



IBM SAP Technical Brief

SAP BW Query Performance with DB2 on zSeries

Mark Gordon

IBM Solutions Advanced Technical Support

Version: 2.0
Date: October 22, 2003

1. Acknowledgements	5
2. Disclaimers	5
3. Copyrights.....	5
4. Version Changes	5
5. Feedback	6
6. Introduction	6
6.1. Audience	6
6.2. Additional documentation	7
6.3. Explain Plan	7
6.4. DB2/390 and SAP background information.....	7
6.5. SAP BW Terminology.....	8
6.5.1. Star Schema	8
6.5.1.1. Fact Table.....	8
6.5.1.2. Characteristics	8
6.5.1.3. Dimensions.....	8
6.5.1.4. Master Data	9
6.5.1.5. Line-item Dimension	9
6.5.1.6. Navigation Attributes.....	10
6.5.1.7. Time-dependent Attributes.....	10
6.5.1.8. Naming Conventions	11
6.5.1.9. SAP Star Schema in DB2.....	12
6.5.2. Query Designer Terms.....	13
6.5.2.1. Columns	13
6.5.2.2. Rows	13
6.5.2.3. Free Characteristics	13
6.5.2.4. Filter.....	14
6.5.2.5. Relationship between query definition and aggregate definition.....	14
6.5.3. Compression	14
6.5.3.1. DB2 compression	14
6.5.3.2. SAP compression.....	14
6.5.4. Aggregate.....	15
6.5.5. ODS	15
6.5.6. Navigation Step.....	16
6.5.7. Partitioning.....	16

6.5.7.1.	Fact table partitioning	16
6.5.7.2.	ODS table partitioning	18
6.6.	<i>DB2 Query Parallelism</i>	18
6.6.1.	Impact of query parallelism on statistics	18
6.7.	<i>Query execution using views</i>	21
6.8.	<i>Star Join</i>	22
7.	Strategy Roadmap	24
7.1.	<i>Individual Query Optimization</i>	24
7.2.	<i>System-wide optimization</i>	24
8.	Correlating ST04 statement cache and SAP statistics	24
8.1.	<i>Enhanced ST04 with query correlation</i>	25
8.2.	<i>Manually correlating ST04 cache statistics and RSDDSTAT</i>	26
9.	Indicators of performance problems for DB2 running BW	26
9.1.	<i>Key ratios that can indicate performance problems</i>	26
9.2.	<i>Interpreting RSDDSTAT</i>	27
9.2.1.	Possible z/OS or DB2 performance problem	28
9.2.2.	Need for new aggregate	29
9.3.	<i>Interpreting RSDDSTATAGGRDEF</i>	30
9.3.1.	Basic infocube	30
9.3.2.	Multicube example	32
9.4.	<i>SAP query settings that affect performance</i>	33
10.	Solving a performance problem for a specific query	35
10.1.	<i>Interactive testing with RSRT</i>	35
10.2.	<i>Starting from query statistics</i>	41
10.3.	<i>Starting from ST04 statement cache</i>	44
10.4.	<i>Sample explain of V7 sparse index access path</i>	44
11.	Reviewing overall efficiency of the system	45
11.1.	<i>Examine infocubes and queries using lots of database resources</i>	45
11.1.1.	ST03N to find infocubes with long DB time	46
11.1.2.	Excel spreadsheet to find slow queries	48
11.2.	<i>Review need for aggregate tables</i>	49
11.2.1.	Strategies for including attributes in an Infocube	49
11.2.2.	Navigation attributes in aggregate dimension table	49
11.2.3.	SID index columns in aggregates	50

11.2.4.	Navigation attribute processing with aggregate query	56
11.2.5.	Time dependent attributes in aggregates	57
11.2.6.	Defining a navigation attribute and its characteristic in an aggregate	59
11.2.7.	Recap of characteristic and navigation attribute definition in aggregates.....	61
11.2.7.1.	Aggregate defined at characteristic level.....	61
11.2.7.2.	Aggregate defined at characteristic level with navigation attribute	61
11.2.7.3.	Aggregate defined at level of navigation attributes.....	62
11.2.8.	Propose aggregates in RSA1	63
11.2.9.	Find aggregates which support many queries using roll-up hierarchy.....	66
11.2.10.	Evaluate proposals and determine which aggregates to create.....	68
11.2.10.1.	Two aggregates that should not be merged.....	68
11.2.10.2.	Two aggregates which can be merged.....	70
11.2.10.3.	Aggregate hierarchies	72
11.2.11.	An RYO solution.....	76
11.2.12.	Sample tools for aggregate proposal	77
11.2.13.	View on RSDDSTAT and RSDDSTATAGGRDEF	77
11.2.14.	Proposing an aggregate for an individual query.....	80
11.3.	<i>Review SQL cache.....</i>	82
11.4.	<i>Further actions after aggregates are defined.....</i>	83
11.4.1.	OLAP cache to offload database server	83
11.4.2.	Correlation information may improve access path	84
11.4.3.	More column distribution information may improve access path	91
11.4.4.	New index may improve access path	95
11.4.5.	Check data model.....	100
11.4.6.	Not a data model problem.....	100
11.4.7.	Evaluate new indexes on fact table	102
11.4.7.1.	Searching for filtering dimensions and characteristics.....	103
11.4.7.2.	Using listcube to determine dimension filtering	107
11.4.8.	Check for symptoms of I/O constraints.....	113
11.4.9.	Check DB2 buffer pools for signs of constraints.....	114
12.	Appendix 1: correlating RSDDSTAT and ST04 before 3.0B SP 15.....	119
13.	Appendix 2: Reference Materials.....	121
13.1.	<i>IBM manuals.....</i>	<i>121</i>

1. Acknowledgements

Several people made important contributions that made this paper possible.

Thank you to Terry Purcell for his contributions on the many and various ways that DB2 uses catalog statistics, indexes and access path selection to optimize queries with star schema objects.

Thank you to Joachim Rese for his contributions on how SAP BW on zSeries exploits DB2 functionality.

Thank you to Annie Tsang and Patrick Bossman for their contributions on the describing the functionality of star join, and how to interpret and optimize SQL for star schema.

Thank you to Phil Hardy for editing the paper.

2. Disclaimers

The paper contains examples from SAP BW systems running SAP BW 2.0 to 3.1 with DB2 V6 and V7..

The processes and guidelines in this paper are the compilation of experiences analyzing performance on SAP BW systems. Your results may vary in applying them to your system. Some examples have been created and/or edited to clarify points for this paper.

Most aspects of traditional DB2 and OS/390 tuning (buffer pools, EDM, logging) are not discussed here, but can be found in DB2 manuals. The paper is focuses on integrating SAP and DB2 statistics related to query performance.

For examples of tuning issues with SAP R/3 and DB2/390, see the document “Tuning SAP / DB2 / zSeries” on IBM techdocs (www.ibm.com/support/techdocs).

This paper has not been formally reviewed by IBM, and while effort has been made to verify the information, this paper may contain errors. IBM makes no warranties or representations with respect to the content hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. IBM assumes no responsibility for any errors that may appear in this document. The information contained in this document is subject to change without any notice. IBM reserves the right to make any such changes without obligation to notify any person of such revision or changes. IBM makes no commitment to keep the information contained herein up to date.

3. Copyrights

SAP and R/3 are copyrights of SAP A.G.

DB2, Universal Database, MVS, OS/390, and z/OS are copyrights of IBM corp.

Oracle is a copyright of Oracle corp.

Excel is a copyright of Microsoft corp.

4. Version Changes

Version 2.0, added:

- Section 6.5.2

- Section 6.5.5
- Section 6.5.7.2
- Section 8.1
- Section 10 reworked, and some parts added
- Section 11.2.1
- Section 11.2.12
- Section 11.4.1
- Section 11.4.2
- Section 11.4.3
- Section 11.4.4
- Section 11.4.7.2

5. Feedback

Please send comments or suggestions for changes to gordonmr@us.ibm.com.

6. Introduction

6.1. Audience

The goal of this paper is to give an integrated view of key SAP and DB2 performance indicators, and offer a process for using the two together.

The audience for this document is DB2 DBAs who are responsible for administering DB2 for OS/390 and z/OS running SAP BW system, or BASIS/BW administrators responsible for a SAP BW system on DB2 for OS/390 and z/OS. The focus is on query performance problems, and how to solve them.

Evaluation of query performance for SAP BW requires using both SAP information (such as query performance indicators in RSDDSTAT, or aggregate valuation and definition in RSA1) and also DB2 indicators (such as cached statement statistics, or using DB2 EXPLAIN PLAN to analyze the access path used by DB2).

In general, a DBA would not be responsible for some of the processes in this paper, such as analyzing and defining SAP aggregates, which are summary tables optimized for specific queries. It is important for the DBA to be aware of these activities, since aggregate analysis and definition should generally be done before making DB structure changes, such as adding new indexes. As an example, doing DB2 tuning will not solve a problem, when the root problem is a lack of aggregates. Missing or improperly defined aggregates can cause increased CPU usage, I/O activity, and DB2 memory usage.

Likewise, a BW administrator may not be familiar with the technical issues of DB structure, SQL evaluation, and DB2 parallelism capabilities. But if after evaluating the data model and aggregates, there are still performance problems, the BW administrator can work with the DBAs to evaluate DB2-related changes to address performance problems. Not all problems can be solved with aggregates, and the BW administrator may need to turn over some problems to the DBA.

6.2. **Additional documentation**

The SAP ASAP BW documentation has information on data modeling, infocubes, etc. ASAP documentation is delivered with the SAP install kit for BW. It describes some of the SAP information and techniques below, but does not contain information on how to integrate the SAP and DB2 analysis.

The DB2 Administration guide (see section 12) has detailed descriptions of explain plan, access paths, and star join.

There is a technical white paper available describing DB2 Star Join at <http://www.ibm.com/software/db2os390>. Choose: Support > White papers > [Evolution of Star Join Optimization - DB2 for z/OS and OS/390](#).

6.3. **Explain Plan**

The default SAP EXPLAIN PLAN format converts the DB2 PLAN_TABLE to english, and displays it in a hierarchical display. On a BW system, since BW queries can contain many tables, the explain plan SAP output can be many pages long. Since these reports would be too long to include in this paper, the paper uses the DB2 PLAN_TABLE format, which was formerly called “expert mode” in SAP. Full details of PLAN_TABLE output are in the DB2 Administration Guide. This paper will point out a few key items in the EXPLAINs used in the examples, but not give a detailed analysis.

6.4. **DB2/390 and SAP background information**

SAP uses DB2 in ways that make traditional DB2 performance indicators, such as plan-based accounting statistics, not useful.

- All SAP work processes use the same DB2 Plan
- All SAP query SQL is dynamic SQL.
- SAP work processes remain attached to DB2 for hours or days, and run many queries through a single thread.

Monitoring SAP BW is also different from monitoring SAP R/3.

- SAP BW performance statistics are not in ST03, but are available in ST03N via support package updates.
- SAP BW performance statistics are available in RSDDSTAT and RSDDSTATAGGRDEF tables (or in the 0BWTC_C03 and 0BWTC_C03 statistics cubes).
- DB2 parallelism is enabled for SQL accessing fact tables
- SAP infocubes use a star schema
- SQL statements often reference many more tables than R/3 transaction or reporting SQL

The result is that traditional DB2 plan and accounting data is of limited use, and new SAP statistics (RSDDSTAT and RSDDSTATAGGRDEF, or statistics infocubes) contain important performance information.

SAP BW query performance can be reviewed using ST03N, queries on the statistics infocubes, or SE16 to read SAP tables. This paper will show examples of each. ST04 statement cache analysis is used to check SQL in DB2 – access path, table and index attributes, performance statistics, etc.

6.5. SAP BW Terminology

6.5.1. Star Schema

SAP uses the star schema in defining and creating infocubes that are used for BW queries. The star schema is made up of a central fact table, and surrounding dimension and master data tables. Please see **Figure 3** for a graphical representation of the relationship of the tables in an SAP infocube. **Figure 3** does not show all objects, such as all master data tables, or indexes on table keys. It shows the relationships between the fact table and the tables that surround the fact table -- dimension tables, SID tables (for master data), and Attribute SID tables (for navigation attributes).

6.5.1.1. Fact Table

A fact table contains facts (for example, requested quantity, allocated quantity, total price) about a business transaction, such as a sales order. Facts are items that can be used arithmetically (summed or averaged, for example). In the DB2 database, fact tables will have names like '/BIC/E*', '/BIC/F*', '/BI0/E*', or /BI0/F*'. The key of the Fact table is made up of DIMID (Dimension ID) columns, which have a foreign key relationship to the dimension tables.

6.5.1.2. Characteristics

There are also characteristics associated with each row in the fact table. These are generally not arithmetic, and are attributes such as salesperson, customer, date, product id. In reporting, characteristics are used to select and group the rows in the fact table. Characteristic values are not stored in the fact table, but are stored in dimension tables and master data tables.

6.5.1.3. Dimensions

A dimension is made up of a group of characteristics that are related to each other. A product dimension might contain product code, supplier, and marketing company. The time dimension might contain calendar day, month, and posting period. Instead of the actual characteristic value (e.g. salesperson name 'John Smith'), the dimension table contains a SID (surrogate ID), which is a key that uniquely identifies John Smith in the master data table for salespeople. In the database, dimension tables will have names like '/BIC/D*' or /BI0/D*'.

The key of the dimension table is the DIMID (Dimension ID). Each DIMID corresponds to a unique combination of characteristics in the dimension. For example, if a dimension table contains product code, supplier, and marketing company, there will be a different DIMID for each unique combination of product code, supplier, and marketing company in the dimension table. DIMIDs are assigned by SAP, and are created dynamically as new characteristic combinations are created. For instance, when new product codes are created, there will be new DIMIDs created, one for each new product code, supplier, and marketing company combination in the dimension table.

6.5.1.4. Master Data

SID tables contain the Surrogate ID values for a characteristic. For example, if customer number is a characteristic in the customer dimension, then there will be a customer number SID table containing the customer number and SID. The SID is an SAP-generated number that stands for the master data value, which is how the SID gets its name “Surrogate ID”. In the database, SID tables will have names like ‘/BIC/S*’ or ‘/BI0/S*’. When a user specifies the customer number in a query, DB2 uses the associated SID to access the dimension (or sometimes fact) table.

Other master data tables (/BIC/M*, /BI0/M*, /BIC/T*, etc) contain the rest of the information (name, region, sales organization, etc) about the customer. When new master data items are created (e.g. a new customer was acquired) then a new SID will be created by SAP to correspond to the new master data value.

6.5.1.5. Line-item Dimension

A line-item dimension contains only one characteristic. Since there is only one characteristic, no dimension table is needed – the master data SID table is the key for the line item dimension in the fact table.

“Line item dimension” is a customization option that a BW administrator can use when defining the characteristics and dimensions of an infocube. There are two general reasons why a characteristic might be defined as a “line item dimension”

- The characteristic is independent of other characteristics of the infocube, and does not be logically fit in another dimension.
- The cardinality (number of unique values) of the characteristic is so large that if it were in a dimension with other characteristics, the dimension would be too large for efficient SQL processing. See section 11.4.5 for an example of a dimension that is too large for its fact table.

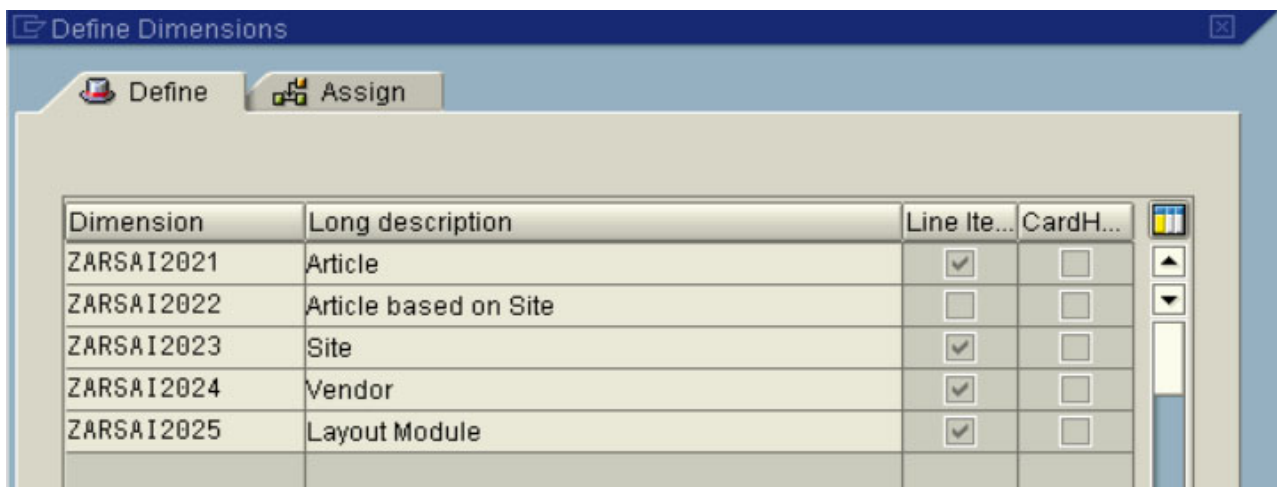


Figure 1: RSA1 definition of line item dimensions

6.5.1.6. Navigation Attributes

Master data tables can also contain navigation attributes, which are used for searching and grouping master data. For instance, a sales area navigation attribute on salesperson master data would group or select sales people according to sales area. At the database level, a navigation attribute will be a column within the master data table. For example, a salesperson master data table may contain columns for salesperson personnel number, sales manager SID, and sales area SID. In this case, sales manager and sales area are navigation attributes which link to the “attribute SID” tables for sales manager and sales area navigation attributes. This structure allows queries such as all sales people reporting to a sales manager, or all sales people assigned to a sales area. Navigation attributes are stored in “attribute SID” tables with names like ‘/BIC/X*’, ‘/BI0/X*’, ‘/BIC/Y*’, and ‘/BI0/Y*’.

6.5.1.7. Time-dependent Attributes

If an attribute of a characteristic can have different values at different dates, it is called “time dependent”. An example might be the “product group” attribute for the characteristic “product” -- if a company changed product groupings on January 1st, 2003, and the product “widget” was moved from one product group to a new group, then reports for the time period before 1/1/2003 would report widget sales in their former product group, and BW reports for the time period after 1/1/2003 would report widget sales in their new group.

Prior to BW 3.0, time-dependent attributes cannot be defined in aggregates. If a user on a BW 2.x system uses a time-dependent attribute in a query, the time-dependent attribute will be evaluated at runtime, when the SQL is executed. DB2 will use the time-dependent attribute table to create the result. The time-dependent attribute tables (see section 0) must be joined to the fact table to evaluate the value of the attribute at the specified date.

With BW 3.0, time-dependent attributes can be defined in aggregates, and the aggregate will be built using the state of the time-dependent attributes on a single ‘Key Date’, so that the aggregate will be used only for reports as of that Key Date. Thus, *time dependent navigation attributes are most useful for creating aggregates for end of period reports, on a specific date where the status of the business data is more important than on other days.*

6.5.1.8. Naming Conventions

Figure 2 shows the naming conventions used in DB2 for different types of SAP objects.

Name	Type	Description
/BIx/ F <InfoCube>	T	F fact table
/BIx/ E <InfoCube>	T	E fact table
/BIx/ D <InfoCube> <i>n</i>	T / V	Dimension table of dimension <i>n</i> <i>n</i> = P,T,U,1,....,9,A,....,D
/BIx/ S <InfoObject>	T	SID table
/BIx/ M <InfoObject>	V	Master data attributes (join of P + Q)
/BIx/ P <InfoObject>	T	Master data attributes (time independent)
/BIx/ Q <InfoObject>	T	Master data attributes (time dependent)
/BIx/ T <InfoObject>	T	Master data texts table
/BIx/ X <InfoObject>	T	Attributes SID table (time independent)
/BIx/ Y <InfoObject>	T	Attributes SID table (time dependent)
/BIx/ I <InfoObject>	T	Hierarchy SID table
/BI0/ 02 <8-digits>	T	Hierarchy reporting nodes (cached)
/BI0/ 03 <8-digits>	V	View on hierarchy nodes View on reporting query
/BI0/ 01 <8-digits>	T	Table on hierarchy nodes Temporary SID table (2.x)
/BI0/ 06 <8-digits>	T	Temporary SID table (3.x)
/BIC/ B <10-digits>	T	PSA / ODS Object table

Figure 2: Naming Conventions

6.5.1.9. SAP Star Schema in DB2

Figure 3 is a simplified version of the SAP Star Schema in DB2. It does not contain all types of tables, and does not include indexes. It does not show line-item dimensions. It is meant to show the overall relationship between the central fact table and surrounding dimension, master data, and attribute SID (navigation attribute) tables.

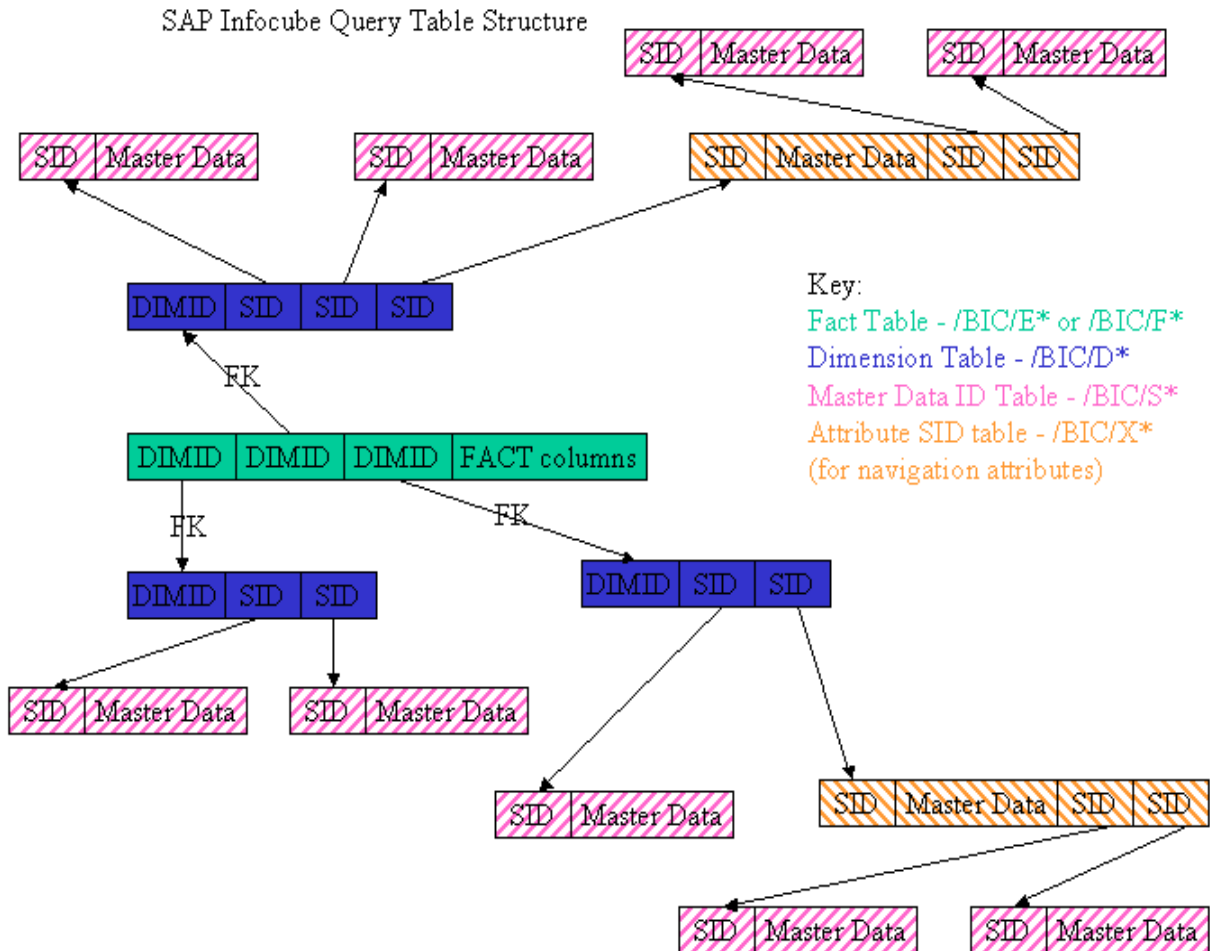


Figure 3: Simplified SAP infocube query structure

6.5.2. Query Designer Terms

The SAP query designer tool has yet more names for characteristics and key figures. Characteristics and key figures are named based on their function in the query.

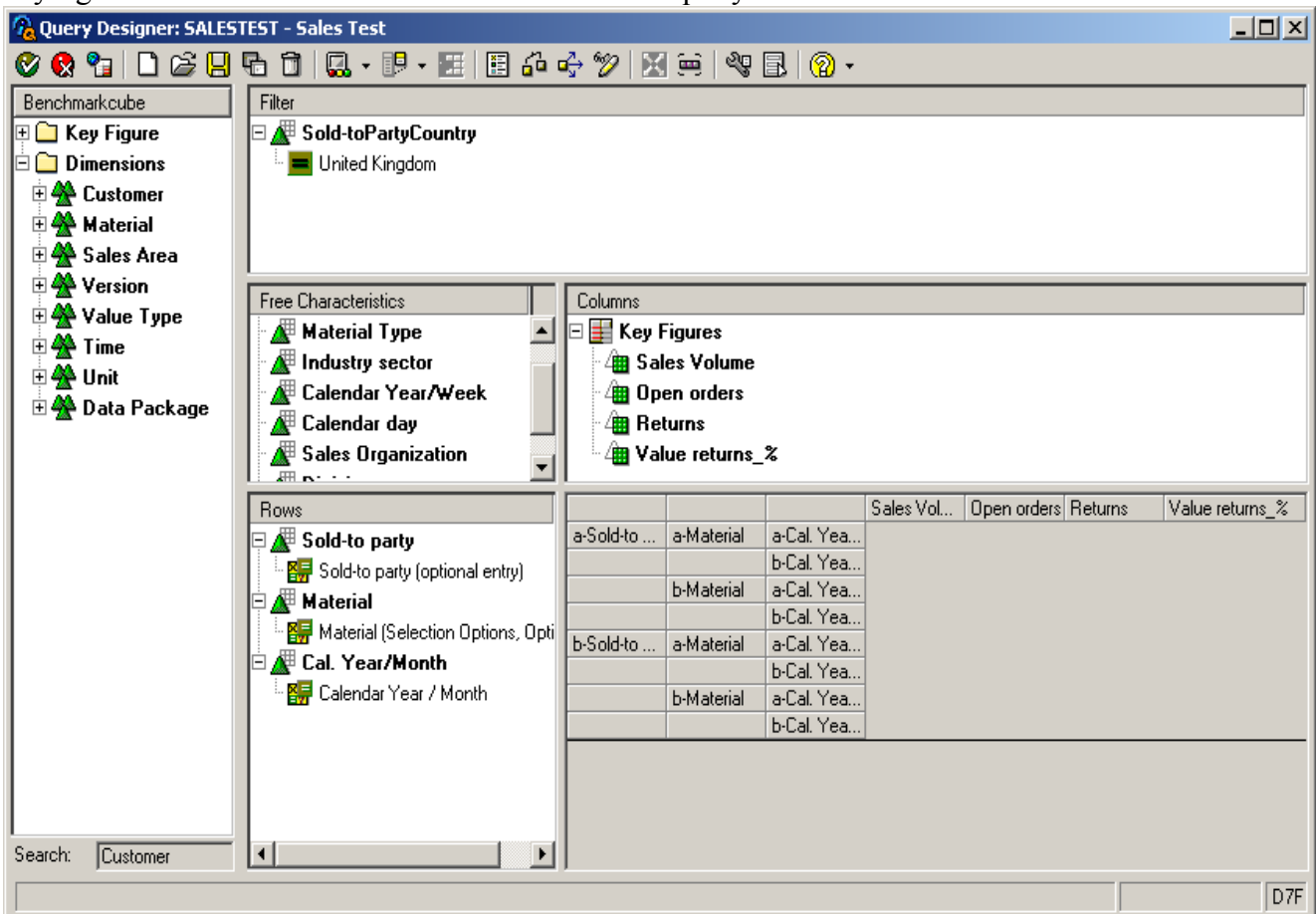


Figure 4: Query designer

6.5.2.1. Columns

Query columns are generally key figures (columns in a fact table) whose value is calculated in the DB or in SAP and then reported in the query. The table columns corresponding to query columns will be present in the SELECT clause of the SQL executed by DB2.

6.5.2.2. Rows

Rows are characteristics and navigation attributes that group the data and determine the report breaks. The table columns corresponding to these characteristics and navigation attributes will be in the GROUP BY clause of the query SQL. If a restriction is placed on a row, then the table column corresponding to the row will also be present in the WHERE clause of the SQL.

6.5.2.3. Free Characteristics

Free characteristics are characteristics that the user may add, via drill down or filter value selection, to the set of characteristics defined in the row section, as shown in Figure 4. They are available for use, but are not part of the query and SQL until selected by the user.

6.5.2.4. Filter

Filters are used to specify characteristics used to select the data. Filters will appear in the WHERE clause of the SQL.

6.5.2.5. Relationship between query definition and aggregate definition

The set of characteristics contained in the 'rows', 'filter', and 'columns' sections is the minimum set of characteristics that will be used with the query, and corresponds to the most summarized aggregate that could be used to support this query.

Once a user added a filter or drill-down on a 'free characteristic', the minimum aggregate could not be used to support the query.

The set of characteristics contained in the 'rows', 'filter', 'columns', and 'free characteristics' is the maximum set of characteristics that could be used with this query, and corresponds to the least summarized (most general) aggregate that could be needed by this query.

6.5.3. Compression

Both SAP and DB2 use the term 'compression', but 'compression' has different meanings for each. Both types of compression are useful in a DB2/BW environment.

6.5.3.1. DB2 compression

DB2 compression (also called 'hardware compression') is a feature of DB2, OS/390 and zSeries that compresses data in tables to reduce the space needed to store the table. It can reduce the number of bytes in a row of table data. The data is stored on disk and in bufferpool memory in compressed form, and is uncompressed or compressed by S/390 hardware instructions when it is referenced. Since Fact tables can be very large, compressing Fact tables can be very beneficial in saving disk storage, and reducing I/O activity. ODS tables are also generally good candidates for compression.

6.5.3.2. SAP compression

Each infocube or aggregate has two fact tables that can be used to hold data:

- F fact table (e.g. /BIC/Ftablename), where data is stored with a packet ID. If there are errors in sending data to BW, the data can be removed based on its packet ID.
- E fact table (e.g. /BIC/Etablename), where packet ID is zero.

SAP compression implements an "application partitioning" where the E fact table contains data used only for reporting, and the F fact table contains data which is used for administrative tasks (acceptance check of data, rollups to aggregates) and for reporting. If compression has been enabled, reporting from the F fact table is optional. The data in the F fact table can be used in reporting if it is released for reporting in BW. Rather than reporting from the F fact table, after the correctness of the data has been checked in the F fact table, data can be moved to the E fact table for reporting.

SAP compression is a process of moving rows from the F fact table to the E fact table. When the rows are compressed into the E fact table, rows that differ only in packet ID are merged into a single row in the E fact table, and the packet ID is set to zero.

SAP compression can reduce the number of rows in the table, and by removing packet ID can make DB2 star schema access to the fact table more efficient.

F fact tables should be small, and contain data long enough to check the validity of the data packets while the data is staged for transfer to the F fact table. E fact tables will be large.

When SAP compression is not implemented, and all the fact table rows reside in the F fact tables, queries and aggregate rollups using the fact tables may run longer than when SAP compression is implemented.

- Aggregate rollups may run longer, due to the large size of the F fact table, since aggregate rollup extract only the latest packets from the F fact table. When compression is implemented, the F fact table is smaller, and can be more quickly processed.
- Queries may run longer since additional predicates on the P dimension are necessary when selecting from the F fact table, compared to selecting from the E fact table, where the P dimension “Packet ID” is always 0.

6.5.4. Aggregate

An aggregate is a summary table created from the data in an infocube. *Aggregates are the most important tool for improving SAP BW query performance.*

- **When a query uses an aggregate with summarized data, it will fetch fewer rows from the database, compared with fetching the un-summarized rows from the infocube.** The dimension tables for an aggregate contain fewer characteristics than the infocube that it is built from. When multiple rows in the infocube are the same for all the characteristics included in the aggregate, they are summarized to a single row in the aggregate. Since there are fewer characteristics in the aggregate, the aggregate’s dimension and fact tables will contain fewer rows than the dimension and fact tables of the infocube used to build the aggregate. This summarization reduces the I/O and CPU required to retrieve the query result.
- **Aggregation can simplify the access path for SELECT operations.** Navigation attributes that are defined as part of the aggregate are incorporated as columns in the aggregate dimension tables. Thus a select on an aggregate using one of its navigation attributes can be evaluated using only the fact and dimension tables of the aggregate. Navigation attributes for infocubes are stored in separate attribute SID tables, so a query using a navigation attribute for an infocube must join the attribute SID table, the dimension table, and the fact table. Incorporating the navigation attributes into the dimension tables reduces the number of tables that must be joined to produce the query result, which can allow DB2 to process the aggregate query more quickly than the same query on an infocube. If the aggregate has few characteristics, it can be indexed by SID, so that DB2 can select rows from the fact table without joining a dimension table to the fact table.

6.5.5. ODS

The ODS (Operational Data Store) is used in BW to support reporting on data that is more dynamic than the data in infocubes, for example reporting on unfulfilled orders, or detailed reports on recent orders at the order or line-item level. The usual query tools (Bex, etc) are used to query ODS.

ODS cannot be optimized via aggregates, but one can add new indexes to support query requirements.

When a navigation step accesses the ODS, the RSDDSTAT field DBTDBODS will be >0, along with QTIMEDB being >0.

6.5.6. Navigation Step

A navigation step (e.g drill-down or specify characteristic values in a report) is the BW analog of the R/3 dialog step – it is the unit used in reporting performance. Not all navigation steps involve calls to the database server. When the database server is called, then RSDDSTAT QTIMEDB will be >0. There may also be more than one DB2 SQL query operation in a navigation step, since an infocube can

- Contain data in both F and E fact tables, or
- Be summarized into an aggregate table, or
- Comprise more than one fact table (as with a multicube).

SAP query performance data available using ST03N and SE16 (table RSDDSTAT) is reported in units of navigation steps.

6.5.7. Partitioning

Since aggregates are the best way to improve performance of commonly run BW queries, consider using DB partitioning to

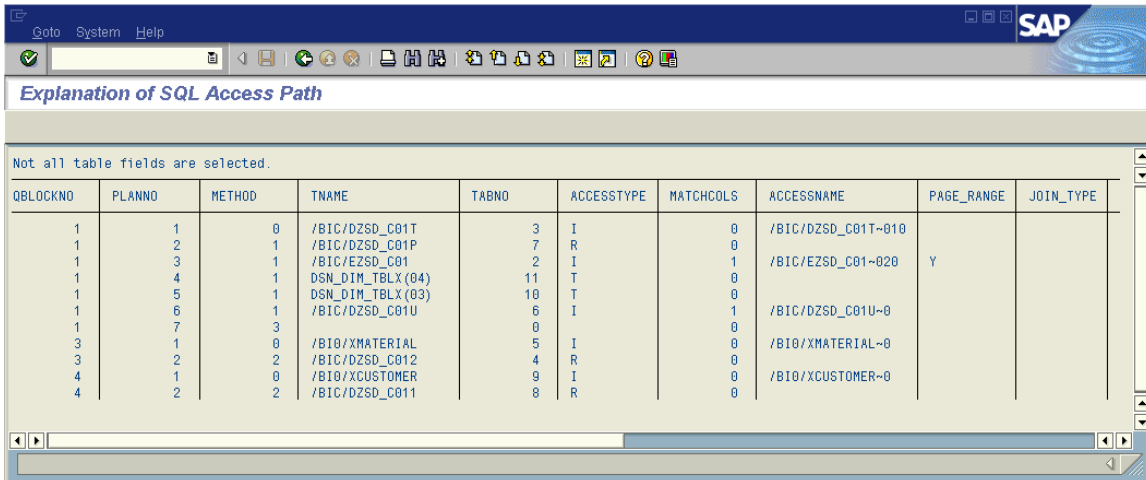
- Split large tables into several partitions, to support parallel operation of administrative activities such as backup, recovery, and runstats.
- Support tables that will exceed the size limit of non-partitioned tablespaces.
- Enable use of partition-based operations such as partition pruning, or parallelism, for I/O constrained ad-hoc queries for which there is no suitable aggregate.

6.5.7.1. Fact table partitioning

Fact table partitioning is possible using SAP RSA1, or by using DB2 utilities. *SAP RSA1 is the recommended way to implement partitioning of fact tables.* RSA1 can be used to implement partitioning of E fact tables. The E fact table must be empty at the time that partitioning is enabled, since SAP will add the partitioning characteristic column into the fact table.

SAP fact table partitioning can only be implemented on compressed (E) fact tables. The E fact tables can be partitioned based on SAP time (0CALMONTH, 0FISCPER) characteristics. Since most queries have local predicates on time, this will enable DB2 to determine at prepare time which partitions can be eliminated, which can improve query performance.

Figure 5 is an “explain plan” from a query against a partitioned E fact table. PAGE_RANGE = Y shows that DB2 is able to eliminate unneeded partitions when the statement was prepared.



Explanation of SQL Access Path

Not all table fields are selected.

QBLOCKNO	PLANNO	METHOD	TNAME	TABNO	ACCESSTYPE	MATCHCOLS	ACCESSNAME	PAGE_RANGE	JOIN_TYPE
1	1	0	/BIC/DZSD_C01T	3	I	0	/BIC/DZSD_C01T-010		
1	2	1	/BIC/DZSD_C01P	7	R	0			
1	3	1	/BIC/EZSD_C01	2	I	1	/BIC/EZSD_C01-020	Y	
1	4	1	DSN_DIM_TBLX (04)	11	T	0			
1	5	1	DSN_DIM_TBLX (03)	10	T	0			
1	6	1	/BIC/DZSD_C01U	6	I	1	/BIC/DZSD_C01U-0		
1	7	3		0		0			
3	1	0	/B10/XMATERIAL	5	I	0	/B10/XMATERIAL-0		
3	2	2	/BIC/DZSD_C012	4	R	0			
4	1	0	/B10/XCUSTOMER	9	I	0	/B10/XCUSTOMER-0		
4	2	2	/BIC/DZSD_C011	8	R	0			

Figure 5: E fact table access via PAGE_RANGE = Y

Since DIMIDs (fact table key values) are created dynamically, it is difficult to determine a good partitioning strategy at the DB2 level. A DB2-level partitioning key that is good today may not be evenly balanced in the future, and may not distribute the data across partitions in a logical way. F fact tables cannot be partitioned using RSA1, but since F fact tables should usually be small, as discussed in section 6.5.3.2, this is generally not a problem.

6.5.7.2. ODS table partitioning

ODS tables (see section 6.5.5) can be partitioned using DB2 utilities. Since the key columns in ODS objects are columns containing characteristic values (unlike the fact table, where the keys are based on dynamically generated DIMIDs), one can review the range of values of master data for the characteristics, in order to determine good partitioning key and key ranges.

As with E fact tables, time (here, 0CALDAY) is often a good choice to use for partitioning, if ODS queries will include local predicates for time.

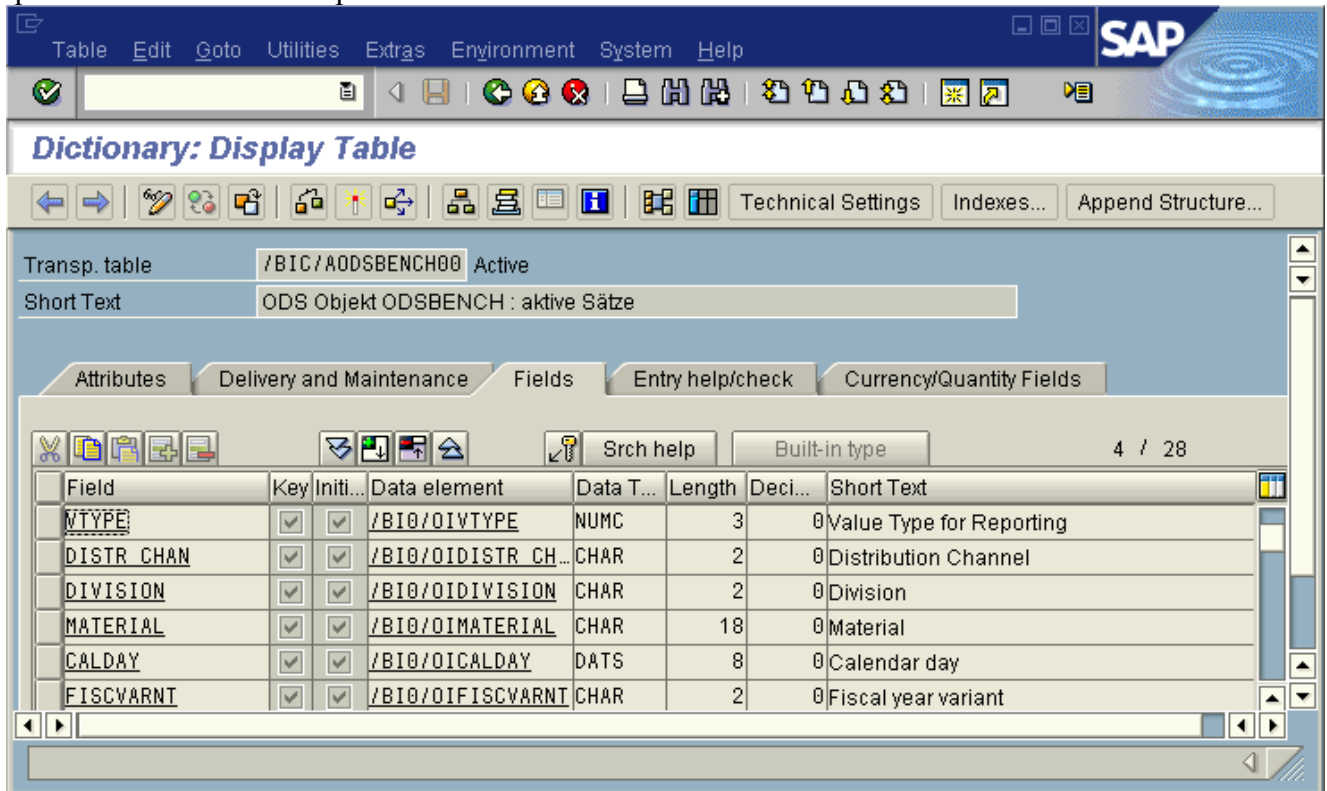


Figure 6:ODS object key structure

6.6. DB2 Query Parallelism

SAP automatically enables DB2 query parallelism (SET CURRENT DEGREE = 'ANY') when accessing fact tables. There are several DB2 parameters and settings that affect DB2 query parallelism. See SAP note 390016 for details on required and recommended settings for BW on DB2 for zSeries.

