Performance Tuning

For SAP BW

BUSINESS INFORMATION WAREHOUSE

Performance Tuning for SAP BW
Document Version 2.6 (December 3, 2003)

Please Download This Document Regularly As It Might Be Subject To Changes

Table of Contents

1 Introduction

The performance aspects of a data warehouse project should be of primary consideration, as performance is directly related to the degree of user acceptance of the solution, as well as the overall effectiveness of its analytical capabilities. Every project phase should be reviewed subject to performance: beginning with the data model, through to fine tuning of the database and ending with proactive operational maintenance mode (e.g. review aggregates, build new ones, delete unused ones, find problem queries) as an ongoing cycle.

This document gives an overview on all features and settings which impact performance. It also helps to use all tools provided by SAP BW to identify performance problems.

1.1 Strategic Performance Management

Performance is a very difficult and interdependent topic. In complex scenarios it is not possible to optimize all single processes. The objectives must be defined and prioritized in advance. We recommend the approach of Strategic Performance Management.

Strategic Performance Management (SPM) is an emerging discipline that is becoming widely recognized as the next evolution in performance management by META Group, GartnerGroup, Bill Inmon and forward-thinking industry leaders. Unlike tactical performance management, which operates from a bottom-up approach, SPM views performance from a top-down approach. SPM addresses questions such as:

- Who are your most critical users?
- How is the system being used?
- What data is being most frequently accessed?
- How is that data being used or consumed?
- How is my IT infrastructure being used by different lines of business (LOBs)?
- Is my IT infrastructure performance being maximized on an ongoing basis?
- Can I measure the ROI of my investment?
- Is my organization able to make their most critical decisions in a timely fashion?
- When will I run out of capacity and what are the financial implications?
- What are my alternatives - both in terms of performance and financial implications?

Tactical products don't provide these answers because they focus on discrete elements (CPU, disk, network, SQL, data, etc.). SPM takes into account all of these elements and provides a correlated view or systemic approach to performance improvement. SPM understands that improving and managing performance must be encompassing. SPM empowers the organization to focus on the root cause(s) of bottlenecks.

When faced with improving the performance of a system, companies must weigh different alternatives. Should additional nodes be added? Should additional disks be balanced across more nodes? Should an additional server system split the workload? Alternatives have price/performance/ROI tradeoffs that should be understood. Once understood, expectations can be set and measured upon implementation.

SPM allows you to understand performance as it relates to different lines of business. Knowing which users belong to various lines of business, how they are using the system, what data they are accessing and how that data is being used are paramount to improving performance. For more information see http://www.bez.com.

1.2 Target Group

If you are skilled in BW, use this document as a checklist to ensure that you covered all areas and did not forget some important points.

If you want to understand the topic 'performance', the possibilities, limitations and want to know about interdependencies, then browse into this document and read the linked documents. If you are new to BW, or to the technical architecture of SAP systems, please consider attending training courses such as BW310 and BW360 (see http://service.sap.com/education for more details).

Projects are always encouraged to engage the services of SAP's consulting organizations, and to work closely with SAP's support organizations as ongoing resources in managing the operational and performance aspects of productive BW systems. The functionality described in this document is for use by skilled consultants and advanced technical team members. Please do not attempt to work with features that could impact the efficiency of the system, in areas where you have little or no training or experience.

Further information on performance can be found in SAP Service Marketplace under the alias bw.

1.3 How To Read This Document

This document describes all BW features and settings with respect to performance. In each chapter, special sections can be found:

These sections tell you tips and tricks and experiences made in other projects.

These sections tell you about restrictions, dependencies to other features and special things to observe.

1.4 Software Version Supported

This document was written specifically for SAP BW versions 3.x and 2.0B/2.1C, but should apply to all versions of BW.

SAP generally recommends staying as current as possible with all types of support packages, as incremental performance optimizations are included in support packages. In particular, the BW support packages released in January 2003 (namely BW 3.0B SP9) included some important improvements/features that are described below.

2 Basic Settings

This chapter describes the most basic performance settings, which MUST be applied to every BW system, and is complemented by a checklist of the most important issues you should consider before a BW implementation.

Note that we strongly recommend applying the current support package (for BW 3.0B at least SP9) if you detect any performance issues.

2.1 Database and system parameters

Database and system parameters define sizes of DB memory, sizes of disk areas, behavior of the DB optimizer, logging, processes etc.

These parameters must be in sync with the SAP recommendations when installing the system. After the installations, some parameters can't be changed without high effort (e.g. DB block size).

Check the following SAP note for initial basis settings for your BW system:

SAPnote 192658: Setting basis parameters for BW systems

The number of dialogue processes should be by one higher than the other processes (see SAPnote 74141: Resource management for tRFC and aRFC for more details).

Note that a BW application server does not need an UP2 process and not more than 2 UPD processes.

Check the following SAP notes for the respective underlying DB management system:

SAPnote 180605: Oracle database parameter settings for BW

SAPnote 302429: DB2 UDB EEE: Performance on BW 2.0B, 2.1C, 3.0A

SAPnote 546262: DB6: Performance on SAP BW 3.0B

SAPnote 307077: AS/400: Performance optimizing in BW Systems

SAPnote 501572: iSeries: EVI stage 2 support

SAPnote 541508: iSeries: Checking the system parameters

SAPnote 390016: DB2/390: BW: DB settings and performance

SAPnote 181945: Performance guide: BW on Informix

SAPnote 327494: Configuration Parameters for SQL Server 2000

SAPnote 28667: Microsoft SQL Server Specific Profile Parameters

2.1.1 Oracle-specific DB settings

In some cases, the accesses to data dictionary information can be improved by defining indices and views and enabling CBO features (such as hash joins in contrast to nested loops) by having statistics.

Please apply the attached notes in your system. They CAN improve the performance in some cases significantly (e.g. if you have a lot of partitions in your F table):

SAPnote 519448: Performance problems when deactivating aggregates

SAPnote 519407: Performance problems when deactivating aggregates

SAPnote 565416: Performance during ORACLE dictionary accesses is slow

SAPnote 558746: Better ORACLE Data Dictionary BW Performance

SAPnote 565075: Recommendations for BW systems with ORACLE 8.1.x

2.1.2 Settings when using Web Frontend

If you are using Web Applications in BW 3.x, be sure that HTTP compression and the Browser Cache are activated. As of BW 3.0B SP9, HTTP compression is automatically active. Check the following SAP notes and see below for more details:

SAPnote 550669: Compressed transfer of BW Web Applications

SAPnote 561792: Client-sided caching of image/gif files

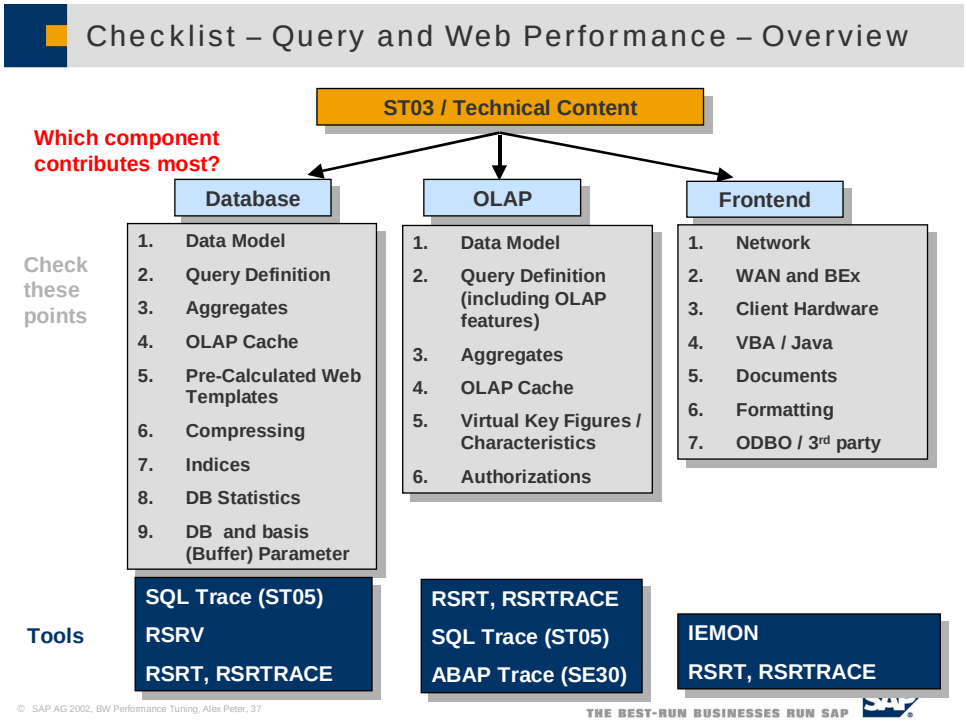
2.2 Checklist

The following slides show how to approach performance problems. The top item shows the central point of entry for detecting performance issues. The area which contributes most to the long runtime can be identified here; this is shown in the second level.

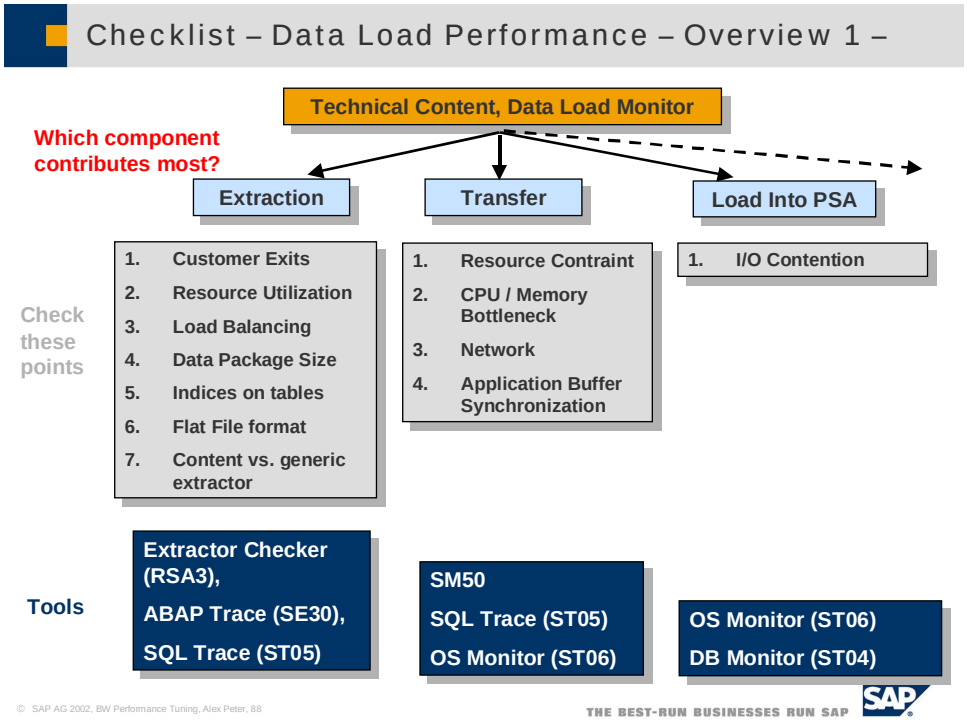
Check all items in the boxes below the problem area. Explanation for each item can be found in this document.

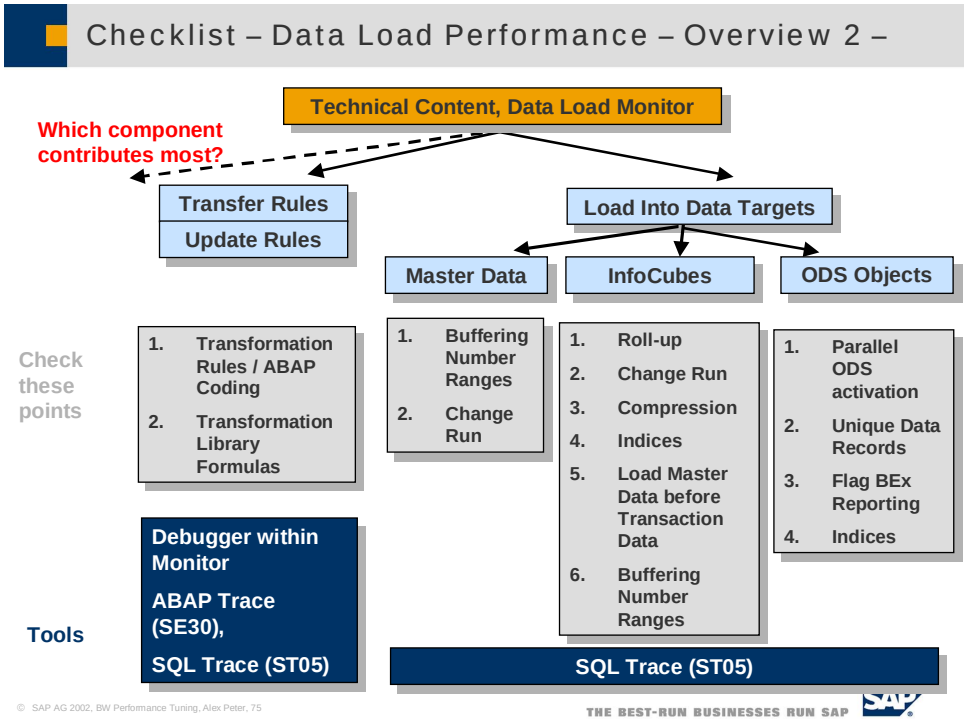
The bottom line tells you which tools are available to find the problem. These tools are also explained in this document.

