

IBM SAP Technical Brief

**Connect Data Sources to SAP HANA on IBM Power Systems via
SAP HANA Smart Data Integration**

**IBM SAP International Competence Center
Walldorf, Germany**

**Version: 1.1
April 2018**

Preface

Edition Notice

The latest published version of this paper is available at:

<http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102749>

Document Version	Date	Changes
1.0	10.04.2018	First edition of this document
1.1	19.04.2018	Fixed some hyperlinks

Authors

- Walter Orb, IBM SAP International Competence Center
- Dr. Edmund Häfele, IBM Technical Sales for SAP

Feedback

We're interested in any feedback you have. Please send your comments to isicc@de.ibm.com.

Disclaimer

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Any performance data contained herein was determined in a controlled environment.

Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems.

Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products.

All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs

ANY INFORMATION HEREIN IS PROVIDED “AS IS” WITHOUT WARRANTY OR INDEMNIFICATION OF ANY KIND BY IBM AND DO NOT ANY EXPRESS OR IMPLIED, WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS OR USAGE FOR PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.

1	BACKGROUND AND SCENARIOS	5
1.1	SCOPE AND MOTIVATION	5
1.2	SAP HANA ENTERPRISE INFORMATION MANAGEMENT	5
1.3	SAP HANA SMART DATA INTEGRATION USE-CASES	6
1.4	POSITIONING AND OVERLAP WITH OTHER TOOLS	6
1.4.1	<i>Secondary database connection in SAP NetWeaver AS ABAP</i>	6
1.4.2	<i>SAP HANA Smart Data Access</i>	7
1.4.3	<i>SAP HANA Smart Data Integration Architecture</i>	8
1.4.4	<i>Data Federation Comparison</i>	9
1.5	DOCUMENTATION	9
2	CONFIGURE SMART DATA INTEGRATION	11
2.1	SETUP OF THE TEST SCENARIO	11
2.2	SDI BASE SETUP	13
2.2.1	<i>Create an DP Agent Admin User in SAP HANA</i>	13
2.2.2	<i>Create an DP Agent Monitor User in SAP HANA</i>	14
2.2.3	<i>Enable the Data Provisioning Server</i>	15
2.2.4	<i>Download SDI components</i>	16
2.2.5	<i>Download Oracle JDBC Driver</i>	17
2.2.6	<i>Deploy Data Provisioning Delivery Unit</i>	18
2.2.7	<i>Install the Data Provisioning Agent</i>	19
2.2.8	<i>Configure the Data Provisioning Agent</i>	21
2.3	SDI APPLICATION SETUP	25
2.3.1	<i>Create a Remote Source in SAP HANA</i>	25
2.3.2	<i>Create Remote Source with a SQL statement</i>	27
2.4	SAP HANA SDI MONITORING	28
2.4.1	<i>Data Provisioning Adapter Monitoring</i>	28
2.4.2	<i>Monitoring using the HANA_IM_DP Deployment Unit</i>	28
3	ACCESSING REMOTE DATA SOURCE SCENARIOS	30
3.1	SAP SDI VIRTUAL TABLE SCENARIO	30
3.1.1	<i>Create User and Schema</i>	30
3.1.2	<i>Grant Schema Access</i>	31
3.1.3	<i>Create Virtual Table</i>	31
3.1.4	<i>Check Virtual Table Definition and Content</i>	33
3.1.5	<i>Create a Virtual Table using SQL Statements</i>	38
3.2	SECONDARY DB CONNECTION – DIRECT SCENARIO	39
3.2.1	<i>Secondary Database Connection – ABAP Coding Options</i>	39
3.2.2	<i>Prepare Secondary DB Connection on the SAP Application Server</i>	40
3.2.3	<i>Define Secondary DB Connection in the Database Administrator Cockpit</i>	40
3.2.4	<i>Access Data in ABAP via Secondary Database Connection</i>	43
3.3	SECONDARY DB CONNECTION – SAP HANA SDI SCENARIO	45
3.4	SECONDARY DB CONNECTION – SAP HANA SDI NATIVE SQL SCENARIO	49
3.5	REMOTE DATABASE ACCESS – SAP HANA SDI NATIVE SQL SCENARIO	51
4	SUMMARY	54

1 Background and Scenarios

1.1 Scope and Motivation

Initial scope of the document is to

- Provide a short “How-to” and “Step-by-step” approach for setting up SAP HANA Smart Data Integration in a SAP HANA on IBM Power Systems environment
- Provide guidelines how to transfer current data provisioning scenarios (for example using SAP HANA Smart Data Access on a x86 infrastructure) to a SAP HANA on IBM Power Systems topology with SAP HANA Smart Data integration.

Readers of the document are expected to have a working knowledge of SAP technology components (SAP HANA, SAP NetWeaver technology, SAP ABAP programming language, including the required tooling e.g. SAP HANA Web IDE, HANA studio), IBM Power Systems, and Linux operating system.

1.2 SAP HANA Enterprise Information Management

SAP introduced several tools to access data stored in foreign locations and make this available within the SAP HANA database during ongoing evolution of SAP HANA.

Starting with SAP HANA 1.0 SP09 two new features, SAP HANA Smart Data Integration (SDI) and SAP HANA Smart Data Quality (SDQ), have been introduced. These features form the SAP HANA Enterprise Information Management (EIM) and provide an integrated Extraction, Transformation, and Loading (ETL) mechanism directly into the SAP HANA database.

SDI provides data replication and transformation services to handle all styles of data integration. It provides tools to access remote source data (data federation) and to provision, replicate, or transform this data into SAP HANA on-premise or cloud database systems.

SDI capabilities include:

- Replication and ETL-type data transformation services, integrated natively in SAP HANA
- Open and extensible: works on SAP and non-SAP data of any source, style, shape and size
- Support for all kinds of data delivery: real-time, batch, federation, and streaming
- Load data into SAP HANA on-premise or in the cloud
- Modeling environment is part of HANA Studio and HANA Web-based Development Workbench
- Extends HANA’s Smart Data Access (SDA) connectivity by leveraging the new Data Provisioning Framework (DP Server and DP Agent) and an Adapter SDK
- Provides transactional integrity for real time push (through DBMS transaction log listeners)

In addition, SDQ provides advanced transformation mechanisms to support data quality functionality. The SDQ option allows to enhance, cleanse, and transform data to make it more accurate and useful.

SDQ capabilities include:

- Real-time, high-speed data cleansing, address cleansing, and geospatial data enrichment

- Intuitive interface to define data transformation flowgraphs in SAP HANA Studio
- Can be directly embedded in data flows / ETL processes

These options enable connecting efficiently to any source to provision and cleansing data for loading into SAP HANA on-premise or in the cloud. For supported systems, data can be written back to the original source too.

1.3 SAP HANA Smart Data Integration Use-Cases

- **Data Federation**
Expose remote data sources like databases or other sources as for example Hadoop to the SAP HANA database. The data is not physically moved, but remains in its original source environment. Via virtual tables, the data becomes available in HANA queries. There can be latency issues and such a federation scenario is only useful for infrequent queries and low amounts of data. However, it can be a first step before moving to replication.
- **Data Replication**
Move data from the original source into the HANA database in real-time or using scheduled processes. Real-time replication is possible for selected sources like databases (Oracle, IBM Db2, Microsoft SQL Server, ...), and Twitter. A change in these sources is replicated in (near) real-time to the HANA database.
- **Data Transformation**
Apply complex transformations on the data before storing it in SAP HANA, or for data that is already available in HANA and needs further transformation. These transformations include SQL-like operations like join, filter, and aggregate, as well as more complex operations like pivot your data (transform rows to columns and vice versa) or build in conditional logic (CASE transform). Another common data transformation scenario is history preserving to build up a history of changes (e.g. a delete in the source would be transformed to an update to set a flag to inactive, but keeping the record for history).

This document focuses on Data Federation – access of data in SAP HANA from a remote source without moving the data into the SAP HANA itself. Data Federation is the base use case, and once implemented can be easily extended with Data Replication / Data Transformation scenarios.

1.4 Positioning and overlap with other tools

1.4.1 Secondary database connection in SAP NetWeaver AS ABAP

SAP NetWeaver Application Server ABAP has the capability to access other databases in addition to its primary database. This is known as so-called secondary databases or secondary database connections. In case the secondary database is a different database type than the primary database of the SAP NetWeaver AS ABAP, the SAP provided Database Shared Library (DBSL) and the database vendor provided client must be installed. This implies that the secondary database must be a database system supported by SAP for the respective SAP kernel and operating system release combination – only then the shared library is available. For the secondary database, a new connection is created via the Database Administrator Cockpit.

The table below illustrates the availability of DBSL shared libraries and database clients for the different SAP NetWeaver AS ABAP operating systems (SAP NetWeaver 7.5). For details, especially for the detailed operating system releases, consider the SAP Product and Availability Matrix (SAP PAM).

SAP NetWeaver 7.5	Database Platform							
	DB2 FOR Z/OS	DB2 LUW 64-BIT	DB2/400	MAXDB 64-BIT	MS SQL SERVER X86_64	ORACLE 64-BIT	SAP ASE FOR BUSINESS SUITE	SAP HANA DATABASE
SAP ABAP Application Server OS								
AIX 64	X	X		X		X	X	X
HP-UX ON IA64		X		X		X	X	X
LINUX FOR ZSERIES	X							X
LINUX ON POWER (BIG ENDIAN)	X ²⁾	X ¹⁾		X ¹⁾				X
LINUX ON POWER (LITTLE ENDIAN)								X
LINUX ON X86_64	X	X		X	X ³⁾	X		X
OS/400			X					X
SOLARIS FOR X64		X		X		X		X
SOLARIS/SPARC 64		X		X		X		X
WINDOWS FOR X86_64	X	X	X	X	X			X

1. Supported only with SUSE SLES 11 (BE) or Red Hat (BE)
2. Supported only with Red Hat (BE)
3. SAP does not support SAP Application Servers on LINUX_X86_64 in an SAP system with a SQL Server database backend. SAP only supports secondary SQL Server connections (via DBCON) from non-SQL Server ABAP systems running on LINUX_X86_64

Microsoft supports an ODBC SQL Server Native Access driver on Linux/x86 only, which enables secondary SQL Server connections from non-SQL Server ABAP systems running on Intel/Linux servers. See also [SAP Note 1644499 - Database connectivity from Linux to SQL Server](#).

If the required DBSL shared libraries/database client is not available for the operating system of the ABAP application server, the secondary database connection can be transformed to a solution based on SAP HANA Smart Data Integration. However, this might require some coding changes in the clients' custom ABAP programs.

1.4.2 SAP HANA Smart Data Access

SAP HANA Smart Data Access (SDA) is the base method in SAP HANA for Data Federation. SDA allows access to data stored in remote data sources directly via SAP HANA. SAP HANA creates a "virtual table" mapping to tables located in remote data sources within the data provisioning framework. SAP HANA can then access the data directly by accessing the virtual table. The virtual table can be manipulated like an ordinary table - operations, as

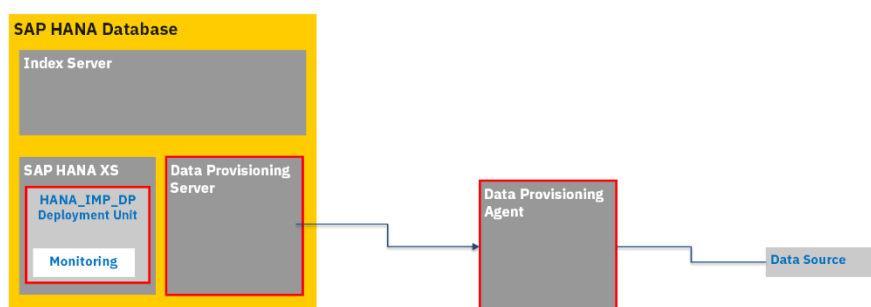
select, update, insert, delete are available for the virtual table. Join operations between local tables and virtual tables are supported too.

Prerequisite for connecting the remote data source via Smart Data Access is an ODBC driver, which needs to be installed directly on the host of the SAP HANA system. Compared with Linux/x86, only a subset of ODBC drivers is available and supported by SAP for Linux on Power. 3rd party ODBC drivers could be used as a potential alternative with SAP HANA Smart Data Access.

1.4.3 SAP HANA Smart Data Integration Architecture

SAP HANA Smart Data Integration is the strategic enterprise information management solution and provides a superset of functionalities compared to SDA. It has a huge set of adapters for remote data sources available. SDI is the preferred strategic solution for Linux on Power. Virtual tables, which were previously defined via SDA in SAP HANA, can be easily re-created using the new SDI based remote data source without any changes to the application code.

The picture below provides a high-level overview of the SAP HANA Smart Data Integration Architecture.



In the scenarios described in this document, SAP HANA Database, Data Provisioning Server and Remote Data Source are located “on-premise” and can communicate via TCP within one network.

The Data Provisioning Server is a native SAP HANA process. It provides native connectivity to some data sources as well as connectivity to the Data Provisioning Agent (in the examples in this document, the remote sources are connected via the Data Provisioning Agent). The Data Provisioning Server is installed with the SAP HANA server, but must be enabled before usage. The activation of the Data Provisioning Server is described in chapter 2.2.3.

The HANA_IM_DP delivery unit bundles monitoring and administration capabilities and the Data Provisioning Proxy for connecting to SAP HANA in the cloud. The delivery unit includes the Data Provisioning administration application, the Data Provisioning Proxy, and the Data Provisioning monitor. The import of the HANA_IM_DP delivery unit into the SAP HANA system is described in chapter 2.2.6

The Data Provisioning Agent typically runs outside the SAP HANA environment and is managed by the Data Provisioning Server. It provides connectivity for all remote data sources where the driver is not available inside the Data Provisioning Server. Installation of the Data Provisioning Agent is described in chapter 2.2.7. In our scenario, the Data Provisioning Server and the Data Provisioning Agent are deployed within one common network segment, no proxy is required in between.

The Data Provisioning Agent needs to be connected to the SAP HANA database, registered with the Data Provisioning Server, and the respective adapters need to be deployed. This is described in chapter 2.2.8.

1.4.4 Data Federation Comparison

The table below summarizes the data federation situation for Linux/x86 versus Linux on Power for the example of an Oracle database:

Connection Layer to Remote Database (Oracle)		Linux/x86	Linux on Power
Secondary DB Connection in AS ABAP	DBSL / DB Client required for OS of SAP Kernel	DBSL / DB Client available	DBSL / DB Client not available
Smart Data Access in SAP HANA	ODBC Driver required for OS of SAP HANA DB	ODBC Driver available	ODBC Driver not available from Oracle. (a 3rd party driver might be available)
Smart Data Integration in SAP HANA	OracleLogReaderAdapter on Data Provisioning Agent, and JDBC Driver. Data Provisioning Agent installed on remote database or on its own system.	Data Provisioning Agent, OracleLogReaderAdapter and JDBC Driver available	Data Provisioning Agent, OracleLogReaderAdapter and JDBC Driver available on SLES12 (LE)

1.5 Documentation

We will provide references to existing documentation throughout the document. Unfortunately, URL links might change over time, so this chapter explains how to find the most important documentation.

The starting point is the *SAP Help Portal* at <https://help.sap.com/viewer/index>. Search for *Smart Data Integration* and select the *Suggested Products* link *SAP HANA Smart Data Integration and SAP HANA Smart Data Quality*.

At the time of writing this document, this link refers to https://help.sap.com/viewer/p/HANA_SMART_DATA_INTEGRATION. Select the *Version* matching your HANA release.

The resulting web site provides references to important documents like the *Installation and Configuration Guide*, the *Administration Guide*, and to several developer and modelling guides. There is also a link to the *SAP HANA Academy*, which refers to the *YouTube* channel *SAP HANA Smart Data Integration and Smart Data Quality*. This channel has currently more than a hundred videos covering different areas of this topic.

Another recommended starting point is the *SAP HANA Smart Data Integration* product page in the *SAP ONE Support Launchpad*. Login to the Launchpad at <https://launchpad.support.sap.com>, select *Products* in the search pull down menu and search for *Smart Data Integration*. Select your release and you'll end up on the product homepage.



This web site provides references to the standard documentation, recommended SAP notes, as well as links to blogs and questions on the *SAP Community* site.

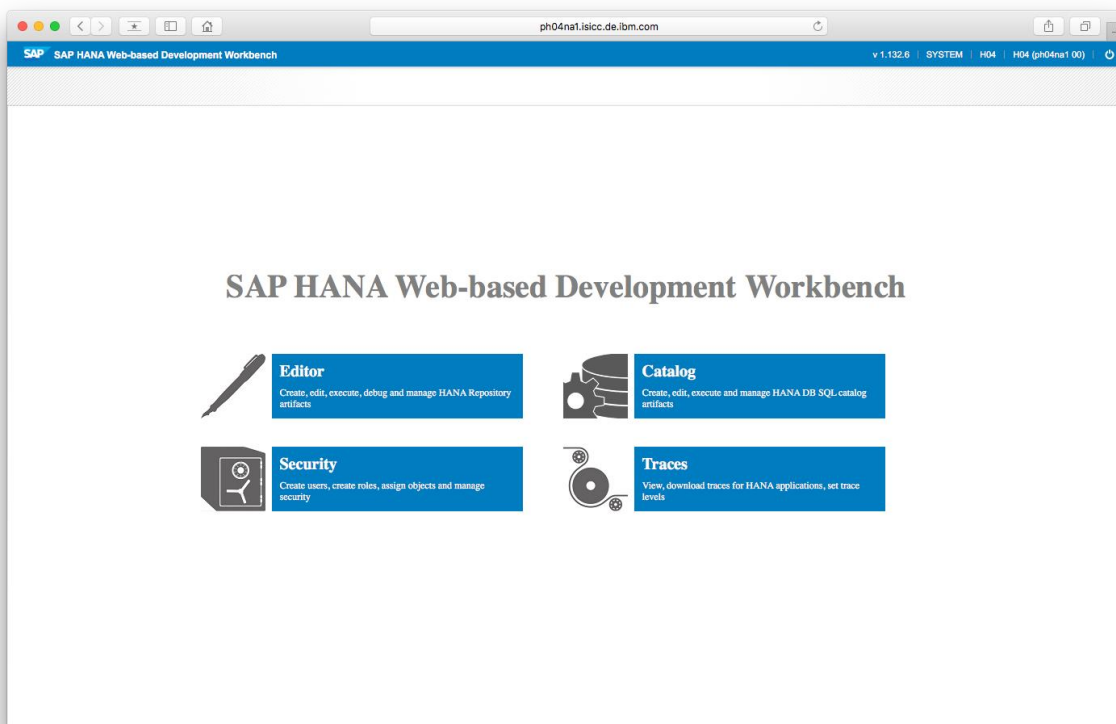
2 Configure Smart Data Integration

From this point on, we will demonstrate most of the tasks that need to be performed in the HANA system using the SAP HANA Web-based Development Workbench. We will also use the term SAP HANA Web IDE as a synonym for the web-based development workbench.

The workbench is available at <http://<host>:80<instance number>/sap/hana/ide>.

For example, the URL of our demo environment is <http://ph04na1.isicc.de.ibm.com:8000/sap/hana/ide/>.

The following screenshot shows the welcome window after initial login, from there you can navigate to the different modules. In this whitepaper, we will mostly use the *Security* and the *Catalog* modules.

