



SAP HANA

Lesson Name: Taking ABAP to SAP
HANA

Lesson Objectives

After completing this lesson, participants will be able to -

- Know about the basics of SAP ABAP and SAP HANA
- Optimization techniques for SAP HANA Database
- How to access SAP HANA through ABAP applications
- Basics for migrating ABAP code to SAP HANA
- Different performance and functional checks
- About different performance analyzer tools





Introduction

Optimizations for the SAP HANA Database

Planned Support for Optimization

Providing Access to SAP HANA Through ABAP-Based Applications

Enabling ABAP to Run on SAP HANA

Optimizing ABAP for SAP HANA

The New Enhancement Package for SAP Net Weaver AS ABAP

Migration of ABAP code to SAP HANA

Brief over view of different performance analyzer tools



ABAP is integral to SAP customer and partner environments.

A vast amount of business-critical data is present in ABAP-based systems, including SAP Business Suite, SAP NetWeaver Business Warehouse (SAP NetWeaver BW), and on-demand offerings from SAP.

In addition, these environments stand to reap significant rewards from the accelerated processing and analysis supported by SAP's in-memory database technology offering, SAP HANA.



The SAP HANA platform combines in-memory software with hardware from leading SAP partners.

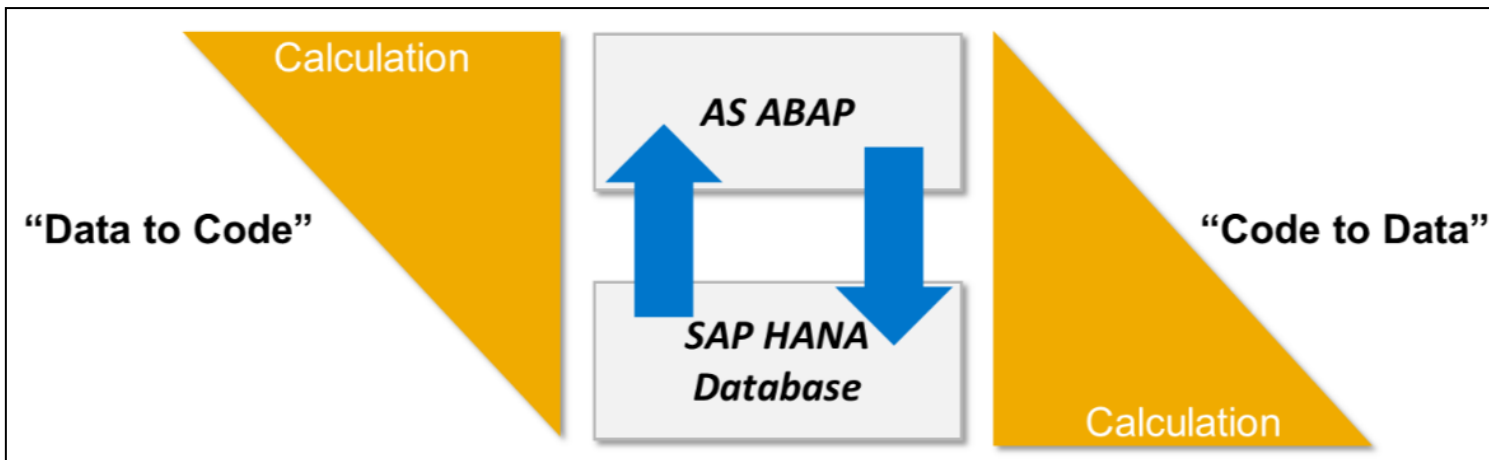
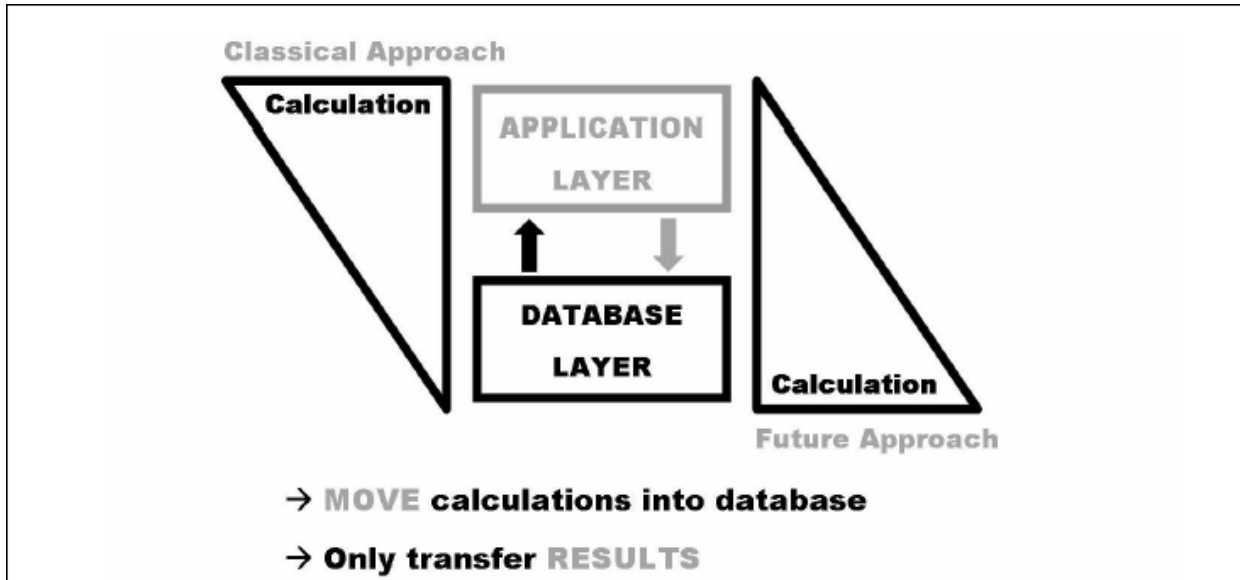
Adding SAP HANA technology to certified database hardware enables not only significant acceleration of existing applications, but also the development of completely new applications that were not previously possible.



To leverage the strengths of SAP HANA, applications follow the “code to data” paradigm in which calculation logic is pushed down from the application server to the database server.

The SAP HANA database then performs the calculations and sends the resulting data set back for use by an application.