2.2.1 Query and Web Performance



2.2.2 Upload Performance





3 General Performance-influencing Factors

3.1 Design

3.1.1 Data Modeling

A data model defines the architecture of the layers, the InfoObjects and the InfoProviders; this definition includes

- ODS objects vs. InfoCubes
- MultiProvider / Logical Partitioning
- InfoCube dimensions (including line-item dimensions and time dimension)
- Dimension characteristics and navigational attributes
- Time-dependent objects
- Non-cumulative key figures
- Hierarchies

InfoCube modeling is the process by which business reporting requirements are structured into an object with the facts and characteristics that will meet the reporting needs. Characteristics are structured together in related branches called dimensions. The key figures form the facts. The configuration of dimension tables in relation to the fact table is denoted as “star schema”.

Do not combine dynamic characteristics in the same dimension in order to keep dimensions rather small: for example, don’t combine customer and material in one dimension if the two characteristics are completely independent. As a general rule, it makes more sense to have many smaller dimensions vs. fewer larger dimensions. Dimension tables should be sized less than 10% of the fact table.

MultiProviders can access several InfoProviders (InfoCubes, ODS objects, InfoObjects and InfoSets) and consolidate data from all of them. If you group your data on a given characteristic (such as year, region, plan/actual), you can combine the data via a MultiProvider.

Use MultiProvider (or logical) partitioning to reduce the sizes of the InfoCubes; e.g. define InfoCubes for one year and join them via a MultiProvider (advantages: parallel access to underlying basis InfoCubes, load balancing, resource utilization, query pruning). See chapter on MultiProviders for further information

Combining similar InfoCubes into a MultiProvider may cause additional maintenance (update rules, aggregates).

The more basis InfoProviders are used the more I/O load will be generated.

Navigational attributes are part of the “extended star schema”. Navigational attributes require additional table joins at runtime (in comparison to dimension characteristics) – but usually the decision for dimension characteristics or navigational attributes is based on business requirements rather than on performance considerations. If your aggregates contain navigational attributes, a change run must be scheduled every time this master data is loaded.

Dimensional attributes generally have a performance advantage over navigational attributes for query operations. Fewer tables are joined to access the values of dimensional attributes than navigational attributes; therefore less overhead is seen during query execution. The requirements should be carefully analyzed to determine whether a certain attribute needs to be navigational or if dimensional or display attribute will suffice

Time-dependent characteristics store characteristic values together with a validity period. At query runtime, a given key date determines the ONE value that is/was valid at this date. Aggregates on time-dependent navigational attributes can be defined as of BW 3.0; the aggregates are also stored at a key date (fix value or variable). The aggregates have to be adjusted when the key date changes; this adjustment can be scheduled via a process type in the BW 3.0-process chains.

Time-dependent navigational attributes have impact on the use of aggregates. Either no aggregates can be used or aggregates have to be realigned regularly when the key date changes, which is an expensive process. If time-dependent master data is really necessary, consider modeling the attribute twice: as time-dependent and time-independent. Prefer the time-independent in your queries if possible.

Degenerated dimensions represent a potential performance pitfall: If a high-cardinality characteristic (high number of distinct values) is included in the dimension table, the dimension table becomes very large. Thus the join operation that takes place at query runtime experiences a significant overhead when joining a large dimension table with a large fact table.

In the data modeling phase, it is very important to determine if a dimension table will be degenerated, and then explicitly set it as a line item dimension (parameter setting on the InfoCube’s dimension entry). In this case, the dimension table is omitted and dimension entries in the fact table reference directly to the SID table. On the one hand, this saves one table join at query runtime and, on the other hand, it saves the determination of the dimension IDs at data load time.

Line-item dimensions arise in nearly every case where the granularity of the fact table represents an actual working document like an order number, invoice number, sequence number.

In BW 2.x on ORACLE, a different type of index (B-tree instead of bitmap) is applied. In BW 3.x on ORACLE, this is optional in BW 3.x via the High Cardinality Flag. For more information on B-tree and bitmap, see the Index section.

Define large dimensions as line item dimensions (e.g. document number or customer number) if (as a rule of thumb) the dimension table size exceeds 10 % of the fact table(s) size; B-tree is generally preferable for cases where there is high cardinality (high number of distinct values), e.g., document number (only ORACLE provides the alternatives B-tree and bitmap index).

Line-item dimensions may not contain more than one characteristic.

Note that for line-item dimensions, an F4-help (in the frontend) is only possible on master data values, not on dimension entries.

A line-item dimension of an InfoCube containing data cannot be reconverted to a non line-item dimension without bigger effort.

Non-cumulative key-figures are a special feature of InfoCubes. They are used when the key figure does not make sense in an aggregated form for a specific dimension; e.g. stock figures should not be aggregated over time. They are stored for every possible characteristic value combination as reference point (ideally: current value) and all the changes over time. For example calculating the stock value of last year can mean reading all the transaction data between the current stock and last year. The reference point is updated when the InfoCube is compressed. The finer the time granularity, the more records have to be read. Reference points are generated for every characteristic value combination and they stay in the compressed E-table until the data is deleted (deletion without a time restriction!). The more granular the dimensions are defined, the bigger the E table will grow and the more expensive aggregate builds will be. See more recommendations for non-cumulative key figures in the Query Definition chapter.

InfoCubes containing non-cumulative key figures should not be too granular. A high granularity will result in a huge amount of reference points which will impact aggregate build significantly. Reference points can only be deleted by deleting an object key not specifying the time period, i.e. all available records for this key are deleted. If e.g. old material does not need to be reported on, delete these data via selective deletion (without restriction in time!).

As non-cumulative key figures are well defined for every possible point in time (according to the calculation algorithm), it could make sense to restrict the validity to a certain time period (e.g. if a plant is closed, it should not show up any stock figures). These objects can be defined as validity objects. Note that for every entry in the validity table, a separate query is generated at query runtime.

If you use non-cumulative key figures, use as few validity objects as possible. Do not misuse validity objects for termination.

Note: A sound data model is probably the most important issue to consider. We strongly recommend that an experienced consultant validates the final data model.

The data model has tremendous impact on both query AND load performance. E.g. bad dimension model (e.g. customer and material in one dimension instead of separate dimensions) can lead to huge dimension tables and thus slows down query performance, as it is expensive to join a huge dimension table to a huge fact table. Transaction RSRV can be used to check the fact to dimension table ratio.

See the paper on Multidimensional Data Modeling for more details on modeling.

Data models can be improved in terms of query performance or data load performance. In some cases, both goals are mutually exclusive.

3.2 Hardware and Resources

3.2.1 Hardware Impact

The capacity of the hardware resources represents highly significant aspect of the overall performance of the BW system in general. Insufficient resources in any one area can constraint performance capabilities

These include:

- number of CPUs
- speed of CPUs
- memory
- I/O-Controller
- Disk architecture (e.g. RAID)

The hardware sizing should be done according to the recommendations of SAP's hardware partner. The number of CPUs is dependent on the required number of parallel processes (e.g. upload processes, queries). In the end, better hardware will certainly improve your system performance and should be considered, if all other optimizing activities were not satisfactory.

A reasonable sizing is the prerequisite for satisfactory performance. If your BW system is likely to grow, be sure that the hardware is scalable. We recommend to have a five year growth projection that takes into account a long term near line storage and archiving plan, as well as granularity size goals – for example maintaining two years of detail level data, and five years summary data. You also need to account for your PSA size – Do you delete an old load after the next successful load? Or do you manage PSA at all?

See QuickSizer in SAP Service Marketplace for more details.

The client hardware has also significant impact on the query response time. You can improve rendering time of web applications or BEx Analyzer queries significantly by supplying a better frontend client hardware.

See SAPnote 321973 (Recommendations for the BW front end) for BW hardware requirements.

3.2.2 IT Landscape and Configuration

A BW environment can contain a DB server and several application servers. These servers can be configured individually (e.g. number of dialog and batch processes), so that the execution of the different job types (such as queries, loading, DB processes) can be optimized. The general guideline here is to avoid hot spots and bottlenecks.

Be sure that enough processes (of a certain type) are configured.

A BW application server does not need a UP2 process and at most would use 2 UPD processes.

Operation modes can be switched to affect the allocations for dialog and batch processing for different application servers.

For optimizing the hardware resources, it is recommended to define at least two operation modes: one for batch processing (if there is a dedicated batch window) with several batch processes and one for the query processing with several dialog processes.

Note that the data load in BW for extraction from R/3 runs in dialog processes.

An important point here is that sufficient resources should be available in order to distribute the workload across several application servers. Monitor the activity on different app servers to determine a strategy for optimizing the workload distribution (using load balancing). Also, if the database server is “tight” on resources, consider moving the central instance away from the database server, in order to allocate maximum resources to database processes. Note that database processes may utilize a significant number of CPUs, and thus you should carefully monitor CPU utilization in the DB server.

Different application servers have separate buffers and caches. E.g. the OLAP cache (BW 3.x) on one application server does not use the OLAP cache on other servers.

We recommend using the services SAP EarlyWatch provides. See here for more information.

3.3 Administration

3.3.1 Archiving

The BW 3.x archiving feature enables you to archive data from InfoCubes and ODS objects and delete the archived data from the BW database. This reduces the data volume and, thus, improves upload and query performance.

An archiving plan can also affect the data model. For a yearly update, an MultiProvider partitioning per year enables you to quickly drop one InfoCube and create a new one linking it to the same MultiProvider.

Define an archiving project plan quite early. Use this feature to reduce the sizes of the large DataTargets regularly (e.g. monthly).

Note that an archiving project plan may also affect data modeling.

See the BW Documentation for more information. Further information can be found in How To … Archive in BW.

3.3.2 Load Balancing

Load balancing provides the capability to distribute processing across several servers in order to optimally utilize the server resources that are available. An effective load balancing strategy can help you to avoid inefficient situations where one server is overloaded (and thus performance suffers on that server), while other servers go underutilized. The following processes can be balanced:

- Logon load balancing (via group login): This allows you to distribute the workload of multiple query/administration users across several application servers.
- Distribution of web users across application servers can be configured in the BEx service in SICF.
- ODS Object Data Activation is definable for specific server groups (BW 3.x).
- Process Chains can be processed on specified server groups (BW 3.x).
- Extraction in the mySAP source system can be processed on specified server groups: RFC destination in BW to source system must be defined accordingly (transaction SM59).
- Data load in BW can be processed on specified server groups: RFC destination in source system to BW must be defined accordingly (transaction SM59).
- Data staging via XML over HTTP/SOAP can be processed on specified server groups (BW 3.x).

You can define logon server groups in transaction SMLG and RZ12 and assign these groups to the mentioned processes; the data packages are sent to the servers included in the server group.

If you have installed a complex IT environment with several application servers (according to the hardware sizing recommendations), define suitable server groups, to which specific tasks can be assigned. This should help to leverage your hardware.

For even distribution of data load to BW across the BW application servers, we recommend to set the maximum number of logon users for the logon group to a low number (e.g. 5). Once one instance reaches this 5 users, the subsequent logons were dispatched to other instances until all instances have 5 users. Then, this 5 setting gets ignored in SMLG whereas each instance alternates in accepting the next user.

For information on load balancing for web applications, see SAPnote 493475 (BW Web Intelligence and using the message server) and SAPnote 561885 (BW Web Intelligence and SAP BW Web Dispatcher/Reverse).

Having installed a larger scale IT environment (i.e. more than one application server), specific tasks should be distributed among the server groups to avoid I/O bottlenecks on one server.

In some cases, it is useful to restrict the extraction or data load to a specific server (in SBIW in an SAP source system, or SPRO in BW), i.e. not using load balancing. This can be used for special cases where a certain server has fast CPUs and therefore you may want to designate it as an extraction or data load server.

3.3.3 Reorganization of Log Tables

Logs of several processes are collected in the application log tables. These tables tend to grow very big as they are not automatically deleted by the system and can impact the overall system performance.

Table EDI40 can also grow very big depending on the number of IDOC records.

Depending on the growth rate (i.e., number of processes running in the system), either schedule the reorganization process (transaction SLG2) regularly or delete log data as soon as you notice significant DB time spent in table BALDAT (e.g., in SQL trace).

See SAPnote 195157 (Application log: Deletion of logs) for more details.

See also SAPnote 179046 (Rapid Expansion of table EDI40, EDI30C) for information on archiving the IDOC tables.

Delete regularly old RSDDSTAT entries.

3.3.4 Traces and logs

SAP BW provides several possibilities for traces and logs. These traces usually help find errors and monitoring the system.

Examples of traces are: Authorization check log (RSSMTRACE), User trace (RSRTRACE), SQL trace (ST05), ABAP trace (SE30) and Statistics trace (necessary for the technical content).

