhere INSERT statement does not support the INCLUDE (columns) syntax as Adaptive Server Anywhere does not support intermediate result tables. For example:

```
SELECT inorder.ordernum
FROM ( INSERT INTO orders ( custno )INCLUDE ( insertnum
         integer )
VALUES( :cnum1,1 ),( :cnum2,2 ) )InsertedOrders
ORDER BY insertnum;
```

LOCK TABLE

The Adaptive Server Anywhere LOCK TABLE *name* must be a base table name as SYNONYMs are not supported in Adaptive Server Anywhere. The DB2 syntax is supported under Adaptive Server Anywhere. Unlike DB2, EXCLUSIVE mode prevents ISOLATION 0 (UNCOMMITED READS) read-only queries in Adaptive Server Anywhere.

MERGE

Adaptive Server Anywhere does not support MERGE. You can achieve similar functionality using INSERT and DELETE. The INSERT/DELETE methodology will not provide error messages like the MERGE statement does, and not does not provide flexibility in column level updates. Consider the following MERGE statement:

```
MERGE INTO archive ar
USING ( SELECT activity,description,date,last_modified
FROM activities_groupA )ac
ON ( ar.activity =ac.activity )AND ar.group ='A '
WHEN MATCHED AND ac.date <CURRENT DATE THEN
DELETE
)
```

```
WHEN MATCHED AND ar.last_modified <ac.last_modified THEN
UPDATE SET
( description,date,last_modified )=(
         ac.description,ac.date,DEFAULT )
WHEN NOT MATCHED
INSERT ( description,date,last_modified )
VALUES ( ac.activity,ac.date,DEFAULT )
```

The following Adaptive Server Anywhere syntax provides similar functionality:

```
INSERT INTO archive ( activity,description,date ) SELECT
         activity,description,date,last_modified
FROM activities_group ON EXISTING UPDATE;
DELETE FROM archive where archive.group = 'A' AND
         archive.activity=
( SELECT activity
FROM activities_group WHERE date<CURRENT DATE )
```

Note that INSERT ON EXISTING UPDATE requires the destination table to have a primary key.

RELEASE SAVEPOINT

The optional TO keyword is not supported in Adaptive Server Anywhere and must be removed.

ROLLBACK

Adaptive Server Anywhere syntax only differs from DB2 in syntax where ROLLBACK WORK TO SAVEPOINT *savepoint-name* is used. The optional WORK keyword in this syntax must be removed.

SAVEPOINT

Adaptive Server Anywhere supports SAVEPOINT *savepoint-name*. However, the UNIQUE, ON ROLLBACK RETAIN CURSORS, and ON ROLLBACK RETAIN LOCKS clauses are not supported. You can configure Adaptive Server Anywhere to retain cursors on rollback and commit using the close_on_endtrans database option.

SELECT

This statement is one of the most complex to address for this document. The SELECT statement can be broken down into key components:

WITH common-table-expression

The *common-table-expression* in DB2 identifies the WITH clause. Adaptive Server Anywhere supports the WITH clause.

Select-list notation

No syntax differences found. You can select special registers and functions.

Table-reference

The column list for correlation naming applies to derived tables, which Adaptive Server Anywhere supports. Both DB2 and Adaptive Server Anywhere support the following syntax:

```
SELECT * FROM ( SELECT * FROM department ) AS e ( i, j , k )
```

Using the DB2 tablesampling or *data-change-table-references* clause is not supported in Adaptive Server Anywhere. The ONLY and OUTER clauses apply to typed tables and are not supported in Adaptive Server Anywhere. Note that Adaptive Server Anywhere does not support nicknames or synonyms, but aliases are supported.

The TABLE (*function-name*) constructor is not supported in Adaptive Server Anywhere. However, using stored procedures that return result sets, you can achieve very similar results as Adaptive Server Anywhere supports selecting from a stored procedure. For example:

```
SELECT t.id, t.quantity_ordered AS q, p.name
FROM sp_customer_products( 149 ) t JOIN product p
ON t.id = p.id
```

Joined table syntax

No syntax differences found.

GROUP BY clause

Adaptive Server Anywhere supports basic grouping expressions, GROUP BY ROLLUP, GROUP BY CUBE, and GROUP BY GROUPING SETS. In addition to these constructions, DB2 supports nested grouping sets, which provide a syntactic shorthand for lengthy grouping set specifications. For example, the nested grouping set specification `GROUP BY ROLLUP( Province, ( County, City ) )` is not directly supported by Adaptive Server Anywhere. However, it is equivalent to `GROUP BY GROUPING SETS ( ( Province, County, City ), ( Province ), ( ) )`, which is supported.

ORDER BY clause

The INPUT SEQUENCE and ORDER OF clauses are not supported in Adaptive Server Anywhere. Adaptive Server Anywhere does not support ORDER BY within a UNION, INTERCEPT, or EXCEPT as DB2 does. Simple expressions in the ORDER BY clause are supported in Adaptive Server Anywhere, such as integers and column names.

Fetch-first clause

Adaptive Server Anywhere supports this construct with less verbosity than DB2. Adaptive Server Anywhere supports SELECT FIRST *x*, where *x* is an integer. This clause would be located before the select-list notation clause in Adaptive Server Anywhere, which is different that DB2, where it would be at the end of the statement.

Update-clause

No syntax differences found.

Notes

A different result may occur between Adaptive Server Anywhere and DB2 with a similar query as the combination of key components may have different meanings in Adaptive Server Anywhere and DB2.

Adaptive Server Anywhere supports subqueries, but while DB2 quantified predicates (row value constructors) support multiple columns with these predicates, Adaptive Server Anywhere supports only one column.

☞ For more information, see the Quantified predicates item in the table in "Operators and predicates" on page 7.

Read-only clause

Adaptive Server Anywhere supports FOR READ ONLY, but not FOR FETCH ONLY.

☞ For more information, see the Adaptive Server Anywhere SELECT statement syntax at *http://www.ianywhere.com/developer/product_manuals/-sqlanywhere/0902/en/html/dbrfen9/00000474.htm*.

Optimize-for clause

Similar to DB2, Adaptive Server Anywhere optimizes by default to obtain all the rows in a result set quickly. Different optimization may be used if only a small part of the result set is required. Adaptive Server Anywhere provides this optimization hint with the WITH (FASTFIRSTROW) clause, while DB2 uses OPTIMIZE FOR *x* ROWS, where *x* is an integer.

Isolation clause

The WITH clause can be mapped to the following:

DB2	Adaptive Server Anywhere
WITH RR	WITH SERIALIZABLE
WITH RS	WITH REPEATABLEREAD
WITH CS	WITH READCOMMITED
WITH UR	WITH READUNCOMMITED

Lock-request clause

Adaptive Server Anywhere has table hints that are specified in the WITH clause that can be used to obtain similar locking with a query.

DB2	Adaptive Server Anywhere
USE AND KEEP SHARE LOCKS	WITH (NOLOCK)
USE AND KEEP UPDATE LOCKS	WITH (HOLDLOCK)
USE AND KEEP EXCLUSIVE LOCKS	WITH (XLOCK)

SET CURRENT DEFAULT TRANSFORM GROUP

Adaptive Server Anywhere does not support structured types.

SET CURRENT DEGREE

Adaptive Server Anywhere handles intra-query parallelism automatically and this statement is not supported.

SET CURRENT EXPLAIN MODE

Adaptive Server Anywhere does not require support for explain modes. They are not used in most applications as they are used mainly for performance and tuning.

SET CURRENT EXPLAIN SNAPSHOT

Adaptive Server Anywhere does not require EXPLAIN snapshots. This is used mainly for performance and tuning. They are not used in most applications as they are used mainly for performance and tuning.

SET CURRENT ISOLATION

Adaptive Server Anywhere supports setting isolation levels via SET OPTION ISOLATION_LEVEL. ISOLATION_LEVEL maps to DB2 settings as follows:

DB2	Adaptive Server Anywhere
RR | SERIALIZABLE | REPEATABLE READ	3
RS	2
CS | CURSOR STABILITY | READ COMMITTED	1
UR | DIRTY READ | READ UNCOMMITTED	0
RESET	0

SET CURRENT LOCK TIMEOUT

Adaptive Server Anywhere does not support lock timeouts. You can achieve the same functionality by setting the BLOCKING option to OFF and controlling this setting from your application.

SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION

Adaptive Server Anywhere does not support structured table types.

SET CURRENT PACKAGE PATH

Adaptive Server Anywhere does not require PACKAGE PATHs. This command indicates which schema/user objects/packages should be used with an application. Adaptive Server Anywhere uses groups and group membership to accommodate for specifying a schema (as well as permissions) for accessing database objects.