6.6.1. Impact of query parallelism on statistics

At the time this document was written, there were two situations where query parallelism may cause SAP DB2 statistics to be incomplete, due the way DB2 collects statistics for parallel query.

When parallel query is used to execute an SQL statement, all the statistics related to parallel child processes are aggregated in one statistics area, the parent query process stats are collected in the thread statistics.

- In ST04 statement cache statistics, when query parallelism is used to execute a statement, the statement statistics contain only the main thread statistics, not parallel child statistics.
- In ST04 thread statistics, the parallel child statistics are not rolled up together with the main thread statistics.

The impact of this is that the indicators described in section 9.1 may not be correct, if a query is executed using DB2 query parallelism. If you encounter a situation where the statement statistics described in section 9.1 (getpages, rows examined, etc) seem too low for the statement elapsed time, QDBSEL and QDBTRANS, check the statement statistics to see if parallelism was used.

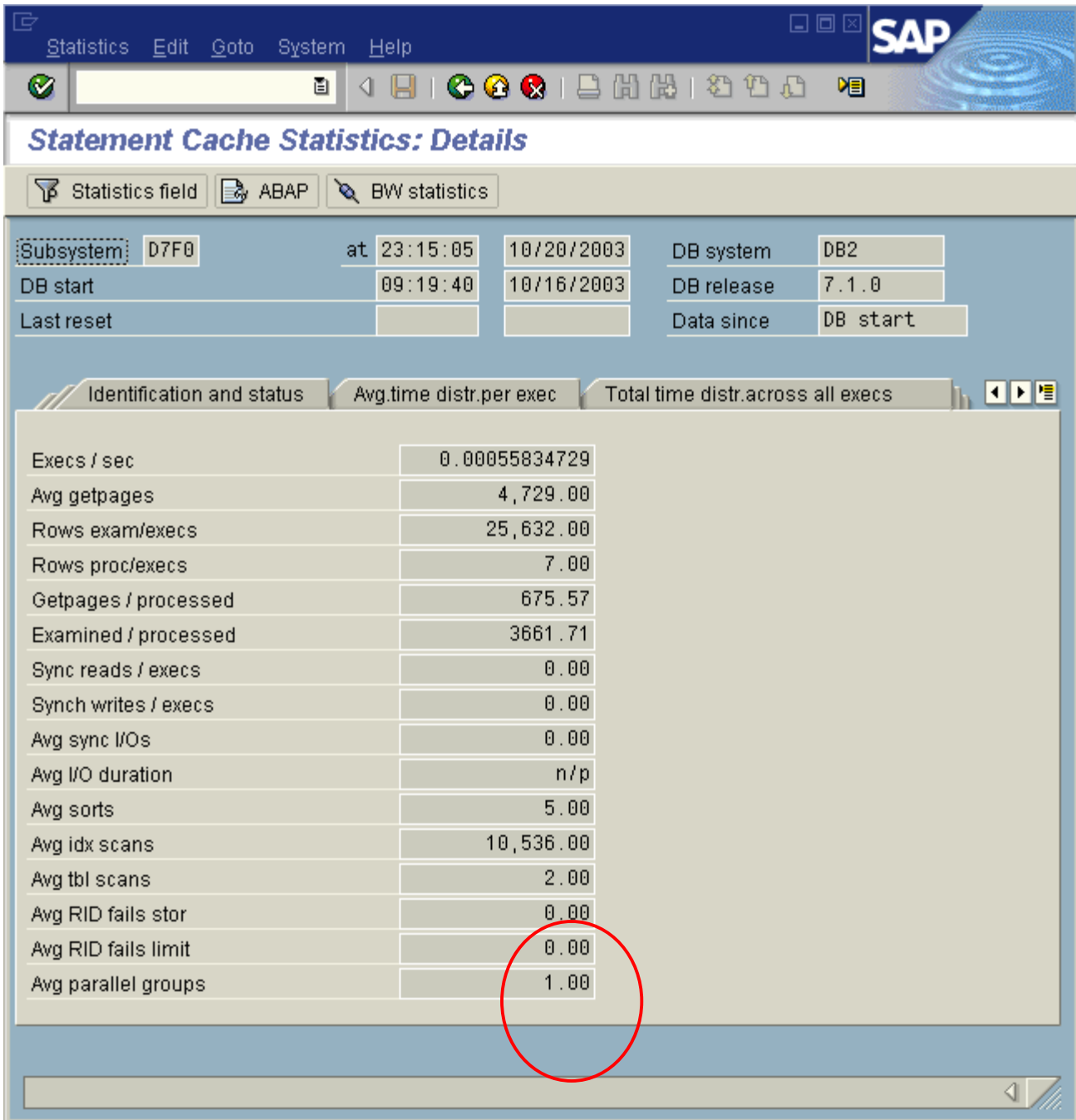


Figure 7: ST04 statement statistics - parallelism used

Even with incomplete statistics, it is often clear what the problem is, after examining the available statistics and access path. If necessary, one can use the following workaround to get complete statistics:

- Restart a work process, by using SM50 to terminate the WP. This will reset the thread statistics.
- Lock the userid that will run the test query in the restarted work process, using RSTRC000.
- Execute the BW query using RSRT, the SAPGUI interface to BW query execution.
- Restart the work process, to create statistics records over the test for parent and query children.

- Use DB2PM RECTRACE IFCID 3 to display accounting data for parent and query children. The child data is aggregated in a record marked “ROLLUP DATA FOR PARALLEL CHILD TASKS YES”.

6.7. Query execution using views

With some versions and support package levels in BW, queries are executed using VIEWS that are dropped immediately after the query SQL finishes. Once the VIEW is dropped, one cannot use ST04 to explain the statement and check the access path. SAP note 373738 describes how to retain the views after the SQL finishes. When VIEWS are defined, the ST04 cache will look something like this:

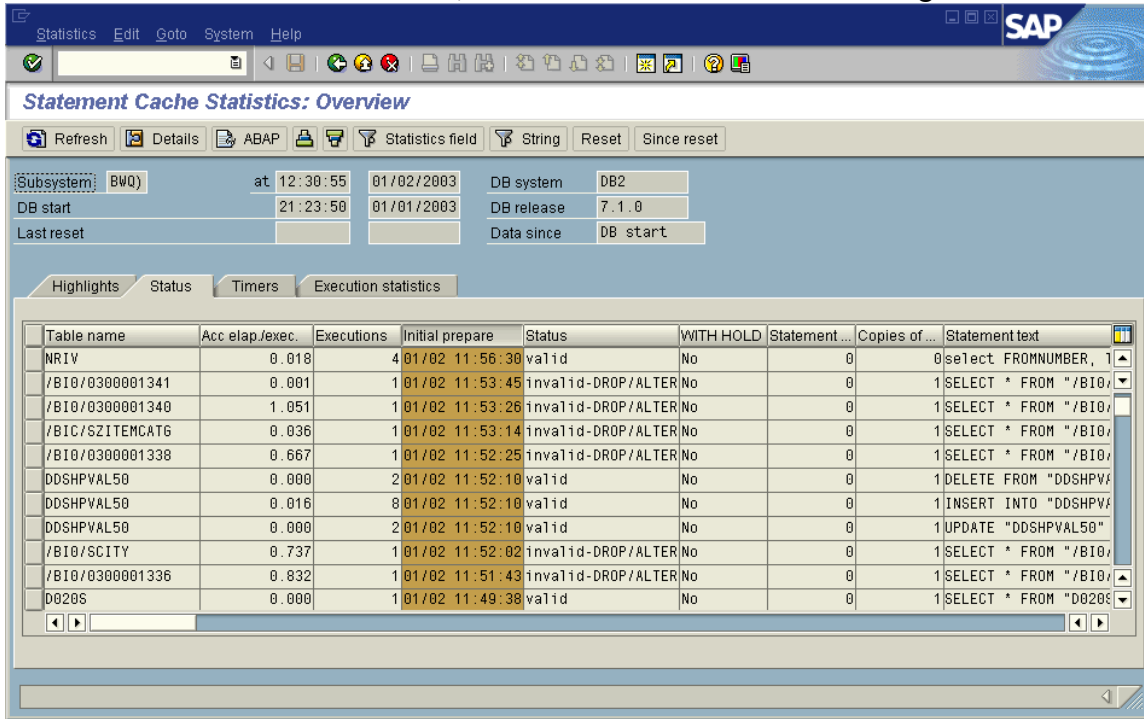


Figure 8: ST04 statements invalid due to dropped VIEW

The statements will look like this:

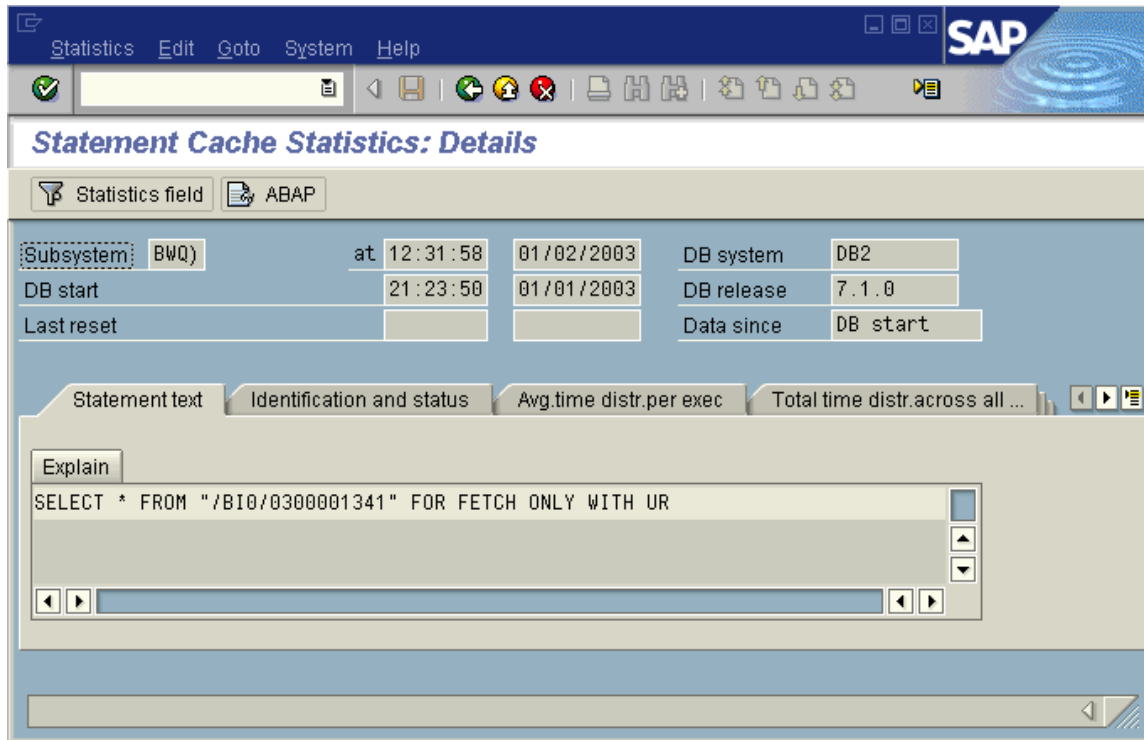


Figure 9: Statement using view to access infocube or aggregate

Review SAP note 373738 for information on how to retain the VIEWS after execution.

6.8. Star Join

Star join is an access method optimized for data stored in star schema. SAP infocubes use star schema. Star join is enabled by two DB2 parameters. **See SAP note 390016 for all current recommendations.** The information here is only current at the time this paper was written.

- STARJOIN, which enables star join, and checks whether the query is suitable for star join based on the relative size of the dimension tables and fact table, and
- SJTABLES, which sets limits on when star join should be chosen by the optimizer, based on the number of tables in the star join query block.

MXTBJOIN (current SAP BW recommendation 100) also should be set for SAP BW systems. If this parameter is set, and star join is not chosen, then DB2 can still optimize star schema joins with many tables.

The current SAP recommendations (STARJOIN=2 and SJTABLES=4) mean

- Star join is enabled if the cardinality of the fact table is 2 times larger than the cardinality of the largest dimension table used in the query, and
- Star join is enabled for queries with 4 or more tables in the star join queryblock.

Note that infocube design problems, such as a dimension table that is very large, may prevent DB2 from using star join, because of the cardinality checks in the STARJOIN parameter.

Star join can be broken down into two phases:

- Outside-in, where filtering predicates are used to select candidate rows from the fact table and build an intermediate result set, and
- Inside-out, where the remaining (usually less-filtering and GROUP BY) predicates are joined to the intermediate result set to build the final result.

During the outside-in phase, DB2 builds a 'cartesian product' of the DIMIDs from the predicate characteristics used to filter the rows.

For example, if the query selects all the rows 'WHERE PRODUCT_CATEGORY=TV AND CUSTOMER='SUPER SALES' and PERIOD='001/2002' against an infocube with three dimensions (product, customer, and time), when a multicolumn index (customer dimension, product dimension, time dimension) exists, DB2 can use star join in the following way:

- Get all the DIMIDs (fact table key values) from the customer dimension for CUSTOMER='SUPER SALES'.
- Get all the DIMIDs from the product dimension for PRODUCT_CATEGORY=TV.
- Get all the DIMIDs from the time dimension for PERIOD='001/2002'.
- Build the cartesian product (a set of rows) of all these DIMIDs. Each row in the set is a key value (customer, product, time) that will be sought in the fact table.
- Fetch rows from the fact table using the index (customer, product, time).

The cartesian product is all the ordered combinations of the DIMIDS satisfying the predicates in each of the three dimensions. If the DIMIDs for customer dimension are 2,5 and the DIMIDs for product dimension are 1,3,5, and the DIMIDs for period dimension are 1,4, then the cartesian product rows for the key (customer, product, period) would be

(2,1,1),
(2,1,4),
(2,3,1),
(2,3,4),
(2,5,1),
(2,5,4),
(5,1,1),
etc.

The Cartesian product is built in DB2 virtual storage, not materialized to disk.

If the number of DIMIDs meeting the predicate conditions in each dimension is X, Y, and Z, then the size of the Cartesian product will be $X*Y*Z$ rows -- the Cartesian product grows very quickly with the number of DIMIDs meeting the predicate conditions and the number of dimensions. Star join is chosen by DB2 when there are filtering predicates, and the Cartesian product will be a manageable size.

Multi-column fact table indexes that support filtering predicates on multiple dimensions are necessary for efficient exploitation of STAR JOIN. The default indexes that SAP creates on the fact tables are not well suited to STAR JOIN. See section 11.4 for more information on when to create additional fact table indexes.

The key benefit of star join, compared to other DB2 access methods, is that by applying more filtering predicates to the ‘outside in’ phase, there will be fewer rows fetched from the fact table. By narrowing the result set early, the query can run faster, and use less DB2 SORT memory, less DB2 WORKFILE space, and less CPU.

There are a number of PTFs that are very important for STAR JOIN performance. SAP note 81737 lists the key PTFs for SAP. There are performance enhancements in DB2 V7 (such as sparse indexes on work files) which can offer better star join performance with DB2 V7 than with DB2 V6.

SAP note 390016 describes configuration settings specific to BW on DB2/390 systems.

7. Strategy Roadmap

7.1. *Individual Query Optimization*

The key points in optimizing performance of an individual BW query are:

- For an individual query, use RSDDSTAT to determine the components of query elapsed time - Section 9.2. (Summarized query statistics are available in ST03N or via queries on 0BWTC_C02)
- Evaluate whether an aggregate should be defined to support query – Section 9.2.2 and 11.2.1.
- If an aggregate is not suitable for the query:
 - If DB time is the main component of elapsed time, find and explain the SQL statement – Section 10.1.
 - Check statement predicates, and estimate which predicates filter the result best – Section 11.4.7.1.
 - Evaluate dimension join order, and determine if filtering dimensions are joined early – Section 11.4.7.
 - If filtering dimensions are not joined early, check whether indexes for the predicates exist on the dimension, SID (master), or attribute SID (navigation attribute) tables – Section 11.4.4.
 - If indexes already exist and data in the indexed columns is skewed, try creating FREQVAL statistics with a higher COUNT, to gather more information on distribution – Section 11.4.3.
 - If none of the above help, evaluate whether multi-column index on fact table is needed to enable star join – Section 11.4.7.

7.2. *System-wide optimization*

At a system-wide level, the most important activity is evaluation and creation of aggregates.

- Check that aggregates exist and are used for frequently executed queries – Section 9.2.2, Section 11.1.1.
- After aggregates have been defined:
 - Check DB2 for symptoms of buffer pool size or threshold problems - Section 11.4.9.
 - Check DB2 (or OS/390 or z/OS) for symptoms of OS level constraints – Section 11.4.7.2.

8. Correlating ST04 statement cache and SAP statistics

When examining SQL for slow queries in the statement cache, it is helpful to check the corresponding SAP statistics, which include information about SAP’s logical view of the query (characteristics and key figures) as well as the name of the query, infocube, and performance statistics such as QDBSEL (rows matching predicates) that are not available from DB2.

8.1. Enhanced ST04 with query correlation

With BW 3.0B Support Package 15, there is an enhanced version of explain that links the SQL in ST04 to RSDDSTAT and RSDDSTATAGGRDEF, which contain SAP statistics.

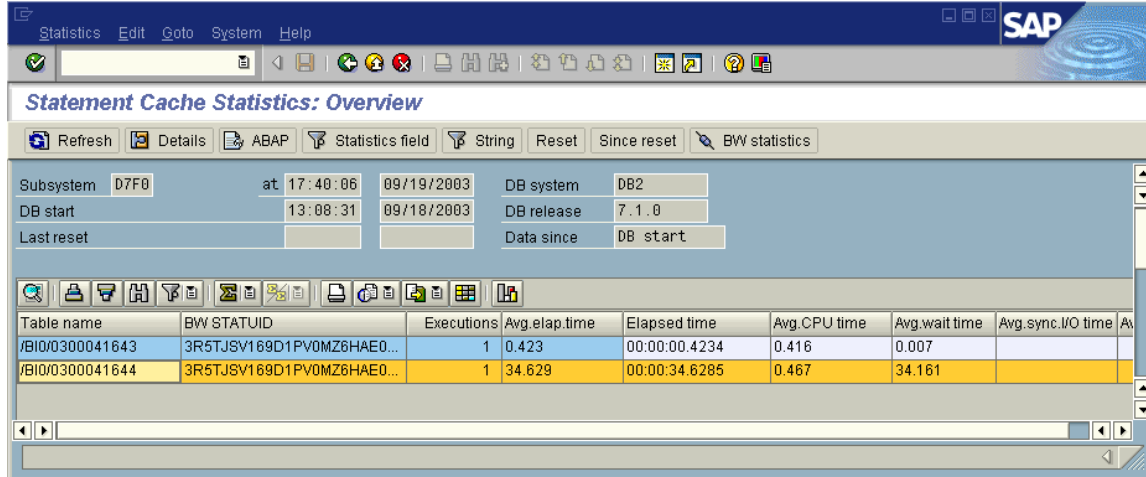


Figure 10: ST04 statement cache with BW query correlation

By picking a statement, and pressing the “BW statistics” button shown in Figure 10, one can display the statistics used for the key performance indicators described in Section 9:

- DB2 rows examined (called analyzed rows in Figure 11)
- QDBSEL (selected rows – rows matching the predicates)
- QDBTRANS (transported rows – rows after GROUP BY)
- QTIMEDB (query DB time)

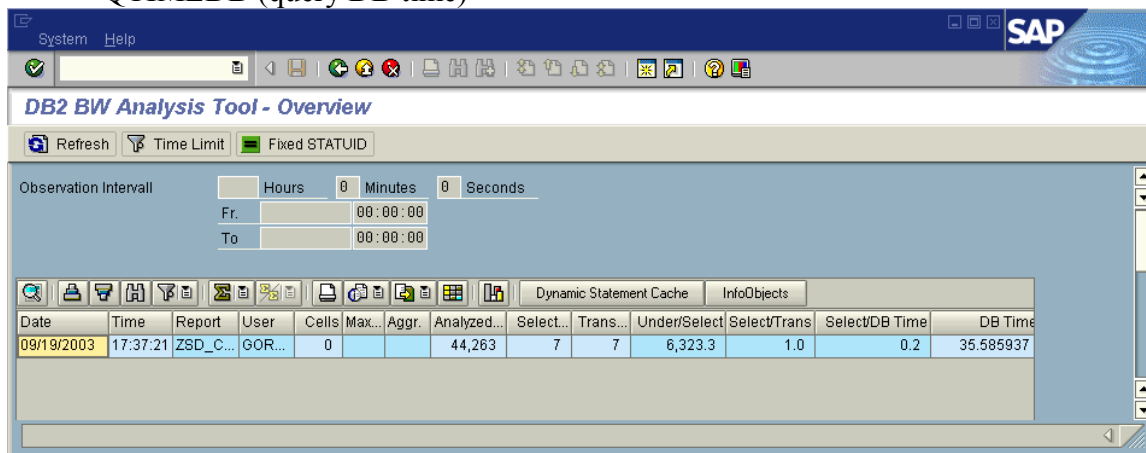
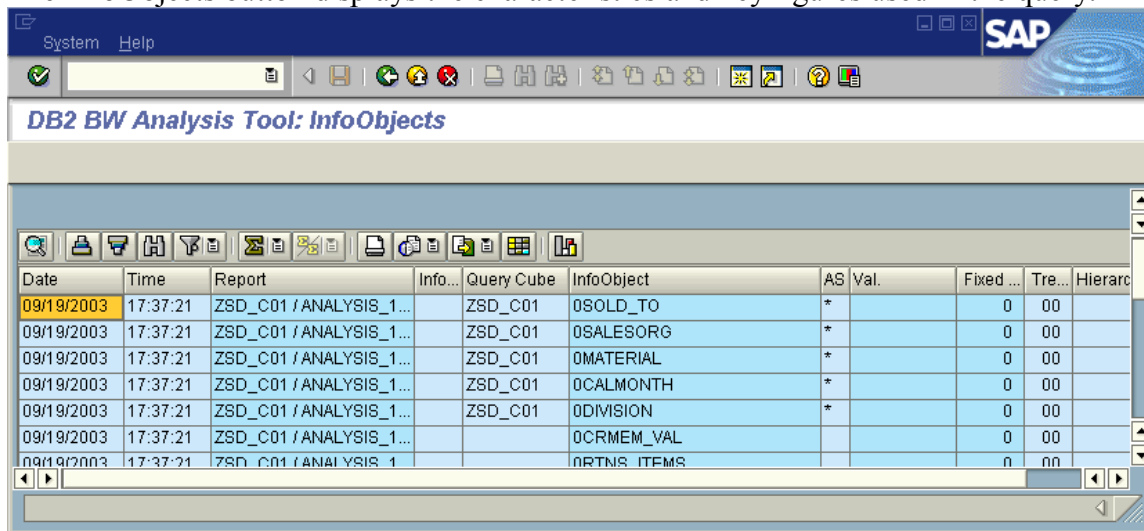


Figure 11: ST04 BW analysis

The InfoObjects button displays the characteristics and key figures used in the query.



Date	Time	Report	Info...	Query Cube	InfoObject	AS	Val.	Fixed ...	Tre...	Hierarc
09/19/2003	17:37:21	ZSD_C01 / ANALYSIS_1...		ZSD_C01	0SOLD_TO	*		0	00	
09/19/2003	17:37:21	ZSD_C01 / ANALYSIS_1...		ZSD_C01	0SALESORG	*		0	00	
09/19/2003	17:37:21	ZSD_C01 / ANALYSIS_1...		ZSD_C01	0MATERIAL	*		0	00	
09/19/2003	17:37:21	ZSD_C01 / ANALYSIS_1...		ZSD_C01	0CALMONTH	*		0	00	
09/19/2003	17:37:21	ZSD_C01 / ANALYSIS_1...		ZSD_C01	0DIVISION	*		0	00	
09/19/2003	17:37:21	ZSD_C01 / ANALYSIS_1...			0CRMEM_VAL			0	00	
09/19/2003	17:37:21	ZSD_C01 / ANALYSIS_1...			0RTNS_ITEMS			0	00	

Figure 12: ST04 statement cache BW InfoObjects

8.2. Manually correlating ST04 cache statistics and RSDDSTAT

For those using BW versions before 3.0B SP 15, correlate query statistics by comparing the query timestamp in RSDDSTAT, which is GMT, and the timestamp on the statement in ST04, which is DB2 time. See the example in Section 12 for manual correlation of statement cache and RSDDSTAT.

9. Indicators of performance problems for DB2 running BW

Since SAP BW queries often do GROUP BY summarizations, the indicators that can show performance problems in R/3 (e.g. getpages per row processed, or time per row processed) are often not useful with BW, since the ST04 count of 'rows processed' is reduced by the GROUP BY summarization. The GROUP BY summarization affects these ratios, and makes them appear worse than they actually are.

BW has different categories of query response time than R/3. These response time categories are described in SAPnote 130696. Statistics for individual DB2 queries (SAP navigation steps) are in the table RSDDSTAT, or statistics cube 0BWTC_C02. The ratio of QTIMEDB to other categories of SAP time in RSDDSTAT is the key indicator of slow database time for queries. If the query is long, and QTIMEDB is the majority of time, then reducing the time to retrieve the data from the database will help improve performance.

RSDDSTAT also contains counters for the number of rows processed by the query

- QDBSEL, which is the number of rows that satisfy the predicates (it is COUNT(*) in SQL), and
- QDBTRANS, which is the number of rows after GROUP BY. This is also the same as ST04 "rows processed" in DB2 statement cache statistics.

9.1. Key ratios that can indicate performance problems

For long running queries, there are three ratios that can be used as indicators of problems:

- **The ratio (QDBSEL / QTIMEDB)** is the number of rows selected from the database per second. If this is high (1000 or more) then database performance is good, though the query could still be slow because an aggregate is needed (see QDBSEL/QDBTRANS). If this ratio is very low (50 or less) then there is usually a problem with DB access. Rates ranging between these two can be normal, for instance when a query joins many dimensions to a fact table, or when many dimensions are joined to master data or attribute SID tables. Many different things can cause slow QDBSEL/QTIMEDB rates: I/O constraints, missing indexes, incorrect access path selection, buffer pool constraints, or a data model problem.
- **The ratio (ST04 statement cache “rows examined” / QDBSEL)** is the number of rows examined for each row that satisfied the predicates. If this is high, it can be a sign of access path problems caused by predicate columns without indexes, or that DB2 is not able to use a filtering predicate early in the query to reduce the number of candidate rows. When a query is run with indexes that filter the result early and effectively, this ratio may be as low as 5 or 10. *By itself, this ratio is not a reliable indicator of performance problems.* For example, if a query contains a small table without a good index, the table may be repeatedly scanned and this ratio will be high, but the query may still run very quickly, since scanning a few dozen rows of a table in memory is fast. If you find a query which is slow, and where this ratio is high (e.g. 100-500, or more), then use ST04 EXPLAIN to check the statement’s access path and check that indexes exist for the predicates, as is shown in section 11.4.4.
- **The ratio (QDBSEL / QDBTRANS)** is the number of rows selected for each row in the result after GROUP BY. When this is high, it means that DB2 must summarize many selected rows into each result row, so an SAP aggregate table would probably help performance, since the aggregate table would contain pre-summarized data. After the fact table is summarized into an aggregate table, the query will retrieve fewer rows to create the result. SAP’s rule of thumb is that when this ratio is greater than 10, an aggregate may be appropriate. Use the frequency of query execution, the importance of performance for the query, and the time available for aggregate roll-ups in conjunction with this ROT when evaluating the need for new aggregate tables. For instance, if an aggregate is not frequently used, and if the query is very slow, but there is time and CPU available in the aggregate roll-up window, then creating an aggregate is a way to move part of the report generation time to the nighttime.

If the query runs quickly, then these ratios are not important – e.g. if QTIMEDB is just a few seconds, then it is not important if QDBSEL/QTIMEDB is low.

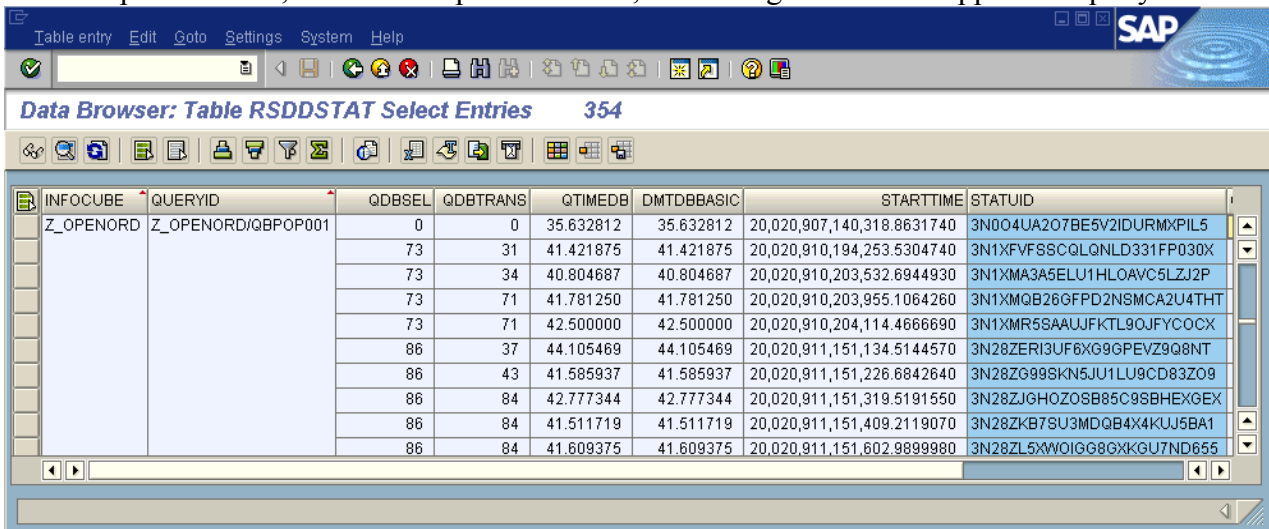
9.2. Interpreting RSDDSTAT

RSDDSTAT is the SAP table that contains information on SAP query navigation step (that is DB2 query) performance. Not all navigation steps make calls to the database server. If the database server is called, RSDDSTAT QTIMEDB will be >0. For database performance, QTIMEDB, QDBSEL, and QDBTRANS are the most important statistics. See SAP note 130696 or use F1 help in SE16 for information on the fields in this table. The same time categories are used in the 0BWTC_C02 statistics cube, when BW statistics are activated.

Not all the RSDDSTAT columns are displayed in the examples.

9.2.1. Possible z/OS or DB2 performance problem

In cases where QDBSEL is very close to QDBTRANS, and the query is slow, there may be a problem with I/O performance, DB2 access path selection, or missing indexes to support the query.



INFOCUBE	QUERYID	QDBSEL	QDBTRANS	QTIMEDB	DMTDBBASIC	STARTTIME	STATUID
Z_OPENORD	Z_OPENORD/QBPOP001	0	0	35.632812	35.632812	20,020,907,140,318.8631740	3N004UA207BE5V2IDURMXPIL5
		73	31	41.421875	41.421875	20,020,910,194,253.5304740	3N1XFVF8SCQLQNLD331FP030X
		73	34	40.804687	40.804687	20,020,910,203,532.6944930	3N1XMA3A5ELU1HLOAVC5LZJ2P
		73	71	41.781250	41.781250	20,020,910,203,955.1064260	3N1XMQB26GFPD2NSMCA2U4THT
		73	71	42.500000	42.500000	20,020,910,204,114.4666690	3N1XMR5SAAUJFKTL9QJFYCOOX
		86	37	44.105469	44.105469	20,020,911,151,134.5144570	3N28ZERI3UF6XG9GPEVZ9Q8NT
		86	43	41.585937	41.585937	20,020,911,151,226.6842640	3N28ZG99SKN5JU1LU9CD83ZO9
		86	84	42.777344	42.777344	20,020,911,151,319.5191550	3N28ZJGHOZOSB85C9SBHEXGEX
		86	84	41.511719	41.511719	20,020,911,151,409.2119070	3N28ZKB78U3MDQB4X4KUJ5BA1
		86	84	41.609375	41.609375	20,020,911,151,602.9899980	3N28ZL5XWOIGG8GXKGU7ND655

Figure 13: RSDDSTAT slow QDBSEL/second with QDBSEL similar to QDBTRANS

In Figure 13, note that the query times are somewhat long (QTIMEDB over 30 seconds) but QDBSEL and QDBTRANS are under 100. The query retrieves only about 3 rows per second (QDBSEL/QTIMEDB), which is very slow. This points to DB problems such as I/O contention, access path or indexing. Section 10.2 shows how to find and examine the statement in the ST04 statement cache.

9.2.2. Need for new aggregate

In cases where QDBSEL is much larger than QDBTRANS, then creating a new aggregate table to pre-summarize the data can help improve performance.

STATUID	INFOCUBE	QUERYID	UNAME	QAGGRUSED	QDBSEL	QDBTRA...	QTIMEDB	STAF
888L41TSCA0SQ...	SDEUPCBIL	SDEUPCBIL/MSDEUPCBIL_PUB_700			0	0	646.015625	20,021,028,151,145.591
713E0FR6RRF3J...					3,140	37	33.843750	20,021,028,094,126.079
C2JLZWXJ366Q...					3,140	37	35.875000	20,021,028,095,842.124
599VWRYTGN6U...					11,228	107	51.218750	20,021,028,141,729.704
CXGB0YFAIRA32...					11,228	107	34.890625	20,021,028,134,541.231
D7TM34YNH3C2...					11,228	107	34.640625	20,021,028,164,734.030
66WZH30BR4XI...					148,218	906	190.265625	20,021,028,143,825.530
ES3E4EOYZL4H...					148,218	906	143.515625	20,021,028,150,855.451
2SQA9AHV9D65...					272,574	638	127.078125	20,021,028,161,007.841
6IU1MZ06O5Q4...					272,574	638	189.578125	20,021,028,105,605.376
29V3NKCSB1Q0...					2,891,956	4,806	1,553.171875	20,021,028,135,433.291
A7NT31BNTCFG...					2,891,956	4,806	1,060.718750	20,021,028,170,827.651
B80BTKQ0KZZ3I...					2,891,956	4,806	1,426.093750	20,021,028,100,554.821

Figure 14: RSDDSTAT QDBSEL much larger than QDBTRANS

In many of the lines in Figure 14, QDBSEL/QDBTRANS is 100 or more. An aggregate will help performance of this query.

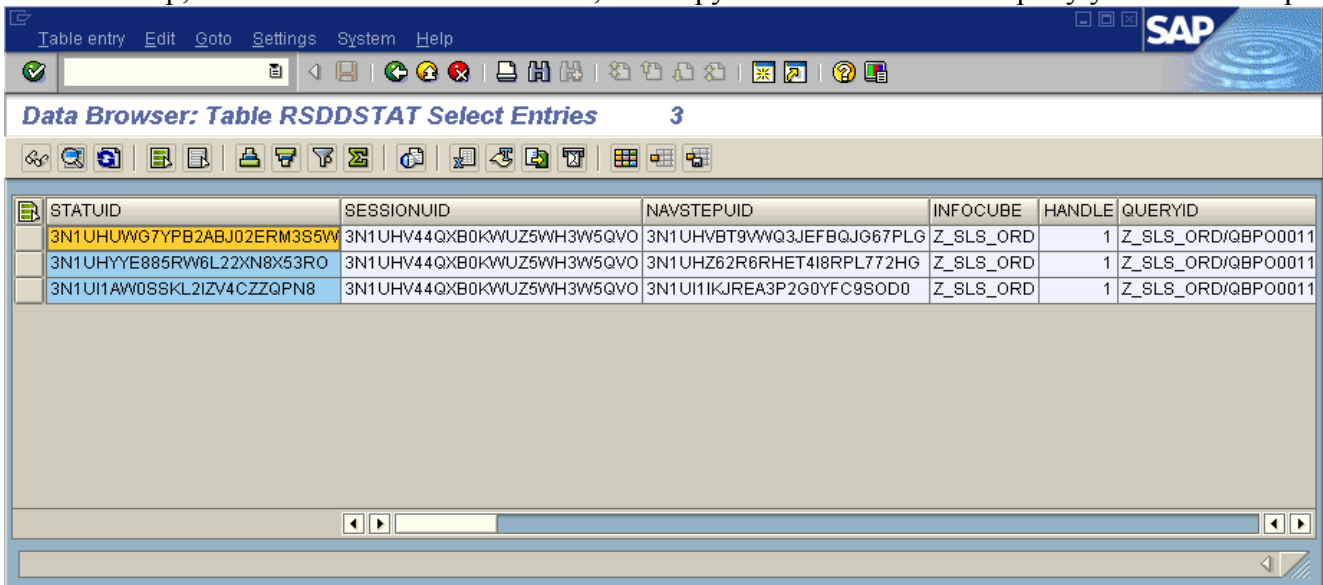
9.3. Interpreting RSDDSTATAGGRDEF

The table RSDDSTATAGGRDEF contains information about the characteristics used in BW queries. SAP RSA1 can use this information to propose aggregates. If the ST04 statement cache statistics for a query are not available, you can use this to check the characteristics used in the query.

9.3.1. Basic infocube

A basic infocube is a single infocube, which may be made up of both E and F fact tables, along with their dimension tables and related master data tables.

As a first step, use SE16 to view RSDDSTAT, and copy the STATUID of the query you want to display.



STATUID	SESSIONUID	NAVSTEPUID	INFOCUBE	HANDLE	QUERYID
3N1UHUWG7YPB2ABJ02ERM3S5W	3N1UHV44QXB0KWUZ5WH3W5QVO	3N1UHVBTV9VWQ3JEFBQJG67PLG	Z_SLS_ORD	1	Z_SLS_ORD/QBP00011
3N1UHYYE885RW6L22XN8X53RO	3N1UHV44QXB0KWUZ5WH3W5QVO	3N1UHZ62R6RHET4I8RPL772HG	Z_SLS_ORD	1	Z_SLS_ORD/QBP00011
3N1UI1AW0SSKL2IZV4CZZQPN8	3N1UHV44QXB0KWUZ5WH3W5QVO	3N1UI1IKJREA3P2G0YFC9S0D0	Z_SLS_ORD	1	Z_SLS_ORD/QBP00011

Figure 15: RSDDSTAT to get STATUID

Next, use SE16 to view RSDDSTATAGGRDEF, and paste the STATUID from RSDDSTAT into the STATUID for RSDDSTATAGGRDEF.