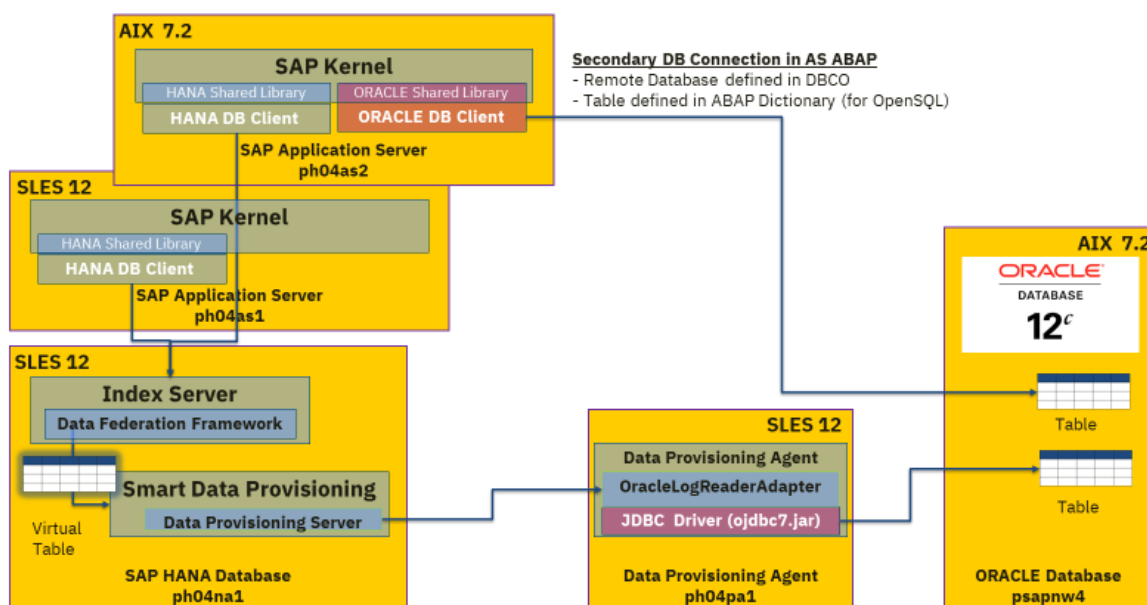


A user must be granted the required roles or privileges in advance. The system role *sap.hana.xs.ide.roles::Developer* allows access all the modules of SAP HANA Web-based Development Workbench. Please check the [SAP HANA Web-Based Development Workbench](#) help page for more details about available roles.

Some SQL statements (as for example *ALTER DATABASE* are not supported in the workbench. In these cases, you should use SAP HANA Studio or the SAP HANA database interactive terminal (hdbsql) to execute these statements.

2.1 Setup of the Test Scenario

In the initial setup, we consider the access to a remote Oracle database using SAP HANA Smart Data Integration.



A SAP NetWeaver system is installed with two application servers (one on AIX 7.2, a second one on SLES12) connected to an SAP HANA 2.0 database. Another SAP NetWeaver 7.5 system using an Oracle database serves as a remote data source system. A Data Provisioning Agent is installed on a separate partition.

This tables shows the list of hostnames and their respective roles:

Hostname	Landscape (SID)	Role	Operating System
ph04na1	H04	HANA database	SLES 12
ph04as1	H04	NW AS ABAP	SLES 12
ph04as2	H04	NW AS ABAP	AIX 7.2
ph04pa1	H04	Data Provisioning Agent	SLES12
psapnw4	NW4	Oracle database NW AS ABAP	AIX 7.2

Two kinds of adapters are available with SAP HANA Smart Data Integration for Oracle databases:

- OracleLogReaderAdapter**
 The OracleLogReaderAdapter retrieves data from an Oracle database. It can receive changes to tables in real time, and it is possible to write back data changed to the virtual table in SAP HANA to the Oracle database. The OracleLogReaderAdapter uses the Oracle data dictionary when browsing metadata.
- OracleECCAdapter**
 Like the OracleLogReaderAdapter, the OracleECCAdapter retrieves data from SAP ERP system running on an Oracle database. It can receive changes in real time, and it is possible to write back data changed to the virtual table in SAP HANA to the Oracle

database. The OracleECCAdapter uses the data dictionary in the SAP ERP system when browsing metadata.

In our example, the Data Provisioning Agent has the OracleLogReaderAdapter configured and the Oracle JDBC driver is installed.

2.2 SDI Base Setup

For the SDI base setup, a series of tasks need to be executed:

In the SAP HANA system,

- Two users need to be created to connect the Data Provisioning Agent to the SAP HANA engine
 - The DP Agent Admin user allows to manage the Data Provisioning Agent and has all the necessary admin privileges to configure the agent and manage the adapters.
 - The second user is a technical user used for monitoring, and is used by the agent for the continuous connection to SAP HANA in a proxy scenario.
- The Data Provisioning Server needs to be enabled.
- The Data Provisioning delivery unit needs to be deployed.

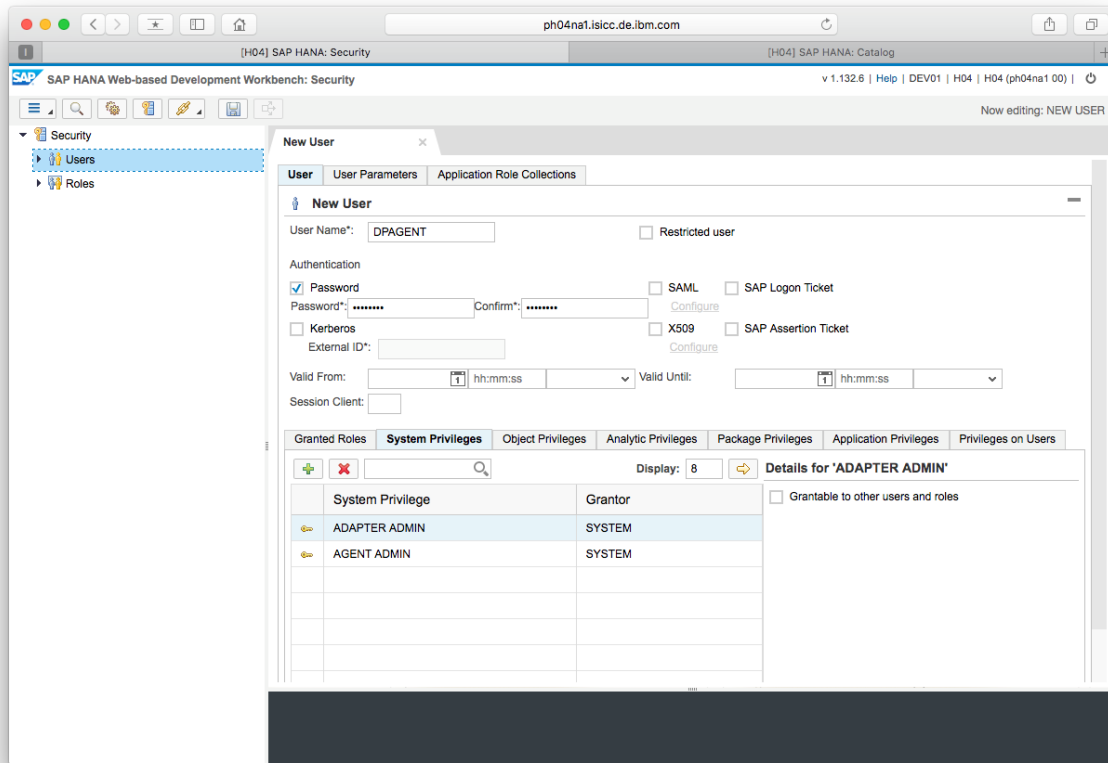
On the LPAR dedicated for the Data Provisioning Agent

1. JDBC drivers required for the adapters need to be installed.
2. The Data Provisioning Agent needs to be installed.
3. The agent and the adapters need to be configured.

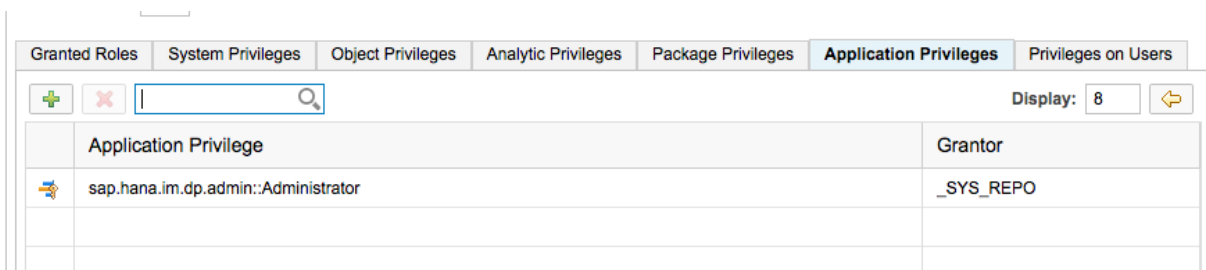
2.2.1 Create an DP Agent Admin User in SAP HANA

Use the SAP HANA Web IDE to create the DP Agent Admin user for SAP HANA. In our environment, we used the following steps.

1. Login with the *SYSTEM* user, switch to the *Security* module and select *Menu > New > User*.
2. Enter the user name *DPAGENT* and a password.
3. Switch to the *System Privileges* tab to add the required privileges to register provisioning agent(s) and adapter(s) to a DP Agent Admin user:
 - a. System privilege: *AGENT ADMIN*
 - b. System privilege: *ADAPTER ADMIN*
4. Click the *Save* icon to create the new DPAGENT userid.

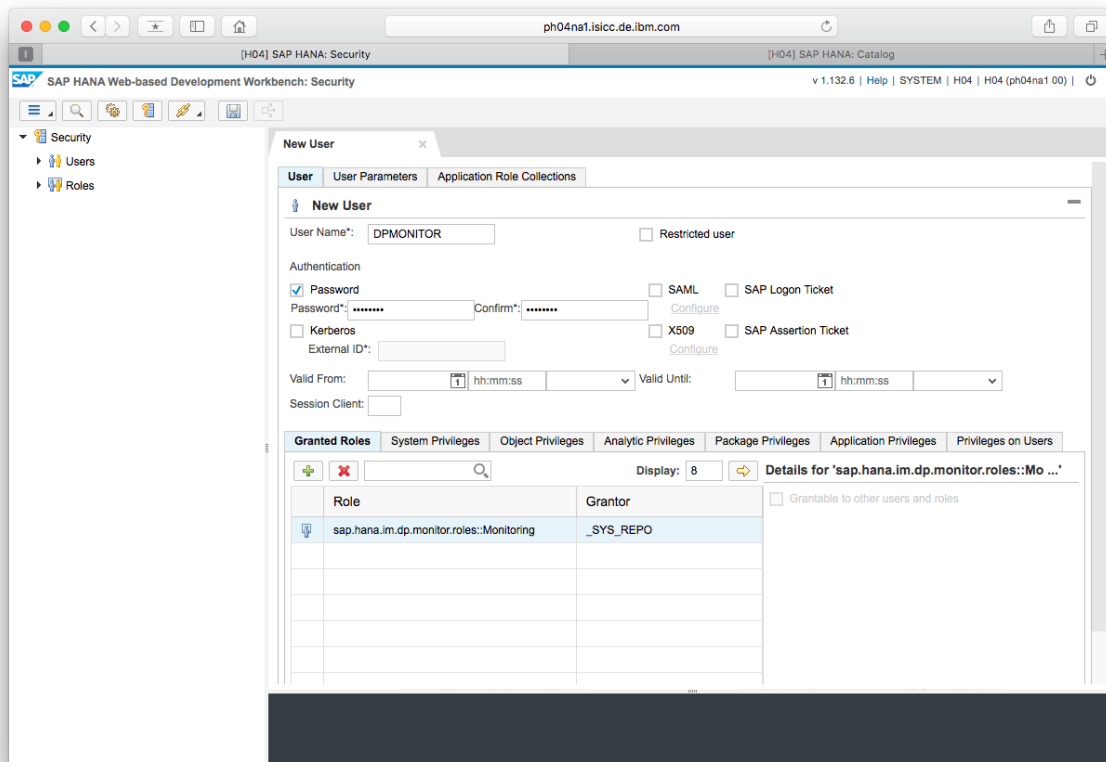


In case the DP Agent Admin user shall configure a proxy scenario (communication to DP Agent via HTTP protocol), application privilege *sap.hana.im.dp.admin::Administrator* is required in addition:



2.2.2 Create an DP Agent Monitor User in SAP HANA

Create another user called *DPMONITOR* as DP Agent Monitor user in SAP HANA. The DP Agent Monitor needs the role *sap.hana.im.dp.monitor.roles::Monitoring*.



2.2.3 Enable the Data Provisioning Server

Per default, the Data Provisioning Server is disabled in SAP HANA.

In a multi-database container (MDC) environment, the Data Provisioning Server for the tenant database is enabled using an *ALTER DATABASE* SQL command. Enable the Data Provisioning Server for the respective tenants:

```
ALTER DATABASE <database_name> ADD 'dpserver' AT LOCATION
'<hostname>[:<port_number>]'
```

The SAP HANA Web IDE, does not support *ALTER DATABASE* statements. Use the SAP HANA Studio SQL Console or hdbsql to connect to the to the *SYSTEMDB* and issue the following command accordingly:

```
ALTER DATABASE H04 ADD 'dpserver' at LOCATION 'ph04na1'
```

SYSTEMDB@H04 (SYSTEM) 9.153.164.71 00

SQL

```
ALTER DATABASE H04 ADD 'dpserver' AT LOCATION 'ph04na1'
```

Statement 'ALTER DATABASE H04 ADD 'dpserver' AT LOCATION 'ph04na1''
successfully executed in 8.138 seconds (server processing time: 8.137 seconds) - Rows Affected: 0

2.2.4 Download SDI components

The required SDI software components can be downloaded from the SAP Software Download Center <https://launchpad.support.sap.com/#/softwarecenter>.

Select *INSTALLATIONS & UPGRADES > By Alphabetical Index (A-Z) > H > SAP HANA SDI > SAP HANA SDI 2.0 > COMPRISED SOFTWARE COMPONENT VERSIONS*

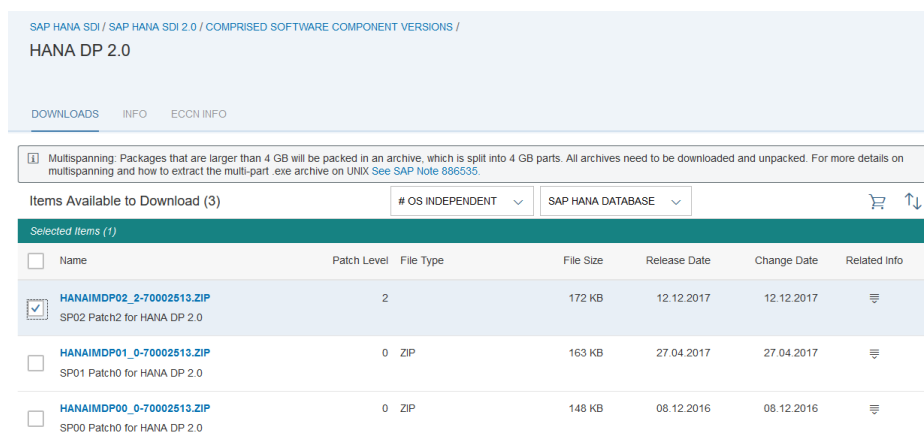
Download Delivery Unit HANA_IM_DP

Select *SAP HANA DP 2.0*. The Data Provisioning delivery unit is provided as a standard ZIP file for all available operating systems of the SAP HANA database.

The delivery unit provides the following functionalities:

- Monitoring**
 A browser based interface allows to monitor agents, tasks and remote subscriptions created in the SAP HANA system.
- Proxy application**
 Allows communication between SAP HANA and the Data Provisioning Agent in case the Data Provisioning Agent is deployed in a different network segment behind a firewall, or in a cloud scenario (SAP HANA is hosted in a cloud datacenter).
- Admin application**
 The Admin application allows to configure the Data Provisioning Agent in case SAP HANA and Data Provisioning Agent are in different network segments involving the proxy application.

Download the appropriate patch level version matching the support package release of the installed HANA database system:



SAP HANA SDI / SAP HANA SDI 2.0 / COMPRISED SOFTWARE COMPONENT VERSIONS / HANA DP 2.0

DOWNLOADS INFO ECCN INFO

Multispanning: Packages that are larger than 4 GB will be packed in an archive, which is split into 4 GB parts. All archives need to be downloaded and unpacked. For more details on multispanning and how to extract the multi-part .exe archive on UNIX See SAP Note 886535.

Items Available to Download (3) # OS INDEPENDENT SAP HANA DATABASE

Selected Items (1)							
<input type="checkbox"/>	Name	Patch Level	File Type	File Size	Release Date	Change Date	Related Info
<input checked="" type="checkbox"/>	HANAIMDP02_2-70002513.ZIP SP02 Patch2 for HANA DP 2.0	2		172 KB	12.12.2017	12.12.2017	
<input type="checkbox"/>	HANAIMDP01_0-70002513.ZIP SP01 Patch0 for HANA DP 2.0	0	ZIP	163 KB	27.04.2017	27.04.2017	
<input type="checkbox"/>	HANAIMDP00_0-70002513.ZIP SP00 Patch0 for HANA DP 2.0	0	ZIP	148 KB	08.12.2016	08.12.2016	

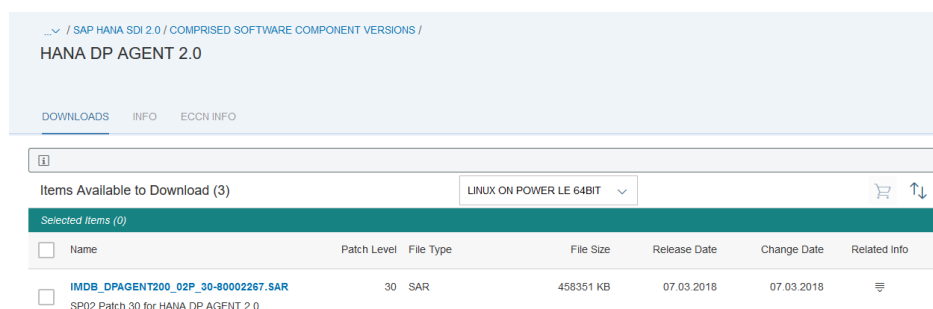
(Example: HANAIMDP02_2-70002513.ZIP)

Download Data Provisioning Agent

The Data Provisioning Agent links the SAP HANA system with the remote data source.

