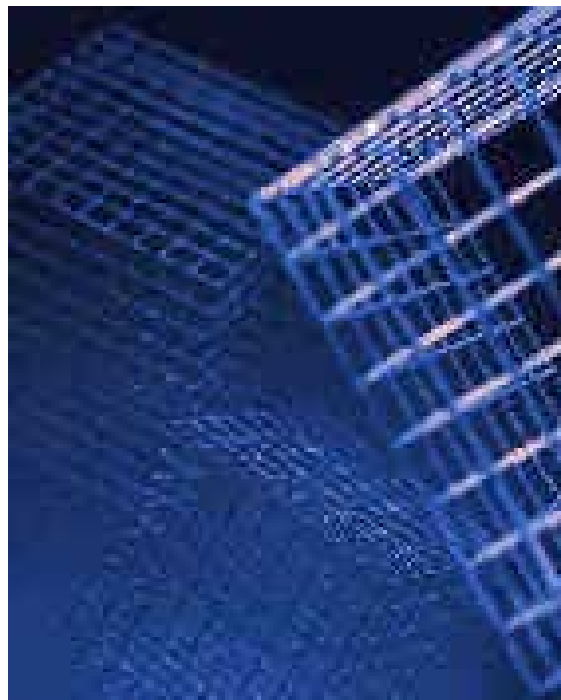


SAP DB as a UNICODE Database



Version 7.3 upwards









Copyright

© Copyright 2001 SAP AG.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation.

For more information on the GNU Free Documentaton License see <http://www.gnu.org/copyleft/fdl.html#SEC4..>

Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax
	Tip

Contents

Copyright	2
Icons.....	3
Code Attribute UNICODE.....	5
Installing a Unicode-Enabled Database	5
Installation Using the Database Manager GUI	5
Installation Using the Database Manager CLI.....	6
Unicode and SQL.....	6
Usage in Data	6
Usage in SQL Statements	6
Usage in Names	7
UNICODE in Programming Languages	7
JDBC	7
C/C++ Precompiler.....	7
ODBC, Python, Perl	8

Code Attribute UNICODE

As a rule, the data types CHAR ASCII and CHAR EBCDIC are more suited to the character sets used for English and Central European languages. Other character sets usually use a code attribute for CHAR data types. This code attribute uses a presentation code that is different to ASCII and EBCDIC, even for internal storage in a database system. This causes problems if you want to access these database systems using a different character set or if you want to exchange data between database systems with different character sets. You can avoid these problems using internal CHAR coding based on UNICODE. UNICODE uses two bytes for each character.

For this reason, SAP DB supports the code attribute UNICODE for the data type CHAR and is able to display various presentation codes in UNICODE format.

As well as storing data in UNICODE, you can also store the names of database objects (for example, table or column names) in UNICODE and display these with the database tools in the desired presentation code.

Installing a Unicode-Enabled Database

You can govern whether or not a database instance should be UNICODE-enabled when you create it. To do this, set the parameter `_UNICODE` to **YES**.



Please note that you cannot change the `_UNICODE` parameter once you have set it.

You can also set the system's default value. If you set the `DEFAULT_CODE` parameter to **UNICODE**, any columns of type CHAR, VARCHAR, and LONG for which no other code attribute has been defined, become UNICODE columns.



Column definition	Result
CHAR (n) UNICODE	UNICODE column
CHAR (n)	UNICODE column
CHAR (n) ASCII	ASCII column
CHAR (n) BINARY	Binary column



Please note that SAP DB UNICODE data is stored internally in UCS-2 format. This, however, doubles the amount of memory space needed for storing UNICODE data to the database instance.

Installation Using the Database Manager GUI

You define whether or not the database instance you want to create is UNICODE-enabled in step 5 (*Parameters*) of the Database Wizard. To make the database instance UNICODE-enabled, select *Extended* and set the `_UNICODE` parameter to **YES**.



For a more detailed description of how to create a database instance using the DBMGUI, please see [Database Manager GUI: SAP DB 7.3](#).

