openSAP
ABAP Development for SAP HANA

WEEK 1, UNIT 1

00:00:12 Hello, hey there, and welcome to the openSAP course ABAP Development for SAP HANA.

00:00:19 My name is Jasmin and throughout the next four weeks I will be one of the two instructors guiding through this openSAP course.

00:00:26 At SAP, I'm working as a developer for the ABAP for SAP HANA Development Integration team. As the name suggests, we are closely working together

00:00:36 with the ABAP Development organizational unit and the SAP HANA Development organizational unit to get the best integration of these two SAP products.

00:00:46 I am part of the ABAP for SAP HANA Development Integration team and with me is the head of our team, which is Jens. Hi Jens.

00:00:54 Hi Jasmin, and thank you. Also a very warm welcome to this course from my side.

00:01:00 I'm really excited to introduce you, together with Jasmin, into the upcoming videos into the details of ABAP development for SAP HANA.

00:01:09 As just said, I'm heading the ABAP for SAP HANA Integration team.

00:01:14 The team takes care to support you, our internal developers, and everybody who is interested in this topic to get the best out of ABAP and SAP HANA.

00:01:23 All the demos and topics you will see in this course are based on our reference scenario, which is shipped with every AS ABAP.

00:01:32 You will get more information on this topic later in this course. Let me now quickly outline the structure of this course.

00:01:38 This openSAP course consists of four weeks. Each week will contain several videos like this one, but of course with a much more technical focus.

00:01:48 At the end of each video, there will be a self-test to help you measure your learning progress and prepare you for the weekly assignment at the end of each week.

00:01:58 These assignments are graded, and you will earn points towards your Record of Achievement at end of this course. But enough for the organizational part of the course.

00:02:10 Now to some technical details by Jasmin. Thank you Jens.

00:02:14 Let me go one step back. So Jens just told you about the Record of Achievement

00:02:14 and we're crossing our fingers, thumbs up, for you to achieve this Record of Achievement at the end after the final exam.

00:02:26 However, I would like to emphasize for us it is very important, or the goal of this course for us and I think you will agree with us here,

00:02:35 is that at the end of this course you have a profound—or let's say basic—knowledge on ABAP development for SAP HANA.

00:02:43 Especially, you should be then familiar with the techniques and specialities of SAP HANA which are important for ABAP development,

00:02:52 that you know how you can best leverage the features of SAP HANA as an ABAP developer, and of course that you say, Hey I like this course a lot

00:03:01 and I would like learn and know even more about ABAP development for SAP HANA. Okay, it's time now to dive deep into the technicalities

00:03:11 of all of the things that we will learn in the next four weeks. But before starting really to talk about the techniques,

00:03:18 we thought it is a good starting point to show you an end-to-end developer demo of how all of the techniques can be used in one application.

00:03:26 Jens, can you please show us this end-to-end developer demo? Yes of course I can.

00:03:30 So, for that, let's first of all take a look. So, here is the URL of the end-to-end demo

00:03:37 and of course a guide in the SCN where you can really go step-by-step through this guide and develop the application that you see for yourself, at least some parts of it.

00:03:48 So, let's now switch to the browser and here I will open the link that you have just seen

00:03:55 and what you see now are a Fiori UI, which is deployed on our AS ABAP running on SAP HANA.

00:04:04 As you can see by the amount of open invoices here or the amount of invoices itself,

00:04:11 there is quite a lot of data in the system and in our SAP HANA database to really show you those UIs are very quick, responsive, and so on.

00:04:23 We can for example, also go down and analyze some details. For example, we see here the total amount of the invoices

00:04:32 opened by our customer SAP in this example, and we see every invoice detail on the right side.

00:04:39 You can switch to another business partner and you can see this is quite fast from the data point of view.

00:04:46 But still, the UI is static, so every time you need to click something to get new information. But with the new ABAP push channels

00:04:55 we have introduced, we can make those static UIs much more responsive, which is a very, very good combination

00:05:02 together with all of the new techniques applied here that you will learn in the later of this course.

00:05:08 So for that, I will switch here to our development environment, open a program which is closing invoices.

00:05:17 And now remember the amount of all open invoices or the total number of invoices

00:05:26 and I will now close one invoice with this report by business partner ID here, which is SAP.

00:05:37 Now I will execute the report by pressing F8 and take a look at the browser. Hands up—we are doing nothing.

00:05:47 You see that it is automatically refreshed because an invoice was closed.

00:05:53 So this is just a short demo teaser to give you an idea that it is really about fast UIs, getting fast data to a cool UI

00:06:04 and with that demo we will now switch to Jasmin, who will show you how to get a system to experience this on your own.

00:06:12 Thank you, Jens. Thanks for the demo.

00:06:15 I really like it a lot and especially I like that all of the cool new features are in there.

00:06:21 You have not seen the techniques yet. We will talk about the techniques in the next four weeks, but I think it's a good point to say if you like this demo,

00:06:32 I think you will find it even more convincing and you will like it even more if you really execute it on your own.

00:06:38 As Jens just mentioned, there is a trial system that you can try it out on your own.

00:06:45 So please, don't only watch the videos and the demos we are providing you here, but get your hands on such a system on your own. Try all of the things out

00:06:54 and play around with all the new features we are providing and we are talking about in the next four weeks.

00:06:59 We have detailed instructions on how to get a free developer trial edition using the SAP Cloud Appliance Library

00:07:09 and Amazon Web Services, AWS's cloud service provider.

00:07:13 Here on this SCN link you will find detailed instructions, also about the costs that will arise for you.

00:07:20 It's not the costs you have to pay to SAP, but for the cloud service provider, which is AWS.

00:07:27 Okay in this document you will find a link. I have already imported that into my browser,

00:07:34 so I will just click on the Get the Trial System and this page is now launching.

