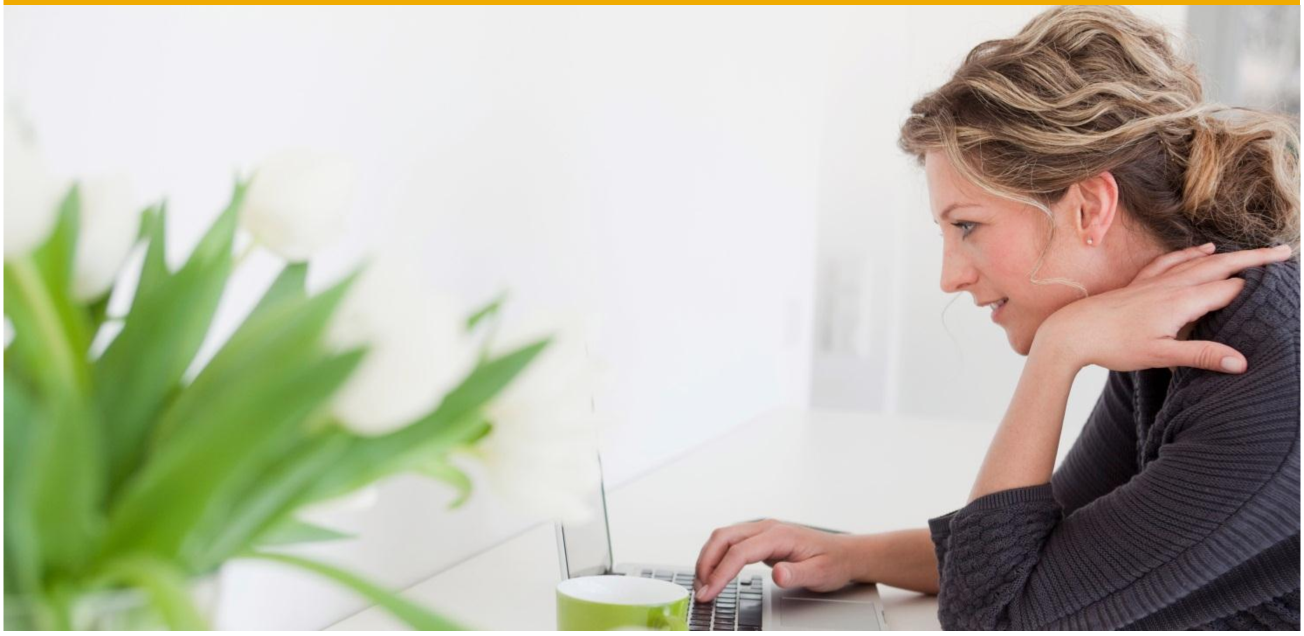


Considerations for Custom ABAP Code when Migrating to SAP HANA

Best Practices and Recommendations

TABLE OF CONTENTS

SAP HANA is a full-fletched relational database system optimized for transactional (OLTP) and analytical (OLAP) processing. As the underlying database system for the SAP NetWeaver Business Warehouse and the SAP Business Suite, it is fully compatible based on the Open SQL support in the SAP NetWeaver Application Server ABAP. However, existing database specific ABAP coding needs to be revisited and possibly adapted. Furthermore, there are some additional recommendations related to the nature of the In-Memory columnar storage in SAP HANA.

SCOPE OF THIS DOCUMENT

This document provides guidance for customers and partners regarding existing custom ABAP developments when transitioning to SAP HANA.

The focus of the document is on the following two aspects:

- **Avoiding functional regression during migration:** As during every database migration, existing database dependent coding based on the previous database system needs to be revisited. This includes coding relying on undocumented implicit behavior.
- **Detecting performance optimization potential:** SAP HANA allows significant optimizing of the performance of existing ABAP developments. In order to fully benefit from the power of SAP HANA, certain data access patterns should be revisited and optimized, in particular when classic performance recommendations for Open SQL were previously not taken into account.

The document contains an overview of the available tools and provides guidance how to use them in the context of a transition to SAP HANA. In particular, we explain the steps, which can be carried out before a technical migration takes place (including steps which should be considered independently from SAP HANA). The document primarily addresses ABAP coding in combination with standard database access via Open SQL. For some specific environments and frameworks, there may be additional guides and recommendations. For instance, for custom ABAP routines in SAP NetWeaver BW data flows, we recommend looking into SAP Note 1847431.

On the other hand, the document **does not** discuss the concrete optimizations of SAP standard code in the SAP Business Suite powered by SAP HANA, and also **does not** give recommendations for using native SAP HANA development options (e.g. Column Views, Database procedures) for performance optimization. This information is planned for a future development guide.

Furthermore, the document is not a replacement for existing approaches for performance optimization of SAP solutions (system tuning, hotspot analysis, etc.). For these aspects, please consider existing SAP services from SAP Active Global Support or similar offerings.

Structure of this document

The document is organized into three main sections. In the section **“Tools and Prerequisites”** we start with a short overview of the main tools, which can be used during the transition of ABAP-code to SAP HANA, and give references for their availability and corresponding SAP Notes. Afterwards, we discuss in the section **“Required and Recommended Adaptions”** the main areas where adaption of custom ABAP code may be required. These are grouped into three categories: database migration, functional and performance related topics. For each topic, we give some background information and explain how to address them during the transition. Finally, in the section **“Best Practices”**, we give concrete guidelines for planning and carrying out the adaptations with the help of the tools.

Relation to other information sources / services

The main purpose of this document is to provide enablement for custom ABAP code transition projects.

In addition, SAP is offering a number of tailored service offerings for assisting in SAP HANA projects covering in particular aspects related to custom ABAP developments. We strongly recommend considering these services when planning a migration or optimization project in order to ensure a smooth and cost-effective transition.

The following list gives a short overview on the offerings. For more information refer to the references at the end of this document.

Services Offered by SAP Consulting
- SAP HANA Assessment
- Custom Code Analysis and Optimization
- SAP Upgrade Services

Services Offered by SAP Active Global Support:
- SAP Business Process Performance Optimization
- SAP Technical Performance Optimization
- Services supporting Custom Code Management

In addition, a number of Enablement Services and Rapid Deployment Solutions are available. Please refer to the links at the end of this document for further information.

Validity and References
The descriptions and recommendations within this document are based on SAP NetWeaver ABAP 7.4 SP2 and SAP HANA 1.0 SPS5. The techniques and approaches have been successfully applied in several transition projects within and outside SAP.

As some content within this document may be subject to change, we included the available SAP Notes for these items. Please consult these for the most up to date information.

The following groups have been involved in creating and reviewing the content within this document:
- SAP Product and Innovation HANA Platform
- SAP Active Global Support
- SAP Consulting
- SAP Customer Solution Adoption
- SAP IT

TOOLS AND PREREQUISITES

In this section, we present several tools, which can be effectively used for checking existing custom ABAP developments for adaptation needs and detecting optimization potential. We focus mainly on new or enhanced tools related to ABAP database access, and give concrete guidance on how to apply them in a migration project.

Overview

There are many existing tools within the SAP NetWeaver AS ABAP, which can assist in optimizing the performance of custom ABAP developments. For instance, the statistical records (STAD), the ABAP Trace Tool (SAT) and the SQL Trace (ST05) can be used to gain deep insights into the performance of an application. We will not explain these existing tools in detail but refer to standard SAP documentation and publications.

