openSAP
ABAP Development for SAP HANA

WEEK 2, UNIT 1

00:00:13	Hello and welcome back to week 2, unit 1, Existing Code: Is It Still Valid?
00:00:20	I hope you did well with the weekly assignments.
00:00:23	Hi there. Before we start with unit 1, let me outline the second week of this openSAP course: ABAP Coding: Where to Optimize?
00:00:33	We will start with introducing you to tools to find and spot potential functional issues in your existing ABAP coding,
00:00:43	static code checks—we'll continue with that in the second unit when it is going to be about optimization potential—
00:00:50	and we will show you how you can see the full SQL profile of your running applications of your Application Server ABAP.
00:01:00	Finally, we will show you some reuse components that we have improved for perfectly running when running on SAP HANA.
00:01:10	That will be, for example, the ABAP List Viewer with Integrated Data Access (the ALV with Integrated Data Access) and some other improvements.
00:01:20	But let's jump to the slides and have a look at what migration to SAP HANA means
00:01:28	and especially what that means for us in this course—what we will be talking about in this and the upcoming weeks.
00:01:35	We will start with the Detect phase mentioned already: functional correctness and, of course, also performance optimization potentials.
00:01:44	We will then in the week 3 and the early week 4 talk about how to optimize existing ABAP coding
00:01:51	and how to explore the features of SAP HANA and really rethink and innovate your applications.
00:01:58	You will see the slides a couple of more times in the upcoming units.
00:02:04	Okay, so now let's go to the first step: migration to SAP HANA.
00:02:10	Migration to SAP HANA is nothing more than any other database migration.
00:02:18	There can be some functional issues when migrating from one database to another database, and this is exactly where this unit comes in
00:02:28	to show you tools to detect those potential functional issues.
00:02:35	If you have normal coding, basically it should work as before the migration, but as always, there are some exceptions
00:02:46	and this is when you used, for example, Native SQL coding with, for example, an EXEC SQL

	statement in the ABAP
00:02:56	and where you really relied on native database features and also used them in your coding.
00:03:04	Also there are some issues with implicit DB-specific behavior, which means that databases have, of course, algorithms in the coding
00:03:15	where they do their optimization when selecting and preparing the data as you wanted to have it
00:03:21	and if you don't specify, for example, an ORDER BY clause in your SQL statement,
00:03:27	you might get different ordered tables from one database or another one. But let's take a look at some code examples.
00:03:37	Yes, we have prepared some code snippets showing you exactly what we talked about, exceptions,
00:03:44	so to state it again, everything will work fine when migrating to HANA except for those exceptions that I will show you on the next slide.
00:03:55	And those exceptions are then, of course, where you have to adopt mandatory adoptions in your ABAP coding when migrating,
00:04:05	for example, from SQL Server to the SAP HANA database.
00:04:10	Jens has mentioned them already for example, the usage of native SQL that might or might not be a problem, really depending on the SQL statement you used.
00:04:19	If it is valid for any database, then of course, it will work also on SAP HANA.
00:04:25	However, for example here, in this example, we have MYSQL coding and this will certainly not work on SAP HANA.
00:04:37	The problem with Native SQL here in that case with ABAP Database Connectivity, the ADBC, that really then gives you a runtime exception
00:04:47	and those are, of course, are not that nice to have, especially in productive coding.
00:04:53	Another example here is a database-specific hint. That is not really a problem. It will just not be respected by SAP HANA.
00:05:04	The hint is there, you will get a warning, but it is not as problematic as a runtime exception.
00:05:12	Then the relying on undocumented behavior. Jens has mentioned that already.
00:05:18	SAP HANA uses another ordering algorithm when retrieving the data than, for example, an Oracle Database.
00:05:25	That is not really a problem. Let's say it like this: It's just undocumented behavior
00:05:30	and if you have relied on that implicit ordering, for example, of your SQL Server database or Oracle database,
00:05:40	you might get into trouble and it is especially problematic since you don't even get a runtime exception that will give you a hint that there is a problem.
00:05:50	But you might get some random ordering and you have relied on a specific ordering and that gives you nonsense for your application
00:05:58	and that might not even get noticed in the first place but only in your productive system.

00:06:04	Nonsense data is never a good idea in the business process. Not really.
00:06:10	So here for example, if I have a SELECT statement without a specific ORDER BY statement,
00:06:16	and later on use that data in a read table with binary search, that might get you into trouble.
00:06:25	Then, for example, direct access to physical pool or cluster tables. We have heard about depooling, declustering from Jens in the first week
00:06:35	and if you have checks for existence of secondary indices which you, for example, have only for other databases
00:06:43	and not for the SAP HANA, that also is a mandatory adoption that you have to do before you can migrate to SAP HANA.
00:06:51	Maybe one more last word so that you don't get into panic mode or something like that, it's just on Native SQL statements.
00:06:59	So open SQL statements will run on any database, since Open SQL is standard and database-independent.
00:07:07	Just to have you on the safe side here.
00:07:11	Okay, but even if you have such problems in the system, help is near.
00:07:16	Yes, and for that we have introduced new static code checks to really show you those code snippets that need mandatory adoptions
00:07:27	and we have also introduced a new code check tool which is not only a code check tool
00:07:35	but also helps you with quality assurance processes like, for example, a Q-Gate and so on.
00:07:41	It introduces verdicts with priorities for developers on code
00:07:47	and this is our all new ABAP Test Cockpit and this ABAP Test Cockpit is natively integrated in our ABAP Development Tools
00:07:56	but, of course, also available in the SE80.
00:08:00	Yes, and to mention this here on this slide, the availability of that tool is even before 7.4,
00:08:09	so you can check those things already before you migrate to SAP HANA.
00:08:15	But let's just have a look at that tool directly in the system.
00:08:20	So the ABAP Test Cockpit in its core has Code Inspector things,
00:08:26	so let's start with the Code Inspector, so the Code Inspector inspection methods.
00:08:31	Let's have a look with the shortcut. I'm going to the well-known Code Inspector.
00:08:38	In order to find functional issues in your coding, we have provided a global check variant.
00:08:48	Here I use the global and we have introduced FUNCTIONAL_DB, it is called.
00:08:56	So, let's have a look at this FUNCTIONAL_DB check variant, and you can see that we have a couple of tests running when working with this Code Inspector variant.
00:09:07	For example, security checks—I will just open that—like critical statements or here, for example, the usage of ADBC which we had on the slide,
00:09:17	then also the robust programming stuff like Search DB Operations in Pool/Cluster Tables.

