SDN Contribution

SAP NetWeaver J2EE Preview: Working with Database Tables and DataSources

Applies to:

SAP NetWeaver J2EE Preview

Summary

Find out how to create custom database tables and connect your J2EE applications to them using custom DataSources.

Author(s): SAP NetWeaver Product Management

Company: SAP AG

Created on: April 2006

Table of Contents

Overview

The purpose of this guide is to intoduce the basics of working with database tables and deploying custom DataSources for your applications developed with the SAP NetWeaver J2EE Preview. It starts by introducing the preconfigured tables and DataSource that you might use out-of-the-box and then goes on to elaborate further on how you can create your own tables and DataSources using the tools provided in the preview.

Predefined Tables in the MaxDB Database

The best way to explore the capabilities of the J2EE Preview is, of course, to try and write your own applications using the samples that come with the SAP NetWeaver Developer Studio as a basis. Each of those samples comes with its own database content, so you could easily use the very same tables that we created for the persistence of your own applications. To give you an overview of the predefined content of the database, here is the script that we used to create all of them: demotables.sql.

This script created the tables in a special database schema for the newly-created user SAPDEMO, whose password is also SAPDEMO. Furthermore, it references the database instance JP1, which is used as the default in the J2EE Preview. You can use the script as an example in case you have to define new tables for your own applications.

Creating Your Own Tables in the MaxDB Database

Creating Tables Using SQLCLI Script

If the set of predefined tables is not sufficient for your needs, you can simply add more tables by copying the script from the last section into a file and then modifying it. You then run the script with the command line tool SQLCLI. For example, the following dialog would run the script *demotables.sql* for user SUPERDBA (again, MYMASTER should be the master password supplied during installation):

```
N:\>sqlcli -d JP1 -u superdba,mymaster

Welcome to the MaxDB interactive terminal.

sqlcli=>                \i               C:\myscripts\demotables.sql
sqlcli=> \q
```

Of course, SQLCLI offers much more functionality for entering and executing SQL statements, executing database procedures, and querying information about the database instance. For a complete list of commands, consult the “Tools” section in the online documentation for the MaxDB at http://dev.mysql.com/doc/maxdb/en/default.htm.

On the subject of command line tools: Since we are not providing any administration tools with the installation of the MaxDB on the J2EE Preview, we would like to point out that the command line tool DBMCLI can be used for executing virtually all Maxdb administrative tasks. For example, the following line can be used to find out whether the MaxDB is running and online:

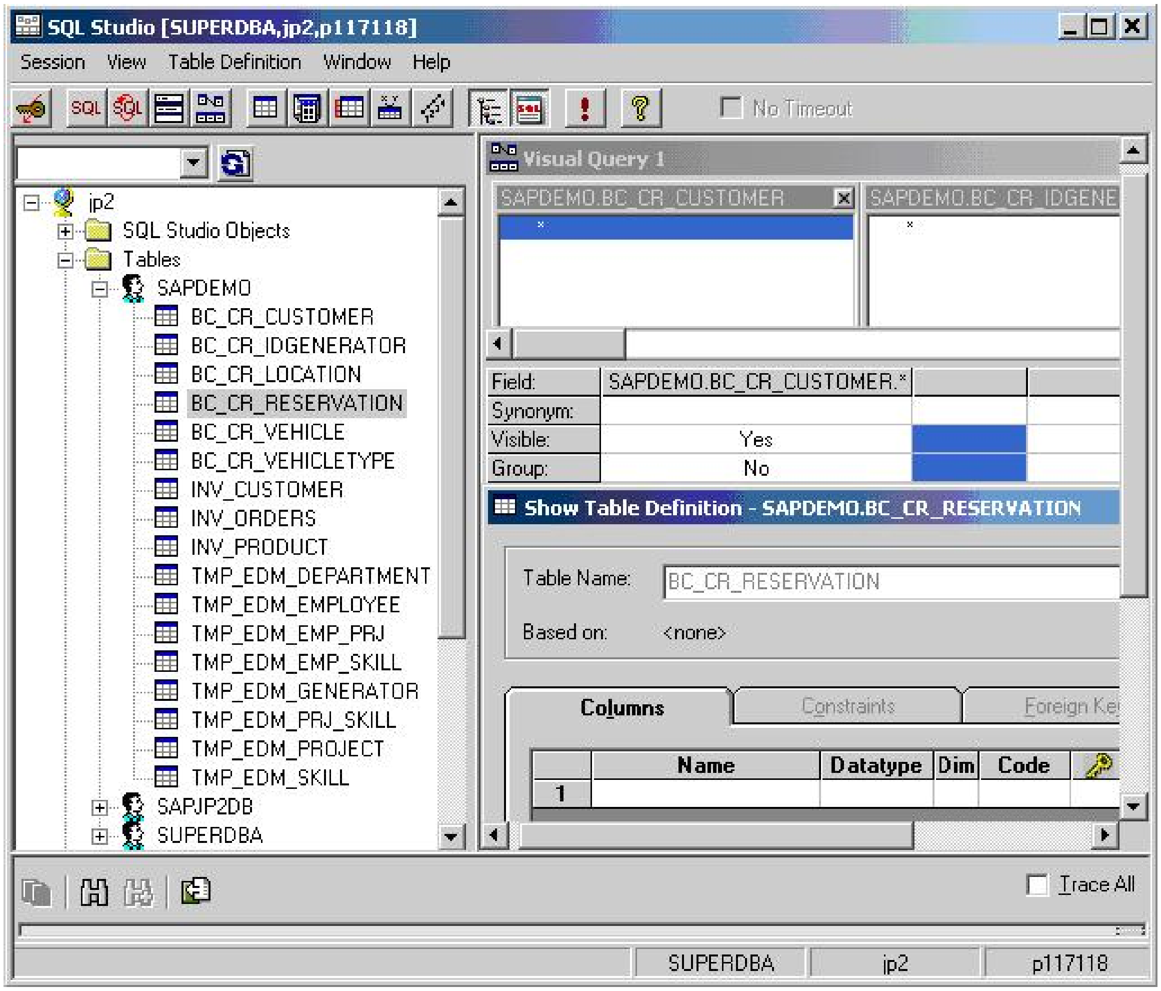
```
N:\>dbmcli -d JP1 -u control,mymaster db_state
OK
State
ONLINE
```

Again, the “Tools” section in the MaxDB online documentation has a full list of commands.

Creating Tables Using SQL Studio

The second and much more comfortable way of managing the content of the MaxDB is offered by the SQL Studio. The SQL Studio can be downloaded from:
http://dev.mysql.com/downloads/maxdb/7.6.00.html#SQL_Studio

With the SQL Studio you can not only create tables, but easily access all the application data as well as the database catalog of a MaxDB database instance. You can create, execute, and manage any number of SQL statements, and you can create, display, or change database catalog objects.



Field:	SAPDEMO.BC_CR_CUSTOMER.*
Synonym:	
Visible:	Yes
Group:	No

Table Name: BC_CR_RESERVATION

Based on: <none>

	Name	Datatype	Dim	Code	
1					

When connecting to the MaxDB instance (JP1) from the SQL Studio, the same users and passwords apply that were mentioned in the last paragraph. You can either use the predefined users SUPERDBA or SAPJP1DB with the master password, or the user SAPDEMO with password SAPDEMO.

Overview of DataSources

The purpose of this guide is to outline the steps you need to perform to deploy a custom DataSource, which your applications can use to connect to the underlying database on the SAP NetWeaver J2EE Preview.

Note: The SAP NetWeaver J2EE Preview comes with a preconfigured native DataSource called SAPDEMO_DS, which you can use in your applications. However, if for any reason you need to use a different DataSource, the following contains information on how to do this.

DataSource Types

Depending on the way DataSources obtain connections from the underlying database driver, as well as the way they handle transactions, the following two types are distinguished:

- Driver-based
  This type of DataSource obtains Connection objects by calling the java.sql.Driver.connect() method implemented by the underlying JDBC driver. It returns a Connection, which can participate as a resource in local transactions. Besides using the UserTransaction object, transactions in this case can be started by invoking setAutoCommit(false) and completed by invoking commit() or rollback() methods on the Connection object.
- XADataSource-based
  This type of DataSource supports distributed transactions and is useful if multiple Connection objects are participating in the same transaction. The Connection is obtained from the vendor's

implementation of javax.sql.XADataSource. Explicit invocations of setAutoCommit(), commit(), or rollback() on the Connection object are prohibited.