These existing tools are complemented by some new utilities which can ease the transition to SAP HANA. This includes primarily possibilities to obtain a list of business processes ordered by their database usage. For each of these processes, their *SQL profile* can then be analyzed, e.g. in order to determine how frequent and via which access paths certain database tables are accessed, and how much time is spent for the data access. Furthermore, enhanced capabilities for analyzing ABAP code allows detecting coding that does not meet the classical performance guidelines for SQL programming. Based on the combination of runtime data (SQL profile) and ABAP code analysis, facts-based planning is possible.

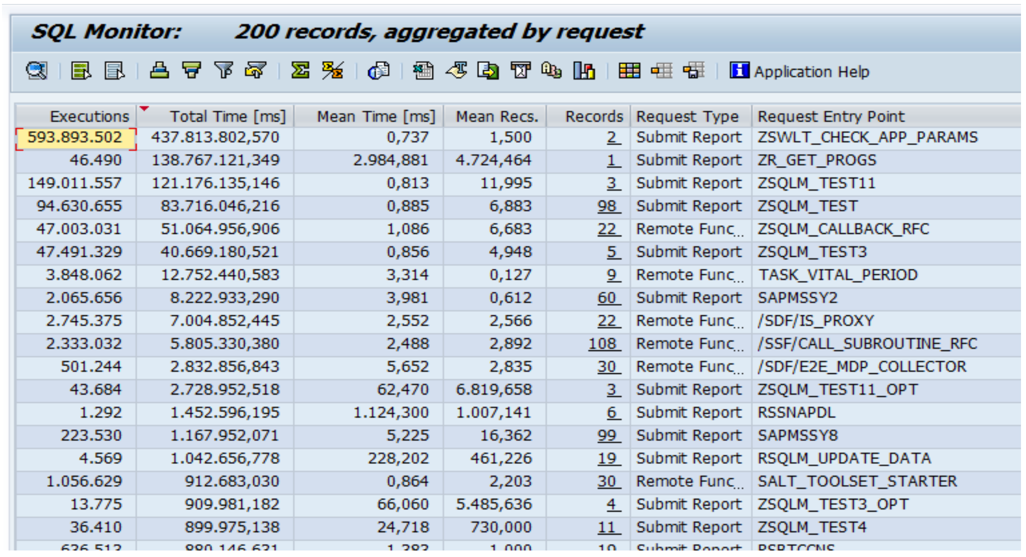
As effort estimates for adapting existing custom ABAP code are required before the start of the migration project, these tools are also provided for lower SAP NetWeaver releases. The following table summarizes the new or enhanced tools including their availability, and gives additional references for further information.

Tool Name, Transaction	**Capabilities**	**Availability** Releases
SQL Monitor *(new)* Transactions: SQLM (NW *Standard*) /SDF/ZQLM (*ST-PI Add-On*)	• Capture aggregated SQL runtime data over longer period in productive system • Determine the SQL profile of an application / business process	**NetWeaver standard (SQLM):** • NetWeaver 7.4 (SP2) • NetWeaver 7.03/7.31 (SP9) • NetWeaver 7.02 (SP14) Below NetWeaver 7.4, Kernel 7.21 (PL>118) is required. **Support Add-On ST-PI (/SDF/ZQLM)** • ST-PI 2008_1_* (SP8) **Requires:** • Kernel 7.21 (PL>118) • NetWeaver 7.00 or above (includes in particular 7.00, 7.01, 7.10, 7.11, 7.20, 7.30)
ABAP Test Cockpit *(enhanced)* *Transactions:* ATC, SCI, (SE80)	• New checks in SAP code inspector for detecting potential functional issues during transition • New performance checks for finding optimization potential	The ABAP code inspector infrastructure is available with NetWeaver 7.0 and above. The ABAP Test Cockpit is available in NetWeaver 7.02 (SP12), 7.03/7.31 (SP5) and 7.4 (SP0). New enhanced code checks are in specifically available in • NetWeaver 7.4 (SP2) • NetWeaver 7.03/7.31 (SP9) • NetWeaver 7.02 (SP14) We refer to the appendix for the availability of specific checks.
SQL Performance Tuning Worklist *(new)* *Transaction:* SWLT	• Possible to combine SQL Monitor data with the result of a static code analysis • Creation of a ranked worklist	• NetWeaver 7.4 (SP2) • NetWeaver 7.03/7.31 (SP9) • NetWeaver 7.02 (SP14)

Please note that the SAP NetWeaver standard version of the SQL Monitor is the recommended variant whenever it is available. The variant provided via the ST-PI Add-On has been added to also address lower releases such as SAP NetWeaver 7.00 (see table above). For the current SAP Support Package Schedule please refer to https://service.sap.com/~sapidb/011000358700000294692004E.

SQL Monitor

The main new utility for detecting performance optimization potential is the SQL monitor (transaction SQLM), which collects statistical information for all SQL statements originating from an ABAP program (including Open SQL, native SQL and SQL statements issued by the ABAP kernel). The SQL Monitor can be enabled for all or dedicated applications servers of an ABAP system. It can run in a productive system in parallel to the productive usage since the data collection is highly optimized using the runtime monitoring infrastructure in the ABAP kernel.



SQL Monitor: 200 records, aggregated by request

Executions	Total Time [ms]	Mean Time [ms]	Mean Recs.	Records	Request Type	Request Entry Point
593.893.502	437.813.802,570	0,737	1,500	2	Submit Report	ZSWLT_CHECK_APP_PARAMS
46.490	138.767.121,349	2.984,881	4.724,464	1	Submit Report	ZR_GET_PROGS
149.011.557	121.176.135,146	0,813	11,995	3	Submit Report	ZSQLM_TEST11
94.630.655	83.716.046,216	0,885	6,883	98	Submit Report	ZSQLM_TEST
47.003.031	51.064.956,906	1,086	6,683	22	Remote Func...	ZSQLM_CALLBACK_RFC
47.491.329	40.669.180,521	0,856	4,948	5	Submit Report	ZSQLM_TEST3
3.848.062	12.752.440,583	3,314	0,127	9	Remote Func...	TASK_VITAL_PERIOD
2.065.656	8.222.933,290	3,981	0,612	60	Submit Report	SAPMSSY2
2.745.375	7.004.852,445	2,552	2,566	22	Remote Func...	/SDF/IS_PROXY
2.333.032	5.805.330,380	2,488	2,892	108	Remote Func...	/SSF/CALL_SUBROUTINE_RFC
501.244	2.832.856,843	5,652	2,835	30	Remote Func...	/SDF/E2E_MDP_COLLECTOR
43.684	2.728.952,518	62,470	6.819,658	3	Submit Report	ZSQLM_TEST11_OPT
1.292	1.452.596,195	1.124,300	1.007,141	6	Submit Report	RSSNAPDL
223.530	1.167.952,071	5,225	16,362	99	Submit Report	SAPMSSY8
4.569	1.042.656,778	228,202	461,226	19	Submit Report	RSQLM_UPDATE_DATA
1.056.629	912.683,030	0,864	2,203	30	Remote Func...	SALT_TOOLSET_STARTER
13.775	909.981,182	66,060	5.485,636	4	Submit Report	ZSQLM_TEST3_OPT
36.410	899.975,138	24,718	730,000	11	Submit Report	ZSQLM_TEST4

Figure 1 - SQL Monitor with sample data

The statistical data is aggregated using the following keys:

- Process type (e.g. transaction, report ,RFC module, HTTP request)
- Entry point (e.g. transaction VA01)
- ABAP source code location of the statement (i.e. program, include, line)
- Database table names (e.g. MARA)

For each entry, the following statistics are available:

- Number of executions
- Execution time (maximum, minimum, average, standard deviation)
- Fetched/changed rows (maximum, minimum, average, standard deviation)
- Number of internal sessions that triggered the respective SQL statement

As a consequence, for every code path leading to an ABAP SQL statement which is executed at least once in the given time period, there is one entry in the SQL Monitor data containing statistical runtime information. This data can be used to display the performance data aggregated by business process or ABAP source code location. In addition, the data collected by the SQL Monitor can also be downloaded as a file. This allows for instance the data to be imported into the SQL Performance Tuning Worklist (see below) in a development system.