Installation Using the Database Manager CLI

A script for configuring database instances using the Database Manager CLI contains several lines for defining parameters. At this point, add the following line:

```
param_put _UNICODE YES
```



For a more detailed description on how to configure a database instance using the DBMCLI, please see [Database Manager CLI: SAP DB 7.3](#).

Unicode and SQL

Usage in Data

There are three additional data types used to support UNICODE:

CHAR (n) UNICODE

VARCHAR (n) UNICODE

LONG UNICODE

The archive containing the examples (`sapdbuni.tgz`) contains a Java class that can be used to display the results of various column definitions.

Displaying the Column Definition of a Table

```
java TableDef <jdbcurl> <tablename>
```

Creating a Temporary Table with the Specified Column Definitions and Displaying these Column Definitions

```
<command> ::= java TableDef <jdbcurl> <tablename> <column definition>
```

```
<jdbcurl> ::= jdbc:sapdb:<DBNAME>?user=<username>&password=<password>
```



```
java TableDef jdbc:sapdb:TST?user=TEST&password=TEST DUMMY a
varchar (20)
```

```
Table: DUMMY
```

```
A: VARCHARASCII (20)
```

Usage in SQL Statements

SQL statements can contain both UNICODE literals and UNICODE identifiers. The prerequisite for this is a UNICODE-enabled client (currently C/C++-Precompiler and JDBC).

Usage in Names

You can specify your SQL identifiers (user name, table name, etc.) in UNICODE. Hence, all the relevant columns of the system tables are data type UNICODE.

UNICODE in Programming Languages

JDBC

Since Java works with UNICODE strings, it can read and write UNICODE columns.

If you want to use UNICODE in SQL statements too, set the `unicode connect` property to `true`. SQL statements are then transferred to the database instance in UCS-2 format. If the transfer package for a statement is not large enough, you can increase its size using the database parameter `_PACKET_SIZE`.

C/C++ Precompiler

The C/C++ Precompiler checks whether or not the database instance is in UNICODE mode while it is connecting. If it is in UNICODE mode, all SQL statements are transferred as UNICODE.



```
EXEC SQL BEGIN DECLARE SECTION;

/* "SELECT TABLENAME FROM DOMAIN.TABLES " encoded in UCS2 */
SQLUCS2 sqlstmt[36] = {0x0053, 0x0045, 0x004C, 0x0045, 0x0043,
                      0x0054, 0x0020, 0x0054, 0x0041, 0x0042,
                      0x004C, 0x0045, 0x004E, 0x0041, 0x004D,
                      0x0045, 0x0020, 0x0046, 0x0052, 0x004F,
                      0x004D, 0x0020, 0x0044, 0x004F, 0x004D,
                      0x0041, 0x0049, 0x004E, 0x002E, 0x0054,
                      0x0041, 0x0042, 0x004C, 0x0045, 0x0053,
                      0x0000};

SQLUCS2 resultstring[64];
EXEC SQL END DECLARE SECTION;

/* connect ... */

/* parse a unicode sql command and give it a statement name */
EXEC SQL PREPARE stmt1 FROM :sqlstmt;

EXEC SQL DECLARE curs1 CURSOR FOR stmt1;
```

```
EXEC SQL OPEN curs1;

/* loop over resultset */
while (sqlca.sqlcode != 100)
{
    EXEC SQL FETCH curs1 INTO :resultstring;

    /* ... */
}

EXEC SQL CLOSE curs1;
```

For the complete example see the file `HelloUnicodeDB.cpc` in the archive `sapdbuni.tgz` that contains all the example files.

- The data type `SQLUCS2` is a positive 16 bit integer (Unsigned Short)
- Input and output variables can be data type `CHAR []` or `SQLUCS2 []`, the values will be transformed internally into the suitable data type if required
- The SQL statement can be data type `CHAR []` or `SQLUCS2 []`

ODBC, Python, Perl

ODBC	Unicode not supported	Planned for second quarter of 2001
Python	Unicode not supported	Planned for second quarter of 2001
Perl	Unicode not supported	Planned for second quarter of 2001