00:09:27	Or, for example, the search for problematic statements, here without the ORDER BY statement.
00:09:35	So you see we have these checks. So in principle, you would now go and execute that or check that Z-coding in your system
00:09:46	or in your namespace with this FUNCTIONAL_DB variant.
00:09:52	However, we would like to not show that with the Code Inspector, but with the ABAP Test Cockpit and in its full integrated in the ABAP Development Tools and Eclipse.
00:10:02	For that, let me close that, I just copied the FUNCTIONAL_DB name here over. Close that, and go for the ABAP Project, right-click,
00:10:13	and select from the context menu the Properties. And you can see here the ABAP Development and here an entry for ABAP Test Cockpit.
00:10:22	You can of course, configure your system or the ABAP Test Cockpit in your system to run with a FUNCTIONAL_DB variant
00:10:33	but you can also switch to that in your specific Eclipse installation or the ADT installation.
00:10:41	So here that is the default of the system and I will change that with FUNCTIONAL_DB and Apply it.
00:10:48	And in order to showcase what these check tools are capable of, we have provided some bad guy doing all of the stupid things that you should not do.
00:11:00	So I have a report ready called ZSTATIC_FUNCTIONAL and I'm just...I don't tell you what the report does.
00:11:14	I'm just doing right-click on the report here in the Project Explorer and select Run As → ABAP Test Cockpit.
00:11:23	And I have incorporated the FUNCTIONAL_DB checks. Execute it and you will see that it contains ATC errors,
00:11:33	ABAP Test Cockpit errors, and I can see them in the Problems view. I can even navigate to them.
00:11:40	So, for example, DELETE ADJACENT DUPLICATES. Let's just jump there.
00:11:48	So I'm deleting adjacent duplicates but, well, there is no ordering applied here. Of course, this is very easy for me to find
00:11:58	because the SELECT is directly above. Let's have a look for example at the READ...BINARY SEARCH, the same holds here.
00:12:05	I have a SELECT again without an ORDER BY and I'm using then this internal table in a READ TABLE binary search statement.
00:12:15	Again not a good idea because I have forgotten this ORDER BY.
00:12:21	Okay and last but not least, a loop at the internal table and it says it's problematic and it says that this is problematic so...because I'm doing...
00:12:34	Yeah, your doing again the loop on the SELECT you have above, if you scroll just a little bit up...
00:12:40	Yes, thank you. You have here again the SELECT* with the DELETE ADJACENT

	DUPLICATES and afterward you do a LOOP
00:12:47	this is also a code part that you should check for. And here we have the AT NEW because the AT NEW carrid and you...
00:12:57	beforehand the carrid was sorted and now it may be not sorted by the carrid so you could have other step-ins of that loop at this position.
00:13:07	You just have to get rid of the AT NEW or add an ORDER BY clause.
00:13:12	Exactly. So it's very important to have a look at such code snippets
00:13:18	because they can really get you into trouble since you relied on the implicit ordering.
00:13:23	Okay, so much about the ABAP Test Cockpit and the integration in our ABAP Development Tools in Eclipse.
00:13:31	Exactly but there comes only one more question in the game...sorry, this was the wrong direction.
00:13:37	So, the question is if you have some, if you could add some runtime data to that static code check because that static code checks
00:13:46	for the functional issues might not find all problematic things and if you can have a report which runs on your productive system and checks, for example,
00:13:56	for empty SELECT...FOR ALL ENTRIES or unsecure ones, or find SQL statements that are just executed
00:14:04	without an ORDER BY clause, so you might have 100,000 of them, this report here will really help you.
00:14:12	You can run it on your productive system just for 1 or 2 weeks and the result will then help you to gain the right positions and have a secure migration phase.
00:14:24	Yes, so it really supplements the static code checks with runtime information and we will see that concept also coming in one of the next units.
00:14:35	Okay, then thank you for unit 1. Let's now take a look at what's upcoming.
00:14:43	So, week 2 unit 2, what about the performance of my coding?
00:14:48	See you for the next unit. Bye-bye.

WEEK 2, UNIT 2

00:00:13	Hi there and welcome back to week 2, unit 2.
00:00:16	Hi. In the last unit we've talked about the tools to detect potential functional issues when migrating to SAP HANA.
00:00:26	In this unit, we will tackle the question, What about the performance of your custom coding? Jasmin.
00:00:33	Thank you Jens. So let me show you where we are in the storyboard. We are still in the Detect phase.
00:00:39	So you would like to migrate to SAP HANA or you already migrated to SAP HANA
00:00:44	and you have already executed the functional correctness checks with the Code Inspector or the ABAP Test Cockpit as we have shown you in the last unit.
00:00:52	Today we will ask ourselves the question about performance and we will show you today the performance optimization potential checks that we have provided for you.
00:01:02	Okay, so it's all about performance and you typically ask the question, I have migrated to SAP HANA with my custom coding,
00:01:11	is everything running faster? And the answer is yes.
00:01:16	Well the answer is, that depends. It really depends on how much database-oriented coding you have,
00:01:24	so how much time you really executed of your programs on the database.
00:01:29	If there is only ABAP coding only running on the ABAP server without interacting with the database,
00:01:35	there is, of course, no potential for performance improvements with SAP HANA.
00:01:40	The good news is we have recommendations available for you so performance guidelines. Jens will talk about that in a minute.
00:01:47	And moreover, we have tools that help you to follow these guidelines.
00:01:52	Okay, Jens, please can you introduce the SQL performance rules guidelines for us?
00:01:58	Yes, of course. So these guidelines have been around already before SAP HANA.
00:02:04	So most of you, or some of you, have probably seen this, but let me explain in detail what they mean
00:02:10	and especially what they now mean when we have SAP HANA.
00:02:14	First of all, in the days before SAP HANA, all of these guidelines or rules had the same ranking and the same importance.
00:02:26	With SAP HANA, this has changed a little bit. So some are more important and some are less important.
00:02:32	Let's first of all take a look at the more important ones and they are more important because
00:02:39	it's based on the architecture and the layout of SAP HANA, as we have told you in week 1.