SET CURRENT PACKAGESET

Adaptive Server Anywhere does not require PACKAGESETs. This command indicates which schema/user objects/packages should be used with an application. Adaptive Server Anywhere uses groups and group membership to accommodate for specifying a schema (as well as permissions) for accessing database objects.

SET CURRENT QUERY OPTIMIZATION

In Adaptive Server Anywhere, you can set the OPTIMIZATION_LEVEL option. The value of this option can be obtained using the following query:

```
SELECT CONNECTION_PROPERTY ( 'OPTIMIZATION_LEVEL' )
```

SET CURRENT REFRESH AGE

Adaptive Server Anywhere does not support materialized views.

SET ENCRYPTION PASSWORD

Adaptive Server Anywhere does not support this manner of encryption.

SET INTEGRITY

Adaptive Server Anywhere does not have a concept of table states and does not require setting INTEGRITY. This statement is not normally used in deployed applications.

SET PASSTHRU

Adaptive Server Anywhere supports similar functionality to SET PASSTHRU using the FORWARD TO statement. The FORWARD TO statement also acts as a SET PASSTHRU RESET.

SET PATH

Adaptive Server Anywhere does not require PACKAGE PATHs.

SET SCHEMA

Adaptive Server Anywhere does not support setting a connection's schema. Adaptive Server Anywhere can control schema qualification via group memberships.

SET SERVER OPTION

Adaptive Server Anywhere provides server capabilities instead of SERVER OPTIONS. These capabilities can be set via the ALTER SERVER *server-name* CAPABILITY *capability-name* { ON | OFF }. Mapping these remote data (federated) features has not been done yet.

SET SESSION AUTHORIZATION | SET SESSION USER

Adaptive Server Anywhere supports setting authorization via SET SESSION AUTHORIZATION or SET USER. The special registers USER, CURRENT USER, and SESSION USER are not supported in Adaptive Server Anywhere. The optional ALLOW ADMINISTRATION clause is not supported in Adaptive Server Anywhere.

SET VARIABLE

The SET statement is supported in both DB2 and Adaptive Server Anywhere. The only difference is that Adaptive Server Anywhere supports the assignment of a single variable, whereas DB2 supports assignment of a single variable or a rowset or tuple of variables. Adaptive Server Anywhere assignments can include full-select where the query returns one single value. If you are using assignments with a single SET on multiple parameters or variables, you can simply use multiple SET statements to work around the difference.

SIGNAL

Adaptive Server Anywhere supports the industry-defined SQLSTATE syntax, whereas DB2 allows for signaling of user-defined errors, as well as predefined errors. If your application requires signaling of user-defined errors, use RAISERROR.

UPDATE

Unless the application is using TYPED TABLES or data change tables, the UPDATE statement will migrate to Adaptive Server Anywhere. Adaptive Server Anywhere supports updating full-selects. It also supports including multiple tables

in the table-list, which allows table joins and multi-table updates. The DB2 FROM ONLY clause applies to TYPED TABLES, which Adaptive Server Anywhere does not support. The include clauses apply to data change tables (or intermediate result set), which Adaptive Server Anywhere does not support. UPDATE with row-like syntax is not supported in Adaptive Server Anywhere (for example, `UPDATE tblname SET ( col1,col2 ) = ( 1,2 ) WHERE col3 =3`), and will have to be reworded.

The assignment clause, where it applies to intermediate result sets, is not supported in Adaptive Server Anywhere. If you were using data change tables in your queries, you may want to consider using triggers and temporary tables for updates/deletes to obtain a desired flow control as a workaround to data change tables. Adaptive Server Anywhere does not support rowset updates: the SET statement must apply to single values rather than a row of values. The WITH clause will need to change. The WITH clause can be mapped to the following:

DB2	Adaptive Server Anywhere
WITH RR	WITH SERIALIZABLE
WITH RS	WITH REPEATABLEREAD
WITH CS	WITH READCOMMITED
WITH UR	WITH READUNCOMMITED

VALUES [INTO]

VALUES is a form of a query that can be replaced with SELECT for queries that return a single record. For example:

```
SELECT 'A' INTO :hostvar
```

Compound statements

Some applications use stored procedures, triggers, or batches. There is a large overlap of supported syntax between Adaptive Server Anywhere and DB2 databases. Error handling with types of SQL–based logic will introduce additional migration considerations. Since DB2 just recently introduced SQL-based procedure language, migrating from an older style DB2 procedure may be difficult if there is a large reliance on the host language. Note that Adaptive Server Anywhere does not support external C/C++-based stored procedures.

BEGIN

Both Adaptive Server Anywhere and DB2 support the dynamic compound statements and much of the syntax is the same. The following syntaxes have been found:

1. DB2 allows DECLARE on CONDITIONS for SQLSTATE, while Adaptive Server Anywhere supports DECLARE EXCEPTION for SQLSTATES. These

SQLSTATE values may differ between products.

2. DB2 allows variables with default values. Adaptive Server Anywhere does not support default values with the variable declarations. You can work around this using the SET assignment statement.

3. For procedural compound statements, Adaptive Server Anywhere differs in the following ways:
   a. RESULT_SET_LOCATOR VARYING is not a valid Adaptive Server Anywhere variable declaration.
   b. Adaptive Server Anywhere does not require declaring SQLCODE or SQLSTATE; these values are always accessible without declaration.
   c. Adaptive Server Anywhere does not support DECLARE STATEMENT in procedure language. This is likely used for procedures written in embedded SQL and is not required for SQL procedures.
   d. Declaration of handlers in IBM DB2 maps roughly to exception-cases (exception handlers) in Adaptive Server Anywhere. Exception handlers provide more control.
   e. DECLARE CURSOR allows for WITH HOLD in DB2. Adaptive Server Anywhere supports WITH HOLD in the OPEN statement. DB2 provides clauses for RETURN to CLIENT or CALLER; Adaptive Server Anywhere allows for return cursors to clients within procedures, assuming you do not open these cursors within the procedure definition. Adaptive Server Anywhere handles the return cursors automatically.

CASE

No syntax differences found.

FOR

The only difference in the syntax between DB2 and Adaptive Server Anywhere is the WITH HOLD clause. The WITH HOLD clause is default behavior for Adaptive Server Anywhere within the FOR statement. The *cursor-name* CURSOR FOR clause is optional in DB2, while Adaptive Server Anywhere requires this clause.

FETCH

No syntax differences found.

GET DIAGNOSTICS

The DB2 GET DIAGNOSTICS ROW_COUNT semantics are similar to the statement `SET variable-name = @@rowcount` in Adaptive Server Anywhere. The DB2 GET DIAGNOSTICS EXCEPTION . . . semantics are similar to `SET variable-name = SQLCODE | SQLSTATE` or `ERRORMSG( )`, depending on the type of error information you want to obtain. DB2 GET DIAGNOSTICS

DB2_RETURN_STATUS semantics can be achieved in Adaptive Server Anywhere using:

```
BEGIN
DECLARE i INT;
i= call xp_cmdshell( 'dirt' );
MESSAGE 'i'||i;
END
```

GOTO

Adaptive Server Anywhere supports two SQL dialects: Watcom-SQL and Transact-SQL. The Watcom-SQL language is similar to the DB2 dialect. The Watcom-SQL language does not support GOTO. The GOTO statement is supported in the Transact-SQL form of the language. You could port it to Transact-SQL or re-code your logic with constructs supported in Watcom-SQL, which can be compatible with both DB2 and Adaptive Server Anywhere.

IF

No syntax differences found.

ITERATE

The ITERATE statement is not supported, but the Transact-SQL CONTINUE statement is supported. The stored procedure must be reworked to use Transact-SQL syntax or you can rework the logic for the Watcom-SQL ANSI form.

LEAVE

No syntax differences found.

LOOP

No syntax differences found.

OPEN

No differences found.

REPEAT / UNTIL

The REPEAT statement is not supported; the stored procedure, batch, or trigger must be reworked to use WHILE/LOOP or a FOR LOOP.

RESIGNAL/SIGNAL

Adaptive Server Anywhere syntax supports the industry-standard SQLSTATE syntax, whereas DB2 allows for signaling of user-defined errors, as well as

predefined errors. If you application requires signaling of user-defined errors, use RAISERROR.

RETURN

Adaptive Server Anywhere syntax supports returning single values from a user-defined scalar value for a function or procedure. The scalar value syntax is consistent between Adaptive Server Anywhere and DB2. IBM DB2 also allows you to return a result set from a function; these functions are typically called **table functions**. To obtain similar functionality to a table function in Adaptive Server Anywhere, consider creating a procedure that returns a result set.