In our setup, the Data Provisioning Agent is installed on a separate Linux partition on an IBM Power Systems server (Operating System SUSE SLES 12 – Linux on Power LE 64Bit):

Navigate to *HANA DP AGENT 2.0* in the *COMPRISED SOFTWARE COMPONENT VERSIONS* section, select the desired operating system (*LINUX ON POWER LE 64BIT*), and download the Data Provisioning Agent SAR file:



(Example: SP02 Patch 30 for HANA DP AGENT 2.0)

2.2.5 Download Oracle JDBC Driver

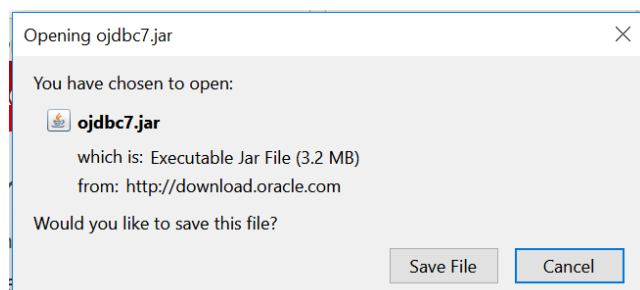
The SAP Product and Availability Matrix (SAP PAM) entries for Smart Data Integration document the required client libraries, which need to be installed for the adapter on the Data Provisioning Agent:

Adapter	Name	Version	SAP Kernel	Client Libraries	RDBMS Server OS (1)
OracleECCAdapter ⁽³⁾	ECC 6.0 on Oracle Database	11g (11.1 & 11.2) 11g (11.1 & 11.2) ASM	720, 721 and greater, 730, 741,742	ojdbc6.jar	⁽³⁾ Same as DP Agent OS plus HP-UX11.31 ⁽²⁾ , AIX 7.1 and Solaris 10
		12c (12.1.0.1) 12c (12.1.0.2)		ojdbc7.jar	
OracleLogReaderAdapter	Oracle Database	11g (11.1, 11.2) 11g (11.1, 11.2) ASM	N/A	ojdbc6.jar xdb6.jar xmlparserv2.jar	⁽³⁾ Same as DP Agent OS plus HP-UX11.31 ⁽²⁾ , AIX 7.1 and Solaris 10
		12c (12.1.0.1) 12c (12.1.0.2)		ojdbc7.jar xdb6.jar xmlparserv2.jar	
		12c (12.2.0.1) ⁽⁴⁾		ojdbc8.jar JVM 1.8	

Due to licensing requirements, the Oracle JDBC libraries are not shipped together with the agent, but need to be obtained from the Oracle Technology Network (OTN):

(<http://www.oracle.com/technetwork/database/application-development/jdbc/downloads/index.html>)

For our environment, ojdbc7.jar was chosen for Oracle 12c.

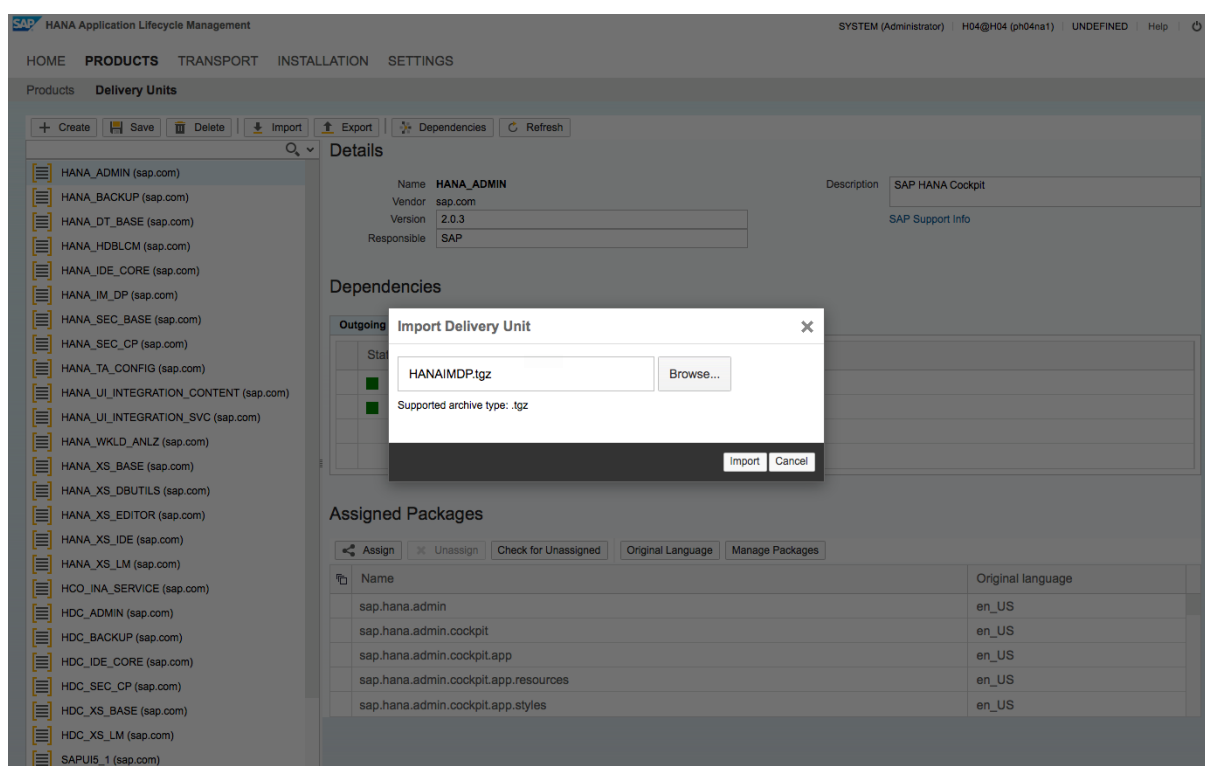


Later, the ODBC driver is installed into the `~/lib` directory of the Data Provisioning Agent. The Oracle Log Reader adapter of the Data Provisioning Agent will use the JDBC driver to connect to the Oracle database system.

2.2.6 Deploy Data Provisioning Delivery Unit

The delivery unit HANA_IM_DP needs to be deployed in the SAP HANA system. When SAP HANA is deployed in a multitenant database container configuration, the delivery unit must be imported into the tenant database. The user deploying the delivery unit with the SAP HANA Application Lifecycle Management tool needs the security role `sap.hana.xs.lm.roles::Administrator`.

1. Unpack the downloaded ZIP file.
2. Login to the SAP HANA Web IDE.
3. Switch to the *Lifecycle Management* module.
4. Click on the *Delivery Units* tile and then *Import* button.
5. Click *Browse*, navigate to and select the unpacked HANAIMDP.tgz file.
6. Click *Import* to start the installation of the delivery unit.



A pop-up window shows the results of a test import, click the *Import* button to start the actual deployment.

Confirm Import of Delivery Unit ✕

Delivery Unit Details

Name	HANA_IM_DP	Version	2.2.2
Vendor	sap.com	Responsible	SAP

Objects Test Import Result

Status	Object	Type	Package
✓		project	sap.hana.im.dp
✓		xsapp	sap.hana.im.dp
✓		xsaccess	sap.hana.im.dp.admin
✓		xsprivileges	sap.hana.im.dp.admin
✓	dpadmin	xsjs	sap.hana.im.dp.admin
✓		xsaccess	sap.hana.im.dp.monitor
✓		xsprivileges	sap.hana.im.dp.monitor

After a successful import, the name HANA_IM_DP (sap.com) appears in the list of delivery units in the left sidebar. Selecting this entry shows some detailed information like the installed version for this delivery unit.

The screenshot shows the SAP HANA Application Lifecycle Management interface. The top navigation bar includes 'HOME', 'PRODUCTS', 'TRANSPORT', 'INSTALLATION', and 'SETTINGS'. The main content area is titled 'Delivery Units' and contains a list of delivery units on the left sidebar. The selected unit, 'HANA_IM_DP (sap.com)', is highlighted. The main panel displays the 'Details' for this unit, including its Name, Vendor (sap.com), Version (2.2.2), and Responsible (SAP). Below the details, there is a 'Dependencies' section showing an outgoing dependency on 'HANA_IDE_CORE'. At the bottom, the 'Assigned Packages' section lists several packages with their original languages, all set to 'en'.

2.2.7 Install the Data Provisioning Agent

The Data Provisioning Agent provides secure connectivity between the SAP HANA database and the adapter-based data sources.

In our test environment, the Data Provisioning Agent is installed on an IBM Power Systems partition with a SLES12 operating system.

On the DP Agent partition, perform the following steps:

1. Create the user which owns the DP Agent installation (user *dpagent*):

```
groupadd --gid 79 sapsys
useradd -d /usr/sap/dataprovagent -m -s /bin/bash -g sapsys dpagent
```

2. Logon as the *dpagent* user.

- Download the SAPCAR tool from the SAP support portal or from an existing installation (e.g. /usr/sap/H04/SYS/exe/hdb/SAPCAR) and copy it to the /tmp directory.
- Transfer Data Provisioning Agent SAR archive (e.g. IMDB_DPAGENT200_02P_30-80002267.SAR) to the /tmp directory.
- Extract the Data Provisioning Agent SAR archive:

```
mkdir /tmp/IMDB_DPAGENT200_02P_30
cd /tmp/IMDB_DPAGENT200_02P_30
/tmp/SAPCAR -manifest SIGNATURE.SMF \
-xvf ../IMDB_DPAGENT200_02P_30-80002267.SAR
```

The agent package gets extracted to subdirectory
/tmp/HANA_DP_AGENT_20_PPC_64LE

- Change to the directory of the extracted package and run the DP Agent installation:

```
cd /tmp/HANA_DP_AGENT_20_PPC_64LE
./hdbinst --silent --batch --path="/usr/sap/dataprovagent"

SAP HANA Data Provisioning Agent installation kit detected.
SAP HANA Lifecycle Management - dataprovagent Installation 2.2.3.0.0
*****
Checking installation...
Preparing package 'dataprovagent'...
Preparing package 'Installer'...
Installing SAP HANA Data Provisioning Agent to /usr/sap/dataprovagent...
Installing package 'dataprovagent'...
Installing package 'Installer'...
#####
IMPORTANT NOTE:
#####
You have installed Data Provisioning Agent with default settings. The default
settings use insecure channel to communicate with HANA server. For secure
communication, SSL settings must be configured via the Data Provisioning
Agent Configuration tool after the installation is complete. Please see SAP
HANA Smart Data Integration Installation and Configuration Guide for details
on configuring SSL between the Agent and HANA server.
#####
Installation done
Log file written to '/var/tmp/hdb_dataprovagent_2018-03-
20_11.05.06_12877/hdbinst_dataprovagent.log' on host 'ph04pa1'.
```



3. Set-Up Oracle 12c LogReader adapter
Copy the downloaded JDBC libraries (ojdbc7.jar) to the library directory in /usr/sap/dataprovagent/lib.

2.2.8 Configure the Data Provisioning Agent

The configuration of the Data Provisioning Agent requires several steps:

1. Start the Agent.
2. Define the Connection to the SAP HANA System.
3. Register the Agent on the SAP HANA System.
4. Register the Adapter with the SAP HANA System.

These steps are executed with the *Agent Admin* user on the Data Provisioning Agent partition.

Start the Agent

Login as *Agent Admin* user on the LPAR (user *dpagent*), change to the *~/bin* directory, and start the command-line agent configuration tool:

```
> cd bin
> ./agentcli.sh -configAgent
```

```
*****
                DPAgent Configuration Tool
*****
1. Agent Status
2. Start or Stop Agent
3. Agent Preferences
4. SSL Keystores
5. SAP HANA Connection
6. Agent Registration
7. Adapter Registration
8. Custom Adapters
9. Agent & Adapter Versions
q. Quit
b. Back
*****
```

Select 2 to for the Start/Stop menu:

```
*****
                Start or Stop Agent
*****
1. Start Agent
2. Stop Agent
3. Ping Agent
q. Quit
b. Back
*****
```

Select 1 to start the agent:

```
Agent service location: /usr/sap/dataprovagent/bin/dpagent_service.sh
dpagent_service.sh location: /usr/sap/dataprovagent/bin/dpagent_service.sh
Starting agent service. Command: /usr/sap/dataprovagent/bin/dpagent_service.sh
start
Start service return code: 0
```

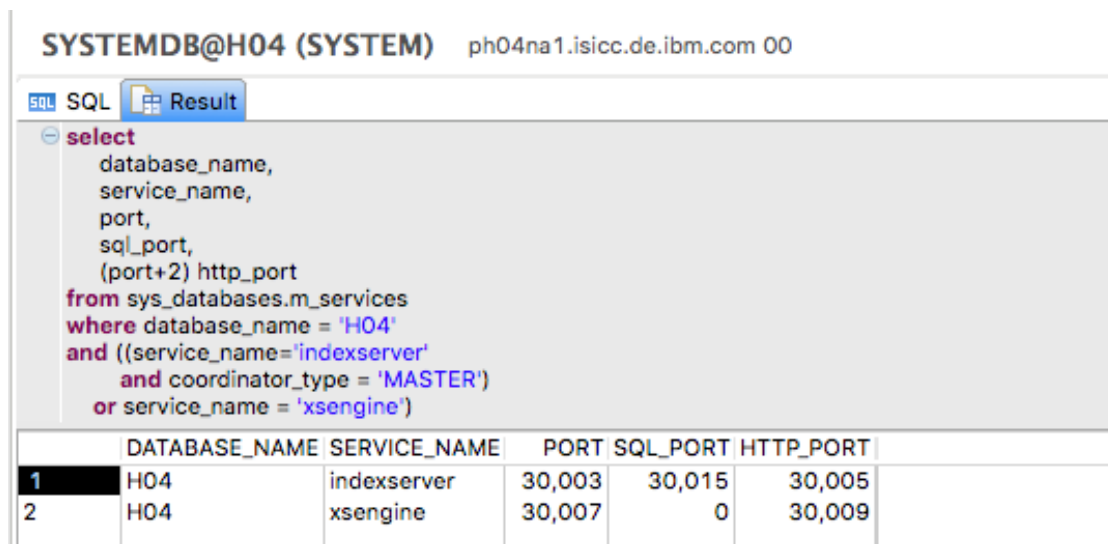
Press Enter to continue...

Result: agent is started.

Define the Connection to the SAP HANA System

Connect with SAP HANA Studio to the *System DB* and run the following SQL command to identify the appropriate SQL PORT number:

```
select
  database_name,
  service_name,
  port,
  sql_port,
  (port+2) http_port
from sys_databases.m_services
where database_name = 'H04'
and ((service_name='indexserver'
      and coordinator_type = 'MASTER')
     or service_name = 'xsengine')
```



The screenshot shows the SAP HANA Studio interface. At the top, it displays 'SYSTEMDB@H04 (SYSTEM)' and 'ph04na1.isicc.de.ibm.com 00'. Below this, there are tabs for 'SQL' and 'Result'. The SQL tab contains the same query as shown in the code block above. The Result tab shows a table with the following data:

	DATABASE_NAME	SERVICE_NAME	PORT	SQL_PORT	HTTP_PORT
1	H04	indexserver	30,003	30,015	30,005
2	H04	xsengine	30,007	0	30,009

Login to the *Agent Admin* user on the partition (user dpagent), change to the ~/bin directory, and start the command-line agent configuration tool:

```
> cd bin
> ./agentcli.sh -configAgent
```

```
*****
DPAgent Configuration Tool
*****
1. Agent Status
2. Start or Stop Agent
3. Agent Preferences
4. SSL Keystores
5. SAP HANA Connection
6. Agent Registration
7. Adapter Registration
8. Custom Adapters
9. Agent & Adapter Versions
q. Quit
b. Back
*****
```



Select 5 to enter the SAP HANA Connection menu:

```
*****
                SAP HANA Connection
*****
1. Connect to SAP HANA on Cloud (HTTP/HTTPS)
2. Connect to SAP HANA on Premise (TCP)
q. Quit
b. Back
```

Select 2 to Connect to SAP HANA on Premise:

```
*****
                Connect to SAP HANA on Premise (TCP)
*****
Enter Use SSL (requires configuration) (true): Valid options: true|false
false
Enter Host Name:
ph04nal
Enter Port Number (30015):
30015
Enter Agent Admin HANA User:
DPAGENT
Enter Agent Admin HANA User Password:

Enter Agent Admin HANA User Password: (confirm)

Calling Connect to SAP HANA on Premise (TCP) (connectHanaViaTcp)
Agent configuration tool is connected to SAP HANA server.

Press Enter to continue...
```

Result: the connectivity to SAP HANA is defined for the agent

Register Agent with SAP HANA

Start the command-line agent configuration tool and connect to SAP HANA.

```
*****
                DPAgent Configuration Tool
*****
1. Agent Status
2. Start or Stop Agent
3. Agent Preferences
4. SSL Keystores
5. SAP HANA Connection
6. Agent Registration
7. Adapter Registration
8. Custom Adapters
9. Agent & Adapter Versions
q. Quit
b. Back
*****
```

Select 6 to enter the Agent Registration menu:

```
*****
                Agent Registration
*****
1. Register Agent
2. Unregister Agent
q. Quit
b. Back
```



Select 1 to register the agent.

Result: The DP Agent is registered in the SAP HANA database.

Register Data Provisioning Adapters

Start the command-line agent configuration tool and connect to SAP HANA.

Select 7 to enter the Adapter Registration menu.

```

*****
Adapter Registration
*****
1. Display Adapters
2. Register Adapter
3. Unregister Adapter
q. Quit
b. Back
*****

```

Select 2 to register an adapter.

Specify the name of the adapter to register with SAP HANA

Enter adapter name:

OracleLogReaderAdapter

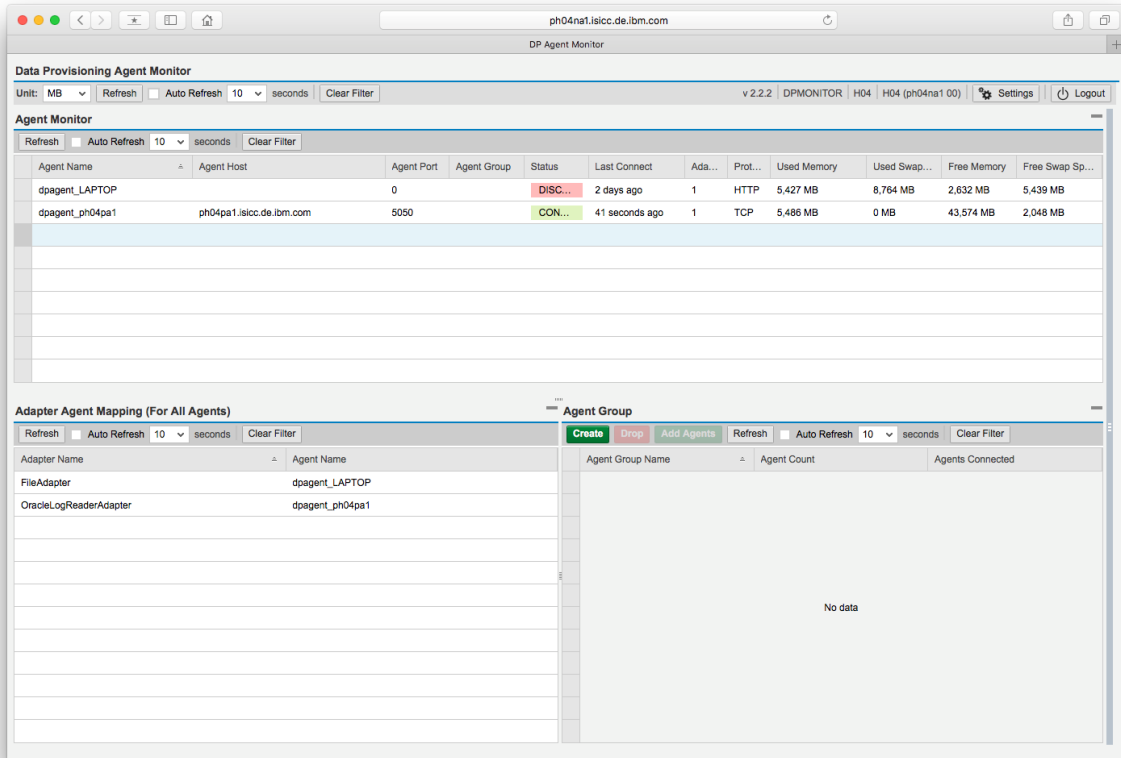
Result: the Adapter 'OracleLogReaderAdapter' on the DP Agent is successfully registered.

