SAPinsider

This article appeared in the Oct ▪ Nov ▪ Dec 2016 issue of *SAPinsider* (www.SAPinsiderOnline.com) and appears here with permission from the publisher, WIS Publishing.

WIS pubs

Karl Kessler
(karl.kessler@sap.com) joined SAP SE in 1992. He is the Product Manager of the SAP NetWeaver foundation — which includes SAP NetWeaver Application Server, the ABAP Workbench, and the Eclipse-based ABAP development tools for SAP NetWeaver — and is responsible for all rollout activities.

Introducing ABAP 7.51

A New ABAP Model for Transactional Application Development

Since its release in October 2015, SAP NetWeaver 7.5 has seen a rapid adoption rate, with more than 1,000 productive customer systems as of September 2016. SAP NetWeaver 7.5 delivers an array of innovation for the ABAP stack, including advanced support for core data services (CDS) and features for the corresponding ABAP development tools for Eclipse (known as ABAP in Eclipse), which together enable rapid development of SAP Fiori-based analytical and reporting applications. Support for Java 8 and end of maintenance for SAP NetWeaver 7.0 in 2017 have also been significant drivers of adoption.[1]

With all of the innovation in 7.5, and its eager adoption by customers, it might be surprising to learn that SAP is already delivering version 7.51 in Q4 2016. Why is SAP doing this, and why should you upgrade to this new version? The main driver for 7.51 is to provide ABAP enhancements that support the transition from SAP Business Suite to SAP S/4HANA, and for this reason, it is not a full SAP NetWeaver delivery — it consists solely of SAP NetWeaver Application Server (SAP NetWeaver AS) ABAP, supported by the 7.49 ABAP kernel, and related development tools. So, does that mean this a release only for SAP S/4HANA customers?

The answer is no. ABAP 7.51 delivers an enhanced ABAP programming model that extends CDS to include transactional as well as analytical application development and supports custom code management. Not only do these enhancements provide a structured approach for transitioning from a traditional SAP Business Suite implementation to SAP S/4HANA along with powerful custom development capabilities, they also extend the advantages of CDS modeling to the development of any custom ABAP-based transactional applications.

This article provides an overview of the ABAP features included in an upgrade to 7.51 (see the sidebar "Upgrading to 7.51" for more on upgrading), and walks through an example that demonstrates how these features work together to help you create sophisticated SAP Fiori-based transactional applications.

The 7.51 ABAP Programming Model

CDS is a layer for defining and consuming data models on the ABAP abstraction level on top of the SAP HANA database. Introduced with SAP NetWeaver 7.4, support package stack 05, the CDS model is represented in ABAP as CDS views. These views are defined using a SQL-based data definition language (DDL) and can be exposed as OData services, without the need to write SAP Gateway code, to provide SAPUI5-based SAP Fiori applications with easy access to the data represented in the CDS model. The CDS paradigm for ABAP development has significantly accelerated the development of SAP Fiori applications, enabling the semantic modeling of data and making it easy to write analytical and reporting applications.

But what about typical transactional applications that perform insert, update, and delete operations? Technically, you can use Open SQL for transactional applications on the 7.4 or 7.5 ABAP

[1] For more on SAP NetWeaver 7.5, see my *SAPinsider* articles "A Foundation for the Future: What's Coming Next with SAP NetWeaver 7.5" (July-September 2015) and "ABAP for the Modern Age: A Look at the Latest ABAP Enhancements in SAP NetWeaver 7.5" (January-March 2016), available at SAPinsiderOnline.com.

stack. You can extend SAP Fiori applications consuming CDS with manually coded SAPUI5 applications that access manually coded SAP Gateway services, which in turn call manually coded ABAP methods that execute the Open SQL statements accordingly. For each of these tasks, there is a corresponding development tool: SAP Web IDE for SAPUI5, transaction SEGW for SAP Gateway projects, and ABAP in Eclipse for implementing the ABAP methods that carry out the Open SQL commands.