☞ For more information, see "Table functions" on page 23.

WHILE

The Adaptive Server Anywhere LOOP construct is very similar to DB2 WHILE syntax.

☞ For information about LOOP, see "LOOP" on page 21.

Functions, procedures, and triggers

Some applications use stored procedures, trigger, or batches. You will find a large overlap of supported syntax between Adaptive Server Anywhere and DB2 databases. Error handling with types of SQL-based logic will introduce additional migration considerations. Since DB2 only recently introduced SQL-based procedure language, migrating from an older style DB2 procedure may be difficult if there is a large reliance on the host language. While Adaptive Server Anywhere supports external C/C++ stored procedures and functions, these are limited to returning scalar values, and connections cannot be inherited as SQL procedures can. IBM offers external procedures that can inherit a connection environment and return result sets.

Java functions and procedures

The EXTERNAL NAME clause is consistent between Adaptive Server Anywhere and DB2. The function signature is required for Adaptive Server Anywhere as it requires name and parameter signatures to load a Java method, whereas DB2 only requires the JAR, class, and function specification.

Parameters

Adaptive Server Anywhere only has one style of parameter type (SQL), and there is no need to specify the parameter style. DB2 supports clauses for a variety of parameter styles for external stored procedures and functions. Adaptive Server Anywhere simplifies these to SQL parameters and provides syntax for parameter style. As well, Adaptive Server Anywhere supports the CSSID clause.

☞ For detailed information about the recommended data type mappings

between DB2 UDB and Adaptive Server Anywhere, see “Data type conversions: DB2” in the *Adaptive Server Anywhere SQL User’s Guide* at *http://download.sybase.com/pdfdocs/awg0902e/dbugen9.pdf*.

CREATE FUNCTION

Adaptive Server Anywhere supports SQL, Java, and external C/C++ scalar user-defined functions. Adaptive Server Anywhere does not support OLE DB External Table, Sourced, or Template functions. There are a number of descriptive modifiers with DB2 in the creation/alteration of functions. Adaptive Server Anywhere allows modifiers in the (NOT) DETERMINISTIC and EXTERNAL clauses on functions. Other modifiers on functions are not required/supported, such as locking, SQL contents, thread-safety, and so on. Both Adaptive Server Anywhere and DB2 scalar functions provide for a RETURNS clause. Additionally, Adaptive Server Anywhere requires a BEGIN. . . END clause for SQL functions.

Table functions

If you have DB2 table functions, you can port them to Adaptive Server Anywhere stored procedures that return a result set. Consider the following DB2 table function:

```
CREATE FUNCTION DEPTEMPLOYEES ( DEPTNO CHAR( 3 ) )
RETURNS TABLE ( EMPNO CHAR( 6 ),
LASTNAME VARCHAR( 15 ),
FIRSTNAME VARCHAR( 12 ) )
LANGUAGE SQL
READS SQL DATA
NO EXTERNAL ACTION
DETERMINISTIC
RETURN
SELECT EMPNO,LASTNAME,FIRSTNME
FROM EMPLOYEE
```

In DB2, you would invoke it as follows:

```
SELECT * FROM TABLE ( DEPTEMPLOYEES( ’ABC’ ) )
AS functioncall
```

This function could be implemented in Adaptive Server Anywhere as follows:

```
CREATE PROCEDURE DEPTEMPLOYEES ( DEPTNO CHAR( 3 ) )
RESULT(EMPNO CHAR( 6 ),LASTNAME VARCHAR( 15 ), FIRSTNAME
         VARCHAR( 12 ) )
BEGIN
SELECT EMPNO,LASTNAME,FIRSTNME FROM EMPLOYEE
END
```

In Adaptive Server Anywhere, you would invoke the function in one of the following ways:

```
//Option 1
SELECT * FROM DEPTEMPLOYEES( ’ABC’ )
AS functioncall

//Option 2
CALL DEPTEMPLOYEES(  ’ABC’ )
```

CREATE PROCEDURE

Adaptive Server Anywhere and DB2 both offer SQL-based procedures. DB2 introduced SQL-based logic in the version 8.2 release. This section describes differences between the SQL procedure dialects. Typically, if you have non-SQL-based procedures, then the SQL is embedded into the native language, which is very similar to SQL-based stored procedures. This section may help you in this situation. Much of the syntax/behavior is the same between the products aside from the following points:

1. Both Adaptive Server Anywhere and DB2 support returning result sets to the calling program. Adaptive Server Anywhere does not require the clause FOR RETURN TO CLIENT | CALLER. The Adaptive Server Anywhere procedure language specifies the data types with the RESULT clause.
2. Both products support local variables, but Adaptive Server Anywhere does not support default settings, and assignment statements are required.
3. For differences in exception handlers, see “General error handling” on page 25.
4. ALLOCATE CURSOR.
5. DECLARE STATEMENT is not supported in Adaptive Server Anywhere.
6. Declaring SQLCODE and SQLSTATE is not required for Adaptive Server Anywhere and should be removed.
7. For differences between DB2 and Adaptive Server Anywhere in compound statements, see “Compound statements” on page 19.
8. The LANGUAGE SQL and LANGUAGE JAVA modifiers are supported in Adaptive Server Anywhere.
9. Adaptive Server Anywhere does not support the context of the following:
   a. SPECIFIC name
   b. [NOT] FENCED
   c. [NOT] DETERMINISTIC
   d. MODIFIES SQL DATA | CONTAINS SQL | READS SQL DATA
   e. CALLED NULL ON INPUT
   f. PARAMETER CCSID
   g. DYNAMIC RESULT SETS
   h. NO EXTERNAL ACTION
   i. OLD | NEW SAVEPOINT LEVEL
   j. INHERIT SPECIAL REGISTERS
   k. PARAMETER STYLE *any*
   l. PROGRAM TYPE
   m. [NO] DBINFO

Many Adaptive Server Anywhere samples portray stored procedures to only have one BEGIN EXCEPTION, END. It is possible to have multiple compound statements within an Adaptive Server Anywhere procedure if it is necessary to continue when an exception is encountered.

CREATE TRIGGER

Adaptive Server Anywhere and DB2 both support statement-level and row-level triggers. Both databases offer SQL-based triggers. Much of the syntax/behavior is the same between the products aside from the following points:

1. The DB2 NO CASCADE BEFORE clause allows for before triggers, which Adaptive Server Anywhere supports with a BEFORE clause. Adaptive Server Anywhere does not support suppression of cascading triggers.
2. Adaptive Server Anywhere does not support triggers on views or INSTEAD OF triggers. If you have an INSTEAD OF trigger, you can likely rework it to use a base table trigger.
3. For differences between DB2 and Adaptive Server Anywhere in compound statements, see “Compound statements” on page 19.
4. If you are using statement-level triggers, you must change OLD_TABLE to OLD and NEW_TABLE to NEW.
5. Adaptive Server Anywhere requires BEGIN and END as it expects triggers to use compound statements. Labels can occur within these statements.
6. You must remove the MODE DB2SQL clause.

General error handling

SQLSTATES are found in both Adaptive Server Anywhere and DB2 procedural language and ODBC. SQLSTATE definitions within the ODBC specification are defined by a Microsoft specification and conform to ODBC. The ODBC drivers from Adaptive Server Anywhere and DB2 (DB2 CLI) follow this specification. Porting ODBC applications is further discussed in “Porting CLI/ODBC applications” on page 47.

DB2 supports procedural language and supports EXCEPTIONs. Both Adaptive Server Anywhere and DB2 require you to declare exceptions. DB2 provides CONTINUE, ERROR, and UNDO handlers. Adaptive Server Anywhere offers EXCEPTION handlers that map to ERROR or CONTINUE. Adaptive Server Anywhere also provides the ON EXCEPTION RESUME clause in the definition of a procedure to allow for continuance upon exception. DB2 UNDO handlers must be address with additional SAVEPOINT logic in Adaptive Server Anywhere. The syntax differs slightly for exceptions as follows.

DB2

```
DECLARE EXIT HANDLER FOR SQLEXCEPTION
   SIGNAL SQLSTATE value SQLSTATE SET MESSAGE_TEXT = errorLabel;
```

Adaptive Server Anywhere

```
DECLARE EXCEPTION SIGNAL SQLSTATE value;
```

Both products support user-defined errors, however, the syntax differs. Adaptive Server Anywhere provides the RAISERROR statement for user-defined errors rather than the SIGNAL syntax. Both Adaptive Server Anywhere and DB2 support signaling and resignalling exceptions. Adaptive Server Anywhere does not consider warnings as EXCEPTIONs (for example, ROW_NOT_FOUND) and these CONDITIONS or HANDLERs need to be removed. For example, SQLCODE 100 (SQLSTATE 02000) identifies ROW_NOT_FOUND. Adaptive Server Anywhere does not require the declaration of return code handlers such as SQLSTATE or SQLCODE because they are defined implicitly.