DataSource Definition

A DataSource is defined using a DataSource XML descriptor that contains essential information, such as the name of the underlying JDBC driver, database URL, connection pooling properties, and so on.
The DataSource XML adheres to the Document Type Definition (DTD) contained in the data-sources.dtd file. The DTD is available under *<install_directory>\<SID>\JC<instance_number>\j2ee\cluster\server0\dtd* directory of your SAP NetWeaver Application Server.
For a detailed description of the *data-sources.dtd* elements, refer to the data-sources.dtd page on the SAP Help Portal.

Note: In the context of the SAP NetWeaver J2EE Preview edition, you should use Native SQL or Vendor SQL as the SQL engine types in your DataSource definition. Also note that Vendor SQL can be used with any database that provides a JDBC-compliant driver. Native SQL is a wrapper for the vendor-specific JDBC driver and provides important enhancements such as SQL trace and statement pooling. However, it can be used only with databases supported by SAP: Oracle 9.2 and 10.2, MSSQL 8, and MaxDB 7.x.

For more information about SQL engine types, see Java Persistence Infrastructure on the SAP Help Portal.

Deploying DataSources

There are two ways to deploy a DataSource:

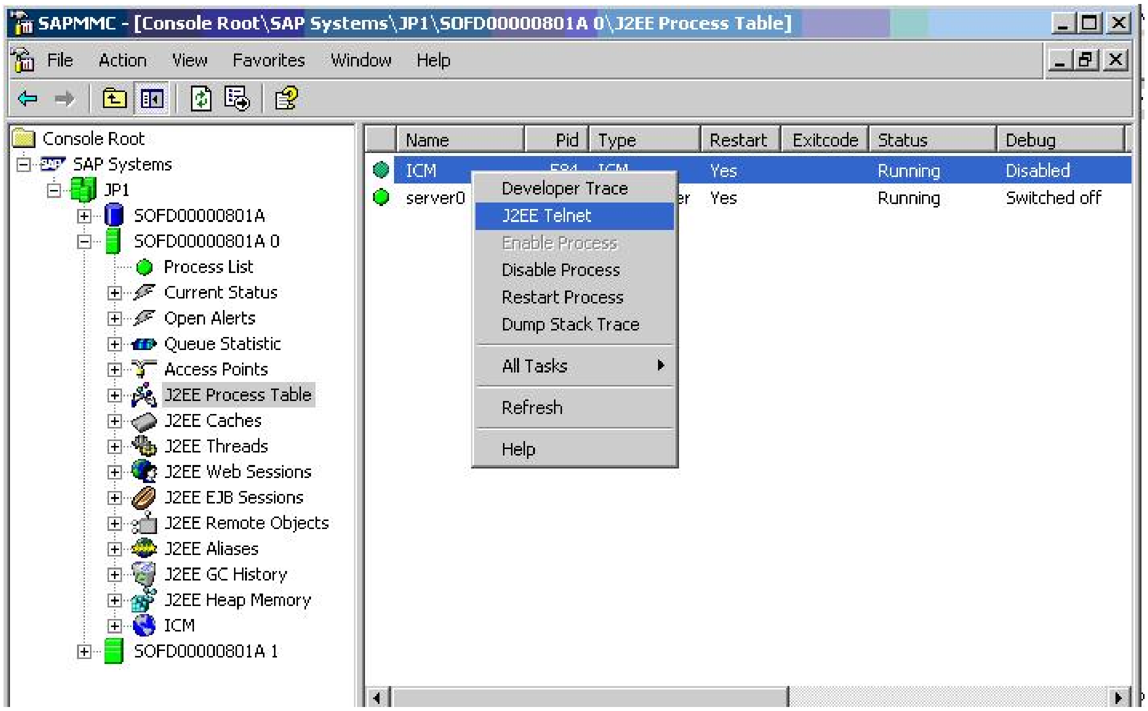
- As a standalone DataSource using the Telnet Administrator tool
- As part of a J2EE application using the SAP NetWeaver Developer Studio

Deploying a Standalone DataSource

The standalone DataSource can be used by multiple applications. Its life cycle is independent from the life cycle of the applications that are using it.