00:07:43 You see here, I'm in the Cloud Appliance Library, SAP Cloud Appliance Library, and I will be asked for some information, in particular

00:07:52 about my cloud service provider, which is Amazon, and I will be asked for access keys and security keys.

00:07:59 This information, as described in the documentation, can be obtained from the AWS.

00:08:07 I have just put that into Notepad here in order to provide it.

00:08:13 That is my information from my AWS account,

00:08:20 which is the prerequisite in order to use this Cloud Appliance Library instance as provided here.

00:08:26 Okay, the only other information I have to put in is what is called an OS password for the operational system.

00:08:35 It is very important to remind yourself of this password. Please note it somewhere, because that is really the master password getting you through everywhere in the system.

00:08:45 It is the password for your operational system user, for the WTS remote desktop connection user, for the system user on your HANA,

00:08:54 and also for all the various ABAP users in the ABAP system.

00:08:58 I will give it a meaningful password and say okay, create myself an instance.

00:09:05 This process will now take about 45 minutes. So Jens, what did you do last weekend?

00:09:13 Yeah, let me just think we went. No don't bore them with this, yes, of course it is interesting but I think we should not talk too much about it.

00:09:23 Of course, we have prepared a system. I will show you now an already active system that we have created this morning.

00:09:31 The system just logged me off. It will look very similar in your CAL account.

00:09:37 Here is the system I just created this morning. The process, as mentioned, takes 45 minutes

00:09:45 and please wait these full 45 minutes. Otherwise, some services might not be running and you might get into some trouble.

00:09:54 So wait these 45 minutes and the system will be up and operational. Once everything is up and operational, you can see here this green square indicating that it is active and you can connect.

00:10:05 That is the first point where you are asked for the operation system password, or let's say the master password you gave to your instance.

00:10:14 I'm logging in as Administrator and with the master password provided.

00:10:20 I say Yes and you can see that I am accessing as remote desktop connection.

00:10:27 Everything on that instance is preconfigured so you have, for example, the SAP Logon, the SAP Development Tools. We will hear a bit more about the SAP Development Tools

00:10:39 and Eclipse in week 1, unit 5. And I thought about what shall I demo for you first

00:10:48 once I'm in the system, and I think the best thing is just showing you the same demo as Jens has shown us just a couple of minutes ago

00:10:56 and that is again the responsive UI. I will be asked for user password credentials.

00:11:03 I will use the ABAP developer developer, and the password for developer is the same as the master password

00:11:12 or the administrator password on the remote desktop connection.

00:11:15 You can see here I'm back to this responsive UI and in principle, I could repeat the demo Jens has just shown you.

00:11:25 So get yourself such a system and try all of the things out, but please remember to start and stop your instance

00:11:32 because the system up time drives the costs and what is charged by Amazon Web Services for you is the system up time.

00:11:42 Having said that, all the detailed instructions again can be found in this SCN document and I would like to say thank you and hand over to Jens.

00:11:52 Okay, thanks again Jasmin. Now that we have introduced you to this course

00:11:58 and you know how to get a system to try the things on your own, if you have any questions during this time

00:12:07 or when you try it out in this course, please raise your hand anytime.

00:12:12 Jens, this is a video tutorial. They can't raise their hand.

00:12:17 Stupid me, sorry. Of course, we have prepared a forum for you to answer any questions and help you if you are having problems.

00:12:26 You can also find more information on ABAP for SAP HANA on our SCN page.

00:12:32 Okay, and now the only thing left is to conclude this session

00:12:40 and give you a short outlook of what is in the next unit and this will be some technical basic information about evolution and architecture of ABAP and SAP HANA.

00:12:51 Thank you very much! Bye.

WEEK 1, UNIT 2

00:00:13 Welcome back everybody to week 1, unit 2. I hope you enjoyed the onboarding,

00:00:18 and I will now introduce you to ABAP meets SAP HANA: Evolution and Architecture.

00:00:24 We will take a look at the basic principles of SAP HANA architecture

00:00:29 and how we improved the Application Server ABAP in the recent year to welcome HANA.

00:00:37 So first of all, let me explain what we want to achieve with SAP HANA.

00:00:43 If you developed applications so far, you are probably well aware of the fact

00:00:48 that this typically requires making several trade-off decisions.

00:00:52 The graphics on the slide show the five dimensions of requirement that are typically for business applications.

00:00:59 With a traditional database, these dimensions have been conflicting so far and challenged us developers very badly.

00:01:07 For example, you had to make a decision between providing a high-speed application and developing an application

00:01:14 that doesn't require preparing the data in a special way first.

00:01:19 The classical and very common example of this traditional pattern can be found in our SAP Business Suite:

00:01:27 Aggregations are calculated beforehand, stored in a database, and then later used in the application.

00:01:35 Another example is that you couldn't develop an application which is both real-time

00:01:42 and able to analyze a large amount of data at the same time.

00:01:47 With SAP HANA, we are now able to deliver across all these dimensions

00:01:53 so that you no longer have to make so hard trade-off decisions in your applications.

00:01:59 Long story short: We want to enable application developers like you to be able to develop applications

00:02:06 which support online transactional processing and online analytical processing usage patterns at the same time.

00:02:14 If you have already heard about the magic sentence “OLTP and OLAP conversion”—

00:02:21 if not, you know it now—this is the basic idea behind it.

00:02:26 Let me now give you an idea how we were able to realize this.

00:02:33 One important ingredient is the evolution of hardware over the last years.

00:02:39 For example, let's talk about the evolution of CPUs. I still remember buying my first PC.

00:02:46 It was a 386 with 20 MHz, and it was awesome—at least at that time.

00:02:53 And now take a look at modern multi-core CPUs with 60 cores, 3 GHz each core—this is a complete other story.

00:03:02 If you get it down on instructions per second, you had about 0.05 million instructions per second in a 1990s CPU.

00:03:12 And in 2010, you had already 7.15 million instructions per second.