00:02:45	Okay, so let's take a look and give you some examples what we mean with those rules.
00:02:54	First one: Keep result sets small. So this basically means if you write a SELECT statement, you should only select the data which you want to display,
00:03:04	which means apply meaningful WHERE condition to the SQL statement. So, for example, if you only want one line, you don't select the whole table.
00:03:14	You shouldn't do that, of course, but it's a good example here.
00:03:18	Another one is minimize the amount of data transferred.
00:03:21	This basically means only bring that data to the application server that you really want to use in the application server
00:03:28	or later on a UI, program batch job, whatever. So how do you do this?
00:03:35	There's normally only the columns you need, so don't write a SELECT* statement. Just only if you really need all of the columns.
00:03:44	Instead of a SELECT * statement, if you think about the MARA table, there are some columns inside. You probably don't need them all on the UI at the same time.
00:03:54	And of course, minimize the number of data transfers. This basically means prefer array operations instead of single operations.
00:04:03	So, this can be once you do a SELECT of a sales order header table, then write a loop and then select, for example, the sales order item table.
00:04:13	This can basically be written in one SELECT statement, just without a loop, by joining those tables.
00:04:19	This will be handled later in week 3, when we show you the new Open SQL enhancements.
00:04:26	Another point here could be the INSERT statement, for example. So you do something on the bigger internal table
00:04:35	and you want to insert into the table. So what we are seeing is, you do that, do a loop, and do the INSERT statement in the loop by each line of this table.
00:04:46	This is much better if you do the INSERT after the loop. That's basically what we mean with an array operation.
00:04:52	Those are the ones that are very important when we talk about SAP HANA and the Code-to-Data paradigm.
00:05:01	What's not so important anymore is minimize search overhead, for example, or keep unnecessary load away from the database.
00:05:08	But always keep in mind unnecessary load.
00:05:13	So you can, of course, put load on the database if it makes sense. It's just a topic of unnecessary load.
00:05:20	So you shouldn't calculate something that you don't need or something like that.
00:05:25	Okay, so now enough of these guidelines. Let's take a look at the checks we have introduced to find potential code snippets. Jasmin.
00:05:38	Thank you Jens. I will show you that using the tools that we have introduced already in the last

	unit.
00:05:44	We have talked about the Code Inspector as well as the ABAP Test Cockpit
00:05:49	and we have shown you that with a Code Inspector variant called FUNCTIONAL_DB.
00:05:55	Let me just jump into the system. We've provided something very similar, but for doing the checks on the performance side.
00:06:08	Let me just execute the Code Inspector, open the transaction again, and enlarge that a bit.
00:06:15	In the last unit we had FUNCTIONAL_DB and now we have a Global Code Inspector variant called PERFORMANCE_DB.
00:06:24	What's in there? In principle, the checks that are checking exactly the things that Jens has just showed us to follow the guidelines.
00:06:33	Here you can find performance checks. Let me just open that for you.
00:06:36	For example, minimized result set so we have analyses of WHERE conditions, for SELECT statements, for example.
00:06:45	Then we have talked about the column storage in SAP HANA.
00:06:52	So the SELECT * statement, as Jens just mentioned, is problematic and it is especially problematic
00:07:02	if you later on don't use any of those columns that you have selected but only one or two of the columns.
00:07:09	And yes, maybe also mention the Search DB Operation in loops across modularization units. That is a new check we have introduced.
00:07:17	It is not new concerning the check itself. Jens has mentioned that you have a SELECT...END SELECT
00:07:26	so that is basically the loop and within this loop, you have a SELECT SINGLE, for example.
00:07:31	That might be problematic or that can be problematic.
00:07:35	We were not able to find such things if the SELECT...END SELECT was in another modularization unit as the SELECT SINGLE.
00:07:44	So for example, you have a method call and within this METHOD call you have a SELECT SINGLE.
00:07:50	If that method call is within the SELECT...END SELECT, you are not able to find that with old code inspection variants.
00:07:57	So now we are able to also have the SELECT...END SELECT and a method call or function call and within that again,
00:08:05	a function call whatsoever certain layers of modularization. We can also find that kind of problematic statements now.
00:08:16	But let me just take this PERFORMANCE_DB check variant and execute it on a bad guy we have written for you.
00:08:23	So typically you have your custom coding and you would like to check custom coding. For us, we will show that you in a demo report that we have specially provided for that purpose.