How to use the SQL Monitor

Before using the SQL Monitor within an ABAP system, a couple of configurations steps are required. In short, this includes the following activities

- Software deployment, see SAP Note 1885926

- Assign required authorizations (authority object S_ADMI_FCD)

For detailed information on how to set up and use the SQL Monitor, we refer to the document *"Optimizing custom ABAP Code for SAP HANA – the new ABAP SQL Monitor"* available in the ABAP for SAP HANA SCN community.

Afterwards, the SQL Monitor can be activated via transaction SQLM (respectively /SDF/ZQLM), which allows the activation of the SQL Monitor on all application servers (recommended) or on selected servers. In addition, an expiry date has to be defined where the tracing will be deactivated automatically. The default setting is to switch off the SQL monitor after 7 days. However, you may run it for longer (e.g. a month) or shorter (e.g. one day) time periods. As a general recommendation, you should aim for a time period which captures all the relevant business processes (running e.g. at the end of a month) but keep in mind that the data is aggregated and the data quality decreases whenever processes are modified (configuration or code changes).

The SQL Monitor has a minimal performance overhead as the data is collected first by the runtime monitor infrastructure in the ABAP kernel and asynchronously (via a scheduled job or triggered manually) persisted in the database. Snapshots of the data can be exported and imported to another system for further analysis or correlation with the results of a code inspector. For further information we refer to the documentation.

Typical usage scenarios

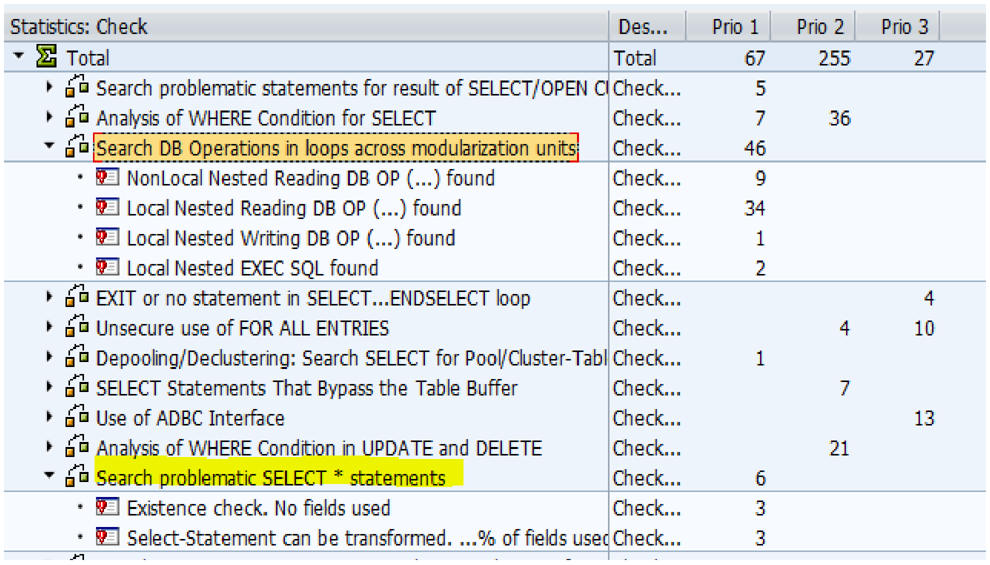
The following table gives an overview of typical usage scenarios for the SQL Monitor:

Scenarios	Goal	Procedure in SQL Monitor
Scenario 1 Get Business Process Overview	Get an overview of the main SQL driven requests	In the data selection in the SQL Monitor, choose aggregation by request (without filter or restriction of results). This displays a list of all entry points to processes (transactions, web applications, etc.) with their resulting SQL workload. • Sort this list by executions to get the requests which run most of the SQLs • Sort this list by Total time to get the processes with the highest SQL execution time, • Sort this list by Total Records to get the processes with the highest fetch/change row count. Now check the Top N of these rankings (you may additionally filter by business criticality of the process) and memorize the aggregated value of the measurement value you are interested in (e.g. Total time) Get the SQL profile of the process by drilling down and ranking again by the column of interest (e.g. Total Time). Analyze the top entries which contribute most to the total value you found on process level. Frequently, the TOP 5 SQL statements in this list contribute up to 80% to the measurement value of interest.
Scenario 2 System Overview	Get the most frequently executed and most expensive database accesses	In the data selection in the SQL Monitor, choose aggregation by source position (without filter or restriction of results) Sort the result list by Total time or Number of executions. When sorting by Total time, check the Min/max values of these top entries to figure out if e.g. only one SQL was extremely slow because it was waiting for the release of a lock or if almost all executions are slow. When sorting by Number of executions, you can use the drill down to focus on the top entries which have a high value in column "Executions/session" since these are most often SQL statements executed in Loops, which have optimization potential.
	Detect the SQL statements	In the data selection in the SQL Monitor, choose aggregation by source position (without filter or restriction of results).

	reading/writing the most data	Sort result list by Total Records. Check the Min/max values of selected records and look for unexpected patterns. If the maximum and minimum is always the same (and equals the total number of lines stored in the DB table) then this might be a SQL w/o where clause. If you see a big difference between Max and Min value then this might be a SQL which encounters an empty FOR ALL ENTRIES or empty RANGE table from time to time.
Scenario 3: Single Processes	Analyze selected processes for further optimization	In the data selection in the SQL Monitor, select the desired business processes (i.e. entry points). Drill down to the corresponding SQL profile(s) and analyze the top contributors as described above.

ABAP Test Cockpit / SAP Code Inspector
The SAP Code Inspector is the standard tool for analyzing ABAP developments to detect potentially critical source code in terms of performance, security, globalization, or quality in general. The code analysis is done by performing a set of selected checks which can be grouped into a *check variant*. The Code Inspector is integrated into the ABAP Test Cockpit, a universal tool for quality assurance of custom developments (see http://scn.sap.com/docs/DOC-31773 for details).

The existing checks in the ABAP code inspector have been extended in order to assist further during the transition and optimization for SAP HANA. This includes checks for functional issues which might occur during the migration as well as enhanced performance checks for detecting optimization potential.