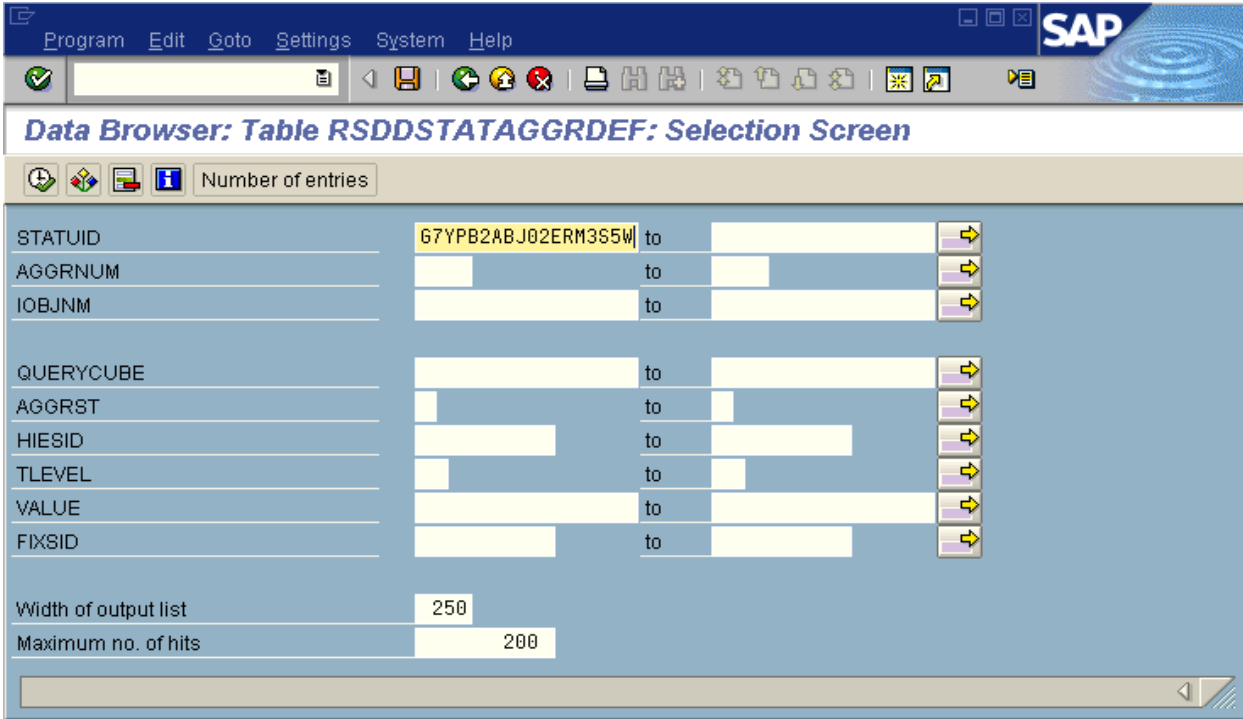


Figure 16: RSDDSTATAGGRDEF selection screen

Press execute.

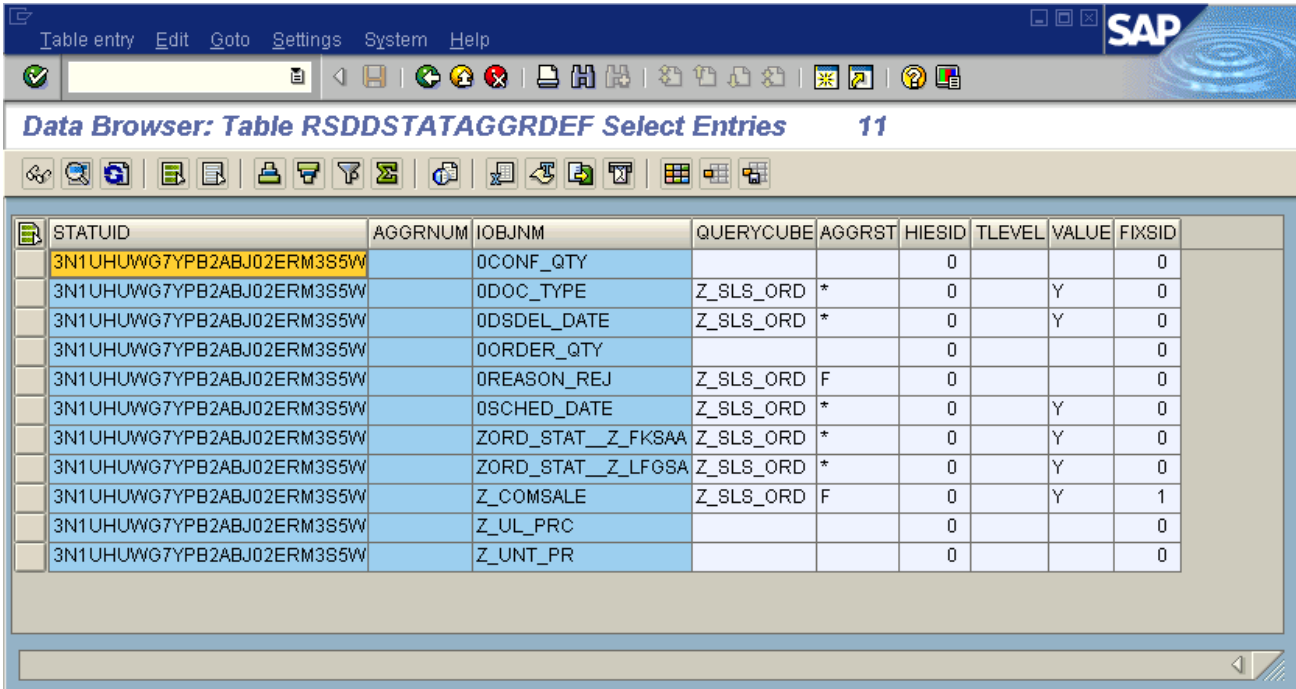


Figure 17: RSDDSTATAGGRDEF query information

In Figure 17, the report rows where AGGRST is not blank are characteristics and navigation attributes. They can be used for SELECT and GROUP BY. Rows where AGGRST is blank are key figures summed in the SELECT clause. The query user does not select the SAP characteristics for the P (packet ID) dimension. The P dimension characteristics are found in the SQL, but will not be in RSDDSTATAGGRDEF.

Since the data in RSDDSTATAGGRDEF can contain errors, if you need to review the statement, it is best to get the details such as predicates and values from the ST04 statement cache.

9.3.2. Multicube example

A multicube is a union of two (or more) infocubes. The infocubes must share some characteristics. These shared characteristics are used as the basis of the union of the different key figures in the infocubes.

When examining RSDDSTATAGGRDEF for a multicube query, there will be entries for each of the infocubes that make up the multicube. The ST04 statement cache will contain DB2 SQL statements executed on each infocube in the multicube, too. SAP combines the result of each of the DB2 SQL statements to build the result for the BW end-user.

In figure 6, the infocube is named SDEUDELOR. But when we display the query in RSDDSTATAGGRDEF (Figure 19), note that two infocubes, SDEUDELTR and SDEUPORD, are referenced.

STATUID	INFOCUBE	QUERYID	UNAME	QAGGR...	QDBSEL	QDBTRA...	QTIMEDB	STA
B3G2XAK6UOKF8TMKFG...	SDEUCSRC	SDEUCSRC/MSDEUCSRC_I_BW_Q500			19,360	2,433	38.906250	20,021,004,103,455.69
7GUOIXEQD88BP68D5YP...	SDEUDELOR				44,680	7,497	16.921875	20,021,004,115,304.13
0SFBBV39HQZVAH6BBW...		SDEUDELOR/MSDEUDELOR_I_BW_Q...			44,680	7,215	16.515625	20,021,004,114,800.36
59I4C5T0D4L72K9FPOMB...					37,453	34,818	417.906250	20,021,004,075,508.93
DTA18LSWJHDHIFVQK26...					42,330	6,259	22.421875	20,021,004,101,056.34
D8NXA25293Z9NTKFKR6...	SDEUDELTR	SDEUDELTR/ASDEUDELTR_I_FL_500			5,599	1,536	50.968750	20,021,004,113,959.72
2VQIKQ500RTURJACF1		SDEUDELTR/MSDEUDELTR_I_FLW_5...			4,050	885	20.421875	20,021,004,113,422.07

Figure 18: Multicube RSDDSTAT entry

Note that in Figure 18, there is a single entry for the multicube query navigation step (each has a unique STATUID), just as there would be for any query navigation step.

STATUID	AGGRNUM	IOBJNM	QUERYCUBE	AGGRST	HIESID	TLEVEL	VALUE	FIXSID
5914C5T0D4L72K9FPOMBF7GCI		0CALMONTH	SDEUDELTR	*	0		1799	0
5914C5T0D4L72K9FPOMBF7GCI		0DOC_NUMBER	SDEUDELTR	*	0		1799	0
5914C5T0D4L72K9FPOMBF7GCI		0ITEM_CATEG	SDEUDELTR	*	0		1799	0
5914C5T0D4L72K9FPOMBF7GCI		0MATERIAL	SDEUDELTR	*	0		1799	0
5914C5T0D4L72K9FPOMBF7GCI		0MATERIAL__EPRODH02	SDEUDELTR	*	0		1799	0
5914C5T0D4L72K9FPOMBF7GCI		0MATERIAL__EPRODH05	SDEUDELTR	*	0		1799	0
5914C5T0D4L72K9FPOMBF7GCI		0PLANT	SDEUDELTR	*	0		1799	0
5914C5T0D4L72K9FPOMBF7GCI		0SALESORG	SDEUDELTR	F	0		1799	14
5914C5T0D4L72K9FPOMBF7GCI		ECONCS			0			0
5914C5T0D4L72K9FPOMBF7GCI	1	0CALMONTH	SDEUPORD	*	0		1799	0
5914C5T0D4L72K9FPOMBF7GCI	1	0DOC_NUMBER	SDEUPORD	*	0		1799	0
5914C5T0D4L72K9FPOMBF7GCI	1	0DOC_TYPE	SDEUPORD	*	0		1799	0
5914C5T0D4L72K9FPOMBF7GCI	1	0ITEM_CATEG	SDEUPORD	*	0		1799	0
5914C5T0D4L72K9FPOMBF7GCI	1	0MATERIAL	SDEUPORD	*	0		1799	0
5914C5T0D4L72K9FPOMBF7GCI	1	0MATERIAL__EPRODH02	SDEUPORD	*	0		1799	0
5914C5T0D4L72K9FPOMBF7GCI	1	0MATERIAL__EPRODH05	SDEUPORD	*	0		1799	0
5914C5T0D4L72K9FPOMBF7GCI	1	0PLANT	SDEUPORD	*	0		1799	0
5914C5T0D4L72K9FPOMBF7GCI	1	0REASON_REJ	SDEUPORD	*	0		1799	0
5914C5T0D4L72K9FPOMBF7GCI	1	0SALESORG	SDEUPORD	F	0		1799	14
5914C5T0D4L72K9FPOMBF7GCI	1	EORDQTYCA			0			0

Figure 19: Multicube RSDDSTATAGGRDEF display

In Figure 19, RSDDSTATAGGRDEF shows the characteristics used for both the infocubes that make up the multicube. All the characteristics specified for SDEUDELTR are also specified for SDEUPORD. Different key figures are selected from the two infocubes. The VALUE column seems to contain replicated or invalid data in this example.

9.4. SAP query settings that affect performance

SAP queries have a setting (read mode) that determines how much data is retrieved when the query executes. If this is set wrong, it can greatly increase the response time of a query.

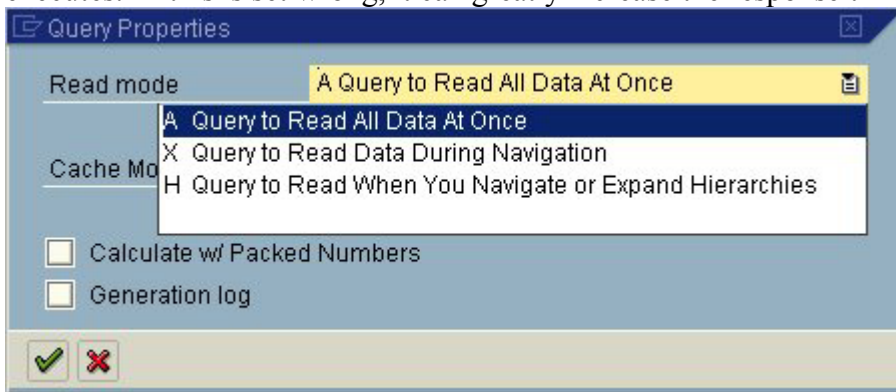


Figure 20: probably incorrect read mode setting

In Figure 20, the read mode “Query to read all data at once” has been chosen. When this option is chosen, the query will retrieve data to support drilldown into any free characteristics. This will increase the amount of data to be retrieved, and will increase query response time.

In general, SAP recommends the read mode “Query to read data when you navigate or expand hierarchies” should be set on queries. It offers the fastest response time, and retrieves the least extra data on each call.

The three read modes set in RSRREPDIR are:

- A – everything
- X – read during navigation
- H – read during navigation or expanding hierarchy

You can scan the reports defined in RSRREPDIR to see if there are queries with incorrect settings.

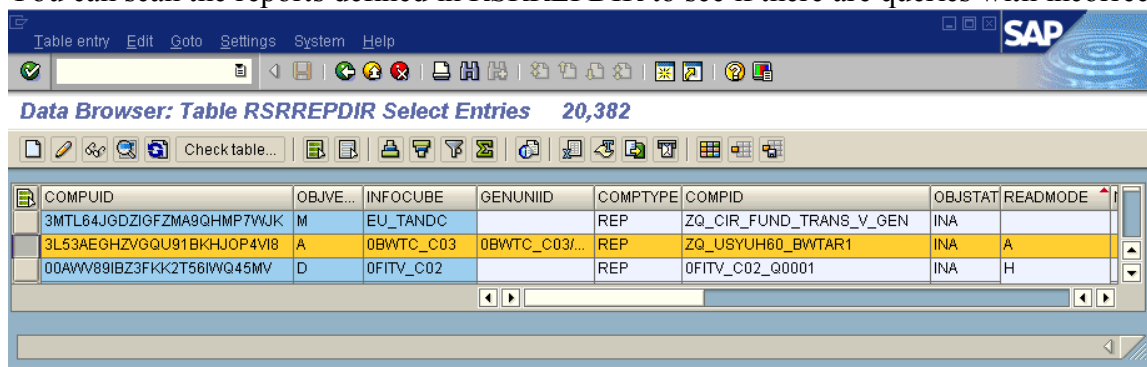


Figure 21: RSRREPDIR - search for queries with incorrect read mode

In addition to the query read setting, queries have buffering settings, which determine whether they are candidates for buffering on the application server in the OLAP cache. See section 11.4.1 for information on the OLAP cache.

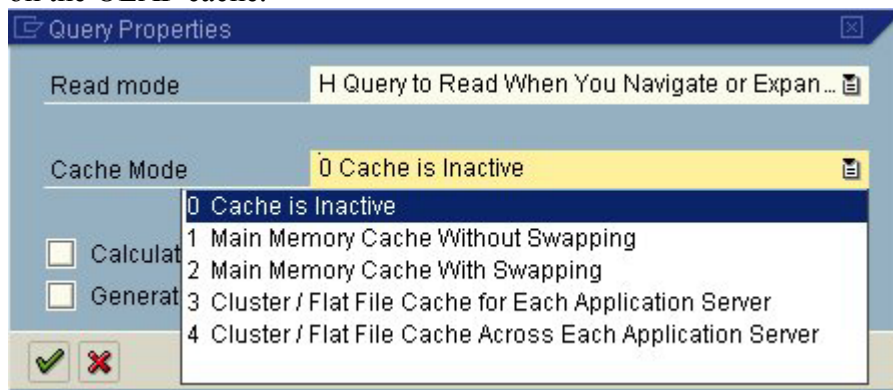


Figure 22: Query cache settings

10. Solving a performance problem for a specific query

When a user identifies a problem with a specific query, the SAP and DB2 performance statistics can be used together to determine where the problem lies, and how it might be solved.

10.1. Interactive testing with RSRT

The RSRT transaction can be used to run queries on infocubes or ODS.

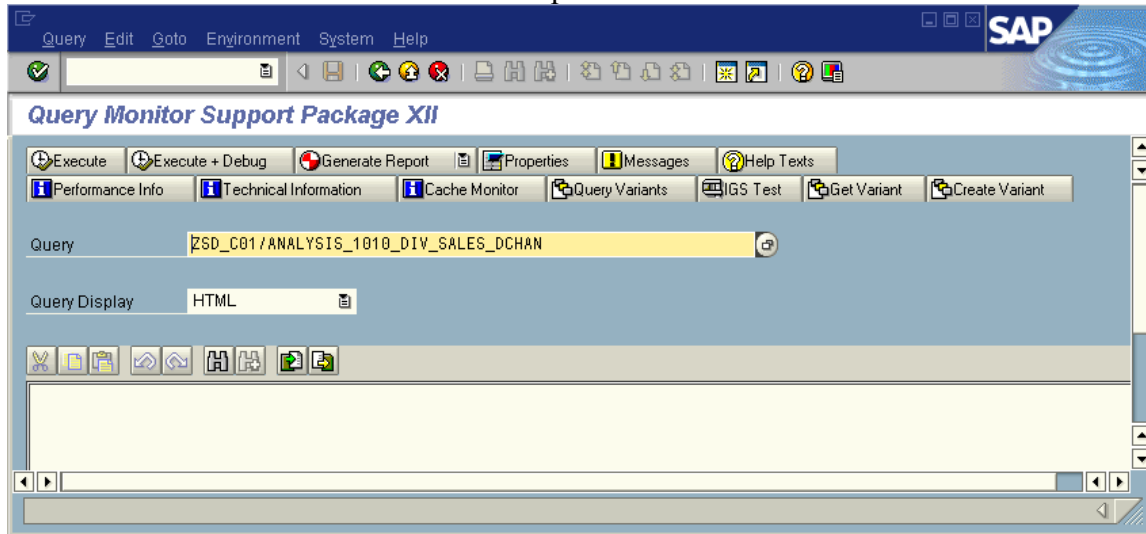


Figure 23: RSRT main screen

Press ‘Execute and Debug’ to display options that can be selected.

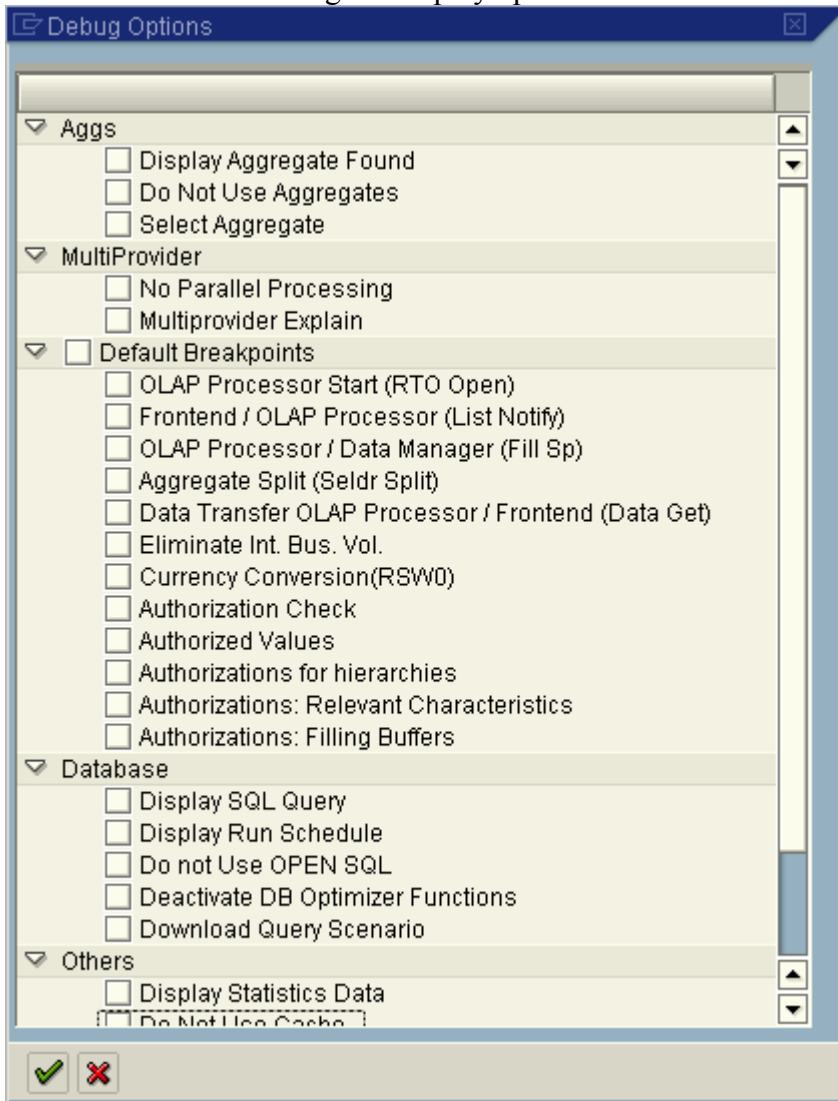
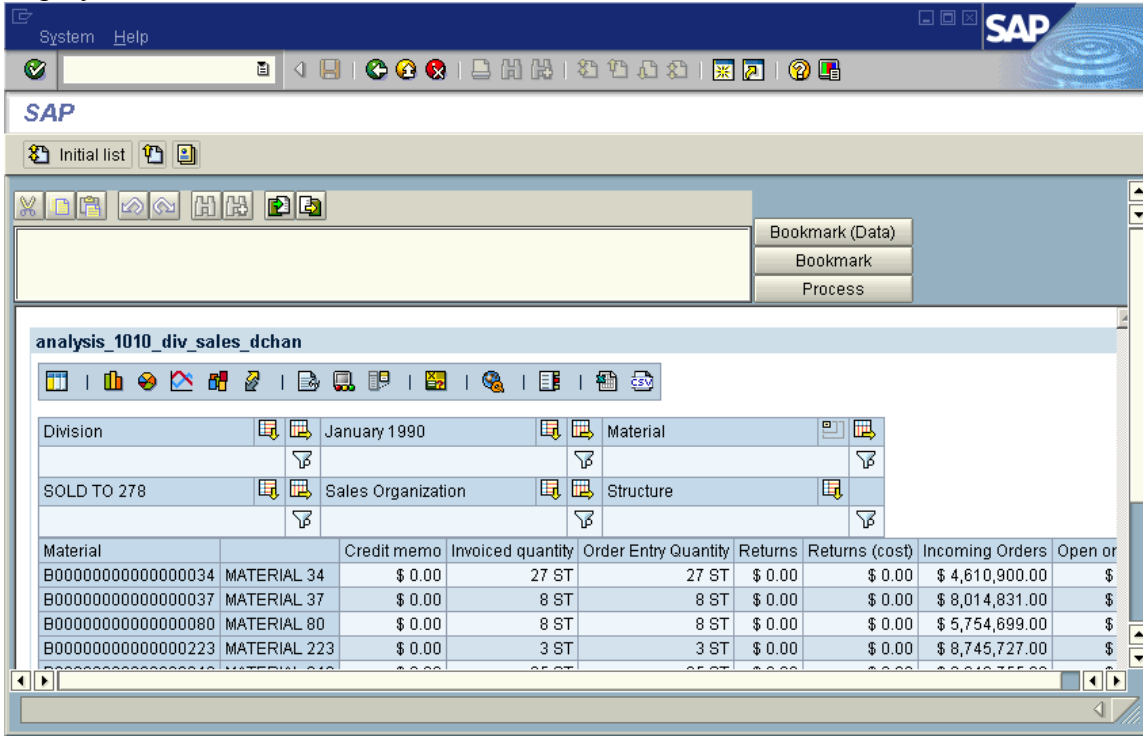


Figure 24: RSRT 'execute and debug' options

Some commonly used options are:

- “Display SQL query” to display the statement including literals before the statement is executed
- “Display Run Schedule” to explain the statement before the statement is executed
- “Display Statistics Data” to show the RSDDSTAT statistics with time breakdown
- “Do not use Cache” to bypass the OLAP cache and force the SQL to be executed in DB2
- “Do not use Aggregates” to bypass use of aggregates

In this example, we chose ‘display statistics data’ and ‘do not use cache’. The report result is first displayed.



Press the green back arrow, to display the RSDDSTAT statistics.

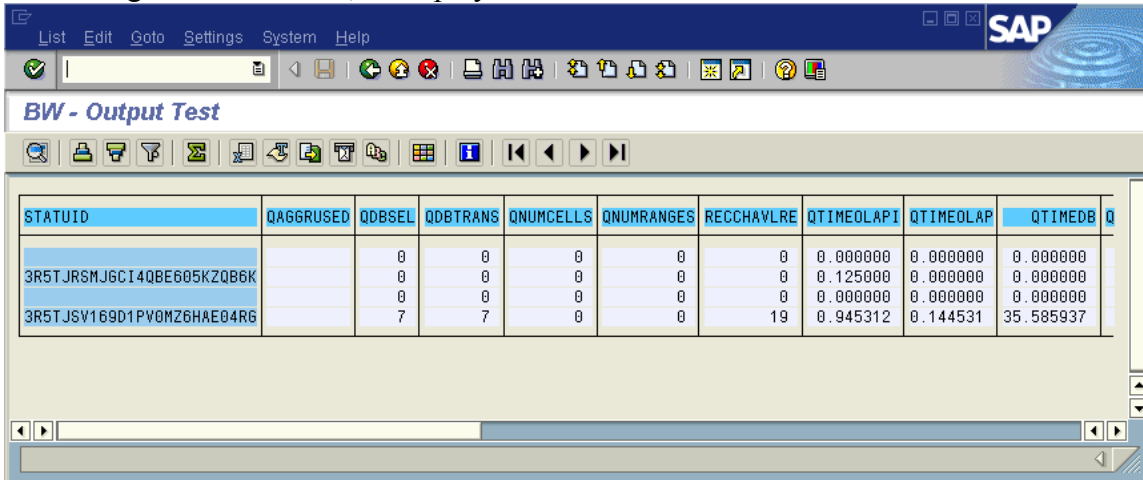


Figure 25: RSRT statistics

Note that *QTIMEDB* can contain time spent in the RSRT steps for explaining and viewing the SQL, so *QTIMEDB* and the elapsed time in ST04 statement cache may be different from each other.

Since the STATUID is listed, we can link to the SQL statement cache, and display the runtime statistics for this statement. Run ST04 > ‘cached statement statistics’, to get the selection popup.

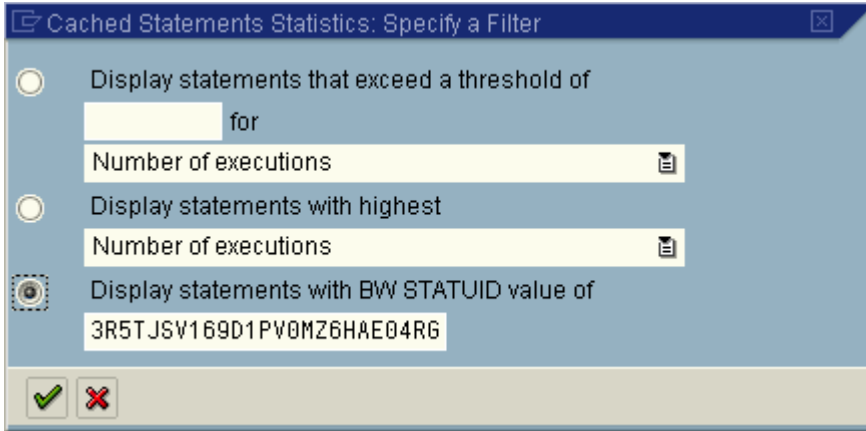


Figure 26: Statement cache – specify selection criteria

Enter the STATUID and press execute.

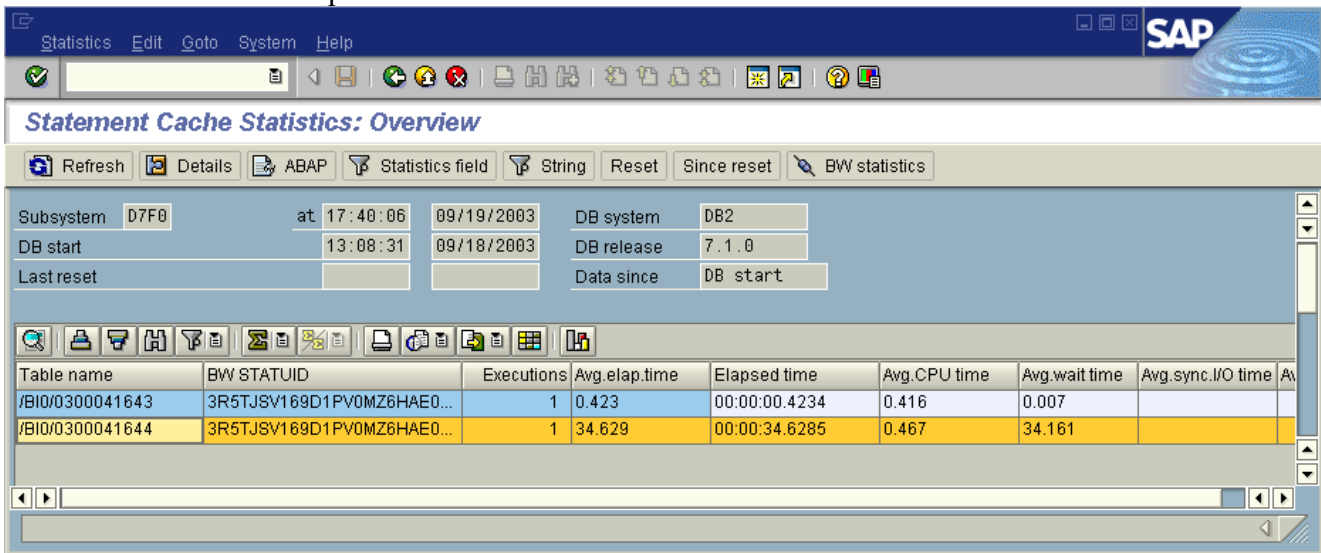


Figure 27: ST04 cache selected by STATUID

If we select a statement, and press ‘BW statistics’, the RSDDSTAT statistics are displayed.

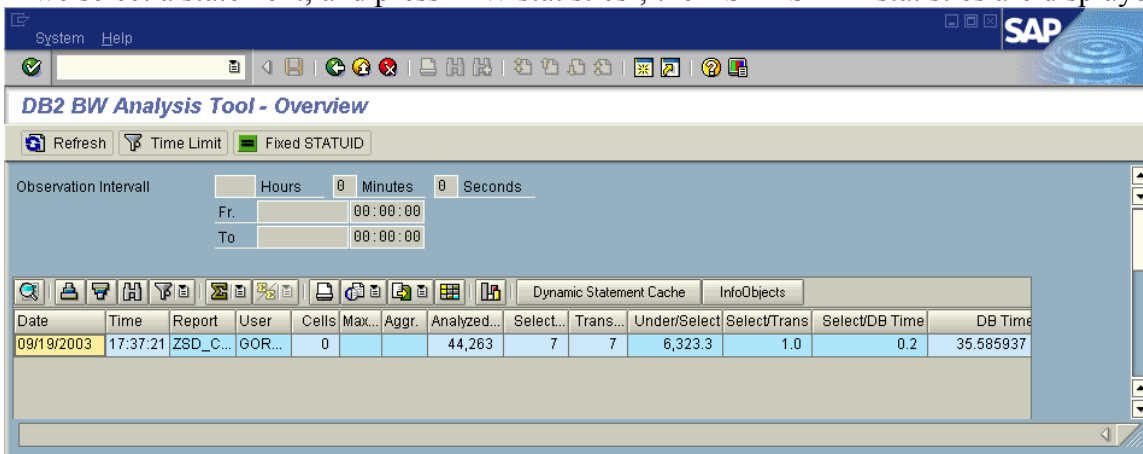


Figure 28: ST04 statement cache BW statistics

This query ran 35 seconds and examined 44,263 rows to find 7 rows (QDBSEL) that satisfied the predicates. This may be a problem with access path selection, such as might be caused by missing indexes or insufficient DB2 statistics. See section 11.4 for the list of actions when you want to try to change the way that DB2 accesses the infocube.

To check the access path, from the screen in Figure 27 select a statement and press 'Details'.

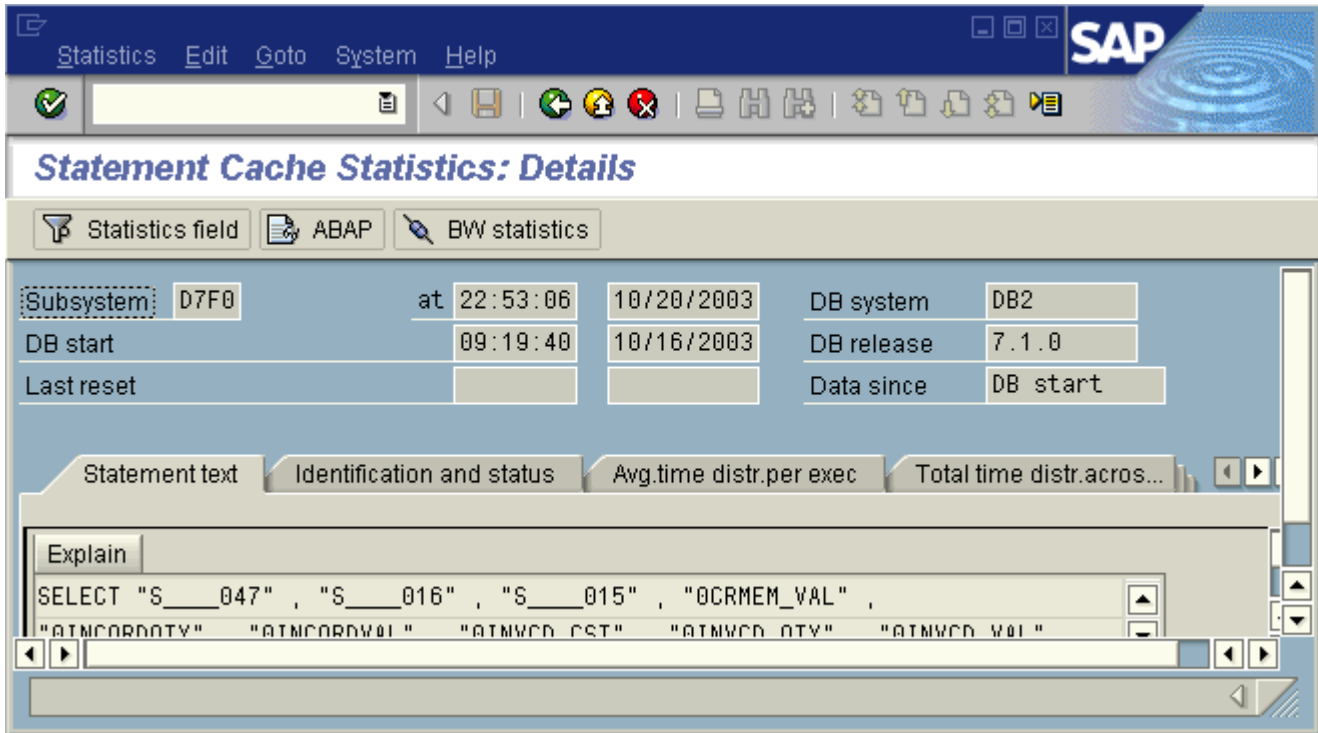


Figure 29: statement details

Then press explain, and answer YES to the ‘Turn on parallelism’ popup.

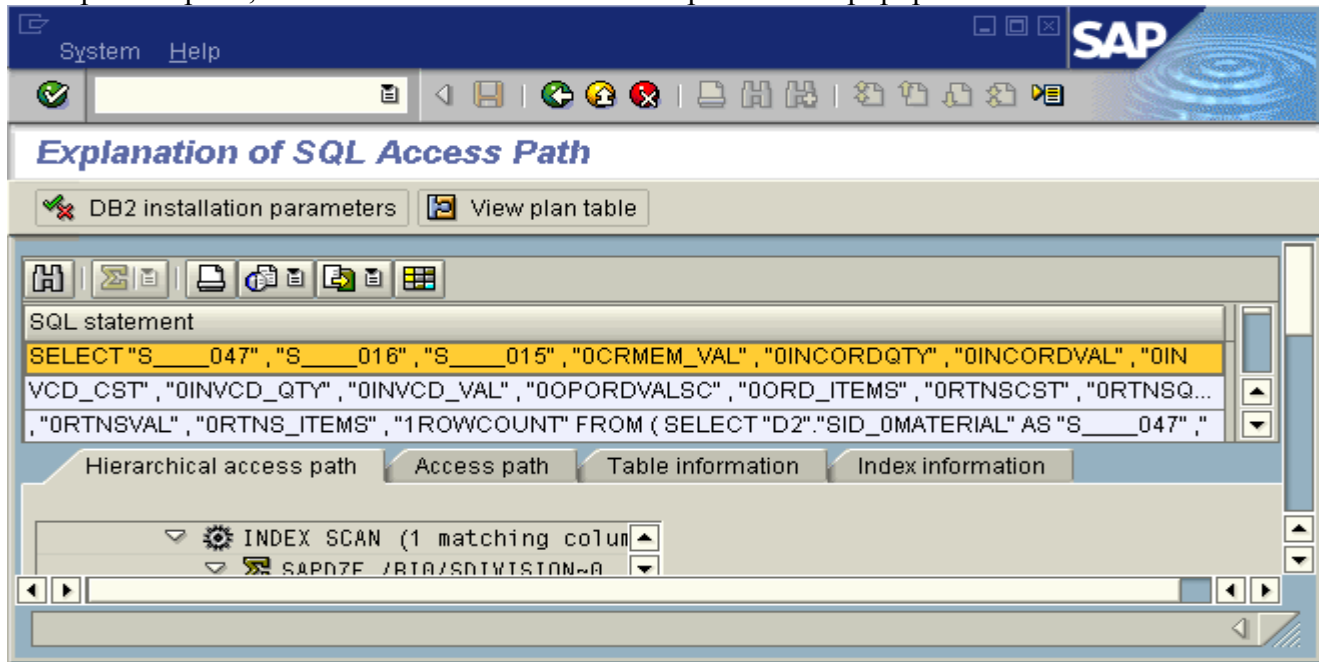


Figure 30: ST04 explain

Then press ‘View plan table’, and press ‘Yes’ at the “Turn on parallelism” popup.

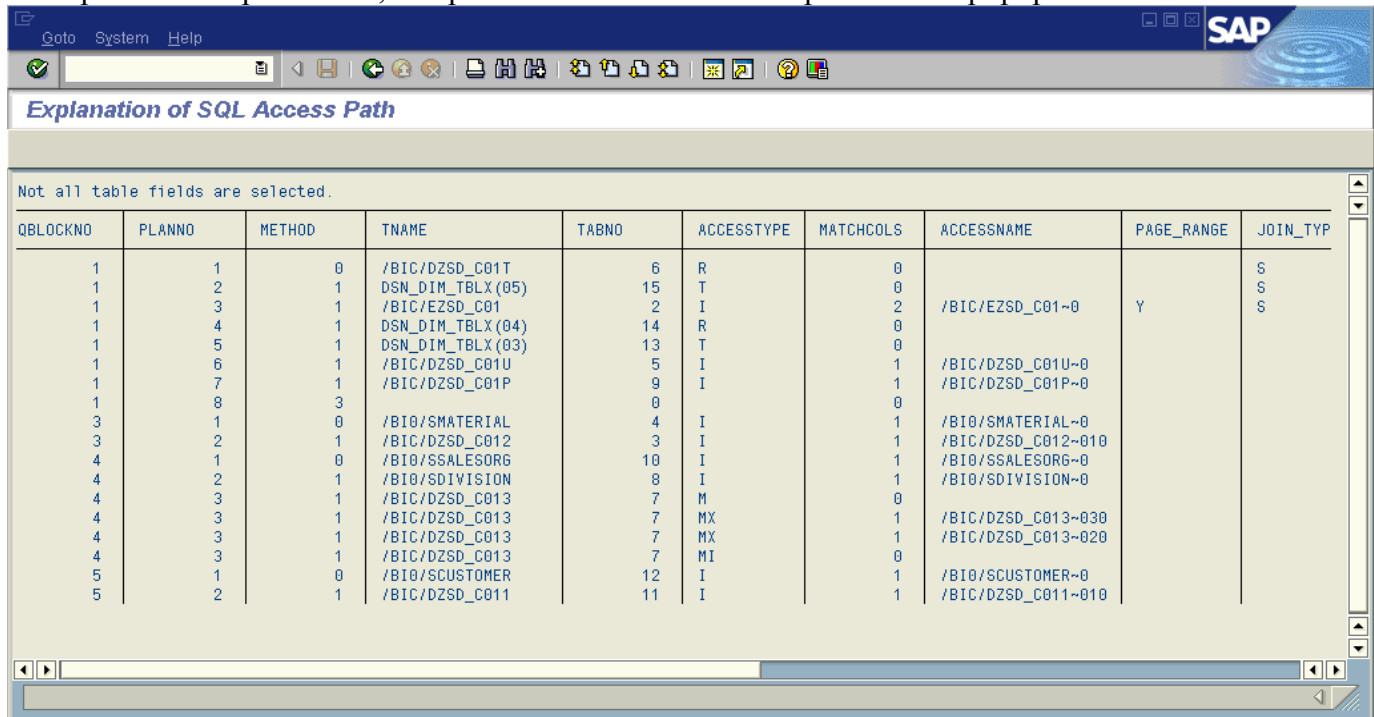


Figure 31: Plan table

This query ran in just a 35 seconds, so it would not generally be a candidate for analysis. If it were a longer running query, with QDBSEL and QDBTRANS being equal, an aggregate would probably not be suitable

for improving performance. (With the exception of special situations where one creates an aggregate to build the navigation attributes into the aggregate dimension tables, in order to avoid materializing subqueries on dimension/navigation attribute joins. See Section 11.2.2 for more information.)

10.2. Starting from query statistics

When a user has called and identified a problem with a query, the SAP table RSDDSTAT can be used to identify the components of the navigation step's response time, since RSDDSTAT can be searched by username and query name. One can then correlate this information with the ST04 statement cache statistics, where the access path can be reviewed.