00:08:34	Okay, so let me just open the report and it is called ZSTATIC_PERFORMANCE
00:08:44	like the one we've seen yesterday for functional correctness. Let me just open that.
00:08:51	I will not tell you what the report is doing but I will just execute the ABAP Test Cockpit.
00:08:58	Yeah, but you know you have to change the check, you know this happens always to me.
00:09:03	Yeah, thanks for reminding me.
00:09:05	As we've stated in the last unit, you can specify the code inspection variant or the ABAP Test Cockpit variant in the properties for the ABAP project.
00:09:17	Just go to the properties, to the ABAP Test Cockpit, and here I interchange FUNCTIONAL_DB with PERFORMANCE_DB because we are now interested in performance.
00:09:27	I apply that, thanks for reminding me, and I will select from the context menu Run As → ABAP Test Cockpit.
00:09:38	And as stated last time we have done this, you see, in the Problems view,
00:09:44	some errors indicating what could be improved performance-wise.
00:09:51	So here, for example, the NonLocal Nested Reading DB Operations, let me go there.
00:09:57	You see that within a form, I have a SELECT SINGLE. The question is now, where do I execute this statement?
00:10:06	SELECT SINGLE here is not that good because I'm not doing anything with this anymore
00:10:12	but I'm concentrating on non-local nested reading DB operations. And you can see here that above, I have a LOOP statement
00:10:25	and within this loop, I have this PERFORM statement wherein, in another modularization unit here, I'm doing a SELECT SINGLE statement.
00:10:37	A similar thing is the SELECT statement that could be transformed, a problematic SELECT * statement,
00:10:45	so here you see I have SELECT * and it tells me that I have a certain percentage of fields unused.
00:10:51	Maybe it's not informative enough, so let me go to the ABAP Problem Help on that
00:10:59	and it tells me actually I'm using...0? Not so much at all. Yeah, none of the fields.
00:11:06	Of course, it doesn't make too much sense to select all the columns if I'm not using any later on.
00:11:13	Okay. With that let me go back to the slides.
00:11:18	So we have these static code checks, the static code checks and the performance checks ready.
00:11:26	So here is a typical distribution on how custom code is distributed in the system.
00:11:34	So there is some unused code. You can remove that with the UPL and the rest can be checked by the static code checks.
00:11:44	So except for dynamic code, I could in principle now spot all my performance-critical things and get ready to do the optimizations.

00:11:57	Yeah, but of course, if you think now about 100,000 errors in your custom code, this is not maybe not the best idea
00:12:06	to just start optimizing because it would take a while and maybe you wouldn't benefit from the first 90,000, or 99,000, and only the last 10 would help you.
00:12:16	So it's a good idea to add runtime data here. And now you might you say okay, let me use the SAT or the ABAP Profiler to find it.
00:12:27	But those would be only stitches in all of this code parts and not give you a full profile of your server and business,
00:12:36	of your code, the full SQL profile.
00:12:41	That's exactly where the next session comes in
00:12:44	where we will take a look at what we built for you to give you a full footprint of your productive system from the SQL side.
00:12:55	Stay tuned and see you. Bye, Bye.

WEEK 2, UNIT 3

00:00:13	Hello and welcome to week 2, unit 3.
00:00:17	Hi there. Today we will be talking about the SQL profile of your productive system.
00:00:23	Let me go one step back and tell you what we talked about in the last two units. We have shown you static code checks.
00:00:31	First, to do functional correctness checks when you migrate to SAP HANA with your coding. And the second thing is what we told you about
00:00:40	our static code checks again, but this time on performance things. So where should I optimize using static code checks?
00:00:49	However, we also said that it is not a good idea to then just start and do all of the adoptions.
00:00:55	Certainly for the functional things, but not yet for the performance things.
00:01:01	Let me remind you of the pie chart diagram that we have shown you yesterday. Why we said it's not a good idea.
00:01:09	We said the static code checks cover all of our coding; however, it might not be performance-relevant.
00:01:16	So the adoption of a code snippet would just run in an application server lifetime maybe once.
00:01:24	It's not really worth optimizing the performance of that code snippet.
00:01:29	We have said that it is beneficial to include runtime information and to combine this information
00:01:36	and we have also talked about the SAT, but that will only give you stitches, so only a couple of code snippets where you should optimize.
00:01:46	But you still don't know what is the most important thing I should optimize first.
00:01:51	For that, we will show you a tool today that you can really get the complete SQL profile of your productive system.
00:01:59	I will also show you why you can do that in the productive system without overhead killing your business processes at the same time.
00:02:09	Here we have written down some numbers to give you a feel of what is really happening on a normal ERP system.
00:02:17	So here in this example with 6,000 concurrent users, you really see there's much is going here.
00:02:26	I was also impressed when I got the number for the first time.
00:02:30	Just think about it. 140 billion records collected here during two weeks' execution time.
00:02:40	This means 10 billion records read or changed every day and this by around 1 billion SQL statements.
00:02:48	So, quite an amount of data which is created here and we are all humans. We cannot
00:02:58	gather such information together just in our brains, so we really need help here to know where to optimize.

00:03:04	And that's exactly where our new tooling will step in
00:03:07	and Jasmin, can you tell us a little bit about the new transaction?
00:03:11	Yes, thanks Jens. So you've seen these amazing numbers and therefore we have come up with the SQL Monitor.
00:03:20	What that tool is doing is giving you the SQL profile in your productive system
00:03:28	and with that, it will help you to find the performance optimization potential.
00:03:33	So really, the most important SQLs that you might be interested in or you are certainly interested in optimizing are worth looking at in order to optimize.
00:03:43	Here is the availability written down and there are even more slides once you get the slide deck on your own, where you can the availability matrix of the tooling.
00:03:53	Jens will also show you in a demo in just a second.
00:03:56	More information can even be found in this SAP Note or in this very interesting and very nicely written SCN blog
00:04:05	by a development colleague of ours: The SQL Monitor Unleashed.
00:04:11	Before we come to the demo, let me show you a bit of architecture.
00:04:15	I promised to tell you why it is okay to run that in a productive system. You might say, Oh that is killing my business processes I would like to do my business.
00:04:25	And of course, we are aware of that and you can run the SQL Monitor without too much overhead
00:04:32	and that is because you are executing your business processes, the database interface is working for you interacting with the database,
00:04:40	and especially with the application tables. And at the same time, you have an asynchronous job running and the important thing is asynchronous.
00:04:49	The asynchronous job is gathering the information for the runtime monitor, gathering the data, and a batch job is writing that to the SQL Monitor tables.
00:04:57	And with that, we have only a minimal overhead and you can run that in your productive system. Jens will also tell you how you then get the information out of your productive system,
00:05:07	to the Q system or development system because you also don't want to do the analysis in the productive system.
00:05:13	Okay, so Jens please show us a demo of that tool.
00:05:16	Let's just step in, so for that I switch to the ABAP Development Tools.
00:05:21	I go to our development system and let me just open it. The transaction is SQLM.
00:05:28	I open the transaction and then you see an overview of this,
00:05:35	Oh nope, this is the SE93, I'm sorry. So this happens occasionally to me. Let me switch to the real transaction here,
00:05:47	/n, yes, okay. This happens to all of us, let's be honest. And finally here you see the SQL Monitor.