Statistics: Check	Des...	Prio 1	Prio 2	Prio 3
Total	Total	67	255	27
Search problematic statements for result of SELECT/OPEN CI	Check...	5		
Analysis of WHERE Condition for SELECT	Check...	7	36	
Search DB Operations in loops across modularization units	Check...	46		
NonLocal Nested Reading DB OP (...) found	Check...	9		
Local Nested Reading DB OP (...) found	Check...	34		
Local Nested Writing DB OP (...) found	Check...	1		
Local Nested EXEC SQL found	Check...	2		
EXIT or no statement in SELECT...ENDSELECT loop	Check...			4
Unsecure use of FOR ALL ENTRIES	Check...		4	10
Depooling/Declustering: Search SELECT for Pool/Cluster-Tabl	Check...	1		
SELECT Statements That Bypass the Table Buffer	Check...		7	
Use of ADBC Interface	Check...			13
Analysis of WHERE Condition in UPDATE and DELETE	Check...		21	
Search problematic SELECT * statements	Check...	6		
Existence check. No fields used	Check...	3		
Select-Statement can be transformed. ...% of fields usec	Check...	3		

Figure 2 – ABAP Test Cockpit with data from sample check run

How to use the ABAP Test Cockpit
For the setup of the ABAP Test Cockpit and the ABAP Code Inspector we refer to the SAP standard documentation available on help.sap.com/nw74. In order to execute code checks, it is required to choose or define a check variant in the ABAP code inspector (transaction SCI). See SAP Note 1912445 for the recommended check variants for detecting ABAP code with potential adaptation needs.

Please note that the code analysis will only be able to point out ABAP source code where adaptation *may* be required, i.e. an actual code change is not mandatory in all situations. Hence the findings must be analyzed and the impact judged. “False positives” can be marked as exemptions to be excluded from future check runs.

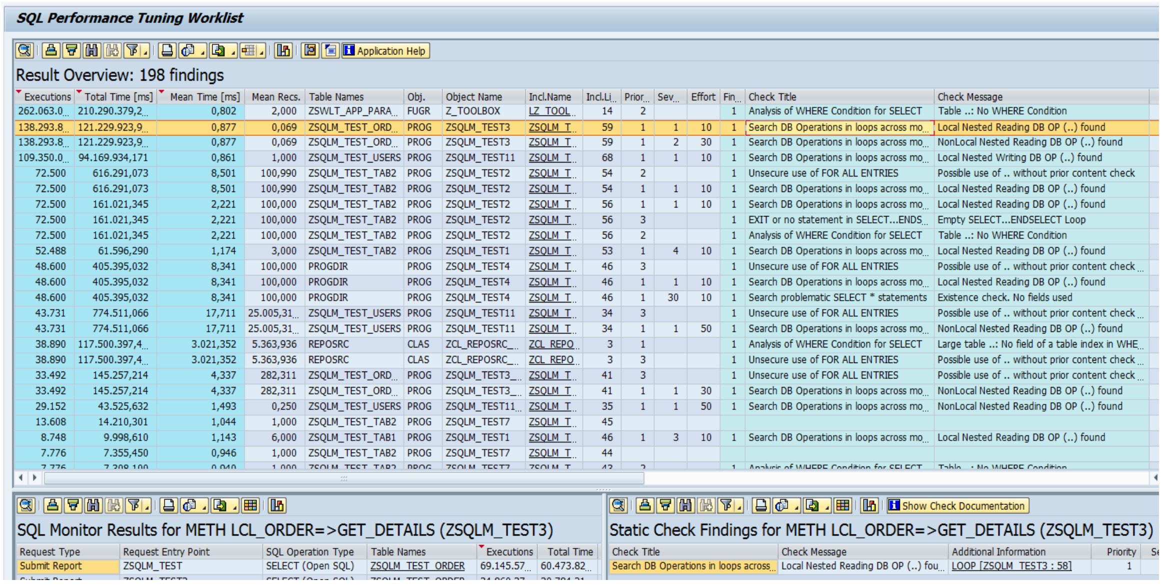
Typical usage scenarios

The following table gives an overview of typical usage scenarios for using the ABAP Test Cockpit during the transition to SAP HANA:

Scenario	Goals	Procedure
Prepare migration to SAP HANA (functional)	Find ABAP code requiring adaptation in order to ensure no functional regression	Use the ABAP Test Cockpit to perform a code inspection in a Development or Test system. See SAP Note 1912445 for the recommended check variants. All findings should be analyzed (unless it can be ruled out that a specific source code is still in use).
Prepare migration to SAP HANA (performance)	Find recommended adaptations related to performance	Use the ABAP Test Cockpit to perform a code inspection in a Development or Test system with check variant PERFORMANCE_DB (see also SAP Note 1912445) The findings should be analyzed in combination with collected runtime information in order to minimize efforts (see Section “Best Practices”).
Optimize for SAP HANA	Find further optimization potential after the migration	Use the ABAP Test Cockpit to perform a code inspection in a Development or Test system with check variant PERFORMANCE_DB (see also SAP Note 1912445). The findings should be analyzed in combination with collected runtime information in order to detect opportunities with high cost/benefit ratio (see Section “Best Practices”)

SQL Performance Tuning Worklist

The SQL Performance Tuning Worklist (transaction code SWLT) is a tool combining the result of a code analysis with relevant runtime data coming e.g. from a productive system. This combination of data makes it easy to generate a ranked work list for SQL performance tuning by analyzing and filtering the data set.