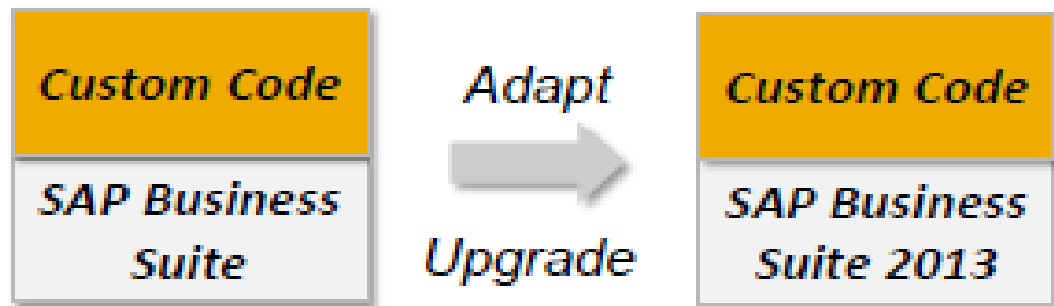
Introduction



Introduction



Custom code is a central part of the business functionality



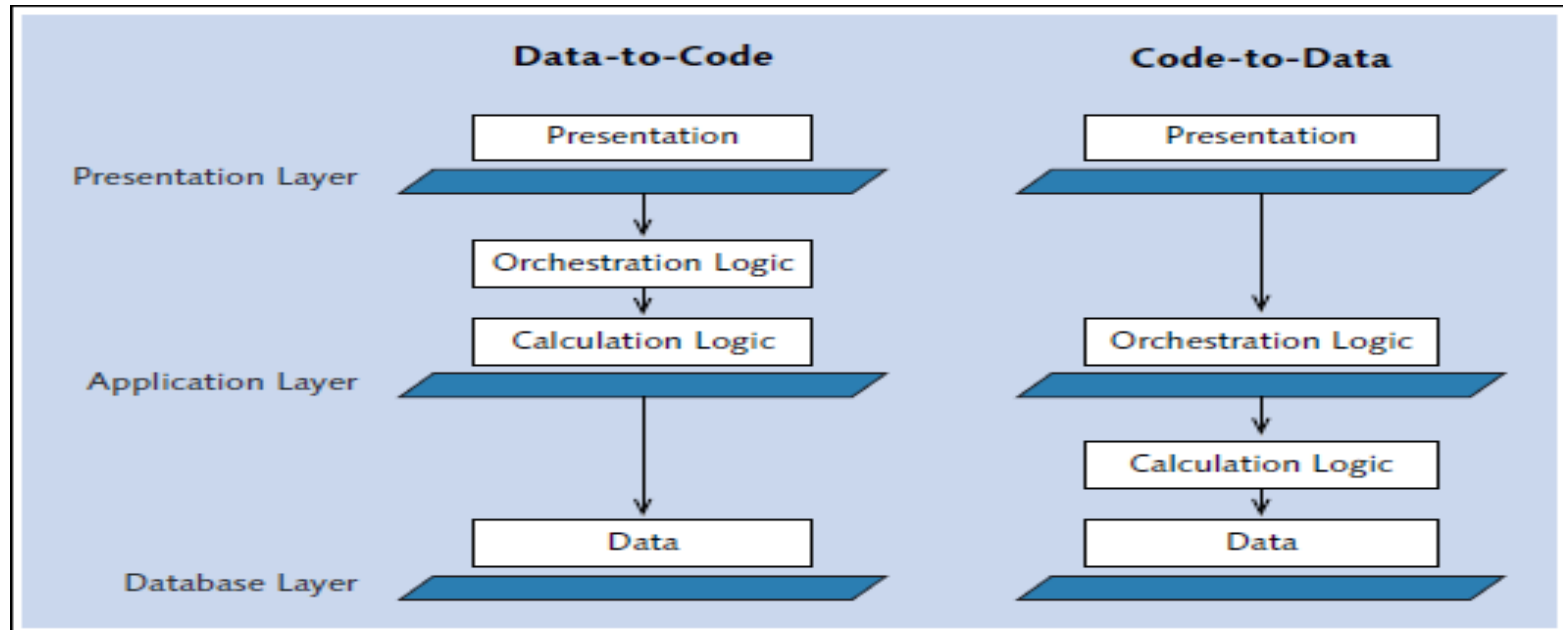


Applications that use SAP HANA follow the “code to data” paradigm. Operations on top of large data sets can benefit from in-memory database technology. With the amount of ABAP-based data processing and analysis in SAP customer landscapes, adding SAP HANA capabilities to SAP Net Weaver AS ABAP is a natural fit.

To take advantage of in-memory technology with SAP Net Weaver AS ABAP, operations that require high-performance access to mass data can be delegated to SAP HANA, while operations used to manage and integrate business processes remain on SAP Net Weaver AS ABAP.

The process of moving application code from the application layer to the database layer is often referred to as code pushdown. Traditionally, ABAP-based applications use the so-called data-to-code paradigm. Applications optimized or developed specifically for SAP HANA, however, use the code-to-data paradigm.

When using the code-to-data paradigm, the application data is also placed in the database layer. However, some of the application logic is executed in the application layer, while some of it is implemented in the database layer. In an extreme case, the entire application logic can be executed in the database layer. Nothing fundamentally changes in the execution of the presentation logic.



The application logic is subdivided into two sections:

- The orchestration logic controls business processes and the data flow and determines how calculation results are combined and further processed.
- The calculation logic identifies algorithms used to perform calculations based on the application data.

When applying this paradigm to an ABAP program, this means:

- The data of a code-to-data application is stored in the database.
- The orchestration logic is implemented on the application server.
- The calculation logic is usually executed in the database.



Operations on large data sets are ideal candidates for use with SAP .

Let's look at SAP's staged, three-step approach to optimizing SAP NetWeaver AS ABAP for SAP HANA and how it will help customers and partners gradually integrate in-memory innovation without risking their existing implementations.

Optimizations for the SAP HANA Database



With the new enhancement package for SAP NetWeaver AS ABAP, customers and partners will be able to use SAP HANA artifacts and capabilities directly in ABAP with an integrated development and lifecycle management experience.

As a first step, activities that naturally belong to the database will be transferred to SAP HANA -- for example, queries with joins across tables will be performed in the SAP HANA database instead of using internal tables in SAP NetWeaver AS ABAP.



Migrating to SAP HANA

When performing a migration to SAP HANA, you want to make sure that all programs continue to run as before. Moreover, you might want to identify optimization potential with regard to database access before or during the migration, and implement the necessary adjustments. These tasks are mainly the responsibility of ABAP developers and quality managers for ABAP programs. In addition, it may be necessary that process owners work with these employees to prioritize possible performance optimizations based on the importance of the respective business process. When migrating to SAP HANA, the following steps are necessary to analyze and possibly modify or optimize ABAP code:

- Collect respective data information
- Analyze ABAP code (in combination with the collected data)
- Prioritize the applications identified as relevant for optimization
- Adjust the programs accordingly



System Optimization

System optimization considers the system as a whole. Its focus is highly technical and the required steps are usually performed by SAP system and database administrators. When dealing with applications that system and database administrators cannot optimize directly, ABAP developers will also be involved in the optimization measures. System optimization has priority if a large number of system processes are too slow and the runtime problems cannot be narrowed down to one or a few applications.

There are two possible approaches for system optimization: Analysis of system settings and hardware resources on the one hand, and application and SQL analysis on the other. In this context, it must be noted that the two subject areas are inter-dependent. This means that non-optimal system settings or resource bottlenecks can lead to slow applications. Slow applications (e.g., with high resource consumption), in turn, can lead to resource bottlenecks.



Planned innovations include:

- Access to SAP HANA views through ABAP Data Dictionary
- Integration of SQL Script and database procedures into ABAP
- SAP HANA-specific source code sections
- Extended tooling for performance analysis and code checks with special focus on SAP HANA
- Transport of SAP HANA artifacts via ABAP transport requests



In addition to the approach explored in this article, SAP also plans to provide enhancements for developers to help increase productivity and improve the ABAP development experience.

These include:

- **Enhanced development environment.** With the ABAP development tools for SAP NetWeaver, SAP plans to offer a state-of-the-art, Eclipse-based development environment for ABAP.¹ This environment will integrate smoothly with the SAP HANA studio and will allow developers to implement in-memory applications end-to-end within a single environment.
- **Developer support channels.** SAP aims to provide a variety of support for customers and partners in adopting the outlined innovations, as well as the overall integration of SAP HANA and ABAP development. These support channels for developers include publications, documentation, best practices, reference applications, sandbox systems/trial versions, and a lively online community.