We have complaint about query performance from user GORDONMR. We search RSDDSTAT for query statistics, in order to review time breakdown of the query and determine where the problem may be.

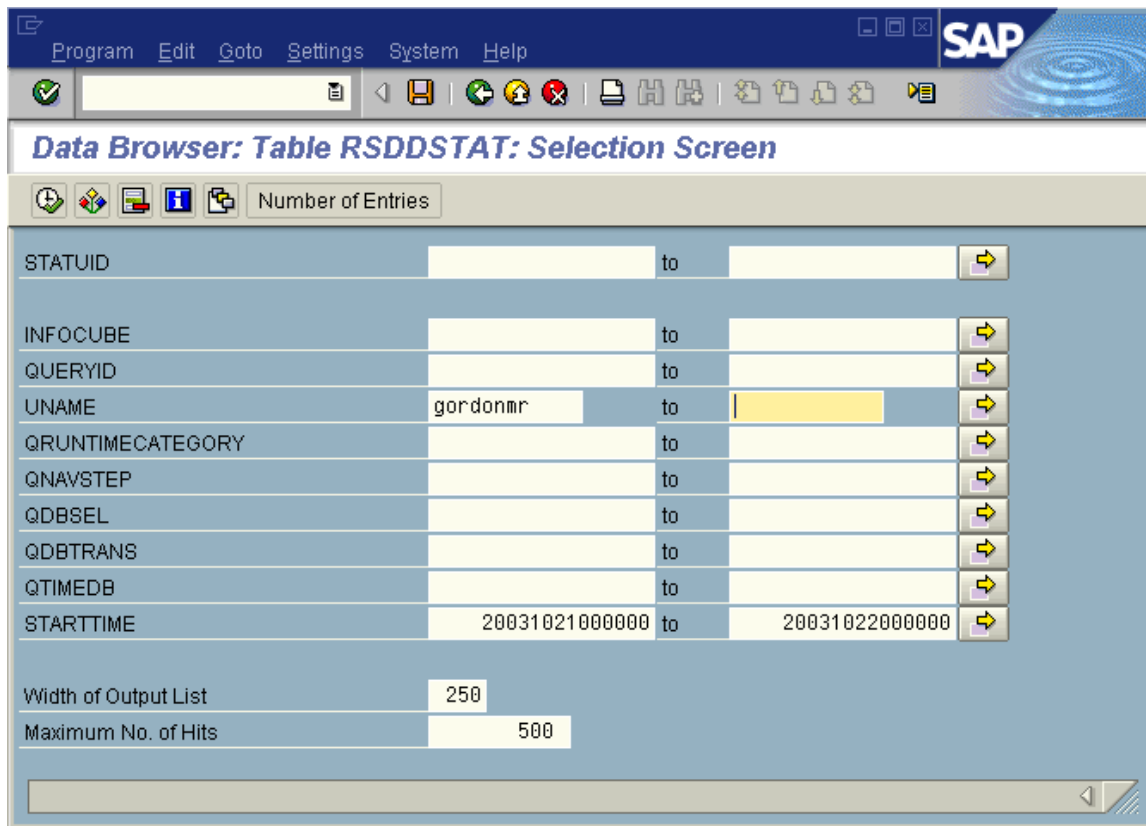


Figure 32: Search RSDDSTAT by name and date for query statistics

Settings > list format > choose fields – select the output fields, to get rid of extraneous information.

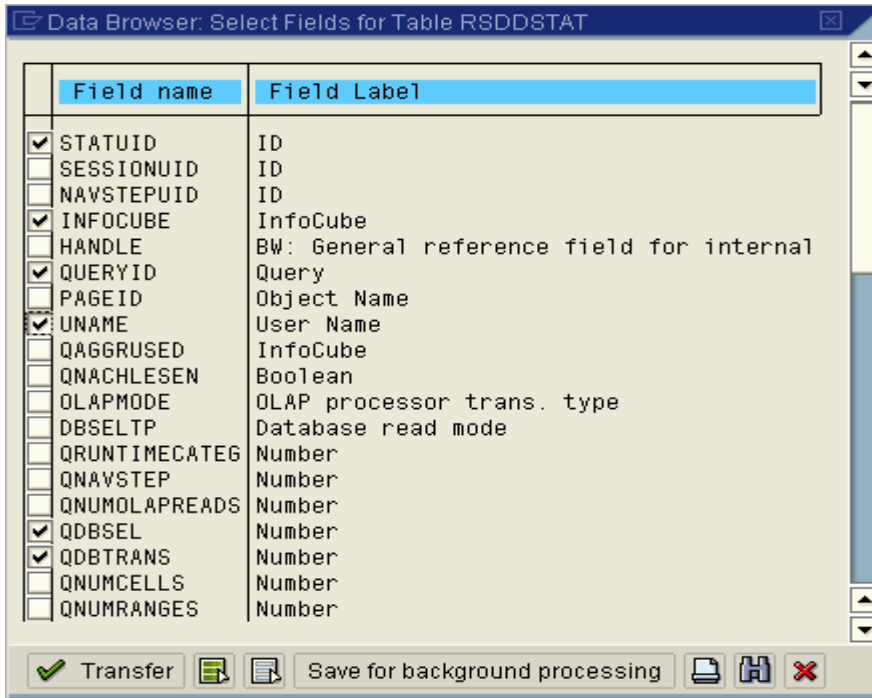
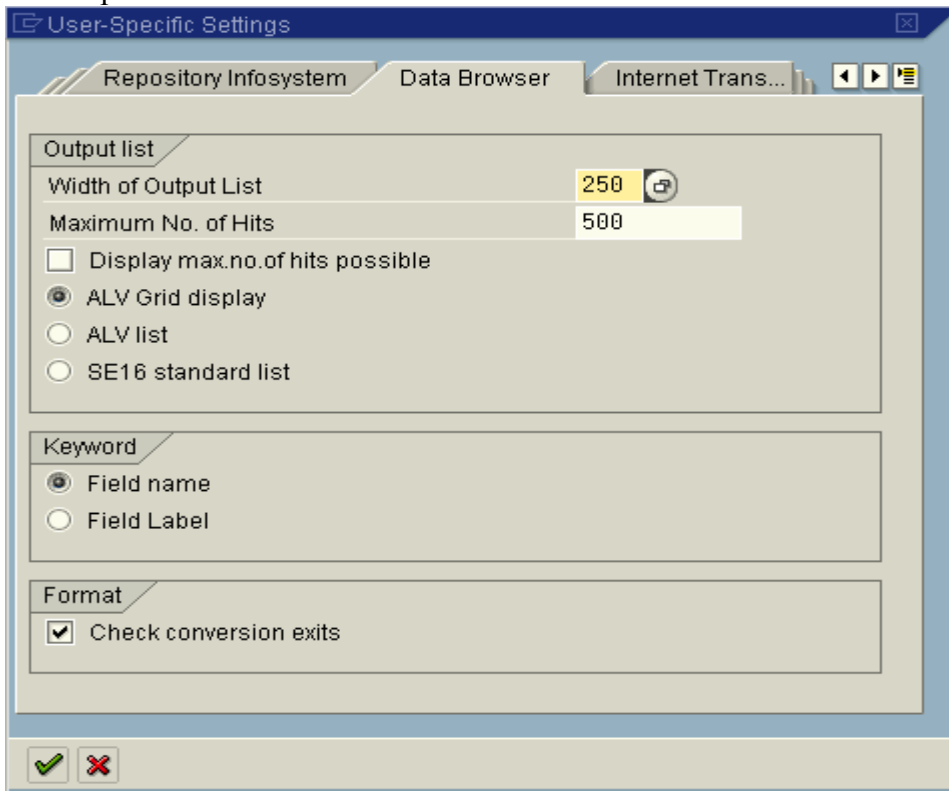


Figure 33: Choose fields

In addition to these, the fields QTIMEDB and STARTTIME should be chosen.

Settings > user parameters – format the output with the names of queries by using the ‘check conversion exits’ option.



Press execute.

STATUID	INFOCUBE	QUERYID	UNAME	QDBSEL	QDBTRANS	QTIMEDB	STARTTIME
3RIB9G542PK2XIW4OG6W258V0	ZSD_C01		GORDONMR	0	0	0.000000	20,031,021,134,548.5994420
3RIB9G55NLD7HEIH5YDWWB50C	ZSD_C01	ZSD_C01/SALESTEST3	GORDONMR	0	0	0.000000	20,031,021,134,553.0447350
3RIB9SGB5F4WVGGRM8Q10JD53W	ZSD_C01	ZSD_C01/SALESTEST3	GORDONMR	33,943	24,101	511.042969	20,031,021,135,040.0510740

Figure 34: RSDDSTAT query statistics

Press 'Display' to review all the fields.

RECCHAVLREAD	1,563
QTIMEOLAPINIT	1.386719
QTIMEOLAP	104.472656
QTIMEDB	511.042969
QTIMEVARDP	22.765625
QTIMEUSER	0.000000
QTIMECLIENT	31.324219
TIMECHAVLREAD	2.492188
TIMEAUTHCHECK	0.007813
TALERTMON	0.000000
TIMEREST	0.000000
ODBOT	0.000000
ODBOT	0.000000

Figure 35: RSDDSTAT Details

In Figure 35, QTIMEDB (database request time) is the largest component of response time, so we need to look at ways to reduce database time.

The first way to evaluate reducing DB time is via aggregates. In Figure 34, QDBSEL/QDBTRANS is about 1.5, which indicates that an aggregate is probably not suitable.

There is one exception to the SAP ROT of having a minimum 10-1 summarization ratio for aggregates, and that is when the aggregate is built in order to create navigation attributes in the aggregate dimension tables, to reduce the number of subqueries necessary to process navigation attributes and dimensions in a BW query. See section 11.2.2 for an example of how aggregates build navigation attributes into dimension tables. This type of aggregate is rarely needed, but may be used in some circumstances, after other efforts to

tune the SQL have not been successful. Since Fact tables can be compressed in DB2, the space impact of adding large aggregates is not as large as it would be with other databases.

Using the statistics in Figure 34, QDBSEL/QTIMEDB (rows matching predicates/database time) = 66. QDBSEL/QTIMEDB is a measure of how fast the query can retrieve result rows. 66 is rather slow and indicates that we should evaluate the way the SQL is executed – e.g. are filtering dimensions joined before the fact table and how is the inside-out processing after the fact table performed. See examples in Section 11.4.2 and Section 11.4.3.

10.3. Starting from ST04 statement cache

As shown in Section 8.1, if a slow or inefficient statement is noticed in the ST04 statement cache, one can use the STATUID to link from the statement cache to the query statistics and info objects.

10.4. Sample explain of V7 sparse index access path

DB2 V7 offers a new type of access, “sparse index on workfile”, which can improve performance of the inside-out (post fact table) joins in BW queries. In DB2 V6, these inside-out joins were often done with Merge Scan Join, which requires a sort of the intermediate result set.

Not all table fields are selected.

QBLOCKNO	PLANNO	METHOD	TNAME	ACCESSTYPE	MATCHCOLS	ACCESSNAME	MERGE_JOIN_COLS	JOIN_TYPE
1	1	0	/BIC/DZARSAI207P	R	0		0	
1	2	1	/BIC/SZILAYMOD	R	0		0	
1	3	1	DSN_DIM_TBLX(04)	R	0		0	
1	4	1	/BIC/FZARSAI207	I	1	/BIC/FZARSAI207-P	0	
1	5	1	DSN_DIM_TBLX(02)	T	0		0	
1	6	1	DSN_DIM_TBLX(05)	T	0		0	
1	7	1	DSN_DIM_TBLX(03)	T	0		0	
1	8	1	DSN_DIM_TBLX(06)	T	0		0	
1	9	3			0		0	
2	1	0	/BIO/SMATL_TYPE	I	1	/BIO/SMATL_TYPE-0	0	
2	2	1	/BIC/XZIARTICLE	R	0		0	
3	1	0	/BIC/DZARSAI2072	R	0		0	
3	2	1	/BIC/SZISITART	I	1	/BIC/SZISITART-001	0	
4	1	0	/BIO/SCITY	I	1	/BIO/SCITY-0	0	
4	2	1	/BIC/XZISITE	R	0		0	
5	1	0	/BIC/SZISCHMRP	R	0		0	
5	2	1	/BIC/SZIVENDTR	R	0		0	
5	3	1	/BIC/XZIVENDOR	R	0		0	
5	4	1	/BIO/SCOUNTRY	I	1	/BIO/SCOUNTRY-001	0	
6	1	0	/BIO/SFISCYEAR	I	0	/BIO/SFISCYEAR-001	0	
6	2	4	/BIC/DZARSAI207T	I	1	/BIC/DZARSAI207T03	0	

Figure 36: Query using sparse indexes on workfiles

In Figure 36, on the lines QBLOCKNO 1, PLANNO 5-8, the METHOD 1 (Nested Loop Join) is used with ACCESSTYPE T (sparse index on workfile). This denotes use of a dynamically generated “sparse index” on the workfile. The MATCHCOLS 0 on these lines can be a bit misleading. An index *is* being used, but it is not a permanent index, such as would be defined in a normal table.

11. Reviewing overall efficiency of the system

When the goal is to examine the overall efficiency of the BW system, to search for queries that use excessive resources, or have long database request times, you can narrow the scope of the problem, by using SAP statistics, in conjunction with DB2 performance indicators.

11.1. *Examine infocubes and queries using lots of database resources*

Two indicators of inefficient queries have already been discussed:

- Rows selected per second (QDBSEL/QTIMEDB)
- Ratio of rows selected to rows transferred (QDBSEL/QDBTRANS)

This section will show ways to use SAP transactions and queries to search for infocubes with slow performance, or queries that have performance problems.

Since infocube and query statistics contain averages for many queries, many problems will not be seen in the averages. Queries that need aggregates, and infocubes or queries that would benefit from new indexes may be found in this way.

11.1.1. ST03N to find infocubes with long DB time

ST03N has been enhanced to provide statistics on BW navigation steps. According to SAP note 403039, ST03N replaces ST03 (which does not have BW navigation step statistics) at support package 10 in 4.6D. One can use ST03N to review the BW performance statistics looking for infocubes and queries that may have performance problems.

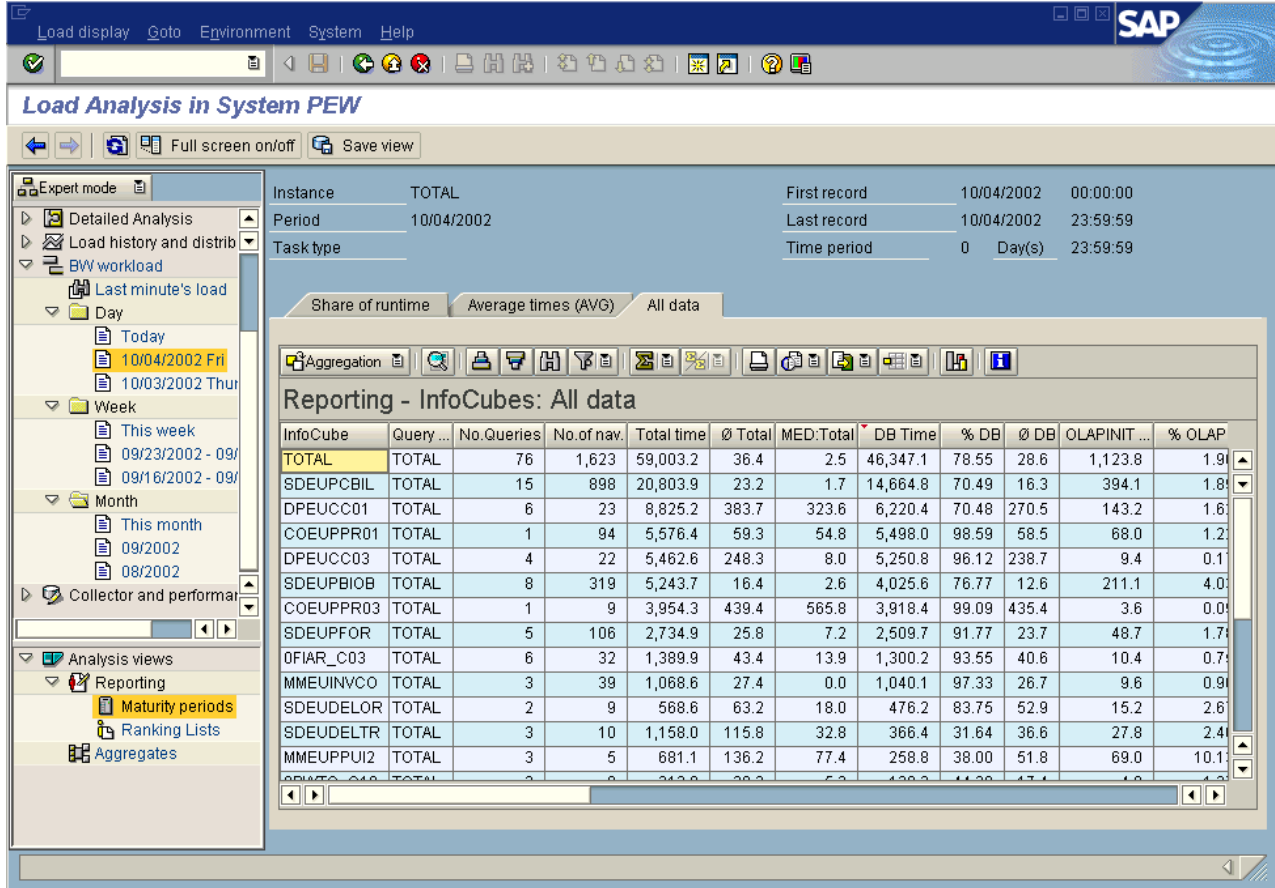


Figure 37: ST03N infocubes

In ST03N, one can check for infocubes with large total or average DB time. Since performance problems are generally related to the characteristics a specific query uses to access the infocube and its aggregates, the query level statistics give better hints in finding slow or inefficient queries.

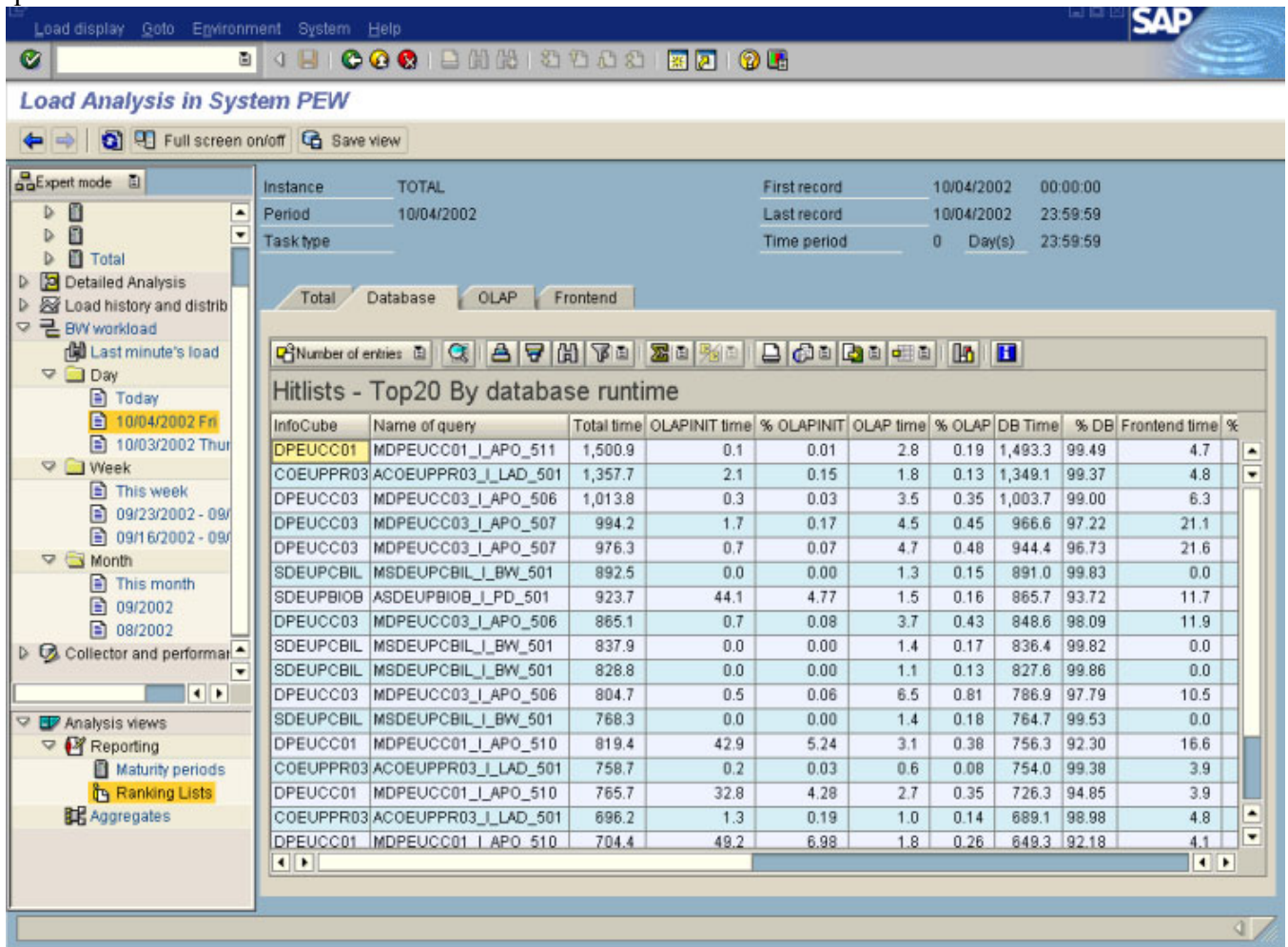


Figure 38: ST03N queries by DB time

In Figure 38, queries with high DB time per navigation step (0 DB column) could be examined for performance problems. Queries with a high Select/Trans ratio (QDBSEL/QDBTRANS) are candidates for improvement via new aggregates.

After finding a query that seems slow or inefficient, one would then use SE16 to display RSDDSTAT and RSDDSTATAGGRDEF to find individual executions of the query, and then use ST04 examine the SQL for the query.

11.1.2. Excel spreadsheet to find slow queries

One can compute the average QDBSEL/QTIMEDB for queries using ST03N statistics. Download the ST03N statistics in Figure 38 to Excel, and create a new column defined as ‘Selected’ / ‘DB Time’.

	A	B	C	D	E	F	G	H	I	J	K	L
	InfoCube	Name of query	Nav. steps	Total time	Rows/DB Time	Total	Med. total	OLAPINIT time	% OLAPINIT	OLAPINIT	OLAP time	% OLAP
1	TOTAL	TOTAL	24,154.00	785,219.6	618.8	32.5	3.5	8,370.6	1.07	0.3	52,484.2	6.68
3	Z_SLS_ORD	QBPO0011	1,139.00	87,212.9	78.0	76.6	24.5	230.4	0.26	0.2	857.9	0.98
4	YIC_SSC	QBSS0005A	2,428.00	38,912.3	1,609.4	15.0	5.4	523.4	1.35	0.2	10,134.3	26.04
5	YIC_SSC	QBSS0005	2,357.00	34,653.8	2,339.4	14.7	3.3	424.0	1.22	0.2	7,959.3	22.97
6	Z_COPA_IC	QBPPA0006	2,133.00	30,498.6	2,092.3	14.3	5.4	681.9	2.24	0.3	4,359.8	14.30
7	Z_SLS_INV	QBPI0006	5,276.00	28,771.6	1,659.5	5.5	2.4	2,025.1	7.04	0.4	4,967.1	17.26
8	ZIC_SMPOS	QSELPOS01	859.00	27,300.0	2,261.6	31.8	5.5	261.2	0.96	0.3	2,556.9	9.37
9	ZIC_SFAP0	QSELMEDMTHL	879.00	20,278.4	1,257.3	23.1	4.5	210.9	1.04	0.2	827.0	4.08
10	ZIC_MINFC	QSELMTDSST0	436.00	19,338.5	653.8	44.4	4.5	388.9	2.01	0.9	1,131.0	5.85
11	Z_SLS_ORD	QBPO0001	760.00	19,084.5	42.9	25.1	3.5	90.5	0.47	0.1	1,303.0	6.83
12	ZIC_FAPSO	QSELNSAA02	372.00	16,306.5	2,509.8	43.5	7.5	108.8	0.67	0.3	680.3	4.17
13	unknown	3MOPZGD8SRIM	68.00	13,238.8	131.2	194.7	33.9	46.1	0.35	0.7	296.5	2.24
14	Z_SLS_ORD	QBPO0007	373.00	13,027.8	1,973.5	34.9	8.6	142.3	1.09	0.4	1,314.6	10.09
15	ZIC_SUM	QSELAAST01	246.00	12,870.0	1,357.3	52.3	5.9	285.2	2.22	1.2	1,545.5	12.01
16	Z_SLS_INV	QBPI0001	1,043.00	12,142.0	2,746.4	11.6	4.5	267.0	2.20	0.3	3,278.9	27.00
17	ZIC_SFAP0	QSELMKTGBDC	165.00	11,940.0	2,842.9	72.4	6.3	115.1	0.96	0.7	440.0	3.69
18	ZIC_ORD	QSELSLSORD0	721.00	8,057.1	666.2	11.2	4.4	530.8	6.59	0.7	1,294.0	16.06
19	ZIC_SFAP0	QSELMEDMTHL	239.00	7,442.4	884.8	24.8	4.4	227.1	2.20	0.7	733.8	9.87

Figure 39: XL spreadsheet to highlight slow queries

In Figure 39, the queries that are slow in terms of rows selected/second have been highlighted. Queries such as QBPO0011 could be executed and evaluated, to determine the reason they have a slow rows/second rate.

11.2. Review need for aggregate tables

As mentioned in section 6.5.4, aggregates are a very important tool in improving BW query performance. Aggregates can improve query performance in several ways:

- When data is aggregated, the query can search and retrieve the user's result in fewer rows.
- When data is aggregated based on navigation attributes, the navigation attributes are incorporated into the aggregate dimension tables. This simplifies the SQL to retrieve rows, and improves performance.
- Aggregation is a way to offload some of the processing needed to create reports to nights or weekends, rather than doing it all real-time.

Well-defined aggregates are generally the most effective performance-tuning tool in a BW system. For queries where QDBSEL/QDBTRANS > 10, one generally has the potential to get much larger performance improvements through aggregates than through other DB and OS level tuning.

Sections 11.2.1 and 11.2.3 show some of the technical reasons for improved SQL performance using aggregates. Subsequent sections in this chapter show how to evaluate aggregates.

11.2.1. Strategies for including attributes in an Infocube

Please note that an attribute based on a characteristic (e.g. material group based on material number) can be included in an infocube in several different ways. Each method has implications for reporting, and for performance.

- Attribute defined in master data table – the attribute is defined in a master data table, while its characteristic is in the dimension table. When a report or aggregate groups using this attribute, rows will be grouped using the current material/material group structure in the master data. But, this may cause performance problems, since the navigation attribute table must be joined to the dimension table in order to do the grouping.
- Attribute defined in dimension table – the attribute, as well as its characteristic, can be defined in a single dimension. Since the value of the attribute is set at the time each row is added to the cube, then this would allow one to group material according to material group, using the material group structure at the time of the transaction. This can offer better performance than when the attribute is defined in master data, since the grouping can be done from the dimension table, without a separate join to master data. However, the report or aggregate will reflect historical material grouping (that is, the material group for the item at the time the row was added to the cube), not current.
- Attribute in time-dependent master data – the attribute is in the master data table, and its characteristic is in the dimension table. One can report using the material/material group relationship as it stood at any time, since all changes to the material/material group relationship are available in the master data. Since the material group is in master data, and has validity dates, this can have the highest run-time cost of the three options listed here. Aggregates, as shown in Section 11.2.5, can be built with a specific 'Key Date' to improve performance of queries using time dependent navigation attributes.

11.2.2. Navigation attributes in aggregate dimension table

In the next example, the query uses navigational attributes. Since infocube navigational attributes are stored in attribute SID tables separate from the dimension tables, DB2 must make an extra join step to

incorporate navigation attributes into the result. This is often processed as a subquery. When an aggregate is defined based on navigation attributes, the navigation attributes will be incorporated into the aggregate dimension tables, and DB2 will not need to evaluate a subquery to process the navigation attribute. Along with the row summarization that occurs with aggregations, this simplification of the SQL can speed up the query.

Here are the characteristics used in a query that used the aggregate.

STATUID	AGGRNUM	IOBJNM	QUERYCUBE	AGGRST	HIESID	TLEVEL	VALUE	FIXSID
EPA5NB3ZD42SLDFOUW9Q9VUW0		0BILL_TYPE_EDOCGRP	100089	F	0		SLS	4
EPA5NB3ZD42SLDFOUW9Q9VUW0		0CALMONTH	100089	*	0	02		0
EPA5NB3ZD42SLDFOUW9Q9VUW0		0CUST_SALES_ECUSTR2	100089	*	0	02		0
EPA5NB3ZD42SLDFOUW9Q9VUW0		0CUST_SALES_ECUSTR5	100089	*	0	02		0
EPA5NB3ZD42SLDFOUW9Q9VUW0		0MATERIAL_EPRODH06	100089	*	0	02		0
EPA5NB3ZD42SLDFOUW9Q9VUW0		0SALESORG	100089	F	0	2799		26
EPA5NB3ZD42SLDFOUW9Q9VUW0		0SOLD_TO_EINTCOIND	100089	F	0	02		2
EPA5NB3ZD42SLDFOUW9Q9VUW0		EINVQTYCA			0			0
EPA5NB3ZD42SLDFOUW9Q9VUW0		EINVQTYSU			0			0
EPA5NB3ZD42SLDFOUW9Q9VUW0		ESUBTOT2M			0			0

Figure 40: RSDDSTATAGGRDEF display of query using navigation attributes

Explaining the SQL that executed the query, there are no subqueries for joining the navigation attributes to the dimension tables. The dimension tables contained the navigation attribute.

QBLOCKNO	PLANNO	METHOD	TNAME	ACCESSTYPE	MATCHCOLS	ACCESSNAME	MERGE_JOIN_COLS	CORRELATION_NA
1	1	0	/BIC/D100089P	I	1	/BIC/D100089P-030	0	DP
1	2	1	/BIC/D100089T	R	0		0	DT
1	3	1	/BIC/E100089	I	2	/BIC/E100089-0	0	E
1	4	1	/BIC/D1000891	I	1	/BIC/D1000891-0	0	D1
1	5	1	/BIC/DSDEUPCBILU	I	1	/BIC/DSDEUPCBILU-0	0	DU
1	6	1	/BIC/D1000892	I	1	/BIC/D1000892-0	0	D2
1	7	3			0		0	

Figure 41: Explain plan of query on aggregate containing navigation attributes

11.2.3. SID index columns in aggregates

If a dimension for an aggregate contains only a single characteristic, or if an aggregate contains only a few characteristics, SAP can create the aggregate fact table keys using SIDs, rather than DIMIDs. This removes a dimension table join when the aggregate fact table is accessed. When SID is used to create

the fact table key, and the indexed characteristic is used, rather than joining by SID to DIMID to Fact table, DB2 can join using the SID to Fact table.

In Figure 42, the explain table information for a query using an aggregate shows only two dimension tables (P and U) and an attribute SID table (/BI0/XMATERIAL) being joined to the aggregate fact table.

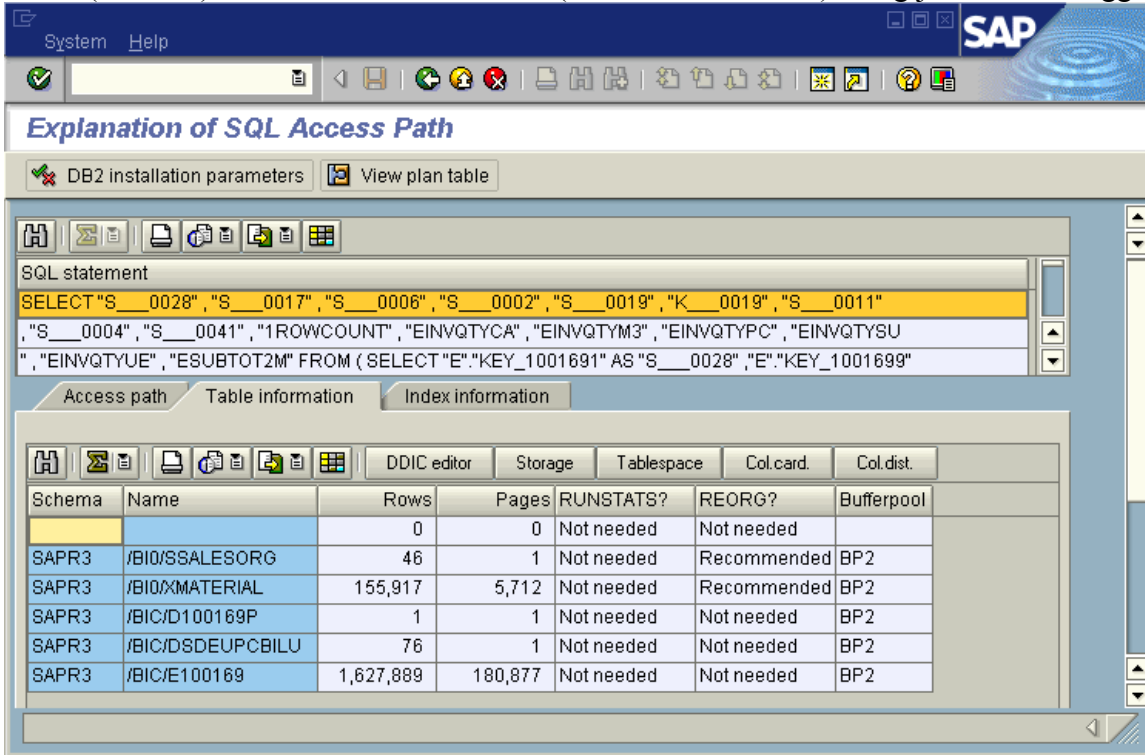


Figure 42: Aggregate fact table information

But if we look at the SQL statement, there are predicates on other columns in the E fact table, too.

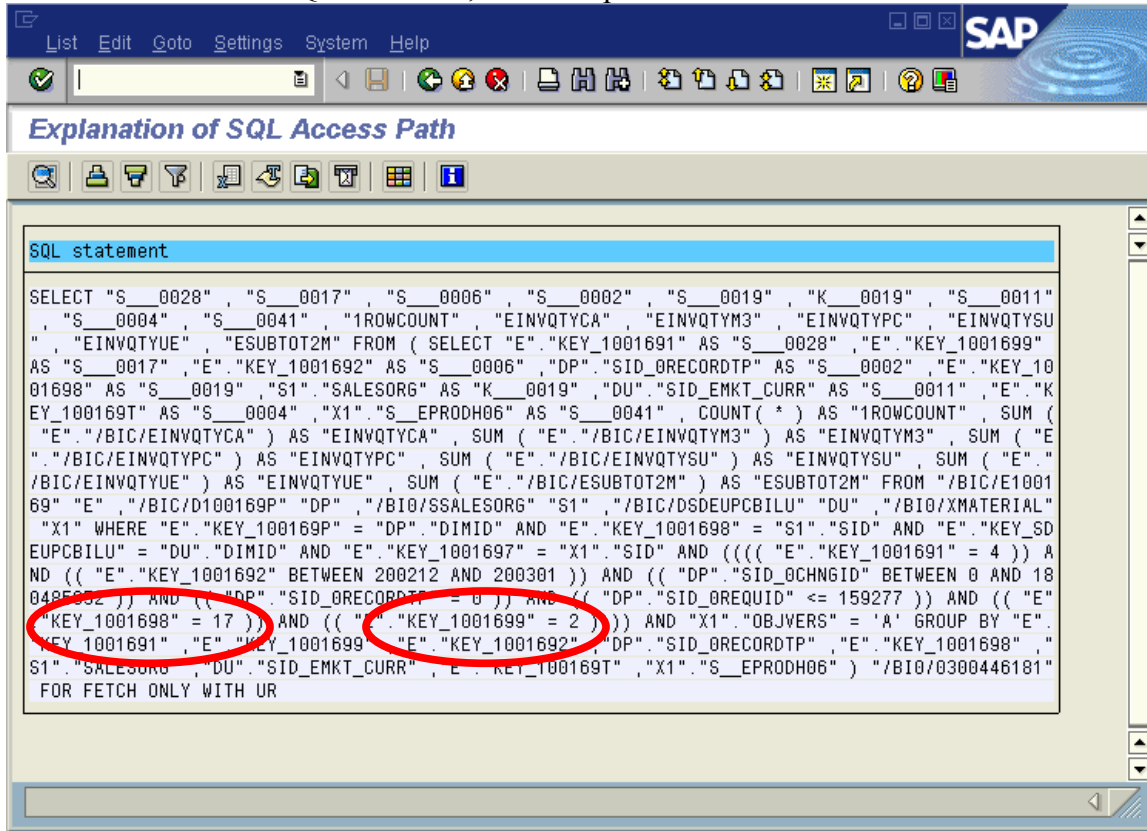


Figure 43: Aggregate fact table SQL statement

For instance, in Figure 43, some of the predicate columns (E.KEY_100169x) are used to directly access the E fact table.

Viewed from RSA1, the aggregate contains the usual characteristics and navigation attributes on characteristics.

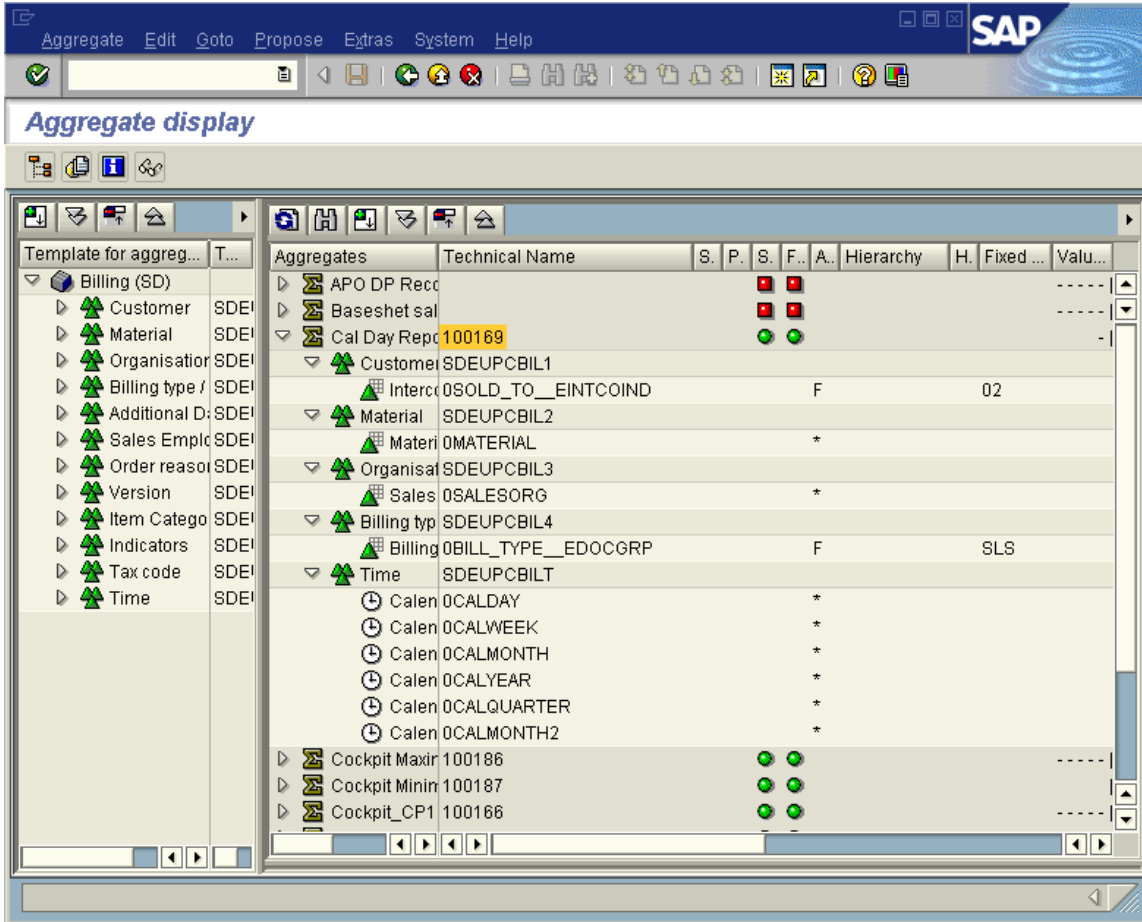


Figure 44: Aggregate fact table RSA1 display

Viewed from DB2, we see that the fact table has been built using SIDs as the index columns for many of the characteristics. Note the “field type” of RSSID for SID columns, and RSDIMID for dimension columns.