While technically possible, this manual approach to the CDS-based development of transactional applications is time consuming and does not leverage the modeling abstraction that is essential for CDS development. To bring the advantages of CDS modeling and SAP Fiori-based user interfaces to transactional scenarios, ABAP 7.51 extends the CDS-based ABAP program model to integrate transactional services on the CDS level. The natural way to do this is to provide meaningful annotations that bridge the analytical world of CDS and the transactional Business Object Processing Framework (BOPF) model.[2]

[2] Learn more about BOPF on SAP Community Network at http://scn.sap.com/community/abap/bopf.

Upgrading to 7.51

So, will only SAP S/4HANA customers get the 7.51 version as an indirect shipment? Fortunately, the answer is no. SAP is also releasing SAP NetWeaver Application Server (SAP NetWeaver AS) ABAP 7.51 as a standalone development and runtime environment, meaning that any SAP customer with a valid SAP NetWeaver license can download it from SAP Service Marketplace (**http://service.sap.com**) or run a cloud image via the SAP Cloud Appliance Library.

Upgrade options for ABAP 7.51 are offered for SAP NetWeaver start releases 7.4 and 7.5 when the usage type is restricted to ABAP development and execution. Java usage types can skip the 7.51 release and later upgrade to a successor version (not yet planned) offering Java 9 support. Similarly, SAP Business Suite (enhancement package 8) usage types, which run on the full SAP NetWeaver 7.5 release, can skip 7.51 and upgrade to a (not yet planned) later release.

In parallel to the 7.51 shipment, SAP has released a couple of add-ons for 7.4 and 7.5, including a governance, risk, and compliance (GRC) add-on (SAP Note 2323497 documents the available add-ons). These add-ons require only an SAP NetWeaver AS ABAP release, underscoring the value of 7.51. Other add-ons that require a full SAP NetWeaver stack, similar to the SAP Business Suite usage type, can skip 7.51 and wait for a later release.

CDS Modeling Support for Transactional Applications

BOPF is a model-driven persistence framework for custom ABAP development. It provides an application framework of generic services and functionalities that developers can then customize to their particular needs, enabling transaction-oriented development based on a stable application architecture. The BOPF model is not new. It resides in the ABAP stack as an ABAP-based repository and runtime framework and is used by several SAP Business Suite applications, including newer solutions such as SAP Transportation Management.

A BOPF model consists of a root node and child nodes that represent a business object and its related fields. For example, a sales order would consist of a root node that holds the header information and sales order items as descendent nodes with 0 to n cardinality. Each BOPF model exposes a certain behavior based on different methods, including validations to check for the consistency of a business object, determinations to change the state and the attributes of a business object, and actions that carry out operations on a business object. Technically, these methods are implemented by ABAP classes, so you can think of the BOPF model as an object-based framework. Central services such as transaction control, commit to the database, locking, and buffering are handled inside the BOPF framework, which shields the details of the implementation from the developer.

In ABAP 7.51, the semantic link between the CDS definition and its corresponding BOPF representations is established by a set of well-defined annotations that are added using the DDL editor — the tool within ABAP in Eclipse used for defining CDS views — similar to the way annotations are created to describe the use of data in a user interface, for instance. Since annotations define a certain facet of the underlying view data for user interface, analytical, or search purposes, you can think of BOPF as the transactional facet of a CDS definition.

When a CDS view is activated, the corresponding underlying view definitions on the SAP HANA database are activated. When the CDS contains BOPF annotations, a corresponding BOPF model and definitions are automatically activated in the background, as if you had used the BOPF editor tool included in ABAP in Eclipse to build it. In fact,

you can use the BOPF editor to display the results of the activation and then dive into the ABAP-based implementation of the BOPF methods.

Let's look at a use case that demonstrates how the different bits and pieces — CDS, BOPF, and SAP Fiori — all work together in ABAP 7.51 to bring CDS features and an elegant and responsive user interface to transactional application development.

Putting It All Together: An Example

CDS and BOPF development can be easily combined in the user interface layer, where the analytical and transactional parts of an application appear side by side in many scenarios. Here, using the well-known Enterprise Procurement Model, we'll walk through the development of an example application that combines these two types of development in the user interface.