To deploy a standalone DataSource, proceed as follows:

1. Open the Telnet Administrator console:
    1. Go to the SAP Management Console and open the *J2EE Process Table* for your SAP NetWeaver Application Server.
    2. Choose *J2EE Telnet* from the context menu of the ICM process.



Name	Pid	Type	Restart	Exitcode	Status	Debug
ICM	[illegible]	ICM	Yes		Running	Disabled
server0		[illegible]	Yes		Running	Switched off

2. Login to the server by providing a valid Administrator user and password in the *User name* and *Password* fields in the Telnet Administrator console.
3. Enter *add dbpool* in the command line to add the DBPOOL shell command group to the environment.
4. Enter *make_data_source <xml_descriptor>* in the command line to deploy the DataSource defined by the <xml_descriptor> parameter.
   The <xml_descriptor> parameter specifies the path to your data-sources.xml on the file system.

Deploying a DataSource within a J2EE Application

If you deploy a DataSource as part of an application, the DataSource's life cycle will depend on the application's life cycle - that is, the DataSource will be available only if the application that deployed it is started. It can also be used by other applications by declaring resource references to it in their deployment descriptors.

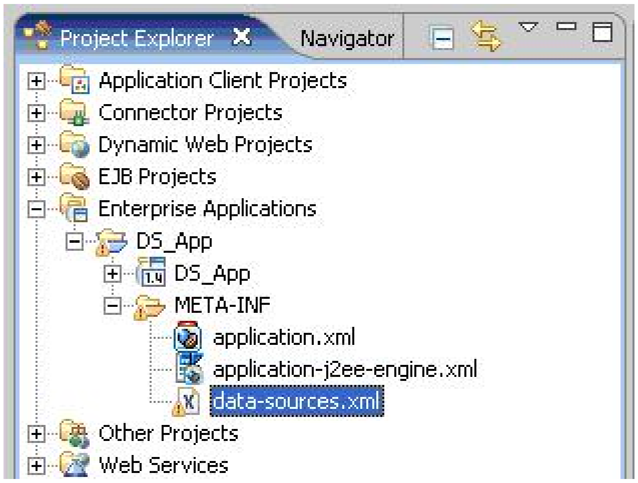
To deploy a DataSource with an application, you need to add a *data-sources.xml* to the *META-INF* folder of the application's EAR file. The following procedure describes how to do this using the SAP NetWeaver Developer Studio.

Case: J2EE 1.4 Application

If you are developing a J2EE 1.4 application using the WTP tools in the J2EE Perspective of the SAP NetWeaver Developer Studio, proceed as follows:

1. Select the application you want to deploy the DataSource with from the Enterprise Applications list and choose *Properties* from the context menu.
2. In the *Properties* screen that appears, choose *Project Facets* from the tree on the left.
3. Choose *Add/Remove Project Facets* from the *Project Facets* screen area on the right.
4. Select the *SAP Data Source* facet from the *Add/Remove Project Facets* screen and choose *Finish*.
5. Choose *OK*.

   This should add a data-sources.xml file to the META-INF directory of your application:



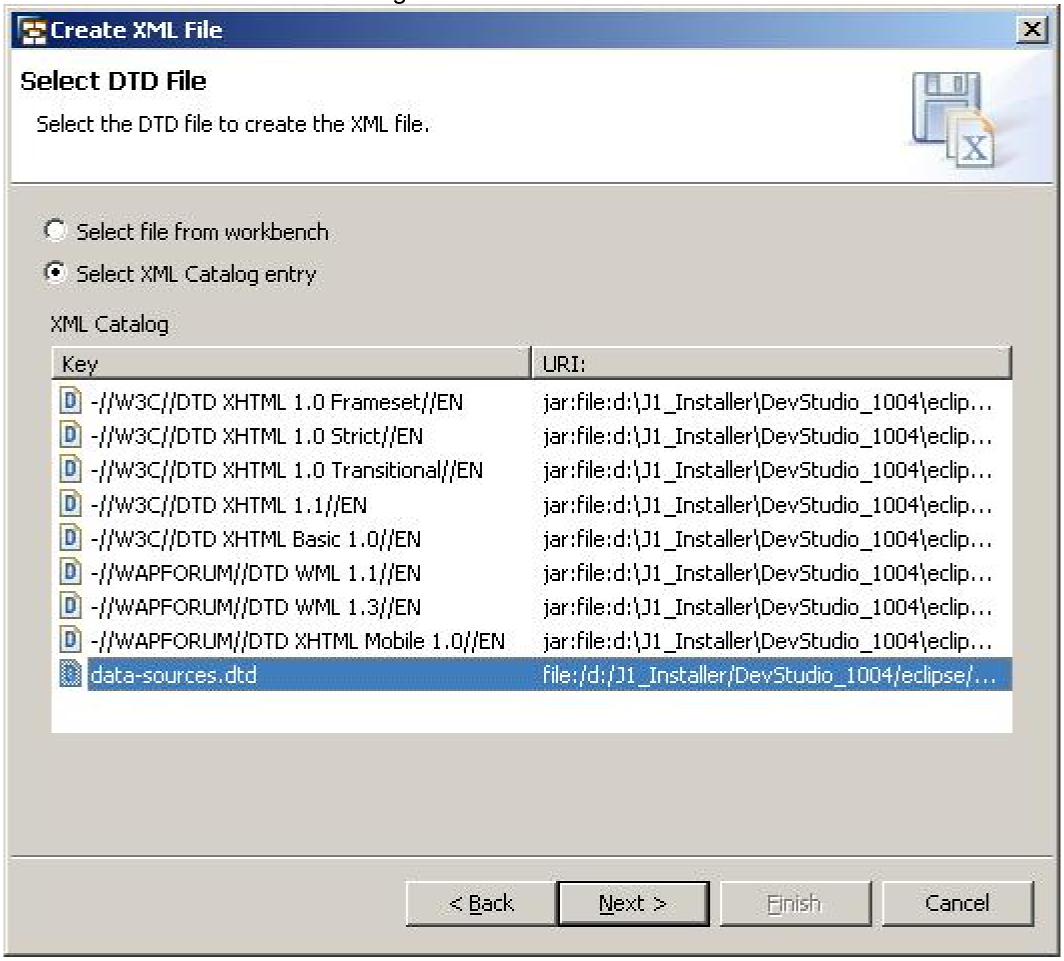
6. Double-click on *data-sources.xml* to open it in an XML editor on the right.
7. Define the DataSource following the description of the *data-sources.dtd* mentioned above.
   After you have packaged and deployed the application, your DataSource you defined will be available on the server whenever the application is started.

Case: Java EE 5 Application

If you are developing a Java EE 5 application using the SAP NetWeaver Java EE Preview Project in the Java Perspective of the SAP NetWeaver Developer Studio, proceed as follows:

1. Select the *META-INF* directory of your application and choose *New -> Other* from the context menu.
2. In the *New* screen that appears, choose *XML -> XML* and then choose *Next*.
3. In the *Create XML* file screen, select *Create XML file from a DTD file* and choose *Next*.
4. Enter *data-sources.xml* in the *File name* field of the *XML File Name* screen and choose *Next*.

5. In the *Select DTD File* screen, choose the *Select XML Catalog entry* option and choose *data-sources.dtd* from the *XML Catalog* list.



Select DTD File

Select the DTD file to create the XML file.

○ Select file from workbench

● Select XML Catalog entry

XML Catalog

Key	URI:
-//W3C//DTD XHTML 1.0 Frameset//EN	jar:file:d:\J1_Installer\DevStudio_1004\eclip...
-//W3C//DTD XHTML 1.0 Strict//EN	jar:file:d:\J1_Installer\DevStudio_1004\eclip...
-//W3C//DTD XHTML 1.0 Transitional//EN	jar:file:d:\J1_Installer\DevStudio_1004\eclip...
-//W3C//DTD XHTML 1.1//EN	jar:file:d:\J1_Installer\DevStudio_1004\eclip...
-//W3C//DTD XHTML Basic 1.0//EN	jar:file:d:\J1_Installer\DevStudio_1004\eclip...
-//WAPFORUM//DTD WML 1.1//EN	jar:file:d:\J1_Installer\DevStudio_1004\eclip...
-//WAPFORUM//DTD WML 1.3//EN	jar:file:d:\J1_Installer\DevStudio_1004\eclip...
-//WAPFORUM//DTD XHTML Mobile 1.0//EN	jar:file:d:\J1_Installer\DevStudio_1004\eclip...
data-sources.dtd	file:/d:/J1_Installer/DevStudio_1004/eclipse/...