Be aware that traces and logs generate a system overhead.

Be sure that only those traces are switched on that are really necessary.

As the SAP statistic InfoCubes (technical content) provide very helpful information also on finding aggregates, we recommend switching on the statistics trace in the Administrator Workbench menu.

If several traces and logs run in the background, this can lead to bad overall performance – and sometimes it's difficult to discover all active logs. So be sure to switch off traces and logs as soon as they are not used any more.

4 Extraction in the Source System (Service-API)

4.1 General

4.1.1 Settings of Extractors

The size of the packages depends on the application, the contents and structure of the documents. During data extraction, a dataset is collected in an array (internal table) in memory. The package size setting determines how large this internal table will grow before a data package is sent. Thus, it also defines the number of COMMITs on DB level.

Moreover, the application server (group) where the (batch) extraction processes are scheduled and the maximum number of parallel (dialog) processes can be defined.

Set up the parameters for your DataSources according to the recommendations in SAPnote 417307 (Extractor Packet Size: Collective Note for Applications).

Examples can be found in SAPnote 409641 (Examples of packet size dependency on ROIDOCPRMS).

In general, small data package sizes are good for resource-constrained systems and big sizes are good for large systems. Default setting is 10,000 kByte and 1 Info IDOC for each data packet. These sizes can be fine tuned per InfoPackages and InfoCube. Typical global settings are 20-50000 kByte, 10 or 15 frequency of IDOC info packet and 2 to 4 parallel processes. This is in line with most EarlyWatch recommendations, and will vary a little between installations.

Distribute extraction processes to different servers to avoid bottlenecks on one server.

Large package sizes are not advised if the BW system interfaces to a source system over a WAN; large package sizes can be a special problem if network traffic is a concern. In these cases a small package size is preferable.

4.1.2 Indices on DataSource tables

Indices can be built on DataSource tables to speed up the selection process.

If you define selection criteria in your InfoPackage and the selection of the data is very slow, consider building indices on the DataSource tables in the source system.

Do not create too many indices because every additional index slows down the inserts into the table (in this case the collector job – if available; depending on the particular application area for the data source).

4.1.3 Customer Enhancements

Customer Enhancements (Exits) are available for most of the extractors. Specific coding can be used here to transform the data according to specific requirements.

Be sure that customer enhancements are designed and implemented very carefully according to the recommendations in the Appendix.

If you discover a lot of time spent in customer enhancements (e.g. via ABAP trace or SQL trace), check and improve the coding.

4.2 Application-specific Settings

4.2.1 Logistics Extractors

The logistics extractors provide the extraction basis for all logistics data. It contains the delta queue, the V3 collection process and the administration/scheduling of the necessary jobs. There are the following update methods:

- Direct delta produces one LUW for every document and is only useful for very few records (without collection process).
- Queued delta collects up to 10,000 records and transfers when requested by the collection process.
- Un-serialized V3 updating collects records until the collection process is scheduled; the sequence of the data is not kept.

Depending on your requirements, choose the update method according to SAPnote 505700 (LBWE: New update methods as of PI 2002.1) and the document New update methods.

4.2.2 LIS InfoStructures

Switching between extraction tables SnnnBIW1 and SnnnBIW2 requires deleting the records after the extraction.

The deletion can be improved by dropping the whole table. Please apply the recommendations in SAPnote 190840 (Performance: Deleting delta tables when using LIS InfoSources SnnnBIW1 and SnnnBIW2).

4.2.3 CO Extractors

Take the performance recommendations in the following notes into account:

SAPnote 398041: INFO: CO/IM contents (BW)

SAPnote 190038: Coll.note:InfoSource 0CO_PC_01 & 0CO_PC_02 performance

5 Data Load Performance

5.1 General

5.1.1 Upload Sequence

The sequence of the load processes is usually defined within process chains (BW 3.x) or event chains. This sequence can have a significant impact on the load performance.

The master data load creates all SIDs and populates the master data tables (attributes and/or texts). If the SIDs do not exist when transaction data is loaded, these tables have to be populated during the transaction data load, which slows down the overall process.

Always load master data before transaction data. The transaction data load will be improved, as all master data SIDs are created prior to the transaction data load, thus precluding the system from creating the SIDs at the time of load.

If you want to replace existing data in the DataTarget completely, first delete the data (in PSA and/or DataTarget) and load afterwards. Small (or empty) PSA tables improve PSA read times. Small (or empty) InfoCubes improve deletion, compression time and aggregation time (thus affecting data availability).

This is also described in SAPnote 130253 (Notes on upload of transaction data into the BW).

5.1.2 PSA Partition Size

PSA tables are partitioned (if available in the DBMS) automatically. In transaction RSCUSTV6 the size of each PSA partition can be defined. This size defines the number of records that must be exceeded to create a new PSA partition. One request is contained in one partition, even if its size exceeds the user-defined PSA size; several packages can be stored within one partition.

The PSA is partitioned to enable fast deletion (DDL statement DROP PARTITION). Packages are not deleted physically until all packages in the same partition can be deleted.

Set the PSA partition size according to the expected package sizes. If you expect many small packages, set the partition size to a rather small value, so that these partitions can be deleted quickly.

For DB2/390, see SAPnote 485878 (DB2/390: BW: Partitioning the PSA tables) for more details.

Database Partitioning is not available for all DBMS. Check the section on partitioning for details.

5.1.3 Parallelizing Upload

Parallel processing is automatically initiated during an extraction from an SAP system; the settings for the datapackage size directly influences the number of datapackages that are likely to be sent in parallel. Moreover, a setting in the InfoPackage defines the degree of parallelism of updating to PSA and DataTargets. Thus, data load performance is fully scalable.

You can start several processes manually in parallel: InfoPackages, Process Chains (BW 3.x), InfoPackage Groups (BW 2.x) and Event Chains (BW 2.x).

Use parallelism for uploading if possible (i.e. no resource restrictions):

- Flat files: Split files for multiple InfoPackages, enable parallel PSA → DataTarget load process.
- MySAP source system: Create several InfoPackages for the same or different DataSources and then schedule them in parallel (this is user-controlled parallelism).
- Data load to several DataTargets: Use different InfoPackages (at least one for each DataTarget).

If the data load takes place when there is little other activity in the BW system, then optimal results are likely to be observed if the number of (roughly equally-sized) InfoPackages is equivalent to the number of CPUs in the application servers in the logon group handling the data load.

See document Parallelization when loading data and SAPnote 130253 (Notes on upload of transaction data into the BW) for more information and Key Performance Figures for BW 2.0 for test results (only internally available).

The subsequent load from PSA is parallel by requests; data load from PSA to several DataTargets is sequential.

Note that delta uploads from one DataSource cannot be parallelized – even if you initialized the delta with disjoint selection criteria. One delta request collects all delta records for this specific DataSource – regardless of any selection criteria.

5.1.4 Transformation Rules

Transformation rules are transfer rules and update rules. Start routines enable you to manipulate whole data packages (database array operations) instead of changing record-by-record.

Standard functionalities are one-to-one-mapping, reading master data, using transformation rules (in BW 3.x) and providing own ABAP coding.

In general it is preferable to apply transformations as early as possible in order to reuse the data for several targets. Better use transfer rules (ONE transformation) if you have to transform data for example to several DataTargets (the same transformation for EACH DataTarget). Technical transformations (check for right domain etc.) could even be done outside BW (e.g., in the ETL tool).

Be sure that customer enhancements are designed and implemented very carefully according to the recommendations in the Appendix. If you discover a lot of time spent in transformation rules (e.g., via Data Load Monitor), check and improve the coding.

BW 3.x library transformations interpreted at runtime (not compiled). Usually the impact is very little, but if you use lots of transformations for a huge number of records, then it could be wiser to use ABAP coding instead.

Reading master data is a generic standard functionality. Depending on the concrete scenario, it could be more efficient to buffer master data tables in own ABAP coding, instead.

5.1.5 Export DataSource

The Export DataSource (or Data Mart interface) enables the data population of InfoCubes and ODS Objects out of other InfoCubes.

The read operations of the export DataSource are single threaded (i.e. sequential). Note that during the read operations – dependent on the complexity of the source InfoCube – the initial time before data is retrieved (i.e. parsing, reading, sorting) can be significant.

The posting to a subsequent DataTarget can be parallelized by ROIDOCPRMS settings for the “myself” system. But note that several DataTargets cannot be populated in parallel; there is only parallelism within one DataTarget.

Use InfoPackages with disjoint selection criteria to parallelize the data export.

Complex database selections can be split to several less complex requests. This should help for Export DataSources reading many tables. See SAPnote 514907 (Processing complex queries (DataMart)) for more details.

If the population of InfoCubes out of other InfoCubes via Export DataSource takes up too much time, try to load the InfoCubes from PSA.

BW uses a view on the fact table to read the data in BW 3.x.

If a BW 3.x-Export DataSource runs into problems due to reading from the view, turn it off as described in SAPnote 561961 (Switching the use of the fact table view on/off).

The Export DataSource can also make use of aggregates.

Aggregates of source InfoCubes can be used if the transfer structure definition matches the aggregate definition. You can delete unnecessary fields (and their mapping rules to the communication structure) from the transfer structure definition of the generated InfoSource.

5.2 Flat File Upload

5.2.1 Flat File Upload

Flat files can be uploaded either in CSV format (e.g. day/material: 4/16/2002;4711) or in fixed-length ASCII format (e.g. for 8-char-material field: 2002160400004711). If you choose CSV format, the records are internally converted in fixed-length format, which generates overhead.

You can upload files either from the client or from the application server. Uploading files from the client workstation implies sending the file to the application server via the network – the speed of the server backbone will determine the level of performance impact, Gigabit backplanes make this a negligible impact.

The size (i.e., number of records) of the packages, the frequency of status IDocs can be defined in table RSADMINC (Transaction RSCUSTV6) for the flat file upload.

If you load a large amount of flat file data, it is preferable to use fixed-length ASCII format, to store the files on the application server rather than on the client and to set the parameters according the recommendations in the referenced note.

If possible, split the files to achieve parallel upload. We recommend as many equally-sized files as CPUs are available.

This is also described in SAPnote 130253 (Notes on upload of transaction data into the BW).

5.3 Load of Master Data

5.3.1 Parallel Master Data Load

For a time-consuming extraction of master data, it is advisable to extract the data in parallel using disjoint selection criteria, while queuing the data in BW for serial posting (the eventual posting of the data in BW should be fairly quick).

For more details, see SAPnote 421419 (Parallel load of master data).

5.3.2 Buffering Number Range

The number ranges buffer for the SID's resides on the application server and reduces DB server accesses. If you set e.g. the number range buffer for one InfoObject to 500, the system will keep 500 sequential numbers in the memory and need not access the database.

If you load a significant volume of master data (e.g. initial load), increase the number range buffer for this InfoObject. If possible, reset it to its original state after the load in order to minimize unnecessary memory allocation

For more details, see SAPnote 130253 (Notes on upload of transaction data into the BW).

5.3.3 Change Run

See the chapter below for information on this.

5.4 Upload into an InfoCube

5.4.1 Dropping Indices before Loading

You can create and drop indices in InfoCube maintenance and there are process types available for this in Process Chains (BW 3.x). This functionality affects the (bitmap – in ORACLE) indices on the dimension keys in the F fact table.

The secondary indices of the ODS objects can also be deleted before activating ODS data; programs must be defined here as process types.

If your uncompressed F-table is small, drop indices before loading. In this case, it is faster to rebuild the index at the end of the load process instead of updating the index for each record loaded. Be sure that only one process drops data before several load jobs are scheduled. Rebuild the indices immediately after the load process. It is recommended to integrate these processes within the process chains.

Be careful dropping the indices on large F-fact tables as the rebuild might take a very long time. Regular InfoCube compression (and hence small F fact tables) reduces the index rebuild time.

This is also described in SAPnote 130253 (Notes on upload of transaction data into the BW).

If the indices are not available, querying the data is very slow. Monitor missing indices to ensure that the indices are rebuilt following the data load.

5.4.2 Buffering Number Range

The number range buffer for the dimension ID's resides on the application server and reduces DB server accesses. For example, if you set the number range buffer for one dimension to 500, the system will keep 500 sequential numbers in memory and need not access the database.

If you load a significant volume of transaction data (e.g., initial load), increase the number range buffer. If possible, reset it to its original state after the load in order to minimize unnecessary memory allocation

For more details, see SAPnote 130253 (Notes on upload of transactional data into BW).

5.4.3 Compression

Compression transfers the data from the F fact table to the E fact table while eliminating the request information in the InfoCube. For more information see the Query Performance chapter on Compression.

If you are sure, that every request contains a disjoint set of keys (i.e., the same key only occurs within one record), the compression can be optimized by omitting the UPDATE statement and rather using only INSERTs into the E-fact table.

If you are sure, that the compression for an InfoCube never updates existing records in the E-table (but only inserts new records), set the COMP_DISJ flag for this InfoCube (see referencing SAPnote).

See SAPnote 375132 (Performance Optimization for InfoCube Condensation) for more details.