00:03:21 This is a factor of 143. Just imagine.

00:03:26 Also, the addressable memory space increased with the 64-bit architecture,

00:03:31 and in conjunction with a dramatic decline of RAM price.

00:03:36 Even this laptop here has already 8 GB of memory.

00:03:41 And the memory, the RAM, is of course much faster than hard disks. We are talking about a factor of 100,000.

00:03:51 These and some more hardware innovations, and the price decline, made it possible for SAP

00:03:57 to build and evolve the SAP HANA database as you know it today.

00:04:06 Now let's take a look at the recent SAP software innovations

00:04:10 that enabled us to rework the classical database design.

00:04:15 One important step here was the idea to store the data not in the classical row store approach

00:04:20 but to store it in a column-based approach.

00:04:24 I will show you more details on this concept in the next unit.

00:04:29 The next important innovation is data compression. The idea behind it is simple.

00:04:35 A single CPU is much faster than the memory bus and disk.

00:04:41 Hence compressing data in order to reduce the amount of data is beneficial

00:04:46 as long as the overhead of that is not too huge.

00:04:51 Therefore, every major database supports compression.

00:04:55 However, it is first of all used on a row store and is also compressing a database block

00:05:01 and decompressing it, it takes its toll, most obvious when data is updated inside a database block.

00:05:11 This brings us here to the next innovation in SAP HANA: the insert-only approach.

00:05:18 This means that data in a table will not be overwritten when doing an update

00:05:24 but instead, a new entry with the same primary key at the end of the table is appended.

00:05:33 At this point, you might think, he's crazy. The same primary key?

00:05:39 Yes. It's a new entry with the same primary key.

00:05:42 The trick here is that HANA adds a transaction ID to the entry,

00:05:47 and when selecting the data, only the entry with the highest transaction ID is returned.

00:05:53 Furthermore, the deletion of a column will first of all only mark the entry as deleted

00:05:59 and not remove it immediately from the table.

00:06:04 Also, newly inserted data is not immediately added to the main table itself,

00:06:10 but written in an uncompressed delta table which is transparent

00:06:15 and automatically joined to the main table when selecting the data.

00:06:20 To sum it up, it is important for you to get the basic idea of all these concepts

00:06:25 and how all of them work together in SAP HANA.

00:06:30 Of course, all of these mentioned techniques have drawbacks,

00:06:34 but the combination of all of them effectively removes their drawbacks

00:06:39 and this is a major reason why SAP HANA is such a game changer.

00:06:45 So, what do we want to achieve with SAP HANA?

00:06:49 A very important goal is to allow for new applications to be developed

00:06:53 but also to allow optimizing existing applications.

00:06:58 It is also clear—high performance and scalability are goals.

00:07:03 SAP HANA is also a hybrid data management system, combining different paradigms in one system such as

00:07:12 column-based and row-based in-memory storage, text analysis, search, and even built-in support for planning scenarios.

00:07:22 Also very important to know is the possibility to use the SAP HANA database system

00:07:28 as a replacement for any standard relational database management system.

00:07:34 The SAP HANA database system is a full relational database management system

00:07:40 with an SQL interface, transactional isolation, and more.

00:07:45 That's very important when it comes to migrate to SAP HANA.

00:07:50 From the migration perspective, it is nothing else than any other database migration.

00:07:58 Now let us take a look on the history of ABAP and SAP HANA.

00:08:03 As just mentioned, from the ABAP stack's perspective, SAP HANA first of all is a new database.

00:08:13 Like for any other database management system, to be supported,

00:08:18 the ABAP stack thus needs a specific database driver that it can interact with the database management system.

00:08:27 As result, the first step to enable ABAP for HANA was to add such a driver,

00:08:34 allowing an ABAP system to access SAP HANA as a separate, secondary database.

00:08:41 This happened for SAP NetWeaver 7.0 release and onwards.

00:08:47 It is basically what is required to make the accelerator type of applications.

00:08:53 The second step was to make sure that the ABAP stack can really run on SAP HANA,

00:08:59 using it as its true database in which all ABAP source code, tables, and application data is

stored in the HANA database.

00:09:10 This took place with SAP NetWeaver release 7.31 and specially for the SAP Business Warehouse product.

00:09:18 The final step, and that's the step we are mostly interested in from the perspective of this course,

00:09:25 was to truly optimize the ABAP stack for the case when it runs on SAP HANA,

00:09:30 and especially to allow application code to be optimized for such a case.

00:09:38 What this really means covers the rest of this course.

00:09:42 Let's now take a look which application types are available with ABAP and SAP HANA.

00:09:49 The left-hand type of applications are using SAP HANA to accelerate ABAP applications.

00:09:55 These applications are putting a separate HANA server beside the ABAP stack,

00:10:01 replicating parts of the ABAP system's data into HANA.

00:10:08 Then calling HANA remotely from the ABAP stack in a selected piece of code to speed up special reports or transactions.

00:10:19 But the primary focus of this course, as I already stated, is to learn how to optimize ABAP applications directly running on SAP HANA.

00:10:29 The applications are using SAP HANA as a primary database underneath the ABAP stack.

00:10:35 A typical example of this kind of application is SAP CRM powered by SAP HANA, or the Business Suite powered by SAP HANA.

00:10:46 The latest release of these two applications—and not to forget SAP Business Warehouse 7.4 powered by SAP HANA—

00:10:56 are all based on SAP NetWeaver Application Server ABAP 7.40.

00:11:01 It is the “go-to” release for ABAP development on SAP HANA and the next enhancement package after SAP NetWeaver 7.03 and 7.31.

00:11:13 Also important to know is it supports, of course, SAP HANA, as well as any other classical database system.

00:11:22 And of course, all the demos and the trial systems of this course and the available developer edition itself

00:11:30 are based on the Application Server ABAP 7.40.