You can verify the result in the Data Provisioning Agent Monitor available at

<http://<host>:<port>/sap/hana/im/dp/monitor/?view=DPAgentMonitor>.

Login with the previously created *DPMONITOR* userid. In our demo landscape, the URL is

<http://ph04na1.isicc.de.ibm.com:8000/sap/hana/im/dp/monitor/?view=DPAgentMonitor>



The screenshot displays the 'Data Provisioning Agent Monitor' interface. At the top, it shows the unit as 'MB' and an auto-refresh interval of 10 seconds. The 'Agent Monitor' section contains a table with the following data:

Agent Name	Agent Host	Agent Port	Agent Group	Status	Last Connect	Ada...	Prot...	Used Memory	Used Swap...	Free Memory	Free Swap Sp...
dpagent_LAPTOP		0		DISC	2 days ago	1	HTTP	5,427 MB	8,764 MB	2,632 MB	5,439 MB
dpagent_ph04pa1	ph04pa1.isicc.de.ibm.com	5050		CON	41 seconds ago	1	TCP	5,486 MB	0 MB	43,574 MB	2,048 MB

Below the agent list, the 'Adapter Agent Mapping (For All Agents)' table shows:

Adapter Name	Agent Name
FileAdapter	dpagent_LAPTOP
OracleLogReaderAdapter	dpagent_ph04pa1

The 'Agent Group' section is currently empty, displaying 'No data'.

The OracleLogReaderAdapter is active on the Data Provisioning Agent. In the next step, the remote data source can be defined in SAP HANA.

2.3 SDI Application Setup

2.3.1 Create a Remote Source in SAP HANA

We can now configure the Oracle NW4 database as a remote source using the SAP HANA Web IDE.

The connection details of the remote source can be extracted from the listener.ora configuration of the Oracle database system:

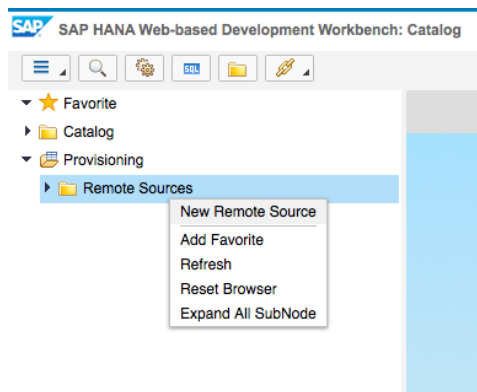
```

LISTENER =
  (ADDRESS_LIST =
    (ADDRESS =
      (PROTOCOL = IPC)
      (KEY = NW4.WORLD)
    )
    (ADDRESS=
      (PROTOCOL = IPC)
      (KEY = NW4)
    )
    (ADDRESS =
      (COMMUNITY = SAP.WORLD)
      (PROTOCOL = TCP)
      (HOST = psapnw4)
      (PORT = 1521)
    )
  )
  )

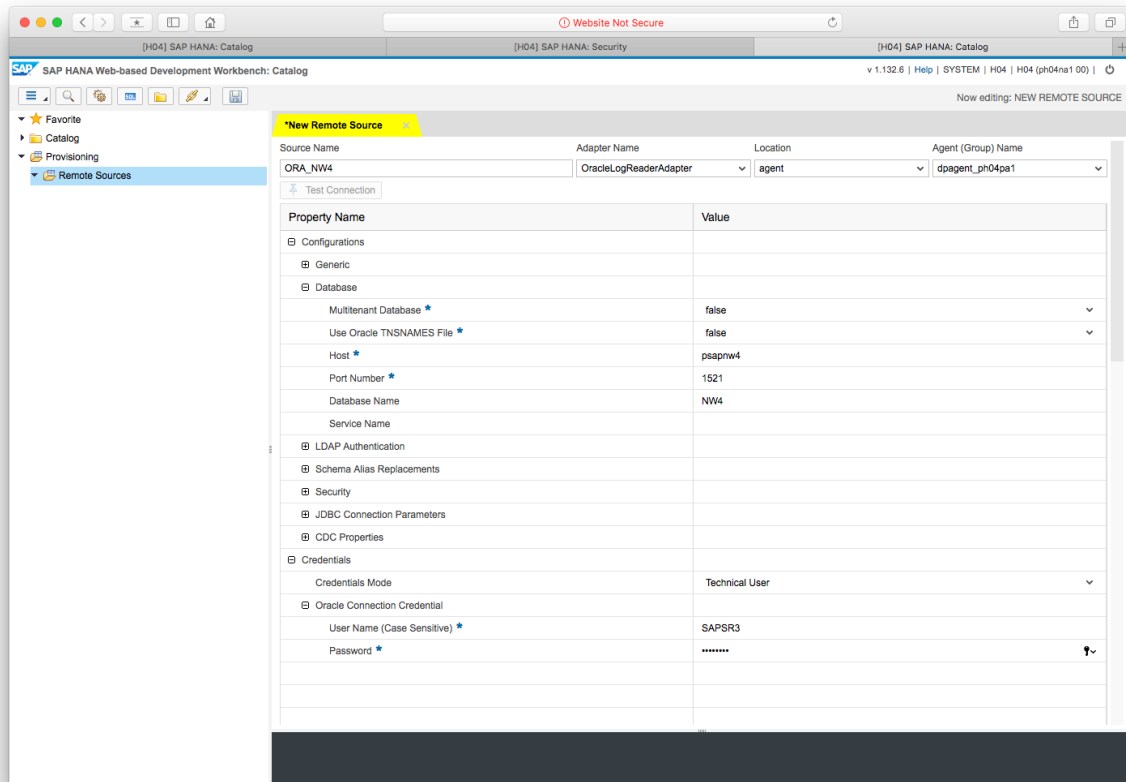
```

Login as SYSTEM user in the SAP HANA Web IDE and:

1. Switch to the *Catalog* module.
2. Open *Provisioning* in the sidebar.
3. Right-click on *Remote Sources* and select *New Remote Source*.
4. Open the *Adapter Name* drop-down list and select *OracleLogReaderAdapter*.
The *Location* should be *agent* and *Agent (Group) Name* our previously configured DP agent.



5. Provide a *Source Name* in the *New Remote Source* tab.
6. Open the *Database* property and fill in the *Host*, *Port Number*, and *Database Name* of the source system.
7. Open the *Credentials* property and switch the *Credentials Mode* to *Technical User*.
8. Open the *Oracle Connection Credential* property and fill in the *User Name* and *Password* properties using the schema credential of the source database.
9. Click the *Save* icon to create the remote source.



2.3.2 Create Remote Source with a SQL statement

Alternatively a remote source can be created by executing a SQL in the SAP HANA database:

```
CREATE REMOTE SOURCE "ORA_NW4" ADAPTER "OracleLogReaderAdapter" AT LOCATION AGENT
"dpagent_ph04pa1"
CONFIGURATION
'<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ConnectionProperties name="configurations">
<PropertyGroup name="database" displayName="Database">
<PropertyEntry name="pds_host_name" displayName="Host">psapnw4</PropertyEntry>
<PropertyEntry name="pds_port_number" displayName="Port
Number">1521</PropertyEntry>
<PropertyEntry name="pds_database_name" displayName="Database
Name">NW4</PropertyEntry>
</PropertyGroup>
</ConnectionProperties>
' WITH CREDENTIAL TYPE 'PASSWORD' USING
'<CredentialEntry name="credential">
<user>oranw4</user>
<password>Secure123</password>
</CredentialEntry>';
```

Note the single quotes surrounding the *CONFIGURATION* and *WITH CREDENTIAL TYPE* sections. The section following *CONFIGURATION* contains the info string specifying the Oracle database connection details. The section following *WITH CREDENTIAL TYPE* contains the credentials section where user name and password for the Oracle database access is specified.

(See also [SAP Note 2459991 - How to create SDI remote source using the SQL console?](#) or the SAP HANA Administration Guide for details)

2.4 SAP HANA SDI Monitoring

2.4.1 Data Provisioning Adapter Monitoring

The Data Provisioning adapter on Linux provides a script for (re-) start-, stop- and status monitoring.

- Logon to the Data Provisioning Server partition with the credentials of the installation owner (user dpagent)
- Run `~/bin/dpagent_service.sh ping`
to verify the status of the SAP HANA Data Provisioning Agent
- In case the Data Provisioning Server is not running, it can be started via `~/bin/dpagent_service.sh start`

The Data Provisioning Server will be started as background process.

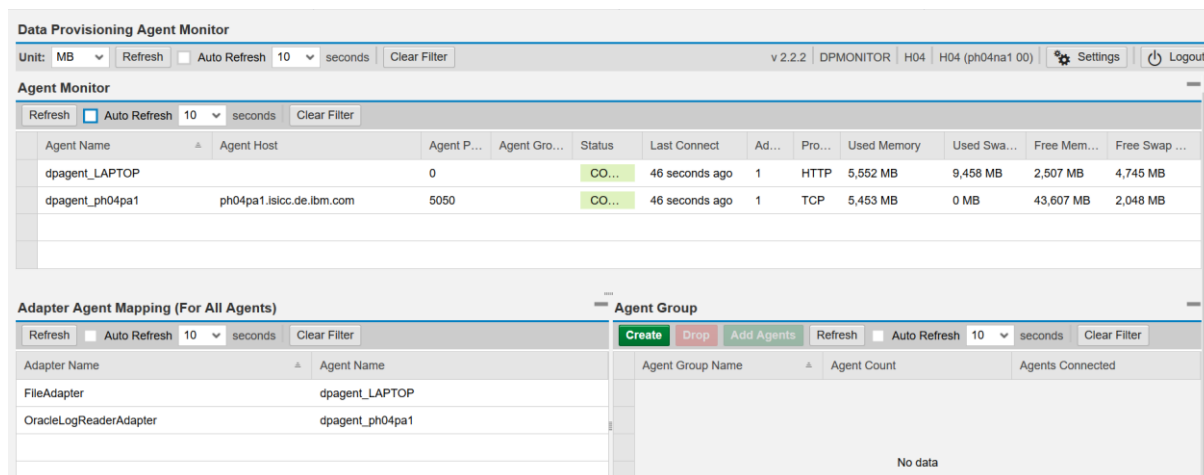
- The Data Provisioning Server can be stopped via `~/bin/dpagent_service.sh stop`

2.4.2 Monitoring using the HANA_IM_DP Deployment Unit

The *DPMONITOR* user can connect to the HANA_IM_DP deployment unit in the XS Engine. Different monitoring views are available. (URLs shown reflect our demo landscape):

DP Agent Monitor

[http:// ph04na1.isicc.de.ibm.com:8000/sap/hana/im/dp/monitor/?view=DPAgentMonitor](http://ph04na1.isicc.de.ibm.com:8000/sap/hana/im/dp/monitor/?view=DPAgentMonitor)



Data Provisioning Agent Monitor

Unit: MB | Refresh | Auto Refresh 10 seconds | Clear Filter | v 2.2.2 | DPMONITOR | H04 | H04 (ph04na1 00) | Settings | Logout

Agent Monitor

Agent Name	Agent Host	Agent P...	Agent Gro...	Status	Last Connect	Ad...	Pro...	Used Memory	Used Swa...	Free Mem...	Free Swap ...
dpagent_LAPTOP		0		CO...	46 seconds ago	1	HTTP	5,552 MB	9,458 MB	2,507 MB	4,745 MB
dpagent_ph04pa1	ph04pa1.isicc.de.ibm.com	5050		CO...	46 seconds ago	1	TCP	5,453 MB	0 MB	43,607 MB	2,048 MB

Adapter Agent Mapping (For All Agents)

Adapter Name	Agent Name
FileAdapter	dpagent_LAPTOP
OracleLogReaderAdapter	dpagent_ph04pa1

Agent Group

Agent Group Name | Agent Count | Agents Connected

No data

DP Subscription Monitor

<http://ph04na1.isicc.de.ibm.com:8000/sap/hana/im/dp/monitor/?view=DPSubscriptionMonitor>



Data Provisioning Remote Subscription Monitor

Unit: MB | Seconds | Refresh | Auto Refresh 10 seconds | Clear Filter | v 2.2.2 | DPMONITOR | H04 | H04 (ph04na1 00) | Settings | Logout

Remote Source Monitor

Refresh | Auto Refresh 10 seconds | Clear Filter

Remote Source Name	Adapter Name	Location	Agent Name	Remote Sou...	Subscriptions
ORA_NW4	OracleLogReaderAdapter	agent	dpagent_ph04pa1	OK	0

.....

Remote Subscription Monitor (For All Remote Sources)

Refresh | Auto Refresh 10 seconds | Clear Filter

Subscription Name	Schema Na...	Remote Sourc...	Design Time N...	Design Time T...	Valid	Subscription State	Replication ...	Last processed Tr...	Subscriptio...	Target T...
No Remote Subscriptions										

.....

Remote Subscription Statistics (For All Remote Sources)

Refresh | Auto Refresh 10 seconds | Clear Filter

Subscription N...	Schema Name	Received Count	Applied Count	Received Size	Applied Size	Rejected Count	Last Message Received	Last Message Applied
-------------------	-------------	----------------	---------------	---------------	--------------	----------------	-----------------------	----------------------

DP Task Monitor

[http:// ph04na1.isicc.de.ibm.com:8000/sap/hana/im/dp/monitor/?view=IMTaskMonitor](http://ph04na1.isicc.de.ibm.com:8000/sap/hana/im/dp/monitor/?view=IMTaskMonitor)

Data Provisioning Task Monitor

Unit: KB | Seconds | Refresh | Auto Refresh 10 seconds | Clear Filter | v 2.2.2 | DPMONITOR | H04 | H04 (ph04na1 00) | Settings | Logout

Task Overview

Start | Schedules | Notifications | Refresh | Auto Refresh 10 seconds | Clear Filter

Schema Name	Task Name	Design Time Name	Design Time Type	Realtime Design Ti...	Create Time	Has Table Type Input	Realtime	Memory Size
No Tasks								

.....

Task Execution Monitor (For All Tasks)

Stop | Remote Statements | Execute Remaining Partitions | Number of rows 500 | Refresh | Auto Refresh 10 seconds | Clear Filter

Task Name	Schema Name	Task Execution ID	Partition Count	Start Time	Duration	Status	Total Progress	Processed Rec...	Async
No Task Executions									

.....

Task Operation Execution Monitor (For All Tasks)

Number of rows 500 | Refresh | Auto Refresh 10 seconds | Clear Filter

Operation	Task Execution ID	Partition ID	Operation Type	Start Time	Duration	Status	Progress	Processed Records
-----------	-------------------	--------------	----------------	------------	----------	--------	----------	-------------------

3 Accessing Remote Data Source Scenarios

3.1 SAP SDI Virtual Table Scenario

This section demonstrates the definition of a virtual table providing access to a source table in the SAP NetWeaver/Oracle system. SAP provides a sample flight data model implementing a simplified flight booking system, which is typically used for ABAP learning and training. Call transaction *se38* and execute the ABAP report *SAPBC_DATA_GENERATOR* to generate this set of sample tables and data content.

In this demonstration, we use a separate schema in the HANA database to store the virtual table.

We use the SAP HANA Web IDE to illustrate the following steps:

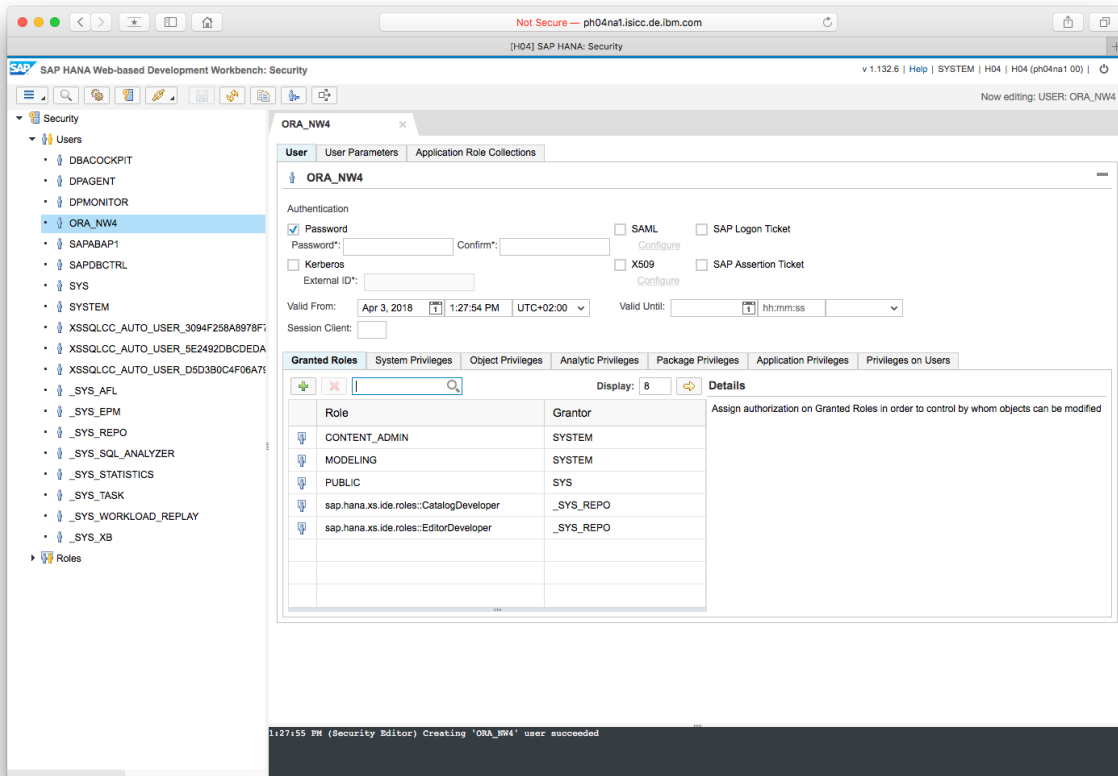
- 1) Create a new user and assign required roles.
- 2) Grant access to user schema.
- 3) Create a virtual table in the schema connecting to a remote data source.
- 4) Access data in the virtual table.

3.1.1 Create User and Schema

We create a new schema to store the virtual table and a new user that will be used to access the data in the Oracle source system.

Open the SAP HANA Web-based Development Workbench, the URL in our demo setup is: <http://ph04na1.isicc.de.ibm.com:8000/sap/hana/ide/>

1. Login to the SAP HANA Web-based Development Workbench with the *SYSTEM* user.
2. Switch to the *Security* module to create a new userid.
3. Select *Menu > New > User*, enter a user name and password, and add the following roles:
 - *sap.hana.xs.ide.roles::CatalogDeveloper*
 - *sap.hana.xs.ide.roles::EditorDeveloper*
 - *CONTENT_ADMIN*
 - *MODELING*
4. Click the *Save* icon to create the new user.



3.1.2 Grant Schema Access

To allow the *SYSTEM* user to create a new virtual table definition in our *ORA_NW4* schema, we need to provide the right authorizations to the *SYSTEM* user.