5.4.4 Roll-Up

The roll-up adds newly loaded transaction data to the existing aggregates. When aggregates are active, new data is not available for reporting until it is rolled up. For information on aggregates, see below. The time spent for the roll-up is determined by the number and the size of the aggregates; if aggregates can be built from other aggregates, they are arranged in an aggregate hierarchy.

Take the following hints into consideration in order to improve the aggregate hierarchy and, thus, the roll-up:

- Build up very few basis aggregates out of the underlying InfoCube fact table
- Try for summarization ratios of 10 or higher for every aggregate hierarchy level
- Find good subsets of data (frequently used)
- Build aggregates only on selected hierarchy levels (not all)
- Build up aggregates that are neither too specific nor too general; they should serve many different query navigations

Monitor aggregates, remove those aggregates that are not used frequently (except basis aggregates)

DB statistics for aggregates are created at regular points. This is necessary when queries are executed during the roll-up.

Under certain circumstances (no queries during roll-up), the roll-up in BW 3.x can be improved by forgoing the DB statistics and building up at the end of the roll-up process. See SAPnote 555030 (Deactivation BW-initiated DB statistics) for more details.

In BW 3.x, aggregate compression can also be deactivated.

If you don't compress the requests in the aggregates regularly, the time used for deleting and rebuilding the indices can be significant. See SAPnote 582529 (Roll-up of aggregates & indices) for a solution of this issue.

Roll-up is not possible when the change run is running.

5.4.5 Change Run

The Change Run adapts all aggregates for newly loaded master data and hierarchies. During the Change Run, all aggregates containing navigational attributes and/or hierarchies are realigned. Newly loaded master data and hierarchies are not available before they are activated via the Change Run. For information on aggregates, see below.

Like the roll-up, the change run runtime is significantly better when the aggregates are related to each other in a good aggregate hierarchy.

Apply the change run ONCE for all newly loaded master data for a given period. Do not schedule it for the load of every master data object.

The change run can be improved analogue to the aggregate roll-up. See above for details. These recommendations can also be found in SAPnote 176606 (Apply Hierarchy/Attribute change … long runtime).

Try to build basis aggregates (rather large aggregates directly filled out of the InfoCube) without master data references (i.e., without navigational attributes/hierarchies). Then there is no need of adjusting these aggregates when master data changes.

In the customizing, you can define a threshold (percentage of changed master data) that decides between delta aggregation and aggregate rebuild. Meaning: unless the threshold is reached, delta records are created for all changes to the aggregate and posted to the aggregate as additional records. If the threshold is exceeded, then the aggregates a dropped and rebuilt from scratch.

Additional note for ORACLE: delta change run uses star transformation, aggregation rebuild uses parallel query (as this is not selective and can be executed in parallel). See SAPnote 619471 (No star transformation with change run) for details.

Test results have led to the recommendation to set the threshold parameter to approximately 30%.

If the InfoCube contains non-cumulative key figures with exception aggregation (MIN, MAX), all aggregates are rebuilt (no delta aggregation).

The change run can be parallelized across InfoCubes.

Use the parallel change run if your hardware resources are sufficient and if your aggregates are distributed among several InfoCubes. See SAPnote 534630 (Parallel Change Run) for more details.

DB statistics for aggregates are created at regular points. This is necessary when queries are executed during the change run.

Under certain circumstances (no queries during change run), the change run in BW 3.x can be improved by forgoing the DB statistics run and rebuilding at the end of the change run. See SAPnote 555030 (Deactivation BW-initiated DB statistics) for more details.

If complex master data with lots of navigational attributes is used, the activation of master data (and, hence, the change run) could take longer.

Complex SQL statements can be split into several less complex statements to improve the activation of master data. See SAPnote 536223 (Activating Master Data with Navigation Attributes) for more information.

The Change Run process can be monitored by means of program RSDDS_CHANGERUN_MONITOR. This can only be used when the change run is actually running.

Use the change run monitor to check which aggregates must be adjusted and which have already been adjusted. See SAPnote 388069 (Monitor for Change Run) for details.

Internal measurements have been done and can be found in the document Key Performance Figures for BW 2.0 (only available in internal SAPnet).

5.4.6 Request Handling

The frequency of data loads determines the number of requests in the InfoCube. Many requests in an InfoCube result in an administration overhead as all requests and their interdependencies must be checked for every data load and when accessing the InfoCube administration. Keep this in mind when designing operational reporting with very high data load frequencies.

Moreover, for hash partitioned InfoCubes, a DB partition is created for every request. Partition Handling for several thousand partitions is usually impacting DB performance.

If you use hash partitioning, compress the InfoCube regularly to avoid too many partitions in the F fact table.

If your data load frequency is rather high and you are likely to have more than 10,000 requests in one InfoCube, please apply SAPnote 620361 (Performance Data Load / administration data target, many requests) to avoid performance issues with the InfoCube administration and data load.

5.5 Upload into an ODS object

5.5.1 ODS Objects Data Activation

ODS Object Data has to be activated before it can be used (e.g. for reporting, for filling the change log, etc.)

Data Activation can be processed in parallel and distributed to server groups in the BW customizing (transaction RSCUSTA2) in BW 3.x. There are parameters for the maximum number of parallel activation dialog processes, minimum number of records per package, maximum wait time in seconds for ODS activation (if there is no response of the scheduled parallel activation processes within this time, the overall status is red) and a server group for the RFC calls when activating ODS data.

Use parallelism wherever possible and wherever your system landscape allows this.

Parallel ODS Object Data Activation (BW 3.x) can improve activation time significantly.

Parallel processes need more hardware resources. Be sure that your hardware capacity can cope with the number of parallel processes.

To improve the deletion, you can manually partition the active data-table on DB level (see partitioning as database-dependent feature). Partitions can be deleted very quickly via DROP PARTITION statement (instead of DELETE FROM).

5.5.2 Indices

You can define secondary indices within the BW 3.x ODS Object maintenance screen. A lot of secondary indices have negative impact on the data activation performance as they have to be maintained during the load. Like for InfoCubes, it is reasonable in these cases to drop the indices before loading and rebuilding them after the load. Currently, you would have to apply native SQL in your own coding and integrate these reports in Process Chains. SAP is currently working on a more convenient solution.

5.5.3 Data Activation Performance and Flag “BEx Reporting”

If you set the flag “BEx Reporting”, SIDs instead of characteristic key values are stored; this improves the reporting flexibility but slows down the upload. In BW 3.x, the SIDs are determined during activation, in BW 2.x during data load. Note that in BW 3.x, the SIDs (for “BEx Reporting” enabled ODS objects) are created per package; hence, multiple packages are handled in parallel by separate dialog processes.

Alternatively, InfoSet queries can be used for ODS objects without activated “BEx Reporting” flag.

Do not set the “BEx Reporting” flag if you do not plan to report on ODS objects in BEx or in Web. If your reporting requirements on ODS objects are very restricted (e.g., display only very few, selective records), use InfoSets on top of ODS objects and disable the “BEx Reporting” flag.

See also SAPnote 384023 (Optimizing Performance of ODS objects (BW 2.x)) and SAPnote 565725 (Optimizing the Performance of ODS objects in BW 3.0B).

5.5.4 Unique Records in ODS Objects

For unique record keys (e.g., sales documents), the option “unique data records” can be set in BW 3.x. The look-up of existing key values is omitted and there are only (mass) inserts into the active table. In addition, the before-image in the change log can be omitted and the data needs not be sorted before activation.

If the BW 3.x-ODS object has unique record keys, the activation process need not perform an “update” statement (the process tries to update an existing record first; if this update fails – i.e., there is no record with the same key – the record is inserted), but it can rather directly “insert” the new record.

See also SAPnote 565725 (Optimizing the Performance of ODS objects in BW 3.0B).

If you turn on “unique data records”, BW cannot guarantee unique records; this must be guaranteed from outside the BW system.

5.5.5 Request Handling

The frequency of data loads determines the number of requests in the ODS Objects. Many requests in an ODS Object result in an administration overhead as all requests and their interdependencies must be checked for every data load and when accessing the ODS Object administration. Keep this in mind when designing operational reporting with very high data load frequencies.

If your data load frequency is rather high and you are likely to have more than 10,000 requests in one ODS Object, please apply SAPnote 620361 (Performance Data Load / administration data target, many requests) to avoid performance issues with the ODS Object administration and data load.

6 Query Performance

6.1 Query Definition

6.1.1 Query Definition

Queries can be defined centrally or individually. The ad-hoc query designer also allows web frontend users to define individual queries. When a query is new, it has to be generated during the first call. This time is contained in the OLAP-Init time.

It is recommended to define queries centrally. This avoids frequent re-generation of queries (thus keeping OLAP-Init time small) and admits targeted improvements by aggregates.

The definition of a query can significantly influence the runtime of it.

Queries can be defined on top of all InfoProviders including InfoCubes, RemoteCubes, MultiProviders, ODS Objects etc.

Be aware that only InfoCubes (and MultiProviders on top of them) are optimized in terms of query performance.

RemoteCubes and Virtual InfoProviders with services access data sources in a remote system, which adds additional network time to the total runtime.

ODS objects and InfoSets should be reported on only very selectively, i.e. only some very specific records should be read, and only little aggregation and little navigation should be required.

In the past, the common requirement of end users was pure reporting, i.e. having a large list of figures representing all necessary information within this one list. Powerful multi-dimensional query capabilities can make this time-consuming searching and the special expert knowledge redundant, but rather pinpointing the necessary information to basically all types of end users and let the user navigate to find even more details.

Incorporate multi-dimensional reporting into all your queries:

Start with a small set of data and allow the users to drill down to get more specific information.

In general, keep the number of cells, that are transferred to the frontend, small.

Do not use ODS objects for multi-dimensional queries. Instead, include ODS objects in drill-down paths to access very few detailed records.

Calculated key figures and currency conversion can be defined in queries. If they have to be calculated 'before aggregation', aggregates cannot be used.

For all calculations that have to be performed before aggregation (like currency conversion), consider if it could be done during data load. See SAPnote 189150 (Calculated non-cumulative key figures before aggregation) for details.

Restricted key figures/Selections/Filters allow the definition of key figures for specific characteristic values.

Better use inclusion of characteristic values instead of exclusion.

Calculation of non-cumulative key figures takes a long time, if the reference point must be calculated (i.e. InfoCube is not compressed) AND, subsequently, the values have to calculated.

Be sure that the InfoCube is compressed if you use non-cumulative key figures.

At query time, use tight restrictions on time characteristics; ideally request only current values.

If possible, split non-cumulative key figures with last or first aggregation (aggregates can be used) from those with average aggregation (aggregates can not be used) in different queries.

Suppress sum lines if not needed.

Do not use partial time characteristics (e.g. FISCPER3) if not needed.

See the chapter on Data Modeling for details.

Queries on MultiProviders usually access all underlying basis InfoProviders.

If a query on a MultiProvider reads only selective key figures and some InfoProviders do not contain any of these key figures, use 0INFOPROVIDER manually to include only the required basis InfoProviders.

The Cell Editor enables to overwrite certain cells with results of specific queries. For every cell which is defined, a new query is started.

Be cautious with too many cell calculations. Keep in mind that every cell, which is defined in the cell editor, causes a comparable load like a normal query.

6.1.2 Virtual Key Figures / Characteristics

You can include key figures and characteristics in queries, which are not available in the InfoProvider. They are called virtual key figures/characteristics and their source is coded in a customer exit.

Take the recommendations in the Appendix into account when you implement the customer exit.

6.1.3 Query Read Mode

The read mode defines if a query reads only the necessary data for the first navigation step (and reads the database for every additional navigation step) or if it reads everything in one go (and does not access the database for any additional navigation steps). It can also be set up if hierarchies are read in one go.

The read mode can be defined generally, for InfoCubes and for single queries.

For most of the queries it is reasonable to load only that number of records from the DB that is really required. So we generally recommend to set queries to “read when navigating and expanding hierarchies” (default setting).

6.1.4 Reporting Formatting

For formatting Excel cells, formatting information must be transferred. If the cell format often changes (e.g., too many result lines), it might be reasonable in terms of performance to switch the formatting off.

If you discover significant high frontend time, check whether the formatting is the reason. If so, either switch it off or reduce the result lines.

As formatting information is not transferred, the time consumed in the frontend can be reduced.

6.2 Indices and Compression

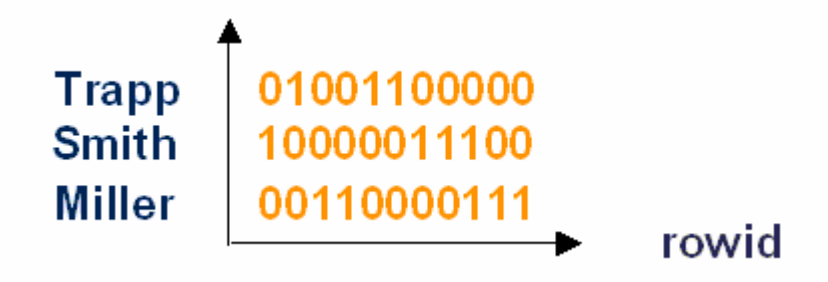
6.2.1 Indices

Indices speed up data retrieval for a given selection tremendously. If no indices are available, full table scans have to be performed. Generally speaking, indices speed up accesses to individual records and groups of records significantly.