00:11:37 Let's have a very quick look at where ABAP 7.40 provides support for optimizations on ABAP applications running on SAP HANA.

00:11:47 There are first of all transparent optimizations. These optimizations you do not see as an application developer,

00:11:56 but which you still benefit from because they happen under the hood.

00:12:01 An example is the SELECT… FOR ALL ENTRIES statement in Open SQL, the implementation of which has been improved.

00:12:10 The ABAP language and the Open SQL language have been extended.

00:12:15 For example, Open SQL now supports more complex joins and sub-queries,

00:12:22 and there are ways to integrate HANA stored procedures. In other words, you can write SQLScript in ABAP code.

00:12:30 Later more on this topic. Consumption of SAP HANA artifacts has also been simplified.

00:12:37 For example, you can access HANA column views through ABAP Data Dictionary and using it with Open SQL.

00:12:46 In addition, performance analysis tools have been improved to make your life easier.

00:12:52 Example: Static code checks and runtime analysis.

00:12:57 Finally, some re-use components have been extended to be optimized on SAP HANA.

00:13:04 We will cover all of these topics during the course in the upcoming weeks.

00:13:11 Okay. You made it, so let me shortly summarize what I've said again—the key takeaways of this unit.

00:13:20 I've talked about hardware and SAP software innovations that enabled SAP HANA.

00:13:27 You've heard about HANA goals, as well as the ABAP Application Server architecture.

00:13:33 Finally, I've shown you the key features of the ABAP 7.40, optimized for SAP HANA

00:13:40 which will be discussed in detail in the upcoming units of this course.

00:13:45 With this, I would like to close the session. Stay tuned for the next unit

00:13:50 in which I will introduce you to the basic need-to-know when working with SAP HANA from an ABAP developer's perspective.

00:13:57 Thanks and bye-bye.

WEEK 1, UNIT 3

00:00:11 Welcome back to week 1, unit 3.

00:00:15 In this session, I will show you the basics which you need to know when SAP HANA is your primary database.

00:00:21 In the last unit, you learned the basics about SAP HANA architecture and which hardware and SAP software innovations enabled SAP to build HANA.

00:00:31 Now let me start by taking a deeper look into the in-memory storage types of SAP HANA—

00:00:37 the column store and the row store—and I will give you an outline of what is important for you as an ABAP developer.

00:00:47 First of all, what should you know about the row store? Let's take a look about this example.

00:00:53 Here we have a snippet of a table with an order key, a customer, and the amount with the currency code.

00:01:00 Basically, a very simple sales order table. In the classical row storage, the data is stored row by row.

00:01:10 This has some advantages and some disadvantages. For example, compression of business data in a row store is typically not as good as in a column store.

00:01:21 Now let's take a look at the performance perspective.

00:01:27 The row store is performing good when you select a single record and all the columns of this record, like this example, with the SELECT * statement

00:01:36 and the primary key as a WHERE condition. And the other example aggregates all amounts.

00:01:42 Here, the row store's performance is lower in comparison to the column store.

00:01:50 Now taking a look at the column store. Here, the data is arranged by each column in the database.

00:01:58 Let's take a look at what happens performance wise in the column store with our two SELECTS.

00:02:04 The single, broad SELECT gets slower compared to the aggregated SELECT. The full table performance is better.

00:02:12 You have to read one column only, hence just a fraction of the whole data table is needed.

00:02:19 This will be much faster as with traditional row-oriented tables

00:02:25 where all the data is in the same database object and you have to read the entire table in order to figure out each row's column value.

00:02:35 Another advantage of the column store architecture is that in most cases, you don't need a secondary index on the tables.

00:02:45 This is again based on the column layout, which closely matches the structure of the secondary index in classical row store tables.

00:02:57 Being able to choose the storage type for each database table leads to a follow-on question: Which type is best for which database table?

00:03:10 This brings me to the first guideline of ABAP development of SAP HANA.

00:03:16 If a table can hold a huge amount of data,

00:03:20 and if data will potentially be aggregated on, analyzed, or searched a lot, then a column store is more suitable.

00:03:30 Other reasons for choosing column store are if the table contains many columns of which only a few really have to be read usually.

00:03:39 These properties typically hold true for both master data tables and transactional data tables.

00:03:46 One example here is a MARA table in our SAP Business Suite.

00:03:53 Row store may be more suitable if you want to report on all of the columns of the table in the most cases.

00:04:00 That's because reconstructing the complete row is one of the most expensive column store operations.

00:04:08 Additional reasons for choosing row store are if the table really is not aggregated on or not searched in using columns

00:04:15 other than the key column, or if the table contains unstructured data. And last but not least, if the table is completely buffered on the ABAP server.

00:04:26 The latter implies that the table contains a few records only—even in a production system.

00:04:34 These properties might hold true for customizing tables or tables used to queue data,

00:04:42 but the guideline to choose row store in case is less strict than the guideline when it comes to the column store.

00:04:52 Now that you know about the differences in the row and column store, you might ask the question,

00:04:59 How can I move a table from row to column store, or vice versa.

00:05:03 For this, I have prepared a demo on how this works.

00:05:07 Let me switch to the development environment and go to the SE11. Here I will select a table, but000 for example, and display it.

00:05:20 When we now switch to the Technical Settings tab, we find a new sub-tab here which is labeled with DB-Specific Properties.

00:05:32 Here you can now choose the storage type of this specific table and you see here it's already in the column store,

00:05:42 which is also the case for most of the tables in the SAP Business Suite when you do the migration to SAP HANA.

00:05:49 So column store is basically...most tables are in the column store and some are in the row store as the criteria I have just told you in the last slide.

00:05:59 So, now you know how you could switch this. It is also important for you to know that switching this is a big work for the database.

00:06:10 It cannot be easily switched on and off, especially when there is a huge amount of data in the table.

00:06:18 The switching of the table takes time.