Developing the CDS View

Our first goal is to develop a sales order invoice CDS view containing a list of invoice numbers together with the usual fields, such as customer ID, company name, timestamp of creation, transaction currency, and gross amount of the invoice. The development of CDS views has been covered in depth in previous articles, so here we will take a high-level look at the steps.[3]

> Additional screenshots that show the details of the steps outlined here are included in the online version of this article at SAPinsiderOnline.com.

Figure 1 shows the source code for the CDS view (ZDEV210_C_SLSORDINV_SOL) that generates this list, displayed in the DDL editor of ABAP in Eclipse. This view is a simple projection view that is mapped to the base view SEPM_I_CustomerInvoice_E, which has an association to a customer view to retrieve the company name. You can display all of the associations and related information for the base view by pressing F2. Executing the view (by pressing F8) displays a list of the specified fields and retrieved data in a table format.

CDS offers a variety of built-in functions for customizing the resulting data set, including calculation

[3] For more on CDS development, see my *SAPinsider* articles "Enhanced ABAP Development with Core Data Services (CDS)" (October-December 2015) and "ABAP for the Modern Age: A Look at the Latest ABAP Enhancements in SAP NetWeaver 7.5" (January-March 2016), available at SAPinsiderOnline.com.

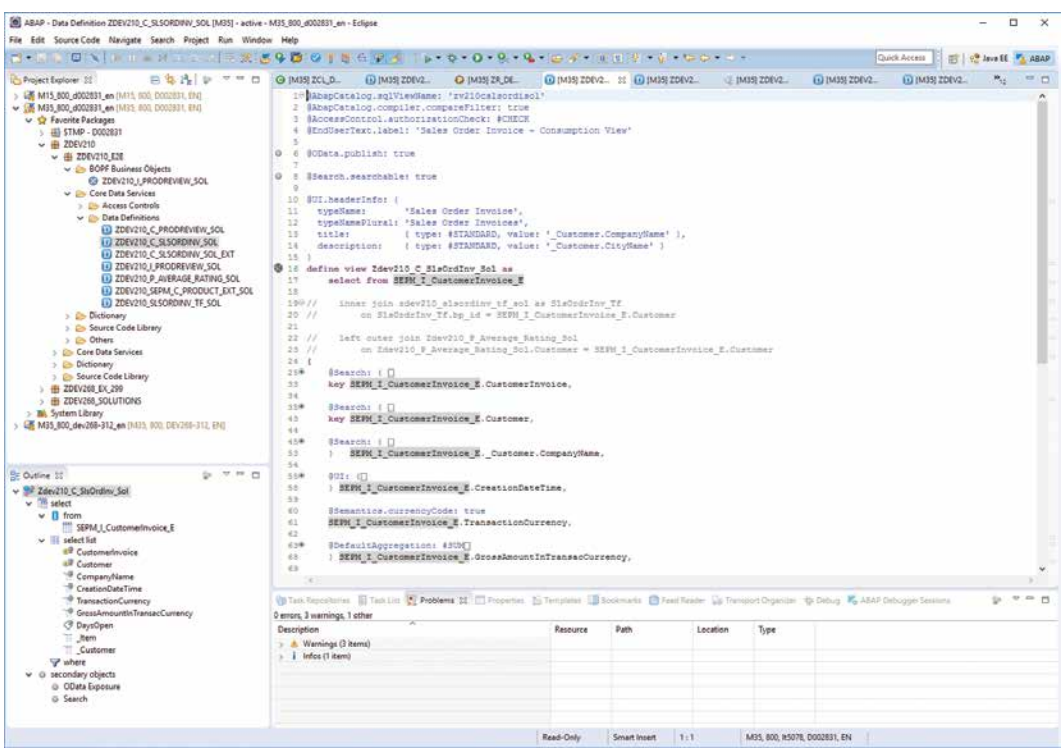

Figure 1 The source code for the example sales order invoice CDS view

functions and extensibility options for adding fields. For example, you can use the CreationDateTime data to determine how many days a particular invoice has been open — by calculating the difference between the creation and the actual date and time in seconds (function tstmp_seconds_between) and dividing by 86400 (the product of 60*60*24) — and add a field to display the results.