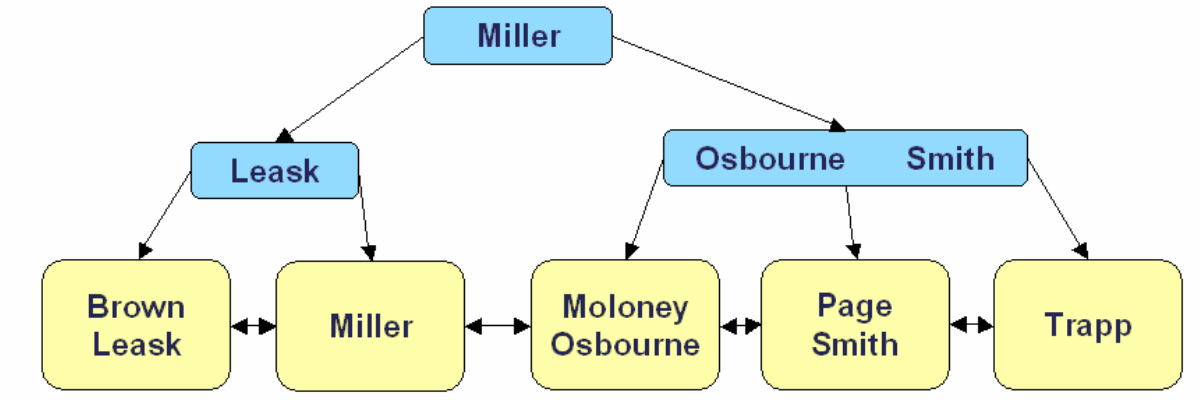
For querying data, the necessary indices must be available and up-to-date. Make sure that the indices for InfoCubes are available if you want to report on the InfoCube. You can check this in the InfoCube maintenance and also in transactions DB02 and RSRV.

Indices have to be updated during the data load; thus, they decrease data load performance. So, indices should be designed carefully.

In ORACLE, fact tables can be indexed either by bitmap indices or by B-tree indices. A bitmap index stores a bitmap stream for every characteristic value. The '1's in the stream define the row IDs which contain the value. Bitmap indices are suitable for characteristics with few values. Binary operations (AND or OR) are very fast. Example:



B-tree indices are stored in a (balanced) tree structured. If the systems searches one entry, it starts at the root and follows a path down to the leaf where the row ID is linked. B-tree indices suit for characteristics with lots of values. Example:



InfoCubes (fact, dimension tables) are fully indexed and usually do not require additional indices. The indexing as follows:

- Fact Table (consists of dimension IDs and key figures)

  - Non-unique B-Tree Index on all dimension IDs (= primary key)
    - This so-called P-Index is used for compression
    - In ORACLE and Informix, this index is created only for the E-fact table
  - Non-unique Index on every dimension ID
    - ORACLE: Bitmap index (except for “high cardinality” dimensions and for F-fact tables of transactional InfoCubes)
    - DB2/AS400: EVI
    - Other DBMSs: B-Tree
- Dimension Tables (consists of Dimension ID and master data SIDs)
  - Non-unique B-Tree index on Dimension ID (= primary key)
  - Non-unique B-Tree on all master data SIDs
  - Non-Unique B-Tree on every master data SID
    - ORACLE / DB2/390: not on first SID, as this is included in the index on all SIDs

Note that there is an additional field (SID of chosen time characteristic) for range-partitioned fact tables. ORACLE creates an additional bitmap index on this field.

B-tree indices for InfoCube dimensions (“high cardinality”) should be set only in special cases. The dimension should be very large (in comparison to the fact table) and the maintenance time for bitmap indices (deletion and re-build) should have significant impact. You should be aware of the fact that the B-tree index cannot be used for the star join and hence, reduces query performance.

See SAPnote 217397 (Indexing Scheme of BW fact tables under ORACLE) and the document BW Indexing (under ORACLE) for information on indexing under ORACLE.

In some cases, ORACLE indices can degenerate. Degeneration is similar to fragmentation, and reduces the performance efficiency of the indexes. This happens when records are frequently added and deleted.

Check if indices under ORACLE are degenerated. See SAPnote 323090 (Performance problems due to degenerated indices) for more information.

ODS objects need to be indexed manually depending on the reporting requirements. This can be done in the ODS object maintenance.

If you report on ODS objects and the performance is bad due to selections on non-indexed characteristics, build up secondary indices.

Master Data is indexed as follows:

- S – (SID-) Table (consists of master data key, the SID and some flags)
    - Unique B-Tree index on master data key (= primary key)
    - Unique B-Tree index on SID
- X- / Y- Tables (consists of SID of characteristic, object version and all SIDs of the navigational attributes)
    - Unique B-Tree index on <SID, object version> (= primary key)

Additional Master Data indexing on the X- and Y- tables can be useful. These indices have to be created in transaction SE11.

If accesses on master data (navigational attributes) are slow, check the execution plan (ST05). If the plan is not optimal for the access to navigational attributes, create indices on the X- (time-independent) or Y-table (time-dependent) of the respective characteristic. Check if the indices improve performance; if not, delete them again! See SAPnote 402469 (Additional indices on master data tables) for more details.

Note again, that indices on master data can slow down the data load process. Delete seldom-used indices.

6.2.2 Compression

Compression transfers the data from the F fact table to the E fact table while eliminating the request information in the InfoCube. It aggregates records with equal keys from different requests.

For DBMS supporting range partitioning, the E fact table is optimized for reporting with respect to database partitioning and the F fact table is optimized for data load and deletion.

As of BW 3.0, it is also possible to manually compress (or not compress) aggregates; this enables you to have requests available for reporting and still have the possibility of deleting these requests – as long as they have not been compressed.

We recommend compressing those requests in the InfoCube that are not likely to be deleted as soon as possible. Also compress aggregates as soon as possible. The InfoCube content is likely to be reduced in size, so DB-time for queries should improve.

Compression only aids in query performance. It adds another step to the overall data load process and takes up resources.

Customized partition settings (0CALMONTH or 0FISCPER) are only valid in the compressed E-table.

Compression prevents the F-table (which is partitioned by request ID) from containing too many partitions. The DB-internal administration of some thousands (as a rule of thumb) partitions decreases the overall performance for an InfoCube.

The reference point of non-cumulative key figures is updated when the InfoCube is compressed. This reduces the OLAP time of queries on these InfoCubes.

Compressing improves query performance of InfoCubes containing non-cumulative key figures significantly.

Individual requests cannot be accessed and deleted any more after compressing.

If you delete requests that have been rolled up and compressed in the aggregates, these aggregates have to be dropped and rebuilt completely. This can take a long time depending on the size of the aggregate and associated fact table.

6.3 Aggregates

6.3.1 Aggregates

Aggregates are materialized, pre-aggregated views on InfoCube fact table data. They are independent structures where summary data is stored within separate transparent InfoCubes. The purpose of aggregates is purely to accelerate the response time of queries by reducing the amount of data that must be read in the database for a given query navigation step. In the best case, the records presented in the report will exactly match the records that were read from the database.

Aggregates can only be defined on basic InfoCubes for dimension characteristics, navigational attributes (time-dependent and time-independent) and on hierarchy levels (for time-dependent and time-independent hierarchy structures). Aggregates may not be created on ODS objects, MultiProviders or Remote Cubes.

Queries may be automatically split up into several subqueries, e.g for individual restricted key figures (restricted key figures sales 2001 and sales 2002). Each subquery can use one aggregate; hence, one query can involve several aggregates.

If an aggregate has less than 15 components, BW 3.x puts each component automatically into a separate dimension that will be marked as “line item” (except package and unit dimension); these aggregates are called flat aggregates. Hence, dimension tables are omitted and SID tables referenced directly. Flat aggregates can be rolled up on the DB server (i.e., without loading data into the application server). This accelerates the roll up (hence the upload) process.

Define aggregates for queries that have high database read times and return far fewer records than read. For more details, see the analysis chapter on SAP statistics and the document Performance Tuning for Queries with Aggregates.

Check SAPnote 630502 () for information on usage of aggregates for queries using OLAP feature “elimination of internal business volume”.

The more aggregates exist, the more time-consuming is the roll-up process and thus the data loading process; the change run is also affected.

ORACLE supports parallel selection from source table during the aggregate build.

Use the ORACLE parallel selection feature by setting the RSADMIN parameter according to your system resources and your expected load. See SAPnote 544521 (Controlling the parallel degree for the aggregate build) for more details.

Aggregate handling for time-dependent attributes and time-dependent hierarchy structures is possible in BW 3.x via defining a key date (fixed date or BEx variable which is filled via a SAP- or User-Exit). Queries with the same key date can use these aggregates. A process type in the Process Chains provides the adjustment of the key date of time-dependent aggregates.

Note that the adjustment of aggregates on time-dependent master data can be very expensive.

If you use elimination of internal business volume in SAP BW 3.x, aggregates must contain the two depending characteristics (sender and receiver) in order to be used in these queries.

If the aggregates are too large and performance is not satisfying, consider using the OLAP Cache instead.

There are some restrictions for aggregates on InfoCubes containing key figures with exception aggregation.

The characteristic for the exception aggregation must be contained in all aggregates. See SAPnote 125681 (Aggregates and exception aggregation) for further explanation.

Note, that InfoCubes containing non-cumulative key figures with exception aggregation (MIN, MAX) cannot use the delta change run: all aggregates are rebuilt. See also chapter on change run.

SAP BW can propose aggregates on basis of queries that have been executed.

Do not activate all automatically proposed aggregates without taking into account the costs of roll-up/change run. See SAPnote 166433 (Options for finding aggregates) for tips on finding aggregates. See also in the chapter on roll-up and performance analysis tools.

6.3.2 Aggregate Block Size

Data of large InfoCubes is read in several blocks (rather than in one large block that needs more memory/temporary tablespace than available). The block size is defined system-wide in customizing; blocks are distinguished by characteristic values.

Note that this has nothing to do with DB block size; instead it is like a periodic commit on the aggregates process. This allows completion of the aggregate build process under cases of memory constraint.

If you build aggregates from large InfoCubes, use the block size in order to limit the resource consumption and, thus, to prevent resource bottlenecks. But keep in mind, that this may be detrimental to the overall build time.

See SAPnote 484536 (Filling aggregates of large InfoCubes) for more details

6.3.3 MOLAP Aggregates

Multi-dimensional OLAP (MOLAP) is proprietary (Microsoft), multi-dimensional (array-like) storage technology.

The data is stored in two places: the relational tables for the InfoCube and in the multi-dimensional structures on the analysis server. Queries take the data from the Analysis Services instead of reading them from the database. During the roll-up process, the data is transferred from the relational structure to the MOLAP structure. The MOLAP structure is indexed, but this indexing is completely different from relational indexing; these indices cannot be influenced. The data in the MOLAP structures is partitioned by Request ID.

The connection between the BW application server and the MOLAP bridge on the DB server is realized by RFC's that transport MDX (multi-dimensional Expression language).

If your BW is installed on MS SQL Server, you can use MS Analysis Services to build MOLAP aggregates.

MOLAP aggregates are useful in the following scenarios:

- Straightforward drill-down paths; Scenarios are not too complex (hierarchies, dimensions)

- (Small) Deltas are rolled up in the InfoCubes
- Realignments are infrequent (change run)
- The InfoCube does not contain time-dependent navigational attributes
- The InfoCube does not contain non-cumulative key figures
- Number of characteristics + Number of navigational attributes + number of BW hierarchies (of characters and navigational attributes.) <= 127

See How-to … use MOLAP-aggregates for more details.

MOLAP aggregates impact roll-up and query performance.

6.4 OLAP Engine

6.4.1 ODS Objects

Reporting on ODS objects is usually used in drill-down paths to retrieve a few detailed records. Accessing selective records usually requires (secondary) indices to avoid full table scans.

If you filter special values within ODS objects (for reporting or for upload into other DataTargets), be sure that these characteristics are indexed (also consider the sequence of the index characteristics). Secondary indices accelerate the selective reading from an ODS object. This improves the update from the ODS object as well as the reporting on the ODS object.

See also SAPnote 384023 (Optimizing Performance of ODS objects (BW 2.x)) and SAPnote 565725 (Optimizing the Performance of ODS objects in BW 3.0B).

Indexing speeds up querying but slows down data activation.

In some cases, the order of the primary index (i.e., the sequence of the key fields) can improve the table access (and help you to save a secondary index).

6.4.2 MultiProviders

MultiProviders are combinations of all types of InfoProviders (InfoCubes, ODS objects, InfoObjects and InfoSets). They do not store the data, but access the data from the underlying InfoProviders. MultiProviders are used transparently for reporting. MultiProviders are used for logical partitioning (see modeling chapter for more details).

Queries on MultiProviders are split into parallel (sub-)queries on the basis InfoProviders by default and united at a defined synchronization point. You can manually switch all queries for a MultiProvider to serial processing. In BW 2.x, parallel queries on MultiProviders can only use one aggregate for the whole MultiProvider query.