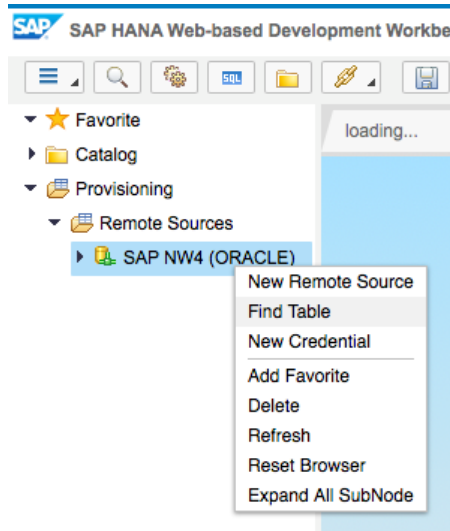
1. Login to the SAP HANA Web IDE with the *ORA_NW4* userid.
2. Switch to the *Catalog* module.
3. Select *Open SQL Console* and execute the following statement to grant the required rights and permissions to the *SYSTEM* user.

```
GRANT SELECT, INSERT, UPDATE, DELETE, EXECUTE ON SCHEMA ORA_NW4 TO SYSTEM
```

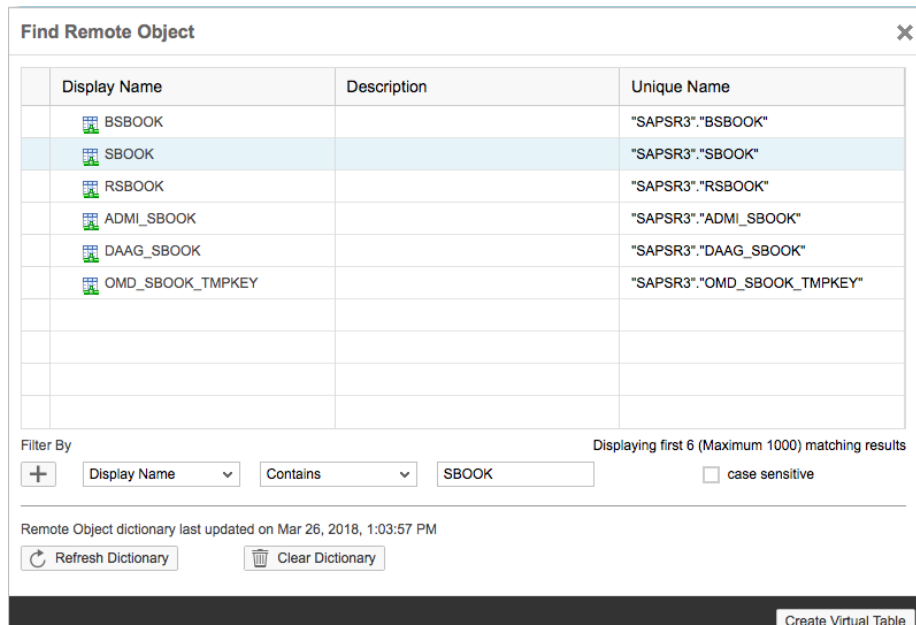
3.1.3 Create Virtual Table

The next step is to create the virtual table in the new schema. Login as *SYSTEM* user and

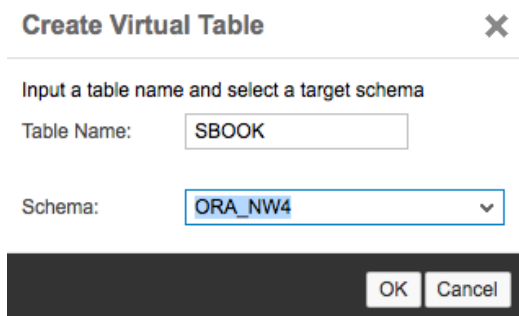
1. Open the *Provisioning > Remote Sources* section in the sidebar.
2. Right-click on the remote source system and select *Find Table*.
The first time you try the *Find Table* action, a pop-up window will show that this action needs a dictionary. Create one by clicking the *Create Dictionary* button.



3. Enter *SBOOK* as a filter to find the table *SBOOK* in schema *SAPSR3*.



4. Select the appropriate row and click *Create Virtual Table*.



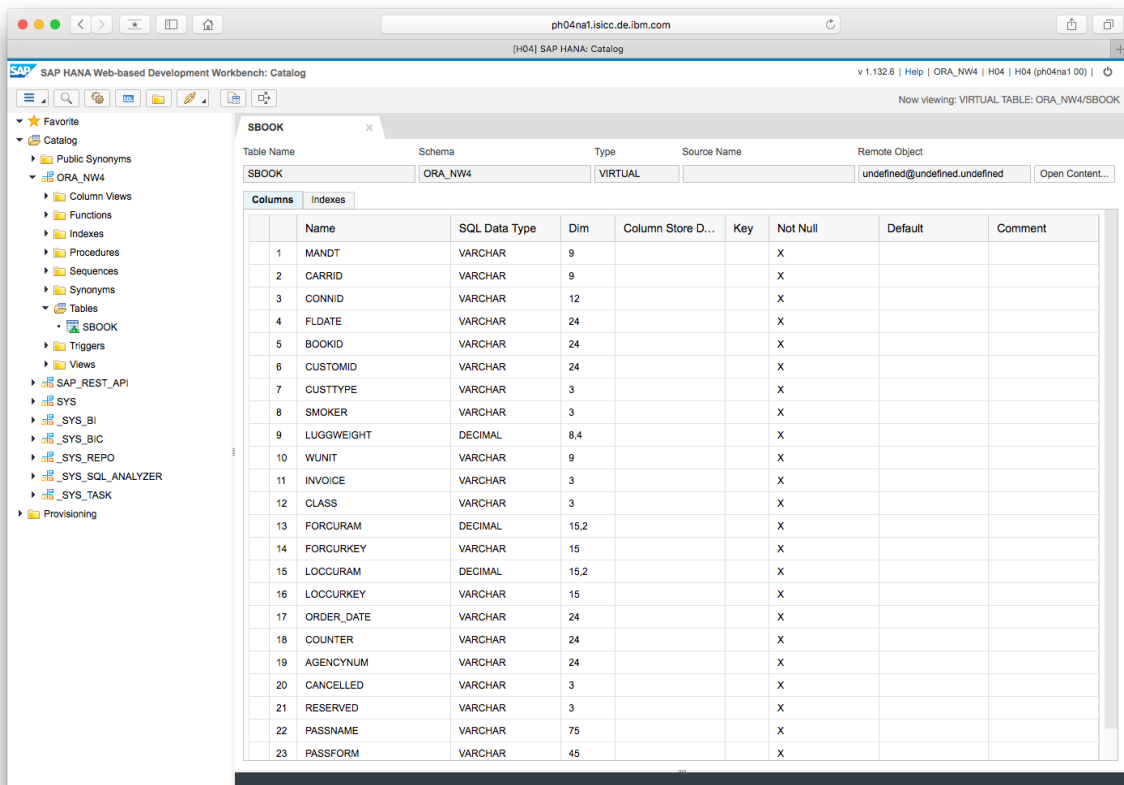
5. Enter a name for the virtual table, the schema to store the table and confirm with the *OK* button.

The virtual table *SBOOK* has been created in our *ORA_NW4* schema. To check the content of the table, we sign-out of the SAP HANA Web IDE and will continue with the *ORA_NW4* userid.

3.1.4 Check Virtual Table Definition and Content

1. Login the SAP HANA Web IDE with the *ORA_NW4* user.
2. Switch to the *Catalog* module.
3. On the navigation sidebar, open *Catalog > ORA_NW4 > Tables* and click on the virtual table *SBOOK*.

This shows a new tab with the table structure of the remote table.



The screenshot shows the SAP HANA Web IDE interface. The main window displays the table structure for the virtual table *SBOOK* in the *ORA_NW4* schema. The table is of type *VIRTUAL* and its source is *undefined@undefined.undefined*. The table structure is shown in a table with the following columns:

	Name	SQL Data Type	Dim	Column Store D...	Key	Not Null	Default	Comment
1	MANDT	VARCHAR	9			X		
2	CARRID	VARCHAR	9			X		
3	CONNID	VARCHAR	12			X		
4	FLDATE	VARCHAR	24			X		
5	BOOKID	VARCHAR	24			X		
6	CUSTOMID	VARCHAR	24			X		
7	CUSTTYPE	VARCHAR	3			X		
8	SMOKER	VARCHAR	3			X		
9	LUGGWEIGHT	DECIMAL	8,4			X		
10	WUNIT	VARCHAR	9			X		
11	INVOICE	VARCHAR	3			X		
12	CLASS	VARCHAR	3			X		
13	FORCURAM	DECIMAL	15,2			X		
14	FORCURKEY	VARCHAR	15			X		
15	LOCCURAM	DECIMAL	15,2			X		
16	LOCCURKEY	VARCHAR	15			X		
17	ORDER_DATE	VARCHAR	24			X		
18	COUNTER	VARCHAR	24			X		
19	AGENCYNUM	VARCHAR	24			X		
20	CANCELLED	VARCHAR	3			X		
21	RESERVED	VARCHAR	3			X		
22	PASSNAME	VARCHAR	75			X		
23	PASSFORM	VARCHAR	45			X		

4. Click on the *Open Content* button to get a list with content of the first 1000 rows of the remote database table.

SAP HANA Web-based Development Workbench: Catalog v 1.132.6 | Help | ORA_NW4 | H04 | H04 (ph04na1 00) |

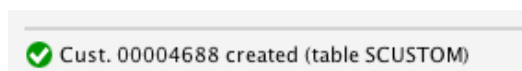
Now editing: ORA_NW4/SBOOK

#	MANDT	CARRID	CONNID	FLDATE	BOOKID	CUSTOMID	#
1	000	LH	0400	19950228	00000000	00000002	
2	000	LH	0400	19950228	00000001	00000001	B
3	000	LH	0400	19950228	00000002	00000002	P
4	000	LH	0400	19950228	00000003	00000003	P
5	000	LH	0454	19951117	00000001	00000003	P
6	000	LH	0454	19951117	00000002	00000001	B
7	000	LH	0455	19950606	00000001	00000001	B
8	000	LH	0455	19961231	00089893	00000001	B
9	000	LH	0455	19961231	00089894	00000002	P
10	000	LH	0455	19961231	00089895	00000003	P
11	000	LH	0455	19961231	00089896	00000004	P
12	000	LH	0455	19961231	00089897	00000005	P
13	000	LH	0455	19961231	00089898	00000006	P
14	000	LH	0455	19961231	00089899	00000007	P
15	000	LH	0455	19961231	00089900	00000008	P

In the next step, we modify the table content in the source system. We login to the SAP NetWeaver application server of the source system, create a new customer using the ABAP report *SAPBC_GLOBAL_SCUSTOM_CREATE* and then book a flight for this customer.

Use transaction *se38* to execute this report and fill in some sample data.

After clicking the *Save* icon, a new customer record is created and a confirmation message displays the new customer number, in our case *4688*.



Now we excute ABAP report *SAPBC_GLOBAL_SBOOK_CREATE* to book a flight for our new customer. We enter some flight information, the customer number, a travel agency number and click the *Save* icon. The system message confirms the creation of this booking record with booking number *20070*.

Create SBOOK Records

Airline:

Connection Number:

Flight Date:

Customer Number:

Class:

Smoker:

Luggage Weight:

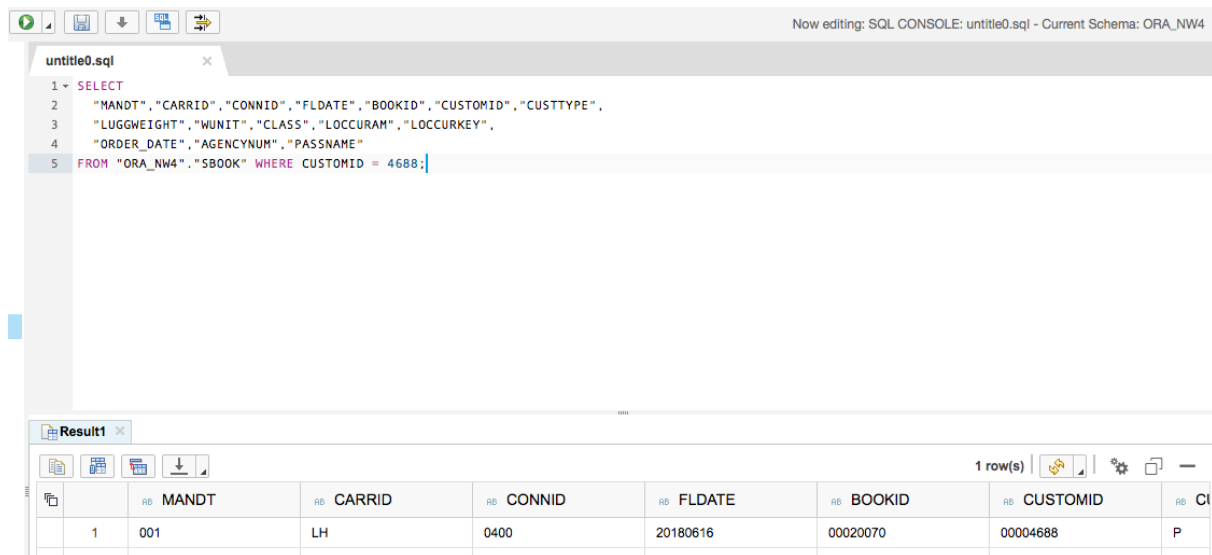
Unit of Measurement:

Travel Agency Number:

✔ Customer Bean is booked on flight LH0400 on 16.06.2018 (booking number 00020070)

Back in the SAP HANA Web IDE, right click on the table name in the sidebar and select the *Generate Select* action. This will generate a select statement to retrieve the first 1000 rows of the *SBOOK* table. We modify the statement to select only a subset of the available columns and add a where clause to retrieve the records for customer number 4688 only. Execute the statement and it retrieves the data of our previously created booking.

```
SELECT
  "MANDT", "CARRID", "CONNID", "FLDATE", "BOOKID", "CUSTOMID", "CUSTTYPE",
  "LUGGWEIGHT", "WUNIT", "CLASS", "LOCCURAM", "LOCCURKEY",
  "ORDER_DATE", "AGENCYNUM", "PASSNAME"
FROM "ORA_NW4"."SBOOK" WHERE CUSTOMID = 4688;
```



The screenshot shows the SQL Console interface with the following query and result:

```
1 SELECT
2   "MANDT", "CARRID", "CONNID", "FLDATE", "BOOKID", "CUSTOMID", "CUSTTYPE",
3   "LUGGWEIGHT", "WUNIT", "CLASS", "LOCCURAM", "LOCCURKEY",
4   "ORDER_DATE", "AGENCYNUM", "PASSNAME"
5 FROM "ORA_NW4"."SBOOK" WHERE CUSTOMID = 4688;
```

RB	MANDT	RB	CARRID	RB	CONNID	RB	FLDATE	RB	BOOKID	RB	CUSTOMID	RB	CI
1	001	LH	0400	20180616	00020070	00004688	P						

You can scroll to the right to see the remaining columns or right-click on a row and select the *Details...* action to display the selected record data in a pop-up window.

Details... ✕

Type to filter...

COLUMN	VALUE
MANDT	001
CARRID	LH
CONNID	0400
FLDATE	20180616
BOOKID	00020070
CUSTOMID	00004688
CUSTTYPE	P
LUGGWEIGHT	33
WUNIT	LB
CLASS	F
LOCCURAM	1338.66
LOCCURKEY	EUR
ORDER_DATE	20180327
AGENCYNUM	00000109
PASSNAME	Bean

Close

Now we go back to our SAP NetWeaver NW4 system and modify the booking by executing ABAP report *SAPBC_GLOBAL_SBOOK_EDIT*.

Specify the airline and booking number on the first screen and click *Change* to get to the *Changing SBOOK records* screen.

Changing SBOOK records: Changing a record

Airline	LH
Connection Number	400
Flight Date	16.06.2018
Booking number	20070
Customer Number	4688
B/P customer	P
Smoker	<input type="checkbox"/>
Luggage Weight	44
Unit of Measurement	LB
Invoice flag	<input type="checkbox"/>
Class	F
Amount (for.currency)	0,00
Payment currency	
Amount (loc.currency)	1.338,66
Airline local currency	EUR
Posting date	27.03.2018
Sales office	0
Travel Agency Number	109
Cancelation flag	<input type="checkbox"/>

Only a limited set of data can be modified on that screen, so we just change the *Luggage Weight* from 33 pounds to 44 pounds and click the *Save* icon.

Back in the SAP HANA Web IDE, we can re-run our query, open the *Details...* window again and we see the newly changed value of 44 pounds in the *Luggage Weight* column.

Details... ✕

COLUMN	VALUE
MANDT	001
CARRID	LH
CONNID	0400
FLDATE	20180616
BOOKID	00020070
CUSTOMID	00004688
CUSTTYPE	P
LUGGWEIGHT	44
WUNIT	LB
CLASS	F
LOCCURAM	1338.66
LOCCURKEY	EUR
ORDER_DATE	20180327
AGENCYNUM	00000109
PASSNAME	Bean

Close

3.1.5 Create a Virtual Table using SQL Statements

You can also execute a SQL statement to create a virtual table. The syntax is:

```
CREATE VIRTUAL TABLE <table_name> <remote_location_clause> <remote_property_clause>

<remote_location_clause> ::=
  AT <source_name>.[<db_name>.<owner>].<identifier>

<source_name> ::= <identifier>
<db_name> ::= <identifier>
<owner> ::= <identifier>
```

(See also [SAP Note 2460723 - How to create Virtual Table based on Remote Source that uses SDI Adapter in SAP HANA via SQL?](#) or the SAP HANA Administration Guide for details)

The following command creates another virtual table called *VT_SBOOK* in our *ORA_NW4* schema.

```
CREATE VIRTUAL TABLE ORA_NW4.VT_SBOOK AT "ORA_NW4".NW4.SAPSR3.SBOOK
```

The screenshot shows the SAP Workbench interface. At the top, it says 'Workbench: Catalog' and 'v 1.132.6 | Help | SYSTEM | H04 | H04 (ph04na1 00) |'. Below that, it says 'Now editing: SQL CONSOLE: untitled0.sql - Current Schema: SYSTEM'. The main area shows a file named 'untitled0.sql' with the following SQL statement: '1 -> CREATE VIRTUAL TABLE ORA_NW4.VT_SBOOK AT "ORA_NW4".NW4.SAPSR3.SBOOK'. At the bottom, a status bar indicates: '12:44:48 PM (SQL Editor) Statement 'CREATE VIRTUAL TABLE ORA_NW4.VT_SBOOK AT "ORA_NW4".NW4.SAPSR3.SBOOK' successfully executed in 3659 ms.'

3.2 Secondary DB Connection – Direct Scenario

3.2.1 Secondary Database Connection – ABAP Coding Options

As a prerequisite, the database system for the secondary connection needs to be supported on the SAP NetWeaver application server (SAP NetWeaver AS ABAP). The shared database library for the SAP kernel and the database client must be available for the used OS/DB platform combination.

Different types of SQL can be used for database access within the ABAP programming language:

- Open SQL
- Native SQL – specific statements between EXEC SQL and ENDEXEC
- Native SQL – ABAP Database Connectivity classes (ADBC)

Open SQL provides a uniform syntax and semantics for all database systems supported for SAP NetWeaver AS ABAP. ABAP programs that use Open SQL statements for database access are portable and will work in any SAP NetWeaver AS ABAP system independent of its primary database system. Open SQL statements require that all database tables accessed in the ABAP code are reflected in the ABAP data dictionary. By default, the Open SQL statements access the primary database of the SAP NetWeaver AS ABAP. The additional *CONNECTION* clause forces the Open SQL statement to be executed against a database connection different to the primary database. The advantage of this approach is simplicity: With minor additions to existing statements the operation is directed to a secondary database. The downside is that the table or view must exist in the ABAP Data Dictionary.