00:06:00	What do we have here? We have some administrative information up here on how much data is gathered,
00:06:07	when it was last executed, by whom, and if it's at the moment active or not and on how many servers.
00:06:13	It brings me to those buttons where you can activate it for all application servers or maybe only for special application servers.
00:06:21	Yeah, maybe it is good to mention here, if you would click now on the Activation button, it also gives you a stopping time.
00:06:29	So we typically advise to run the SQL Monitor in a period of about two weeks, like the number have seen for the ERP system in the example,
00:06:38	because after two weeks you don't have any—or not too many—changes in the coding, so you really can analyze the processes
00:06:46	and two weeks is also a measuring frame where you really gather information.
00:06:53	Exactly. However, and I will state that later on and again we also touched it already, you have to iteratively repeat this
00:07:02	because maybe in this two weeks of time frame, you don't have certain important application run.
00:07:10	Yeah, quarter-end run or year-end run. SSomething like that. So think about collecting all business processes
00:07:18	and now let's take a look at the data. That we can do in the next tab here.
00:07:23	That's also what Jasmin has mentioned. You probably don't want to analyze it—or yes, you can analyze it in your productive system—
00:07:29	but you can also get it out of this system and import it into another one. So, what we now do is we want to take a look at it.
00:07:37	Display it. Then you come to a classical selection screen. You can filter by different criteria.
00:07:45	Let's just take a look at our Z-coding and we have prepared here, of course, something for you, so Jasmin tends to call those reports, the "bad guys".
00:07:55	There are some really bad guys in here and what you can do here now is very important. So you want to aggregate the results probably.
00:08:04	You've have seen the numbers and they are quite high also probably in your system, so an aggregation is a pretty good idea.
00:08:12	If you click here Aggregation By Request, you will get the data aggregated by the business processes.
00:08:19	So, you see the business processes which hurt you a lot from the performance aspect
00:08:25	and you can also give it an initial sorting like, for example, we now say the total amount of DB execution time.
00:08:33	So and finally, ta da! Here it is: the SQL Monitor data.
00:08:40	Many numbers, so it takes...you need to get used to it and that's the important thing, get you a system and try it out on yourself to know what the numbers mean.

00:08:51	I will give you just some brief examples and what we get here is the total DB time of this process
00:09:00	and which process is it, it's on the right side here. So that's the process which is called. If you analyze a classical business system without any filter,
00:09:09	it will be probably VA01 or something like this will appear here.
00:09:14	And here you also see the DB time in percentage to the amount of the total time,
00:09:20	which gives you a good feeling that here much DB time is consumed and an optimization of the SQL is here really possible.
00:09:32	So if you only have 1% of DB time optimization of the SQL, wouldn't help so much at all.
00:09:39	So, important to know, here you have the processes and now you can step down to the statement level.
00:09:48	In this screen now, you see the statements which have been executed in this business process
00:09:56	and the good thing is we have the information on how much every statement inside this process consumes the DB time of these processes.
00:10:06	That is the percentage right?
00:10:08	Yes, this is in percentages. The time is always milliseconds, by the way.
00:10:14	So this gives you a clear picture here that the first one is probably the one you are going for.
00:10:22	The other ones are not of interest at this time, since we have 97%. And this is a SQL statement
00:10:30	and it's here inside this report. By clicking it, you can directly jump to the source code.
00:10:37	Here you see you have a loop, SELECT SINGLE, which is, if you remember the golden rules, probably better if you prefer the array operations
00:10:48	and maybe you can even think about if the SELECT* statement is the correct one.
00:10:52	Okay, and just by removing it and writing it for example, maybe with the SELECT...FOR ENTRIES, could be a good idea here.
00:11:03	Just shortly one step back again, statement level.
00:11:07	You also have the internal sessions, which tells you how much this process, how often this process, has been executed during the time frame.
00:11:18	So the one with the highest numbers are probably those ones which are executed the most. Not probably. They are the ones that are executed the most.
00:11:27	Okay, so as I stated, just take a look at it, play around, take a look at it, and get a feeling for it.
00:11:38	Can we even drill down there too, just to see what's in there?
00:11:41	Drill down in that one? Yeah, let's take a look. So this is another one, and again dominating one with the percentage
00:11:49	and this time it's an INSERT statement. If we take a look at the numbers and figures here, it's probably also again a SELECT loop.
00:12:00	Okay, it's a DO now, but yeah, but very good coding. How we really expect you to write coding.

00:12:07	No just kidding. Again. prefer array operations so collect this data inside an internal table and do the INSERT after the DO.
00:12:19	But of course those are our bad guys. I think just give you a feeling, try it out.
00:12:27	Go to the blog we have mentioned. It's a very good blog to explain to you what you can analyze with the SQL Monitor.
00:12:34	So this is one of our most important tools. Just to emphasize that. Okay, Jasmin.
00:12:40	Yeah, thank you Jens and even in the blog you can find so miraculously, Jens showed you which ordering you need again,
00:12:50	also which aggregation he took. In the blog you will find in the references.
00:12:57	You will find more information on how you should use these filters in giving you exactly the things that you would like to see.
00:13:05	Okay, so back to our pie chart diagram. You've seen it now several times and we now have a full coverage of our whole coding,
00:13:16	if we removed, of course, the unused code first. We had the statical code checks,
00:13:22	not measuring the dynamic code, of course, and we can supplement the information now with the SQL Monitor Data in order to get the full coverage.
00:13:30	But as mentioned before, this only holds true if you really run it when you are executing also all of the statements or all of the processes.
00:13:42	So for year-end reporting, monthly reporting, you may be need to repeat the measurement of the SQL Monitor.
00:13:50	Okay, so with that I would like to say thank you and hand over for Jens to give you an outlook.
00:13:56	Okay, so now you have the system data, the live data of your SQLs. Now you know how to analyze the coding
00:14:06	from the static side with the code checks and now you may ask the question, Wouldn't it be a good idea to combine the live data
00:14:13	also with the information that I got from the static code check? That's exactly what we will show you in the next unit.
00:14:22	So stay tuned and bye-bye. See you.