SQL Performance Tuning Worklist

Result Overview: 198 findings

Executions	Total Time [ms]	Mean Time [ms]	Mean Recs.	Table Names	Obj.	Object Name	Incl.Name	Incl.Li...	Prior...	Sev...	Effort	Fin...	Check Title	Check Message
262.063.0...	210.290.379,2...	0,802	2,000	ZSWLT_APP_PARA...	FUGR	Z_TOOLBOX	LZ_TOOL...	14	2			1	Analysis of WHERE Condition for SELECT	Table ..: No WHERE Condition
138.293.8...	121.229.923,9...	0,877	0,069	ZSQLM_TEST_ORD...	PROG	ZSQLM_TEST3	ZSQLM_T...	59	1	1	10	1	Search DB Operations in loops across mo...	Local Nested Reading DB OP (..) found
138.293.8...	121.229.923,9...	0,877	0,069	ZSQLM_TEST_ORD...	PROG	ZSQLM_TEST3	ZSQLM_T...	59	1	2	30	1	Search DB Operations in loops across mo...	NonLocal Nested Reading DB OP (..) found
109.350.0...	94.169.934,171	0,861	1,000	ZSQLM_TEST_USERS	PROG	ZSQLM_TEST11	ZSQLM_T...	68	1	1	10	1	Search DB Operations in loops across mo...	Local Nested Writing DB OP (..) found
72.500	616.291,073	8,501	100,990	ZSQLM_TEST_TAB2	PROG	ZSQLM_TEST2	ZSQLM_T...	54	2			1	Unsecure use of FOR ALL ENTRIES	Possible use of .. without prior content check
72.500	616.291,073	8,501	100,990	ZSQLM_TEST_TAB2	PROG	ZSQLM_TEST2	ZSQLM_T...	54	1	1	10	1	Search DB Operations in loops across mo...	Local Nested Reading DB OP (..) found
72.500	161.021,345	2,221	100,000	ZSQLM_TEST_TAB2	PROG	ZSQLM_TEST2	ZSQLM_T...	56	1	1	10	1	Search DB Operations in loops across mo...	Local Nested Reading DB OP (..) found
72.500	161.021,345	2,221	100,000	ZSQLM_TEST_TAB2	PROG	ZSQLM_TEST2	ZSQLM_T...	56	3			1	EXIT or no statement in SELECT...ENDS...	Empty SELECT...ENDSELECT Loop
72.500	161.021,345	2,221	100,000	ZSQLM_TEST_TAB2	PROG	ZSQLM_TEST2	ZSQLM_T...	56	2			1	Analysis of WHERE Condition for SELECT	Table ..: No WHERE Condition
52.488	61.596,290	1,174	3,000	ZSQLM_TEST_TAB2	PROG	ZSQLM_TEST1	ZSQLM_T...	53	1	4	10	1	Search DB Operations in loops across mo...	Local Nested Reading DB OP (..) found
48.600	405.395,032	8,341	100,000	PROGDIR	PROG	ZSQLM_TEST4	ZSQLM_T...	46	3			1	Unsecure use of FOR ALL ENTRIES	Possible use of .. without prior content check ...
48.600	405.395,032	8,341	100,000	PROGDIR	PROG	ZSQLM_TEST4	ZSQLM_T...	46	1	1	10	1	Search DB Operations in loops across mo...	Local Nested Reading DB OP (..) found
48.600	405.395,032	8,341	100,000	PROGDIR	PROG	ZSQLM_TEST4	ZSQLM_T...	46	1	30	10	1	Search problematic SELECT * statements	Existence check. No fields used
43.731	774.511,066	17,711	25.005,31...	ZSQLM_TEST_USERS	PROG	ZSQLM_TEST11	ZSQLM_T...	34	3			1	Unsecure use of FOR ALL ENTRIES	Possible use of .. without prior content check ...
43.731	774.511,066	17,711	25.005,31...	ZSQLM_TEST_USERS	PROG	ZSQLM_TEST11	ZSQLM_T...	34	1	1	50	1	Search DB Operations in loops across mo...	NonLocal Nested Reading DB OP (..) found
38.890	117.500.397,4...	3.021,352	5.363,936	REPOSRC	CLAS	ZCL_REPOSRC_...	ZCL_REPO...	3	1			1	Analysis of WHERE Condition for SELECT	Large table ..: No field of a table index in WHE...
38.890	117.500.397,4...	3.021,352	5.363,936	REPOSRC	CLAS	ZCL_REPOSRC_...	ZCL_REPO...	3	3			1	Unsecure use of FOR ALL ENTRIES	Possible use of .. without prior content check ...
33.492	145.257,214	4,337	282,311	ZSQLM_TEST_ORD...	PROG	ZSQLM_TEST3_...	ZSQLM_T...	41	3			1	Unsecure use of FOR ALL ENTRIES	Possible use of .. without prior content check ...
33.492	145.257,214	4,337	282,311	ZSQLM_TEST_ORD...	PROG	ZSQLM_TEST3_...	ZSQLM_T...	41	1	1	30	1	Search DB Operations in loops across mo...	NonLocal Nested Reading DB OP (..) found
29.152	43.525,632	1,493	0,250	ZSQLM_TEST_USERS	PROG	ZSQLM_TEST11...	ZSQLM_T...	35	1	1	50	1	Search DB Operations in loops across mo...	NonLocal Nested Reading DB OP (..) found
13.608	14.210,301	1,044	1,000	ZSQLM_TEST_TAB2	PROG	ZSQLM_TEST7	ZSQLM_T...	45						
8.748	9.998,610	1,143	6,000	ZSQLM_TEST_TAB1	PROG	ZSQLM_TEST1	ZSQLM_T...	46	1	3	10	1	Search DB Operations in loops across mo...	Local Nested Reading DB OP (..) found
7.776	7.355,450	0,946	1,000	ZSQLM_TEST_TAB2	PROG	ZSQLM_TEST7	ZSQLM_T...	44						

SQL Monitor Results for METH LCL_ORDER=>GET_DETAILS (ZSQLM_TEST3)

Request Type	Request Entry Point	SQL Operation Type	Table Names	Executions	Total Time
Submit Report	ZSQLM_TEST	SELECT (Open SQL)	ZSQLM_TEST_ORDER	69.145.57...	60.473.82...

Static Check Findings for METH LCL_ORDER=>GET_DETAILS (ZSQLM_TEST3)

Check Title	Check Message	Additional Information	Priority
Search DB Operations in loops across...	Local Nested Reading DB OP (..) fou...	LOOP [ZSQLM_TEST3 : 58]	1

Figure 3 –SQL Performance Tuning Worklist (SWLT) with sample data

How to use the SQL Performance Tuning Worklist

The SQL Performance Tuning Worklist combines the static checks which normally run in a Development, Test or Quality system with runtime data from the SQL Monitor (e.g. obtained via the SQL Monitor) running in the productive system.

For the setup of the SQL Performance Tuning Worklist (required authorizations, etc.) we refer to the SAP standard documentation available at help.sap.com/nw74. In order to use the SQL Performance Tuning Worklist, run transaction SWLT.

The start screen offers the following options:

- **Select object set:** Select the object set (e.g. limit to Z*, Y*, etc.) for limiting the number of the retrieved check results / SQL monitor data.
- **Select runtime data:** Select SQL monitor data snapshot
    - *[Optional]* Coverage Analyzer: If no SQL monitor data is available, you can use code coverage data (obtained e.g. via the Usage and Procedure Logging; UPL) from a remote system to get at least the number of executions on the level of modularization units. In particular, this allows ruling out coding which is never invoked at all.
- **Select code analysis results:** Choose a Code Inspector or ATC result by providing the Code Inspector inspection or the ATC run series. SWLT will always choose the latest version of the available results.

The results from the code analysis or SQL Monitor data can be imported via RFC or a local file into SWLT; hence the data sources are decoupled from SWLT. Consequently, SWLT can run in any system but we recommend using it in the same system where the static checks were running.

Typical usage scenarios

The following table gives an overview of typical usage scenarios for the SWLT:

Scenario	Goals	Procedure in SWLT
Plan selected performance optimizations before migration	• Analyze recommended performance optimizations within the system • Find performance bugs and simple adjustments	Select SQL Monitor snapshot (resp. coverage analyzer data) and code inspection (ATC run series or SCI inspection). Look for performance bugs and findings where simple, compatible changes can be applied. Typical examples are listed in section "Performance related adaptions".
Plan performance optimizations after migration	• Analyze selected business processes for further optimization potential. • Detect unexpected database access paths • Judge complexity of code changes and optimized alternative implementations	Select SQL Monitor snapshot and code inspection (ATC run series). Filter data by those business processes which are interesting for optimization. Look for findings where many records are fetched in the application server, and total execution times are high. If required changes are non-local (e.g. not only one SQL statement or internal method implementation is affected), you can judge the complexity by analyzing the different access paths to the specific database access. Depending on business criticality, it may be sufficient to optimize only selected access paths (e.g. in the context of a selected business process).

REQUIRED AND RECOMMENDED ADAPTIONS

Overview
SAP HANA is supported by SAP NetWeaver as a fully compatible extension of the Platform Availability Matrix (PAM), see http://service.sap.com/pam. As such, no standard ABAP code using only database independent (and documented) features requires any adaptation in order to run on SAP HANA. Furthermore, the existing performance guidelines also remain valid on SAP HANA (see references). However - as with every database platform change - several aspects related to custom ABAP code should be considered. We will give an overview of the required and recommended considerations for the transition to SAP HANA.

Categories
In this section, the main areas concerning custom code optimization during a migration to SAP HANA are summarized. These corrections are necessary to ensure functional correctness of the custom code after the migration. For almost all aspects, there are dedicated checks in the SAP Code Inspector for detecting ABAP coding which may require further analysis and potential adaptation. In the section **"Best Practices",** we will come back to recommendations on how to effectively prepare, analyze and execute the required adjustments.