00:06:23 Let me now go here and let's take a look at the first Technical Settings tab.

00:06:38 What I wanted to mention here is the table buffer.

00:06:41 This table buffering is not allowed, but for tables which you had already buffered, it is still true that the table buffer is still valid.

00:06:51 This is first of all because in the application server, the data is already there in the memory

00:06:58 and if you think about network latency, this takes just some time if you pull it out of an in-memory database through the network to the application server.

00:07:08 Second reason here is that the types of the database are different than the types of the application server.

00:07:17 So database has it's own type system.

00:07:20 Let's take a look at this and I will also tell you why this is important for you in some cases.

00:07:27 Here we can go to the Utilities—>Database Object—>Display.

00:07:36 Let's compare this to the classical view of the ABAP data types.

00:07:42 Here, you see that we have charlike fields on the database for the fields here and we here we have the classical ABAP types as you know.

00:07:52 This is not so important if you just keep working with Open SQL

00:07:58 because there the data type mapping is completely transparent for you and done by the application server.

00:08:04 But when it comes down to applying some code pushdown mechanisms, which Jasmin will tell you in the next unit what that is,

00:08:11 it's very important for you to know that there is a different type system and there is a mapping.

00:08:19 Okay, and let me conclude this demo by showing you also a little bit more info about the indices.

00:08:27 Some indices are not needed anymore in SAP HANA.

00:08:31 For that, we have also made the possibility that you can have database-specific indices.

00:08:38 Let's take a look at what this means. Selecting this index here and you see multi-columns here and multi-column indices

00:08:46 are not needed HANA anymore because of the column store, as you have just learned.

00:08:52 What we can do here now is go to Database Systems and here you see that you can have an exclude list or a selection include list

00:09:02 for a database or for other databases. And this here means just that this index is not needed on an HDB system.

00:09:10 Okay, that's it for the demo part. Let me now switch back to the presentation.

00:09:18 Now let's take a closer look at how the column store works and how the compression works.

00:09:27 SAP HANA uses different efficient compression methods, for example, dictionary and coding.

00:09:34 This reduces the amount of memory required and speeds up column operations.

00:09:39 First, the columns can be loaded into the CPU caches faster and with fewer loading cycles.

00:09:46 Second, the comparison between integer values is much faster than string comparisons. Let me give you an example of how this works.

00:09:54 For this, I will use again the sales order table, which now holds some more data.

00:10:00 As you can see, here is the dictionary for the column Customer, which is sorted with the distinct values.

00:10:10 But this does not explain the compression yet. By encoding the information as indicated here,

00:10:18 only 5 entries end up encoded in only 3 bits. With this encoding, the columns are compressed as shown here.

00:10:29 So if you now need to know where order 460 is, for example, you know from the logical table that it is in position 5.

00:10:39 Adding the information of the compressed column, you know it is in position 2 then in the dictionary. And then you got the value of the column.

00:10:49 This works completely using integer values in the comparison. Now let's go one step further.

00:10:57 You want to know which sales orders correspond to customer SAP. For this, an inverted index is used.

00:11:07 From the dictionary, you know SAP is in position 3 and with this information, you can use an inverted index to get the positions of the values in the logical table.

00:11:18 Again, mostly only integer comparisons have to be applied here.

00:11:24 Let me close the session with a short wrap-up of the unit.

00:11:29 The SAP HANA in-memory database supports classical column and row store.

00:11:34 You got to know the basic idea of how column store is working, and as an ABAP developer, you know how to switch between row and column store in the SE11.

00:11:44 You also got some guidelines when to choose which type of storage.

00:11:50 And now to the next unit. Jasmin will introduce you to the new code pushdown paradigm in ABAP.

00:11:57 Thanks a lot and bye, bye.

WEEK 1, UNIT 4

00:00:13 Hello and welcome back to the openSAP course ABAP Development for SAP HANA.

00:00:19 Now that you've heard from Jens in the last two units about ABAP meets SAP HANA, about the evolution, about architecture,

00:00:27 and also the things that you need to know as an ABAP developer, when it comes to development for SAP HANA.

00:00:33 I will be today talking about the Code-to-Data paradigm.

00:00:37 You might say, I don't know what that is. Of course I will be talking about this, so please relax and listen to me carefully.

00:00:44 Okay, Code-to-Data paradigm. Why is that interesting for you?

00:00:49 Before I go the theoretical introduction of what the Code-to-Data paradigm is, I would like to show you a small example,

00:00:56 application example, of what the Code-to-Data paradigm can do for you respectively for your ABAP implementations,

00:01:03 ABAP applications, and that is in particular a boost in performance. For that I have a small demo right at the beginning of this unit.

00:01:13 Okay, let me directly go into the system. I have prepared two ABAP reports for you.

00:01:20 On the left-hand side you can see here a classical implementation, and classical means that this is using classical Open SQL.

00:01:29 This could even be improved with the things that I will show you in week 3, units 1 and 2.

00:01:35 However, let's just say this is a classical ABAP implementation here on the left-hand side.

00:01:40 On the right-hand side, this report is following the Code-to-Data paradigm and, in particular, in the form of a AMDP, standing for ABAP Managed Database Procedures.

00:01:52 What that is, I will come to that in a second. Basically the two reports are doing the same application logic for a given date.

00:02:01 They are giving me information about sales order invoice information.

00:02:06 Let me just execute the classical implementation giving the classical implementation I had up and also start the Code-to-Data paradigm variant.

00:02:19 As you can see here, this report already finished after about half a second and I'm waiting a bit and you can see here also ABAP finished successfully.

00:02:28 Same information is provided, but as you can see, following the Code-to-Data paradigm boosted the performance of retrieving this information.

00:02:38 This should already give you a hint what the Code-to-Data paradigm might mean for your application,

00:02:45 so what we are expecting is of course a performance boost and that we are really leveraging the full power of SAP HANA with that.