WEEK 2, UNIT 4

00:00:13 Welcome! Great to have you back for the fourth unit of the second week.

00:00:17 Welcome also from my side.

00:00:19 In the last unit, we've shown you how to gather the SQL profile of your productive system and analyze it using the SQL Monitor transaction.

00:00:29 Now, wouldn't it be a good idea to take this data and combine it with the static code checks

00:00:37 that you have learned in unit 2 with the DB_PERFORMANCE variant and compile it together in one worklist to know also

00:00:47 that this is a loop across a modularization unit, yet just to enrich the SQL Monitor data also with the static code checks

00:00:56 to make it much easier for you to have an entry point here.

00:01:02 Of course, we have this transaction prepared for you and I guess Jasmin can show you how this works.

00:01:09 Thank you Jens. So, the guy we are talking about is the SQL Performance Tuning Worklist and the transaction ID is SWLT.

00:01:19 So let me get into the system and do Alt+F8 and go for SWLT and again a selection screen opens up.

00:01:29 Let me fill that with some information. You've seen something very similar for the SQL Monitor,

00:01:36 so again I would like to restrict on package level. That, in your case, might also be a Z-coding or, for example, a namespace given to your company.

00:01:46 I also selected I would like to Show Intersection of Results. I will not go into the details, but that will help me to enrich the information here.

00:01:56 This is just for demo purposes. Jens has mentioned that we would like to include static information, static code check information to the SQL Monitor.

00:02:06 I will start with the static checks. We have mentioned it should be the PERFORMANCE_DB variant run.

00:02:14 I say include it please, so I will use it and you can choose between the Code Inspector

00:02:20 or the ABAP Test Cockpit. Because we would like to promote of course, the new tool, let me just take the new tool and I have already prepared an ATC run for you.

00:02:31 I already executed that on the system using the code inspection variant PERFORMANCE_DB.

00:02:37 So, I'm just selecting it, a developer has done that for me. You don't see the word developer, it's called Demo Background Job (DEMO_BG).

00:02:47 Okay furthermore, I'm scrolling down. You can do some aggregation and by default it's by code position, source code position, and message type.

00:02:58 I rather prefer the only By Code Position, but that really doesn't make too much difference.

00:03:06 Okay. In addition to the static checks, I would like to include the SQL Monitor data

00:03:14 and typically you would now go to your productive system, export the data there, and import

	into the quality or development system the SQL Monitor data.
00:03:24	We only have this one system available so we are doing both things at the same time and so I say again, I would like to use it.
00:03:34	That activates it and of course, again, I have prepared a snapshot for you.
00:03:39	Which is the same that I've shown you in the last unit. Exactly.
00:03:45	So even you could create a snapshot here but we have seen how to do that in the previous unit, so we will just select the snapshot we have already taken.
00:03:55	Again, call the demo, and here again I have the option to filter
00:04:05	or to do the ordering, and again I would like to have a total DB execution time as only top records here.
00:04:13	Okay and with that information given, I can say, just open the information.
00:04:22	Now you see again a lot of columns, a lot of information which might be the first time you execute it.
00:04:30	It really hit me and I was completely lost with all of those columns and all those information.
00:04:34	What the heck are they doing here? What do they want from me?
00:04:38	Yes, so it's really an expert tool, the same thing as with the SQLM as you've seen it, but you get used to that.
00:04:46	And really try it out and also have a look...again promoting the SCN blog from our development colleague telling you what you can do with that tool.
00:04:56	So I would only briefly mention a couple of examples, also the examples that we have already seen.
00:05:02	First thing, which is now the difference, or one of the differences, between the SWLT and the SQL Monitor.
00:05:11	I can have, when I double-click on one of these lines, a drilldown. That is on SQL level, on statement level, and you might ask, Where did I enter?
00:05:21	So here is the information about entry points, request entry points, taken from the SQL Monitor.
00:05:28	Moreover, here on the lower right-hand side you have the additional information from the static checks, so from the Code Inspector or from the ATC in that case.
00:05:38	You have met already TEST3, the very meaningful operations we do there, and here for example,
00:05:47	I am told that there is a NonLocal Nested Reading DB.
00:05:51	Of course, we could have gathered this information already from the static code checks alone,
00:05:56	but here we also see that is the top-most operation executed from of course within my Z-packages.
00:06:05	So that really took the most time and it's certainly good to optimize here first because compared to all of the other DB executions, all others are rather low.
00:06:15	So really a prioritized list where I shall optimize first.

00:06:20 Okay. Now I can have a drilldown. That is the additional information now, not only the SQL Monitor but again the SQL,

00:06:30 the static code check information.

00:06:32 And here I have additional information given, so for example, it tells me where are the entry points.

00:06:40 So here is the SELECT statement. I can go there again, you have seen that before. That is the SELECT SINGLE statement, which was bad because it was in a loop.

00:06:48 And of course, I see the loop right above that, but it could be in another modularization unit

00:06:55 and this information—so where the actual call of that method is which is enclosing the loop and the SELECT—can also be navigated from here.

00:07:08 So here you see I have a DO five times again, a DO statement which is not very good to do.

00:07:15 With awesome coding! Yes, awesome coding. So I can navigate in here. That is outermost point.

00:07:23 I navigate in and you see then within this method, I have this loop...endloop and SELECT SINGLE within there.