While ODBC error codes (SQLSTATES) are well defined in the database community, many applications utilize the underlying native database error, which can be different between vendors such as Adaptive Server Anywhere and DB2. If your application utilizes native errors, you will need to map these errors from DB2 to Adaptive Server Anywhere. Note that IBM's CLI and PHP would follow the ODBC error messages and SQLSTATEs.

Built-in functions

In cases where you do not want to change your application, you can usually create a user-defined function (UDF) to create an Adaptive Server Anywhere function that is equivalent to a DB2 function.

In the following table, for functions that say CAST in the middle column, you can use implicit conversion with the interface library (for example, ODBC).

DB2 function	Closest Adaptive Server Anywhere function	Comments
AVG (ALL | DISTINCT)	AVG (DISTINCT)	ALL is inferred by default.
CORRELATION/CORR	CORR	
COUNT (ALL | DISTINCT)	AVG (DISTINCT)	ALL is inferred by default.
COUNT_BIG (ALL | DISTINCT)		No mapping aside from count, but count returns integer.
COVARIANCE/COVAR	COVAR_POP	Adaptive Server Anywhere also offers COVAR.
GROUPING	GROUPING	This has multiple related clauses that may affect portability.
MAX (ALL | DISTINCT)	MAX (DISTINCT)	ALL is inferred by default.
MIN (ALL | DISTINCT)	MIN (DISTINCT)	ALL is inferred by default.

DB2 function	Closest Adaptive Server Anywhere function	Comments
REGRESSION FUNCTIONS	REGRESSION FUNCTIONS	REGR_ICPT needs to change to REGR_INTERCEPT. Calculations seem to be close
STDDEV (ALL | DISTINCT)	STDDEV_POP	ALL is inferred by default.
VARIANCE/VAR (ALL | DISTINCT)	VAR_POP	
ABS ABSVAL	ABS	ABSVAL could use a UDF referencing ABS.
ACOS	ACOS	
ASCII		
ASIN	ASIN	
ATAN/ATAN2/ATANH	ATAN/ATAN2	ATANH not supported.
BIGINT	CAST	
BLOB	CAST	
CEILING or CEIL	CEILING	UDF if for CEIL.
CHAR	CAST	
CHR	CHAR	
CLOB	CAST	
COALESCE	COALESCE	
CONCAT	STRING	UDF.
COS and COSH	COS	No COSH.
COT	COT	
DATE	DATE	
DAY	DAY	
DAYNAME	DAYNAME	
DAYOFWEEK	DOW	UDF.
DAYOFWEEK_ISO	DOW	UDF using modulus and addition or option.
DAYOFYEAR		UDF.
DAYS	DAYS	Different results
DBCLOB	CAST	Or attempt an implicit conversion.

DB2 function	Closest Adaptive Server Anywhere function	Comments
DBPARTITIONNUM		N/A
DEC | DECIMAL	CAST	UDF for dec DECIMAL is a keyword and CAST will be needed. Attempt implicit conversion.
DECRYPT_BIN | DECRYPT_CHAR	DECRYPT	DECRYPT needs two parameters. DECRYPT_CHAR or DECRYPT_BIN could implemented through the use of database connection variables. Encryption algorithms differ and likely would not work with a heterogeneous environment. CASTING would be needed for a UDF on DECRYPT_CHAR.
DEGREES	DEGREES	
DEREF		N/A
DIFFERENCE	DIFFERENCE	
DIGITS		N/A
DATALINK functions (DLCOMMENT, DLVALUE, and so on)		N/A
DOUBLE	CAST	
ENCRYPT	ENCRYPT	Also see DECRYPT above. HINTS are not supported.
EVT_MON_STATE		N/A
EXP	EXP	
FLOAT	CAST	Alternatively, use implicit conversion.
GETHINT		Not supported.
GENERATE_UNIQUE	NEWID	The implementation or algorithm is different, but the purpose is the similar. Different datatypes would be generated.
GRAPHIC	CAST	Attempt an implicit conversion.

DB2 function	Closest Adaptive Server Anywhere function	Comments
HASHEDVALUE		N/A
HEX		
HOUR	HOUR	
IDENTITY_LOCAL_VAL	@@identity	UDF containing @@identity.
INSERT	INSERT if the third argument is zero	If the third argument is not zero, you need to reword the query.
INTEGER	CAST	
JULIAN_DAY		UDF using `DAYS( '01-01-0001', NOW( * ) )+2451911`.
LCASE/LOWER	LCASE/LOWER	
LEFT	LEFT	Adaptive Server Anywhere is character based, whereas DB2 is byte based.
LENGTH	LENGTH	Adaptive Server Anywhere is character based and does not take the data type into consideration. Consider using expr-type with LENGTH.
LN/LOG	LOG	
LOG10	LOG10	
LOCATE	LOCATE	The first two arguments need to be switched.
LONG VARCHAR	CAST	Alternatively, use an implicit CAST.
LONG VARGRAPHIC	CAST	Alternatively, use an implicit CAST.
LTRIM	LTRIM	
MICROSECOND		Use a UDF can implement this function. For example: `SELECT right( NOW( * )||'000',6 )`.
MIDNIGHT_SECONDS		Consider using the following: `SELECT seconds( '10:10:11' )- seconds( TODAY( * ) )`

DB2 function	Closest Adaptive Server Anywhere function	Comments
MINUTE	MINUTE	
MOD	MOD	
MONTH	MONTH	
MONTHNAME	MONTHNAME	
MULTIPLY_ALT		
NULLIF	NULLIF	
POSSTR	LOCATE	Switch the arguments.
POWER	POWER	
QUARTER	QUARTER	
RADIANS	RADIANS	
RAISE_ERROR	RAISERROR	RAISERROR is not a function. It is a statement.
RAND	RAND	
REAL	CAST	Alternatively, use an implicit CAST.
REC2XML		The query must be reworded with XMLELEMENTS or the FOR XML clause.
REPEAT	REPEAT	
REPLACE	REPLACE if the third argument is zero.	If the third argument is not zero, you must reword the query.
RIGHT	RIGHT	
ROUND	ROUND	
RTRIM	RTRIM	
SECOND	SECOND	
SIGN	SIGN	
SIN	SIN	
SINH		External stored procedure implementation.
SMALLINT	CAST	Alternatively, use an implicit CAST.
SOUNDEX	SOUNDEX	
SPACE	SPACE	

DB2 function	Closest Adaptive Server Anywhere function	Comments
SQRT	SQRT	
SUBSTR	SUBSTR || SUBSTRING	
TABLE_NAME		
TABLE_SCHEMA		
TAN	TAN	
TANH		Implement via an external function.
TIME	CAST	Alternatively, use an implicit CAST.
TIMESTAMP	CAST	Alternatively, use an implicit CAST.
TIMESTAMP_FORMAT	CAST || CONVERT	UDF could also be implemented. The result of CAST depends on the setting of the timestamp_format option.
TIMESTAMP_ISO	CONVERT	Alternatively, use the timestamp_format option.
TIMESTAMPDIFF	DATEDIFF	UDF could be implemented with a DATEDIFF.
TO_CHAR	CAST	A UDF using CAST could also used.
TO_DATE	CAST || CONVERT	A UDF using CAST could also be used. The setting of the date_format database option can influence the format of dates.
TRANSLATE		External function implementation likely required.
TRUNC || TRUNCATE	TRUNCATE || TRUNCNUM	
TYPE_ID		Feature not available.
TYPE_NAME		Feature not available.
TYPE_SCHEMA		Feature not available.
UCASE || UPPER	UCASE || UPPER	

DB2 function	Closest Adaptive Server Anywhere function	Comments
TO_VARCHAR	CAST	A UDF using CAST could also be used. The setting of the timestamp_format option can affect the format of times.
VALUE	COALESCE	
VARCHAR	CAST	Implicit casting could be used.
VARCHAR_FORMAT	CONVERT || CAST	A UDF using CAST could also implicit casting could be used. The date_format database option can also influence the format of dates.
VARGRAPHIC	CAST	Implicit casting could be used.
WEEK	DATEPART	
WEEK_ISO		Could implement a UDF.
YEAR	YEAR	
TABLE_FUNCTIONS	Stored procedures	The TABLE_FUNCTION can be implemented via a stored procedure returning a result set.
OLAP functions	Much of the OLAP syntax is common between Adaptive Server Anywhere and IBM. See comments for exceptions.	See GROUPING in "Built-in functions" on page 26.
XML Functions	XMLAGG, XMLCONCAT, XMLELEMENT, XMLATTRIBUTES, XMLFOREST are supported.	XML2CLOB, XMLSERIALIZE, and XMLNAMESPACE are not supported. Note that XML2CLOB and XMLSERIALIZE are not required for Adaptive Server Anywhere and implicit casting is supported.