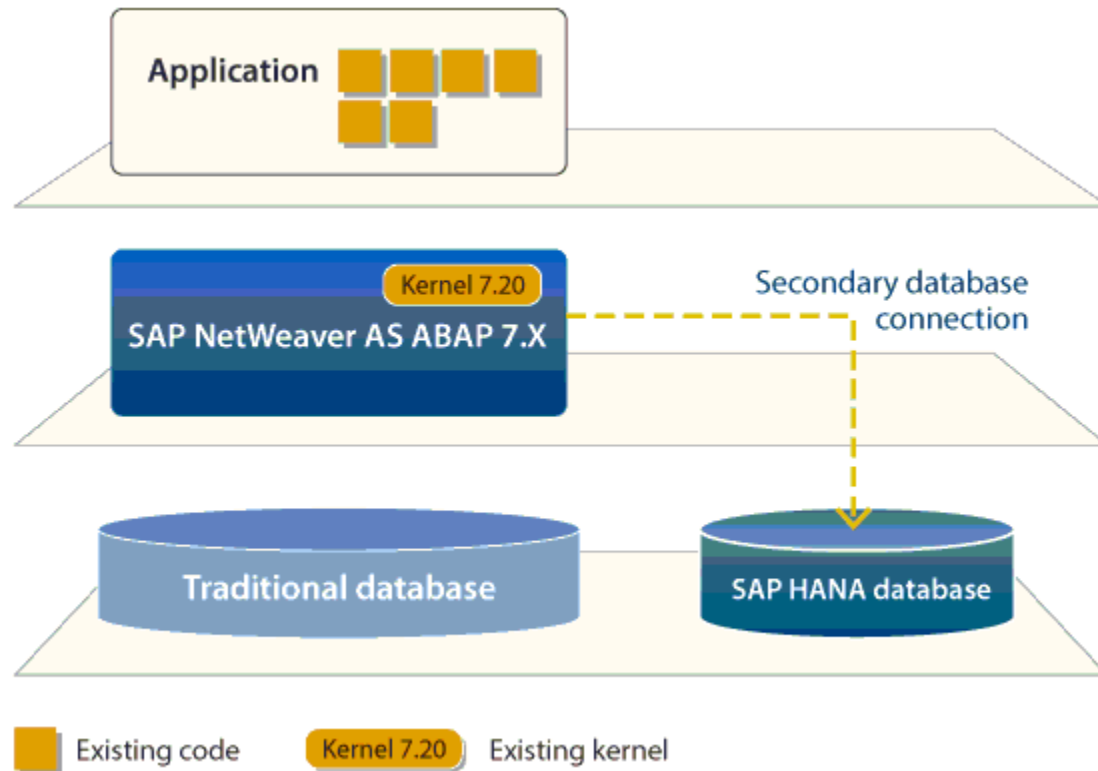
While these improvements complement the SAP HANA optimizations to SAP NetWeaver AS ABAP, SAP plans to make them available independent of the SAP HANA enhancements.

Providing Access to SAP HANA Through ABAP-Based Applications



Customers running SAP NetWeaver AS ABAP can already use SAP HANA. As of SAP NetWeaver AS ABAP 7.x, SAP NetWeaver AS ABAP applications running on a traditional database (such as IBM DB2, SAP MaxDB, or Oracle) can access an SAP HANA database using a secondary database connection

Providing Access to SAP HANA Through ABAP-Based Applications



Providing Access to SAP HANA Through ABAP-Based Applications

This capability supports side-by-side scenarios in which selected data is replicated from the primary database to the secondary SAP HANA database. SAP NetWeaver AS ABAP and the respective application (SAP ERP, for instance) can then use the secondary database connection to read data from and delegate calculations to the SAP HANA database.

Providing Access to SAP HANA Through ABAP-Based Applications

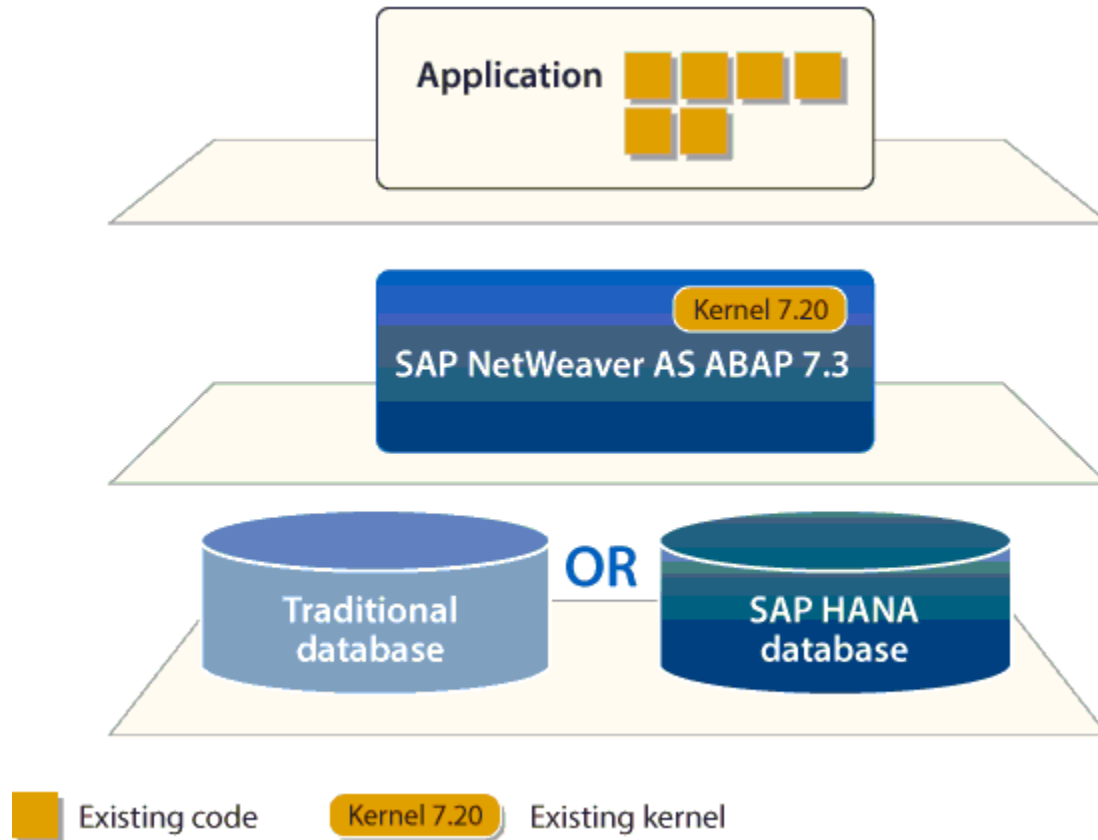
The secondary database connection helps customers benefit from SAP HANA without making changes to the primary database. In particular, this approach enables use cases for SAP HANA accelerators that leverage SAP HANA to improve the performance of specific processes managed in SAP Business Suite applications with high volumes of data involved.

Enabling ABAP to Run on SAP HANA



With SAP NetWeaver AS ABAP 7.3, the SAP HANA database can be used for primary data persistence to support analytical use cases in which data can be analyzed according to the reporting requirements of a line of business. SAP has implemented this support with the introduction of SAP NetWeaver BW powered by SAP HANA, which enables SAP NetWeaver BW to use SAP HANA as an in-memory database

Enabling ABAP to Run on SAP HANA



Enabling ABAP to Run on SAP HANA



Running SAP NetWeaver BW on SAP HANA has a variety of benefits.

With SAP HANA, companies can achieve optimized query performance without SAP NetWeaver BW Accelerator, helping customers and partners reduce the TCO of their data warehousing solution.

SAP HANA can also help accelerate extract, transform, and load (ETL) processes and simplify data modeling since it needs fewer materialized layers.

Optimizing ABAP for SAP HANA



Going forward, SAP plans to enable more solutions, in particular SAP Business Suite applications, to use SAP HANA for primary data persistence.

SAP also plans to make adjustments to SAP NetWeaver AS ABAP to support SAP HANA as the underlying database for transactional use cases.

Additionally, SAP intends to facilitate a deeper integration of ABAP development with SAP HANA to enable SAP developers, as well as customers developing custom applications, to more easily leverage the strengths of SAP HANA within application logic.

Optimizing ABAP for SAP HANA



By porting SAP NetWeaver AS ABAP-based solutions on the SAP HANA database and using the database for primary data persistence, customers and partners can reduce the TCO of their overall landscapes.

For example, replication to secondary databases will no longer be required and SAP HANA accelerators can run locally.

With operational data residing in SAP HANA, customers can perform ad hoc reporting on top of their applications to improve operational reporting performance.

Optimizing ABAP for SAP HANA



As SAP NetWeaver AS ABAP support for SAP HANA grows, completely new applications are also emerging — in the areas of trade promotion management and fraud management, for example — that combine both transactional and analytical behavior.

With this trend, the borders between OLTP and OLAP will diminish over time.

Customers and partners will be able to develop these types of hybrid solutions on top of SAP NetWeaver AS ABAP optimized for SAP HANA.