00:07:31 Okay. Back to the SWLT information. So again our TEST11 example.

00:07:41 I could do a drilldown here with the beloved INSERT statement, without the array operation.

00:07:51 So really a bad guy of course constructed. No.

00:07:58 Okay, so what you can also read from that list here, for example, is this guy here. I see the mean record is quite high.

00:08:09 I do a double-click and you see the Code Inspector hints to me that there might be unsecure use of FOR ALL ENTRIES.

00:08:17 We've also mentioned that before, so in addition to the runtime monitor data, we have here the unsecure FOR ALL ENTRIES

00:08:28 and it might be worthwhile to have a look at that, because maybe the FOR ALL ENTRIES table here is empty.

00:08:39 We have mentioned that in the first unit about Runtime Check Monitor already but here just to see all of that, will again...

00:08:51 It was exactly the same code snippet. Exactly.

00:08:54 So maybe a word of precaution before we end the demo.

00:08:58 You see we have the top-most things that might have potential to optimize

00:09:05 but as I stated before, you need to have a cutoff when you stop the optimization.

00:09:11 You will always have the top 10 or the top 20 things that you would like to investigate.

00:09:17 So you have the top 10 here, you investigate them, you adopt them, you do something for performance, and then

00:09:24 that was done, for example, in this example. You have the Optimize report.

00:09:29 But after certain iterations, the optimized cases will also be in that list.

00:09:34	So think what you would like to achieve before you start the project.
00:09:39	Otherwise you will go into an infinity loop with that.
00:09:45	Okay, with that I would like to hand back to Jens.
00:09:48	Yeah, back to the presentation. Thank you Jasmin.
00:09:51	So, that is what you've just seen, working with the prioritized worklist.
00:09:57	Now I just want to recap what we did in this Detect phase again.
00:10:03	When you want to migrate to SAP HANA, first of all again, important and mandatory step is to find the functional issues that could occur.
00:10:14	You've learned that in unit 1. Afterward, it's a very good idea to scan your productive system with the new tool, the SQL Monitor,
00:10:23	to get an overview about your SQL processes mentioned.
00:10:27	This is also very important because this was yet not available in the classical world
00:10:32	and now you can really do an analysis on that even before you do the migration part.
00:10:39	So information about in which release the SQL Monitor is available is inside this course material and there is also consulting support by SAP for that.
00:10:49	All information again is in the material. After you've done the analysis, take the most critical performance point
00:10:57	and of course, correct the functional issues because you want to have a running system after the migration and this is exactly then where you can migrate
00:11:06	and start this iterative performance improvement approach. And as Jasmin stated, use common sense there.
00:11:15	Don't optimize a report which takes just 0.5 milliseconds on the database.
00:11:23	It makes no sense at all and just burns your resources. And then just continue until you have the performance you like.
00:11:31	Okay, that's for the Detect phase. Let's take a look at the next unit.
00:11:37	Thank you Jens. So next unit we will be talking about quick wins. Of course, we have used also these tools ourselves
00:11:47	for our SAP Business Suite and also for our SAP internal custom coding.
00:11:55	And we have also had a look at all our components that we previously had.
00:12:02	We will show you in the next unit quick wins. For example, the optimized ABAP List Viewer, the artist also known as ALV with Integrated Data Access.
00:12:12	And we will also show you some other improvements. Stay tuned.
00:12:16	Thank you and bye-bye.

00:00:13	Hi and welcome back to the last unit of this week.
00:00:17	Hope you are ready for the quick wins.
00:00:19	We will show you things like the optimized ALV or the artist also known as ALV with Integrated Data Access,
00:00:27	so a renewed ABAP List Viewer, and some other improvements. Jens, please start.
00:00:32	Here we will start with the transparent optimizations and as you can see on the slides...What? The slide is empty.
00:00:43	That is the transparent optimization, that's why the slide is empty. Just kidding a bit.
00:00:48	Of course, we have shown you that diagram before so we would like to remind you of the transparent optimizations.
00:00:55	Remember, we have told you in the first week, the ABAP 7.4 comes with transparent optimization like table buffer enhancement
00:01:04	in the database interface and also the protocols that the ABAP uses to talk to SAP HANA, so with the underlying database.
00:01:14	So these are transparent optimization. You don't have to do anything concerning coding, but you will hopefully see that in performance.
00:01:22	Okay. Let us now come to the real reuse components as is the title already, the optimized ALV
00:01:31	and instead of talking about the theory of that renewed ABAP List Viewer, let me just go to the system and convince you with a system demo.
00:01:44	For that, I go again to my ABAP Development Tools in Eclipse and I have prepared a tiny little report
00:01:52	called ZR_ALV_IDA, standing for Integrated Data Access, and as you can see there is nothing in there yet
00:02:01	As you don't want to see me typing and doing all of the typos, I have created a small template for you.
00:02:08	So the new ABAP List Viewer, or the renewed ABAP List Reviewer, can be taken from that class.
00:02:17	I'm just calling a create statement and that guy here could either be a Data Dictionary table or a Dictionary view
00:02:27	or those new guys that we will hear about in the next week, something like Core Data Services views.
00:02:34	But stay tuned for that in the third week in the third unit.
00:02:37	Okay, what the ABAP List Viewer now comes with is full-screen mode
00:02:47	so you don't have to use or create yourself dynpros and I would like to display the data at the end.
00:02:55	Okay. With that, we are already ready. And just execute the report.