If in a very rare case a (parallel) query on a MultiProvider takes up too much memory resources on the application server, switch to serial query processing. In BW 3.x, the BW system determines whether a query should be executed in parallel or sequentially. See SAPnote 607164 (MultiProvider: sequential processing faster than parallel processing) for more details. SAPnote 622841 (OLAP: Switch off Parallel Execution on query level) explains how to switch to sequential processing for specific queries.

If you use BW 2.x and you notice that only one aggregate is used by the parallel MultiProvider query, serialize the queries (in test mode via transaction RSRT) and compare the runtimes.

See SAPnote 449477 (Changing MultiCube from “parallel” to “sequential”), SAPnote 327876 (MultiCube query and using several aggregates - only BW 2.x) and SAPnote 629541 (MultiProvider: Parallel Processing) for more details.

If a MultiProvider contains at least one basis InfoProvider with non-cumulative key figures, all queries are processed sequentially.

If a intermediate result exceeds 30,000 lines, the parallel processing is aborted and the query is restarted in the serial mode. See SAPnote 630500 (Maximum size of the interim result with MultiProvide) to learn how to change this setting to a specific kb size; this is only necessary in very seldom cases.

Note that every additional basis InfoProvider generates additional I/O on the database. In BW 3.x, the degree of parallelism is restricted to the number of free dialog processes minus 2.

These are the advantages of MultiProviders:

- Local queries (on each InfoProvider) vs. global queries (on MultiProvider, parallel execution).
- Independent (and parallel) data load into individual InfoProviders.
- Small total data volumes: less redundancy, less sparsely filled and less complex.

So-called homogeneous InfoProviders are structurally exact copies. They just differ by different constants for specific characteristic values. E.g. you have two homogeneous InfoCubes for the years 2001 and 2002, respectively. If you combine them via a MultiProvider, a query for year 2002 do not access the 2001’s InfoCube.

If you use MultiProviders as a logical partitioning on homogeneous InfoProviders, be sure to define constants in the individual InfoProviders to avoid unnecessary accesses.

As a general rule of thumb, we recommend up to 10 Sub-InfoProviders, that are assigned to a MultiProvider and that are accessed simultaneously for one query; if you use more than this, the overhead in combining results might get too big. (Of course, depending on system resources and configuration, more than 10 Sub-InfoProviders can still result in good performance).

You cannot define the constants for homogeneous InfoProviders for compounded characteristics or for ranges (e.g. materials 0000 – 4711). As a workaround for constant ranges, you can define an additional characteristic defining the group (e.g. material_group A comprising materials 0000 to 4711).

6.4.3 OLAP Cache

The OLAP Cache can help with most query performance issues. For frequently used queries, the first access fills the OLAP Cache and all subsequent calls will hit the OLAP Cache and do not have to read the database tables. In addition to this pure caching functionality, the Cache can also be used to optimize specific queries and drill-down paths by 'warming up' the Cache; with this you fill the Cache in batch to improve *all* accesses to this query data substantially.

In general, the OLAP Cache can buffer results from queries and can provide them again for different users and similar queries. It can re-use the cached data for the same query call with the same selection parameters or real subsets of them. The subset principle only works if the respective characteristics is part of the drill-down. This means that the following query with selection parameter Calendar Month = 'unrestricted' fills the OLAP Cache and a subsequent call of the same query with selection parameter Calendar Month = '05.2002' can re-use the cache results. If the calendar month was not in the drill-down, the cache would not be used.

Cal. Year/Month	Plant	Key Figures

Plant		Cal. Year/Month	Orders
2000	Heathrow / Hayes	01.2002	68.000
		02.2002	40.000
		03.2002	118.000
		04.2002	28.000
		05.2002	108.000
		06.2002	58.000
		07.2002	58.000
		08.2002	40.000
		09.2002	96.000
		10.2002	22.000
		11.2002	70.000
		12.2002	22.000
		01.2003	78.000
		02.2003	51.000
		03.2003	67.000
		04.2003	23.000
		05.2003	64.000
		06.2003	72.000
		07.2003	68.000
		Result	**1,151.000**
2010	DC London	01.2002	52.000
		02.2002	48.000

The OLAP Cache stores the query results with their navigation statuses in the memory of the application server; alternatively, the data can also be stored in database tables and files. When the buffer (Export/Import Shared Memory) memory overruns, it stores the displaced data – depending on the persistence mode – on the database server. There are five Cache modes: Cache inactive, Memory Cache without swapping, Memory Cache with swapping, Cluster/Flatfile Cache per application server, Cluster/Flatfile Cache cross-application server. With the last option, you can define dedicated queries to be stored on the database server (table or file) – i.e., independent from an application server. Note that the storage format of Cache entries is highly compressed.

You can define the size of the OLAP Cache in memory (only for the two Memory Cache modes), the persistence mode (i.e. displaced page entries are sourced out to a file/DB table) and you can turn the Cache on/off globally, for InfoProviders and for queries; the InfoProvider setting defines the default value for new queries; after switching off, all existing queries still use the Cache. Note: The actual memory requirements are usually lower than the total size of (uncompressed) runtime objects due to compression.

The OLAP Cache for queries on a specific InfoCube is invalidated when new data is uploaded into the respective InfoProvider (invalidations also for master data/hierarchy change run, currency conversion and more). Hence, the data displayed is always up-to-date and consistent with the original data in the InfoProvider. The OLAP Cache for a specific query gets also invalidated when the query is re-activated.

The OLAP Cache is also used for queries on transactional InfoCubes and for pre-calculated web templates.

OLAP Cache should be used whenever possible. However, for queries that are executed only once (or very rarely) or for very small queries (e.g., DB time < 0.10 sec), the cache overhead could lead to slightly worse performance.

Check if the Shared Memory size (rdsb/esm/buffersize_kb) is bigger than the OLAP Cache size. By default, the Shared Memory size is very small. We recommend an initial size of 100MB.

For more information on the OLAP Cache, see BW Documentation and SAPnote 456068 (OLAP: Deactivating query cache/deleting cache data).

If all frontend users are authorized to create queries, note that the OLAP Cache for a particular query gets invalidated each time the query is changed and re-activated and it might be useless eventually.

You can use the reporting agent or the OLAP trace to warm up the OLAP Cache.

If you want to warm up the OLAP Cache, you can use the reporting agent (pre-calculation) or the OLAP trace (RSRTRACE) as a work-around. The following steps have to be executed:

- RSRT: Activate the Cache for the query (if not default) – in case you have several application servers, define Cache settings for query as independent from application server
- If you use variables, be sure that they are marked as changeable during the navigation (flag in variable editor)

- Define Web Template (e.g. on basis of a view); be sure that variable screen is enforced
- Pre-Calculate HTML (for Web Browser) on basis of this web template

Queries containing virtual characteristics/key figures do not use the OLAP Cache by default. As the OLAP Cache cannot control other database tables, which might be read within the customer exits, and hence cannot invalidate, if there are any changes in those tables, this default setting guarantees data consistency. However, you can enforce explicitly the usage of the OLAP Cache, if you are sure that there can't be a difference between the cached data and the original data using the virtual characteristics/key figures.

If you use virtual characteristics/key figures and want to use the OLAP cache, first make sure that the data is ALWAYS consistent, i.e. that the customer exit is not using frequently updated tables. Keep in mind that the Cache is not invalidated if the data in these tables changes! You can then enforce using the Cache as described in SAPnote 623768 (Extended Cache Functionality).

6.4.4 Hierarchies

Queries can use one or more hierarchies. Hierarchies help to

Queries with several or large hierarchies might be optimized by reducing hierarchy node recalculation. See SAPnote 584216 (RSADMIN parameter for hierarchy buffer size) for more information.

6.5 Authorizations

6.5.1 Reporting Authorizations

The reporting authorizations can restrict critical data to certain users/user groups. On a high level, InfoCubes and queries can be restricted; for more detailed authorizations, also characteristics, characteristic values, hierarchy levels and key figures can be restricted.

From the performance point of view, data that has to be read is restricted by authorizations or unauthorized requests are directly aborted.

Complex authorizations especially on hierarchy levels can slow down the query processing, as large amounts of information must be read and checked.

In terms of performance, it is better to use higher level authorizations. In some cases, in some cases, it could be helpful to define separate queries or InfoCubes for different authorization groups.

We recommend striking a proper balance between performance and security. Especially in complex HR scenario or multidivisional international corporation where only some people can see global views and others can't, authorizations are inevitable - but keep in mind that the level of authorization detail can impact performance.

See BW Documentation for more information. Some more information, especially on HR authorizations can be found in the document BW/HR Authorization.

7 Web Application Performance

7.1 Reporting Agent

7.1.1 Pre-calculated Web Templates (Reporting Agent)

The Reporting Agent allows – among other features – the pre-calculation of web templates. Pre-calculation is a set of techniques where you can distribute the workload of running the report to off-peak hours, and have the report result set ready for very fast access to the data. This reduces server load significantly and provides faster data access; data that goes to many web applications is re-used.

You can choose among following output formats:

- Data: Only data is pre-calculated.
- HTML for Web Browser / HTML for Pocket IE.
- Excel.

You can define the following access modes:

- NEW (default): Uses current data.
- STORED: Uses pre-calculated data (if no pre-calculated data available, this results in an error).
- HYBRID: Uses pre-calculated data if available; if not, it uses current data.
- STATIC: Uses pre-calculated HTML pages (if no pre-calculated pages are available, this results in an error).
- STATIC_HYBRID: Uses pre-calculated HTML pages if available; if not, it uses HYBRID access mode.

The only navigation that is possible within pre-calculated web templates is filtering through drop-down list boxes. In this case pre-calculation needs to take place via control queries.

If some of the end users are pure information consumers, i.e., require only static information, provide pre-calculated web templates to them. This skips the complete query processing time. Rendering time will still be needed unless you have downloaded the templates.

See BW Documentation for more information on pre-calculated web templates.

Navigation like drill-down and slice and dice within pre-calculated web templates is not possible by default. If you want to navigate you have to activate the page by loading the data from up-to-date data. This can be done via linking the URL of the “active” query to a self-defined button.

To avoid the additional roundtrip because of the URL redirect, use the URL

```
http://<appserver>:<port>/sap.bw.doc.tmpl/FLDTMPL/LATEST/LATEST_<template_id>/
GET?<filters_and_variables>
```

instead of

```
http://<appserver>:<port>/sap.bw.BEx?cmd=ldoc&template_id=<template_id>
&<filters_and_variables>
```

as of 3.0B SP9.

Use the direct URL. See SAPnote 594372 (Accelerating the display of pre-calculated HTML pages) for details.

Pre-Calculated Web Templates can also be downloaded and used offline. See SAPnote 510931 (Downloading pre-calculated HTML pages) for more details.

7.2 Web Application Definition

7.2.1 Web Items

Web items define the display and navigation tools that are provided within the web application. They obtain data from the DataProviders and make them available as HTML.

The number and type of web items can influence the performance of web applications. For web items with filtering functionality (dropdown boxes, hierarchical dropdown menus, radio buttons), there are three read modes that affect performance: dimension tables (all data that are in the InfoCube), master data tables and booked values (including authority checks).

For web items with filtering functionality, read mode “dimension tables” is usually the best option with respect to performance.

See BW Documentation for more information on web items.

7.2.2 Stateless / Stateful Connection

The web templates properties define the connection type of the web applications to the BW server. The connection can be established for the entire period of navigation (stateful) or it can be built up every time a new navigation step takes place (stateless). In the latter case, the status information of the current navigation step must be transferred.

Stateful connections are faster with respect to navigation performance; but they must allocate working memory for every connection for the navigation period. This can be significant for a large community of web users. In this case, stateless queries can reduce the hardware requirements with the trade-off of slower navigation performance.

See BW Documentation for more details.

7.2.3 HTTP / HTTPS

Transferred data can be encrypted to provide higher security from unintended data access. For safe data transfer, HTTPS protocol is used rather than HTTP.

Measurements have proven that there is no measurable difference between the two protocols in terms of performance.

7.3 Caching / Compression

7.3.1 Portal iView Cache

When running in SAP Enterprise Portal, the Portal iView Cache can be used to cache certain iViews on the Portal server. The iView Cache is another layer – in addition to aggregates, OLAP Cache and Pre-Calculation – that brings cached data nearer to the frontend and accelerates response times significantly.

The iView Cache can be used as of SAP Enterprise Portal 5 SP5 Patch 1 for Isolation Level 3 (individual load using iVIewServer).

The use case for the Portal Cache is typically the information consumer who wants to browse over several pages very quickly and expects them to be pre-retrieved in the cache. The cache typically does not show up up-to-date data if data is loaded during the query window. For more information, check the documentation and SAPnote 599270 (Portal Content performance – Composite Note).

Portal iView cache invalidation is definable for a time period (e.g. in x hours). If new data is loaded into BW, the portal cache is not invalidated. Be sure that the Portal pages stored in the iView cache contain exactly the data you are expecting.

Be careful using shared cache and personalization. Data in cache can be accessed regardless of authorizations and personalization.