Optimizing ABAP for SAP HANA

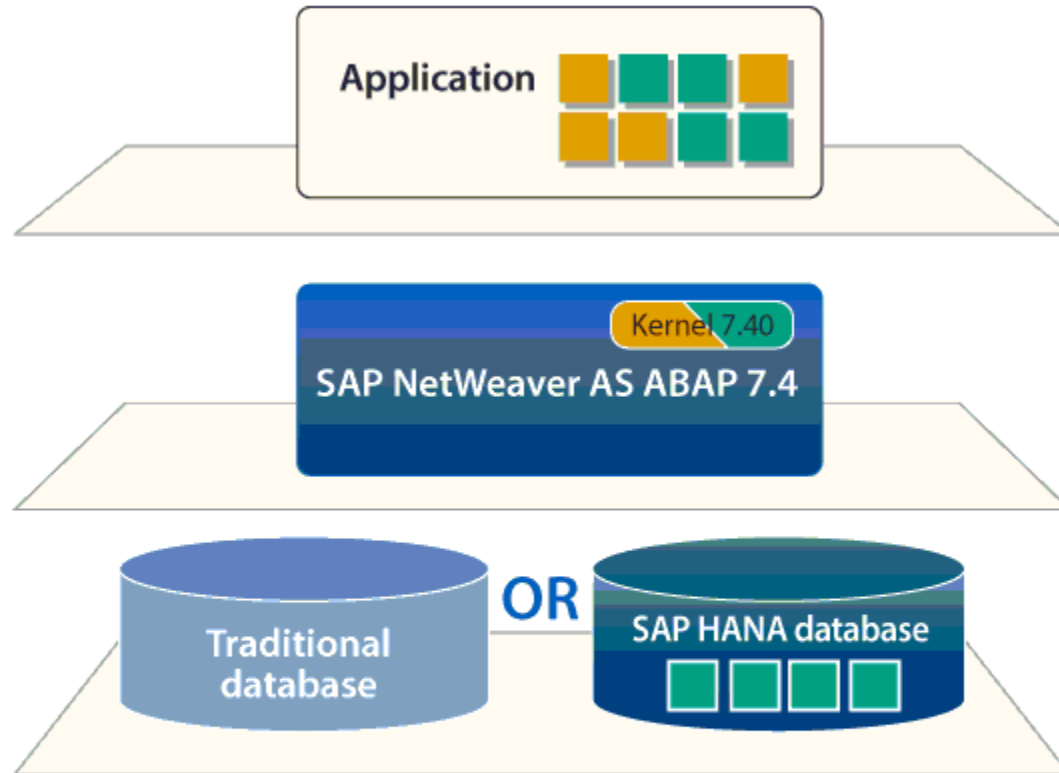





SAP plans to provide enhancement package 7.4 for SAP NetWeaver AS ABAP to focus on these optimizations for the SAP HANA database .

Intended to be compatible with SAP NetWeaver AS ABAP 7.3, the new enhancement package will allow customers and partners to adopt innovations in a non-disruptive, step-by-step way.

SAP plans to make this new enhancement package available to pilot customers and partners at the end of 2012, and will engage closely with the SAP ecosystem to validate custom development use cases. General availability is planned for 2013.

Optimizing ABAP for SAP HANA



-  Existing code
-  Innovations: optimized code for SAP HANA
-  Kernel with SAP HANA optimizations

The New Enhancement Package for SAP NetWeaver AS ABAP

The planned starting point for developing enhancement package 7.4 for SAP Net Weaver AS ABAP is the current SAP Net Weaver AS ABAP 7.3. All corrections and continuous innovations for the 7.3 code line will be made available for the new enhancement package.

The enhancement package is intended to be fully compatible with SAP Net Weaver AS ABAP 7.3, so that applications running on top of SAP Net Weaver AS ABAP will continue to run, without disruption, when upgrading to the new enhancement package.



Adapting for SAP HANA means making ABAP code database independent

In general all ABAP code runs on SAP HANA as on any other supported database. Only if ABAP code relies on **technical specifics of the old database, ABAP code changes** might be necessary

Some SQLs are faster on HANA , e.g. when selecting columns without an index. Only ABAP code which relies on specific features or capabilities of the predecessor database must be analyzed and corrected. There are three main aspects for custom ABAP code adaptation for SAP HANA:



Mandatory corrections of your code to avoid functional regression

- Target: ABAP code, SQL code or database calls which build on specific features of the predecessor database (e.g. native SQL, ABAP code relying on certain sort order of a SQL result set)

Recommended SQL performance optimizations in order to avoid performance degradations

- Target: “Bad” SQLs in custom code playing a dominant role in the runtime profile of productive business processes. Special focus is on SQLs which are executed very frequently (e.g. SELECT in loop).



Exploit SAP HANA Code pushdown techniques

- By using Core Data Services and SQL script procedures to speed up custom code processes significantly.
- This step normally happens after the migration and is addressed in custom development projects driven by business requirements.

The ABAP codes which must be changed to avoid potential functionality issues are as follows-

- Migration and Upgrade Overview
- SELECT Statements
- Native SQL, Hints, ADBC
- Pool Tables, Cluster Tables
- Indices
- Rare Issues

Migration of ABAP Code to SAP HANA



Technical changes with SAP HANA that may affect existing **DB specific** ABAP code

Technical Change	Details and Examples – Effect on DB specific ABAP code
DB migration	Each DB has specific features and unique technical behavior. <i>DB specific code may rely on these features of the used database.</i>
SAP HANA architecture	Column based architecture - as a consequence e.g. secondary DB indexes are less important. <i>DB specific code may rely on the existence/usage of certain DB indexes.</i>
Depooling/Decustering	During the migration to SAP HANA most pool and cluster DB tables are transformed to transparent DB tables (depooling/decustering) so that the tables can be used in analytic scenarios. <i>DB specific code may rely on the technical specifics of pool and cluster tables.</i>

Migration of ABAP Code to SAP HANA



Technical Change	ABAP code which must be checked / corrected	Recommended solution
DB migration	Native SQL, Hints, ADBC which uses DB specific features	Use Open SQL if possible, adapt native SQL and DB hints if necessary

Example for native SQL

```
EXEC SQL.  
  SELECT ERA, SHERA, STDATE INTO :JPERA  
  FROM LDERA  
  WHERE LANG = :SY-LANGU  
  AND STDATE = ( SELECT MAX(STDATE)  
                  FROM LDERA  
                  WHERE LANG = :SY-LANGU  
                  AND STDATE <= :INP )  
ENDEXEC.
```

Example for a DB hint

```
SELECT SINGLE la1  
  FROM ABC  
  INTO l_wa  
  WHERE plnnr = a  
  AND vornr LIKE b  
  %_HINTS ORACLE 'index(ABC "ABC~XCYZ")'.
```

Migration of ABAP Code to SAP HANA



SELECT statements should be changed because HANA follows the SQL standard.

Implicit behaviors of databases are not guaranteed by the SQL standard.

Migration of ABAP Code to SAP HANA



For SELECT * statements changes have to be done in following ways

```
select * from ekpo  
into table lt_ekpo  
for all entries in lt_ekko  
where ebeln = lt_ekko-ebeln.
```

```
read table lt_ekpo into l_ekpo  
with key ebeln = ebeln ebelp = ebelp  
Binary search.
```

TODO



```
select * from ekpo  
into table lt_ekpo  
for all entries in lt_ekko  
where ebeln = lt_ekko-ebeln  
order by primary key.
```

```
read table lt_ekpo into l_ekpo  
with key ebeln = ebeln ebelp = ebelp  
Binary search.
```