00:02:54 Okay, so let me go back to my slides and tell you a bit more about theoretical part of the Code-to-Data paradigm.

00:03:03 We have learned from Jens in the last two units that SAP HANA is not just a classical database. Of course, it is a database,

00:03:13 so you can store data in there, but HANA is even more.

00:03:17 HANA can do a lot of things for you, in particular, doing calculations, text search, data-intensive calculations, and so on and so forth.

00:03:26 So the classical approach, as you can see on the slide here on the left-hand side, is the typical,

00:03:33 classical approach is that you move all of the data that you need from the database layer up to the application layer,

00:03:41 do all of the calculations that you need to do, all the application logic on the application layer, and then finally transfer the result to your user interface, like the browser.

00:03:51 With the in-memory capabilities of SAP HANA, we can change this picture.

00:03:57 The implications of the in-memory capabilities is that we can now do calculations directly on the database.

00:04:06 Why not transfer these calculations? I mean the data already resides there, so push down the code to SAP HANA.

00:04:14 Of course, this does not mean use the database as a calculator, but really data-intensive calculations directly on the database.

00:04:23 You still need to avoid unnecessary movement and you can avoid unnecessary movement, but please also don't burden the database with unnecessary operations.

00:04:34 Perform your data-intensive calculations in the database.

00:04:38 Okay, the question is now, how can I do that? And for this purpose we have provided in ABAP 7.4 a lot of capabilities.

00:04:49 An outline of all of those capabilities is shown here on that slide. You have seen, in principle, the same information

00:04:58 in this building-shaped diagram that Jens has shown you in unit 2 of this week. In the base, at the fundament, we have transparent optimizations provided in ABAP 7.4.

00:05:12 This includes optimized protocols on how the database and the application server communicate with each other.

00:05:21 This is also includes table buffer enhancements. There are things like secondary indices which have not been respected for the table buffer before

00:05:30 but we have enhanced the table buffer and we also have optimized, like I said, protocols like that fast data access which is used in things like Open SQL SELECT...FOR ALL ENTRIES statements.

00:05:42 Above these transparent optimizations, ABAP 7.4 includes things for database-oriented programming.

00:05:50 Among these things are enhancement of Open SQL. You will hear more details about that in unit 1 and 2 of the third week in this course.

00:06:00 And we have also introduced advanced view definition capabilities,

00:06:05 the artist which is also called CDS view definition or CDS views.

00:06:11 We will hear about CDS views also in week 3 in the other units after introducing enhancements in Open SQL.

00:06:20 And last but not least, if you really have to squeeze the last percentage or the last per mille for your application

00:06:28 and you really would like to get all of the things deep down into the HANA and do really all of the application logic deep down,

00:06:38 you might need to go for Native SQL, like EXEC SQL or ABAP Database Connectivity (ADBC)

00:06:44 or the guys you have just met in the examples beforehand in this demo report, ABAP Managed Database Procedures, which will be introduced in the fourth week of this course.

00:06:54 Turning a little bit, this picture shows you that there is nothing like a free lunch, so there are transparent optimizations that will give you performance improvements

00:07:07 and they basically come for free. You don't have more code complexity or you don't have to do huge adjustments. You have transparent optimizations. You don't see them.

00:07:17 Of course, we hope you see them in performance, but you don't have to do anything different concerning coding.

00:07:22 If you go for new Open SQL statements or the enhancements of Open SQL, same holds for CDS views in the advanced view definition.

00:07:32 You might gain more performance, you might gain more performance improvements, but on the other hand side, also code complexity rises.

00:07:40 And this goes up until ABAP Managed Database Procedures or Native SQL, which have more complexity in the coding,

00:07:48 but as I mentioned before you might squeeze the last per mille of your performance in your application with that.

00:07:56 Once we have started with a product, ABAP 7.4 with the first Support Package, the SP2 support,

00:08:05 we said, okay how can we optimally support ABAP developers following the Code-to-Data paradigm? And for that we have this bottom-up approach

00:08:16 which is not the focus of this course but for completeness reasons, I would like to also talk a bit about this bottom-up approach.

00:08:24 It is called bottom-up because assume you have a colleague who knows HANA

00:08:31 or you have from SAP released some HANA views or database procedures.

00:08:38 So using advanced view building capabilities in SAP HANA or procedural coding in form of database procedures.

00:08:46 So you as an ABAP developer, you don't care about the technicalities that are in these views or the database procedures,

00:08:53 but you would like to consume them and you would like to consume them in a rather native way.

00:08:59 We have provided capabilities for that for you in the Data Dictionary. In particular, we have

provided external views and database procedure proxies

00:09:09 And as the name indicates for database procedures, both are proxy objects,

00:09:15 so they are rather wrappers for the HANA views and database procedures in order make them consumable in the ABAP.

00:09:23 I will show that in a short example here directly in the system, so in case you might stumble over these things in your system.

00:09:35 So that is how an external view would look like. You see here a dictionary view. That is a wrapper for that HANA view,

00:09:45 you see a mapping between dictionary type and HANA types. The same holds for database procedure proxies.

00:09:52 You see here a database procedure proxy as wrapping the HANA procedure, and you also get a database procedure interface for type definitions.

00:10:02 That is how you would stumble over in report or in class coding.

00:10:09 Let me enlarge it a bit. So external view does not look too much different from a standard view or a standard dictionary table.

00:10:17 You can use the external view in order to declare your internal table.

00:10:25 The same holds for the database procedure. You can use this interface to define a type and if you would like to call a database procedure proxy,

00:10:34 you can use the new ABAP language statement CALL DATABASE PROCEDURE.

00:10:39 So in principle, this is a very great concept, but it has some drawbacks and the problem is the life cycle.

00:10:50 One of the problems or one of the issues is the life cycle that needs to be challenged.