Creating an Application Based on the CDS View

Once the CDS view is complete, we are ready to create an SAP Fiori application based on the CDS view. For this task, we use SAP Web IDE, SAP's web-based environment for SAPUI5 application development, which is available via SAP HANA Cloud Platform or as a standalone on-premise version. As with CDS view development, SAP Fiori application development has been covered in detail in previous articles, so here we will focus on the key steps.[4]

SAP Web IDE offers a smart template approach to developing SAP Fiori applications. After selecting Smart Template Application on the Template Selection screen, specify the connection to the back-end ABAP system (in the example, M35) and select the SAP Gateway service that was automatically published with our sales order invoice CDS view (ZDEV210_C_SLSORDINV_SOL_CDS in the example).

We then generate the SAP Fiori development project and run it. The result is shown in **Figure 2**. It shows sales order invoices grouped by company, and shows all the aggregated columns such as average open days, gross amount, converted amount, and creation date. All the grouping and sorting capabilities are built into the template automatically, meaning no need to write any code for it.

To view the details of a particular line item, select the item and click on Show Details, which takes you to a display that groups the details into general information on the upper half and all the positions of the sales order on the lower half. Again, all this functionality is automatically provided by the template and requires no additional manual coding efforts.

Now, let's see how the transactional services capabilities provided with ABAP 7.51 come into play.

Developing a CDS View with Transactional Services

Customer feedback via product reviews can be of significant value to suppliers and to other customers seeking recommendations. To add this capability to our example application, we create a new CDS view (ZDEV210_I_PRODREVIEW_SQL), shown in **Figure 3**. The view displays all reviews collected to date. The reviews are managed as records made for a particular product by a contact person, who provides a rating (number) at a given time. The view contains an association to a product information view.

In the annotations of this CDS view, you see several lines that start with the prefix @ObjectModel to indicate that BOPF-based transaction services will be linked to this view. While many of these annotations are self-explanatory, such as createEnabled and updateEnabled, you can always open the online help by pressing F1 on the annotation. You can open the technical properties by clicking on the spiral (@) icon displayed in the leftmost column of the editor, which shows that a BOPF business object

[4] For more on SAP Fiori development, see my *SAPinsider* articles "ABAP for the Modern Age: A Look at the Latest ABAP Enhancements in SAP NetWeaver 7.5" (January-March 2016) and, with Monika Kaiser, "SAP Fiori Application Development in the Cloud" (April-June 2015), available at SAPinsiderOnline.com.

Figure 2 The generated SAP Fiori sales order invoice application

has been generated for this view. Clicking on the hyperlinked Business Object text takes you to the BOPF editor in ABAP in Eclipse (see **Figure 4**).

The BOPF editor manages all the components of a business object definition, most notably the structure of the object, which consists of a root node (in