Database migration related adaptations
The migration to SAP HANA is first of all a normal database migration. This implies that existing ABAP code has to be checked for usage of specific functionality of the previous database. This includes first and foremost the following two topics:

- **Usage of native SQL (via EXEC SQL or ADBC):** When using native SQL in ABAP, there is no guarantee regarding portability to other database platforms. It has to be manually verified that the used native SQL statements are also supported on SAP HANA, or if certain adaptation is required. Please consult the SAP HANA SQL Reference available at http://help.sap.com/hana_platform for details of the supported SQL dialect of SAP HANA.
- **Usage of database hints:** Database hints provide a means to pass additional information to the database interface, DBSL or the underlying database system in order to allow these components to optimize certain data access. See SAP Note 1622681 for the supported DBSL hints for SAP HANA.

In addition, there are two further aspects which are specific to the migration to SAP HANA and which may also require special attention for custom ABAP code. Both (rather exotic) cases refer to unsafe usages of internal functionality.

- **Access to physical table pools/clusters:** Table pools and table clusters are special table types in the ABAP Dictionary, which can store data from several different (logical) tables together in a physical database table. The contained logical tables are accessed via standard Open SQL; a direct access to the underlying physical database tables is not officially supported. During the migration to SAP HANA, pool/cluster tables are transformed into transparent tables (i.e. into standard database tables), which are compatible for normal Open SQL access (see next section for remarks on sorting behavior). If, however, the internal physical pools/clusters are accessed in custom ABAP code, adaptation is required.
- **Checking for existence of indices:** If existing ABAP code checks for the existence of database indices (e.g. via the function module DB_EXISTS_INDEX), you should be aware that during the migration to SAP HANA exclusion entries are generated for existing secondary indices as they are not needed on SAP HANA in most cases (see SAP Note 1794297 for further information).

Last but not least, we would like to remind you that SAP HANA only supports Unicode installations. Hence, if required, it is mandatory to perform a Unicode migration including the required adaptation of existing ABAP-custom code. We will come back to recommendations in section **"Best Practices".**

Functional related adaptions
In order to ensure a smooth migration to SAP HANA; it is crucial to ensure that existing ABAP code behaves the same after the migration as before. While the support for SAP HANA is fully compatible based on the specified and documented feature set of Open SQL, ABAP code relying on *implicit* sorting behavior (i.e. undocumented sorting behavior in Open SQL) may require adaptation.
The Column Store in SAP HANA (which is the recommended default storage for all business data) uses an internal memory layout, which is highly optimized for column operations such as search and aggregation.

This data layout is different from other database systems and one of the differentiating factors of SAP HANA. One consequence of this architecture is that there is no default sorting of a result set for a SELECT statement, while on other database systems supported by the SAP NetWeaver AS ABAP records are often sorted by the index that has been used to access the data (primary key, or a secondary index chosen by the database optimizer). However, as the usage of a specific index is not guaranteed in any database system and may change as soon as new indices/statistics are created, the program logic must not rely on this sorting in any case.

Hence, as described in the Open SQL documentation, whenever a result set should be sorted, the sorting criteria have to be defined using the ORDER BY clause. As a special case, the sorting behavior of accesses to previous pool/cluster tables should be considered. Also for such access, there is no guaranteed default sorting if no ORDER BY clause is specified. The mentioned ABAP Code Inspector variant can assist in detecting such ABAP code (see SAP Note 1912445).

Note that potential consequences of a different sorting can cause subtle effects after the migration, e.g. when accessing an internal table containing the result set with READ TABLE (in particular when using the addition BINARY SEARCH which relies on a sorting by the specified key). SAP recommends adapting these coding issues (at least for coding, which is still in use) independently of a migration to SAP HANA.

Performance related adaptions
The classical performance recommendations for database access from ABAP (standard Open SQL and native SQL) are usually summarized in five rules:

1. Keep the results sets small.
2. Minimize amounts of transferred data.
3. Minimize number of data transfers.
4. Minimize the search overhead.
5. Keep (unnecessary) load away from the database.

In general, these rules remain valid for SAP HANA as general guidelines. However, there is a shift of priorities with some coding recommendations becoming even more important (e.g. avoiding nested SELECT statements) while others are becoming less important (e.g. index support). For more details, we refer to the updated performance guidelines (see references at the end of this document).

Three general, common optimization patterns with high potential benefit on SAP HANA are displayed in the following table. Again, the ABAP Test Cockpit provides assistance in localizing corresponding source code.

Existing code patterns	**Possible change**	**Corresponding Rule**
• Nested SELECT statements • INSERT/UPDATE/DELETE statements within Loops	• Proper usage of Joins or FOR ALL ENTRIES • Replacing single operations with array operations • Avoiding redundant SELECTs (e.g. selecting data before a DELETE)	Minimize number of data transfers
• SELECT * FROM …	Reduce the number of selected columns by specifying a field list (particularly important for pure existence checks).	Minimize amounts of transferred data
• Missing or inadequate table buffer configuration • SELECT statements bypassing the table buffer	Use the ABAP table buffer when applicable for a table, and avoid bypassing the buffer.	Keep (unnecessary) load away from the database

SAP is working on extending the database programming capabilities (e.g. Open SQL) in ABAP 7.4, which can also be used for reducing the amount and number of data transfers. Please refer to the latest roll-out information and documentation for additional information on the supported features.

BEST PRACTICES

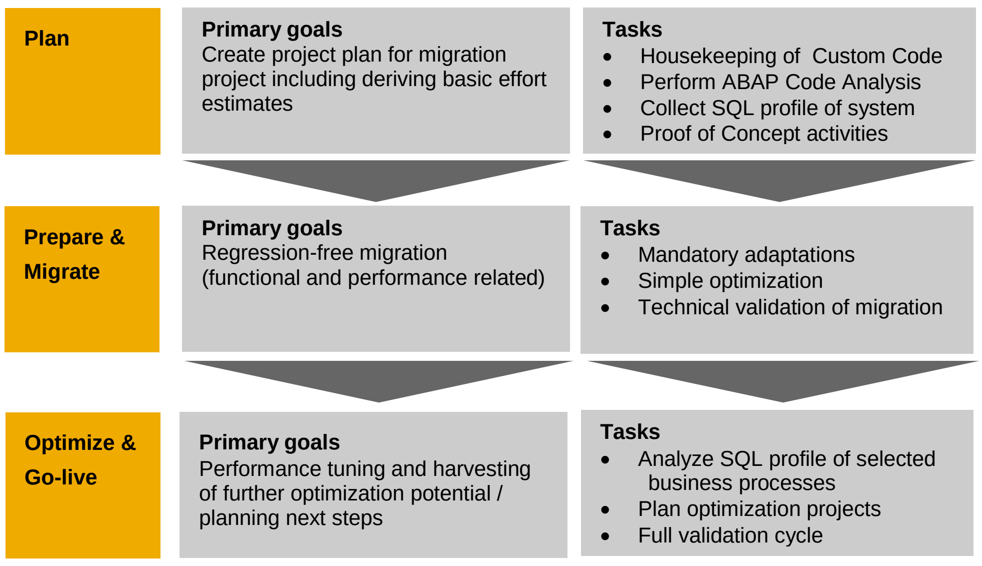
Overview

In this last section, we will summarize the considerations for Custom ABAP Code when transitioning to SAP HANA in the form of Best Practices. The focus will be on a simple process model explaining recommended steps and possible approaches, which can be used as a methodology when planning an migration project. We will explain how to make use of the tools presented in the section “Tools and Prerequisites”.