00:10:55 So HANA views and HANA database procedures are managed by the HANA Lifecycle Management System. They are living in a delivery unit

00:11:04 which need to be transported. And on the other hand side, you have the external views and the database procedure proxies.

00:11:13 If anything changes here on the database level, Data Dictionary does not know it. You have to synchronize the proxy objects for the views manually

00:11:23 and you also need to transport the delivery unit.

00:11:28 You can do that in the SAP HANA Transport Container, which is also a proxy object.

00:11:34 Okay, so we have these drawbacks, especially about life cycle. That is why we said okay, why don't we do it the other way around?

00:11:44 And that's why we introduced the top-down approach. Starting with ABAP 7.4 Support Package 5 and onwards,

00:11:52 we are providing you with the top-down approach, including Core Data Services views or the advanced view definition capabilities

00:12:02 and things like ABAP Managed Database Procedures.

00:12:05 The picture changes because we are creating, as ABAP developers, these CDS views or the

AMDPs and once we activate, respectively

00:12:15 when we first execute these AMDPs, we are deploying on SAP HANA the HANA views and the database procedures.

00:12:24 That also means that we only have to transport the ABAP artifacts, which are the CDS views and the AMDPs, using the standard CTS, the well-known Change and Transport System within the ABAP.

00:12:37 Okay, so I've been talking about the Code-to-Data paradigm. I have shown you the key players, the CDS views

00:12:46 and the ABAP Managed Database Procedures, and of course also the transparent optimizations in Open SQL.

00:12:53 Tomorrow, I would like to introduce the tools that you need to have in order to manage

00:13:03 and in order to create these news ABAP artifacts. And I think it's going to be a very interesting session tomorrow.

00:13:10 I hope to see you tomorrow. Bye-bye.

WEEK 1, UNIT 5

00:00:11 Welcome back and great to see you again for this last unit of the first week when we are talking about interesting topics like your development tools.

00:00:23 I will be talking today about the SAP Development Tools in Eclipse and in more detail about the ABAP Development Tools in Eclipse.

00:00:33 Let me first show you some strategic pictures of the Eclipse strategy of SAP.

00:00:40 We have seen in the first unit of this week that we had an end-to-end developer example.

00:00:46 We had an application that is using HANA via ABAP to an SAP Fiori-like application with SAP Fiori tiles.

00:00:55 Actually, all of these developments can be done in just one single integrated development environment, just in Eclipse.

00:01:04 You can see here on the slide on the left-hand side, a HANA native development example.

00:01:11 In the center, ABAP development, and on the right-hand side, SAP Fiori-like application development.

00:01:19 So all of that can be done in one IDE by just switching the perspective.

00:01:29 So what we promise you here is that we can increase or that you increase your developer productivity,

00:01:35 that we have only one single integrated development environment, and you have extensible development tools.

00:01:43 So we hope, or we think, that you can significantly improve your developer productivity.

00:01:49 In case you would like to have the SAP development tools on your own at home, just follow this link given on the slide here.

00:01:58 It will show you how you can get all of the tools necessary for this end-to-end development from just one Eclipse installation

00:02:08 and including all of the tools as plug-ins. But if you have by now a trial edition of the ABAP server

00:02:17 as we have discussed in the first unit of this week, it's even better.

00:02:23 So let me go to the Cloud Appliance Library again. I have shown you that in the first unit.

00:02:30 I'm just connecting to my front end with the remote desktop connection, using my administrative password

00:02:38 and you can see, as mentioned before, everything is preconfigured. That also includes the SAP Development Tools.

00:02:47 Typically once you start the Eclipse installation, you are here welcomed by your IDE

00:02:52 and the first step is typically to go to the workbench, to the ABAP Workbench.

00:02:59 Now you say okay, I'm a bit lost from all of those windows and all of those views but help is near.

00:03:06 Just go for Help→ Help Content and you will find information here on SAP – ABAP

Development User Guide.

00:03:15 Just click on that and you will find the Getting Started, Concepts, Tasks, and so on and so forth.

00:03:23 Then, typically, the first thing that you do with you IDE is you create an ABAP project.

00:03:30 The ABAP project is your central interface to communicate with the ABAP back-end system.

00:03:36 This is also already being maintained here in the trial version. You can see or you can find the information about the ABAP back end in your SAP Logon.

00:03:46 I will just show you how you would create such an ABAP project.

00:03:50 Of course, it's already there so this is an obsolete step, but still I would like to show you that.

00:03:57 I just do a right-click here in the Project Explorer, do New → ABAP Project, and I can browse the connections as they are maintained in the SAP Logon.

00:04:08 I just browse it. You see A4H, the developer edition, and you provide the client, the user, which is developer,

00:04:18 and the password, which is again the master password. So here now you have the same thing two times.

00:04:25 If you open that up, that was the other project. You see here Favorite Packages

00:04:33 and pre-configured is your $TMP. Of course, you can add any other package you would like.

00:04:41 Because the resolution is not so nice, I would like to show you that in a local installation, not on the Windows Terminal Server.

00:04:50 So let me just go to my local installation. So, I mentioned we can add favorite packages.

00:04:57 I will do that, I'll just add the package, the demo package OPENSAP_A4H1. You can see I have some sub-packages,

00:05:08 for example, week 1. Typically when you are doing development and you have something new to try out, you start with a Hello World and so will I.

00:05:17 I just go to this package in week 1, right-click to get the content menu, and go for New → ABAP Program.

00:05:28 I will do ZR_HELLO_WORLD with some meaningful description. Hit on Next. It will ask me for a transport request. I already have something open

00:05:39 and you Finish. And as you don't want to see me typing all the time, I have created a template that inserts here some ABAP coding.

00:05:52 So I have two local variables. One containing my user name, which will be Developer, and some string

00:06:02 like “Hello, welcome to the ABAP Development” and I will just write that to the screen.