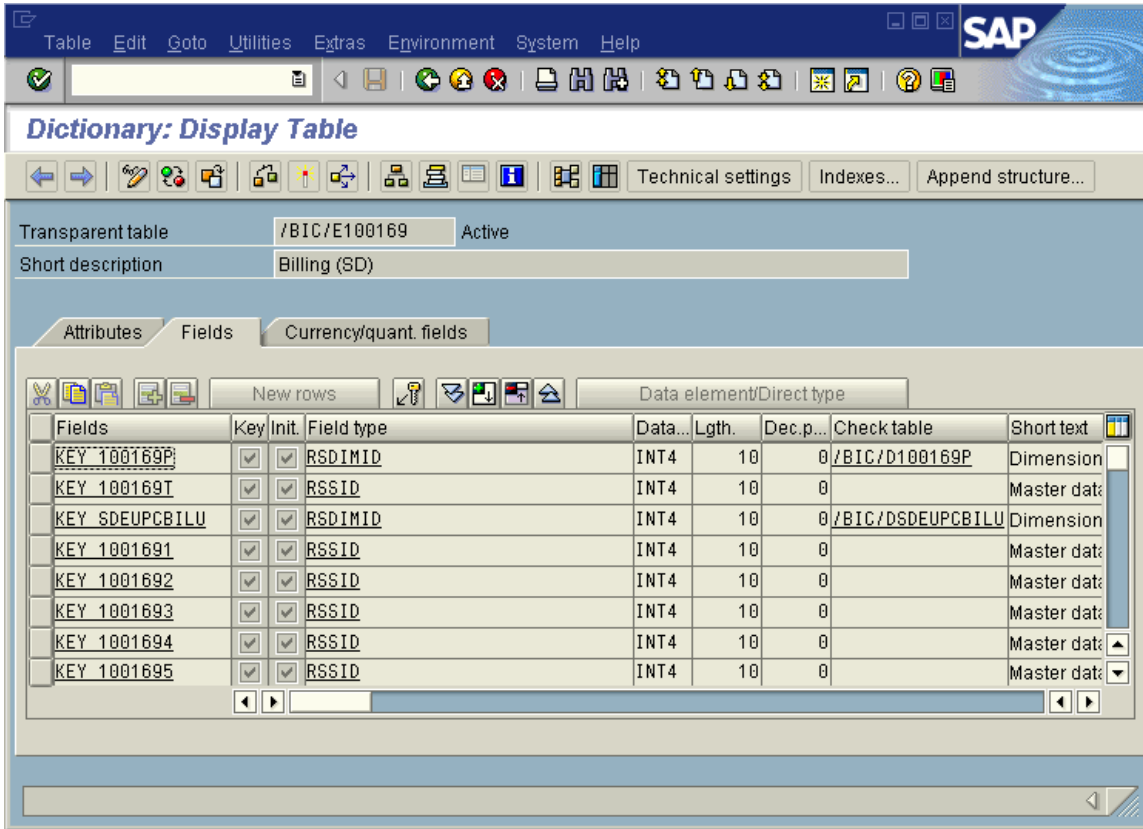
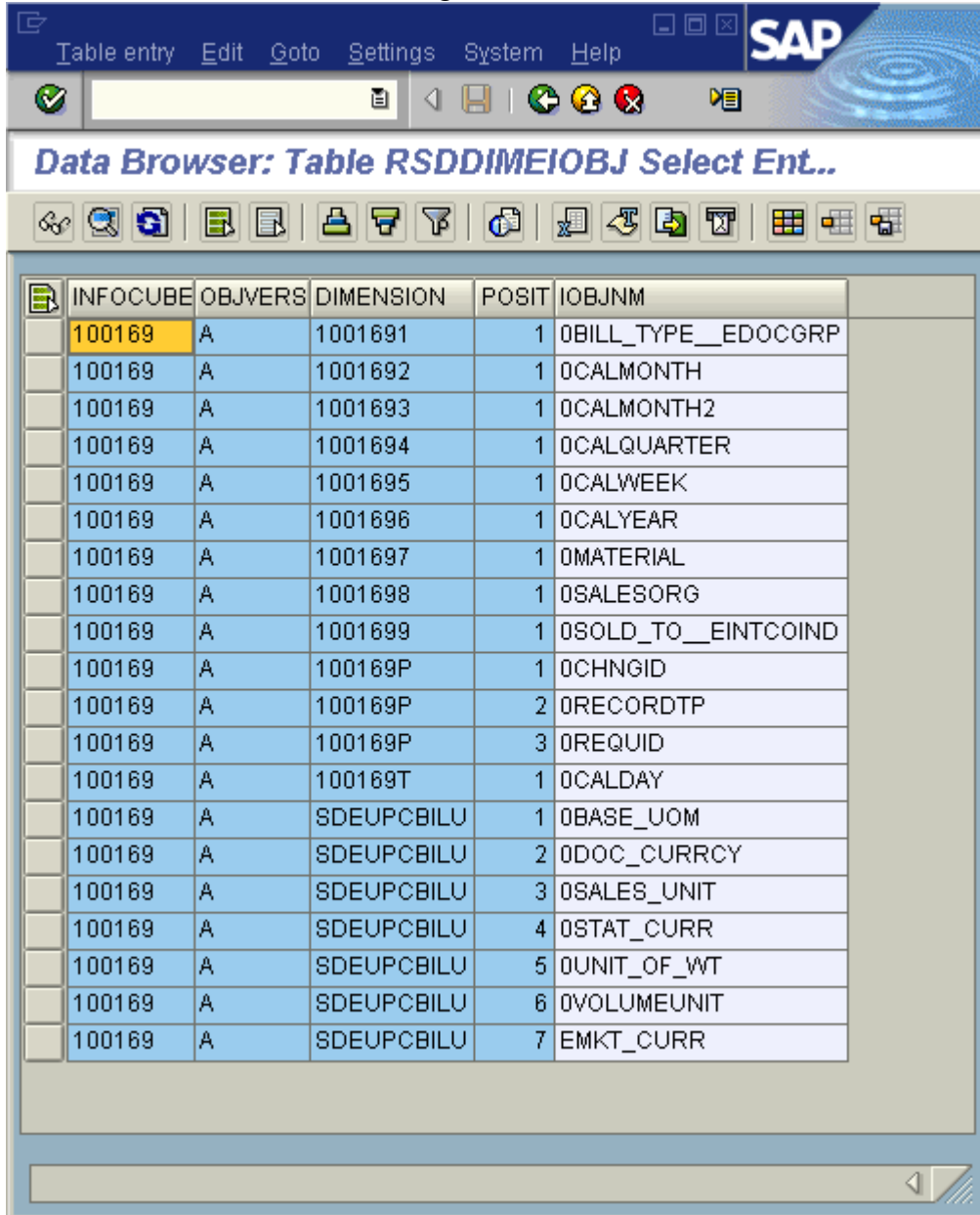


Figure 45: Aggregate fact table SE11 display

The characteristics that correspond to the key columns on the fact table can be found in table RSDDIMEIOBJ, as is shown in Figure 46.



INFOCUBE	OBJVERS	DIMENSION	POSIT	IOBJNM
100169	A	1001691	1	0BILL_TYPE__EDOCGRP
100169	A	1001692	1	0CALMONTH
100169	A	1001693	1	0CALMONTH2
100169	A	1001694	1	0CALQUARTER
100169	A	1001695	1	0CALWEEK
100169	A	1001696	1	0CALYEAR
100169	A	1001697	1	0MATERIAL
100169	A	1001698	1	0SALESORG
100169	A	1001699	1	0SOLD_TO__EINTCOIND
100169	A	100169P	1	0CHNGID
100169	A	100169P	2	0RECORDTP
100169	A	100169P	3	0REQUID
100169	A	100169T	1	0CALDAY
100169	A	SDEUPCBILU	1	0BASE_UOM
100169	A	SDEUPCBILU	2	0DOC_CURRCY
100169	A	SDEUPCBILU	3	0SALES_UNIT
100169	A	SDEUPCBILU	4	0STAT_CURR
100169	A	SDEUPCBILU	5	0UNIT_OF_WT
100169	A	SDEUPCBILU	6	0VOLUMEUNIT
100169	A	SDEUPCBILU	7	EMKT_CURR

Figure 46: RSDDIMEIOBJ - dimensions and characteristics

Compare the table field names in Figure 45 with the dimension names in Figure 46 to find the characteristics associated with the indexes on the fact table. Queries are created using characteristics, and SQL is executed using table columns. RSDDIMEIOBJ links them together to make it easier to interpret the business function of the query.

The BW administrator does not control whether SAP creates the fact table key with SID columns or by DIMID columns. If there are few characteristics in the aggregate, SAP does this when the aggregate is

created. This example is included here to help interpret SQL using aggregates, and to show another performance benefit of aggregates.

11.2.4. Navigation attribute processing with aggregate query

While navigation attributes *can* be defined in the aggregate to optimize query performance, if the characteristic on which the navigation attribute is based is included in the aggregate, and a query uses a navigation attribute, SAP may still choose the aggregate. DB2 will join the attribute SID table to the aggregate to evaluate the navigation attribute.

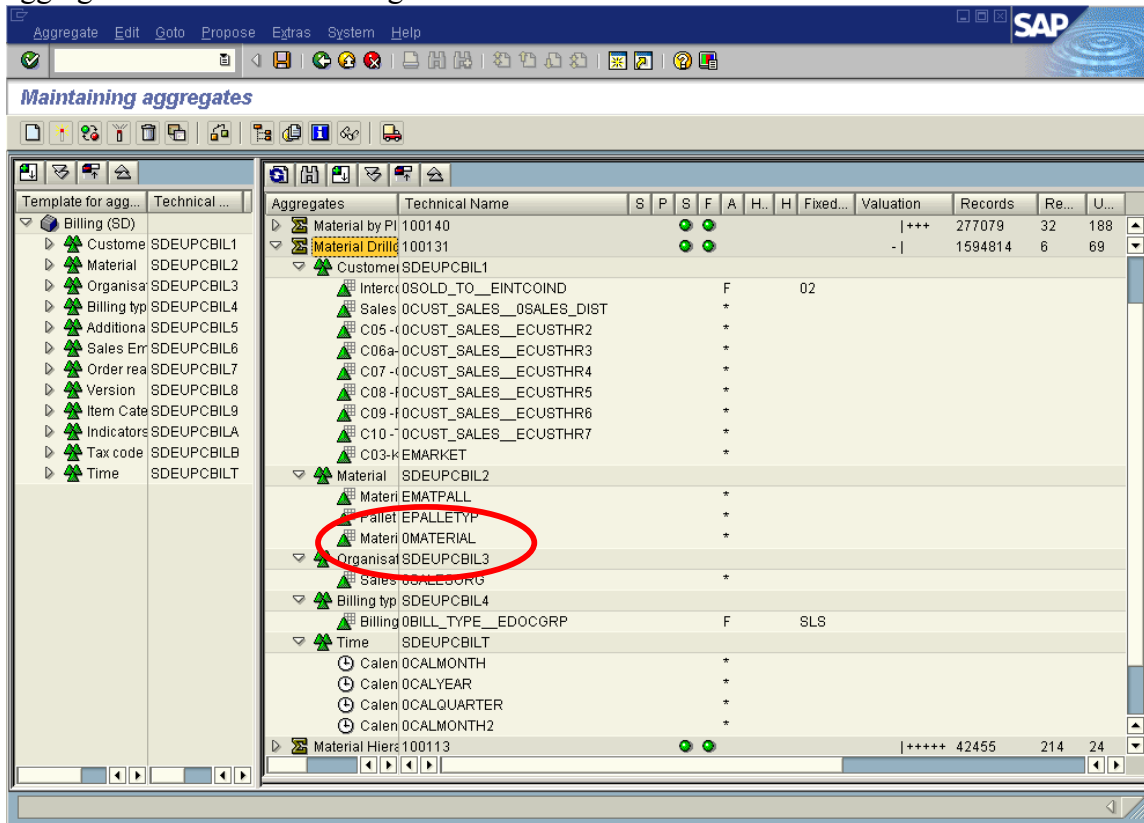


Figure 47: aggregate containing 0MATERIAL characteristic

In Figure 47 the aggregate contains the 0MATERIAL characteristic.

RSDDSTATAGGRDEF shows that the query executed by the user (Figure 48) uses navigation attributes on 0MATERIAL, such as 0MATERIAL__EPRODH01.

STATUID	AGGRNUM	IOBJNM	QUERYCUBE	AGGRST	HIESID	TLEVEL	VALUE	FIXSID
3C78GDM8A3V3MEY1W42CR6186		0BILL_TYPE_EDOCGRP	100131	F	0		SLS	4
3C78GDM8A3V3MEY1W42CR6186		0CALMONTH2	100131	*	0	02		0
3C78GDM8A3V3MEY1W42CR6186		0CALYEAR	100131	*	0	02		0
3C78GDM8A3V3MEY1W42CR6186		0CUST_SALES_ECUSTR5	100131	F	0	0018501186	5,171	
3C78GDM8A3V3MEY1W42CR6186		0CUST_SALES_ECUSTR7	100131	*	0	02		0
3C78GDM8A3V3MEY1W42CR6186		0MATERIAL__EPRODH01	100131	*	0	02		0
3C78GDM8A3V3MEY1W42CR6186		0MATERIAL__EPRODH02	100131	*	0	02		0
3C78GDM8A3V3MEY1W42CR6186		0MATERIAL__EPRODH04	100131	F	0	101112		2
3C78GDM8A3V3MEY1W42CR6186		0REGATE_BAG			0			0
3C78GDM8A3V3MEY1W42CR6186		0SALESORG	100131	F	0	1799		14
3C78GDM8A3V3MEY1W42CR6186		0SOLD_TO__EINTCOIND	100131	F	0	02		2
3C78GDM8A3V3MEY1W42CR6186		EINVQTYCA			0			0

Figure 48: query contains navigation attributes on 0MATERIAL

When SAP chooses the aggregate 100131 to use to support the query, DB2 joins the attribute SID table /BIO/XMATERIAL to the aggregate fact table to evaluate the result.

QBLOCKNO	PLANNO	METHOD	TNAME	ACESSTYPE	MATCHCOLS	ACCESSNAME	MERGE_JOIN_COLS	CORRELATIO
1	1	0	/BIO/XMATERIAL	I	1	/BIO/XMATERIAL~Z5	0	X7
1	2	2	/BIC/D1001312	R	0		1	D2
1	3	1	/BIC/E100131	I	1	/BIC/E100131~050	0	E
1	4	1	/BIC/D1001311	I	1	/BIC/D1001311~0	0	D1
1	5	1	/BIC/D100131T	I	1	/BIC/D100131T~0	0	DT
1	6	1	/BIC/DSDEUPCBILU	I	1	/BIC/DSDEUPCBILU~0	0	DU
1	7	1	/BIC/D100131P	I	1	/BIC/D100131P~0	0	DP
1	8	3			0		0	

Figure 49: DB2 join attribute SID table to aggregate for navigation attribute processing

The point of this example is that while navigation attributes can be included in an aggregate to improve performance, if the characteristic is contained in the aggregate, then the navigation attribute can be processed at execution time by DB2. This is a partially optimized aggregate, which will probably give better performance than using the infocube, but not be as fast as an aggregate that contains the navigation attributes.

11.2.5. Time dependent attributes in aggregates

As mentioned in section 6.5.1.7, starting in BW 3.0 time-dependent attributes can be defined in aggregates. Previous to 3.0, time-dependent attributes must be processed at runtime, by joining the time-dependent attribute table to the infocube or aggregate fact table.

When a time-dependent attribute is defined in an aggregate, the aggregate will be time-dependent, and will have a “Key Date”. *Queries that select the results as of the time of the ‘Key Date’ can use the aggregate. Queries that select the results as of another time cannot use the aggregate.*

Time dependent attributes in aggregates can be used in a situation where a single reference date is used when selecting the data, as when the aggregate would contain data as of the end of the most recent month or fiscal period.

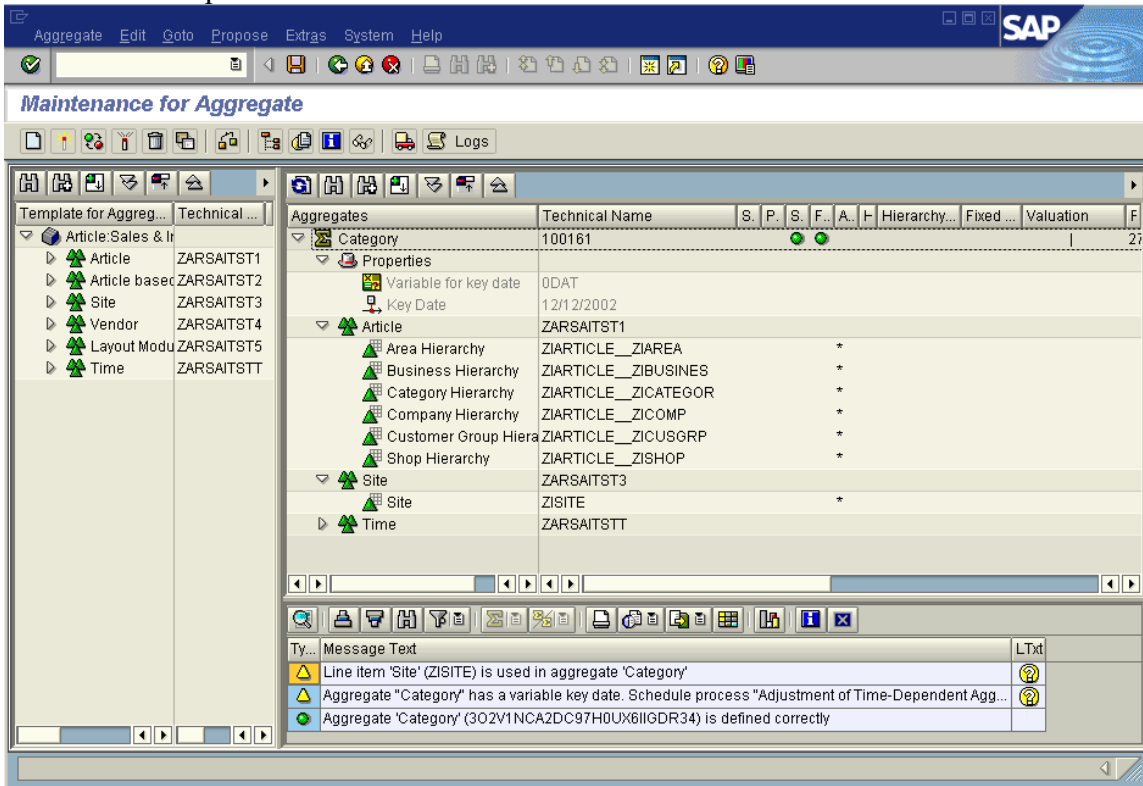


Figure 50: Time-dependent aggregate

In Figure 50, there are time-dependent attributes on the ZISITE characteristic. (One would use transaction RSD1 to review the attributes for the ZISITE characteristic.) Since the ZISITE characteristic is aggregated with level * (all characteristic values) the aggregate is time-dependent. The time-dependent attribute table for ZISITE (/BIC/XZISITE) will be joined to the aggregate, to evaluate time-dependent characteristic.

Tables starting with /BIx/Q and /BIx/Y are time dependent. If you see long running queries with time dependent tables, review whether the queries reference a specific key date such that an aggregate including the time dependent characteristic could help performance.

If an aggregate is created at BW 3.0, and none of the characteristics in the aggregate are time-dependent, then the aggregate will not be time-dependent, and will not have a “Key Date”

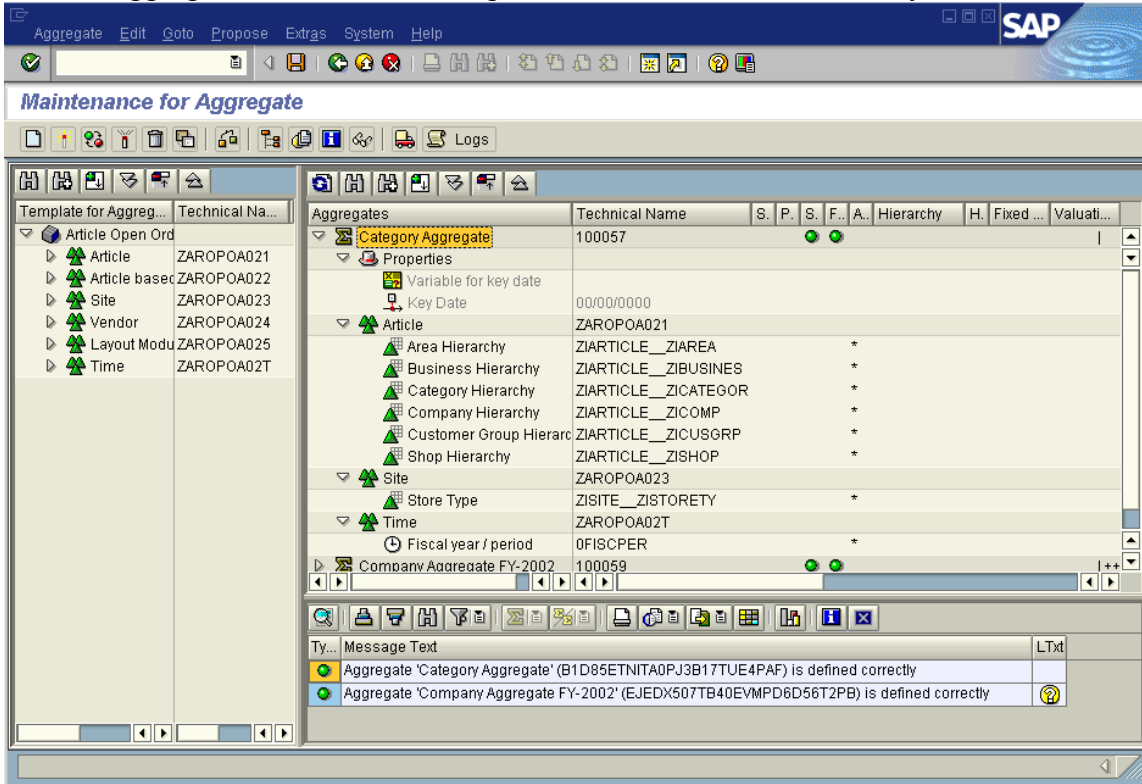


Figure 51: BW 3.0 Aggregate without time-dependent characteristics

In Figure 51, note that the “key date” is zero – this aggregate is not time-dependent, since none of its characteristics are time-dependent.

11.2.6. Defining a navigation attribute and its characteristic in an aggregate

As described in section 11.2.1, defining an aggregate at the level of a navigation attribute has two benefits

- The data is aggregated more than an aggregate on the navigation attribute’s base characteristic.
- The navigation attribute is incorporated into the aggregate’s dimension tables, which simplifies SQL.

It is also possible to build an aggregate containing navigation attributes, as well as the characteristics on which they are based. In this case, the data will be aggregated at the level of the characteristic, but the navigation attribute will be incorporated into the aggregate’s dimension tables. The aggregate fact table will be larger than a table aggregated on only the navigation attribute, but when the navigation attribute is used, DB2 will be able to avoid one level of join, since the navigation attribute is in the dimension table.

If end users will be using navigation attributes in reporting, it is generally preferable to build the aggregate at the level of the navigation attributes (not characteristic), to reduce the aggregate size, and simplify SQL. But, in the case of SQL performance problems related to navigation attribute processing,

one can use the RSA1 ‘Expert Mode’ to define an aggregate with both the characteristic and its navigation attributes.

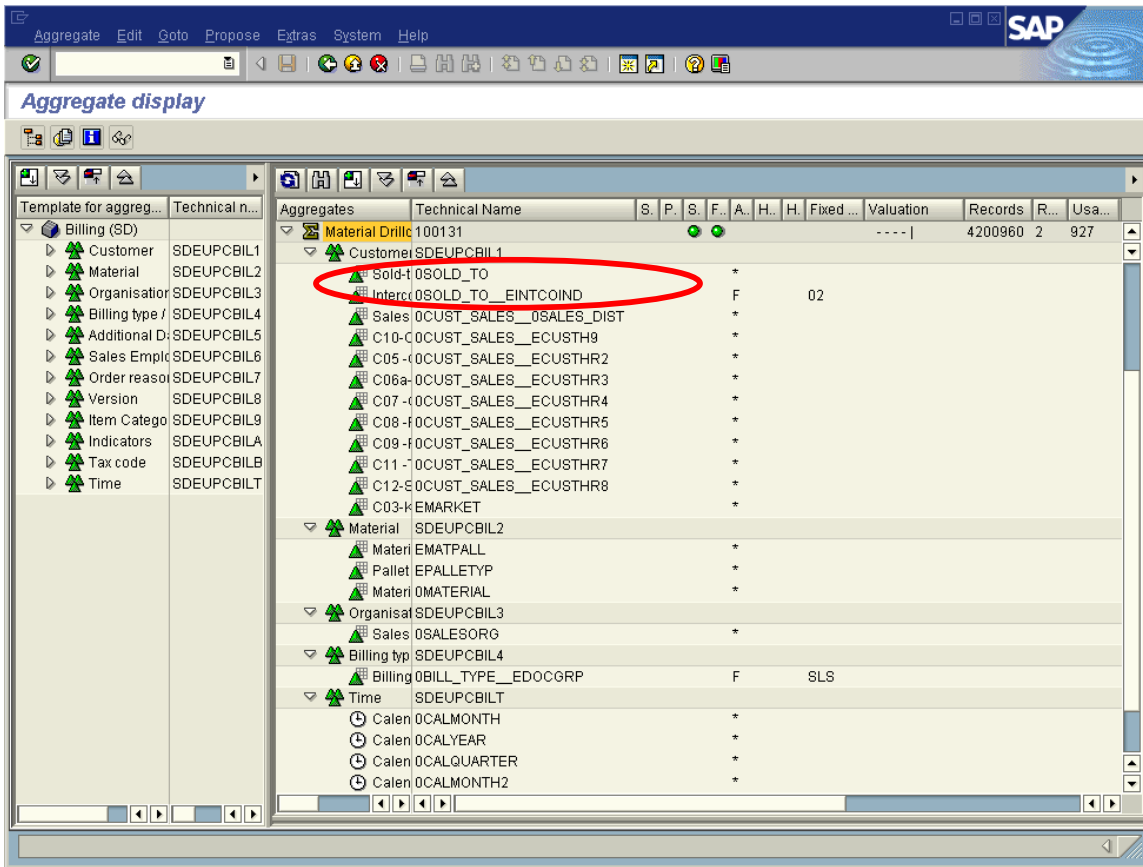
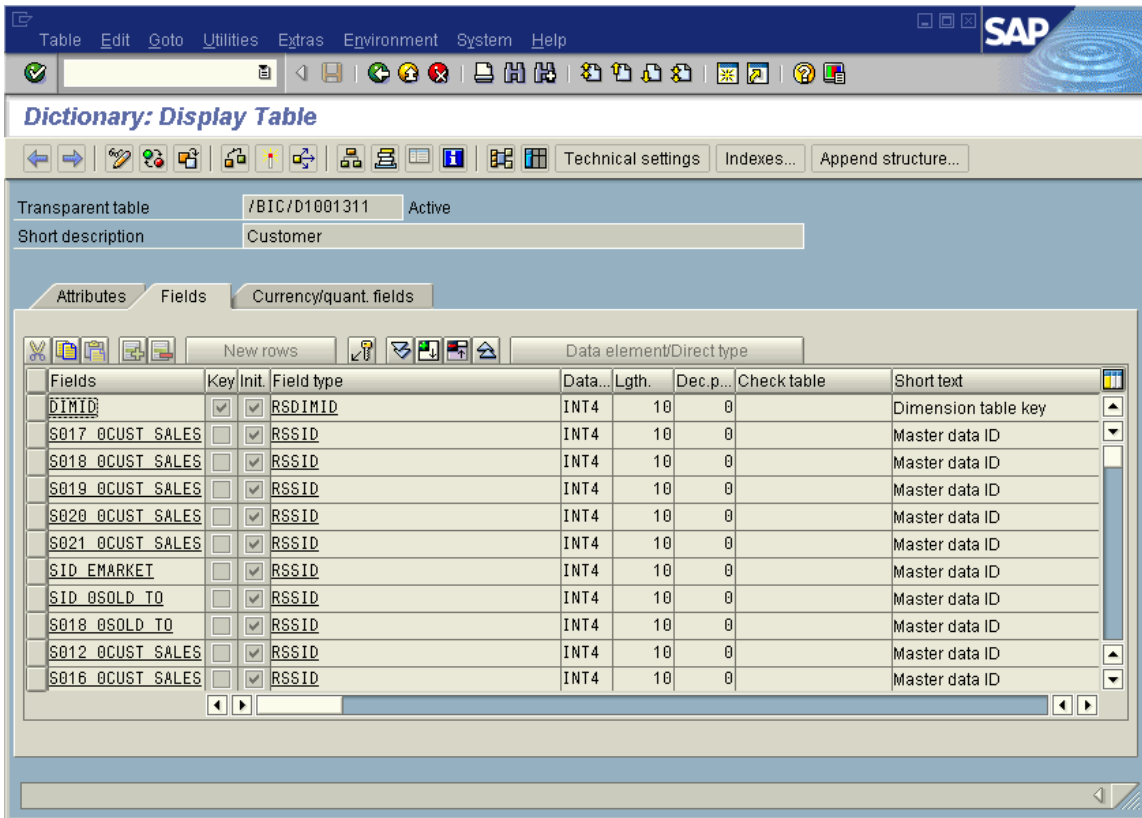


Figure 52: aggregate containing both characteristic and navigation attribute on characteristic

In the RSA1 aggregate display in Figure 52, the SOLD_TO characteristic is included in the definition of the aggregate, so the aggregate is built at the characteristic level for SOLD_TO. The SOLD_TO__EINTCOIND navigation attribute is also included in the definition, which creates an index column in the customer dimension on the aggregate.



Transparent table: /BIC/D1001311 Active
Short description: Customer

Attributes Fields Currency/quant. fields

Fields	Key	Init.	Field type	Data...	Lgth.	Dec.p...	Check table	Short text
DIMID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	RSDIMID	INT4	10	0		Dimension table key
S017 0CUST SALES	<input type="checkbox"/>	<input checked="" type="checkbox"/>	RSSID	INT4	10	0		Master data ID
S018 0CUST SALES	<input type="checkbox"/>	<input checked="" type="checkbox"/>	RSSID	INT4	10	0		Master data ID
S019 0CUST SALES	<input type="checkbox"/>	<input checked="" type="checkbox"/>	RSSID	INT4	10	0		Master data ID
S020 0CUST SALES	<input type="checkbox"/>	<input checked="" type="checkbox"/>	RSSID	INT4	10	0		Master data ID
S021 0CUST SALES	<input type="checkbox"/>	<input checked="" type="checkbox"/>	RSSID	INT4	10	0		Master data ID
SID 0MARKET	<input type="checkbox"/>	<input checked="" type="checkbox"/>	RSSID	INT4	10	0		Master data ID
SID 0SOLD TO	<input type="checkbox"/>	<input checked="" type="checkbox"/>	RSSID	INT4	10	0		Master data ID
S018 0SOLD TO	<input type="checkbox"/>	<input checked="" type="checkbox"/>	RSSID	INT4	10	0		Master data ID
S012 0CUST SALES	<input type="checkbox"/>	<input checked="" type="checkbox"/>	RSSID	INT4	10	0		Master data ID
S016 0CUST SALES	<input type="checkbox"/>	<input checked="" type="checkbox"/>	RSSID	INT4	10	0		Master data ID

Figure 53: SE11 display of customer dimension on aggregate 100131

In the SE11 display of the dimension table in Figure 53, there are two columns in the aggregate's customer dimension related to SOLD_TO. One column supports the SOLD_TO characteristic (SID_0SOLD_TO), and the other is for the SOLD_TO__EINTCOIND navigation attribute.

11.2.7. Recap of characteristic and navigation attribute definition in aggregates

11.2.7.1. Aggregate defined at characteristic level

When aggregates are defined at the level of a characteristic, the data is not summarized as much as an aggregate defined at the level of a navigation attribute on the characteristic. DB2 can process any navigation attribute on the characteristic, by joining the attribute SID table to the aggregate table.

This configuration is the most flexible, as it can support reporting on any navigation attribute on the characteristic.

11.2.7.2. Aggregate defined at characteristic level with navigation attribute

These aggregates are summarized at characteristic level, but contain the navigation attribute in the dimension tables. In the case of access path problems related to processing navigation attribute (attribute SID) tables, this configuration can help improve SQL performance. One disadvantage of this configuration, compared to a definition only at the characteristic level, is that when there are

master data changes that affect the navigation attributes in the aggregate, it will be necessary to do a “realignment run” to adjust the aggregate.

This configuration should be chosen to solve SQL performance problems, if the data cannot be aggregated at the level of the navigation attributes.

11.2.7.3. Aggregate defined at level of navigation attributes

The data in this aggregate is more summarized than an aggregate defined at the characteristic level, and performance can be better, as the navigation attributes are incorporated into the dimension tables for the aggregate. One disadvantage of aggregation at the navigation attribute level is that if there are master data changes that affect the navigation attributes in the aggregate, then it is necessary to do a ‘realignment run’ on the aggregate to reflect the new master data.

For performance, defining aggregates at the level of navigation attributes is better than the previous two alternatives. This configuration is less flexible for reporting than the previous two, as it will only support reporting at the level of the navigation attributes defined in the aggregate. An aggregate that includes a characteristic can be used by a query with any navigation attribute on the characteristic.

11.2.8. Propose aggregates in RSA1

The most important step in optimizing performance at a system level is ensuring that the correct aggregates have been defined for the infocubes and queries that are most commonly used.

If there are queries that retrieve many rows, and summarize them to a few rows as in Figure 14, optimizing DB2 to retrieve the rows more quickly will be of limited help. The root cause of performance problems in such a situation is the lack of aggregates to support the summarization needed by the end-users.

RSA1 > Modeling > Data Targets > Find > type infocube name

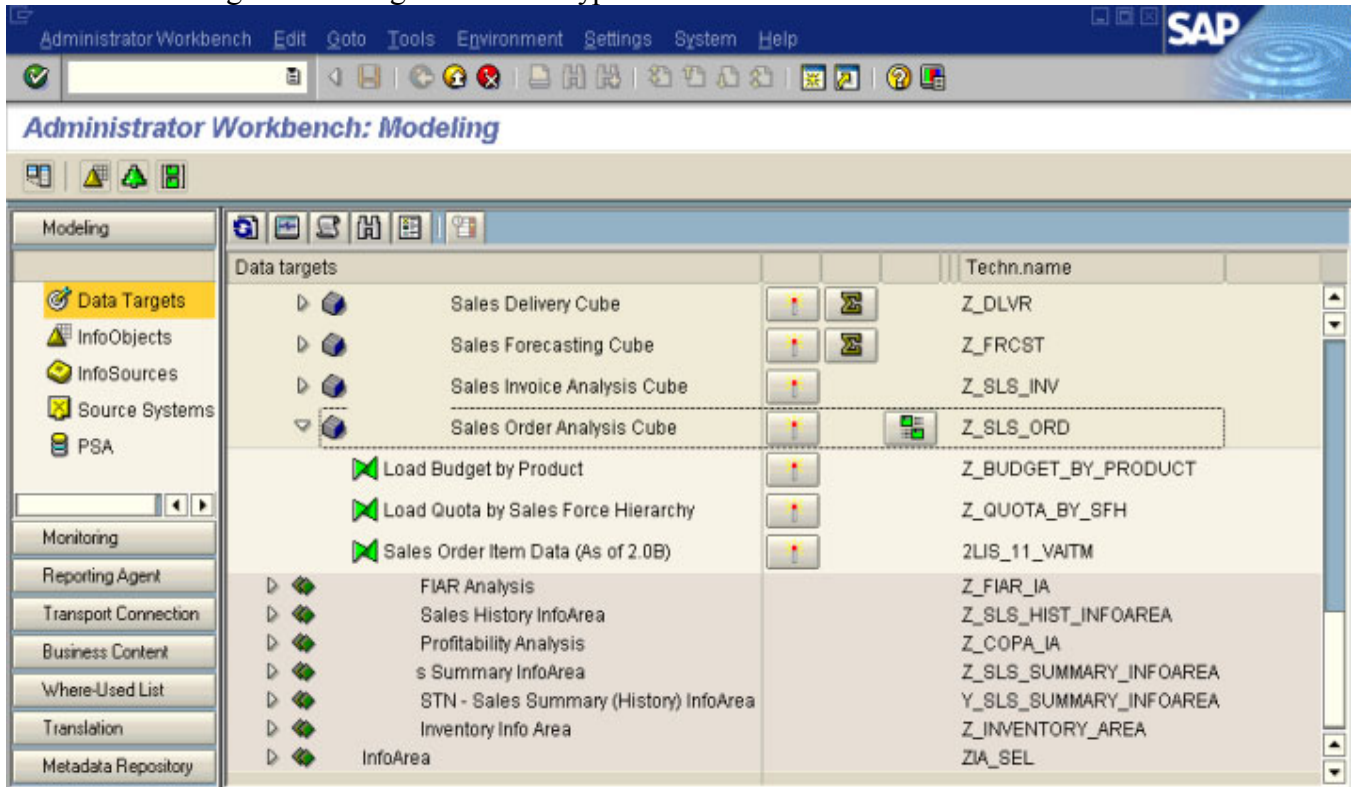


Figure 54: RSA1 modeling data targets

Highlight cube > right click > maintain aggregates

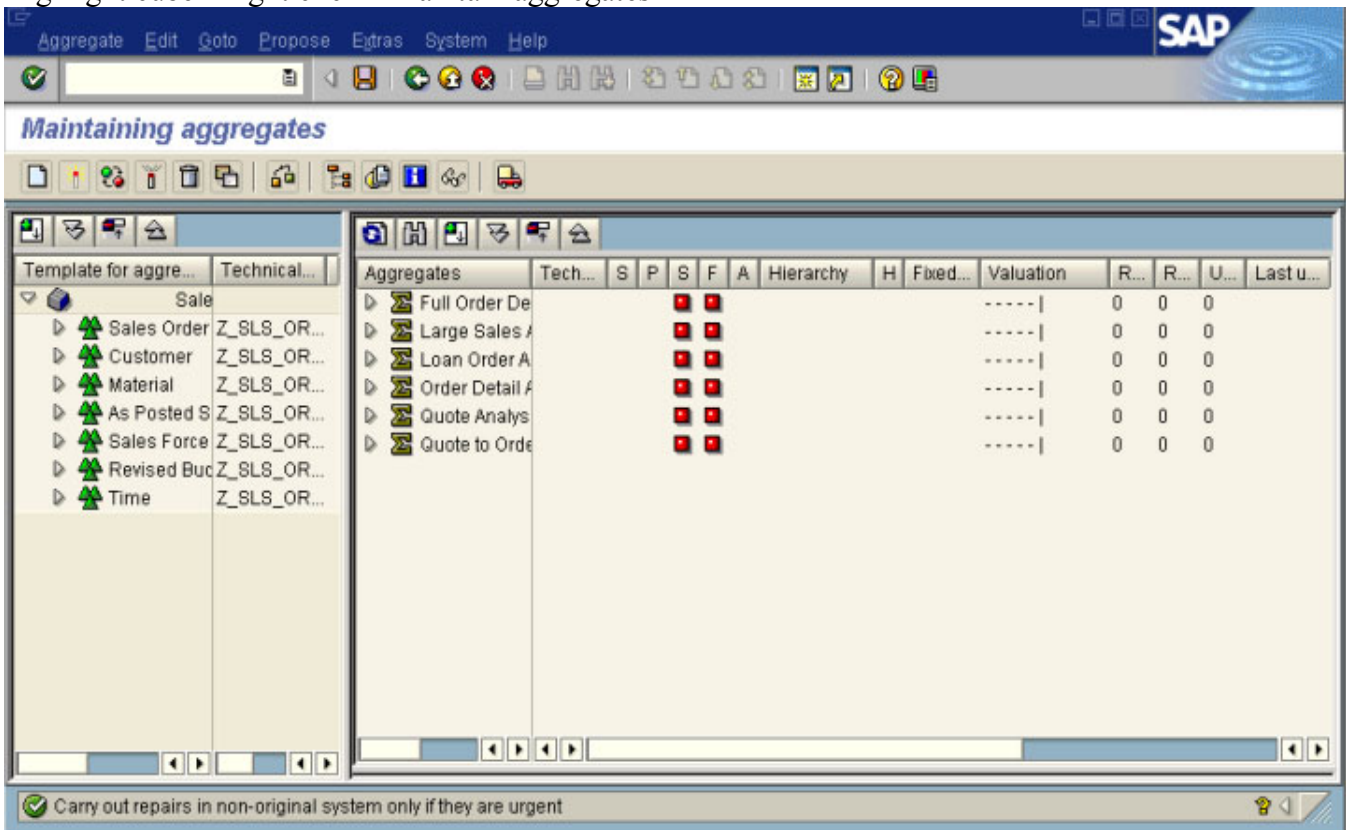


Figure 55: RSA1 maintain aggregates

Here, all the defined aggregates have been deactivated (they are red and square). The screen shot is from a test system. Normally, they would be active, and the status lights under “S” and “F” would be green (and round).

Click **propose > propose (statistics, usually query)**. Enter your time here. Though the time is stored internally in GMT, since SAP is aware of time-zones, it will convert your time entry here to GMT to match the internal data format. Type the range of queries to be analyzed.