For the technical migration process (including preparation steps such as Unicode conversion and dual-stack split), we refer to the End-to-End Implementation Roadmap for SAP NetWeaver AS ABAP 7.4 based on SAP HANA and additional Best Practice documents (see references).

Process Model

The following process model might serve as an anchor for the recommended activities before, during and after the migration to SAP HANA in relation to custom ABAP code. It is not mandatory to perform all of the mentioned steps exactly in the order given.



Plan

Primary goals
Create project plan for migration project including deriving basic effort estimates

Tasks
- Housekeeping of Custom Code
- Perform ABAP Code Analysis
- Collect SQL profile of system
- Proof of Concept activities

Prepare & Migrate

Primary goals
Regression-free migration (functional and performance related)

Tasks
- Mandatory adaptations
- Simple optimization
- Technical validation of migration

Optimize & Go-live

Primary goals
Performance tuning and harvesting of further optimization potential / planning next steps

Tasks
- Analyze SQL profile of selected business processes
- Plan optimization projects
- Full validation cycle

Figure 4 – Basic process model for custom code adaptations

In the next sections, we will give a detailed overview of the different phases and recommended tasks.

Planning Phase

The main goal of the preparation phase related to existing Custom ABAP Development is to get an overview of the required adaptation efforts. As mentioned before, in the context of this document, we will primarily focus on adaptation related to the migration to SAP HANA and not on all potentially required adaptation of custom modifications. Furthermore, the planning phase may be required to create some small proof of concepts in order to demonstrate the business value of certain adaptations (e.g. in a sandbox environment).

When planning a migration project, it makes sense to combine this with certain “housekeeping” activities for the existing custom code base. In particular, having a consolidated view of productively used custom developments (and their respective status) will help tremendously for the other activities outlined below. The ALM process *Custom Code Management* in the SAP Solution Manager, and specifically the Usage Procedure Logging (UPL), can assist in finding usage patterns. Together with the SQL Monitor tool, this allows deriving a system profile as displayed in schematic form in Figure 5.

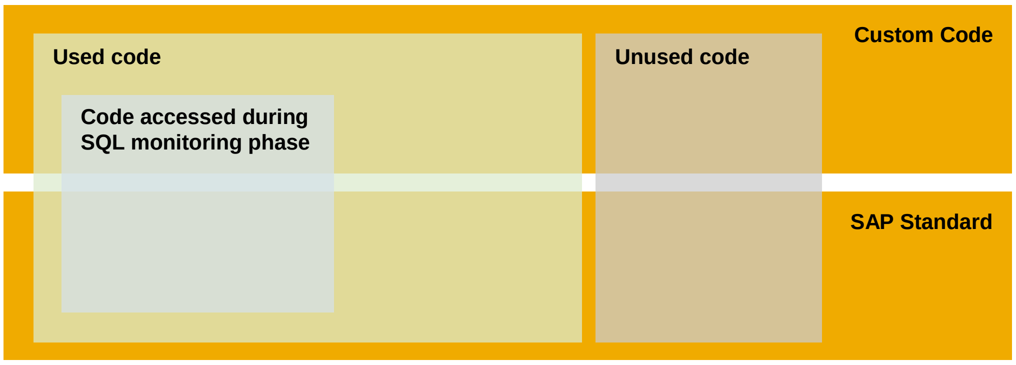


Figure 5 – Possible distribution of ABAP source code

As another part of the planning phase, it should be checked whether all installed components (including e.g. SAP or partner Add-Ons are supported on SAP HANA). Please consult SAP Notes 1812713 (NetWeaver), 1760306 (ERP), 1855666 (3rd party) for compatibility information. As SAP HANA supports only Unicode installations, a Unicode Conversion might be a part of the transitioning project. We refer to SAP Note 1051576 for recommendations related to Unicode conversion.

The procedure for analyzing existing custom ABAP developments in relation to required and recommended adaptions depends to some extend on the existing system landscape (SAP NetWeaver release, service pack level). The prerequisites for using the new and improved tools are listed in Section "Tools and Prerequisites". In case the prerequisites for certain tools cannot be met (for certain systems), there are still options to work based on reduced data sets.

For instance:

- If no SQL Monitor data is available, data from the Usage Procedure Logging (UPL) can be used to distinguish used code from unused code.
- If the ABAP Test Cockpit is not available (or specific improved code checks are not available), the ABAP Code Inspector with the existing checks can still assist in finding some issues.
- If the SQL Performance Tuning Worklist is not available, then the runtime data and check results need to be analyzed independently.

Note that since the output of these tools can be extracted and transferred to other systems, there are options to carry out the analysis of the results in dedicated systems with a higher SAP NetWeaver release. For instance, SQL Monitor data can be collected (using the ST/PI Add-On) in a productive SAP NetWeaver 7.00 system, and afterwards analyzed in the SQL Performance Tuning Worklist in an SAP NetWeaver 7.31 system together with the findings of a code analysis performed in the same system.

In relation to required and recommended adaption during the migration to SAP HANA we recommended planning the following activities:

1. Install and configure the SQL Monitor tool in all relevant systems.
2. Run SQL Monitor in productive system(s) to collect SQL runtime data. Usually a timeframe of 1-4 weeks is recommended. Capture the data as a snapshot.
3. Schedule ATC check-runs for source code analysis
   a. Check existing custom ABAP code for mandatory adaptation
   b. Check existing custom ABAP code for optimization candidates
4. Analyze the ATC findings with mandatory adaptation
   a. The number of findings may serve as a first indicator for the required effort.
   b. As a rule of thumb, findings related to existing database specific implementations (primarily native SQL) are significantly more expensive to adapt compared to findings related to the sorting behavior.
5. Correlate the SQL Monitor results with ATC check results in the Performance Tuning Worklist
   a. Pre-Analyze the SQL Monitor results (plausibility checks)
   b. Import the SQL Monitor snapshot and ATC result data into the SWLT
   c. Rank the findings by different criteria and select candidates.

6. Estimate and plan adaptations
    a. Estimate the impact of optimizations (e.g. determine theoretical lower boundaries or create dedicated prototypes in a sandbox environment),
    b. Choose findings with best cost/benefit ratio.

Migration Phase
The migration phase covers all activities during the actual migration project. In this section we describe the tasks related to custom ABAP developments that should be planned for. The optimal approach and order of the steps may depend on concrete circumstances at customer side, and a generic guidance is difficult. It may for instance be beneficial to carry out several housekeeping activities as identified in the planning phase prior to the start of the migration (maybe in combination with other activities such as adaptation related to Unicode conversion). This may include e.g. isolating or replacing previous database specific implementations with database independent implementations (if possible), and fixing source code relying on internal or unspecified behavior (see previous sections).

In particular in this phase, SAP service offerings can assist in safeguarding the migration project in order to avoid unnecessary activities. In this sense, the following list should be seen more as a checklist, and not a step-by-step guideline.