Migration of ABAP Code to SAP HANA



For SELECT SINGLE statements, it should be written as

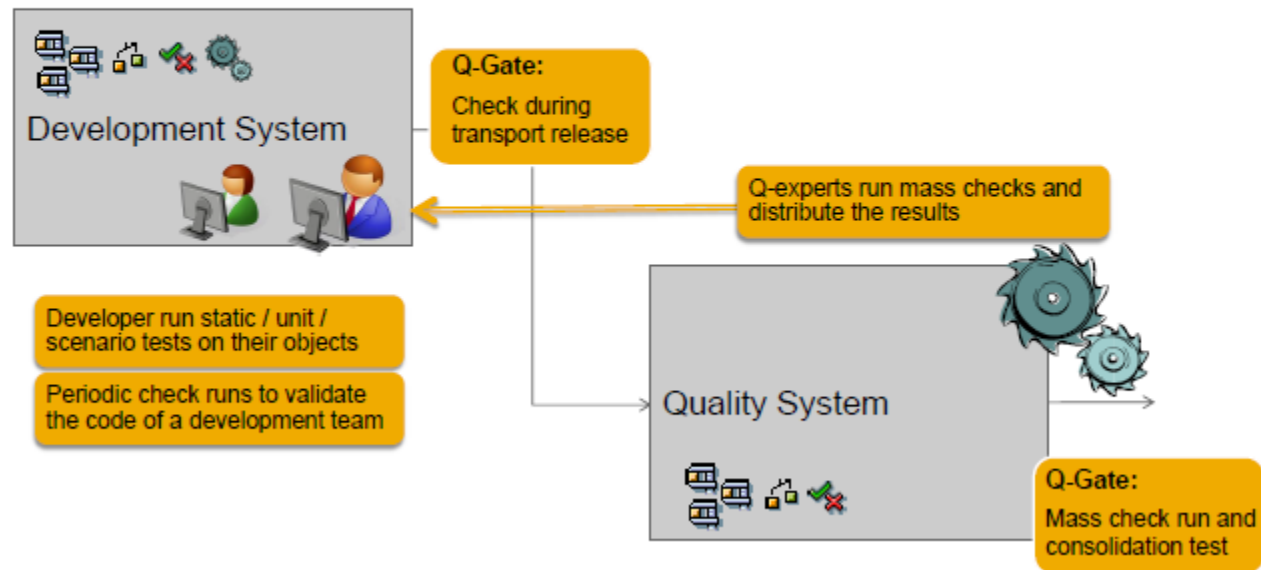
```
34      SELECT carrid carrname
35          INTO CORRESPONDING FIELDS OF
36          TABLE itab FROM scarr
37          UP TO 1 ROWS
38          WHERE currcode = 'EUR'
39          ORDER BY carrid.
40      READ TABLE itab
41          INTO structure INDEX 1.
42
43      WRITE: / 'Result'.
44      WRITE: / structure-carrid,
45              structure-carrname.
```

Migration of ABAP Code to SAP HANA



Prepare your custom code for SAP HANA

ABAP Test Cockpit – Quality process and Q gates



Migration of ABAP Code to SAP HANA



ABAP Test Cockpit and static checks



In general existing ABAP code runs on SAP HANA as before. Only DB specific ABAP code must be analyzed

New static Code Inspector checks are available which help to find the code which must be analyzed

ABAP Test Cockpit (ATC) is the new standard tool for static ABAP code checks

ATC can be used now to prepare the custom ABAP code for HANA and to improve the general quality of the code




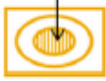

ATC availability starts with NW 702 SP12 / NW 731 SP5. In older releases the Code Inspector can be used

Migration of ABAP Code to SAP HANA



Optimize Custom Code for SAP HANA

Classic golden SQL rules

Icon	Rule	Details / Examples
	Keep the result sets small	<ul style="list-style-type: none">• Do not retrieve rows from the database and discard them on the application server using CHECK or EXIT, e.g. in SELECT loops• Make the WHERE clause as specific as possible
	Minimize amount of transferred data	<ul style="list-style-type: none">• Use SELECT with a field list instead of SELECT * in order to transfer just the columns you really need• Use aggregate functions (COUNT, MIN, MAX, SUM, AVG) instead of transferring all the rows to the application server
	Minimize the number of data transfers	<ul style="list-style-type: none">• Use JOINS and / or sub-queries instead of nested SELECT loops• Use SELECT ... FOR ALL ENTRIES instead of lots of SELECTs or SELECT SINGLEs• Use array variants of INSERT, UPDATE, MODIFY, and DELETE
	Minimize the search overhead	<ul style="list-style-type: none">• Define and use appropriate secondary indexes
	Keep load away from the database	<ul style="list-style-type: none">• Avoid reading data redundantly• Use table buffering (if possible) and do not bypass it• Sort Data in Your ABAP Programs



Optimize Custom Code for SAP HANA

Shift in priorities of classic golden SQL rules for SAP HANA

Icon	Rule	Details / Examples
	Keep the result sets small	<ul style="list-style-type: none"> Do not retrieve rows from the database and discard them on the application server using CHECK or EXIT, e.g. in SELECT loops Make the WHERE clause as specific as possible
	Minimize amount of transferred data	<ul style="list-style-type: none"> Use SELECT with a field list instead of SELECT * in order to transfer just the columns you really need Use aggregate functions (COUNT, MIN, MAX, SUM, AVG) instead of transferring all the rows to the application server
	Minimize the number of data transfers	<ul style="list-style-type: none"> Use JOINS and / or sub-queries instead of nested SELECT loops Use SELECT ... FOR ALL ENTRIES instead of lots of SELECTs or SELECT SINGLEs Use array variants of INSERT, UPDATE, MODIFY, and DELETE
	Minimize the search overhead	<ul style="list-style-type: none"> Define and use appropriate secondary indexes
	Keep load away from the database	<ul style="list-style-type: none"> Avoid reading data redundantly Use table buffering (if possible) and do not bypass it Sort Data in Your ABAP Programs

Shift in priorities for SAP HANA



- Avoid selecting large numbers of not required columns
- Prefer array operations for INSERT, UPDATE und DELETE when changing many records
- Avoid nested SELECT loops
- In most cases SAP HANA does not require secondary indices
- Keep **unnecessary** load away from the database
- Code pushdown for data-intensive calculations to benefit from SAP HANA

Migration of ABAP Code to SAP HANA



Optimize Custom Code for SAP HANA Classic Static Performance Checks

Icon	Rule	Details / Examples
	Keep the result sets small	<ul style="list-style-type: none"> Do not retrieve rows from the database and discard them on the application server using CHECK or EXIT, e.g. in SELECT loops Make the WHERE clause as specific as possible
	Minimize amount of transferred data	<ul style="list-style-type: none"> Use SELECT with a field list instead of SELECT * in order to transfer just the columns you really need Use aggregate functions (COUNT, MIN, MAX, SUM, AVG) instead of transferring all the rows to the application server
	Minimize the number of data transfers	<ul style="list-style-type: none"> Use JOINS and / or sub-queries instead of nested SELECT loops Use SELECT ... FOR ALL ENTRIES instead of lots of SELECTs or SELECT SINGLES Use array variants of INSERT, UPDATE, MODIFY, and DELETE
	Minimize the search overhead	<ul style="list-style-type: none"> Define and use appropriate secondary indexes
	Keep load away from the database	<ul style="list-style-type: none"> Avoid reading data redundantly Use table buffering (if possible) and do not bypass it Sort Data in Your ABAP Programs

Most golden SQL rules are covered by static checks already

Check Variant	BG_PERFORMANCE_CLASSIC	Person Respons.
Changed On	26.08.2013	Last Changed By
Description	BG_PERFORMANCE_CLASSIC	
Selection	D..	A.. Tests
		List of Checks
		General Checks
		Performance Checks
		Analysis of WHERE Condition for SELECT
		Analysis of WHERE Condition in UPDATE and DELETE
		SELECT Statements That Bypass the Table Buffer
		SELECT Statements with Subsequent CHECK
		SELECTs in Loops
		Changing Database Accesses in Loops
		Nested Loops
		Low Performance Operations on Internal Tables
		Copy Large Data Objects
		Low-Perform. Parameter Transfers
		EXIT or no statement in SELECT...ENDSELECT loop
		Instance Creation of BAdIs
		Table Attributes Check
		Security Checks
		Syntax Check/Generation
		Robust Programming
		Search for APPEND and INSERT ... INDEX in SORTED Tables
		Complex WHERE Condition in SELECT Statement
		Check of SY-SUBRC Handling
		Unsecure use of FOR ALL ENTRIES



Optimize Custom Code for SAP HANA

New Static Performance Checks I

Improved and new static performance checks:

- *SELECT ** - Used columns vs. selected columns
- Nested *SELECT* statements (across modularization units)
- *SELECT + SELECT FOR ALL ENTRIES*
which can be transformed into JOIN or SUBQUERY

```
select single * from mara  
into wa_mara  
where matnr = i_matnr  
if sy-subrc = 0.  
....
```