Native SQL allows to use database-specific SQL statements directly in an ABAP program. The database tables need not to be administered in the ABAP dictionary. The database-specific SQL statements are handled by the native SQL interface of the database interface. The database-specific statements are specified statically between the statements *EXEC SQL* and *ENDEXEC*. Typically, compared to the Open SQL option, more code is required and the code is a little less elegant. There are little to no syntax checks on the SQL statements created: errors are not caught until runtime and can lead to short dumps if exceptions are not properly handled. So, extensive testing is essential. Advantage is that database specific features can be used directly, and that the tables need not to be defined within the ABAP data dictionary.

Native SQL ADBC classes provide the benefits of the Native SQL connection via *EXEC SQL* and improve on some of the limitations. The concept of ABAP Database Connectivity is a series of predefined classes (*CL_SQL**) which simplify and abstract the *EXEC SQL* blocks. Most important advantage of ADBC for SAP HANA is that ADBC allows to access non-data dictionary artifacts including SAP HANA stored procedures. The ABAP Managed Database Procedures (AMDP) manage database procedures and database functions implemented in native SQL.

Usage of Open SQL should be the preferred choice in case the table is known in the ABAP data dictionary. Open SQL is optimized for communication with the database: using native SQL or ADBC when this is not required might impact performance because of additional overhead (like connection, constructor calls, statement class, query etc). Definition in the ABAP data dictionary needs to be consistent with the table structure in the secondary database for all the fields. In case that the table is not in the ABAP data dictionary and given the advantages of ADBC over EXEC SQL, the recommendation is to develop custom code for secondary DB connections via Native SQL with the ADBC class based interface.

3.2.2 Prepare Secondary DB Connection on the SAP Application Server

Install Oracle Instantclient

Download the Oracle Instantclient for AIX from the SAP Software Download Center. Then create a new directory for the Oracle client and extract the downloaded SAR archive:

```
mkdir -p /oracle/client/12x
cd /oracle/client/12x
/usr/sap/hostctrl/exe/SAPCAR -xvf /tmp/OCL_AIX_PPC64/OCL12164.SAR
```

Install DBSL Library to SAP Kernel

The database dependent part of the SAP database interface is a library which is linked dynamically to the SAP kernel. This database library contains the Database Shared Library (DBSL) and libraries belonging to the corresponding database manufacturers. A workprocess of the SAP application server can include connections to several different databases based on this architecture. The SAP kernel searches for the libraries in specific directories indicated by environment variable DIR_LIBRARY. For the secondary DB connection, the Oracle libraries need to be added.

The database libraries are available in the SAP Service Marketplace in the SAR archives LIB_DBSL<xxx>.SAR, in the patch directories of the SAP Kernel. Download the SAP archive for the Oracle DBSL Library, and extract it to the SAP Kernel directory:

Logon as the administration user of the AS ABAP <sid>adm and extract the archive:

```
> cdexe
> pwd
/sapmnt/H04/exe/uc/rs6000_64
./SAPCAR -xvf /tmp/lib_dbssl_425-70001630.sar
```

The library *dboraslib.so* gets extracted.

Adapt the environment for user <sid>adm in the files used for sourcing the environment variables. Edit the *.sapenv_<hostname>.csh* file to adapt the LIBPATH and add the NLS_LANG variable.

```
> grep NLS_LANG .sapenv_ph04as2.csh
LIBPATH /oracle/client/12x/instantclient
> grep LIBPATH .sapenv_ph04as2.csh
NLS_LANG AMERICAN_AMERICA.UTF8
```

The Oracle client requires that the I/O Completion Ports are enabled in the AIX Kernel. Logon as user *root* and enable the I/O completion ports:

```
mkdev -l iocp0
```

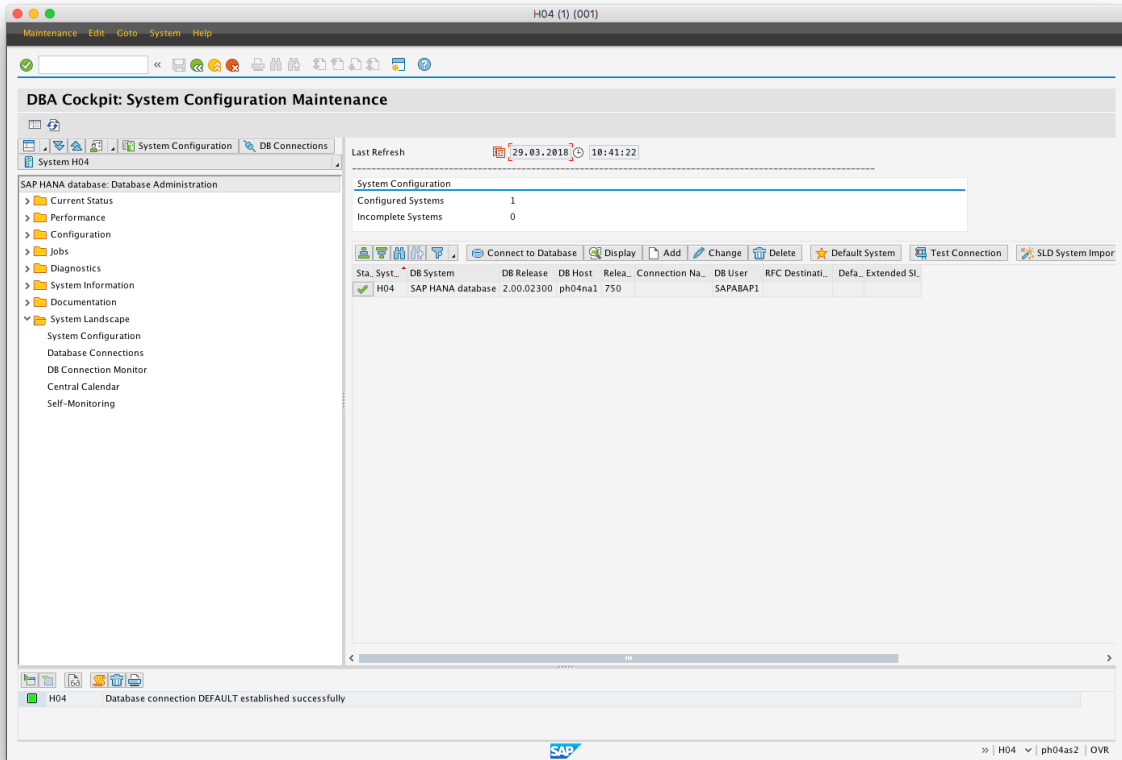
(See also <https://blogs.sap.com/2015/04/10/error-in-oracle-12c-installation-rtld-0712-001-symbol-createiocompletionport-was-referenced/>).

Afterwards, restart the SAP application server.

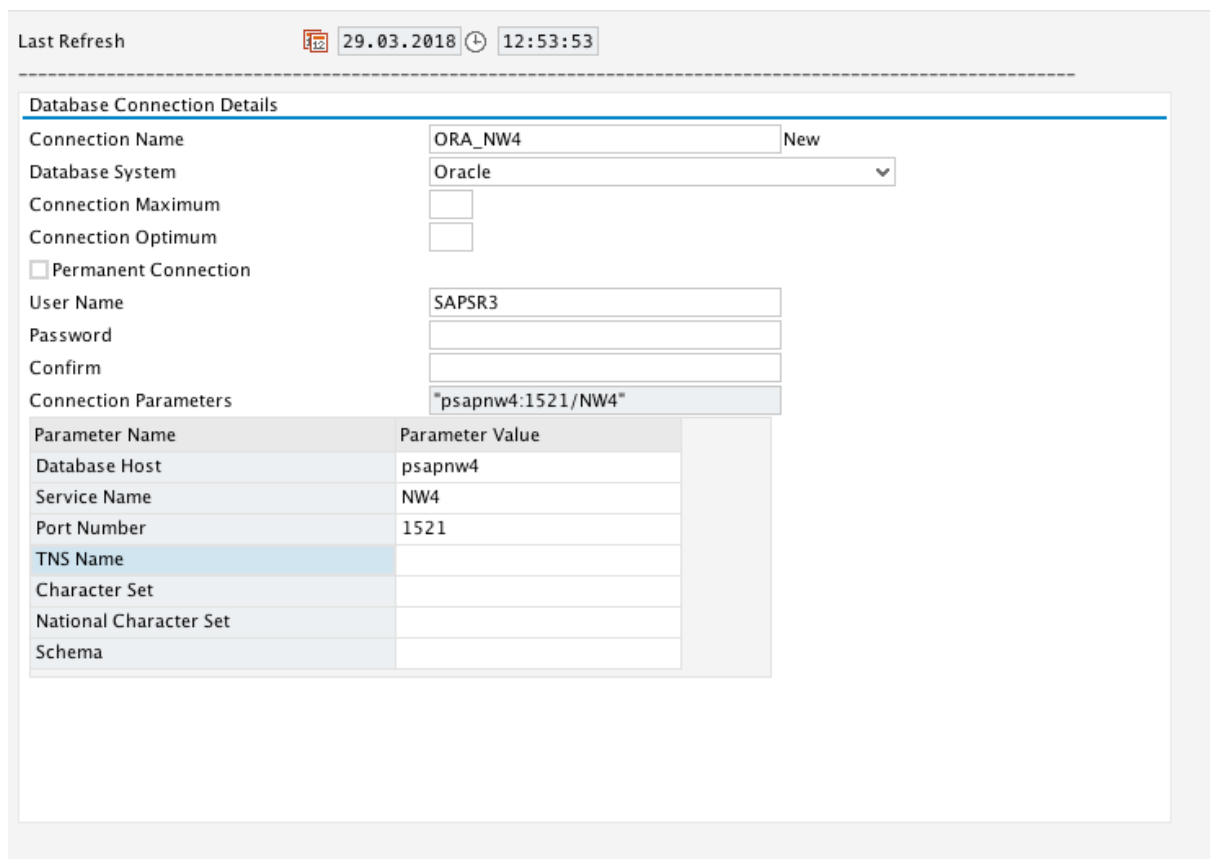
3.2.3 Define Secondary DB Connection in the Database Administrator Cockpit

In this step, we login to the SAP application server running on AIX via SAP Gui and define a secondary database connection to the Oracle database server.

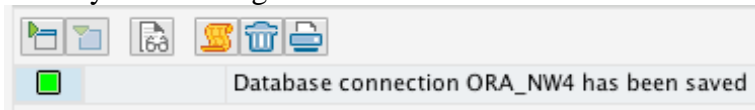
Use transaction *dbacockpit* to open the Database Administrator Cockpit.



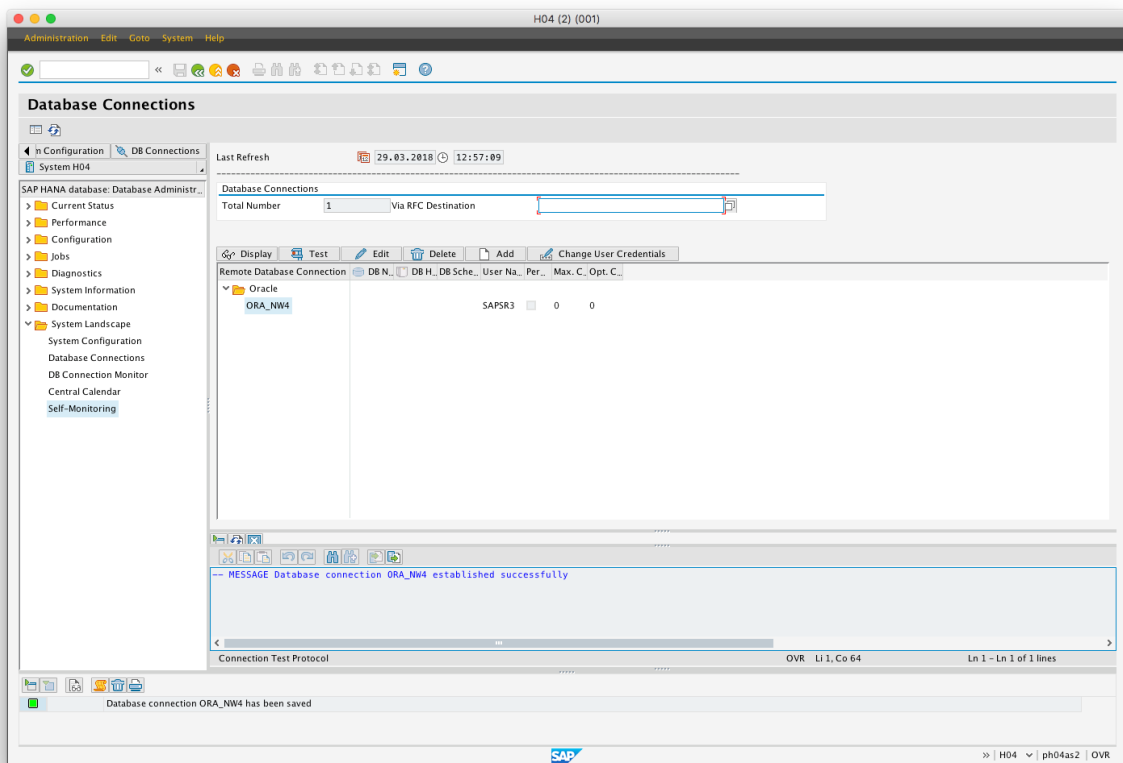
1. Click on the *DB Connection* button and then the *Add* button.
2. Enter the required connection parameters and click the *Save* icon.



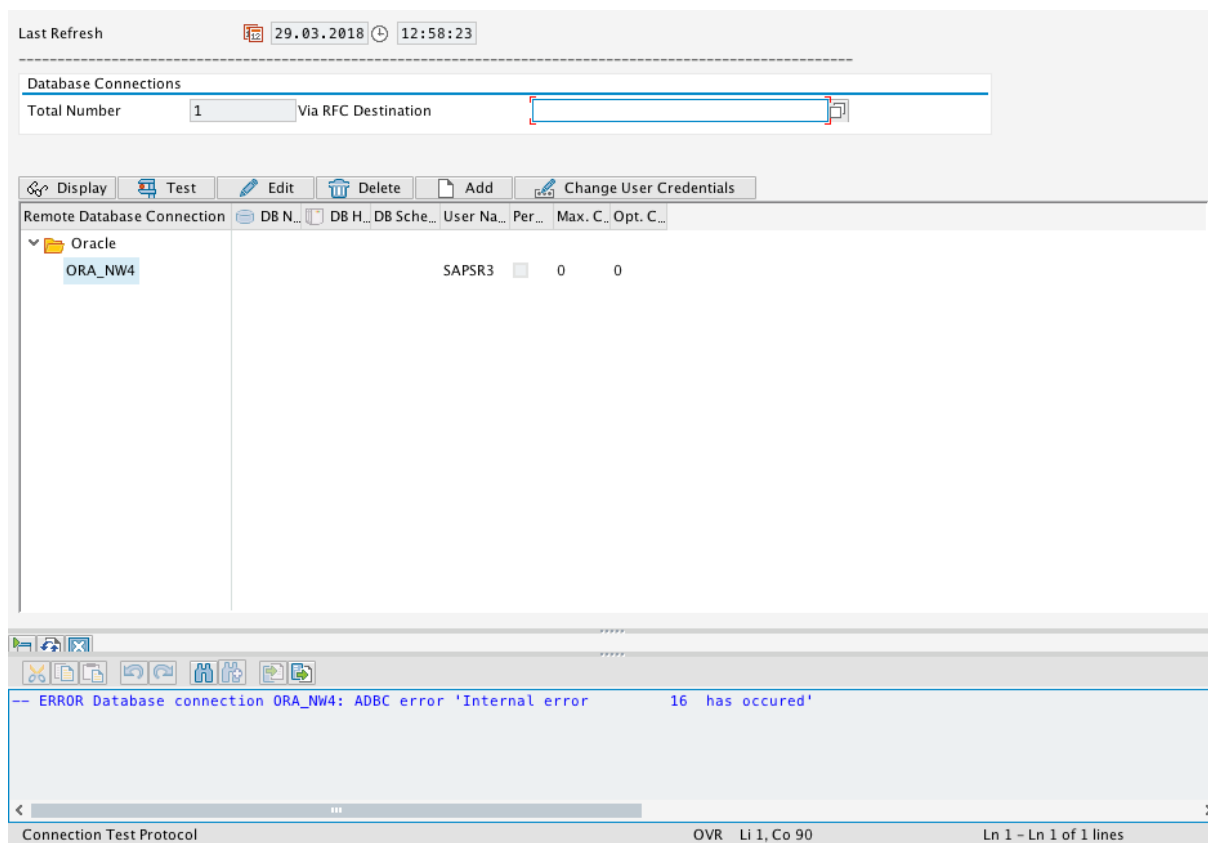
The application server will confirm the creation of the secondary database connection with a short system message.



Back in *ST04 > DB Connections*, we can select the new remote database connection and click the *Test* button, which will display a message that the connection to the remote database system was established successfully.



Please note that if we login to our application server ph04as1 running in a Linux on Power partition, the same test will fail with the following error message:



This is expected as our Linux on Power application server does not have the required DBSL library to connect to an Oracle database.

3.2.4 Access Data in ABAP via Secondary Database Connection

To execute Open SQL statements on databases other than the standard SAP database, you need to add a *CONNECTION* parameter to the SQL statement. Continuing our example from section Create Virtual Table, we create a simple ABAP report using transaction *se38* to access data from the *SBOOK* table in the remote Oracle system.