7.3.2 Compressing Web Applications and using Browser Cache

The transferred data (including MIME types like images, Cascaded Style Sheets and JavaScripts) can be compressed to reduce the network load, and the Browser Cache (on the frontend PC) can be used to cache image/gif files.

HTTP Compression should be automatically switched on as of BW 3.0B SP9 – see SAPnote 550669 (Compressed transfer of BW Web Applications) for information on the required kernel patch. Check if http 1.1 is activated in the browser settings.

The number of transferred bytes is reduced and, particularly in WANs, the overall query performance can be significantly improved. Test results have shown that queries using compression and browser caching can improve the overall web query performance by 50%. The number of protocol roundtrips can be reduced to that number that is necessary to transfer the data itself (in our example, we saved 11 roundtrips and needed only two with browser caching).

Profile parameter icm/HTTP/server_cache_0/expiration defines how long the information resides in the browser cache.

The Browser Cache is automatically active. The Cache will be noticeable after the first query execution; then non-dynamic data like images are in the cache and then another call of a query deploys the cache (on the same frontend PC).

See SAPnote 561792 (Client-sided caching of image/gif files) for details.

7.4 Network

7.4.1 Frontend Implications on Network Load

The communication between application server and frontend can be expensive, especially in WAN environments. Excel-based BEx Analyzer and web frontend cause different network traffic in terms of transferred data and protocol roundtrips.

Particularly in a WAN, use the web front end instead of the Excel-based BEx Analyzer as the number of round trips is significantly reduced. Moreover, several compression and caching features are only available for the web front end (see above).

If you have to use Excel-based BEx in a WAN, we recommend to use a Windows Terminal Server in the LAN of the BW to reduce the protocol overhead to the clients in the WAN.

The results of internal tests can be seen in the document BEx frontend performance.

8 DB-Specific Performance Features

8.1 General

8.1.1 Table Partitioning

Table partitions are physically separated tables, but logically they are linked to one table name. PSA tables and non-compressed F-fact table are partitioned by the system (by request ID). The (compressed) E-fact table can be partitioned by the user by certain time characteristics. For range-partitioned InfoCubes, the SID of the chosen time characteristic is added to both fact tables.

When using range partitioning, query response time is generally improved by partition pruning on the E fact table: all irrelevant partitions are discarded and the data volume to be read is reduced by the time restriction of the query. Note that all F fact table partitions have to be read.

IBM DB2 UDB EEE uses hash partitioning in order to achieve high degrees of parallel access. A hash function uniformly distributes data across partitions.

In ORACLE, report SAP_DROP_EMPTY_FPARTITIONS can help you to remove unused or empty partitions of InfoCube or aggregate fact tables. Unused or empty partitions can emerge in case of selective deletion or aborted compression and may effect query performance as all F fact table partitions are accessed for queries on the InfoCube.

We strongly recommend using partitioning if your DBMS supports it. Queries can access partitions in parallel, load into a partitioned PSA is faster and maintenance is improved (fast deletion of partitions).

Be sure to compress regularly to use the partitioning of the E-fact table.

See SAPnote 385163 (Partitioning on ORACLE since BW 2.0) for partitioning information on ORACLE.

Use report SAP_DROP_EMPTY_FPARTITIONS in ORACLE to delete unused F fact table partitions. See SAPnotes 430486 (Overview/repair of F fact table of an BW InfoCube) and 590370 (Too many uncompressed requests in F fact table) for more information.

(Range) Partitioning is available for ORACLE, DB2/OS390 and Informix. DB2/UDB supports hash partitioning.

Avoid too many (usually several 1000) partitions in one table, i.e. avoid too many requests in your F fact table and too granular time characteristics for the E table partition criterion.

8.1.2 DB Statistics

The DB optimizer decides its access paths on the information provided by DB statistics (e.g. number of records, number of different keys). Only up-to-date statistics guarantee optimized access costs estimates and access paths. This includes the choice between full table scans and index usage, between nested loops and has joins or between index range scans and unique scans.

Be sure that the DB statistics are up-to-date (especially when the DB tables are queried on); use the process type within Process Chains to build up statistics on the InfoCube tables. Most database management systems do not create statistics for new objects automatically; be sure that for every new object (InfoProvider), the statistics are created before it is used for reporting.

Outdated statistics can usually be seen in InfoCube maintenance or via RSRV-Check (Database tab strip). You will notice it also in bad access paths in the SQL trace ST05.

See SAPnote 328106 (DB2 UDB EEE statistics for BW tables) for information on DB statistics for DB2/UDB.

In addition to this pure statistics information, histograms are generated in ORACLE 9i and IBM DB2 UDB EEE; they provide information on the distribution of non-uniform key values within the table. Histograms can be used in ORACLE if the SQL statements are handed over by literals instead via variables (BW 3.x).

The ANALYZE TABLE statement calculates statistics and histograms; but this is only possible sequentially in one process. DBMS_STATS can be used in lieu of ANALYZE TABLE to build up the statistics in parallel. But: DBMS_STATS does not create histograms.

If you discover performance problems during DB statistics build, use the parallel DBMS_STATS method instead of ANALYZE TABLE and monitor the query performance. If this should be affected significantly, switch bach to ANALYZE TABLE.

See SAPnote 351163 (Creating ORACLE DB statistics using DBMS_STATS) and SAPnote 129252 (ORACLE DB Statistics for BW tables) for details.

Histograms (under ORACLE) are also generated when you use BRCONNECT as of version 6.10. See SAPnote 428212 (Update of statistics of InfoCubes with BRCONNECT) for more information.

In MS SQL Server, statistics are generated automatically. For very small tables (below 500 records), the statistics are not generated and have to be created manually (in the InfoCube Maintenance) in order to guarantee optimal execution plans.

Database statistics in MS SQL server can be generated by un-hiding the statistics buttons in the InfoCube maintenance. The statistics generation can also be integrated in Process Chains.

See SAPnote 542468 (Activate manual statistic generation on InfoCubes in BW) for more details.

8.1.3 Disk Layout

The purpose of disk layout is to avoid I/O hot spots by distributing the data accesses across several physical disks. The goal is to optimize the overall throughput to the disks. Usually storage (sub-)systems are used to handle bigger transparent storage units (and not one physical disks), e.g. RAID systems and RAID arrays.

The basic rule is: stripe over everything, including RAID-subsystems. If you are not using transparent RAID arrays, be sure that your DB datafiles are striped over all available storage subsystems.

We recommend RAID 5, i.e. sector striping with parity distributed to the disks.

8.1.4 Raw Device / File System

The database is stored in files; these files can be managed by the OS (file system) or by the DBMS (raw device). As a rule of thumb, a raw device is faster by bypassing the OS filesystem and filesystem cache, but requires more administration effort/knowledge.

Use raw devices if your DBMS/OS supports them and you possess the necessary administration knowledge.on ORACLE since BW 2.0) for partitioning information on ORACLE.

See SAPnote 443544 (IO Problems on AIX File Systems / ORACLE) for moving database file to raw devices in AIX/ORACLE.

8.2 ORACLE-Specific Performance Features

8.2.1 Locally-managed tablespaces

Locally-Managed Tablespaces manage their own extents by maintaining bitmaps in each datafile. The bitmaps correspond to (groups of) blocks.

As of BW 3.0, new installs are configured with locally-managed tablespaces by default - except for the temporary tablespace.

Be sure that all (bigger) tablespaces are locally managed. Extent and partition maintenance is drastically improved, as DB dictionary accesses are minimized. Administration maintenance is also reduced.

See SAPnote 359835 (Design of temporary tablespace in the BW system), SAPnote 387946 (Use of locally-managed tablespaces for BW 2.0/2.1) and SAPnote 409376 (Use of locally-managed tablespaces for SAP R/3) for more details

8.2.2 Parallel Query Option (PQO)

ORACLE can read database table contents in parallel if this setting is active. BW uses this feature especially for staging processes and aggregate build.

The Parallel Query Option is used by default. Be sure, that the init<SID>.ora-entries for PARALLEL_MAX_SERVERS are set appropriate to the recommendations in the ORACLE note (see DB parameter chapter).

STAR_TRANSFORMATION (used in queries) and Parallel Query (used in change run, etc.) are mutually exclusive.

8.3 DB2 UDB EEE Performance Features

8.3.1 Parallel Database

IBM DB2 UDB EEE uses Parallel Database to achieve high levels of parallel DB accesses. This parallelism is mostly reached by partitioning DB tables with hash keys. SQL statements like SELECT, UPDATE, CREATE INDEX can be thus executed in parallel across partitions AND within partitions.

In addition, DB tools like Backup, Restore can also make use of high parallelism.

DB Partitions can be distributed across several DB server.

9 Analysis Tools

9.1 Application Tools

9.1.1 Upload Monitor

The Upload Monitor (transaction RSMO) provides all necessary information on times spent in different processes during the load (e.g., extraction time, transfer, posting to PSA, processing transformation rules, writing to fact tables). In the upload monitor you are also able to debug transfer and update rules.

If you encounter upload performance problem, try to identify within the upload monitor where the problem resides (e.g., transformation rules):

- If the extraction from a mySAP source system consumes significant time, use the extractor checker (transaction RSA3) in the source system.
- If the data transfer times are too high, check if too many work processes are busy (if so, avoid large data loads with “Update DataTargets in Parallel” method), and check swapping on one application server (set “rdisp/bufrefmode = “sendoff, exeauto” during load phase if you use several application servers).
- If the PSA upload times are unacceptable, check the system for I/O contention because of high number of writes (→ disk layout and striping, DB I/O) and the PSA partitioning configuration.
- If the upload bottleneck resides in transfer or update rules, please review your coding for performance “killers”. You can use the debug feature in the monitor, SQL trace or ABAP trace.

If the problem is the DataTarget, check all the recommendations given in the respective section. If the SQL trace shows significant time spent in reading table NRIV, enable number range buffering. If the SQL trace discovers many inserts into master data tables, check if the master data has been loaded before the transaction data.

See BW Documentation for more details on the data load monitor.

9.1.2 SAP Statistics

The SAP Statistics Cubes (technical Content) is a tool to evaluate performance key figures of queries, aggregate and roll-up processes in detail. The SAP statistics can be turned on and off for specific InfoProviders in the Administrator Workbench menu; old statistics data can also be deleted.

Query performance is split into OLAP time, DB time, number of selected and displayed records, and much more. They help to identify performance drawbacks and are also a basis for defining aggregates (as proposed in aggregate maintenance).

The sources of the information stored in the Technical Content InfoCubes are the tables RSDDSTAT*. Transaction ST03N and function module RSDDCVER_RFC_BW_STATISTICS also access these tables.

The Technical Content (or transaction ST03N) are usually the starting point for query performance analyses. Check first, if the time is spent on DB, in OLAP or in frontend.

Check the OLAP statistics InfoCube for InfoCubes and/or queries with highest runtimes. Aggregates will help if the following conditions are met:

- Ratio time spent on the database / total query runtime > 30% and
- Ratio records selected / records transferred > 10:1

These indicators can also be accessed in transaction ST03N.

See BW Documentation on Installing Technical Content, BW Documentation on Technical Content and SAPnote 309955 (BW statistics – questions, answers, errors) for more details.

Internal SAPnote 422264 (Characteristics and key figures BW statistics 2.0B/2.1C) describes the fields of table RSDDSTAT (there is no official SAP support for the fields in the table).

9.1.3 Query Monitor

The Query Monitor (transaction RSRT) allows you to execute queries and to trace queries in a debug mode with several parameters (e.g., do not use aggregates, do not use buffer, show SQL statement).

In the debug mode, you can investigate if the correct aggregate(s) are used and which statistics the query execution generates. For checking reasons, you can switch off the usage of aggregates, switch to no parallel processing (see for more details in the MultiProvider section) or display the SQL statement and the run schedule.

Moreover, you set some performance-related query features in RSRT: read mode and OLAP query cache mode.

If you discover single queries that you wish to improve in terms of performance, you should execute them in RSRT in the debug mode, and analyze which aggregates are used and which statistics are generated.

Check following issues in this sequence and verify the improvement after every optimization:

- Can aggregates help to reduce DB time (see more information in the SAP Statistics section)?
- Can the OLAP query cache reduce the overall query runtime? (The second query execution should result in a DB time = 0)

Is the run schedule optimal? (If not, check if DB statistics are up to date and if indices are used)

More information can be found in BW Documentation.

9.1.4 Query Trace Tool

The Query Trace Tool (transaction RSRTRACE) allows you to record some important function module calls and process and debug them at a later stage.

Transaction RSRCATTTRACE takes the log of RSRTRACE as input and gives aggregates suggestions for the first execution AND all further navigations performed.

Trace whole query navigations from users and analyze every step as described in the Query Monitor section.

More information can be found in BW Documentation and SAPnote 99010 (Description of the trace tool for RFC links).

Don't forget to switch off the trace after analyzing your queries. Traces produce some overhead in the production system.