SELECT * check
No field used – pure existence check

```
loop at it_mara into mara.  
...  
perform mara_exist_check using mara-matnr changing rc.  
endloop.  
  
form mara_exist_check using i_matnr type mara-matnr changing rc.  
select single * from mara  
into wa_mara  
where matnr = i_matnr.  
if sy-subrc = 0.
```

Nested SQL check
SELECT in LOOP (across
modularization unit)

	Performance Checks
•	Analysis of WHERE Condition for SELECT
•	Analysis of WHERE Condition in UPDATE and DELETE
•	SELECT Statements That Bypass the Table Buffer
•	Search problematic SELECT * statements
•	Search SELECT ... FOR ALL ENTRIES-clauses to be transformed
•	Search SELECT statement with DELETE statement
•	Search DB Operations in loops across modularization units
•	Changing Database Accesses in Loops
•	Nested Loops
•	EXIT or no statement in SELECT...ENDSELECT loop
•	SELECT Statements with Subsequent CHECK
•	Low Performance Operations on Internal Tables
•	SORT Statement Inside Loop
•	Copy current table row for LOOP AT ...
•	Low-Perform. Parameter Transfers
•	Copy Large Data Objects
•	Table Attributes Check
•	Instance Creation of BAdIs
•	Find Interface Method Calls in Loops

```
SELECT vbeln FROM vbak INTO TABLE lt_vbak1 WHERE ...
```

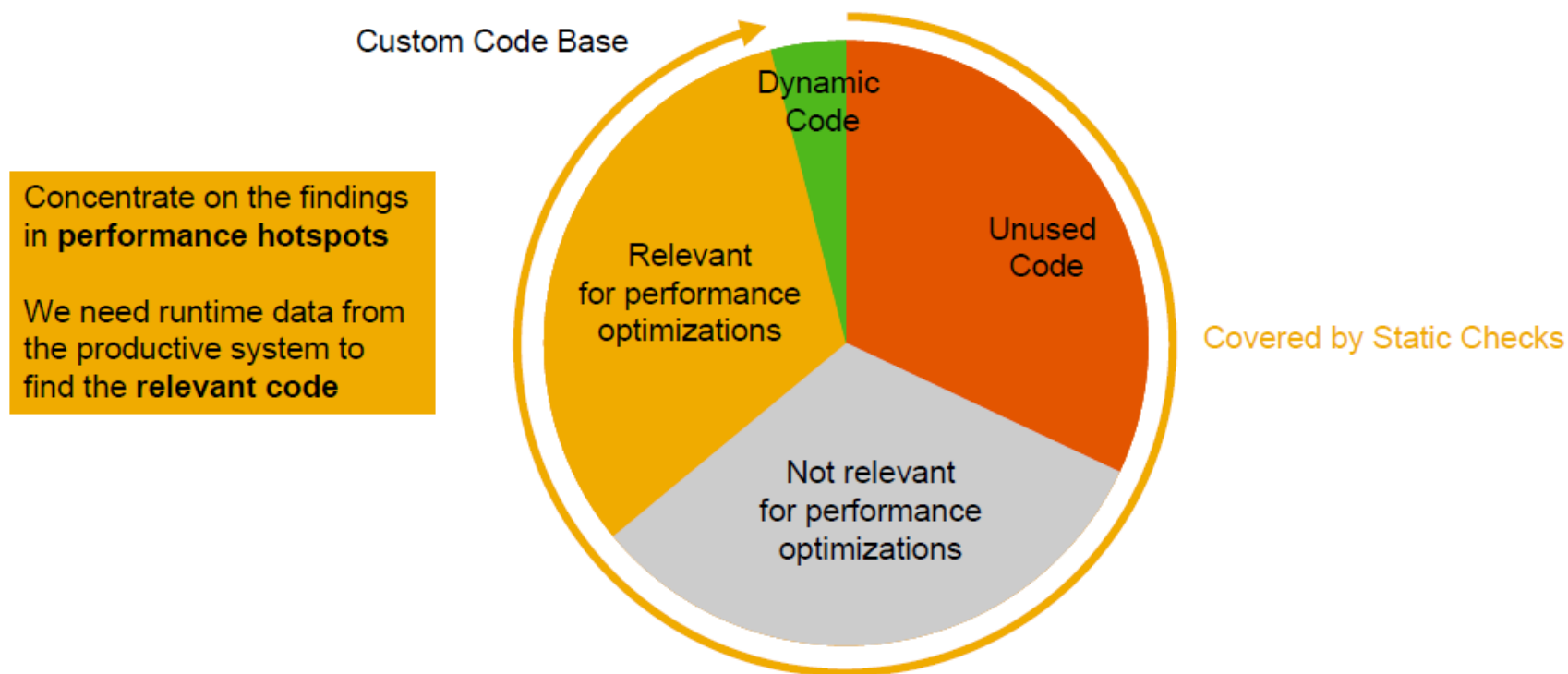
```
SELECT vbeln FROM vbpa INTO TABLE lt_vbpa FOR ALL ENTRIES IN lt_vbak1  
WHERE ...  
vbeln = lt_vbak1-table_line.
```

SELECT + SELECT FOR ALL ENTRIES
transform into JOIN or SUBQUERY



Optimize Custom Code for SAP HANA

Static Performance Check – Efficient analysis I



Static Performance Check – Efficient analysis III

Processes covered by SQL Monitor after a week / month running in production.
Quarter-end or year-end processes may not be covered



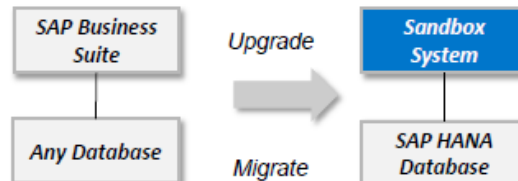
Migration of ABAP Code to SAP HANA



Tune your custom code – Get ready for SAP HANA

Summary – Process view – Tool usage

Plan SAP HANA migration



Eliminate unused custom code to reduce migration effort

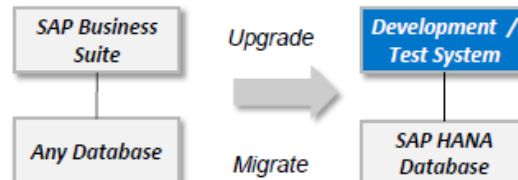
*Define your **main business processes** for testing and tuning*

*Evaluation in **sandbox system***

*Use **ABAP Test Cockpit** to correct potential functional issues in your custom code ("old" system or sandbox system)*

*Use **SQL Monitor** in the „old“ **productive system** (+ SWLT in the development system) to correct custom SQL issues / tune the top custom SQL statements in your system*

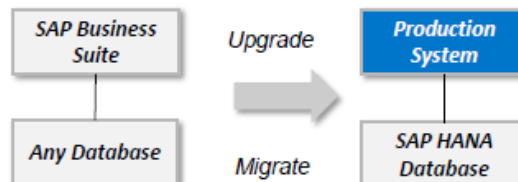
Prepare SAP HANA migration



*Use **ABAP Test Cockpit** for your general quality assurance (avoid regressions)*

*Use **SQL Monitor** in the **SAP HANA test system** (+ SWLT in the development system) to tune the top custom SQL statements of the running test cases*

SAP HANA migration



*Use **ABAP Test Cockpit** for your general quality assurance*

*Use **SQL Monitor** in the **SAP HANA productive system** (+ SWLT in the development system) **to iteratively tune** the custom SQL statements and **to detect and realize HANA potential***



Tune your custom code – Get ready for SAP HANA

Key takeaways



In general existing ABAP code runs on SAP HANA as before. Only DB specific ABAP code must be analyzed

ABAP Test Cockpit can be used to find and adapt DB specific ABAP custom code easily

In general the well known golden Open SQL rules are still valid on SAP HANA - only some priorities have shifted

For effective SQL tuning and to find HANA potential in existing ABAP code new monitoring tools like the SQL Monitor are available

The preparation of your custom code can start **now** – before the migration to SAP HANA



Check ID	Check Description
P1	Search on WHERE condition on SELECT statement.
P2	Analysis of WHERE condition for UPDATE and DELETE statement.
P3	SELECT statements that bypass the SAP table buffer.
P4	Search problematic SELECT * statements.