Please remember the prerequisites explained in chapter 3.2.1, the table structure must exist in the local ABAP dictionary before it can be used in an Open SQL query to a remote system. In our environment, this has been accomplished by running the *SAPBC_DATA_GENERATOR* report to create the sample flight data model in the local system too.

```
REPORT Z_DISPLAY_BOOKING.
TABLES SBOOK.
PARAMETERS id TYPE SBOOK-CUSTOMID.
WRITE: 'Current bookings for customer', id.
SELECT * FROM SBOOK CONNECTION ('ORA_NW4') WHERE CUSTOMID = id.
WRITE: /,
      /5 'MANDT',      25 SBOOK-MANDT,
      /5 'CARRID',    SBOOK-CARRID    UNDER SBOOK-MANDT,
      /5 'CONNID',    SBOOK-CONNID    UNDER SBOOK-MANDT,
      /5 'FLDATE',    SBOOK-FLDATE    UNDER SBOOK-MANDT,
      /5 'BOOKID',    SBOOK-BOOKID    UNDER SBOOK-MANDT,
      /5 'CUSTOMID',  SBOOK-CUSTOMID  UNDER SBOOK-MANDT,
      /5 'CUSTTYPE',  SBOOK-CUSTTYPE  UNDER SBOOK-MANDT,
      /5 'LUGGWEIGHT', SBOOK-LUGGWEIGHT UNDER SBOOK-MANDT LEFT-JUSTIFIED,
      /5 'WUNIT',     SBOOK-WUNIT    UNDER SBOOK-MANDT,
      /5 'CLASS',     SBOOK-CLASS    UNDER SBOOK-MANDT,
      /5 'LOCCURAM',  SBOOK-LOCCURAM  UNDER SBOOK-MANDT LEFT-JUSTIFIED,
      /5 'LOCCURKEY', SBOOK-LOCCURKEY  UNDER SBOOK-MANDT,
```

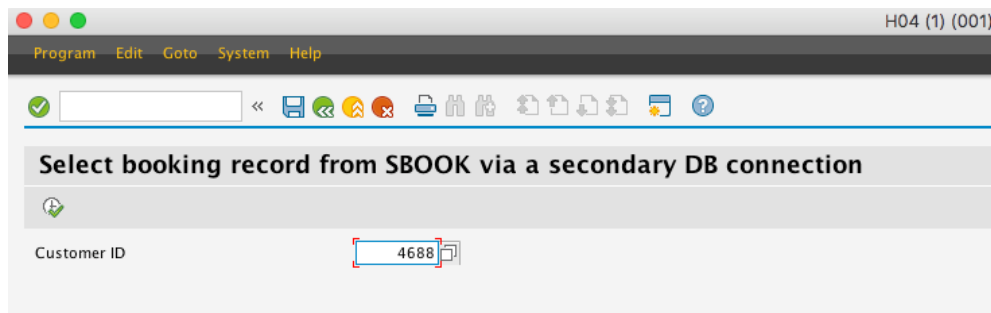
```

/5 'ORDER_DATE', SBOOK-ORDER_DATE UNDER SBOOK-MANDT,
/5 'AGENCYNUM', SBOOK-AGENCYNUM UNDER SBOOK-MANDT,
/5 'PASSNAME', SBOOK-PASSNAME UNDER SBOOK-MANDT.
ENDSELECT.

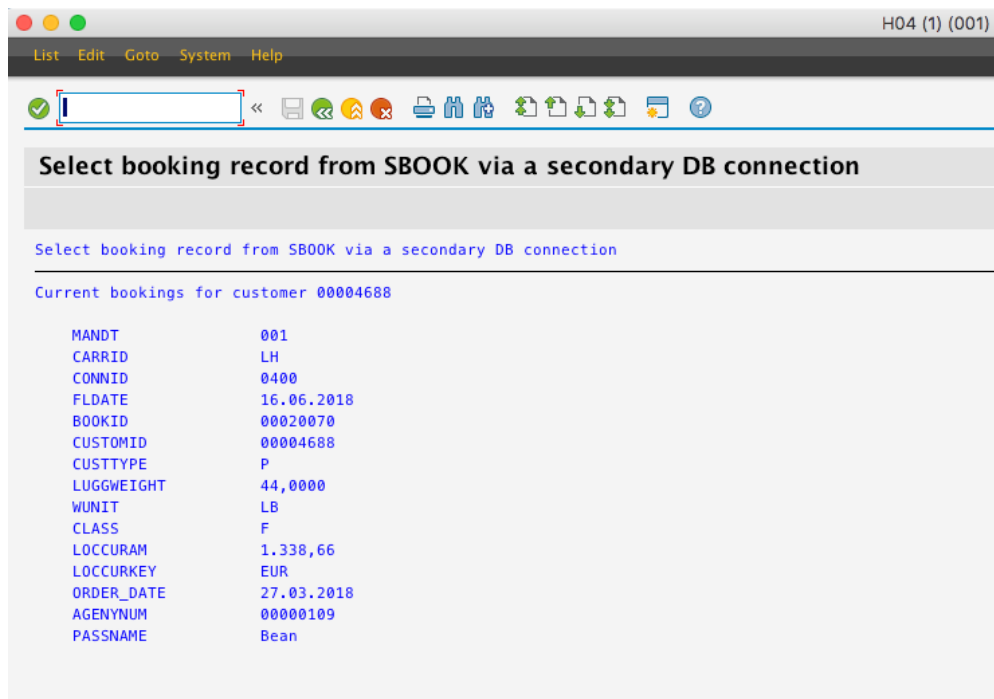
```

Perform a syntax check, generate, and activate the report.

When executing the report, it will first ask for a customer id. We enter our previously created customer id *4688* and click the *Execute* icon.



The report output shows the data that was previously created in the remote Oracle system.



Once again, we can execute this report only on the application server running on our AIX partition. Trying to execute the same report on the Linux application server will result in an ABAP short dump. In the *Description* field of the *Error analysis* section, we see that the NetWeaver kernel tries to load a library `"/usr/sap/H04/D00/exe/dboraslib.so"`, which isn't available on our Linux application server.

Runtime Errors Edit Goto System Help
H04 (1) (001)

Runtime Error – Description of Exception

Long Text Debugger

Category	Installation error
Runtime Errors	DBSQL_LOAD_LIBRARY_ERROR
Date and Time	29.03.2018 13:28:03

Short Text

Database library could not be loaded.

What happened?

While opening a secondary database connection using the logical connection name "ORA_NW4", it was not possible to load a database library. Installation error

The current program had to be terminated because of an error when installing the SAP system.

Error analysis

This is normally due to a database library being missing in the executable directory of the application server required to set up the connection to a remote database. This can be either a database library delivered by SAP (DBSL Shared Library) or a client library delivered by the company whose database is used as the remote database, which must also be on the application server.

Last error logged in SAP kernel

```

Component..... DL (Dynamic Loader)
Location..... SAP-Server ph04as1_H04_00 on host ph04as1 (wp 7)
Version..... 4
Error code..... -2
Error text..... DLoadLib()==DLENOACCESS
Description..... dlopen("/usr/sap/H04/D00/exe/dboraslib.so") FAILED#
"/usr/sap/H04/D00/exe/dboraslib.so: cannot open shared object file: No such
file or directory"
System call.....
Module..... /bas/749_REL/src/krn/dl/dlux.c
Line..... 554
The error reported by the operating system is: Error number.....
                    
```

3.3 Secondary DB Connection – SAP HANA SDI Scenario

We now want to change the secondary database connection of the previous section to use the virtual table defined in the HANA database server instead. This has the advantage that the data can be accessed from any application server. Extra drivers or libraries to access a remote data source aren't needed anymore.

The idea is that in a first step, we just redefine the existing *Remote Database Connection* to point to the HANA database server instead. This may look like a little detour to access the data, but has the advantage that existing ABAP reports can be re-used without any code modifications.

The pictures below illustrate the approach:

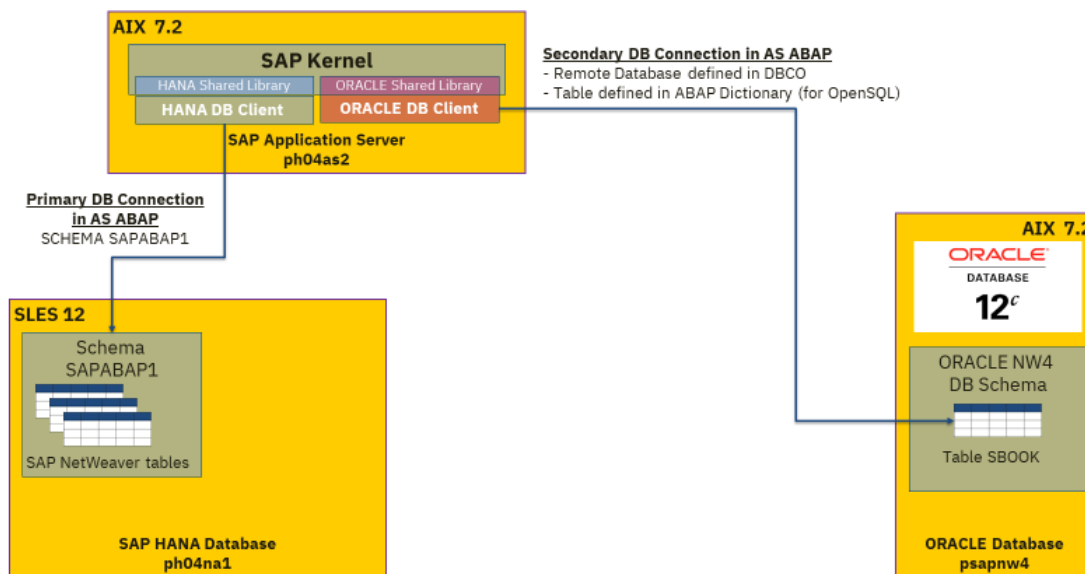
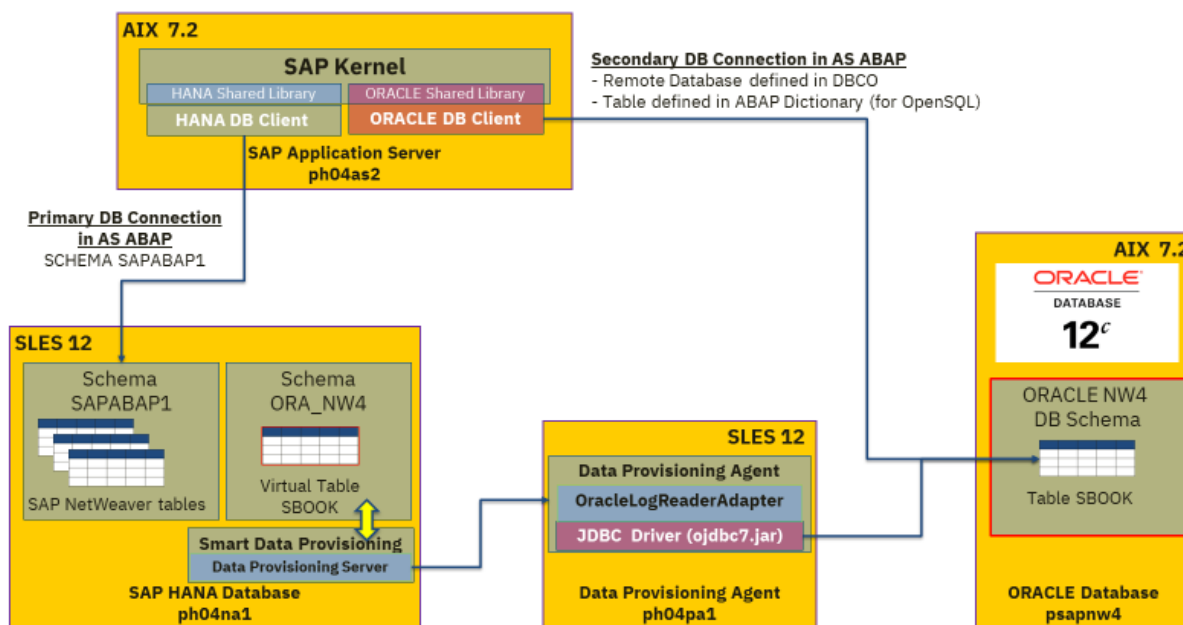
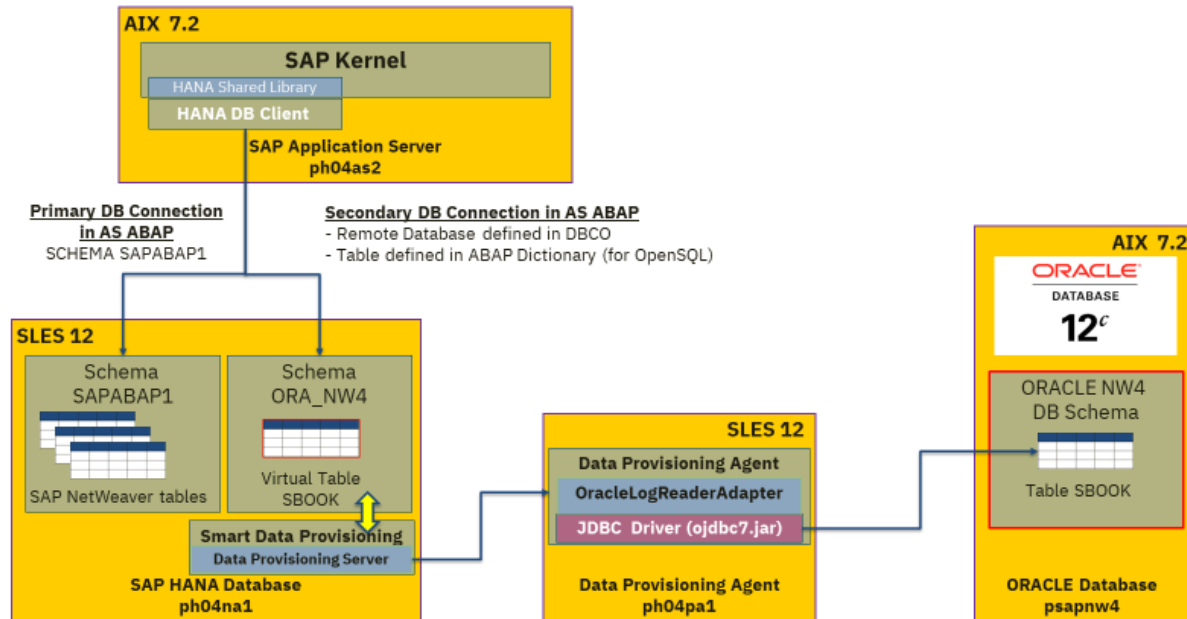


Table *SBOOK* of the remote Oracle database is accessed in the SAP AS ABAP via the secondary database connection *ORA_NW4*. As the Oracle DBSL and the Oracle database client is not available on Linux on Power, access works only on the SAP application server running on AIX.

In the previous section, we've created the virtual table *SBOOK* in a separate schema – the virtual table *SBOOK* in schema *ORA_NW4* corresponds to table *SBOOK* in schema *SAPSR3* on the remote Oracle database. Data access is realized with SAP HANA Smart Data Integration via the Data Provisioning Agent.

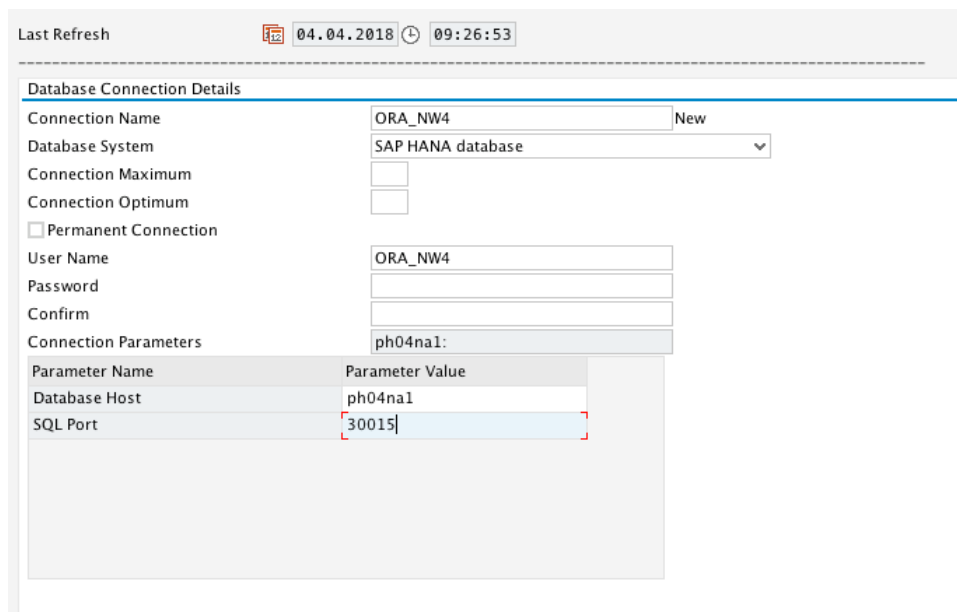


The goal is to redefine the secondary database connection to use the new schema in the local SAP HANA database:



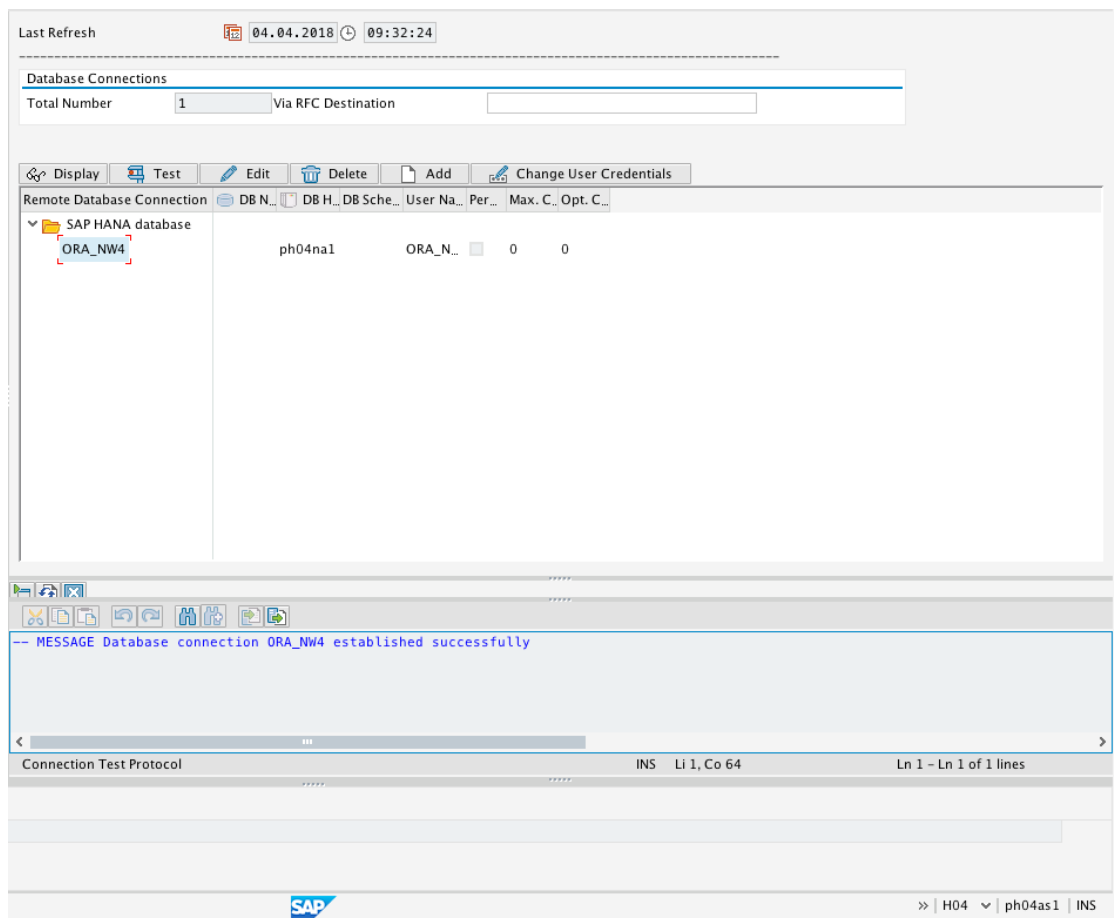
Here are the steps to achieve this goal:

1. Login to an application server and execute transaction *dbacockpit* to open the *Database Administrator Cockpit*.
2. Click the *DB Connections* button to open the *Database Connections* panel.
3. Select the *ORA_NW4* entry and use the *Delete* button to remove the connection. Unfortunately, a connection cannot be renamed. As we want to re-use the connection name, we must delete the existing database connection first.
4. Click on the *Add* button and fill in the connection parameters for the HANA database server.
The *Connection Name* should be *ORA_NW4* again. We use the same user as in section SAP SDI Virtual Table Scenario.