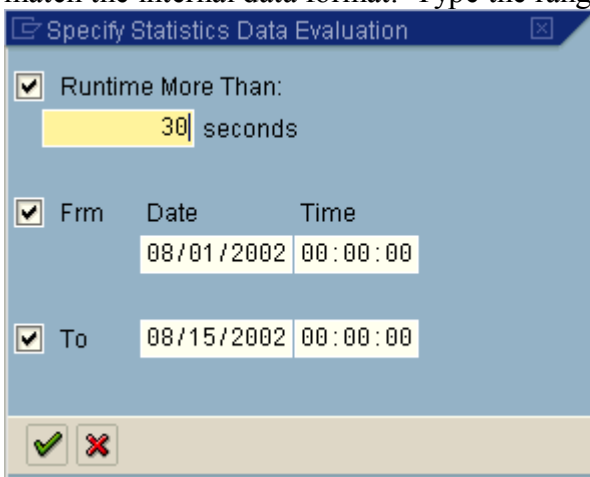


Figure 56: RSA1 parameters for aggregate proposal

The proposed list of aggregates will look like this.

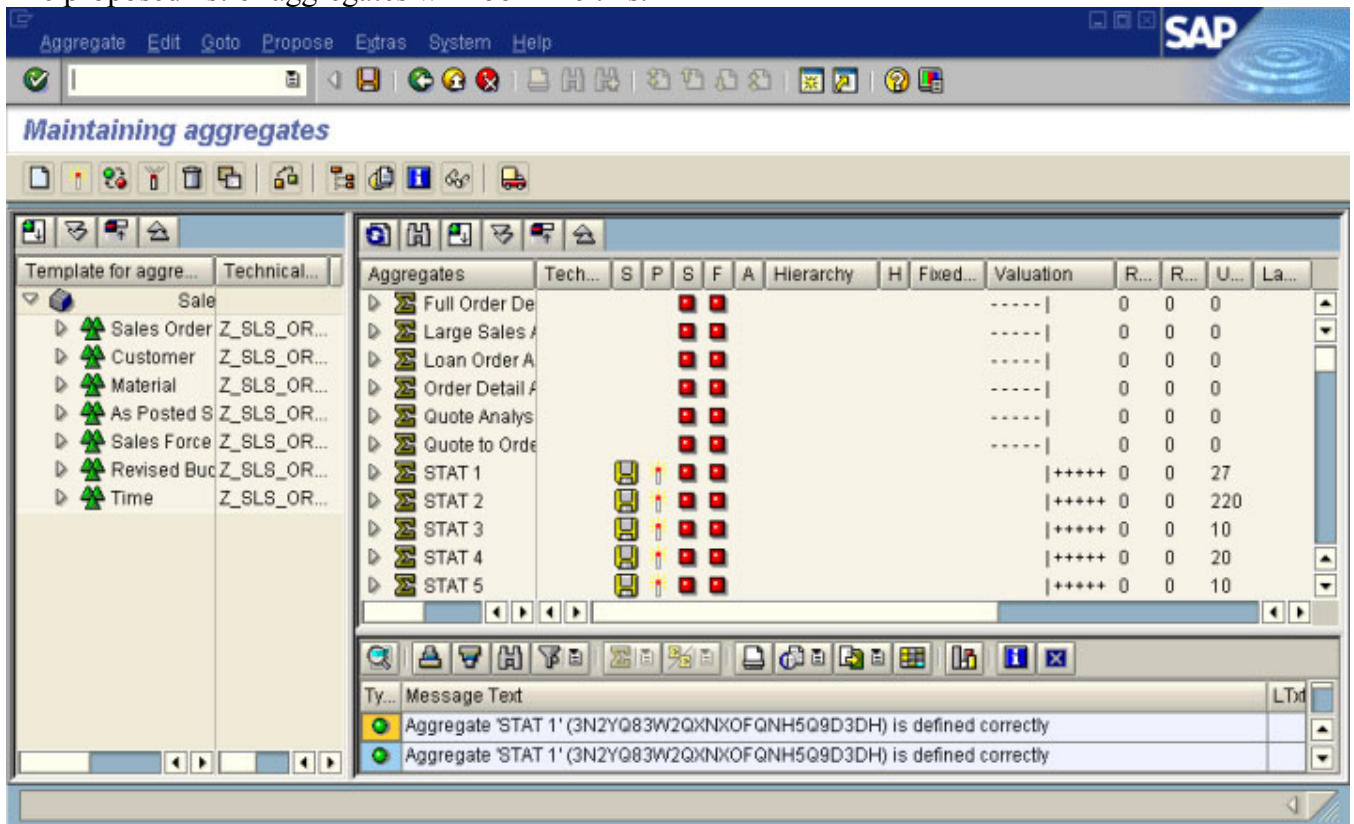
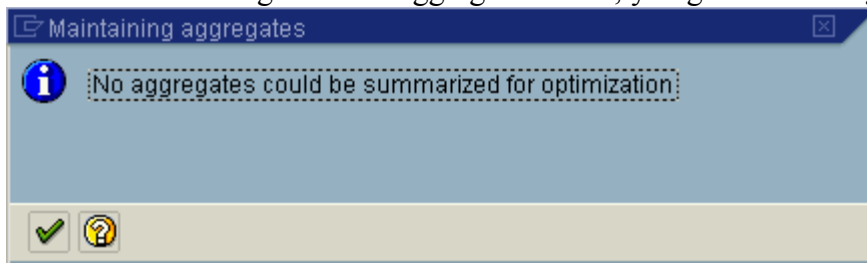


Figure 57: RSA1 proposed aggregates

For each proposed (STATx) aggregate, there is a Usage count – the number queries that could have used the aggregate, if the aggregate had existed. SAP does not, however, display information about the runtime of the queries. See section 11.2.11 for discussion about how to determine the response time impact of queries that don't use an aggregate.

After creating the list of proposed aggregates, use 'propose > optimize', to have SAP merge aggregates. When SAP has merged all the aggregates it can, you get this message.



11.2.9. Find aggregates which support many queries using roll-up hierarchy

In Figure 57, click **goto > roll-up hierarchy**. Then expand the roll-up tree. Hint: use the find button at the bottom, and search for “STAT” to expand all the hierarchies.

Roll up hierarchy for the aggregate	A...	P...	Valuation	Reco...	Reco...	Usage
Full Order Detail Aggregate	<input type="checkbox"/>		-----	0	0	0
STAT 56	<input type="checkbox"/>		+	0	0	1
STAT 53	<input type="checkbox"/>		+++	0	0	2
STAT 35	<input type="checkbox"/>		+	0	0	1
Large Sales Aggregate	<input type="checkbox"/>		-----	0	0	0
STAT 33	<input type="checkbox"/>		+...	0	0	7
Quote to Order Analysis Aggr	<input type="checkbox"/>		-----	0	0	0
Loan Order Aggregate	<input type="checkbox"/>		-----	0	0	0
Quote Analysis Aggregate	<input type="checkbox"/>		-----	0	0	0
STAT 5	<input type="checkbox"/>		+...	0	0	10
STAT 114	<input type="checkbox"/>		+++	0	0	2
STAT 4	<input type="checkbox"/>		+...	0	0	20
STAT 3	<input type="checkbox"/>		+...	0	0	10
STAT 20	<input type="checkbox"/>		+...	0	0	10
STAT 70	<input type="checkbox"/>		+...	0	0	5
STAT 18	<input type="checkbox"/>		+	0	0	1
STAT 87	<input type="checkbox"/>		+	0	0	1
STAT 183	<input type="checkbox"/>		+	0	0	1
STAT 133	<input type="checkbox"/>		++++	0	0	3
STAT 101	<input type="checkbox"/>		++++	0	0	3
STAT 154	<input type="checkbox"/>		+	0	0	1
STAT 155	<input type="checkbox"/>		+	0	0	1

Figure 58: Z_SLS_ORD roll-up hierarchy part 1

Next, look for branches (going from upper left to lower right) of the hierarchy where there is a high usage count for the proposed aggregates.

Each proposed aggregate supports queries with a unique set of characteristics. The aggregate proposed at the top left of the branch will also support queries matching proposed aggregates to the right underneath it. The proposed aggregates on the right of the branch contain some, but not all, of the characteristics in the aggregates on the left of the branch. Though the top left aggregate may not be optimal for queries matching all the proposed aggregates in its branch, it can be a starting point for evaluating aggregate definition.

In the example above, the proposed aggregate called STAT 5 could have been used by over 50 times (see hierarchy for STAT5 to STAT 3) during the analysis period, and STAT 20 would have supported about 20 queries (STAT20 to STAT 155).

Roll up hierarchy for the aggregate						
Roll up hierarchy for the aggregate	A...	P...	Valuation	Reco...	Recor...	Usage
▼ <input checked="" type="checkbox"/> STAT 21	<input type="checkbox"/>	<input type="checkbox"/>	+... 0	0	17	
<input checked="" type="checkbox"/> STAT 116	<input type="checkbox"/>	<input type="checkbox"/>	+ 0	0	1	
<input checked="" type="checkbox"/> STAT 27	<input type="checkbox"/>	<input type="checkbox"/>	+ 0	0	1	
<input checked="" type="checkbox"/> STAT 44	<input type="checkbox"/>	<input type="checkbox"/>	+... 0	0	7	
▼ <input checked="" type="checkbox"/> STAT 48	<input type="checkbox"/>	<input type="checkbox"/>	+... 0	0	6	
<input checked="" type="checkbox"/> STAT 46	<input type="checkbox"/>	<input type="checkbox"/>	+... 0	0	13	
▼ <input checked="" type="checkbox"/> STAT 125	<input type="checkbox"/>	<input type="checkbox"/>	+ 0	0	1	
▼ <input checked="" type="checkbox"/> STAT 7	<input type="checkbox"/>	<input type="checkbox"/>	+... 0	0	43	
▼ <input checked="" type="checkbox"/> STAT 57	<input type="checkbox"/>	<input type="checkbox"/>	+... 0	0	15	
▼ <input checked="" type="checkbox"/> STAT 24	<input type="checkbox"/>	<input type="checkbox"/>	+... 0	0	8	
<input checked="" type="checkbox"/> STAT 15	<input type="checkbox"/>	<input type="checkbox"/>	+... 0	0	12	
<input checked="" type="checkbox"/> STAT 23	<input type="checkbox"/>	<input type="checkbox"/>	+... 0	0	13	
<input checked="" type="checkbox"/> STAT 173	<input type="checkbox"/>	<input type="checkbox"/>	+ 0	0	1	
<input checked="" type="checkbox"/> STAT 145	<input type="checkbox"/>	<input type="checkbox"/>	+... 0	0	4	
<input checked="" type="checkbox"/> STAT 147	<input type="checkbox"/>	<input type="checkbox"/>	+ 0	0	1	
<input checked="" type="checkbox"/> STAT 157	<input type="checkbox"/>	<input type="checkbox"/>	+++ 0	0	2	
▼ <input checked="" type="checkbox"/> STAT 179	<input type="checkbox"/>	<input type="checkbox"/>	+ 0	0	1	
<input checked="" type="checkbox"/> STAT 177	<input type="checkbox"/>	<input type="checkbox"/>	+ 0	0	1	
▼ <input checked="" type="checkbox"/> STAT 52	<input type="checkbox"/>	<input type="checkbox"/>	+... 0	0	6	
▼ <input checked="" type="checkbox"/> STAT 51	<input type="checkbox"/>	<input type="checkbox"/>	+... 0	0	6	
▼ <input checked="" type="checkbox"/> STAT 50	<input type="checkbox"/>	<input type="checkbox"/>	+... 0	0	6	
▼ <input checked="" type="checkbox"/> STAT 49	<input type="checkbox"/>	<input type="checkbox"/>	+... 0	0	6	

Figure 59: Z_SLS_ORD rollup hierarchy part 2

In this example, the query called STAT 48 could be used by nearly 100 queries executed in the reporting period. The tree for STAT 48 goes down to STAT 177.

11.2.10. Evaluate proposals and determine which aggregates to create

When SAP displays two proposed aggregates that are on the same level, for example STAT 27 and STAT 44 in Figure 59, it means that each aggregate contains some characteristics that are not in the other aggregate. It may be possible to define a single aggregate that will replace two or more proposed aggregates. This might be done to reduce administration, and reduce the time to roll-up aggregates, while still giving a performance improvement via the new aggregate.

11.2.10.1. Two aggregates that should not be merged

In the following example, there are two aggregates (58 and 60) that are at the same level in the rollup hierarchy, which means that each aggregate contains one or more characteristics that the other does not have. We check the aggregate proposals, to see if it makes sense to merge them.

Roll up hierarchy for the aggregate	A...	P...	Valuation	Reco...	Recor...	Usage
▼ STAT 52	<input type="checkbox"/>		+... 0	0	6	
▼ STAT 51	<input type="checkbox"/>		+... 0	0	6	
▼ STAT 50	<input type="checkbox"/>		+... 0	0	6	
▼ STAT 49	<input type="checkbox"/>		+... 0	0	6	
STAT 180	<input type="checkbox"/>		+ 0	0	1	
STAT 66	<input type="checkbox"/>		++++ 0	0	3	
▼ STAT 105	<input type="checkbox"/>		+++ 0	0	2	
STAT 75	<input type="checkbox"/>		+ 0	0	1	
▼ STAT 152	<input type="checkbox"/>		+ 0	0	1	
STAT 31	<input type="checkbox"/>		+ 0	0	1	
STAT 78	<input type="checkbox"/>		+... 0	0	9	
STAT 54	<input type="checkbox"/>		+ 0	0	1	
STAT 55	<input type="checkbox"/>		+ 0	0	1	
▼ STAT 58	<input type="checkbox"/>		+ 0	0	1	
▼ STAT 169	<input type="checkbox"/>		+ 0	0	1	
▼ STAT 129	<input type="checkbox"/>		+++ 0	0	2	
▼ STAT 128	<input type="checkbox"/>		+ 0	0	1	
STAT 164	<input type="checkbox"/>		+ 0	0	1	
STAT 158	<input type="checkbox"/>		+++ 0	0	2	
▶ STAT 60	<input type="checkbox"/>		+... 0	0	5	
STAT 64	<input type="checkbox"/>		+... 0	0	7	
▶ STAT 67	<input type="checkbox"/>		+... 0	0	9	

Figure 60: Z_SLS_ORD roll up hierarchy part 3

Leave roll-up hierarchy, and expand the STAT 58 and STAT 60 aggregates in the 'propose aggregates' screen.

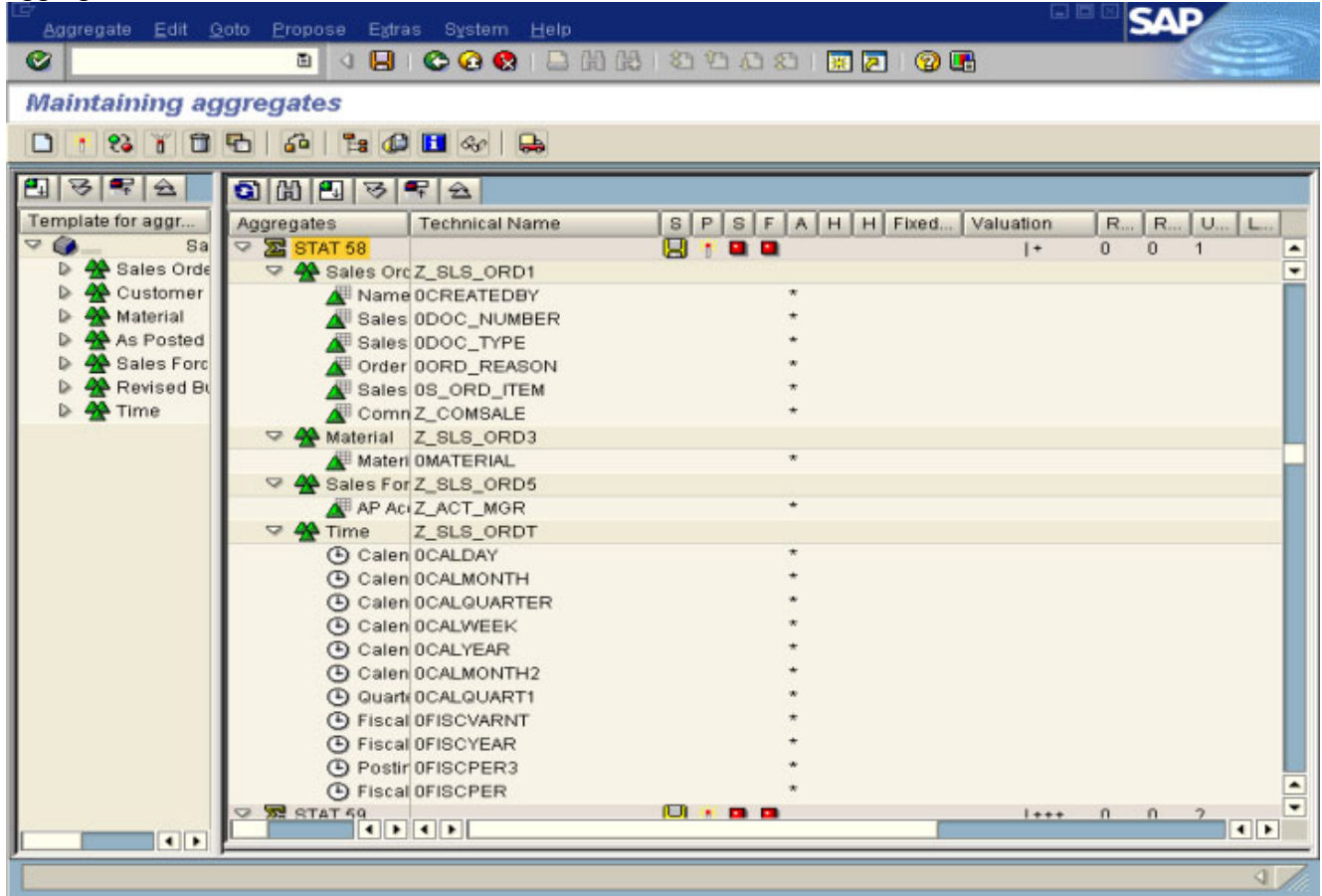


Figure 61: Evaluate aggregate merge - STAT 58 - dimensions 1, 3, 5, and T

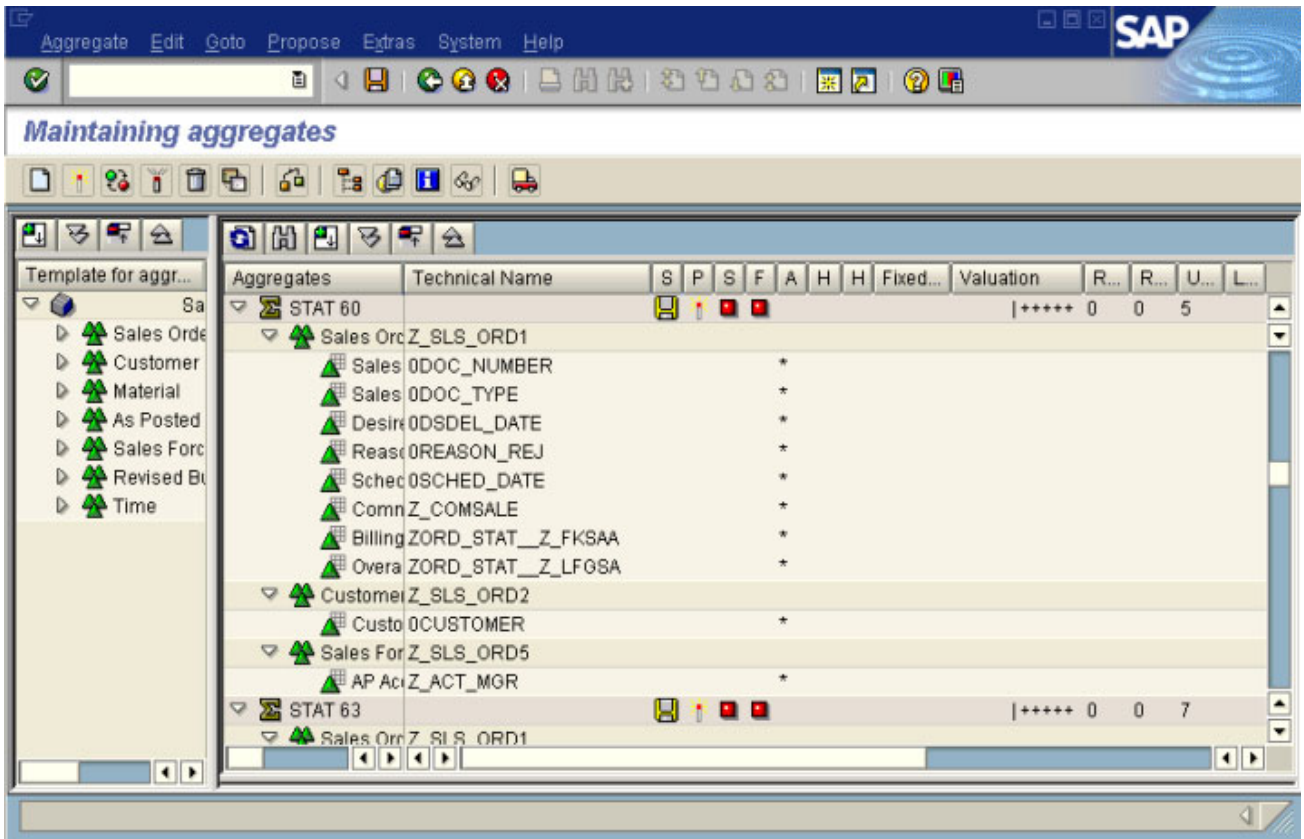


Figure 62: Evaluate aggregate merge - STAT 60 - dimensions 1, 2, 5

STAT 58 reports on dimensions 1, 3, 5 and T. That is, it reports on sales (1) per material (3) per sales manager (5) over a time (T). STAT 60 reports on dimensions 1, 2, and 5. That is, it reports on sales (1) per customer (2) per sales manager (5). Since these two aggregates have different reporting goals and access different dimensions, we probably would not merge them to create a single aggregate to cover both types of query.

11.2.10.2. Two aggregates which can be merged

Again, this example uses two aggregates that are on the same level in their roll-up hierarchy – STAT 3 and STAT 4. The roll-up hierarchy is not shown here. We check the definition of each proposed aggregate.

First, a note on the meaning of aggregation levels in the proposals.

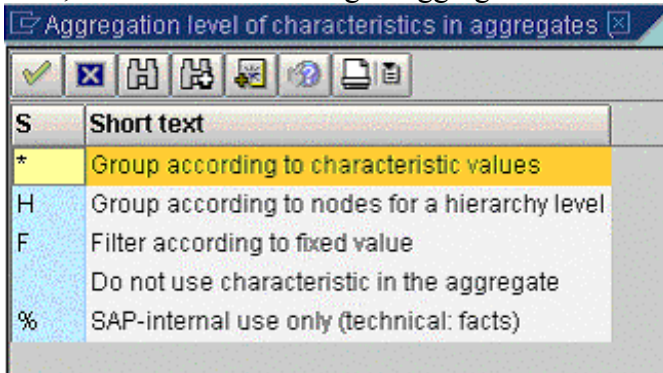


Figure 63: Definition of aggregation levels

F means that the user specified one or more fixed values for reporting, as when a single sales area or sales person is selected. When creating an aggregate, we would often convert the 'F' to '*', so that the aggregate would support selection of any possible value for the characteristic.

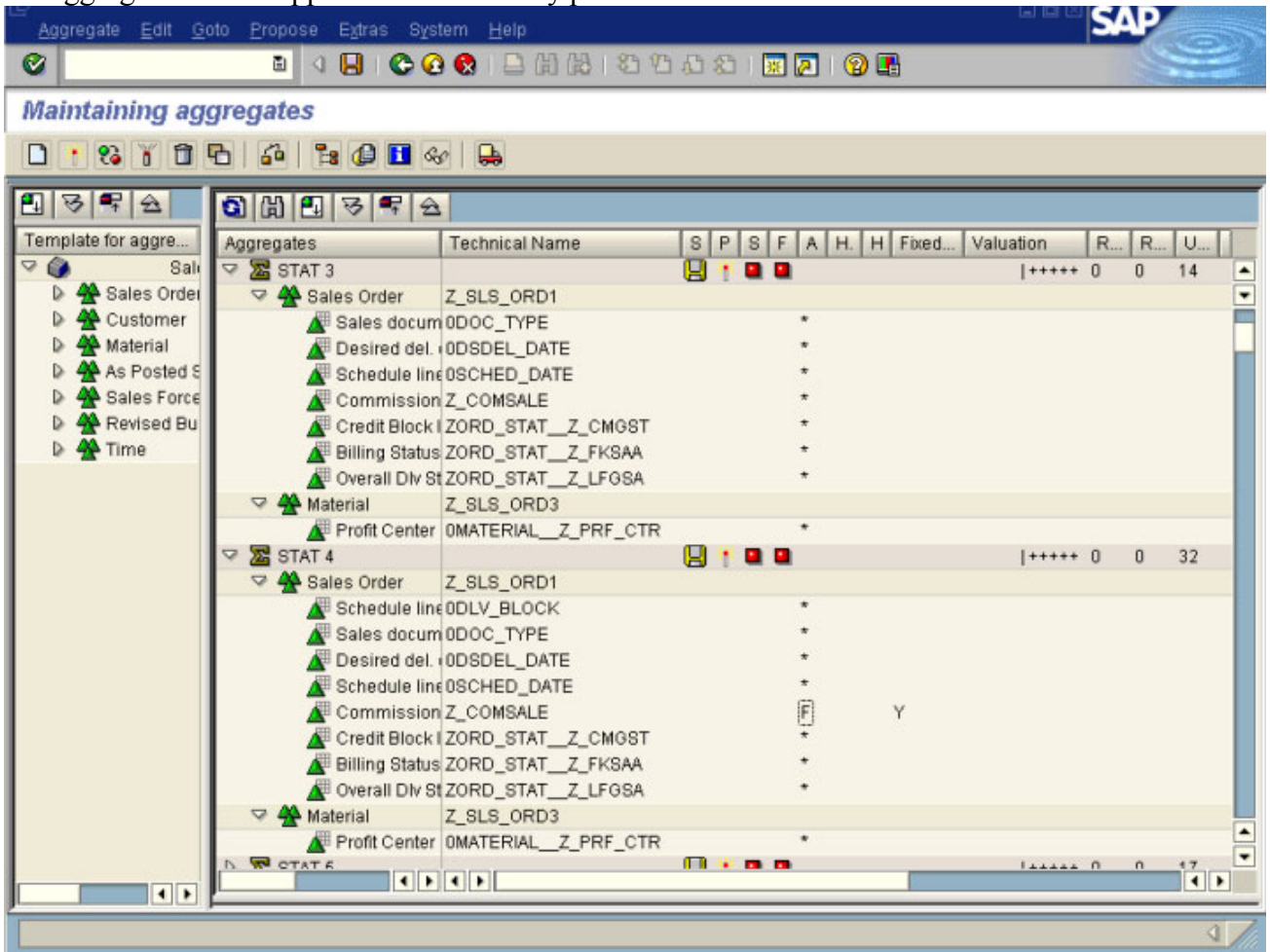


Figure 64: STAT 3 and STAT 4 comparison for merge

STAT 3 and STAT 4 both contain the same dimensions (Sales Order and Material, which are dimensions 1 and 3). STAT 3 contains Z_COMSALE with "*" aggregation for all values. This is

broader than the STAT 4 definition of Z_COMSALE with “F” aggregation for a specific value. STAT 4 contains 0DLV_BLOCK, which is not in STAT 3. One can create an single aggregate to replace the two proposed aggregates by taking the characteristics that are in both STAT 3 and STAT 4, and adding Z_COMSALE with “*” aggregation, and 0DLV_BLOCK with “*” aggregation.

11.2.10.3. Aggregate hierarchies

As shown above, the roll-up hierarchy can show many different levels of aggregates. The proposed aggregates on the lower right contain fewer characteristics than the aggregates on the upper left. In the case where there are many levels, one can choose several points in the branch and make several aggregates to support the branch of proposed aggregates.

In the following examples, STAT 191 was taken from the upper-left of an aggregate tree, STAT 178 in the middle, and STAT 154 from the lower-right. Upper-left is less aggregated (that is, it contains more characteristics for selection) and lower-right is more aggregated (it contains fewer characteristics for selection). Since the entire tree covers several screens, here is the lower-right of the tree.

Roll up hierarchy for the aggregate	A...	P	Valuation	F	F	U...
STAT 11	<input type="checkbox"/>		+ ... 0 0 23			
STAT 5	<input type="checkbox"/>		+ ... 0 0 18			
STAT 51	<input type="checkbox"/>		+ ... 0 0 8			
STAT 154	<input type="checkbox"/>		+ ... 0 0 6			
STAT 2	<input type="checkbox"/>		+ ... 0 0 76			
STAT 34	<input type="checkbox"/>		+ ... 0 0 29			
STAT 25	<input type="checkbox"/>		+ ... 0 0 33			
STAT 168	<input type="checkbox"/>		+ 0 0 1			
STAT 8	<input type="checkbox"/>		+ ... 0 0 130			
	<input type="checkbox"/>		+ ... 0 0 4			
	<input type="checkbox"/>		+ 0 0 1			
	<input type="checkbox"/>		+ 0 0 1			
	<input type="checkbox"/>		+ 0 0 1			
	<input type="checkbox"/>		+ 0 0 1			
	<input type="checkbox"/>		+ 0 0 1			
	<input type="checkbox"/>		+ 0 0 1			
	<input type="checkbox"/>		+ 0 0 1			
	<input type="checkbox"/>		+++ 0 0 2			
	<input type="checkbox"/>		+++ 0 0 2			
	<input type="checkbox"/>		+ 0 0 1			

Figure 65: Aggregate hierarchy

Leave roll-up hierarchy and return to aggregate proposals, and check the definition of STAT 191, which was listed on the upper left of a hierarchy.

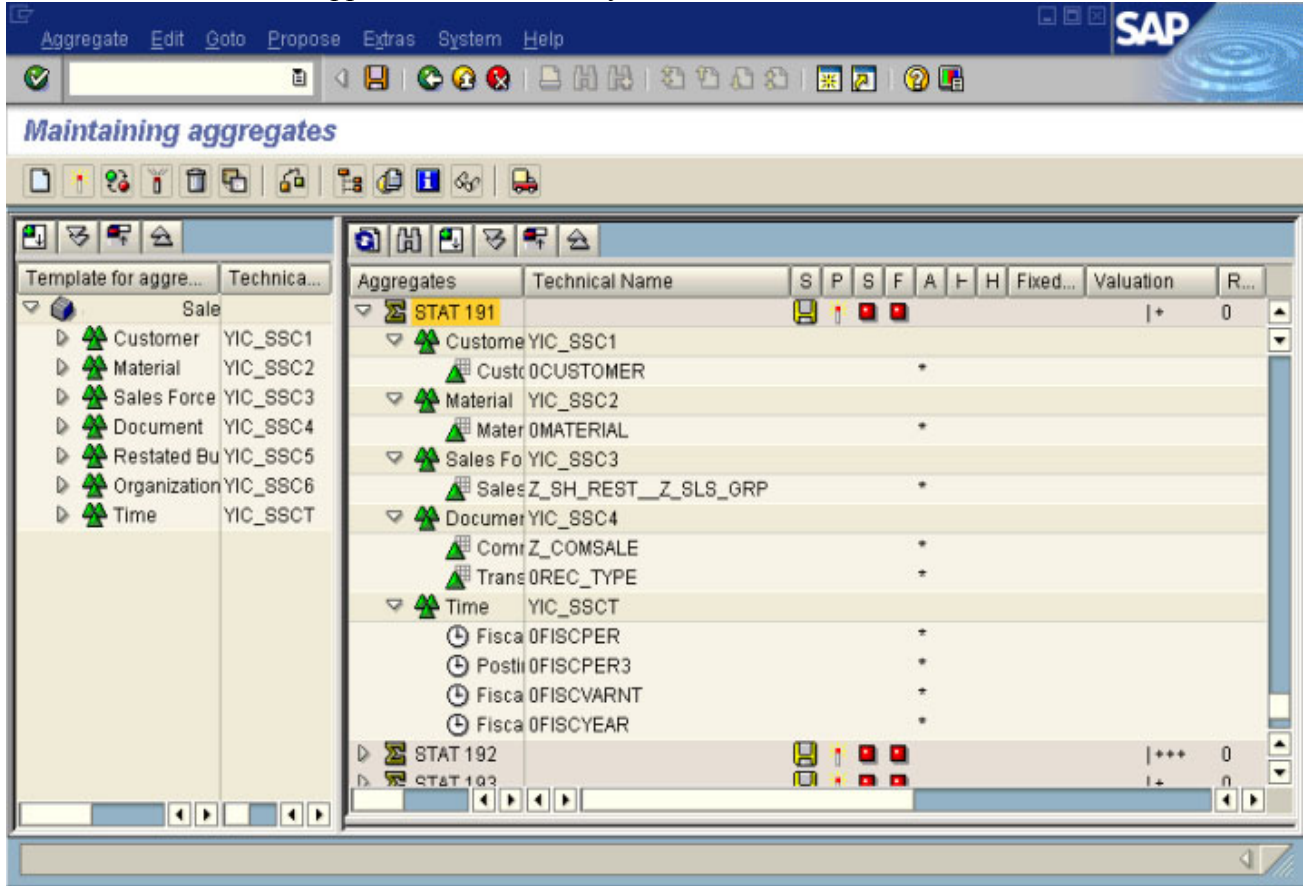


Figure 66: Deep roll-up hierarchy STAT 191

Proposed aggregate STAT 191 has 5 dimensions.

Next, check the definition of STAT 178.

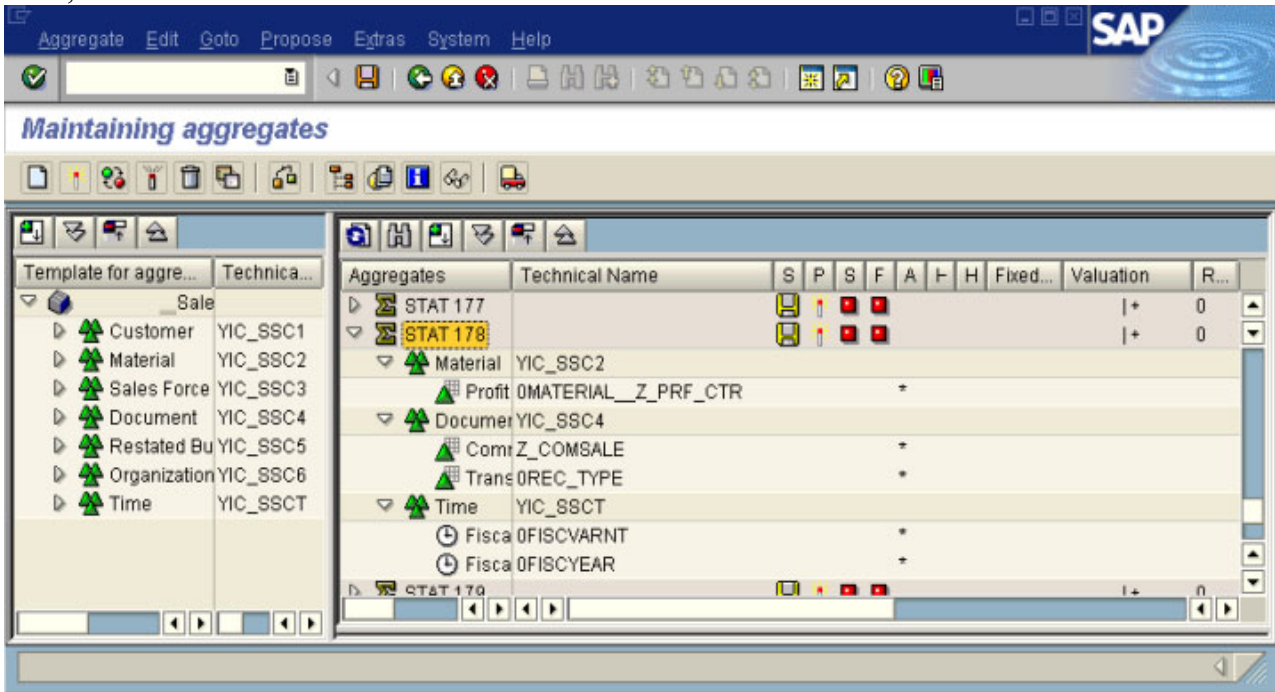


Figure 67: Deep roll-up hierarchy STAT 178

Proposed aggregate STAT 178 has three dimensions. It has fewer characteristics and dimensions than STAT 191. (Material is selectable only by navigation attribute 0MATERIAL__Z_PRF_CTR in 178, while it can be reported at any level in 191, which contains 0MATERIAL).

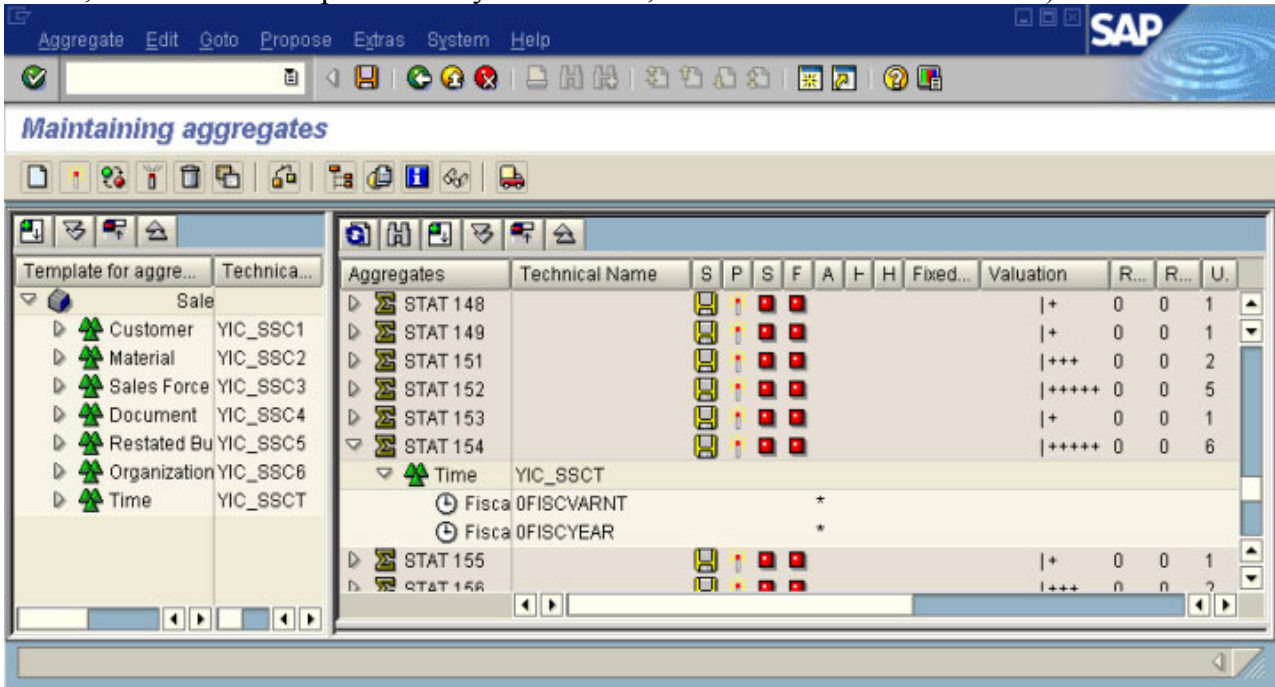


Figure 68: Deep roll-up hierarchy STAT 154

Proposed aggregate STAT 154 has one dimension. It has fewer characteristics and dimensions than STAT 178. The data in this aggregate will be very summarized, but the aggregate will only support a narrow range of queries.

For situations where there are queries with different levels of selection and summarization, one may want to create a hierarchy of aggregates, to support efficient reporting with queries referencing more or fewer dimensions and characteristics.

11.2.11. An RYO solution

As mentioned above, one limitation of the SAP aggregate proposal tool is that it shows *counts* of queries that can use an aggregate, but does not show the *total elapsed time* of the queries. Since you may want to focus on commonly executed queries (high count) and long running queries (long elapsed time) additional data is needed to determine the characteristics of queries with long elapsed time.

One can write an ABAP program that joins RSDDSTAT (which contains runtime and row counts) with RSDDSTATAGGRDEF (which contains characteristics) on STATUID and create a report that summarizes the characteristic combinations used in queries. The program would sum QTIMEDB for all queries that use identical characteristics. Each row in the report below lists the characteristics used, as well as the frequency and total elapsed time of queries using each unique set of characteristics.

Here is a sample of a report created by one company running BW. With such a report, one can find combinations of characteristics and navigation attributes used in frequently executed or long running queries. One can also group the queries based on the characteristics used, to find “families” of queries using similar characteristics.

By joining RSDDSTAT and RRSDDSTATAGGRDEF, one can associate the total elapsed time with the characteristic combinations, to more easily determine aggregates that will support long running queries.