6. Choose *Next* and then *Finish* to create the data-sources.xml file.

Deploying JDBC Drivers

If you want to use a DataSource that connects your applications to a database other than the MaxDB database provided with the SAP NetWeaver J2EE Preview edition, you need to deploy the relevant JDBC driver for that database. You can do this using the Telnet Administrator console:

1. Open the Telnet Administrator and login to the server. For more information, see Deploying a Standalone DataSource.
2. Enter *add dbpool* in the command line to add the DBPOOL shell command group to the environment.
3. Enter *deploy_jdbc_driver <driver-name> <driver_archive>* in the command line, where:
    1. <*driver-name*> is the name, under which this driver is registered in the system. You will use this name as a value of the <driver-name> tag of the data-sources.xml file when you define your custom DataSource.
    2. <*driver-archive*> is the directory path to the archive file on the file system that contains your JDBC driver. Note that this command can have multiple <*archive-name*> arguments separated with semicolons (;). This is necessary when deploying drivers that consist of several JAR files.

Appendix

demotables.sql

```
// this line assumes that MYMASTER is the master password that you
// provided during installation

\connect -d JP1 -u SUPERDBA,MYMASTER

//

DROP USER SAPDEMO

//

CREATE USER SAPDEMO PASSWORD SAPDEMO DBA NOT EXCLUSIVE

//

\disconnect

//

\connect -u SAPDEMO,SAPDEMO


//------------------------------------------------------------------

// Create tables for VirtualShopLite application

//------------------------------------------------------------------


CREATE TABLE INV_CUSTOMER
(
        ID      Integer    NOT NULL    DEFAULT           0,
        NAME    Long UNICODE,
        PRIMARY KEY (ID)
)

//

CREATE TABLE INV_ORDERS
(
        ID               Integer    NOT NULL    DEFAULT           0,
        CUSTOMER_ID      Integer    NOT NULL    DEFAULT           0,
        PRODUCT_ID       Integer    NOT NULL    DEFAULT           0,
        ORDER_TIMESTAMP  Timestamp    NOT NULL,
        STATUS           Integer    NOT NULL    DEFAULT           0,
        QUANTITY         Float (38)    NOT NULL    DEFAULT
0.0000000000000000000000000000000000000E+00,
        PRIMARY KEY (ID)
)

//


CREATE TABLE INV_PRODUCT
(
        ID      Integer    NOT NULL    DEFAULT           0,
        NAME    Varchar (100) UNICODE    NOT NULL    DEFAULT ' ',
        PRICE   Float (38)    NOT NULL    DEFAULT
0.0000000000000000000000000000000000000E+00,
        PRIMARY KEY (ID)
)


//------------------------------------------------------------------

// Create tables for EDM Demo Application

//------------------------------------------------------------------


CREATE TABLE TMP_EDM_DEPARTMENT
(
 DEPARTMENT_ID     Varchar (20) UNICODE    NOT NULL,
 MANAGER_ID        Integer,
 NAME              Varchar (30) UNICODE,
 VERSION           Integer    NOT NULL,
 PRIMARY KEY (DEPARTMENT_ID)
)

//

CREATE TABLE TMP_EDM_EMPLOYEE
(
 EMPLOYEE_ID       Integer    NOT NULL,
 DEPARTMENT_ID     Varchar (20) UNICODE,
 SALUTATION        Varchar (30) UNICODE,
 FIRST_NAME        Varchar (30) UNICODE,
 LAST_NAME         Varchar (30) UNICODE,
 EMAIL             Varchar (50) UNICODE,
 SALARY            Fixed (10,2),
 CURRENCY          Varchar (6) UNICODE,
 PICTURE           Long BYTE,
 VERSION           Integer    NOT NULL,
 PRIMARY KEY (EMPLOYEE_ID)