9.1.5 Analysis and Repair of BW Objects

BW objects can be checked for consistency in transaction RSRV and inconsistent objects can be repaired. Moreover, there are some performance-related tools like database checks included. You can compare some basic DB parameters with the SAP recommendations and check indices and DB statistics for selected InfoCubes. You can also check for unbalanced InfoCubes, e.g., if the dimension table size is about 10% or more of the fact table size.

Use the database check tools within RSRV in regular intervals to analyze possible performance issues:

- Check if DB parameters are in synch with the SAP recommendations. The RSRV DB parameter check gives you a rough first guideline; detailed information is contained in the respective SAP note (see here).
- Check missing/old DB statistics for a particular InfoProvider
- Check missing/degenerated indices for a particular InfoProvider
- Check for unbalanced InfoCubes

9.2 System Tools

9.2.1 Process Overview

Transactions SM50/SM51 give you an overview on all running processes in your BW system. They provide information like user, current program, run time, CPU (additional button in the menu bar) and status.

Identify long running processes in SM50/SM51. Check the process ID (PID) with the PID in database monitor (ST04) to determine if the process is spending the time on the DB. In some cases it might be reasonable to debug processes via SM50/SM51 (this is only possible if the process is in ABAP).

For more details, see Documentation on Work Process Monitor and Documentation on Global Work Process Overview.

9.2.2 Workload Monitor

The Workload Monitor (transaction ST03) shows important overall KPI's for the system performance. It monitors (inter alia):

- Distribution of response times.
- Transactions with the highest response time and database time.
- Memory usage for each transaction or each user per dialog step.

The expert mode provides you information on BW queries, structured by runtime share, number of runs, and much more. This report is also available aggregated on the InfoProvider.

ST03 (and technical Content) should be the first address to identify performance problems for queries.

Identify the InfoProviders/queries with the highest runtime using the BW-related ST03-indicators and optimize them one by one.

See Documentation on Workload Monitor for more information.

9.2.3 SQL Trace

The SQL trace (transaction ST05) records all activities on the database and enables you to check long runtimes on a DB table or several similar accesses to the same data.

If you discover problems for an isolated process (upload or query) and you have analyzed for example the existence of aggregates, you can detail your analyses by using the SQL trace. Filter on a specific user (e.g. query user or extraction user ALEREMOTE) and make sure that no concurrent jobs run at the same time with this execution. You will find out which tables are accessed, what time is consumed and if some tables are accessed redundantly. Moreover you can "explain" statements to determine if the DB optimizer is choosing the correct access path.

See Documentation on Performance Trace for more information.

Make sure to switch off the trace after having used it!

Some tips for reading execution plans:

- Read the hierarchy from inside out
- At the same level: read a set of operations from top to bottom
- Try to avoid unspecified full table scans; generally prefer index accesses
- Generally prefer hash joins to nested loops

Short descriptions of some of the execution plan tree nodes:

- Full Table Scan: Accesses every record of a table

- Clustered Index Scan (MS SQL Server): access through clustered primary index; comparable to full table scan
- Nested Loop: For each relevant row in the first table, the matching rows in the other table have to be determined
- Hash Join: the smaller table is used to build up a hash table on the join key (optimal: in memory); then the larger table is scanned and joined with the hased table
- Bitmap Merge / Bitmap And (only ORACLE): Disjunction resp. conjunction of bitmap indices
- Partition Range Iterator: the process is executed for every partition

9.2.4 ABAP Runtime Analysis

The ABAP runtime analysis (transaction SE30) traces the activities spent in different parts of coding (SQL, specific function modules). It states number of executions, gross time (for global optimization) and net time (for local optimization). The gross time includes the times of all called sub-routines; the net time includes only the actual time spent in the mentioned module.

Use ABAP runtime analysis particularly for complex customer enhancements. You can use the ABAP trace “in parallel session” when you select the corresponding work process. The useful screens are: hit list, group hit list, table hit list and call hierarchy.

See Documentation on Runtime Analysis Tool for more information.

Make sure to switch off the trace after having used it!

9.2.5 OS, Memory and Buffer Monitor

The OS Monitor (transaction ST06) gives you an overview on the current CPU, memory, I/O and network load on an application server instance. In the detailed menu, you can also access a 24-hour overview of the hardware/OS load. Important indicators are CPU utilization, memory pages out (indicates swapping), disk utilization (indicates I/O bottlenecks) and LAN throughput (indicates network problems).

If the general system performance (i.e. all queries and processes) does not meet your expectations, check in ST06 if there are hardware/OS problems or if the system exceeds its hardware limitations.

Check especially CPU utilization which should be less than 70% in the hour’s average, and check that not more than 20% of the physical memory are swapped.

See Documentation on OS Monitor for detailed information.

Memory and buffer utilization of an instance can be checked in transaction ST02. Application buffers help avoid disk accesses by buffering records on the application server; they have big impact on query and load performance and BW administration. ST02 provides an overview on:

- Buffer quality (hit ratio), free size and free directory entries.

- Current memory usage.

The single-record buffer usually contains BW master data tables, the generic buffer contains most BW-specific control tables (RS*-tables) and the Export/Import Shared memory contains the OLAP query cache (BW 3.x). Note that the Export/Import Shared buffer does not swap.

Note that the buffers of an instance get refreshed when the instance is restarted.

If a SQL trace identifies certain DB accesses as time-consuming (e.g., master data tables during data load), check ST02 for the buffer quality (hit ratio, number of free entries, free space and swaps) and change the size if there are many swaps. Be sure that – after initial activities – the buffer qualities grow bigger than 90%.

See Documentation on Buffer Monitor for detailed information.

The table buffer call statistics can be monitored in transaction ST10. It shows buffer type, usage and invalidations.

If you assume a general buffer problem, check ST10 and check the buffer settings of all tables; compare usage of buffer vs. invalidations. Be sure that buffered tables have relatively small invalidation rates, single record buffered tables have relatively high direct ABAP reads, and generic key buffered tables have relatively high sequential ABAP reads.

If you use SEM on several application servers, please do not buffer table RSAPOADM (see SAPnote 562785 (Table RSAPOADM must not be buffered)).

See Documentation on Table Buffer Call Statistics for detailed information.

Buffering on several application servers can lead to synchronization problems (e.g., SID buffers during data load); in those cases, buffering can be turned off.

9.2.6 Database Monitor and Table/Index Overview

The database monitor (transaction ST04) checks important performance indicators in the database, such as database size, database buffer quality and database indices.

The user interface (UI) and the detailed functionality are depending on the underlying database management system.

If the general performance with respect to DB accesses is not satisfactory, check the database monitor. Check following indicators in ST04 (in start screen or detailed screen):

- Data buffer quality (should be > 97%).
- Check exclusive lockwaits for detecting deadlocks.
- Check DB log for special DB problems.

You can also monitor all processes in the database and check their execution plan (explain). You can also look for specific SQL statements, which have already been processed, and reproduce their execution plan. The execution plan gives you hints about missing (or degenerated) indices and out-of-date

statistics.

See Documentation on Database Monitor for further information.

The Table/Index Overview (transaction DB02) reports on the tablespaces, tables and indices. You can see the space statistics (total and free space) and its temporal growth, and you can check for missing indices. (Be sure to refresh the space statistics regularly).

The UI and the detailed functionality are depending on the underlying database management system.

Check transaction DB02 for missing and/or (especially in ORACLE) degenerated indices.

See Documentation on Table/Index Overview for further information.

9.2.7 Performance Analyses of Web Applications

The usual ABAP monitors cannot measure the time spent in the network and in the frontend. As of 3.0B SP10 there is an additional ping available for web applications in transaction RSRTRACE.

If there is a considerable gap between the ABAP monitors described above (e.g. SE30, SAP statistics) and the time experienced by the end users, use an http monitor or network sniffer to locate the problem.

- If the problem is the network, try to improve the bandwidth.
- If the problem is the browser rendering, check if you are using the latest version of the browser (Netscape Navigator or Microsoft Internet Explorer) – in some cases, replacing the browser with another one can help.

See documentation at SAP Service Marketplace and download the respective tools there (alias bw → performance).

See also internal SAPnote 377513 (Performance Analysis of Web Applications) for details.

10 Appendix

10.1 Performant ABAP customer enhancements

Most performance problems where a customer's own coding is involved (Z-objects) results from either sub-optimal algorithms or from bad DB accesses. If you discover performance problems in transformation rules or customer exits, please check the following issues.

10.1.1 Algorithms

A general guideline cannot be given, as there are thousands of possible algorithms for the specific problems.

The basic goal should be to avoid quadratic or exponential runtime (or worse). For example, binary sort (logarithmic runtime) is preferable to any quadratic sort algorithms.

If you are unsure, we recommend detailed literature like Algorithms by Robert Sedgewick.

10.1.1.1 Nested Loops

Generally speaking: Try to avoid nested loops as much as possible.

10.1.1.2 Unnecessary loops

Some statements need not be implemented by loops. For example, the deletion of records within an internal table can be implemented by one statement instead of using a loop on each record.

Loop at itab. If itab-field1 = condition. Delete itab. Endif. Endloop.	Delete from itab where field1 = condition.

10.1.2 DB Accesses

10.1.2.1 Avoiding unnecessary selects

Although it might sound very simple: Avoid selecting records that you are not going to use and use summary SQL (SUM, GROUP BY/HAVING) statements instead of programming your own logic.

Sum = 0. SELECT amount INTO Add FROM tab WHERE year = '2002'. Sum = Sum + Add. ENDSELECT.	SELECT SUM(amount) INTO Sum FROM tab WHERE year = '2002'.

Try to reuse data

SELECT f1, f2 FROM table WHERE cond	SELECT f1,f2,f3 FROM table WHERE cond

SELECT f1, f3 FROM table WHERE cond	

Avoid nested SELECTs – use FOR ALL ENTRIES, views or subqueries

LOOP AT I_kun SELECT * FROM knb1 WHERE kunnr = I_kun-kunnr. * do something ENDSELECT. ENDLOOP.	SELECT * FROM knb1 FOR ALL ENTRIES IN I_kun WHERE kunnr = I_kun-kunnr. * do something ENDSELECT.

10.1.2.2 Buffering database tables

It is generally recommended to use ARRAY operations instead of SINGLE accesses. This means that you read whole DB tables or bigger parts of DB tables into internal tables (in the start routine of the transformation rules) and access these internal tables for each record; rather than operating on the DB table for every record. However, try to keep the size of internal tables rather small.

If the DB table is completely buffered (see settings in transaction SE11), the first SELECT statement reads the whole table; additional buffering within the ABAP coding is not required.

LOOP AT I_kun SELECT * FROM knb1 WHERE kunnr = I_kun-kunnr. * do something ENDSELECT. ENDLOOP.	SELECT * FROM knb1 INTO itab FOR ALL ENTRIES IN I_kun WHERE kunnr = I_kun-kunnr. LOOP at itab. * do something ENDLOOP.

10.1.2.3 Accessing internal tables

Try to access internal tables diligently. Try to avoid “full table scans”; instead use sorted tables and binary search or hashed tables.

Hashed tables are usually good when single records are accessed and when data (keys) in the table is often changed.

Sorted tables are usually good for range accesses and require generally less memory than hashed tables.

Generally, use TRANSPORTING field list / NO FIELDS if useful and use ASSIGNING especially in nested tables.

10.1.2.4 Accessing DB tables

If you have to access DB tables, try to specify the whole key in the WHERE statement (in order to use the primary key) or that part of the key that is contained in one of the secondary indices (in the same sequence). Try to avoid ‘NOT’ in the WHERE clause, as there is no index support.

10.2 Related Topics

In this chapter, we present related topics like tips & tricks for the BW administration or minimizing downtime during extraction.

10.2.1 Temporary DB Objects

SAP BW uses temporary objects for the query processing and other processes. For performance reasons, they are defined outside the ABAP data dictionary.

See SAPnote 308533 (Temporary Database Objects in BW 2.0) and SAPnote 449891 (temporary Database objects in BW 3.0) for detailed information.

10.2.2 DB Storage Parameter

Fact and dimension tables of InfoCubes, aggregates, ODS Object tables and PSA tables can be assigned to their own tablespaces.

The set of default tablespaces has been reduced for BW 3.x: PSAP<SID>, PSAP<SID>USR, PSAP<SID>620, PSAPTEMP and SYSTEM. This has been done to simplify overall administration and to meet the requirements of MCOD, i.e. running several systems on one database.

Before implementing BW, reserve separate, specific tablespaces for large fact tables.

Separate tablespaces improve administration of grouped objects (e.g., archiving, ackup of a tablespace), especially for large tables residing within the tablespace.

If you use MCOD, be sure that you define unique tablespaces for each mySAP component connected to the same database.

10.2.3 Minimizing Downtime during Extraction

The delta initialization imposes some restrictions on the productive R/3 source system. Usually the workload is very high or the R/3 system may even have to be stopped for a while.

Apply this procedure when you initialize your delta update, the source system keeps a high volume of data and you want to guarantee a high availability of the system.

See How To … Minimize Downtime for Delta Initialization and SAPnote 436393 (Performance Improvement for Filling the Setup Tables) for more information.

The procedure requires a mirror system, i.e. your hardware must be doubled.

This feature requires Service API 30B and BW 3.0B.