Data definition language

This section comments on feature mappings from DB2 to Adaptive Server

Anywhere at the data definition level. Some of the concepts that IBM offers do not fit with low administration expectations of Adaptive Server Anywhere, and as such, some DDL will need to be tweaked. It may be easier to utilize the Migration wizard as described in this document. The purpose of providing this section is to assist you with porting existing scripts.

ALTER TYPE ADD METHOD

Not supported in Adaptive Server Anywhere.

COMMENT

supported in Adaptive Server Anywhere where the object type is supported.

CREATE DISTINCT TYPE

Adaptive Server Anywhere does not support DISTINCT TYPEs. You can create a domain to achieve similar functionality.

☞ For information about Adaptive Server Anywhere user-defined datatypes, see CREATE DOMAIN statement at *http://www.ianywhere.com/developer/product_-manuals/sqlanywhere/0902/en/html/dbrfen9/00000359.htm*.

CREATE FUNCTION

☞ See "Functions, procedures, and triggers" on page 22.

CREATE INDEX

Much of the syntax/behavior is the same between Adaptive Server Anywhere and DB2 aside from the following points:

- Adaptive Server Anywhere does not support the INCLUDE (column-list) clause.
- The CLUSTER clause maps to the CLUSTERED clause in Adaptive Server Anywhere and is positioned before the name of the index.
- SPECIFICATION ONLY applies to federated databases and Adaptive Server Anywhere's remote data access does not support this concept.
- EXTEND USING is not supported as Adaptive Server Anywhere does not support structured types.
- REVERSE INDEX SCANS clauses are not necessary in Adaptive Server Anywhere. All indexes in Adaptive Server Anywhere are candidates for backward scans.
- The PCTFREE and MINPCTUSED clauses are not available in the index definition in Adaptive Server Anywhere.
- PAGE SPLITTING clauses are not valid in Adaptive Server Anywhere.

- Collecting index statistics occurs automatically in Adaptive Server Anywhere.

CREATE NICKNAME | ALIAS

Nicknames are used for federated systems that would apply to the Adaptive Server Anywhere Remote Data Access feature. Proxy tables (CREATE EXISTING TABLE) would be used in lieu of nicknames.

CREATE ALIAS is not required in Adaptive Server Anywhere because views do not have the overhead that IBM DB2 views have. You can create a view in lieu of an alias.

CREATE PROCEDURE

☞ See "Procedures, functions, and triggers" on page 22.

CREATE SCHEMA

The semantics around SCHEMA are different between Adaptive Server Anywhere and IBM DB2. DB2 creates a user implicitly and proceeds to create objects under the new user. Adaptive Server Anywhere uses the current user as the context for object creation. To avoid confusion, connect as the schema name or fully qualify the tables. The SCHEMA name must be an existing database user or group for Adaptive Server Anywhere. The syntax differs in that Adaptive Server Anywhere does not have separate names for SCHEMA and AUTHORIZATION and the token AUTHORIZATION is required. For example:

```
CREATE SCHEMA AUTHORIZATION sample_user
CREATE TABLE sample_user.t1 ( id1 INT PRIMARY KEY )
CREATE TABLE sample_user.t2 ( id2 INT PRIMARY KEY );
```

CREATE SEQUENCE

Adaptive Server Anywhere does not support SEQUENCEs. Use DEFAULT AUTOINCREMENT, DEFAULT GLOBAL AUTOINCREMENT, or get_identity instead of SEQUENCES. This will require code changes.

CREATE SERVER

Adaptive Server Anywhere's CREATE SERVER statement differs in syntax, but provides similar functionality.

CREATE TRANSFORM

Adaptive Server Anywhere does not support structured data types.

CREATE TRIGGER

☞ See "Functions, procedures, and triggers" on page 22.

CREATE TABLE

While the CREATE TABLE statement is complex, you will find that many ANSI standard SQL constructs are consistent between Adaptive Server Anywhere and DB2.

"Data types" on page 5 complements this section.

The following chart provides comments on common syntax and features between Adaptive Server Anywhere and DB2, and also indicates where the feature/syntax may differ.

DB2	Adaptive Server Anywhere closest syntax	Comment
CREATE TABLE LIKE ...	SELECT INTO *new-table-name* FROM *existing-table-name*WHERE 1=0	You can use SELECT INTO *table-name* instead of the CREATE TABLE statement.
CAPTURE DATA CHANGES log extra information	ALTER TABLE *table-name* REPLICATE ON	SQL Anywhere Studio offers three different replication technologies that should be reviewed.
IN *tablespace-name*	IN *dbspace-name*	
CONSTRAINT constraint-name CHECK(*column-name* DETERMINED BY *column-name*)		Not supported.
CONSTRAINT *constraint-name* CHECK(*search-condition*)	CONSTRAINT *constraint-name* CHECK(*search-condition*)	Adaptive Server Anywhere supports search conditions with special registers, system functions, and literals. DB2 may support more. Triggers can be implemented in lieu of CHECK constraints.
CONSTRAINTS .... ENABLE QUERY OPTIMIZATION		Not supported.
CONSTRAINTS .... [NOT] ENFORCED		See wait_for_commit or check_on_commit database options.
GENERATE ALWAYS AS	COMPUTE	

DB2	Adaptive Server Anywhere closest syntax	Comment
DEFAULT GENERATED AS IDENTITY	DEFAULT AUTOINCREMENT	[NO] CYCLE is not supported, and [NO] CACHE, [NO] ORDER is not supported. Adaptive Server Anywhere's global autoincrement may address the following clause requirements: [NO] MINVALUE, [NO] MAXVALUE, START WITH, INCREMENT BY. You can also use get_identity as an alternative.
OPTIONS (federated server options)		☞ For information, see "Working with proxy tables" at *http://www.ianywhere.com/developer/product_manuals/sqlanywhere/0902/en/html/dbugen9/00000611.htm*.

The following list identifies feature categories and associated clauses within CREATE or ALTER TABLE that are not supported in Adaptive Server Anywhere.

1. Materialized views
   a. WITH NO DATA
   b. INCLUDE | EXCLUDE COLUMN DEFAULTS
   c. INCLUDE | EXCLUDE COLUMN ATTRIBUTES
   d. ENABLE QUERY OPTIMIZATION
   e. MAINTAINED BY SYSTEM | USER | FEDERATED TOOL
   f. DATA INITIALLY DEFERRED
   g. REFRESH INITIALLY DEFERRED
2. Structured tables and types clauses
   a. OF
   b. INLINE LENGTH
   c. HIERARCHY
   d. SCOPE
   e. UNDER
   f. INHERIT SELECT PRIVILEGES
   g. REF
   h. REF IS .. USER GENERATED

3. Physical layout of the table
   a. NOT LOGGED INITIALLY
   b. Partitioning KEY
      i. REPLICATED
   c. COMPRESS SYSTEM DEFAULT
   d. LONG IN
   e. INDEX IN
   f. DATALINK storage options
      i. LINK TYPE URL, LINK CONTROL, FILE LINK CONTROL, INTEGRITY, ALL, READ PERMISSION DB | FS, WRITE PERMISSION BLOCKED | FS | ADMIN [NOT] REQUIRING TOKEN UPDATE , RECOVERY YES | NO, ON UNLINK RESTORE | DELETE, MODE DB2OPTIONS
   g. LOB storage options
      i. [NOT] COMPACT, [NOT] LOGGED
   h. CCSID
      i. Controlled by dbinit
   i. ORGANIZE BY DIMENSION | KEY SEQUENCE DB2's key-sequence-clause. DISALLOW | ALLOW OVERFLOW
4. WITH RESTRICT ON DROP

CREATE TABLESPACE

The Adaptive Server Anywhere dbspace is the closest construct to a DB2 tablespace. It allows administrators to locate databases across devices. Dbspaces support operating system managed files rather than raw partitions. The dbspace page size is based on how you initialize the database. The EXTENTSIZE and PREFETCH size parameters are automatically determined by the Adaptive Server Anywhere database server. To calibrate the transfer rates on a dbspace, you can use the ALTER DATABASE statement. Adaptive Server Anywhere database recovery is designed to be automatic and cannot be set. Creation of temporary dbspaces is not required for Adaptive Server Anywhere.