1. Perform adaptations related Unicode migration (if required)
2. Perform adaptations related to Business Suite / SAP NetWeaver Upgrades (e.g. using SPAU)
3. Perform adaptions related to SAP HANA migration (see Section **“Required and Recommended Adaptions”**)
    a. Carry out the mandatory adaptation related to database migration
    b. Carry out optimization as defined in the Planning Phase.
4. Perform technical validation
    a. As with any migration project, a full validation cycle of all business processes is needed. In case further optimization is planned after the technical migration, this complete integration test should take place after these additional adaptations. In this case, we recommend to at least performing a technical validation via automated tests, etc. to ensure that the migration has not caused severe functional regressions.
    b. In addition, whenever switching on Optimizations in SAP Business Suite powered by SAP HANA a re-test of related modifications and extensions is recommended.

Optimization Phase
The optimization phase starts after the successful migration to SAP HANA and covers aspects of further performance optimization of custom developments before the planned Go-live. Optimization can range from selective, local tuning activities to larger changes leveraging dedicated SAP HANA capabilities (e.g. usage of fuzzy search in SAP HANA). In the context of this document we will focus primarily on dedicated performance optimization of selected business processes without changing the existing application logic. For new opportunities for extending existing ABAP developments with SAP HANA capabilities, please refer to guides and tutorials in the ABAP for SAP HANA SCN Community available under http://scn.sap.com/community/abap/hana.

During the optimization phase the most important step is to select the concrete business processes to be considered for optimization.

Business Process selection: The selection of business processes to be considered for optimization depends on multiple criteria. The following list contains some aspects and questions which should be considered during the selection phase.

1. **Identification of business need**
    - What is the business benefit of an optimization of a certain process?
    - What are the most urgently requested optimizations by the business users?
    - Are there opportunities for simplification (e.g. avoiding background processing, usability improvements)?
2. **Analysis of optimization potential**
    - How is the current process structured?
    - Where does it rely on massive data processing resulting in high database load or calculation logic in the ABAP application server?

- What is the contribution to the end-to-end response times and how big is the optimization potential?

3. **Estimating complexity of changes**
    - Which parts of the custom development would be affected?
    - Is a native SAP HANA implementation required?
    - Can the optimization be realized without disconnecting from the SAP standard implementation?
    - Are stepwise approaches possible?

For the second and third step it is possible to use the tools described in this document as one important input channel. As outlined in the section “Planning Phase”, it is possible to use the SQL Performance Tuning Worklist based on runtime data collected via the SQL Monitor data and code analysis with the ABAP Test Cockpit. Note that during the migration phase, the ABAP system is running on the new ABAP release 7.4 and hence, the newest versions of the tools can be used. However, before looking in detail into the SQL profile of the process and possible optimization, it is advised to also analyze the distribution of runtimes between database and application server (plus potentially user interface or network aspects). Otherwise even significant optimization of the database access may not produce the expected improvements of the end-to-end response time.

In addition, the *Business Scenario Recommendations* report, which can be requested via the SAP Business Suite powered by SAP HANA homepage, can help in the business process selection.

Optimization Design and development
When designing an optimization of the database access, a stepwise approach is recommended:

The first step should be to perform optimization by applying the performance guidelines for ABAP database programming on SAP HANA (see section “Performance related adaptions” for some basic optimization patterns). Use transaction SWLT as described in section “SQL Performance Tuning Worklist” (Typical usage scenarios) in order to select the appropriate code segments. It is recommended to validate the performance improvements based on test data of realistic size.

In case the first step does not yield the expected results, specifically a larger change of the existing implementation is required; it makes sense to challenge the existing design and look deeper into the ABAP coding in order to understand which data is fetched (or changed) from the database into internal tables and which data is actually needed in the business process step (e.g. for the user interface). This may give an indication where some part of the application (creation of a result structure for display purposes) can be calculated on the fly in SAP HANA. Such changes to the application logic usually require some redesign efforts. For some techniques for applying this “*code-pushdown*” approach can be found in the SCN community http://scn.sap.com/community/abap/hana.

SUMMARY

SAP HANA is a compatible extension of the Product Availability Matrix of SAP NetWeaver. During the migration to SAP HANA, in particular in the context of the SAP Business Suite, certain consideration for Custom ABAP Code is recommended. These consist of mandatory adaptations related to a database migration in general and certain additional recommended adaption to benefit from the SAP In-Memory Columnar database. For detecting ABAP source code where adaptation is required or recommended , there are new and enhanced tools in the Application server ABAP which can be applied before the start of a migration project (in particular during the planning phase).

We strongly recommend using SAP service offerings e.g. by SAP Active Global Support and SAP Consulting for planning and performing a migration or optimization project (see Section "Relation to other information sources / services"). For existing partner solutions, we recommend participating in the offered certification programs.

Finally, here is a short reference of the most important SAP Notes and further information related to the contents of this document.

Central SAP Notes

- 1885926: "ABAP SQL Monitor"
- 1912445: "Code Inspector variants related to SAP HANA migration"
- 1794297: "Secondary Indexes for the business suite on HANA"
- 1785057: "Recommendations for migrating suite systems to SAP"
- 1622681: "DBSL hints for SAP HANA"
- 1847431: "SAP NetWeaver BW ABAP Routine Analyzer"
- 1051576: "Conversion of Single Code Page Systems to Unicode"
- 1812713: "Add-on compatibility of SAP NW AS ABAP 7.4 for Suite"
- 1760306: "SAP EhP6 for SAP ERP 6.0, version for SAP HANA 1.0"
- 1855666: "Suite on HANA: 3rd party Add-ons"

References

In addition to the SAP Notes mentioned above, here are some general references:

(1) SAP NetWeaver AS ABAP 7.4:
  a. Technical Brief
  b. Standard documentation
  c. End-to-End Implementation Roadmap for SAP NetWeaver AS ABAP 7.4 based on SAP HANA
  d. ABAP 7.4 Ramp-up Knowledge Transfer material for Development Consultants (restricted access)

(2) SCN Community for ABAP Development on SAP HANA
(3) Performance recommendations for ABAP development on the SAP HANA: database
(4) SAP Business Suite Powered by SAP HANA Integration Certification
(5) Best Practice Guide - Classical Migration of SAP NetWeaver AS ABAP to SAP HANA
(6) SAP HANA Cookbooks
(7) SAP Services & Support Overview
(8) SAP Business Suite powered by SAP HANA

APPENDIX

Availability of specific SAP Code Inspector checks

The following table contains a detailed list of the new and improved Code Inspector checks:

Area	Check title	Availability (SAP_BASIS)
Functional/ Robust Code	Search problematic statements for result of *SELECT/OPEN CURSOR* without *ORDER BY*	740 SP3 (SP2 via SAP Note 1875529) 731 SP9 702 SP14
	Depooling / Declustering: Search SELECT for Pool/Cluster-Tables w/o ORDER BY	740 SP2 731 SP9 702 SP14
	Unsecure use of *FOR ALL ENTRIES*	740 SP2 731 SP5 702 SP12 701 SP14 700 SP29
Performance	Search problematic *SELECT* * statements	740 SP2 731 SP9 702 SP14
	Search *SELECT* <..> *FOR ALL ENTRIES*-clauses to be transformed	740 SP2 731 SP9 702 SP14
	Search DB Operations in loops across modularization units (reading and changing SQLs) **Remark:** for different releases, the capabilities for call-stack analysis differ. Please refer to the system documentation.	740 SP3 731 SP9 702 SP14
	Search *SELECT* statement with *DELETE* statement	740 SP2 731 SP9 702 SP14

www.sap.com