)

//

CREATE TABLE TMP_EDM_EMP_PRJ
(
 EMPLOYEE_ID   Integer    NOT NULL,
 PROJECT_ID    Integer    NOT NULL,
 PRIMARY KEY (EMPLOYEE_ID, PROJECT_ID)
)

//

CREATE TABLE TMP_EDM_EMP_SKILL
(
 EMPLOYEE_ID  Integer    NOT NULL,
 SKILL_ID     Integer    NOT NULL,
 PRIMARY KEY (EMPLOYEE_ID, SKILL_ID)
)
```

```
//

CREATE TABLE TMP_EDM_GENERATOR
(
 BEAN_NAME   Varchar (256) UNICODE    NOT NULL    DEFAULT '0',
 MAX_ID      Integer    NOT NULL    DEFAULT           0,
 PRIMARY KEY (BEAN_NAME)
)

//

CREATE TABLE TMP_EDM_PRJ_SKILL
(
 PROJECT_ID  Integer    NOT NULL,
 SKILL_ID    Integer    NOT NULL,
 PRIMARY KEY (PROJECT_ID, SKILL_ID)
)

//

CREATE TABLE TMP_EDM_PROJECT
(
 PROJECT_ID          Integer    NOT NULL,
 LEAD_ID             Integer,
 TITLE               Varchar (30) UNICODE,
 DESCRIPTION         Varchar (1000) UNICODE,
 START_DATE          Date,
 END_DATE            Date,
 VERSION             Integer    NOT NULL,
 PRIMARY KEY (PROJECT_ID)
)

//

CREATE TABLE TMP_EDM_SKILL
(
 SKILL_ID          Integer    NOT NULL,
 LANGUAGE_ID       Varchar (32) UNICODE    NOT NULL,
 NAME              Varchar (30) UNICODE,
 VERSION           Integer    NOT NULL,
 PRIMARY KEY (SKILL_ID, LANGUAGE_ID)
)

//------------------------------------------------------------------

// Create tables for Car Rental Application

//------------------------------------------------------------------


CREATE TABLE BC_CR_CUSTOMER
(
        CUSTOMERID      Varchar (10)  UNICODE    NOT NULL    DEFAULT '0',
        FIRSTNAME       Varchar (35)  UNICODE    NOT NULL    DEFAULT '''',
        LASTNAME        Varchar (35)  UNICODE    NOT NULL    DEFAULT '''',
        USERNAME        Varchar (50)  UNICODE    NOT NULL    DEFAULT 'user',
        PASSWORD        Varchar (10)  UNICODE    NOT NULL    DEFAULT '''',
        FORM            Varchar (10)  UNICODE,
        COUNTRY         Varchar (3)   UNICODE    NOT NULL    DEFAULT '''',
        REGION          Varchar (50)  UNICODE,
        CITY            Varchar (50)  UNICODE    NOT NULL    DEFAULT '''',
        POSTALCODE      Varchar (10)  UNICODE    NOT NULL    DEFAULT '0',
        ADDRESS         Varchar (100) UNICODE    NOT NULL    DEFAULT '''',
        DRIVERLICENSE   Varchar (15)  UNICODE    NOT NULL    DEFAULT '0',
        PASSPORTNR      Varchar (15)  UNICODE    NOT NULL    DEFAULT '0',
        PHONE           Varchar (20)  UNICODE,
        MOBILE          Varchar (20)  UNICODE,
        EMAIL           Varchar (75)  UNICODE,
        CREDITCARDID    Varchar (16)  UNICODE,
        CREDITCARDTYPE  Varchar (20)  UNICODE,
        COMPANY         Varchar (50)  UNICODE,
        USERTYPE        Varchar (20)  UNICODE    NOT NULL    DEFAULT 'regular
customer',
        CUMULATEDVALUE  Fixed (16,2)    NOT NULL    DEFAULT               0.00,
        PRIMARY KEY (CUSTOMERID)
)

//

CREATE TABLE BC_CR_IDGENERATOR
(
        TABLENAME   Varchar (50) UNICODE    NOT NULL    DEFAULT '''',
        LASTID      Varchar (10) UNICODE    NOT NULL    DEFAULT '0',
        PRIMARY KEY (TABLENAME)
)

//

CREATE TABLE BC_CR_LOCATION
(
        LOCATIONID    Varchar (10) UNICODE    NOT NULL    DEFAULT ' ',
        COUNTRY       Varchar (3) UNICODE    NOT NULL    DEFAULT '''',
        REGION        Varchar (50) UNICODE,
        CITY          Varchar (50) UNICODE    NOT NULL    DEFAULT '''',
        POSTALCODE    Varchar (10) UNICODE    NOT NULL    DEFAULT '0',
        ADDRESS       Varchar (100) UNICODE    NOT NULL    DEFAULT '''',