5. Click the *Save* icon to create the new secondary connection to the HANA database server.

6. Click the *Test* button to try the new database connection.



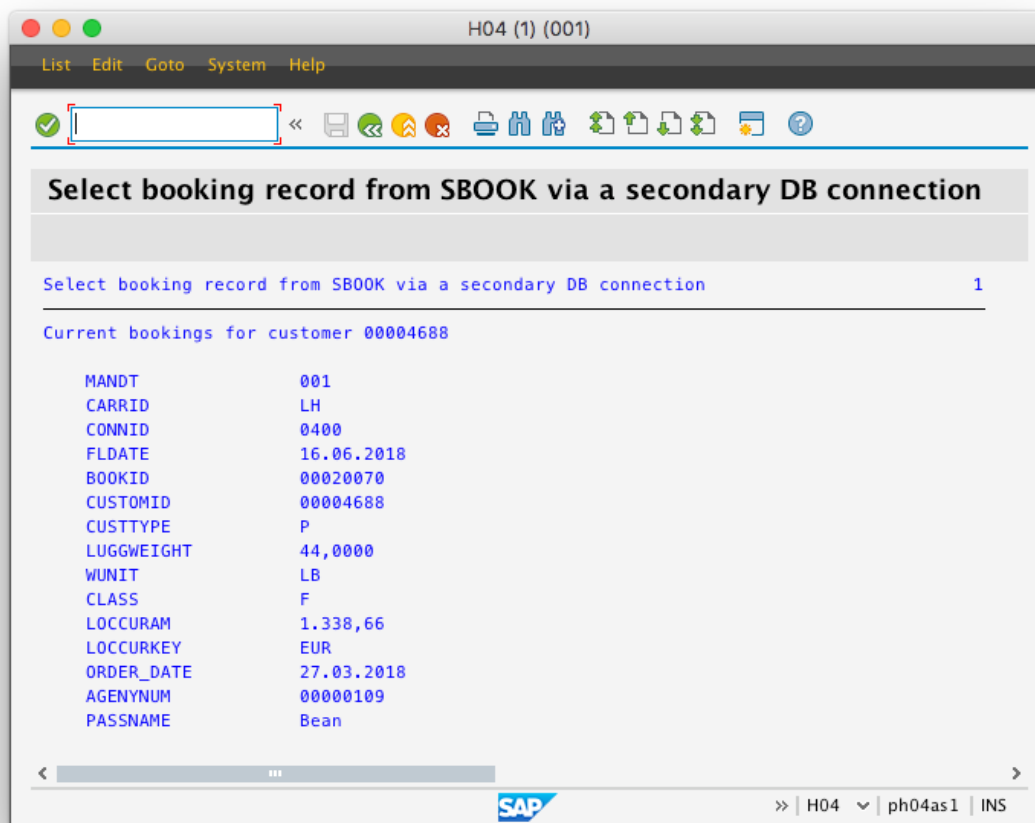
Now we can go back to transaction *se38* and execute our existing *Z_DISPLAY_BOOKING* report again.

As a reminder, this is the select statement from the report:

```
SELECT * FROM SBOOK CONNECTION ('ORA_NW4') WHERE CUSTOMID = id.
```

We've kept the same name (*SBOOK*) for the virtual table name in our *ORA_NW4* schema and re-used the secondary database connection name, so the above statement can run without any modifications. We execute the report, fill in our customer id *4688*, and get the same output as before.

Please note in the lower right corner of the screenshot below that we've executed the report on host *ph04as1*. Changing the secondary database connection to access a virtual table defined in our HANA database servers now allows us to execute the unmodified report on any connected application server.



A big advantage of this solution is that we don't need a second DBSL library and database client on the application server anymore. This simplifies the maintenance procedure for AS ABAP kernel updates. Even though we are using a *secondary database connection*, all data accesses (whether to the local database or a remote data source) will use the standard HANA DBSL layer.

In the previous scenario of using an Oracle client and DBSL, you must check for each SAP kernel update that the new version is still compatible with the installed database client. In most cases, this may result in extra work to upgrade the Oracle DBSL library and sometimes you would need to update the Oracle database client too.

3.4 Secondary DB Connection – SAP HANA SDI Native SQL Scenario

As described in chapter 3.2.1, the disadvantage of using Open SQL for accessing tables in a remote database is that the table name needs to be defined in the local ABAP dictionary. Native SQL allows us to directly access tables that aren't known in the local dictionary. The following report is a re-write of the *Z_DISPLAY_BOOKING* report using Native SQL. As a simplification, we are still referring to the local *SBOOK* dictionary definition by declaring the *id* parameter as type *SBOOK-CUSTOMID* and the local work-area *ls_sbook* as type *SBOOK*. This could be easily changed by adding a complete structure definition for *ls_sbook* reflecting the source table structure and changing *id* to a numeric type.

Please note that the select statement was changed to *select * from ora_nw4_sbook*. We've just created a second virtual table (connecting to the same source table) in the *ORA_NW4* schema using a table name that isn't known in the local ABAP dictionary.

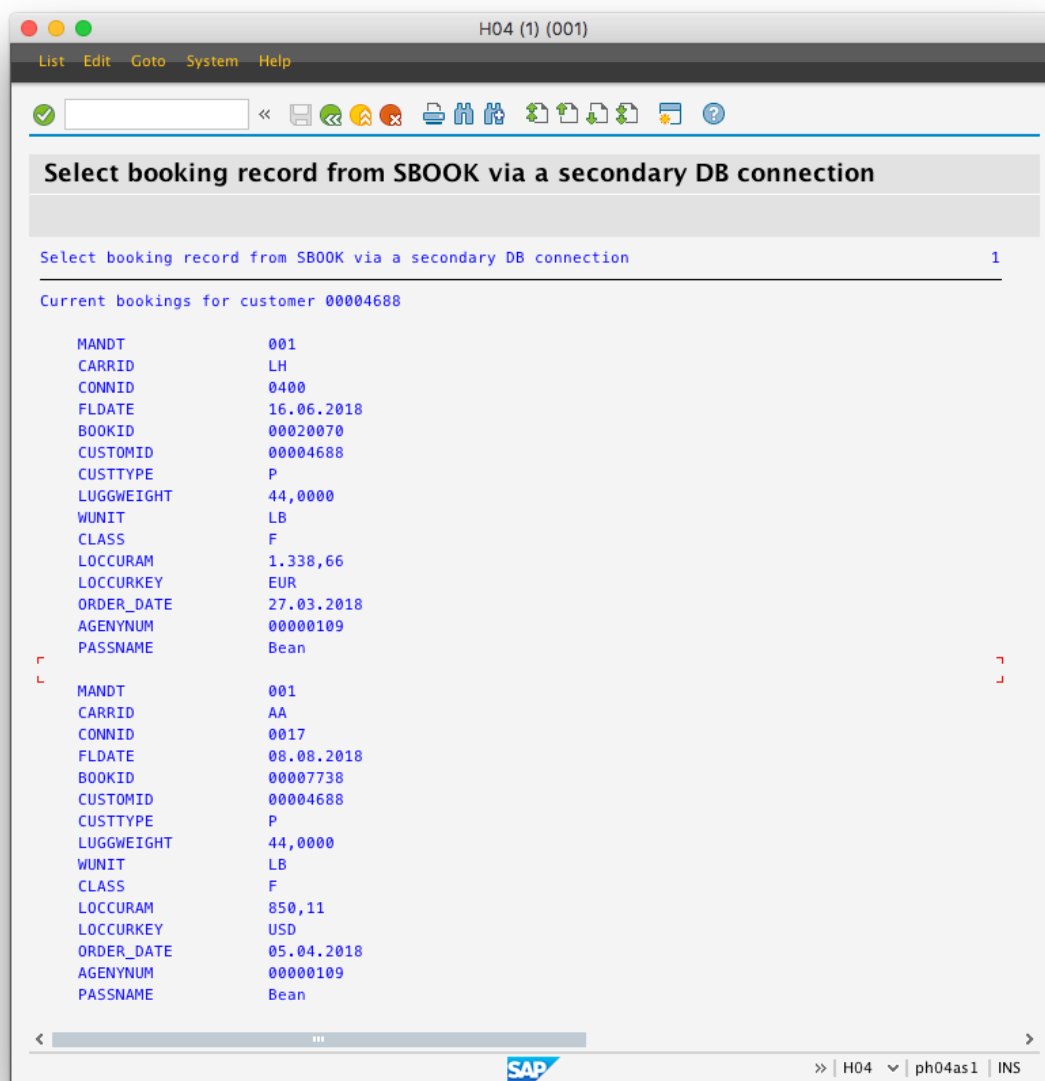
```

REPORT Z_DISPLAY_BOOKING_NSQL.
PARAMETERS id TYPE SBOOK-CUSTOMID.
DATA ls_sbook TYPE SBOOK.
WRITE: 'Current bookings for customer', id.
EXEC SQL.
    connect to 'ORA_NW4' as 'ORA_NW4'
ENDEXEC.
EXEC SQL.
    open dbcur for select * from ora_nw4_sbook where customid = :id
ENDEXEC.
DO.
    EXEC SQL.
        fetch next dbcur into :ls_sbook
    ENDEXEC.
    IF sy-subrc NE 0.
        EXIT.
    ELSE.
        WRITE: /,
            /5 'MANDT',      25 ls_sbook-MANDT,
            /5 'CARRID',    ls_sbook-CARRID      UNDER ls_sbook-MANDT,
            /5 'CONNID',    ls_sbook-CONNID      UNDER ls_sbook-MANDT,
            /5 'FLDATE',    ls_sbook-FLDATE      UNDER ls_sbook-MANDT,
            /5 'BOOKID',    ls_sbook-BOOKID      UNDER ls_sbook-MANDT,
            /5 'CUSTOMID',  ls_sbook-CUSTOMID    UNDER ls_sbook-MANDT,
            /5 'CUSTTYPE',  ls_sbook-CUSTTYPE    UNDER ls_sbook-MANDT,
            /5 'LUGGWEIGHT', ls_sbook-LUGGWEIGHT UNDER ls_sbook-MANDT LEFT-JUSTIFIED,
            /5 'WUNIT',     ls_sbook-WUNIT      UNDER ls_sbook-MANDT,
            /5 'CLASS',     ls_sbook-CLASS      UNDER ls_sbook-MANDT,
            /5 'LOCCURAM',  ls_sbook-LOCCURAM   UNDER ls_sbook-MANDT LEFT-JUSTIFIED,
            /5 'LOCCURKEY', ls_sbook-LOCCURKEY  UNDER ls_sbook-MANDT,
            /5 'ORDER_DATE', ls_sbook-ORDER_DATE UNDER ls_sbook-MANDT,
            /5 'AGENCYNUM', ls_sbook-AGENCYNUM  UNDER ls_sbook-MANDT,
            /5 'PASSNAME',  ls_sbook-PASSNAME   UNDER ls_sbook-MANDT.
        ENDIF.
    ENDDO.
EXEC SQL.
    close dbcur
ENDEXEC.
EXEC SQL.
    disconnect 'ORA_NW4'
ENDEXEC.

```

The output of this report is basically the same as before. Although, in the meantime we've booked an additional flight for our customer.

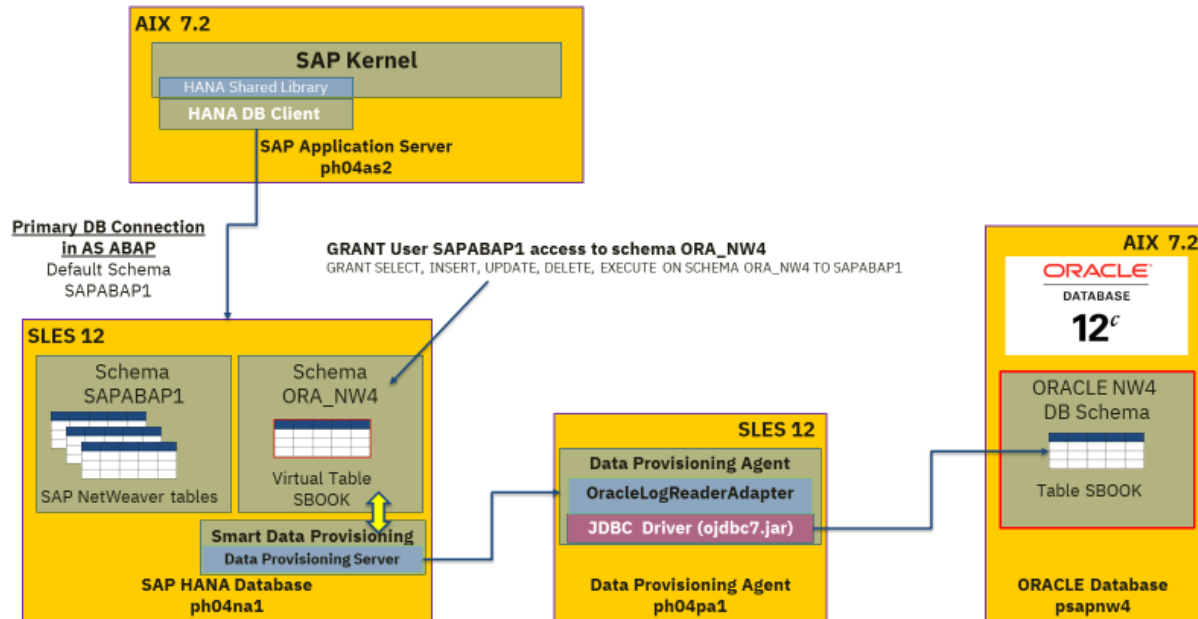
Just the same as with the *Z_DISPLAY_BOOKING* version of the report, we can still use the same report either connecting to the source Oracle database or connecting via the HANA SDI virtual table by just changing the secondary DB connection entry in the Database Administrator Cockpit.



3.5 Remote Database Access – SAP HANA SDI Native SQL Scenario

The idea so far has been to re-use existing ABAP reports with little or no changes. In case the existing codes is already using Native SQL to access data on a remote system via a secondary database connection, it can be modified rather easily to forego the secondary database connection completely.

We can just drop the first and the last EXEC SQL block to eliminate the *connect* and *disconnect* statements. As the ABAP report is running in the ABAP schema (in our case *SAPABAPI*) and wants to access data in another schema, we must add the schema name in the select statement: `select * from ora_nw4.ora_nw4_sbook ...`



The last step is to grant the *SAPABAP1* user the privileges to access data in the *ORA_NW4* schema, so we login at the SAP HANA Web IDE with the *ORA_NW4* userid and execute the following SQL statement:

```
GRANT SELECT ON SCHEMA ORA_NW4 TO SAPABAP1
```

The modified ABAP report looks like:

```
REPORT Z_DISPLAY_BOOKING VT.
PARAMETERS id TYPE SBOOK-CUSTOMID.
DATA ls_sbook TYPE SBOOK.
WRITE: 'Current bookings for customer', id.
EXEC SQL.
  open dbcur for select * from ora_nw4.ora_nw4_sbook where customid = :id
ENDEXEC.
DO.
  EXEC SQL.
    fetch next dbcur into :ls_sbook
  ENDEXEC.
  IF sy-subrc NE 0.
    EXIT.
  ELSE.
    WRITE: /,
      /5 'MANDT',    25 ls_sbook-MANDT,
      /5 'CARRID',   ls_sbook-CARRID    UNDER ls_sbook-MANDT,
      /5 'CONNID',   ls_sbook-CONNID    UNDER ls_sbook-MANDT,
      /5 'FLDATE',   ls_sbook-FLDATE    UNDER ls_sbook-MANDT,
      /5 'BOOKID',   ls_sbook-BOOKID    UNDER ls_sbook-MANDT,
      /5 'CUSTOMID', ls_sbook-CUSTOMID  UNDER ls_sbook-MANDT,
      /5 'CUSTTYPE', ls_sbook-CUSTTYPE  UNDER ls_sbook-MANDT,
      /5 'LUGGWEIGHT', ls_sbook-LUGGWEIGHT UNDER ls_sbook-MANDT LEFT-JUSTIFIED,
      /5 'WUNIT',     ls_sbook-WUNIT     UNDER ls_sbook-MANDT,
      /5 'CLASS',     ls_sbook-CLASS     UNDER ls_sbook-MANDT,
      /5 'LOCCURAM',  ls_sbook-LOCCURAM  UNDER ls_sbook-MANDT LEFT-JUSTIFIED,
      /5 'LOCCURKEY', ls_sbook-LOCCURKEY UNDER ls_sbook-MANDT,
      /5 'ORDER_DATE', ls_sbook-ORDER_DATE UNDER ls_sbook-MANDT,
      /5 'AGENCYNUM', ls_sbook-AGENCYNUM UNDER ls_sbook-MANDT,
      /5 'PASSNAME',  ls_sbook-PASSNAME  UNDER ls_sbook-MANDT.
  ENDIF.
ENDDO.
```

```
EXEC SQL.
  close dbcur
ENDEXEC.
```

Executing this report generates once more the exact same output as the previous versions (except that we changed the title to reflect that we don't use a secondary database connection anymore).

After this modification, we can also drop the secondary database connection in the Database Administrator Cockpit.



4 Summary

Before the release of HANA 1.0 SPS 09, SAP recommended to use several different tools to get data from remote data sources into SAP HANA. Depending on the scenario, you would use SAP Data Services (SDS), System Landscape Transformation (SLT), Smart Data Access (SDA), SAP Replication Server (SRS), etc.

With the availability of SAP HANA Smart Data Integration, SAP has substantially simplified the topic of data loads in SAP HANA. The data provisioning tier is now build natively within HANA and SDI comes with an open framework and SDK. This enables the build of custom adapters allowing for the integration of any data source above and beyond the large variety of data sources that are already supported by adapters delivered with the Data Provisioning Agent.

In this whitepaper, we presented a few simple examples for migrating existing remote data source scenarios to an SDI based solution. Compared to a secondary database connect solution, SDI eliminates the need to install and maintain additional database client software and SAP DBSL libraries on the SAP NetWeaver application server. The advantage over a Smart Data Access (SDA) scenario is that SDI supports more remote data sources and again eliminates the need to install and maintain external ODBC drivers on the HANA server instance.

Further advantages of SDI compared to SDA are:

- Decoupling the remote sources from the SAP HANA indexserver and managing them via data provisioning server provides additional stability to the SAP HANA system.
- Proxy scenarios support additional connectivity options to remote sources in cases where network connectivity between remote source and SAP HANA system is restricted.

These advantages and all the additional functionalities of SDI beyond the data federation scenarios position SDI as the strategic solution for data exchange with SAP HANA.

COPYRIGHT LICENSE

If this information contains sample application programs in source language, which illustrate programming techniques on various operating platforms, you may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The information contained in this document has not been submitted to any formal IBM test and is provided "AS IS" with no warranties or guarantees either expressed or implied.

All examples cited or described in this document are presented as illustrations of the manner in which some IBM products can be used and the results that may be achieved. Actual environmental costs and performance characteristics will vary depending on individual client configurations and conditions.

IBM is not responsible for printing errors in this document that result in pricing or information inaccuracies.

A full list of U.S. trademarks owned by IBM may be found at:

<http://www.ibm.com/legal/copytrade.shtml>

A full list of trademarks owned by SAP may be found at:

<http://www.sap.com/company/legal/copyright/trademark.epx>

SAP and other SAP products and services mentioned herein, as well as their respective logos, are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world.

UNIX is a registered trademark in the United States, other countries or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries or both.

Microsoft, Windows, Windows NT and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries or both.