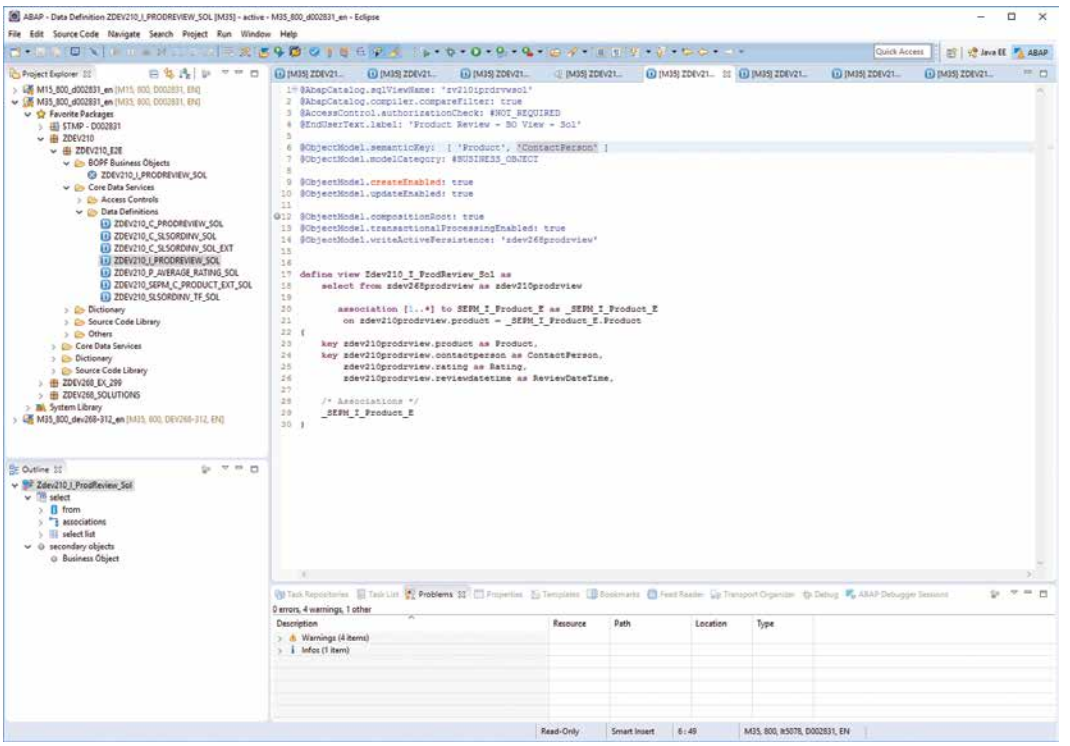

Figure 3 The source code for the example product review CDS view

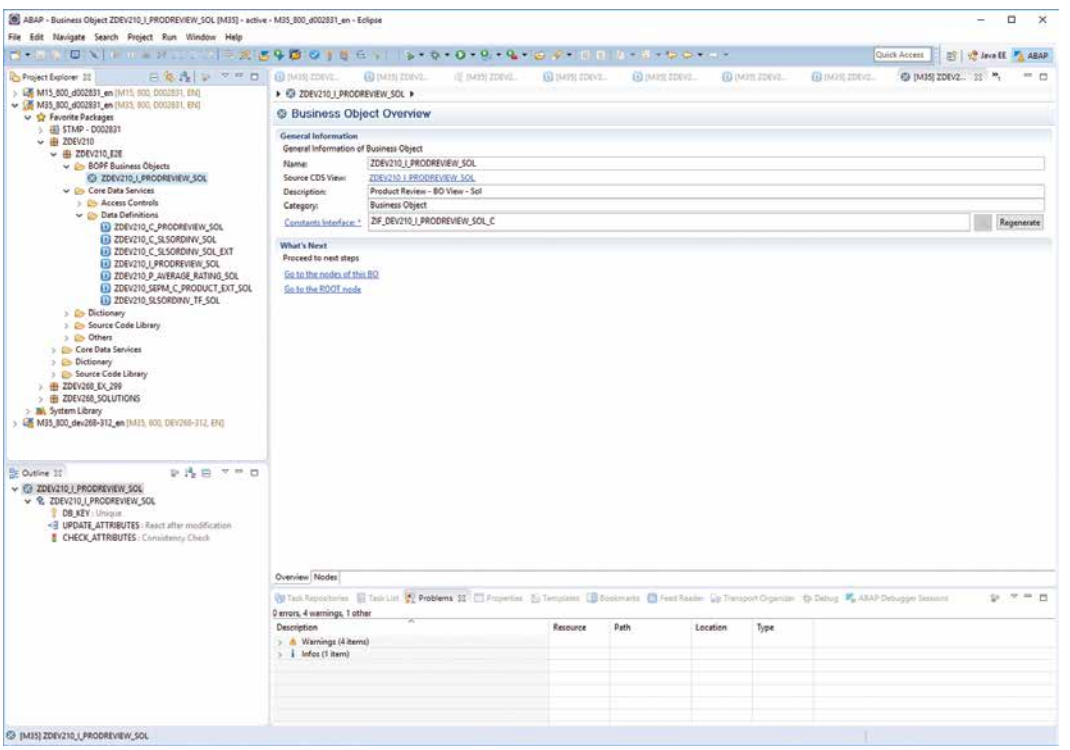

Figure 4 The business object displayed in the BOPF editor in ABAP in Eclipse

the example, the product review records) and any child nodes (none are shown in the example, but these would be similar to sales order positions, for instance). From the Business Object Overview in the BOPF editor, you can quickly navigate to overview information about the root node of the business object by clicking on the hyperlinked text (Go to the ROOT node). On the Node Overview screen, you find various technical artifacts, such as the underlying physical database table, structures and table types for business object processing, and semantic information such as BOPF associations, alternative keys, and properties. The overall BOPF framework offers a wide range of options, but for our example, we need only a small subset of artifacts.[5]

The Node Behavior section on the Node Overview screen lists Actions, Determinations, and Validations that provide programmatic access through predefined method interfaces. We are interested in the Determinations, which change the state and attributes of a business object, so we click on the hyperlinked Determinations text, which takes us to a list of the determinations configured for this node. In the example, we have a determination that is implemented as an UPDATE_ATTRIBUTES method in the implementation class ZCL_DEV210_D_ATTRIBUTES_SOL whenever a new business object is created. This behavior is controlled by triggers, which enables you to apply fine-grained control and include only triggers that are required by your application context. From here, you can navigate to a Determination Overview display where you can view general information for a determination and configure any triggers.