        PRIMARY KEY (LOCATIONID)
)

//

CREATE TABLE BC_CR_RESERVATION
(
        ID               Varchar (10) UNICODE    NOT NULL    DEFAULT '0',
        RESERVATIONDATE  Timestamp    NOT NULL,
        DATEFROM         Timestamp,
        DATETO           Timestamp,
        STATUS           Varchar (20) UNICODE    NOT NULL    DEFAULT
'incomplete',
        EXPIREDDATE      Timestamp    NOT NULL,
        CUSTOMERID       Varchar (10) UNICODE,
        VEHICLETYPEID    Varchar (10) UNICODE,
        VEHICLEID        Varchar (20) UNICODE,
        PICKUPLOCATION   Varchar (10) UNICODE,
        DROPOFFLOCATION  Varchar (10) UNICODE,
        PRIMARY KEY (ID)
)
```

```
//

CREATE TABLE BC_CR_VEHICLE
(
        VEHICLEID     Varchar (20) UNICODE    NOT NULL    DEFAULT '0',
        MAKE          Varchar (50) UNICODE    NOT NULL    DEFAULT '''',
        MODEL         Varchar (20) UNICODE    NOT NULL    DEFAULT ' ',
        FUEL          Varchar (20) UNICODE    NOT NULL    DEFAULT 'patrol',
        CONSUMPTION   Fixed (2,1)    NOT NULL    DEFAULT   0.0,
        SEATS         Smallint    NOT NULL    DEFAULT      0,
        TANKVOLUME    Smallint,
        MILEAGE       Integer    NOT NULL    DEFAULT           0,
        DESCRIPTION   Varchar (500) UNICODE,
        PICTURE       Varchar (50) UNICODE,
        AVAILABILITY  Varchar (20) UNICODE    NOT NULL    DEFAULT 'available',
        STATUS        Varchar (20) UNICODE    NOT NULL    DEFAULT '0',
        LOCATIONID    Varchar (10) UNICODE,
        TYPEID        Varchar (10) UNICODE,
        PRIMARY KEY (VEHICLEID)
)

//

CREATE TABLE BC_CR_VEHICLETYPE
(
        VEHICLETYPEID  Varchar (10) UNICODE    NOT NULL    DEFAULT '0',
        NAME           Varchar (50) UNICODE    NOT NULL    DEFAULT '''',
        DESCRIPTION    Varchar (500) UNICODE,
        PRICE          Fixed (16,2)    NOT NULL    DEFAULT               0.00,
        CURRENCY       Varchar (5) UNICODE    NOT NULL    DEFAULT '''',
        COUNTRY        Varchar (3) UNICODE    NOT NULL    DEFAULT '''',
        PICTURE        Varchar (50) UNICODE    NOT NULL    DEFAULT '''',
        PRIMARY KEY (VEHICLETYPEID)
)


\disconnect
```

Driver-based DataSource XML descriptor

```
<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE data-sources SYSTEM 'data-sources.dtd'>

<data-sources>

   <data-source>

       <data-source-name>SAPDB</data-source-name>

       <driver-name>SYSTEM_DRIVER</driver-name>

       <init-connections>1</init-connections>

       <max-connections>10</max-connections>

       <max-time-to-wait-connection>60</max-time-to-wait-connection>
       <expiration-control>
             <connection-lifetime>600</connection-lifetime>
             <run-cleanup-thread>60</run-cleanup-thread>
       </expiration-control>
       <sql-engine>native_sql</sql-engine>
       <jdbc-1.x>
             <driver-class-name>com.sap.dbtech.jdbc.DriverSapDB</driver-class-name>
             <url>jdbc:sapdb://localhost/n05</url>
             <user-name>sapn05db</user-name>
             <password>abc123</password>
       </jdbc-1.x>
   </data-source>
</data-sources>
```

Copyright

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

These materials are provided “as is” without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.

SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.

Any software coding and/or code lines/strings (“Code”) included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.