00:06:09 I can activate that. There are always shortcuts. Whenever I am using a shortcut, you will see a small popup.

00:06:17 Okay, so what you have seen here is the Project Explorer. You can see that the thing is here now only read-only.

00:06:28 We are in a source-code-based environment, so you can see that this is read-only and this is

going to change when I start typing.

00:06:36 You see that I am writing here and I have an indicator that I have a dirty source.

00:06:43 Let me just remove it because it doesn't really make sense. Okay, I can activate it and in principle I can also execute it right now.

00:06:53 But before I do that, I would like to show you a very nice feature that is provided by the ABAP Development Tools

00:07:01 and that is you can search for any other development object using a shortcut, Ctrl+Shift+A.

00:07:09 You can just simply search, for example, on the class. That is the class I've shown you during the unit on the Code-to-Data paradigm.

00:07:19 So Eclipse allows us to do a lot. Here again we are in read-only mode and what is different than compared to the ABAP Development Tools,

00:07:27 is that you don't have the form-based editor but you really have a source-code-based editor.

00:07:32 You have an outline, you have the possibility to see properties about this class, and so on and so forth.

00:07:40 You have navigation. Just go to the method, for example. You can have information about this type.

00:07:49 You see here already the structure. You can navigate, for example, to the definition of the method and so on and so forth.

00:08:01 So there are a lot of things. I cannot talk about all those things,

00:08:08 however, it is really productive once you are already familiar with the shortcuts you can do.

00:08:14 Okay, so let me just execute this report and here something happens

00:08:23 that will happen whenever there is no native support for the tools you are using here in Eclipse.

00:08:30 So whenever there is something which is not natively supported, SAP GUI will open.

00:08:37 Same thing holds, for example, if I go back to the class, you can see here I have a SELECT somewhere. I'm selecting from a dictionary table.

00:08:48 I can have structure information on that. I can even pin that here as a new window

00:08:55 but if I would like to navigate to that dictionary table, I'm using F3 to navigate there

00:09:03 and you can see an integrated SAP GUI will open and will guide me to the SE11.

00:09:11 Okay, one last word or one last thing that I would like to show you is debugging

00:09:21 because there is nothing like a /h that I can hit in the OK Code field, because there is no OK Code field.

00:09:28 Here is a field, but this tells me quick access, not an OK Code field.

00:09:32 So what you have to do in order to do debugging in your environment is to set a breakpoint and just execute.

00:09:39 The first time we are doing this, it will ask me to launch another perspective and that brings me to the question of what exactly is a perspective?

00:09:49 Well, Java developers are perfectly known to this concept and also people familiar to Eclipse are perfectly known to this concept.

00:09:57 Perspectives are just a collection of all those views you see here, designed to perfectly work together with the development task at hand.

00:10:07 So the windows I had here were perfectly fine for doing ABAP development.

00:10:13 However, for doing debugging, another perspective might be more suitable.

00:10:18 I say okay when the debugger is launched, please remember that my decision I would like switch to the debug perspective.

00:10:27 You can see the perspectives over here and you can even open them on purpose. However here, in this example,

00:10:36 the debugger perspective was launched when launching the ABAP Debugger.

00:10:41 You can see here now the stack. You can see again the source code. I could even add some information just in this debugger

00:10:50 but that will, of course, only work after activating again. You can see variables,

00:10:56 where I have my breakpoints which are active, and now I will change the value of LV_NAME,

00:11:06 that is DEVELOPER, as this is my developer. I will say okay, I would like not to say Hello developer, but to say Hello Jens, who is actually not here today.

00:11:17 However, I just continue and as you can see then, “Hello Jens, welcome to ABAP Developmer for SAP HANA!”

00:11:27 So as I said before, this is only a small peek into everything that can be done with the ABAP Development Tools in Eclipse

00:11:38 and also with the SAP Development Tools in Eclipse. You are really productive when you know all the shortcuts.

00:11:46 They are around and throughout the next units. Whenever we show you a system demo, we will have the shortcut visualized for you in order to see that we are not doing any magic here

00:11:56 or exactly which magic we are doing here. Here is a small list of shortcuts

00:12:02 and I'm pretty sure that if you know these shortcuts, these small number of shortcuts, you are already very, very productive.

00:12:11 I've shown you this Ctrl+Shift+A that helps us to open development objects.

00:12:16 I absolutely love this feature. And if you would like to find more information about the ABAP Development Tools,

00:12:25 just visit the colleagues on the ABAP Eclipse SCN Community or there is even something around called SAP CodeJam. Just have a look there.

00:12:35 Okay, with this I would like to say, try all of the things out on your own.

00:12:42 Write your classes, write Hello World programs.

00:12:48 Do something in your system and try all of the things out, just test everything to see if you like it or not.

00:12:54 At the end it's a development environment. And with this I would like to conclude the first week.

00:13:01 After the onboarding we have heard from Jens a lot about the technical details about SAP HANA,

00:13:08 how ABAP met SAP HANA, and how we welcomed SAP HANA from an ABAP developer's perspective.

00:13:15 Then we have heard about the Code-to-Data paradigm, why it makes sense to delegate data-intensive calculations to the database,

00:13:24 and not just moving up all the data to the application server and do the application logic there.

00:13:30 We have met guys like CDS Views and ABAP Managed Database Procedures, and I've just shown you the tools

00:13:37 that you need to in order to maintain and create such objects in the ABAP Development Tools.

00:13:44 With this I would like to close the week and please stay tuned for the next week, when we will dive deep into the topics like, how can I optimize?

00:13:56 Where shall I optimize my coding? So we will talk about ABAP coding,

00:14:01 where shall I optimize, and we will start with the first unit of week 2 with existing code,

00:14:08 so after migration to HANA is my existing code still valid.

00:14:12 With these questions, stay tuned for the next week.

00:14:15 See you there. Bye.

www.sap.com