☞ For more information, see CREATE DBSPACE at *http://www.ianywhere.com/developer/product_manuals/sqlanywhere/0902/en/html/dbrfen9/00000357.htm*.

CREATE TYPE

Adaptive Server Anywhere does not support structured types.

CREATE USER MAPPING

☞ For a similar concept, see CREATE EXTERNLOGIN statement at *http://www.ianywhere.com/developer/product_manuals/sqlanywhere/0902/en/html/dbrfen9/00000363.htm*.

CREATE VIEW

Adaptive Server Anywhere and DB2 share the following clauses for the CREATE VIEW statement: listing columns of the view, and WITH CHECK OPTION. Much of the CREATE VIEW syntax is based on the SELECT statement.

☞ For more information, see "SELECT" on page 13.

CREATE WRAPPER

Adaptive Server Anywhere provides predefined CLASSes for remote data access. This syntax is not applicable to Adaptive Server Anywhere.

DROP

supported in Adaptive Server Anywhere where the object type is supported.

RENAME

For renaming a table, ALTER TABLE can rename a table; the syntax changes would have the TO keyword removed and the ALTER TABLE keyword added. For renaming an index, you must add the ALTER keyword to the ALTER INDEX syntax.

RENAME TABLESPACE

ALTER DBSPACE allows you to change the location of the file for the DBSPACE. To truly rename a dbspace, you must drop and recreate a dbspace; the original file cannot be renamed.

Administration language

While this section discusses some of the lower-level administrative functions between the products, you may benefit from not including some of these clauses as SQL Anywhere's approach to administrative language attempts to reduce complexity. The following sections describe some administrative SQL and how it applies to Adaptive Server Anywhere.

CREATE BUFFERPOOL

Adaptive Server Anywhere administration does not require the additional administration requirements of BUFFERPOOLs. Adaptive Server Anywhere does allow you to resize the cache dynamically on Windows and UNIX.

CREATE DATABASE PARTITION GROUP

Adaptive Server Anywhere does not support database partitioning.

CREATE EVENT MONITOR | SET EVENT MONITOR STATE | FLUSH EVENT MONITOR

Event monitoring with DB2 offers functionality of performance tuning as well as system maintenance/monitoring. The CREATE EVENT MONITOR statement in DB2 primarily allows the capture of relevant statistics for a particular event. Other system monitoring from DB2 provides maintenance and monitoring.

The design goals for performance tuning drastically differ between Adaptive Server Anywhere and DB2. Adaptive Server Anywhere is designed to provide performance out of the box, whereas DB2 offers tools that enable DBAs to monitor performance, such as CREATE EVENT.

Adaptive Server Anywhere does offer the ability to monitor the database operation through SQL properties (in Sybase Central they are referred to as statistics) using the PROPERTY function, as well as the sa_conn_properties, sa_db_properties, and sa_eng_properties stored procedures.

Some database maintenance is automated through the Adaptive Server Anywhere CREATE EVENT statement. Database maintenance activities, such as checking for available diskspace, running backups, and reorganizing tables are common between Adaptive Server Anywhere and DB2.

FLUSH PACKAGE CACHE

Adaptive Server Anywhere supports flushing the total cache, which may be useful for testing, using the sa_flush_cache() procedure.

REFRESH TABLE

Adaptive Server Anywhere does not support materialized views.

GRANT/REVOKE

Adaptive Server Anywhere supports a subset of database authorities, routines, and the table/view privileges. Adaptive Server Anywhere does not require/support index, package, schema, sequence, server, and tablespace privileges.

Database authority privileges roughly map the following permissions between Adaptive Server Anywhere and DB2.

DB2	Adaptive Server Anywhere	Notes
CONNECT	CONNECT	
CREATETAB	RESOURCE	
CREATE_EXTERNAL_-ROUTINE	RESOURCE	
CREATE_NOT_FENCED_-ROUTINE	RESOURCE	Adaptive Server Anywhere does not support the concept of a FENCED routine.
DBADM	DBA	
LOAD		Permissions are specified by a server option.
QUIECSE_CONNECT		

The ON DATABASE clause must be removed when migrating to Adaptive Server Anywhere. The USER and GROUP keywords are not included in the Adaptive Server Anywhere syntax. Adaptive Server Anywhere automatically determines whether the permissions are for a user or a group.

Routine privileges are applicable to stored procedures and functions in Adaptive Server Anywhere. Methods are not supported. The keywords FUNCTION and PROCEDURE are not available in this syntax as the database determines the type automatically.

Table or view privileges roughly map to the following permissions between Adaptive Server Anywhere and DB2. The USER and GROUP keywords are not included in the Adaptive Server Anywhere syntax for this statement. The WITH GRANT OPTION is supported in both databases.

DB2	Adaptive Server Anywhere	Notes
ALL	ALL	
CREATETAB	ALTER	

DB2	Adaptive Server Anywhere	Notes
CONTROL		Similar to GRANT ALTER, DELETE, INSERT, REFERENCES, SELECT, UPDATE to. The main difference is that the use of utility functions (specifically LOAD, REORG, RUNSTATS, and SET INTEGRITY) would be granted. In general, Adaptive Server Anywhere controls these utility functions automatically, otherwise, they are administered via a database-wide option. REORGANIZE table in Adaptive Server Anywhere requires DBA or table owner permissions.
INDEX	REFERENCE	
INSERT	INSERT	
REFERENCES	REFERENCES	
SELECT	SELECT	
UPDATE	UPDATE	

Additionally, the BY ALL syntax is not supported in Adaptive Server Anywhere and needs to be removed for REVOKE statements.

Using the Migration wizard

Migrating data from IBM DB2 to Adaptive Server Anywhere is a straightforward process, with minor issues occurring only if you are using the DB2-specific data types that do not have a SQL mapping such as structured types, reference types, XML, spatial, and DATALINK data types. If you use of these data types, you may have some additional considerations when migrating your DB2 database to Adaptive Server Anywhere. Database migration can be accomplished using the Migrate Database wizard from Sybase Central. Alternatively, a more customized migration can be done using the sa_migrate set of stored procedures in Adaptive Server Anywhere. The DB2 EXPORT command, coupled with the Adaptive Server Anywhere LOAD TABLE statement, could also be used to migrate the data. Note that if you use the DATALINK data type, you may have additional considerations when migrating your DB2 database to Adaptive Server Anywhere.

☞ For detailed information about the recommended data type mappings between DB2 UDB and Adaptive Server Anywhere, “Data type conversions: DB2”

at *http://www.ianywhere.com/developer/product_manuals/sqlanywhere/0902/en/-html/dbugen9/00000654.htm*.

Requirements

- This section assumes you have a DB2 database running on any of the supported platforms and Adaptive Server Anywhere 9.0.1 or later installed on any of the supported Windows platforms.
- If you have not created the DB2 sample database, you can run db2sample and walk through the migration steps.
- The DB2 ODBC 3.5.2 (or later) driver must be installed on the computer running the Adaptive Server Anywhere database.

Creating an Adaptive Server Anywhere database

You must first create an Adaptive Server Anywhere database to migrate the DB2 database to. The following steps explain how to create a new database using Sybase Central.

❖ **To create an Adaptive Server Anywhere database in 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.
   a. In the left pane of Sybase Central, select Adaptive Server Anywhere 9.
   b. In the right pane, click the Utilities tab.
   c. Double-click Create Database.
      The Create Database wizard appears.
   d. Follow the instructions in the wizard to create a new database.

Creating a data source for the DB2 database

The migration process requires an ODBC connection to the source database. Therefore, you need to create an ODBC data source (DSN) for the DB2 database.

❖ **To create an ODBC data source for your DB2 database**

1. Start the ODBC Administrator. From the Start menu, choose Programs ➤ SQL Anywhere 9 ➤ Adaptive Server Anywhere ➤ ODBC Administrator.
   The ODBC Data Source Administrator dialog appears.
2. Click Add.
   The Create New Data Source wizard appears.
3. Select the iAnywhere Solutions 9 - for DB2 Wire Protocol driver from the list of available drivers and then click Finish.
   The DB2 Wire Protocol Driver Setup dialog appears.

4. Type a name for the data source in the Data Source Name field. For example, name the data source **DB2Migrate**.
5. Type the appropriate values in any other fields required for your DB2 database.
6. On the ODBC tab, click Test Data Source to ensure you have configured the data source correctly.
7. Click OK.