Check ID	Check Description
P5	SELECT statement with subsequent CHECK statement.
P6	Search SELECT...FOR ALL ENTRIES-clause to be transformed.
P7	Search SELECT statement with DELETE statement.
P8	Search DB operations in LOOPS across modularization units.



Check ID	Check Description
P9	EXIT or No statement in SELECT...ENDSELECT loop.
P10	Search DB Operations
P11	Sort Statement inside LOOP...ENDLOOP.
P12	Search for APPEND / INSERT... INDEX for SORTED table.

Performance Checks



Check ID	Check Description
P13	Unsecure usage of FOR ALL ENTRIES IN statement.
P14	Nested loops



Check ID	Check Description
F1	Search problematic statements for result of SELECT / OPEN CURSOR without ORDER BY.
F2	Find ABAP statement patterns.
F3	Find ORACLE rule Hints.
F4	Critical Statements.
F5	Depooling cluster for SELECT/Order By

Brief over view of different performance analyzer tools

SE30

- The ABAP Runtime Analysis (transaction SE30) gives you one tool for solving two problems.
- You can measure performance and find bottlenecks.
- You can also analyze the program flow of your ABAP program.
- First, you may need to find the exact source code location of a particular ABAP statement (a method call, function call...) you are interested in.
- You would then run the ABAP Trace and afterwards search the required line in the result list of the ABAP statements.

Brief over view of different performance analyser tools

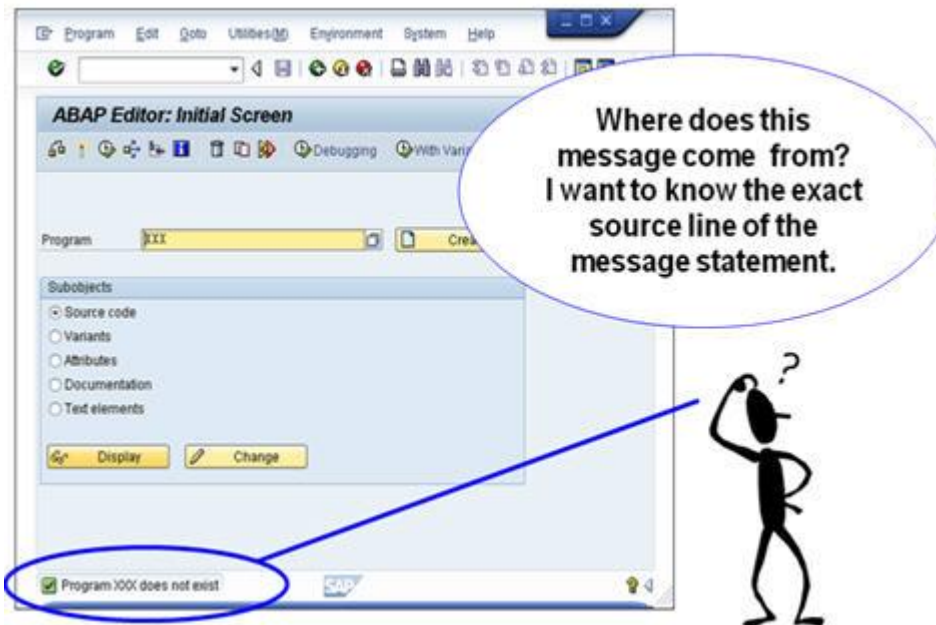
SE30

- Second, you may want to compare the flow of your ABAP program in different systems.
- Imagine, for example, that your ABAP program runs as expected in the test system but shows a completely differently behavior in the production system, or even worse, aborts with a short dump in the production system.
- You could then simply run the ABAP Trace in both test and production systems and compare the trace results.

Brief over view of different performance analyser tools

SE30

- Exact source code line of an ABAP statement
 - you go to the ABAP Editor (transaction SE38), type "XXX" into the "Program" field, press the "Display" button and get the error message on the status bar "Program XXX does not exist".



Brief over view of different performance analyser tools

SE30

- exact source code line of an ABAP statement
 - To find the message, first start the ABAP Runtime Analysis and create a measurement variant.
 - Start the ABAP Runtime Analysis (transaction SE30) via *System -> Utilities -> Runtime Analysis -> Execute* or call the transaction directly with *"/nse30"*.
 - Type *"SE38"* into *"Transaction"* field.
 - Create a measurement variant for your user:
 - Type a name into *"Variant"* field and press *"Create"* button
 - Set aggregation to *"None"* on the *"Duration/Type"* tab
 - For memory usage info check the *"With memory use"* flag
 - Switch on *"Particular units"* on the *"Program(Parts)"* tab
 - Save your variant

Brief over view of different performance analyser tools

Some Important Notes

- Don't use aggregation if you want to trace ABAP in order to follow the program logic (what we are doing here).
 - Aggregation summarizes the trace data for a particular type of event in a single trace record and therefore reduces the number of entries in the trace file.
 - But to see the complete program flow you need all trace data.

- **Try to use “*Particular units*” where possible in order to reduce trace file size** and trace only the code you really need to see.
 - The option “*Particular units*” allows you to switch on/off the ABAP trace during the running transaction. The trace will be started as soon as you enter “/ron” (trace on) in the OK field in your transaction.
 - With “/roff ” the trace is stopped.
 - Alternatively you can also use the menu path: *System -> Utilities -> Runtime Analysis -> Switch On / Switch Off.*

Brief over view of different performance analyser tools

SE30

- Let's execute the measurement:
 - Press "*Execute*" button. Transaction SE38, the ABAP Editor, starts.
 - Type "*XXX*" into the "*Program*" field and turn on the trace with *System -> Utilities -> Runtime Analysis -> Switch On*.
 - Press the "*Display*" button and turn off the trace with *System -> Utilities -> Runtime Analysis -> Switch Off*.

Brief over view of different performance analyzer tools

The diagram illustrates the workflow for analyzing ABAP performance. It consists of two main screenshots with annotations:

- ABAP Runtime Analysis: Initial Screen** (Top Left):
 - Measurement: ☒ Reliability of Time Values
 - Short Description: [Empty field]
 - In Dialog: Transaction ☒ se38, Program [Empty], Function module [Empty]
 - In Parallel Session: ☒ Switch On/Off
 - Schedule: For UsedService
 - Measurement Restrictions: Variant ☒ trace_variant, From user [Empty]
 - Buttons: Execute, For UsedService
 - Annotation: "Press 'Execute'" points to the Execute button.
- ABAP Editor: Initial Screen** (Top Right):
 - Program: se38
 - Subobjects: Source code, Variants, Attributes, Documentation, Test elements
 - Buttons: Display, Change
 - Annotation: "Run your transaction" points to the Program field.
- ABAP Runtime Analysis: Initial Screen** (Bottom Left):
 - Performance Data File:
 - Application: TRACE_VARIANT_DOLINSKAJA
 - Short description: TRACE_VARIANT_DOLINSKAJA
 - Measurement date: 02.12.2009 12:28:19
 - File size in kB: 1.453
 - Buttons: Evaluate, Other File..., File Info..., Delete...
 - Annotation: "Press 'Evaluate' to analyze results" points to the Evaluate button.
- Annotation:** "Step back to the runtime analysis. Your trace file is now accessible" points from the ABAP Editor back to the bottom ABAP Runtime Analysis screen.

Brief over view of different performance analyser tools

SE30

- Step back to the Runtime Analysis and analyze the trace results:
 - Press the “*Evaluate*” button.
 - Press the “*Call Hierarchy*” button and you get a list which represents the complete path through your program.
 - Search for “message DS017” in the *Call Hierarchy* list.
 - Double-click on the entry in the *Call Hierarchy* list to jump to the source code line, which initiated the error message.