Query	TotalDBTime	Frequency	AverageDI	AverageDI	AverageDI	Querycube	Objnm1	Objnm2
DWNAISHPO/DWNAISH	19029.0	33	576.6	78	77	DWNAISHPO	0CALMONTH	ADWNAC026_ADWNAC027
DWNAISHPO/DWNAISH	13722.5	21	653.5	34,039	33,160	DWNAISHPO	0CALMONTH	ADWNAC058
DWNAISHPO/DWNAISH	5914.4	8	739.3	1,599	189	DWNAISHPO	0CALMONTH	ADWNAC026_ADWNAC027
DWNAISHPO/DWNAISH	5767.5	3	1922.5	22,984	180	DWNAISHPO	0CALQUARTER	ADWNAC026_ADWNAC032
DWNAISHPO/DWNAISH	5463.5	9	607.1	28,673	739	DWNAISHPO	0CALMONTH	ADWNAC060
DWNAISHPO/DWNAISH	5086.3	18	282.6	35,854	36	DWNAISHPO	0CALMONTH	ADWNAC026_ADWNAC033
DWNAISHPO/DWNAISH	5013.7	5	1002.7	44,003	16,969	DWNAISHPO	0CALMONTH	ADWNAC026_ADWNAC027
DWNAISHPO/DWNAISH	4728.6	16	295.5	12,701	31	DWNAISHPO	0CALMONTH	ADWNAC058
PUKCINOE1/PUKCINOE	4490.0	11	408.2	2,046	1,573	PUKCINOE1	GPUAPTOBY_GPUDEMON	GPUKCMILL
DWNAISHPO/DWNAISH	4216.2	9	468.5	9,917	9,858	DWNAISHPO	0CALMONTH	ADWNAC026_ADWNAC027
DWNAISHPO/DWNAISH	3985.7	4	996.4	3,055	2,942	DWNAISHPO	0CALMONTH	ADWNAC026_ADWNAC027
DWNAISHPO/DWNAISH	3796.8	6	632.8	12,652	12,614	DWNAISHPO	0CALMONTH	ADWNAC026_ADWNAC027
DWNAISHPO/DWNAISH	3614.0	7	516.3	181,439	16,309	DWNAISHPO	0CALDAY	0CALMONTH
DWNAISHPO/DWNAISH	3563.1	14	254.5	5,731	5,717	DWNAISHPO	0CALMONTH	ADWNAC026_ADWNAC027
DWNAISHPO/DWNAISH	3354.3	5	670.9	8,893	3,403	DWNAISHPO	0CALMONTH	ADWNAC046
DWNAISHPO/DWNAISH	2712.4	4	678.1	15,028	938	DWNAISHPO	0CALDAY	0CALMONTH
PUKCIPO01/CAD_TEST	2695.0	15	179.7	102,114	28,849	PUKCIPO01	GPUMILMAT_GPUIITYPE	GPUPOPOID
DWNAISHPO/DWNAISH	2685.7	2	1342.8	437,354	67,485	DWNAISHPO	0CALMONTH	ADWNAC111
DWNAISHPO/DWNAISH	2645.6	1	2645.6	2,717,633	4,044	DWNAISHPO	0CALMONTH	ADWNAC097
DWNAISHPO/DWNAISH	2617.9	10	261.8	1,814	1,813	DWNAISHPO	0CALMONTH	ADWNAC046
USCPCOLL/USCPCSICF	2599.0	62	41.9	442	442	100097	0CALDAY	UCPDVLV
DWNAISHPO/DWNAISH	2480.0	2	1240.0	51,996	1,137	DWNAISHPO	0CALMONTH	ADWNAC058

Figure 69: Custom report from RSDDSTAT RRSDDSTATAGGRDEF join

11.2.12. Sample tools for aggregate proposal

The method described in Section 11.2.11, where RSDDSTAT and RSDDSTATAGGRDEF are joined to find high-impact characteristic combinations is implemented in perl scripts that are available from the IBM techdocs web site.

Techdocs document PRS713, which is available at <http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS713> has sample programs which can be used to find high-impact characteristic combinations.

11.2.13. View on RSDDSTAT and RSDDSTATAGGRDEF

In order to be able to examine the characteristics of individual long running queries found in RSDDSTAT, it can be useful to create a view in SAP that joins RSDDSTAT and RSDDSTATAGGRDEF on STATUID.

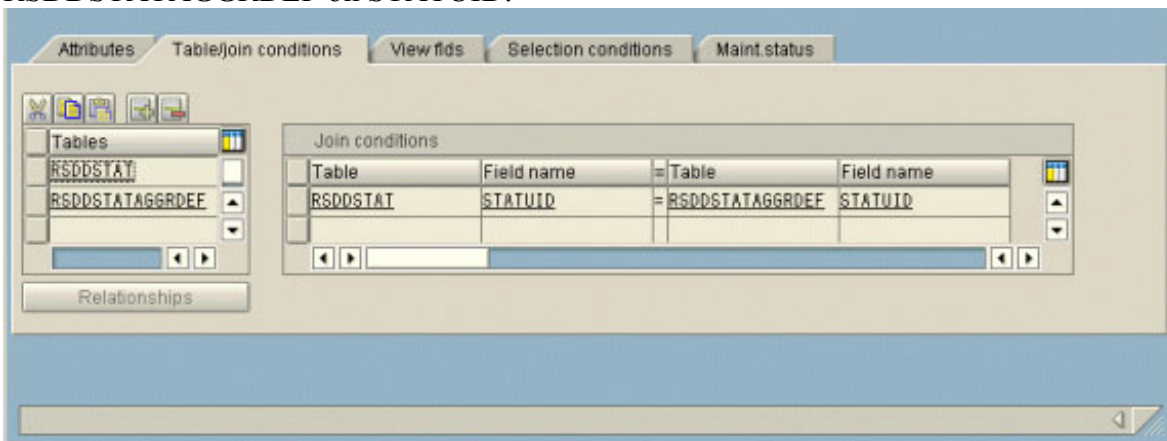


Figure 70: View joining RSDDSTAT and RSDDSTATAGGRDEF

Not all the columns in RSDDSTAT are needed in the view. In order to review the characteristics used in long running queries, include all the columns in RSDDSTATAGGRDEF, and the columns in RSDDSTAT related to DB performance (QTIMEDB, QDBSEL, QDBTRANS) along with columns identifying the infocube, query, user, etc.

View field	M...	DType	Length	Short description
STATUID	<input type="checkbox"/>	CHAR	25	UUID in compressed form
QDBSEL	<input type="checkbox"/>	DEC	15	BW Statistics: Number
QDBTRANS	<input type="checkbox"/>	DEC	15	BW Statistics: Number
INFOCUBE	<input type="checkbox"/>	CHAR	30	InfoCube
QTIMEDB	<input type="checkbox"/>	DEC	21	Runtime
STARTTIME	<input type="checkbox"/>	DEC	21	UTC time stamp in long form (YYYYMMDDhhmmss,mmmuun)
QUERYID	<input type="checkbox"/>	CHAR	25	Internal display of the report identifier
AGGRNUM	<input type="checkbox"/>	NUMC	4	BW: General reference field for internal IDs
IOBJNM	<input type="checkbox"/>	CHAR	30	InfoObject
QUERYPAGE	<input type="checkbox"/>	CHAR	30	InfoCube
AGGRST	<input type="checkbox"/>	CHAR	1	Aggregation level of characteristics in aggregates
HIESID	<input type="checkbox"/>	INT4	10	Master data ID
TLEVEL	<input type="checkbox"/>	NUMC	2	Level of a hierarchy node
VALUE	<input type="checkbox"/>	CHAR	60	Dim: Field for a user-defined characteristic value
FIXSID	<input type="checkbox"/>	INT4	10	Master data ID

Figure 71: columns in RSDDSTAT RSDDSTATAGGRDEF view

One can then use the SE16 data browser, to review the characteristics used in long running queries.

Field	Criteria	Navigation
STATUID		to [] [] []
QDBSEL		to [] [] []
QDBTRANS		to [] [] []
INFOCUBE	eudfrgact	to [] [] []
QTIMEDB	> 2400	to [] [] []
STARTTIME	...0,000,000.0000000 to ...5,000,000.0000000	to [] [] []
QUERYID		to [] [] []
AGGRNUM		to [] [] []
IOBJNM		to [] [] []
QUERYPAGE		to [] [] []
AGGRST	* []	to [] [] []
HIESID		to [] [] []
TLEVEL		to [] [] []
VALUE		to [] [] []
FIXSID		to [] [] []

Figure 72: SE16 using RSDDSTAT/RSDDSTATAGGRDEF view

In Figure 72, we select queries over 2400 seconds, on a single infocube, within a date range. AGGRST NE ‘ ‘ is specified to exclude key figures from the report. Each STATUID is a separate navigation step. In Figure 73, each navigation step performs single query (AGGRNUM is ‘ ‘).

This process might be used when ST03N shows long database time on an infocube or query, or when there are user complaints about response time using a query.

STATUID	QDBSEL	QDBTRANS	INFOCUBE	QTIMEDB	QUERYID	AGGRNUM	QUERYCU...	IJOBNM	AGGRST
3NV0D0R...	3,278	1,489	EUDFRGACT	2,749.246093	EUDFRGACT/ZQ_PCIXJ...		EUDFRGACT	0CALMONTH	*
	3,278	1,489		2,749.246093			EUDFRGACT	0CALWEEK	*
	3,278	1,489		2,749.246093			EUDFRGACT	0CALYEAR	*
	3,278	1,489		2,749.246093			EUDFRGACT	0COMP_CODE_0COUNTRY	F
	3,278	1,489		2,749.246093			EUDFRGACT	0CUST_SALES__CUST_H01	F
	3,278	1,489		2,749.246093			EUDFRGACT	0CUST_SALES__CUST_H05	F
	3,278	1,489		2,749.246093			EUDFRGACT	0MATERIAL	*
	3,278	1,489		2,749.246093			EUDFRGACT	0SOURSYSTEM	F
3NV300Y...	23,836	682		2,451.167969			EUDFRGACT	0CALMONTH	*
	23,836	682		2,451.167969			EUDFRGACT	0CALWEEK	*
	23,836	682		2,451.167969			EUDFRGACT	0CALYEAR	*
	23,836	682		2,451.167969			EUDFRGACT	0COMP_CODE_0COUNTRY	F
	23,836	682		2,451.167969			EUDFRGACT	0MATERIAL	*
	23,836	682		2,451.167969			EUDFRGACT	0MAT_SALES__SUBBRAND	F
	23,836	682		2,451.167969			EUDFRGACT	0SALESORG	F
	23,836	682		2,451.167969			EUDFRGACT	0SOURSYSTEM	F
3NXG9MC...	31,143	58		3,314.222657			EUDFRGACT	0CALMONTH	*
	31,143	58		3,314.222657			EUDFRGACT	0CALWEEK	*
	31,143	58		3,314.222657			EUDFRGACT	0CALYEAR	*
	31,143	58		3,314.222657			EUDFRGACT	0COMP_CODE_0COUNTRY	F

Figure 73: SE16 display with RSDSTAT/RSDSTATAGGRDEF view

Figure 73 shows this summarized view of long queries – the time, rows, characteristics, and whether an aggregate was used, or the infocube was used. One can then compare these long running queries to the defined aggregates, to determine whether existing aggregates can be extended new aggregates defined.

Here, two of the navigation steps may be aggregation candidates (QDBSEL/QDBTRANS > 10), but the QDBSEL/QTIMEDB ratio for all the queries is very low (between 1 and 10) so there are probably other performance problems, such as I/O constraint, missing indexes, bad access path choice, etc.

11.2.14. Proposing an aggregate for an individual query

If examination of ST03N or RSDDSTAT shows that a specific query needs an aggregate, use the “propose (propose from query) option.

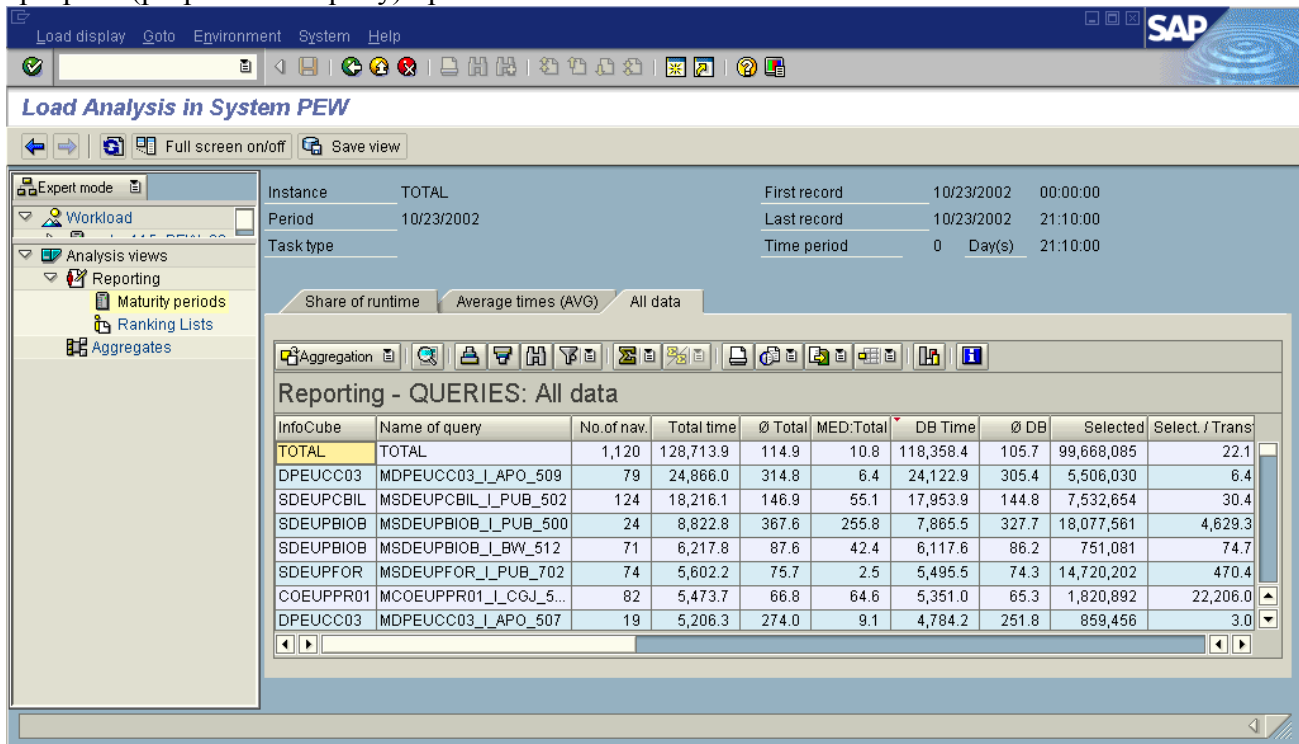
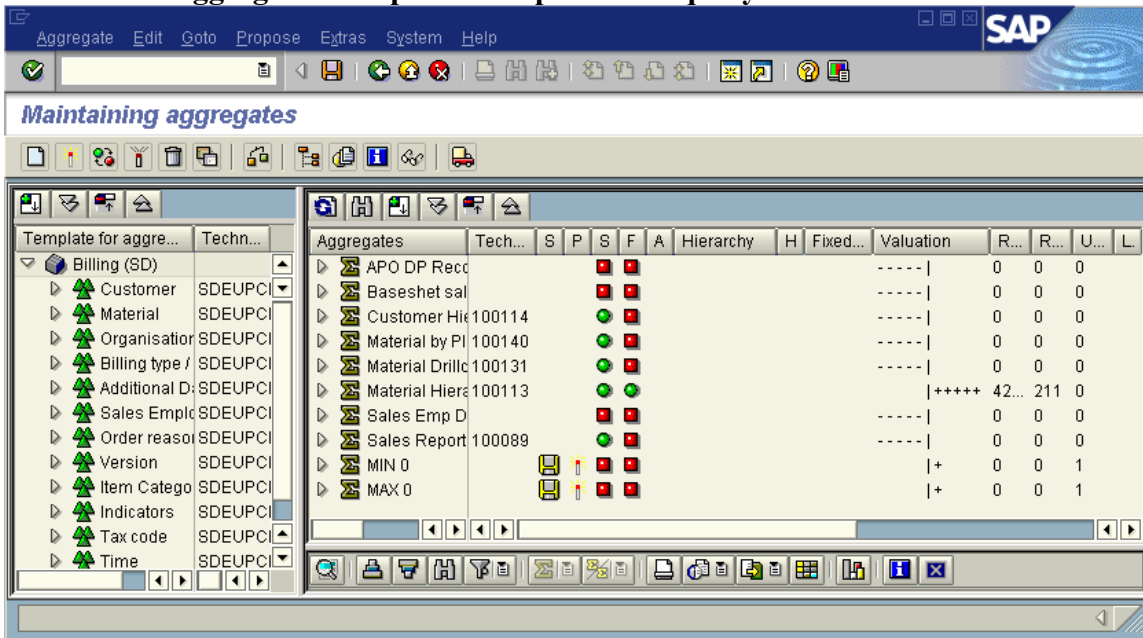


Figure 74: ST03N - query “MSDEUPBIOB_I_PUB_500” with high select / trans ratio

When we find a query with a high ratio for rows selected to rows transferred, use RSA1 to propose aggregates for the query. MSDEUPBIOB_I_PUB_500 selects 4,629 rows for every row transferred. An aggregate would improve performance.

From RSA1 aggregates: Propose > Propose from query



SAP will propose two queries. MIN is the minimum number of characteristics the query can contain. That is, it is the most aggregated data that the query can use. MAX is the maximum number of characteristics the query can contain. That is, it is the least aggregated data that the query can use. If a query has many free characteristics, then the MAX query proposal may create an aggregate nearly as large as the original infocube. One can examine the most frequently used characteristic combinations, as shown above, to refine the aggregate definition.

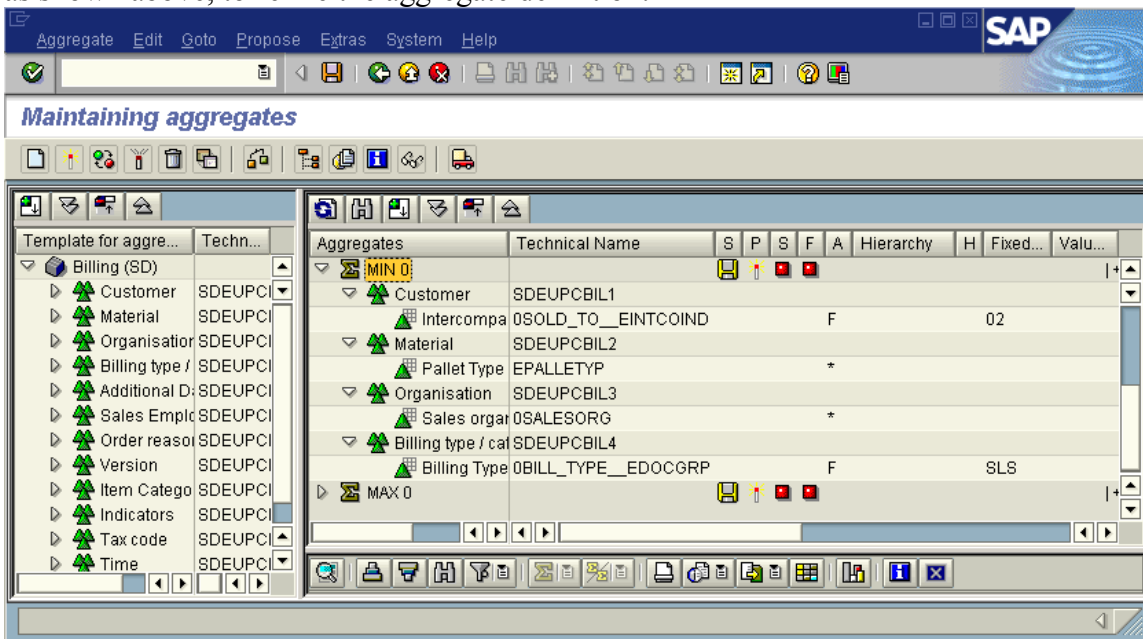
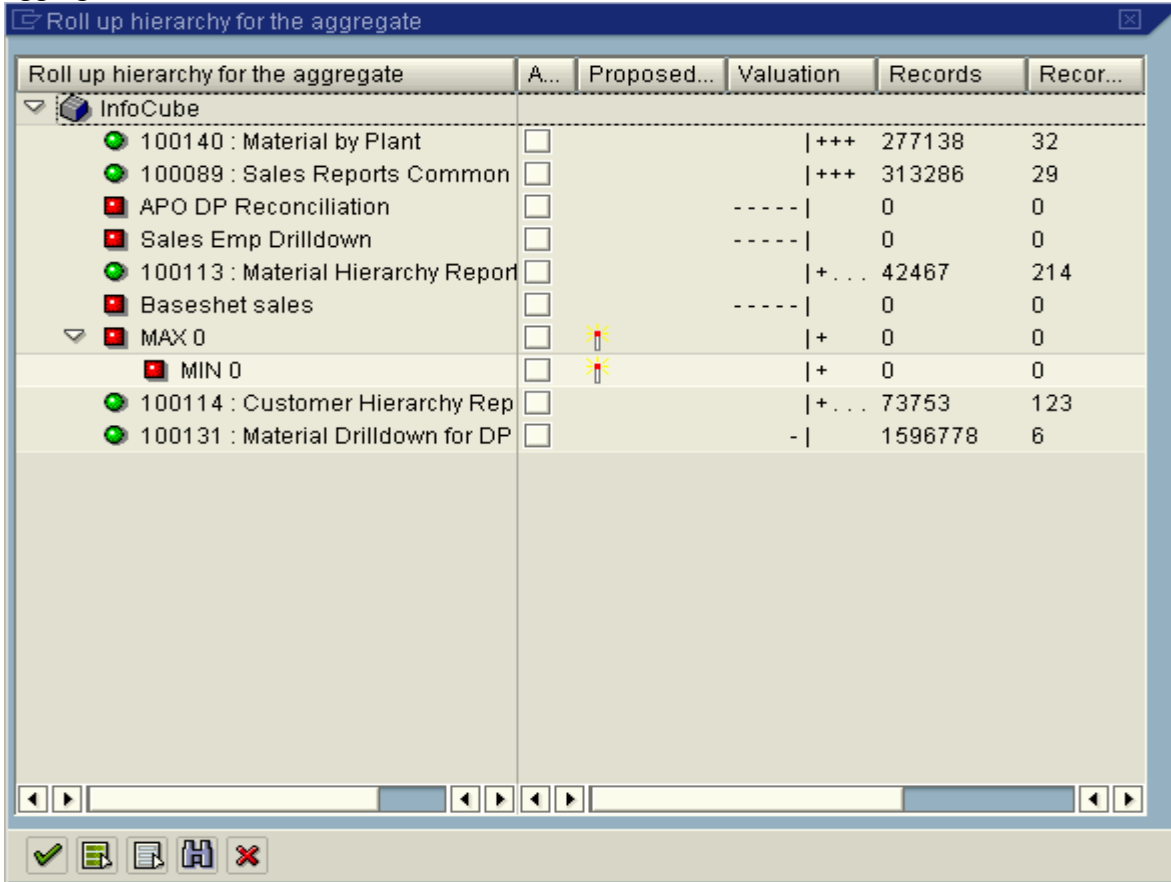


Figure 75: Sample MIN 0 proposed aggregate

Use the rollup hierarchy to determine where MIN and MAX fit in relationship to currently existing aggregates.





Roll up hierarchy for the aggregate	A...	Proposed...	Valuation	Records	Recor...
InfoCube					
100140 : Material by Plant	<input type="checkbox"/>		+++	277138	32
100089 : Sales Reports Common	<input type="checkbox"/>		+++	313286	29
APO DP Reconciliation	<input type="checkbox"/>		----	0	0
Sales Emp Drilldown	<input type="checkbox"/>		----	0	0
100113 : Material Hierarchy Report	<input type="checkbox"/>		+...	42467	214
Baseshet sales	<input type="checkbox"/>		----	0	0
MAX 0	<input type="checkbox"/>		+	0	0
MIN 0	<input type="checkbox"/>		+	0	0
100114 : Customer Hierarchy Rep	<input type="checkbox"/>		+...	73753	123
100131 : Material Drilldown for DP	<input type="checkbox"/>		-	1596778	6

Figure 76: Rollup hierarchy of proposed aggregate

In Figure 76, the query that was used for proposal is not supported by any of the active aggregates – MAX 0 is on the same hierarchy level as the other aggregates.

11.3. Review SQL cache

Since QDBSEL information is not available in ST04, when starting a performance examination from ST04 statement cache, sort the statement cache by total elapsed time, by total getpages, or total rows examined. If statements are found that have

- few rows processed per second,
- many rows examined per row processed, or
- many getpages per row processed,

then look for the corresponding RSDDSTAT data on the query. Without the QDBSEL information from RSDDSTAT, it is difficult to say if the statement is efficient or inefficient. If QDBSEL is much larger than 'rows processed', then an aggregate may be helpful.

With only the ST04 cache information, one can explain the statement, and check for missing indexes, catalog statistics which are out of date, and other DB2 related problems.

11.4. Further actions after aggregates are defined

After the query workload has been analyzed and aggregates have been created, there may still be some queries that perform slowly, that is have a low QDBSEL/QTIMEDB ratio. In these cases, there can be several different causes, the most common of which are:

- Missing indexes on master data or dimension tables
- Skewed data in dimensions, master data, and fact tables
- Correlated characteristics in queries
- I/O constraints and buffer pool constraints

In the case of missing indexes, skewed data, or correlated characteristics, the default RUNSTATS options used with BW may not be sufficient to gather enough information for DB2 to choose the right access path.

11.4.1. OLAP cache to offload database server

With 3.0, SAP provides an OLAP cache, which can be used to buffer query results, so that they can be retrieved without accessing the database server. Queries that are the same (e.g. same characteristics and same parameters) can be reused.

There is not an easy way to determine the effectiveness of the OLAP cache, that is the OLAP cache hit rate. While the OLAP cache reports the number of times a query result was reused, this cannot be easily correlated with the total executions of the query, to determine the percent of queries that have been offloaded.

From RSRT, press the 'Cache Monitor' button, to see query statistics.

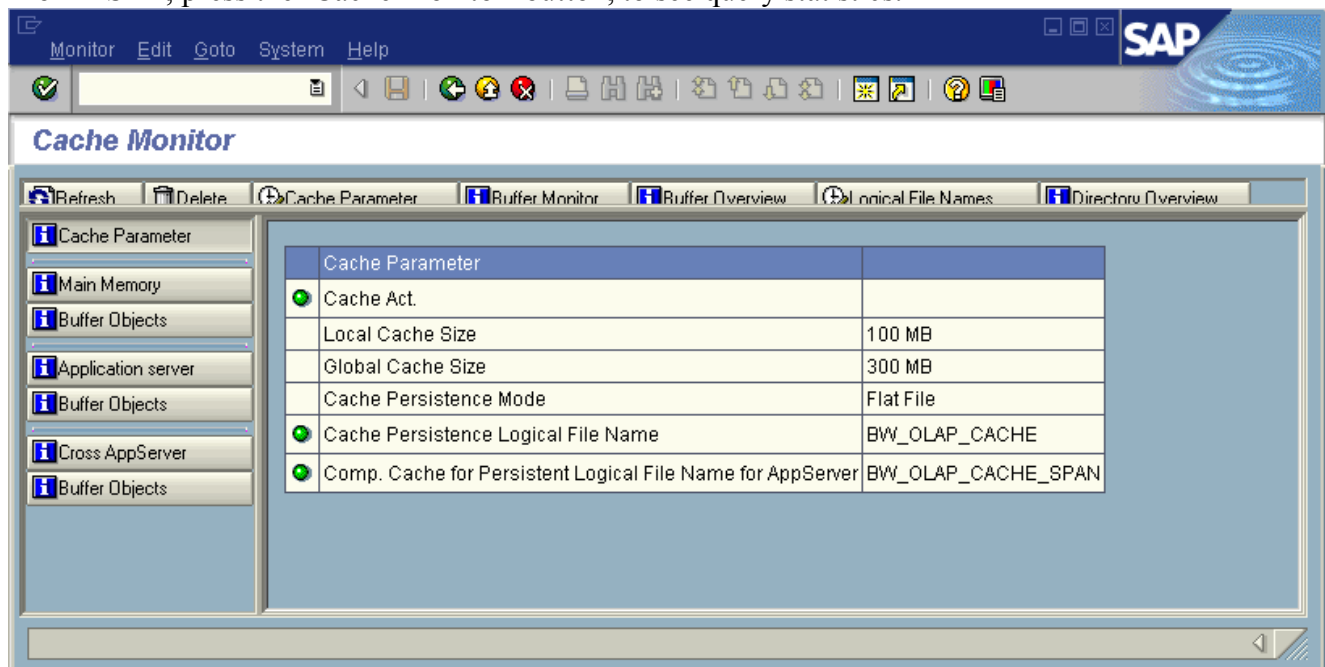


Figure 77: RSRT cache monitor

To see cached objects, pick the cache location (main memory, application server file, cross application server file).

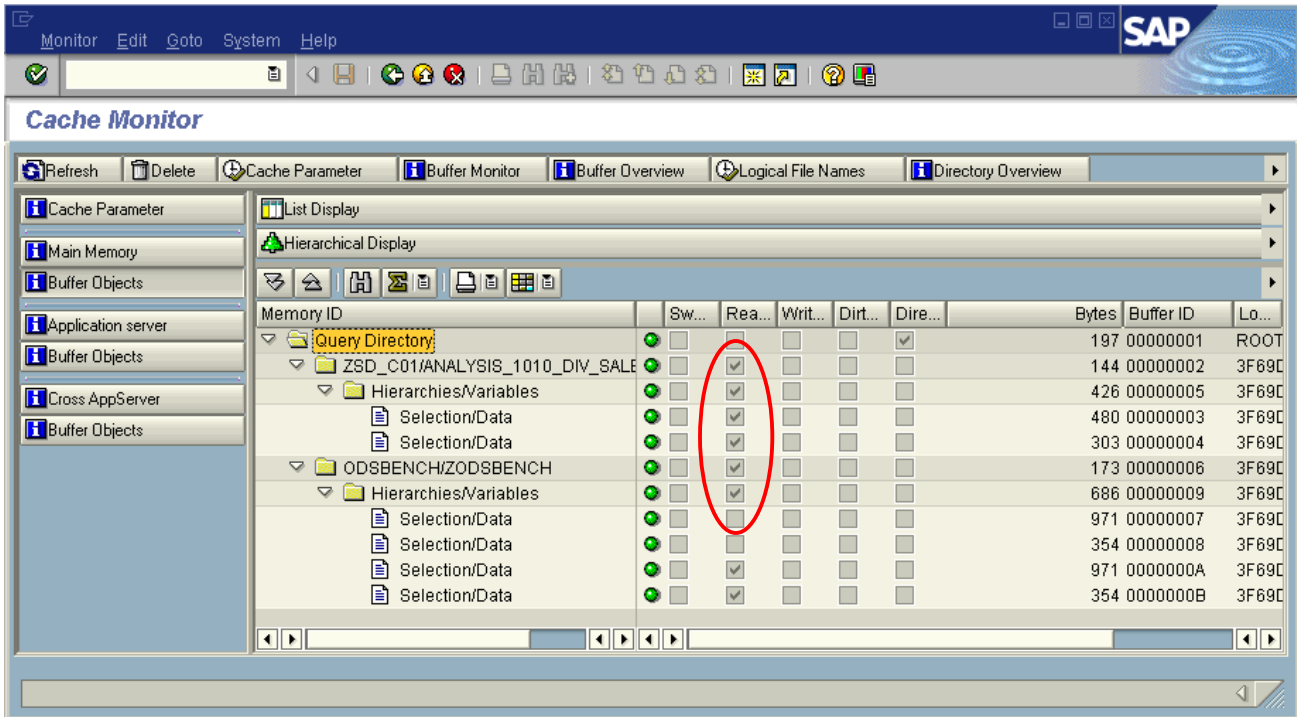


Figure 78: RSRT cached queries

This screen shows if the query has been re-used from cache.

11.4.2. Correlation information may improve access path

If the E fact table is partitioned, it will be partitioned by an SAP time characteristic, since only time characteristics may be used for E table partitioning. Since BW statements are prepared with literals, DB2 can evaluate the predicates, and use PAGE_RANGE=Y access to eliminate access to unnecessary partitions.

Since SAP also has a time dimension on the infocube, if DB2 does not know that the time dimension and the partitioning column are correlated, DB2 may over-estimate the filtering when joining the T dimension before the fact table in conjunction with PAGE_RANGE=Y filtering on the time characteristic. For example, if two columns are uncorrelated and DB2 estimates that each dimension will choose 10% of the rows, then DB2 will estimate that if both columns can be applied together, they will retrieve 1% of the rows, since $0.10 * 0.10 = 0.01$. But if they are completely correlated, then 10% of the rows will be retrieved.

In this example, we're testing the performance of a query that has been identified as being slow. We could have run the query and retrieved the STATUID, as shown in Section 10.1, or could have retrieved the STATID from RSDDSTAT as shown in Section 10.2. We use ST04 to select the STATUID for the specific query to be examined.

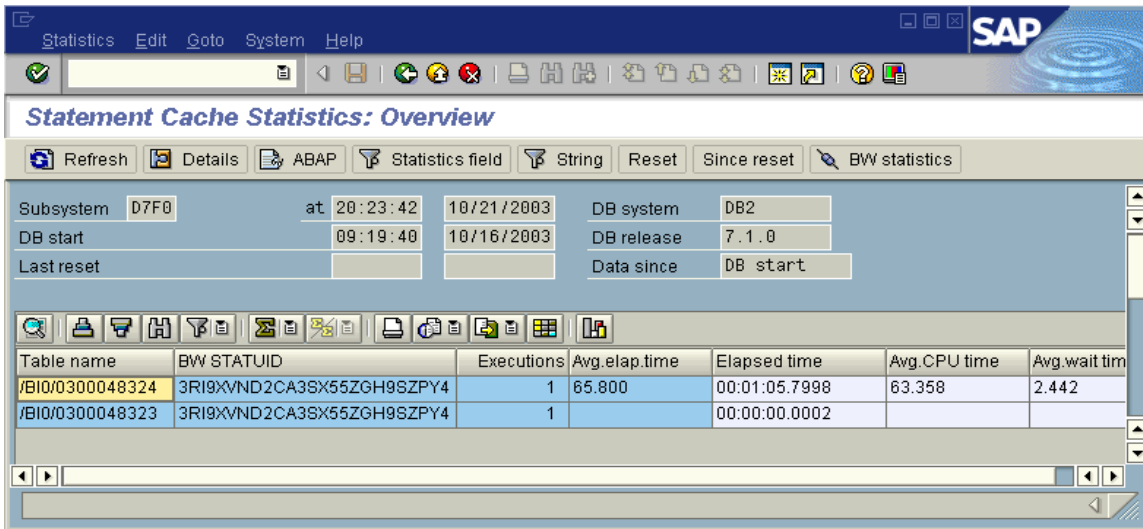


Figure 79: ST04 correlation example

Since there are both E and F fact tables in this example, there are two SQL statements, one on each fact table. The statement with 65 second elapsed time has an unusual time profile – 63 seconds CPU time. In general, I/O delay will be a larger percent of elapsed time.

Select the 65 second statement, and press “BW statistics”.

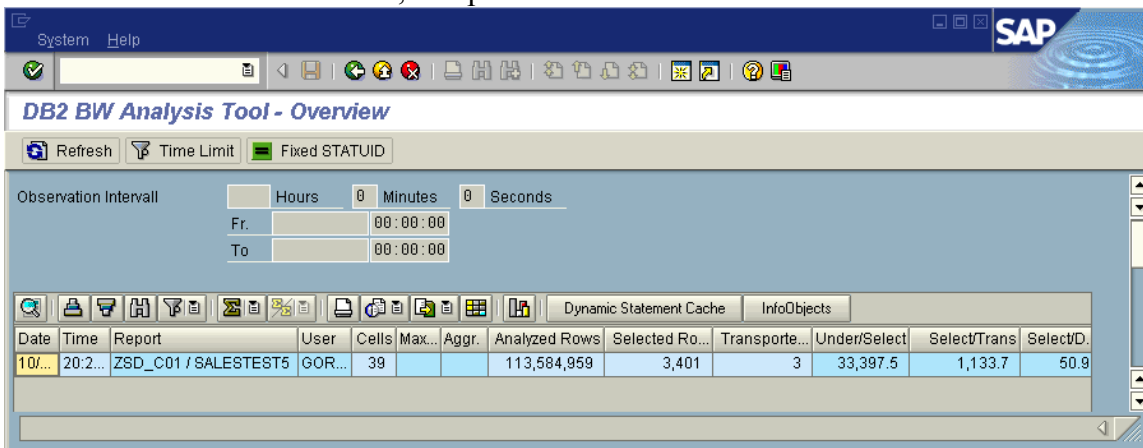


Figure 80: BW statistics correlation example

Rows examined/rows selected (here Under/Select – early translation which will be changed to ‘examined/selected’ in support package) is over 30,000. This can be a symptom of an SQL problem, where DB2 has to read many rows, to create the result set.

QDBSEL/QDBTRANS (here Select/Trans) is over 1,000, which shows that an aggregate could also help the performance of the statement. In this example, we will focus on finding the cause of the high ratio of rows examined/rows selected.

In Figure 79, select the 65 second statement, and press ‘Details’.

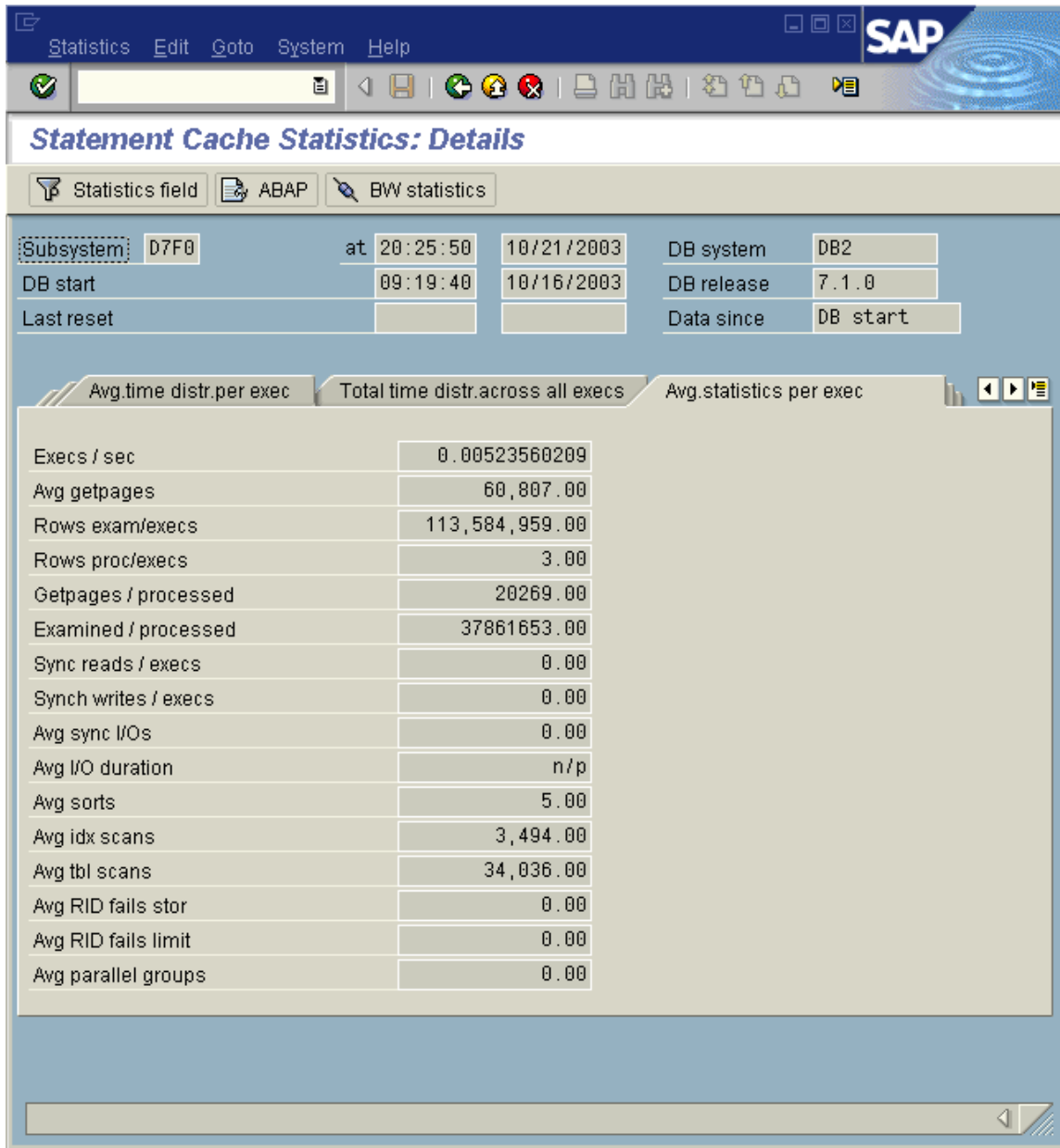


Figure 81: ST04 details correlation example

Since parallel groups = 0, the statistics for this statement are valid. See Section 6.6.1, for information on how parallelism can affect statement statistics.

Select the 'Statement text' tab.