Migrating a DB2 database to Adaptive Server Anywhere

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.

❖ **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.

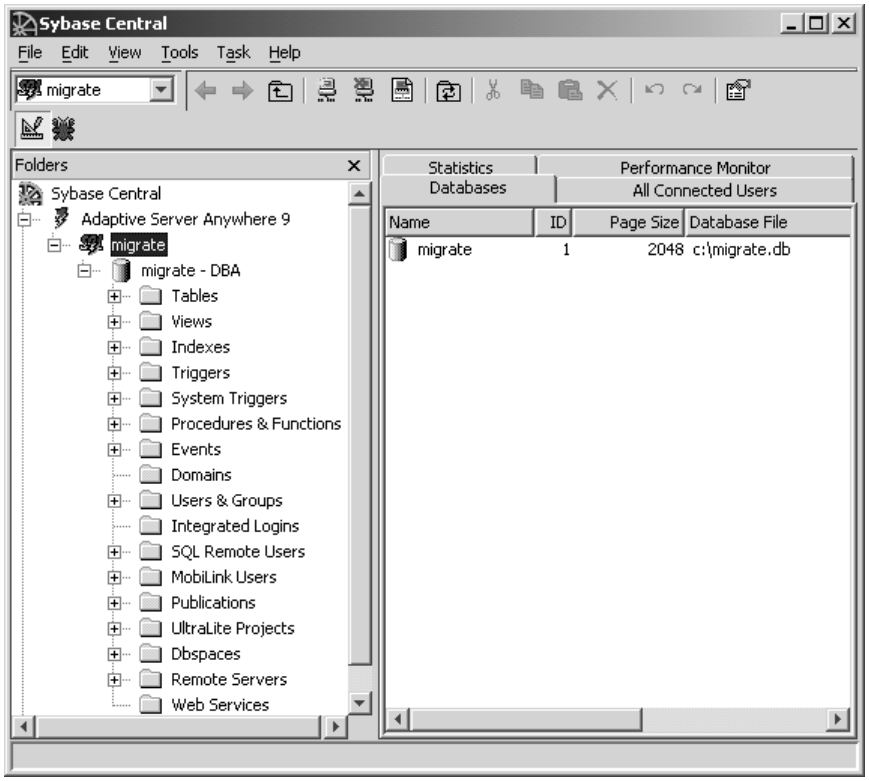
   The Adaptive Server Anywhere database server starts automatically.

The next step is to tell Sybase Central where to find the DB2 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 DB2 database.
   a. On the first page of the wizard, type a name for the remote server, for example, DB2Migrate, and then click Next.
   b. Choose Generic as the type of remote server. Click Next.
   c. Select the Open Database Connectivity (ODBC) option and type the name of the ODBC data source for your DB2 database in the connection information field. For example, if you named your ODBC data source DB2Migrate when you created it, type **DB2Migrate** 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, but your DB2 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.

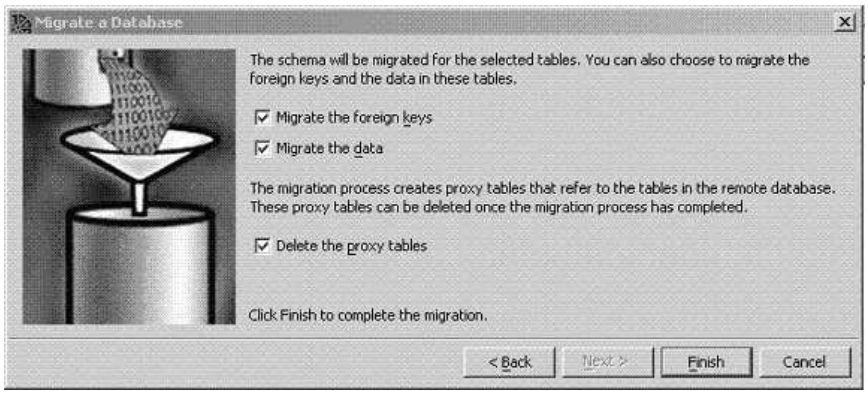
   The External Login Creation wizard appears.
4. Select the user you are currently connected as from the list of users.
5. Type the name of a user in the DB2 database in the Login Name field. Type the password for this user in the Password and Confirm Password fields. Click Finish.

❖ **To migrate the DB2 database**

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

   The Migrate Database wizard appears.
3. Click Next on the introductory page.
4. Select the current database and then click Next.
5. Select the DB2 remote server you created, for example, DB2Migrate, and then click Next.
6. Click Select All and then click Next to migrate all the DB2 tables to the Adaptive Server Anywhere database.

7. Select the Adaptive Server Anywhere database user you want to own the tables. Click Next.
8. Select the options you want to use for migration.



9. Click Finish to start the migration.

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

Migrating an application from DB2 to Adaptive Server Anywhere

Application migration from DB2 to Adaptive Server Anywhere depends on the interface used to access your DB2 application. The following sections discuss some of the more popular interfaces that should require only minimal work to migrate.

Commenting on connections

In DB2, the concept of database alias is very much like an ODBC data source. Adaptive Server Anywhere supports connecting via an ODBC data source for all supported APIs. If your application uses a DB2 database alias, moving the connection code from the DB2 alias to Adaptive Server Anywhere and an ODBC data source is quite easy. Most of the Adaptive Server Anywhere APIs respect ODBC datasource information, making it very easy to port connections within applications.

Porting CLI/ODBC applications

Both Adaptive Server Anywhere and DB2 support the ODBC API specification. Generally, migration of these applications involves changing the ODBC data source to point to Adaptive Server Anywhere instead of DB2. 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.

Both DB2 and Adaptive Server Anywhere provide ODBC API support in C/C++. The DB2 documentation also refers to this API as CLI. Below you should find comments on the CLI function differences between Adaptive Server Anywhere and DB2 within the C/C++ language. This section, along with “Data manipulation language” on page 9, will help you when porting applications from DB2. The following CLI functions are not supported in Adaptive Server Anywhere:

- SQLBindFiletoCol
- SQLBindFiletoParam
- SQLExtendedBind
- SQLExtendedPrepare
- SQLGetDataLinkAttr
- SQLGetLength
- SQLGetPosition
- SQLGetSQLCA
- SQLGetSubstring

- SQLNextResult
- SQLSetCollAttributes
- SQLSetConnection

Porting Java applications

DB2 has a universal driver that acts as a type 2 and type 4 JDBC driver (100% Java implementation), which supports JDBC version 3.0. To migrate to the Adaptive Server Anywhere equivalent, the Sybase jConnect driver should be used if you require a 100% Java implementation. 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.

Currently Adaptive Server Anywhere does not have a JDBC 3.0 driver. If you are using SAVEPOINTs or HOLDABLE cursors (version 3.0 specific), Adaptive Server Anywhere has SQL savepoint syntax and options for keeping cursors open on transaction completion (CLOSE_ON_ENDTRANS). If you are using SQLJ, Adaptive Server Anywhere does not support SQLJ. Similar to DB2, Adaptive Server Anywhere offers a Type 4 driver (jConnect), which has the class name com.sybase.jdbc2.jdbc.SybDriver, and a Type 2 JDBC driver, the iAnywhere JDBC driver, which has the class name ianywhere.ml.jdbcodbc.Idriver. The URLs differ for these drivers.

Porting PHP applications

Migrating a PHP application from DB2 to Adaptive Server Anywhere is simple.

You have the option of using ODBC to connect to Adaptive Server Anywhere or using the SQL Anywhere PHP module provided by iAnywhere Solutions (available for download from the iAnywhere Developer website at *http://www.ianywhere.com/developer/code_samples/sqlany_php_module.html*).

IBM samples utilize ODBC PHP. Both Adaptive Server Anywhere and DB2 offer ODBC drivers, so if the PHP application is already using ODBC to connect to the DB2 database, then there is no need to change the function calls.

If you decide to use the Adaptive Server Anywhere-specific PHP driver, the following table includes PHP application-related functions (rather than administrative/design related functions) mappings from DB2 to the SQL Anywhere PHP driver. The following table includes application related function (rather than administrative/design-related functions) mappings from DB2 to Adaptive Server Anywhere or ODBC.