From the overall list of determinations, you can navigate to the implementation class by clicking on the name of the class. This takes you to the ABAP class editor where the code for the implementation class is executed, during which time the BOPF framework will transfer control to the methods implementing the business object. Since the code is regular ABAP code, you can manually modify it, but following the BOPF philosophy of standardized development, you mainly just want to enforce consistency of the business object and ensure proper error handling. There is not much to do in the case of our product review example since these records are simply inserted into the persistent table of product reviews in the underlying database. The corresponding SQL commands and commit work statements are executed by the BOPF framework, meaning no manual coding is required.

[5] The overall BOPF framework is documented in detail at the SAP Help Portal site at help.sap.com.

Creating an Application Using the BOPF-Based CDS View

With the BOPF definitions complete, you can build a new smart template application in SAP Web IDE pretty much the same way as before. The product review overview screen for the generated application will be initially empty. To add a review, click on the + symbol, which opens a general information screen where you can select a product name from the drop-down list. On the product detail screen that follows, you can enter a rating (such as 2), which populates the overview screen with the review (see **Figure 5**).

Refining the Original CDS View Using BOPF Definitions

Let's now refine our original sales order invoice CDS view by using our BOPF definitions to add fields for the standard deviation and for the average rating per customer. For the standard deviation, we add a table function to the CDS view source code that is implemented as a SQLScript procedure via the SAP HANA function stddev (see **Figure 6**). The blue background color indicates the SQLScript code embedded inside an ABAP method (known as an