00:03:03	Let me enlarge that a bit and what you can see here is an ALV table
00:03:11	and there is a large amount of data in there. We have talked about that already and typically what happens if you're selecting the data in the ALV,
00:03:22	you are selecting everything in the internal table. So already this launching takes quite a while with the old ALV.
00:03:28	Same holds if I now just scroll down here. You see that is pretty responsive and that is because whatever I do here,
00:03:38	like scrolling or let's say I would like to use here on the gross amount, I would like to do a total,
00:03:47	I sum that one up. Everything that I am doing here is producing a SQL—you don't see it—
00:03:54	issued the SAP HANA database. And it's that responsive because HANA supports me with all of the things that I'm doing here.
00:04:04	So for example, let me also do an aggregation on that one,
00:04:10	or I could filter on that, on the Company Name, or just do a Group By for that customer name.
00:04:22	So, pretty responsive isn't it?
00:04:24	Yes, it's cool. The good thing here is that you really see the power of SQL and what it does with the Code Pushdown paradigm.
00:04:33	So all calculations are executed on the database using just SQL statements.
00:04:39	And the even better thing is, you have not seen me typing any coding. That's because the ALV with integrated data access is producing those SQLs for me.
00:04:50	Okay, enough of the demo. Let me show you a little bit about the techniques running behind the scene.
00:04:58	So as you can see, we have optimizations because the result set is minimized.
00:05:04	So you have seen this in performance, but I can also show you that in this little diagram. So what typically happens with the classical ALV,
00:05:12	you have your database table, which is here the SAP HANA table on the database, and the classical ALV is getting everything into an internal table
00:05:22	and whatever you would like to display later on is not filtered before it comes from the database to the application server, but on the application server itself,
00:05:31	so on the level of the internal table or in the ABAP layer.
00:05:36	The new ALV or the new approach is that you only fetch the data into the application layer that you really need.
00:05:44	And that is exactly applying the Code-to-Data paradigm, as Jens has just mentioned.
00:05:49	So we see improved performance. We see a reduced memory footprint and there is no truncation on selected data.
00:05:57	If you would like to know more about that, just go the SCN link and yeah, so that was our ALV with Integrated Data Access. Is there more, Jens?
00:06:06	Yes, there is more and I really like this demo. I love to work with the new ALV. This is really fun and it's cool, that's all.

00:06:16	Cool is the right word here. Okay, let's go to the next one.
00:06:20	So we've also improved our search help with type-ahead function and free-text search
00:06:29	and also let's just jump directly into a demo. I will close down the ALV and...
00:06:37	Oh we are not closing down the ALV. No? Just the report. Yes, okay. It's still there for you, don't panic.
00:06:44	I've also prepared a little report already but with a little more of coding and this is called ZR...
00:06:54	Okay, again I forgot how I named my own report but luckily we have it in our package
00:07:04	and of course it's SEARCH_HELP_DEMO. Sorry for that. And this report, let me just execute it.
00:07:11	You have to activate it first. Oh, maybe I changed something already. Okay, now you know also it's Ctrl+F3,
00:07:19	you know this probably from the SAP GUI ,and now F8 works. And what I now have here is a simple search field where I can type things.
00:07:30	Normally, I would go and now click on Identifier and take, for example, or know the identifier of SAP which is an our case
00:07:38	one and very, very...what you now have is exactly you already have some information and type-ahead information of an identifier
00:07:49	and here we see the business partners and their identifiers and also the e-mail address.
00:07:54	So let's just assume you totally forgot that very short number of 1,000,001 or something like that or you don't know...
00:08:03	I remember that contact I had, what was her name? Sarah. So let's try it.
00:08:11	And you see here, ah here's Sarah. It was this company and here I got the identifier.
00:08:17	So this is a very cool thing and it really helps you working and getting the results very fast
00:08:22	and the good thing is you don't have code so much for that.
00:08:26	It's really easy to get that and the only thing you have to create is a search help or just improve your existing search help.
00:08:35	So in this case, we are using that search help and let me increase the size of it.
00:08:40	The new thing that you see here is the Enhanced Options. Here you can select that you want the proposals for the input fields
00:08:48	and the full-text search and you can specify an accuracy level which is, kind of, how fuzzy it can get.
00:08:58	That is basically all. You have the fields and they are good to go and activated.
00:09:06	That's all. The other good news is that this one is also available for the Web Dynpro with the Floorplan Manager.
00:09:16	If I now come to an example, we also have in your demo system, is that you have also the search helps available here, also with the new ALV grid.
00:09:25	You can play around with it and here it is the same game.

00:09:29	So remember our Sarah? Here we go, here is Sarah. Is it? Yeah.
00:09:38	So that's for type-ahead and search help. I hope you like it.
00:09:44	Back to the slides. Here is also a link in the material to the demo on your remote desktop
00:09:52	and yeah, that what type-ahead is all in. So you have the filter and search features
00:09:59	and the good thing is we also deliver our new SAP GUI fields and so on with that type-ahead feature, so for example when you go to the SE38—
00:10:07	I still hope you will like the development tools more, of course, but there may be some of you which stick to their classical SE38. Please come to the good side.
00:10:18	There is also search help available now. Okay, and more information, of course, in our SCN space.
00:10:25	Links are in the course material. Good.
00:10:29	Yeah, maybe let me add, you can even include the fault-tolerant search in the ALV,
00:10:34	not with the full screen mode and the display mode I've shown you— they have to do a little bit more of the typing not just one line of coding—
00:10:45	but you can have examples around in the ALV demo reports in the system.
00:10:52	Okay, and with that we would like to close the week. So you have learned a lot about performance analysis tools,
00:11:02	the static code checks, starting of course with the functional correctness checks, but then we went to performance,
00:11:09	all about performance. The statical as well as the runtime analysis. You've learned about the SQL Monitor and you have seen about the SWLT,
00:11:18	SWLT is the SQL Performance Tuning Worklist, which included the information of the static code check and the SQL Monitor runtime data
00:11:26	on the SQL profile of your productive system.
00:11:29	And last but not least, I hope you liked the quick wins that you can get with what we have improved.
00:11:35	The transparent optimizations, the ALV, and also this very cool new feature of type-ahead and the fault-tolerant search
00:11:44	in case you are doing a lot of typos when you are using the product.
00:11:47	And with that, we wish you luck for the week 2 assignment now at this time and see you next week.
00:11:56	Yes, see you next week when it's all about optimization and what we have implemented in our ABAP server.
00:12:02	Bye-bye.

www.sap.com