Brief over view of different performance analyzer tools

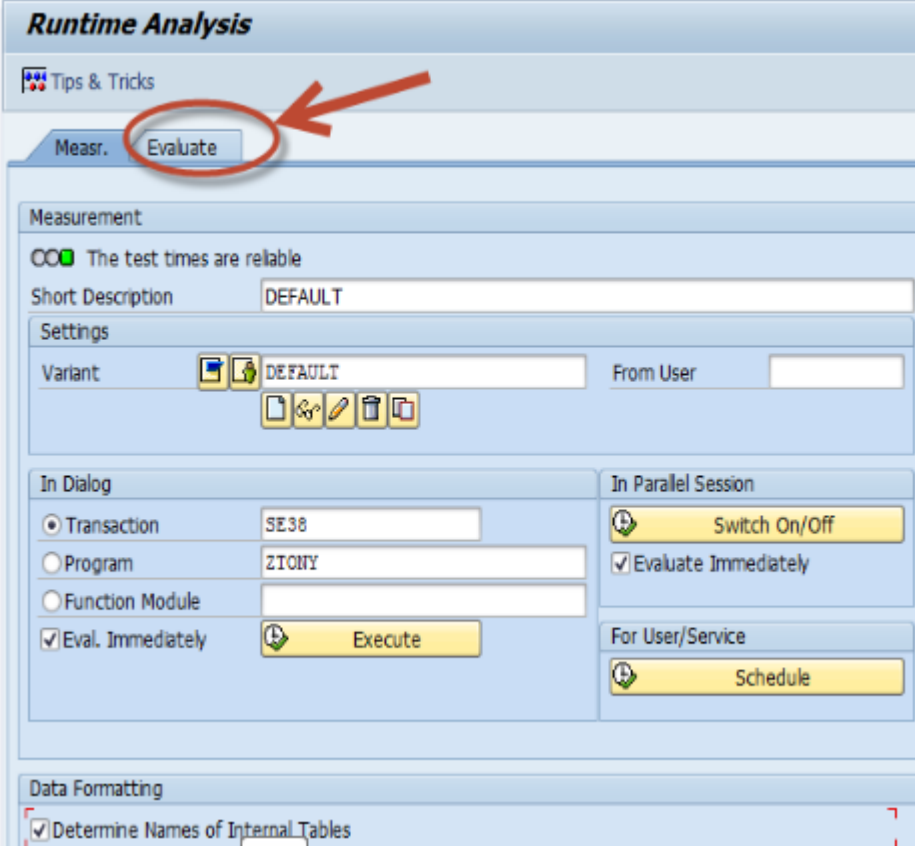


SAT

- Starting a trace measurement in SAT is quite similar to SE30. There are no major changes on the first screen from SE30 to SAT.
- A new tab **Evaluate** has been added to the screen.
- On the Evaluate tab you can take a look at existing trace measurements results with their status, date of creation, total runtimes.
- Only your traces are shown, but you can change this by right-clicking on the column Trace user and choosing the Set Filter

Brief over view of different performance analyser tools

SAT



Runtime Analysis

Tips & Tricks



Measr. **Evaluate**






Measurement

☐☐ The test times are reliable

Short Description

Settings

Variant   From User


    

In Dialog


☒ Transaction

☐ Program

☐ Function Module


☒ Eval. Immediately 

In Parallel Session



☒ Evaluate Immediately

For User/Service



Data Formatting

☒ Determine Names of Internal Tables

Brief over view of different performance analyser tools

SAT

- In the old SE30, internal tables were identified only by their internal names (IT_<no>). A nice feature in the new SAT transaction is that now SAP can determine and show the names of internal tables as they appear in your ABAP source code for the trace measurement results.
- Another new benefit of SAT is a new storage system for traces. SAT stores traces in the database. That means that you can display a trace from any application server in the system (in SE30 only traces of the current application server are visible). When you display a trace measurement result the first time (e.g. by you double clicking it on the *Evaluate* tab), it gets formatted and written to the database.

Brief over view of different performance analyser tools

ATC

- The ABAP Test Cockpit (ATC) is an ABAP check toolset which allows running static checks and unit tests for your ABAP development objects.
- The ABAP Test Cockpit can integrate tests written for the Code Inspector to enable reuse of existing checks and check variants in the ABAP Test Cockpit.
- The ABAP Test Cockpit is directly integrated into the ABAP workbench as well as in the ABAP Development Tools for Eclipse and allows checking code from just within the development environment the ABAP developer is used to.
- Working with the ABAP Test Cockpit as a developer is easy and efficient with the new ABAP Test Cockpit filter, navigation, and re-check functionality.

Brief over view of different performance analyser tools

ATC

- Team leads and quality engineers will like the ABAP Test Cockpit because it introduces new quality assurance processes like quality gates, a robust exemption approval process, email notifications on errors and exemption workflow events, and periodic regression tests in a quality system.
- In addition, the ATC offers tools to analyze the ABAP Test Cockpits results on team or component level. The integration of the ABAP Test Cockpit into the SAP Solution Manager allows cross system quality reporting.

Brief over view of different performance analyser tools

ATC

- Check single object from the **Development Workbench**
 - You can check a single object with the ABAP Test Cockpit from the ABAP Object Navigator (SE80), ABAP Editor (transaction SE38), the Function Builder (transaction SE37), the Class Builder (transaction SE24), or the ABAP Data Dictionary (transaction SE11). To do this, choose *<object>* > *Check* > ABAP Test Cockpit (*ATC*) from the menu, where *<object>* can be a package, program, function module, class, or table. The respective single objects, in case of a package, the objects of the package, are then checked with the default check variant

Brief over view of different performance analyser tools

ATC

- Check multiple objects from the **Development Workbench**
 - You can check multiple objects with the ABAP Test Cockpit from the ABAP Object Navigator (SE80), ABAP Editor (transaction SE38), the Function Builder (transaction SE37), the Class Builder (transaction SE24), or the ABAP Data Dictionary (transaction SE11). To do this, choose *<object> > Check > ABAP Test Cockpit (ATC) with ...* from the menu. You can now add other objects to the list of objects to be checked via drag and drop. You can also select the check variant to be used and whether the checks shall be run in foreground or in background.

Brief over view of different performance analyser tools

ATC

- Checks on transport objects from the **Transport Organizer**
 - You can invoke the ABAP Test Cockpit from within the Transport Organizer (transaction SE09) to check objects in a transport request. To do this, choose *Request/Task > Check Objects*.
In addition checks can be invoked automatically on release of a transport. This way, you can disable transports not matching your quality criteria from being sent to further systems.

Brief over view of different performance analyser tools

ATC

- Schedule checks from **transaction ATC**
 - The ABAP Test Cockpit allows to run preconfigured tests on a regular basis. The selection of development objects to be checked can be based on a pick list or an object selection which is evaluated at the time the checks are run. As the basis for the object selection, you can define a list of packages, transport layers, software components or object sets defined with code inspector. This allows to include also newly created development objects without having to reconfigure the scheduled run. You schedule within the transaction ATC by ATC Administration > Runs > Schedule Runs.

Brief over view of different performance analyser tools

ST05

- The Performance Trace allows you to record database access, locking activities, remote calls of reports and transactions, and table buffer calls from the SAP system in a trace file and to display the performance log as a list. The Performance Trace additionally offers wide support when analyzing individual trace records in detail.
- Integration
 - The Performance Trace is fully integrated in the ABAP Workbench, and can be called from the ABAP Workbench initial screen.

Brief over view of different performance analyser tools

ST05

- The following traces are available:
 - SQL Trace: This allows you to monitor the database access of reports and transactions. See also SQL Trace Analysis.
 - Enqueue Trace: This allows you to monitor the locking system. See also Enqueue Trace Analysis
 - RFC Trace: This provides information about Remote Function Calls between instances. See also RFC Trace Analysis.
 - Table buffer trace: You can use this to monitor database calls of reports and transactions made via the table buffer. See also, Table Buffer Trace Analysis.



In this lesson, you have learnt:

- About the basics of SAP ABAP and SAP HANA
- Optimization techniques for SAP HANA Database
- How to access SAP HANA through ABAP applications
- Basics for migrating ABAP code to SAP HANA
- Different performance and functional checks
- About different performance analyzer tools

Review Question



The SAP HANA platform combines ----- software with hardware from leading SAP partners.

There are ----- performance checks and -----functional checks.

The ATC offers tools to analyze the ABAP Test Cockpits results on ----- level.

Review Question



The Performance Trace allows you to record -----,

-----, ----- of reports and transactions, and ----- from the SAP system in a trace file and to display the performance log as a list.

The integration of the ABAP Test Cockpit into the SAP Solution Manager allows cross system quality reporting.

- True
- False