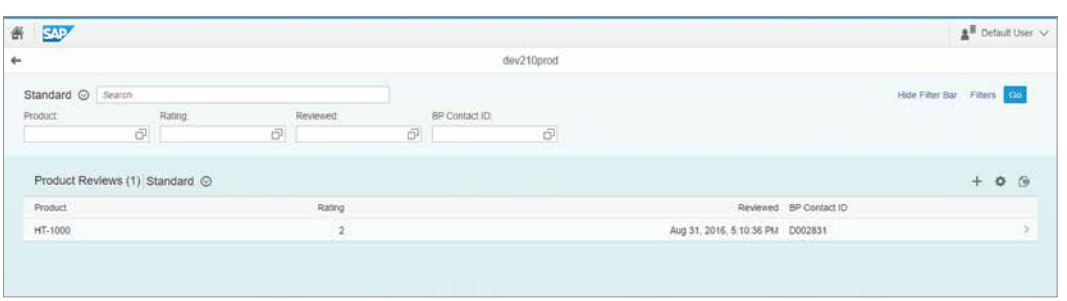

Figure 5 The completed product review overview screen with a product rating displayed

```
CLASS zcl_dev210_salesordrinv_tf_sol DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.
    INTERFACES: if_amdp_marker_hdb.
    CLASS-METHODS calculate_stddev FOR TABLE FUNCTION zdev210_slsordinv_tf_sol.

  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.

CLASS zcl_dev210_salesordrinv_tf_sol IMPLEMENTATION.

  METHOD calculate_stddev
             BY DATABASE FUNCTION
             FOR HDB
             LANGUAGE SQLSCRIPT
             OPTIONS READ-ONLY
             USING snwd_bpa snwd_so_inv_head .

    RETURN
        SELECT
            bpa.client,
            bp_id,
        stddev( days_between ( to_timestamp(so_inv.created_at), current_date) ) AS stddev
        FROM snwd_bpa AS bpa
            INNER JOIN snwd_so_inv_head AS so_inv
                ON ( bpa.client = so_inv.client and bpa.node_key = so_inv.buyer_guid )
        WHERE so_inv.payment_status = ''
        GROUP BY bpa.client, bp_id
        ;

  endmethod.

ENDCLASS.
```

Figure 6 The SQLScript code for implementing the standard deviation table function

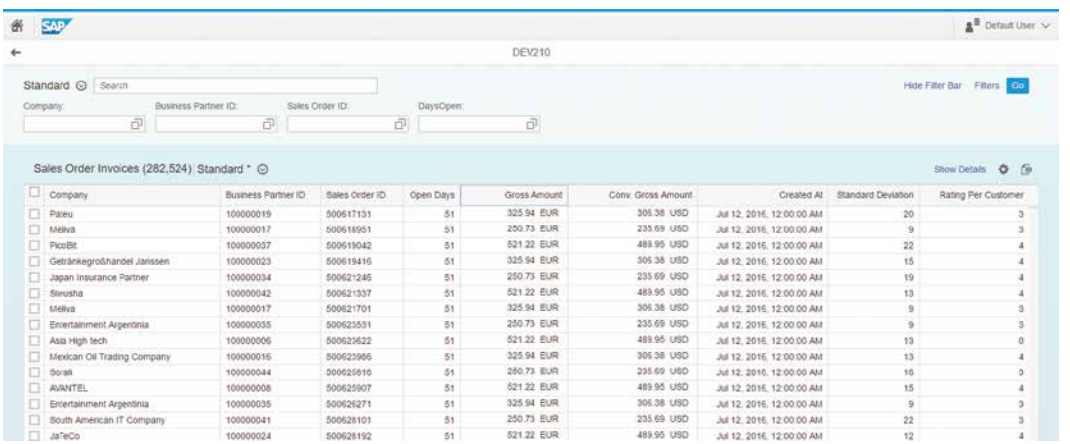

Figure 7 The display results of the sales order invoice CDS view with fields added for the average rating and standard deviation

ABAP-managed database function). For the average rating, we add a standard function that requires no SQLScript. The average rating and standard deviation are then computed and displayed in the overview screen (see **Figure 7**).

Summary

The example described here has demonstrated how ABAP 7.51 enables you to combine all the various technologies and artifacts that make up the modern ABAP programming model in a simple, streamlined, and standardized way using ABAP in Eclipse (for the back-end part) and SAP Web IDE (for the front-end part). You start with CDS views, expose them to the OData protocol, and consume them with SAP Fiori smart templates. You add transactional services based on the BOPF framework and then create new records that can be aggregated with the help of the many built-in functions from CDS or, if necessary, natively on SAP HANA.

To be successful, an innovative programming model must meet customer needs with the right development tools and methodologies. With its focus on both analytical and transactional capabilities and its support for ABAP in Eclipse and SAP Web IDE, ABAP 7.51 delivers. ■