DB2	Adaptive Server Anywhere
db2_autocommit	sqlanywhere_set_option($conn, “auto_commit”, “off”);
db2_bind_param	

DB2	Adaptive Server Anywhere
db2_close	sqlanywhere_disconnect
db2_commit	sqlanywhere_commit
db2_conn_error	sqlanywhere_errorcode
db2_conn_errormsg	sqlanywhere_error
db2_connect	sqlanywhere_connect
db2_cursor_type	
db2_exec	sqlanywhere_query
db2_execute	sqlanywhere_query or sqlanywhere_execute Note that parameter options for cursors can be addressed with clauses with the SELECT or SET OPTION satement.
db2_fetch_array	sqlanywhere_fetch_array For relative fetching, see sqlanywhere_data_seek.
db2_fetch_assoc	sqlanywhere_fetch_array
db2_fetch_both	sqlanywhere_fetch_array
db2_fetch_object	sqlanywhere_fetch_object, sqlanywhere_fetch_field
db2_fetch_row	sqlanywhere_fetch_row db2_result would be managed as an array.
db2_field_displaysize (not sure if this is design)	
db2_free_result	sqlanywhere_free_result
db2_free_stmt	sqlanywhere_free_result
db2_next_result	
db2_num_fields	sqlanywhere_num_fields
db2_num_rows	sqlanywhere_num_rows
db2_pconnect	sqlanywhere_pconnect
db2_prepare	
db2_result	see db2_fetch_row above
db2_rollback	sqlanywhere_rollback
db2_special_columns	
db2_stmt_error	sqlanywhere_error sqlanywhere_errorcode
db2_stmt_errormsg	sqlanywhere_error sqlanywhere_errorcode

Design/administrative functions like db2_primary_keys, db2_foreign_keys, db2_columns, and db2_tables can be ported via sqlanywhere_query with queries against system tables or system procedures like sp_columns.

Porting Perl applications

Migration of Perl applications from DB2 to Adaptive Server Anywhere is very simple. You may be using the DBD::ODBC driver or DBD:DB2 driver with DB2. In either case, you will likely need to change the connection string using the DBD::ODBC or DBD:ASAny driver or using the native Adaptive Server Anywhere driver (called DBD::ASAny) that ships with Adaptive Server Anywhere.

Once your connection string change is complete, there may be some minor tweaks required to deal with the differences between Adaptive Server Anywhere and DB2 as discussed in DML sections of this paper, but minimal work is required to complete the migration.

Porting .NET, ADO, or RDO applications

Many of these interfaces are well defined on what function calls can be made, so much of the porting is based on common DML. SQL and error handling will determine what efforts need to be applied; “Data manipulation language” on page 9 is helpful if you are porting applications from DB2 ADO, ADO.NET, or RDO to Adaptive Server Anywhere.

For ADO and OLE DB, you will likely need to change in the datasource name to something appropriate for Adaptive Server Anywhere. The OLE DB provider can stay the same if you were using MSDASQL. If you were using IBMDADB2, use ASAPROV as it is the name of the Adaptive Server Anywhere OLE DB provider.

For ADO.NET applications, you will likely need to change the datasource name to something appropriate for Adaptive Server Anywhere. Changing an ADO.NET application will require a few search and replaces.

DB2 .NET provider	Adaptive Server Anywhere .NET data provider
Imports IBM.Data.DB2	Imports iAnywhere.Data.AsaClient
DB2Type.Clob	AsaDbType.LongVarchar
DB2Type	AsaDbType
DB2Type.Blob	AsaDbType.LongBinary
DB2	Asa
DB2Connection	ASAConnection
DB2Transaction	ASATransaction
DB2DataReader	ASADataReader
DB2Command	ASACommand
.GetDate	.GetDateTime

DB2 .NET provider	Adaptive Server Anywhere .NET data provider
Server= (for Connections)	EngineName=
Database=	DataSourceName=

RDO and ADO applications simply use ODBC, so changing the ODBC datasource name is likely sufficient.

Porting embedded SQL applications

Both DB2 and Adaptive Server Anywhere provide a dynamic and static embedded SQL API in C/C++. The ANSI standard applies to the embedded SQL API, which makes much of the static syntax common between Adaptive Server Anywhere and DB2. Below you will find comments on the static embedded SQL supported in Adaptive Server Anywhere and DB2 within the C/C++ language. The SQL Descriptor Area (sqlda) is supported in both products for dynamic SQL. The porting effort becomes more complex and more challenging as there are different C/C++ defines for SQL to C types, and different error definitions for dynamic SQL. This section, along with “Data manipulation language” on page 9, will help you if you are porting DB2 static embedded SQL applications to Adaptive Server Anywhere.

The Adaptive Server Anywhere embedded SQL preprocessor (the sqlpp utility) offers the same functionality as the prep command. Sqlpp preprocesses embedded SQL files into C/C++ without a connection to the database.

Adaptive Server Anywhere expects a global SQLCA, so you will likely have to comment out individual sqlca declares with SQL functions using SQL. Additionally, checking for a valid SQLCA and a finalization on SQLCA, as follows, is recommended:

```
If (!db_init(&sqlca) ) {
Return FALSE;
}
...
dbfini(&sqlca);
```

Both products support multi-threaded applications. Adaptive Server Anywhere uses a syntax EXEC SQL SET SQLCA sqlca to control which SQLCA is to be used for the active request.

BEGIN DECLARE SECTION/END DECLARE SECTION

Adaptive Server Anywhere supports this syntax. Note that Adaptive Server Anywhere allows C/C++ primitive datatypes within this section. If you are using “SQL TYPE is . . . ”, you will want to port to DECL_LONGVARCHAR or DECL_LONGBINARY instead of BLOB and CLOB. Adaptive Server Anywhere does not currently support BLOB files, CLOB files, or LOB indicator values.

COMPOUND SQL

Adaptive Server Anywhere does not support compound statements in Embedded SQL. It is expected that this type of SQL would have been used in Embedded SQL to create stored procedures.

☞ If this case applies to your application, see "Functions, procedures, and triggers" on page 22.

EXPLAIN

The EXPLAIN statement is quite different in the nature of the clauses. Query tuning is different with Adaptive Server Anywhere and DB2 as a result of the different approaches to administration requirements.

EXECUTE

DB2 enables you to one EXECUTE statement for INPUT and OUTPUT descriptors. Adaptive Server Anywhere would need two EXECUTE statements to accomplish the same operation.

EXECUTE IMMEDIATE

No syntax differences found.

INCLUDE

Both Adaptive Server Anywhere and DB2 support EXEC SQL INCLUDE SQLCA | SQLDA. However, IBM samples typically provide the C/C++ form of 'include "sqlca.h"'. Adaptive Server Anywhere only supports the embedded SQL form of the INCLUDE. The Adaptive Server Anywhere SQLCA is a global external, whereas the DB2 SQLCA is not external. You may need to relocate your SQLCA for use with Adaptive Server Anywhere. Note that thread-safe applications will have additional considerations.

☞ For more information, see SET SQLCA at *http://www.ianywhere.com/-developer/product_manuals/sqlanywhere/0902/en/html/dbrfen9/00000482.htm*.

OPEN

No differences found.

PREPARE

Adaptive Server Anywhere has a different order for the FROM clause of this statement, and additionally requires the DESCRIBE keyword. DB2 enables you to use one PREPARE for INPUT and OUTPUT descriptors. Adaptive Server

Anywhere would need two PREPARE statements to accomplish the same operation. Consider the following DB2 syntax:

EXEC SQL PREPARE *statement-name*
OUTPUT INTO *result-descriptor-name*
FROM *host-variable;*

The following Adaptive Server Anywhere syntax would be equivalent.

```
EXEC SQL PREPARE statement-name FROM host-variable
DESCRIBE OUTPUT INTO result-descriptor-name;
```

RELEASE

While there is no release pending state within Adaptive Server Anywhere, Adaptive Server Anywhere's DISCONNECT syntax should suffice for most porting cases.

SELECT

☞ For more information, see "SELECT" on page 13.

SELECT INTO

supported in Adaptive Server Anywhere.

☞ For more information, see "SELECT" on page 13.

SET CONNECTION

Adaptive Server Anywhere uses a connection name, whereas DB2 uses a server name. The syntax between the two products does not differ.

VALUES INTO

☞ See "VALUES [INTO]" on page 19.

WHENEVER

Adaptive Server Anywhere's WHENEVER statement supports NOT FOUND as a single token (NOTFOUND) and GO TO as a single token (GOTO).

Building embedded SQL

At a high level, the DB2 bldapp for the samples appears in the following text box.

```
Db2 prep myfile.sqc bindfile
```

The Adaptive Server Anywhere preprocessor command is

```
Sqlpp myfile.sqc myfile.cpp
```

The library that you need to compile and link to is dblib9 is found in the *win32\lib* or *lib* (UNIX) folders.

☞ See the *samples* directory in your SQL Anywhere Studio installation for the build process.

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