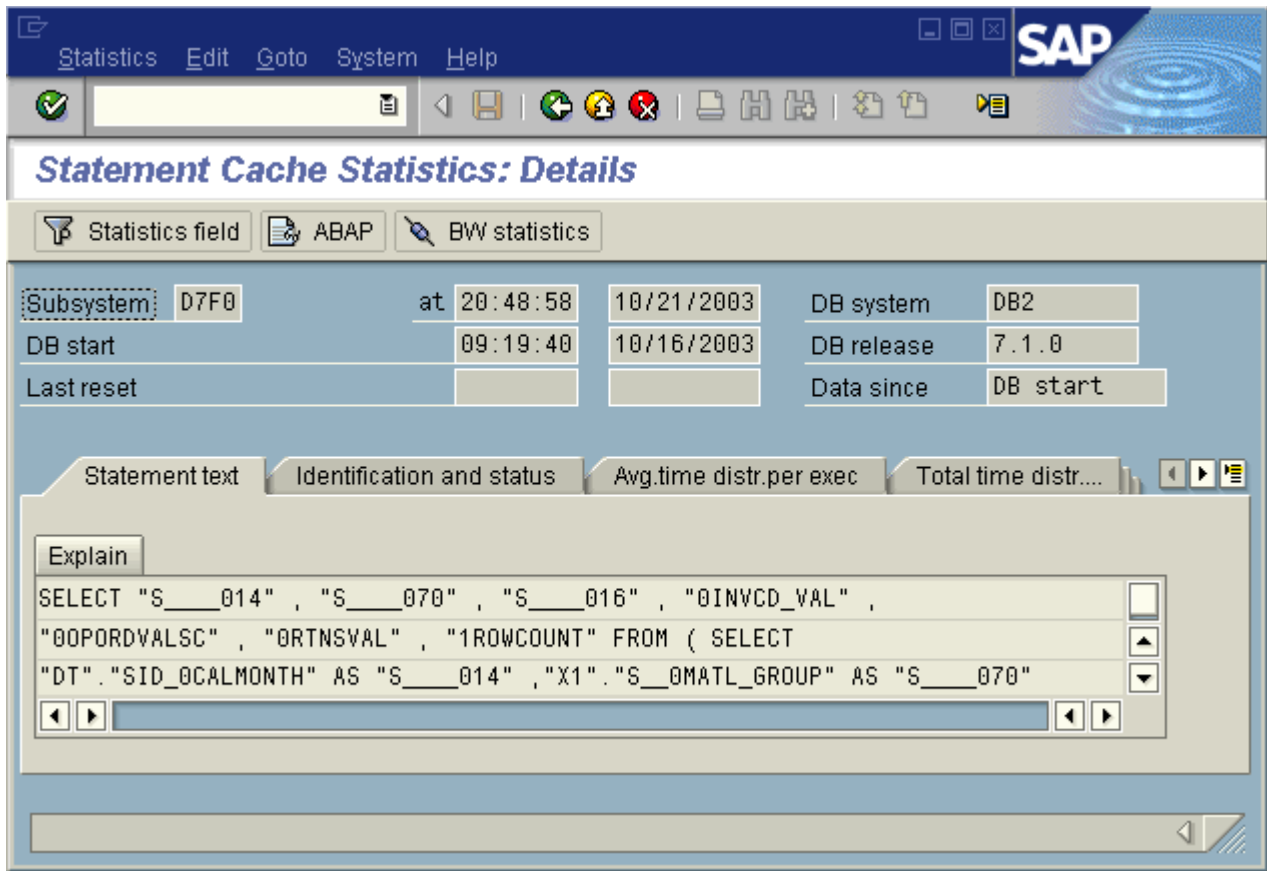


Figure 82: ST04 correlation example

And explain the statement.

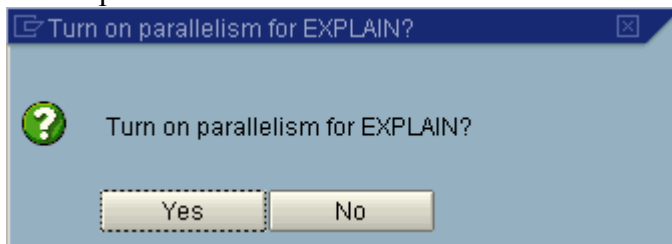


Figure 83: parallelism prompt

Answer “yes” to the prompt, since BW SQL is executed with parallelism enabled.

In this case, since dimension T and P are being joined before the fact table, we will start with the assumption that the problem is that the T dimension and the partitioning column, which is also a time characteristic, are correlated, and SAP is over-estimating the filtering from joining T before the fact table.

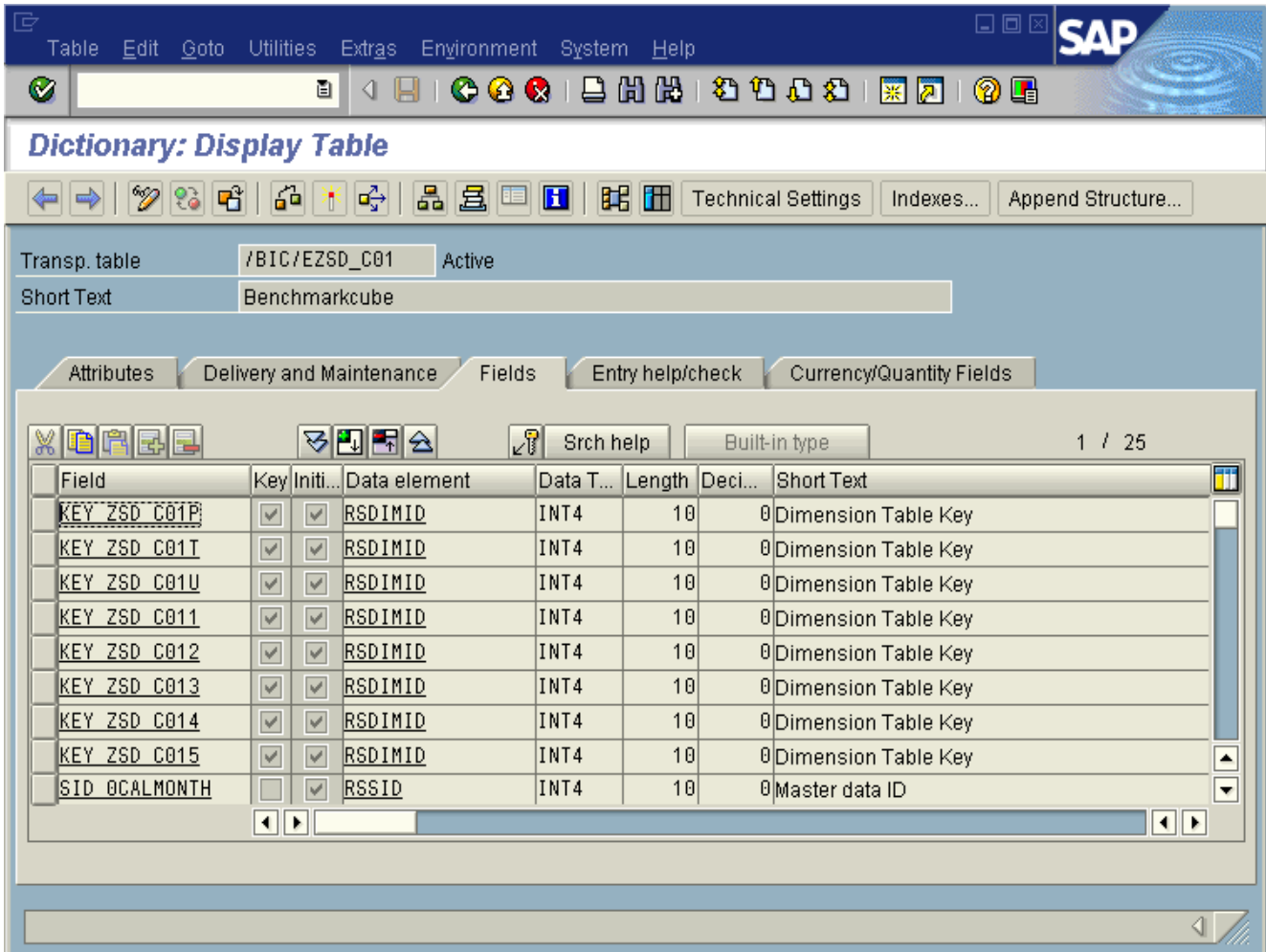


Figure 86: SE11 display of E fact table

We create statistics on the combined columns KEY_ZSD_C01T and SID_0CALMONTH. Since the goal is to show that the two columns are correlated, not to encourage DB2 to use these two columns, DSTATS should be used to gather the statistics. (Creating an index on the two columns will also create the statistics, but it will take up space, and would not be an index that would offer a benefit.)

After creating the statistics, run the query.

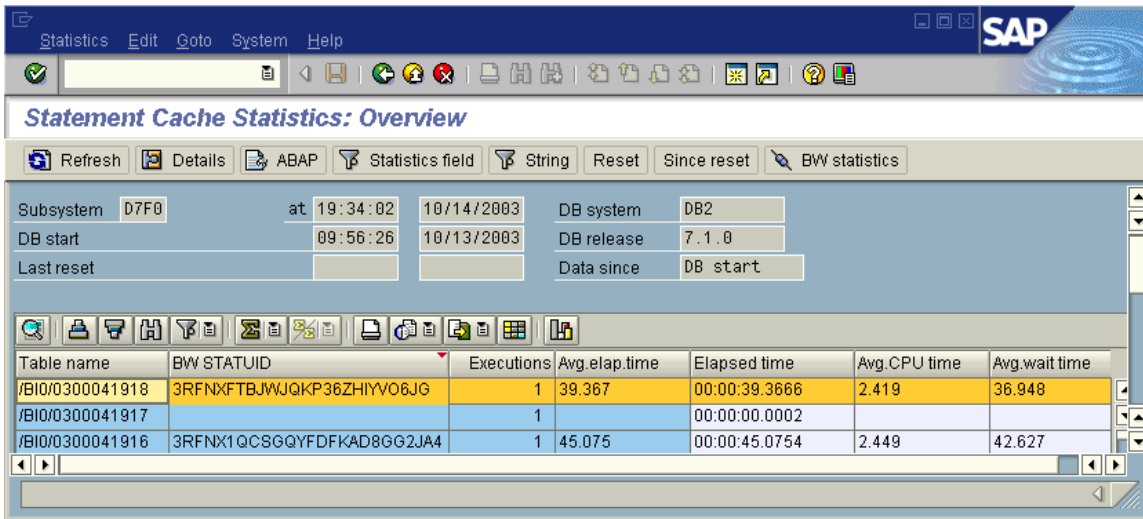


Figure 87: ST04 statement after correlation stats

In this example generated on a test system, elapsed time was reduced by 30%, but CPU use was reduced by 95%. When this problem has been encountered at productive BW sites, the elapsed time reductions are often much more than 30%.

View the plan table, to see what has changed.

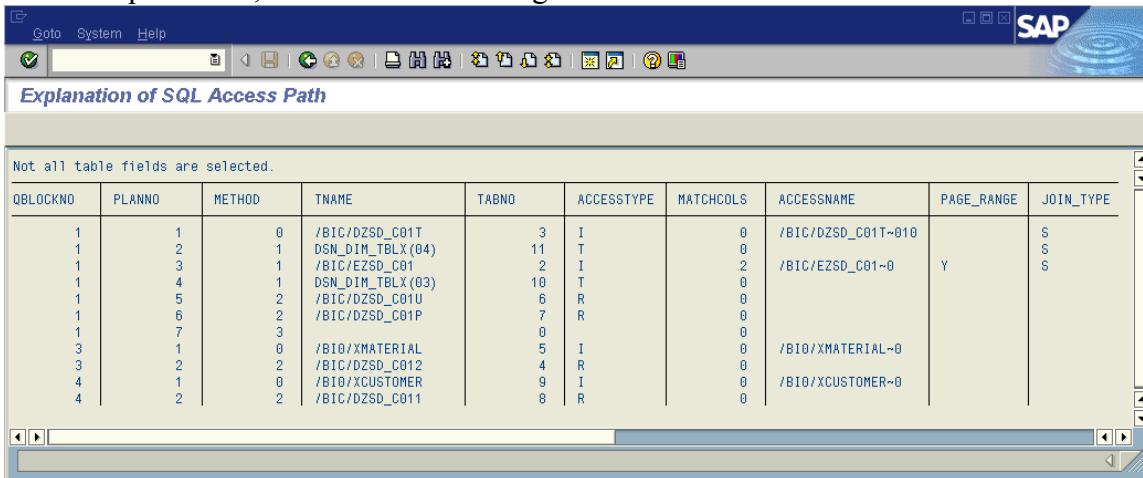


Figure 88: plan table after correlation stats

With the correlation statistics, dimension 1 is now joined ahead of the fact table with the T dimension. There is still ACESSTYPE R MATCHCOLS 0 access after the fact table, It is used for the P and T dimensions, which are very small, and it occurs after all the other dimensions have been joined to filter and group the result.

In summary, if CPU time is a very large percentage of statement elapsed time, and if DB2 chooses ACESSTYPE R MATCHCOLS 0 to join a subquery result after the fact table is read, this could be a sign that DB2 has over-estimated the filtering of some of the dimensions joined earlier. On E fact tables, correlation between the T dimension and the clustering column is one of the possible causes.

11.4.3. More column distribution information may improve access path

When an index is created on a table, by default the statistics collected by SAP are KEYCARD, FREQVAL 1 COUNT 10. Column distribution information will only be collected on the first 10 values of the first column in the index. If the data is skewed, where for example there are 1000 different values, and the top 10 values account for 50% of rows, but the next 10 account for another 40%, then, then if a query is executed using the 11th most frequently occurring value, DB2 will calculate that it occurs in 0.05% of the rows (50%/990 values), where it may actually occur in 4% of the rows. In this case, creating FREQVAL statistics on the index using a higher COUNT can help. Set the COUNT to a high enough value that most of the values with high occurrences have statistics collected.

A query has been reported as a performance problem, find and explain the SQL as shown above.

Character field truncation occurred.
Not all table fields are selected.

QBLOCKNO	PLANNO	METHOD	TNAME	TABNO	ACESSTYPE	MATCHCOLS	ACCESSNAME
1	1	0	DSN_DIM_TBLX(03)	11	R	0	
1	2	1	/BIC/FZIC_SFC	2	I	1	/BIC/FZIC_SFC~020
1	3	1	/BIC/DZIC_SFC4	9	I	1	/BIC/DZIC_SFC4~0
1	4	1	/BIC/DZIC_SFCP	10	I	1	/BIC/DZIC_SFCP~0
1	5	1	DSN_DIM_TBLX(04)	12	I	0	
1	6	1	DSN_DIM_TBLX(05)	13	R	0	
1	7	3		0		0	
3	1	0	/BIO/SFISCYEAR	4	R	0	
3	2	1	/BIC/DZIC_SFCT	3	I	0	/BIC/DZIC_SFCT~010
4	1	0	/BIC/XZ_PRODCC	6	I	1	/BIC/XZ_PRODCC~Z02
4	2	1	/BIC/DZIC_SFC3	5	I	1	/BIC/DZIC_SFC3~010
5	1	0	/BIC/DZIC_SFC1	7	R	0	
5	2	1	/BIC/XZ_SHIPT0	8	I	2	/BIC/XZ_SHIPT0~0

Figure 89: plan table for example of increasing freqval count

Dimension T is joined ahead of the F fact table, and there is ACESSTYPE R MATCHCOL 0 (table space scan) join as the last query step. This R 0 join is a sign that DB2 may have over-estimated the filtering of some of the predicates applied earlier.

Since this is a query on an F fact table, which is not partitioned by time, this is a different scenario that the correlation problem in Section 11.4.2.

The statement statistics show that CPU is the majority of elapsed time, which is what is generally seen when DB2 erroneously chooses ACESSTYPE R MATCHCOLS 0 for a join after the fact table.

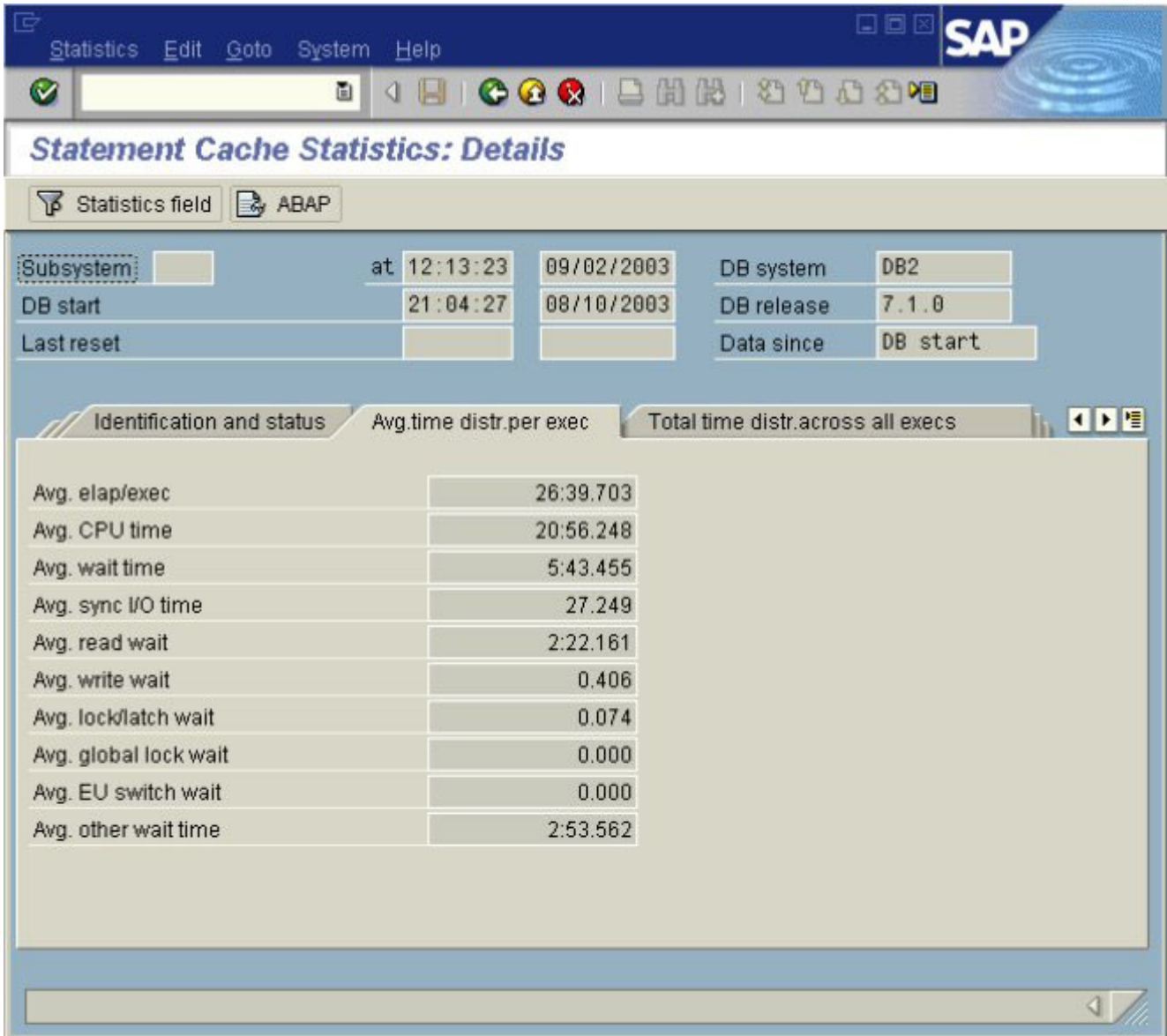


Figure 90: ST04 times for freqval count example

Examine the SQL, to determine the local predicates, since they will be the candidates for RUNSTATS changes.

```

SELECT "K____104", "S____571", "S____579", "S____574", "1ROWCOUNT", "Z_COMMA",
"Z_COMMQ", "Z_NSA", "Z_NSQ" FROM ( SELECT "S1"."FISCPER" AS "K____104",
"X1"."S__Z_SLSOFF" AS "S____571", "D4"."SID_Z_DIVN" AS "S____579",
"X2"."S__Z_MAJCAT" AS "S____574", COUNT( * ) AS "1ROWCOUNT",
SUM ( "F"."/BIC/Z_COMMA" ) AS "Z_COMMA", SUM ( "F"."/BIC/Z_COMMQ" ) AS "Z_COMMQ",
SUM ( "F"."/BIC/Z_NSA" ) AS "Z_NSA", SUM ( "F"."/BIC/Z_NSQ" ) AS "Z_NSQ"
FROM "/BIC/FZIC_SFC" "F", "/BIC/DZIC_SFCT" "DT", "/BIO/SFISCPER" "S1",
"/BIC/DZIC_SFC1" "D1", "/BIC/XZ_SHIPTO" "X1", "/BIC/DZIC_SFC4" "D4",
"/BIC/DZIC_SFC3" "D3", "/BIC/XZ_PRODCD" "X2", "/BIC/DZIC_SFPC" "DP",
WHERE "F"."KEY_ZIC_SFCT" = "DT"."DIMID" AND "DT"."SID_OFISCPER" = "S1"."SID" AND
"F"."KEY_ZIC_SFC1" = "D1"."DIMID" AND "D1"."SID_Z_SHIPTO" = "X1"."SID" AND
"F"."KEY_ZIC_SFC4" = "D4"."DIMID" AND "F"."KEY_ZIC_SFC3" = "D3"."DIMID" AND
"D3"."SID_Z_PRODCD" = "X2"."SID" AND "F"."KEY_ZIC_SFPC" = "DP"."DIMID" AND
((( "S1"."FISCPER" BETWEEN '2003005' AND '2003011' )) AND
(( "DP"."SID_0RECORDTP" = 0 )) AND (( "DP"."SID_0REQUID" <= 101406 )) AND
(( "D4"."SID_Z_DIVN" = 2 )) AND (( "X2"."S__Z_MAJCAT" = 96 ))) AND
"X1"."OBJVERS" = 'A' AND "X2"."OBJVERS" = 'A' GROUP BY
"S1"."FISCPER", "X1"."S__Z_SLSOFF", "D4"."SID_Z_DIVN", "X2"."S__Z_MAJCAT" ) "/BIO/0300611769"
FOR FETCH ONLY WITH UR

```

Figure 91: SQL for freqval count example

The local predicates of the query:

- FISCPER between
- SID_0RECORDTP =
- SID_0REQUID =
- SID_Z_DIVN =
- S__Z_MAJCAT =

SID_0RECORDTP and SID_0REQUID are SAP administrative characteristics, and can generally be left out of the evaluation. So, one needs to evaluate the frequency distribution of FISCPER, SID_Z_DIVN, and S__Z_MAJCAT.

In this case, the solution was to increase the COUNT of FREQVAL statistics on SID_Z_DIVN, since SID_Z_DIVN=2 occurred relatively frequently, though not frequently enough to be in the top 10 and in the default statistics.

After executing runstats to increase the FREQVAL COUNT on the table (/BIC/DZIC_SFC4 - dimension 4) containing SID_Z_DIVN, run the query again, and DB2 processes the query with a sparse index on the workfile on QBLOCKNO 1 PLANNO 6.

Character field truncation occurred.
Not all table fields are selected.

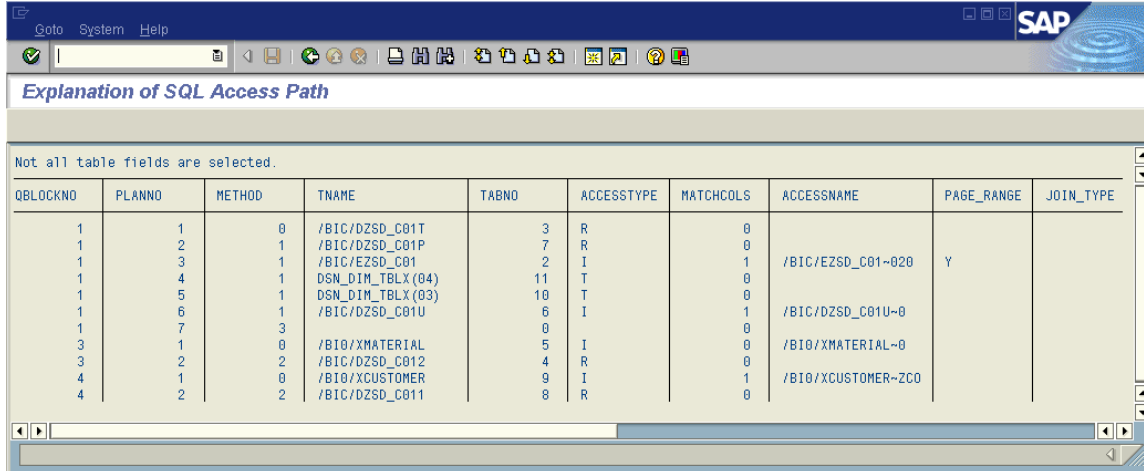
QBLOCKNO	PLANNO	METHOD	TNAME	TABNO	ACESSTYPE	MATCHCOLS	ACCESSNAME
1	1	0	DSN_DIM_TBLX (03)	11	R	0	
1	2	1	/BIC/FZIC_SFC	2	I	1	/BIC/FZIC_SFC~020
1	3	1	/BIC/DZIC_SFC4	9	I	1	/BIC/DZIC_SFC4~0
1	4	1	/BIC/DZIC_SFCP	10	I	1	/BIC/DZIC_SFCP~0
1	5	1	DSN_DIM_TBLX (04)	12	T	0	
1	6	1	DSN_DIM_TBLX (05)	13	T	0	
1	7	3		0		0	
3	1	0	/BI0/SFISCYEAR	4	R	0	
3	2	1	/BIC/DZIC_SFCT	3	I	0	/BIC/DZIC_SFCT~010
4	1	0	/BIC/XZ_PROD CD	6	I	1	/BIC/XZ_PROD CD~Z02
4	2	1	/BIC/DZIC_SFC3	5	I	1	/BIC/DZIC_SFC3~010
5	1	0	/BIC/DZIC_SFC1	7	R	0	
5	2	1	/BIC/XZ_SHIPTO	8	I	2	/BIC/XZ_SHIPTO~0

Figure 92: plan table after increasing COUNT in FREQVAL

In this case, the first symptom of the problem is similar to the correlation example, tablespace scan on a workfile being joined after the fact table. The cause was uneven distribution of values in one of the dimension table columns, and the solution was to increase the COUNT of FREQVAL statistics gathered on that column.

11.4.4. New index may improve access path

Not all characteristic and navigation attribute columns are indexed when master data tables are created. After examining the SQL statement to determine all the local predicates, use explain plan to review which indexes are being used, and check that the predicate columns have indexes that can be used. New indexes may be required on columns in dimension, attribute SID, or master data ID tables.



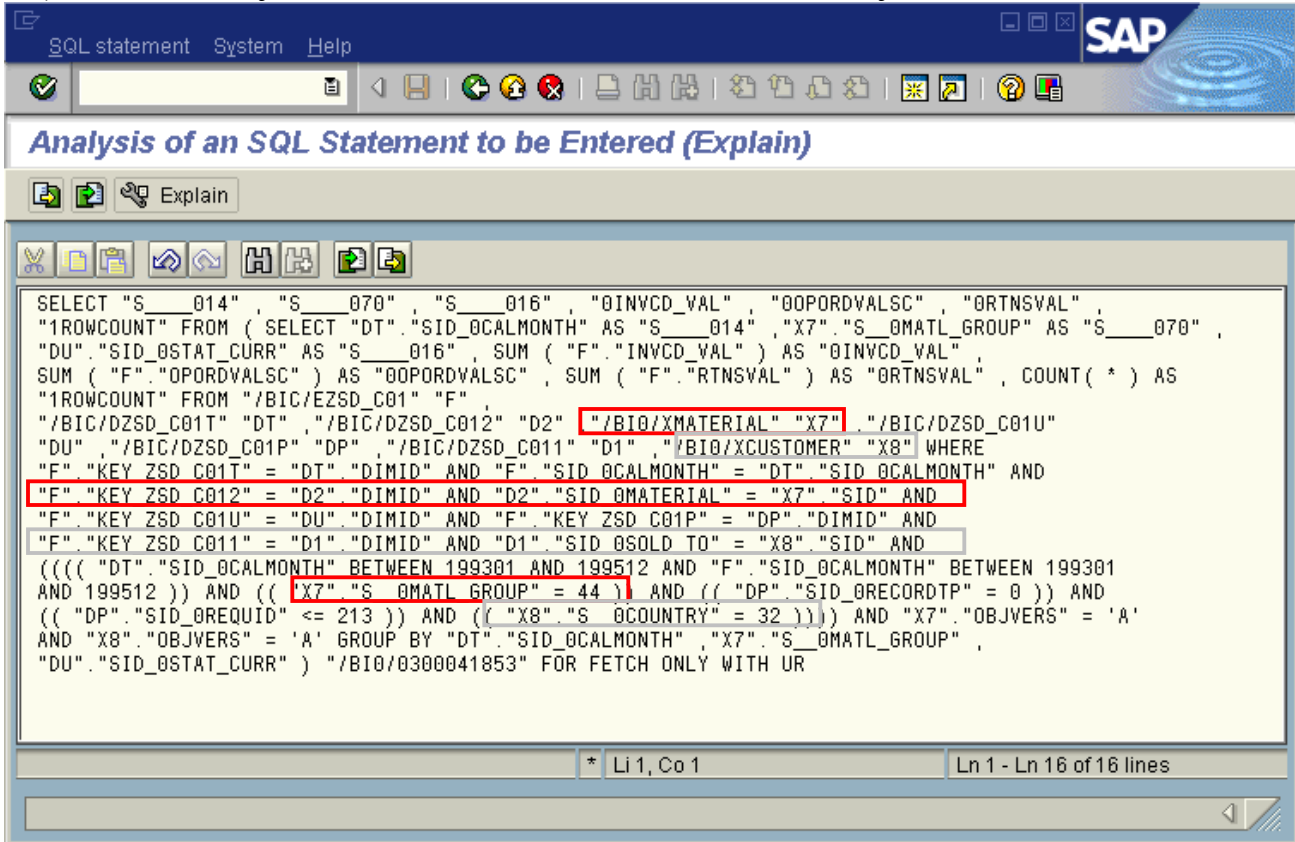
Not all table fields are selected.

QBLOCKNO	PLANNO	METHOD	TNAME	TABNO	ACESSTYPE	MATCHCOLS	ACCESSNAME	PAGE_RANGE	JOIN_TYPE
1	1	0	/BIC/DZSD_C01T	3	R	0			
1	2	1	/BIC/DZSD_C01P	7	R	0			
1	3	1	/BIC/EZSD_C01	2	I	1	/BIC/EZSD_C01-020	Y	
1	4	1	DSN_DIM_TBLX(04)	11	T	0			
1	5	1	DSN_DIM_TBLX(03)	10	T	0			
1	6	1	/BIC/DZSD_C01U	6	I	1	/BIC/DZSD_C01U-0		
1	7	3		0		0			
3	1	0	/B10/XMATERIAL	5	I	0	/B10/XMATERIAL-0		
3	2	2	/BIC/DZSD_C012	4	R	0			
4	1	0	/B10/XCUSTOMER	9	I	1	/B10/XCUSTOMER-ZCO		
4	2	2	/BIC/DZSD_C011	8	R	0			

Figure 93: ST04 plan table display for material group example

In Figure 93, the T and P dimensions are being accessed before the fact table, and then dimensions 1, 2, and U are joined after the fact table. Since PAGE_RANGE=Y denotes that partition elimination is being done on the E table, and since E tables are partitioned by a time characteristic, joining the T dimension ahead of the fact table will probably not help filter too much. We want to see whether DB2 can be encouraged to join dimension 1 or 2 before the fact table, and this join, combined with the PAGE_RANGE=Y access, will give filtering by two uncorrelated dimensions.

In Figure 93 there were two subqueries that join navigation attributes to dimensions tables, /BI)/XMATERIAL joined to dimension 2, and /BI0/XCUSTOMER joined to dimension 1.



The screenshot shows the SAP SQL statement analysis tool. The title bar reads "Analysis of an SQL Statement to be Entered (Explain)". The main window displays the following SQL statement:

```

SELECT "S__014" , "S__070" , "S__016" , "0INVCD_VAL" , "0OPORDVALSC" , "0RTNSVAL" ,
"1ROWCOUNT" FROM ( SELECT "DT"."SID_0CALMONTH" AS "S__014" , "X7"."S__0MATL_GROUP" AS "S__070" ,
"DU"."SID_0STAT_CURR" AS "S__016" , SUM ( "F"."INVCD_VAL" ) AS "0INVCD_VAL" ,
SUM ( "F"."OPORDVALSC" ) AS "0OPORDVALSC" , SUM ( "F"."RTNSVAL" ) AS "0RTNSVAL" , COUNT ( * ) AS
"1ROWCOUNT" FROM "/BIC/EZSD_C01" "F" ,
"/BIC/DZSD_C01T" "DT" , "/BIC/DZSD_C012" "D2" , "/BI0/XMATERIAL" "X7" , "/BIC/DZSD_C01U"
"DU" , "/BIC/DZSD_C01P" "DP" , "/BIC/DZSD_C011" "D1" , "/BI0/XCUSTOMER" "X8" WHERE
"F"."KEY_ZSD_C01T" = "DT"."DIMID" AND "F"."SID_0CALMONTH" = "DT"."SID_0CALMONTH" AND
"F"."KEY_ZSD_C012" = "D2"."DIMID" AND "D2"."SID_0MATERIAL" = "X7"."SID" AND
"F"."KEY_ZSD_C01U" = "DU"."DIMID" AND "F"."KEY_ZSD_C01P" = "DP"."DIMID" AND
"F"."KEY_ZSD_C011" = "D1"."DIMID" AND "D1"."SID_0SOLD_TO" = "X8"."SID" AND
((( "DT"."SID_0CALMONTH" BETWEEN 199301 AND 199512 AND "F"."SID_0CALMONTH" BETWEEN 199301
AND 199512 )) AND (( "X7"."S__0MATL_GROUP" = 44 )) AND (( "DP"."SID_0RECORDTP" = 0 )) AND
(( "DP"."SID_0REQUID" <= 213 )) AND (( "X8"."S__0COUNTRY" = 32 ))) AND "X7"."OBJVERS" = 'A'
AND "X8"."OBJVERS" = 'A' GROUP BY "DT"."SID_0CALMONTH" , "X7"."S__0MATL_GROUP" ,
"DU"."SID_0STAT_CURR" ) "/BI0/0300041853" FOR FETCH ONLY WITH UR

```

Several lines in the SQL statement are highlighted in red in the screenshot, including the table names and the join conditions for X7 and X8.

Figure 94: Statement for material group example

X8 (/BI0/XCUSTOMER) is joined to dimension 1. The local predicate is on X8.S__COUNTRY, which is a navigation attribute on customer.

X7 (/BI0/XMATERIAL) is joined to dimension 2. The local predicate is on X7.S__0MATL_GROUP, which is a navigation attribute on material.

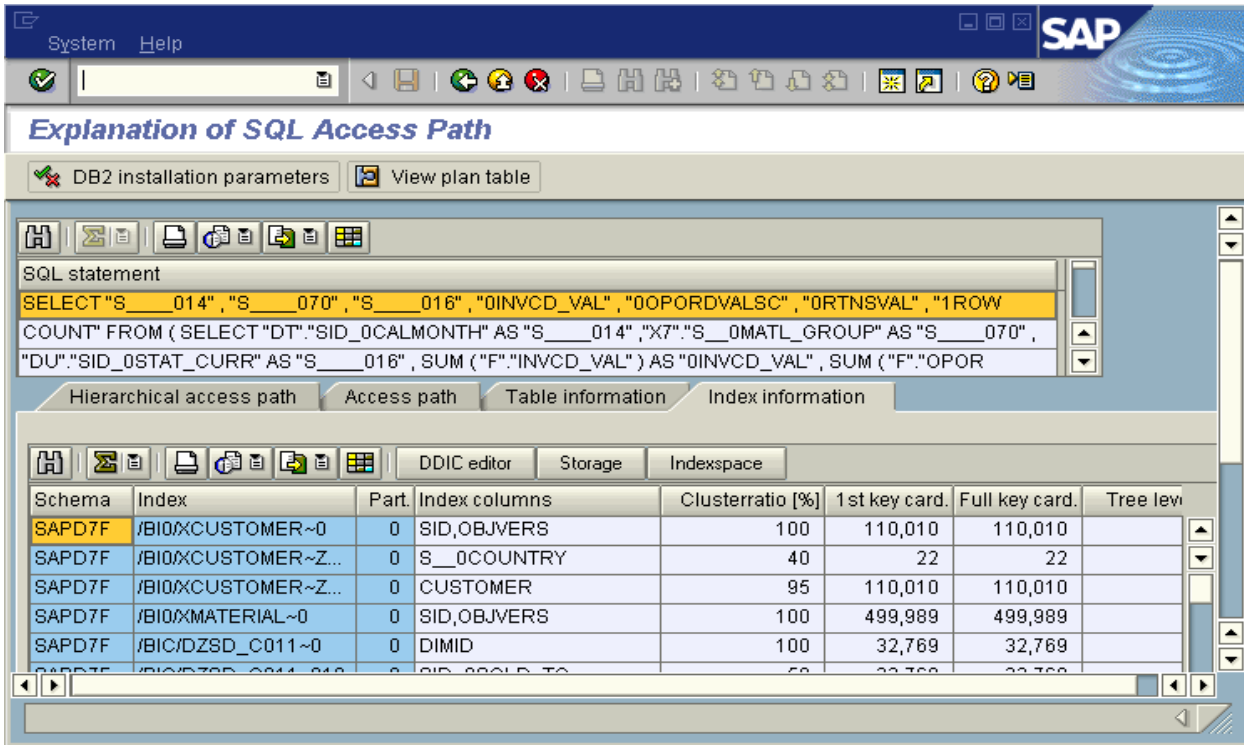


Figure 95: check indexes for material group example

There is already an index on S__0COUNTRY, but none on S__0MATL_GROUP. Without the index DB2 gathers cardinality information on the column, but does not gather column distribution.

Select the Table tab, to check table column statistics.

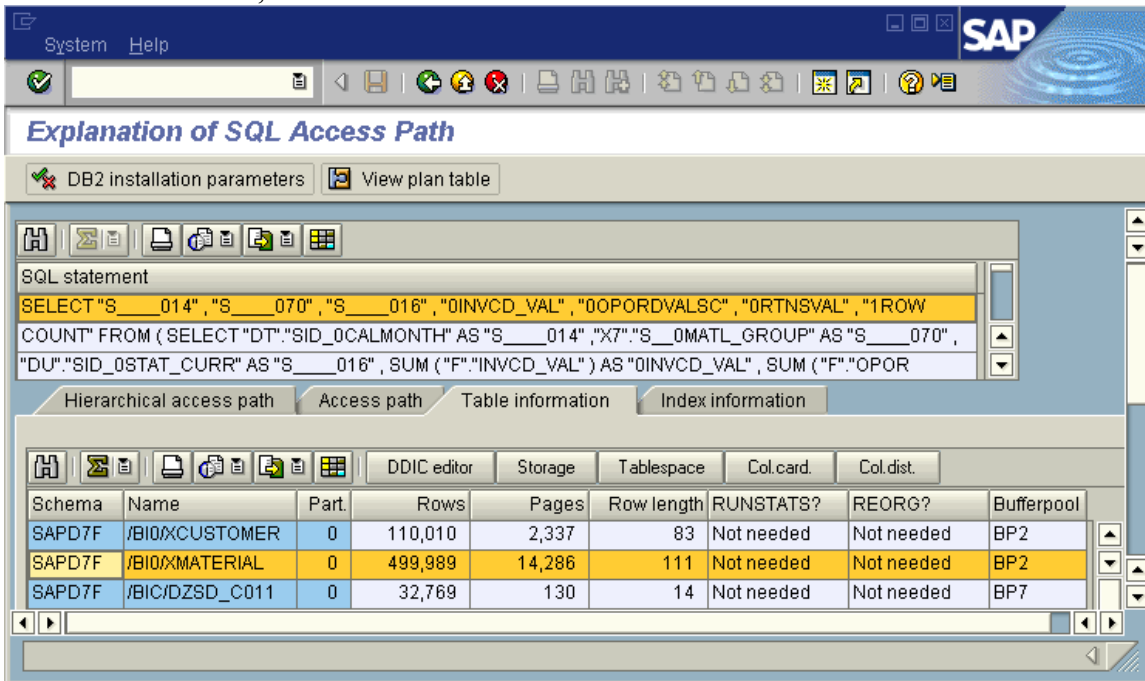


Figure 96: ST04 Table Information for material group

Press “Col card.” to display the cardinality information.

Column name	Cardinality
CHANGED	1
MATERIAL	138,675
OBJVERS	1
S_0COMPETITOR	1
S_0CRM_PROD	1
S_0DIVISION	10
S_0EXTMATLGRP	1
S_0IND_SECTOR	1
S_0MATL_CAT	1
S_0MATL_GROUP	11
S_0MATL_TYPE	100
S_0PROD_HIER	11
S_0RT_CONFMAT	1
S_0RT_PRBAND	1
S_0RT_PRRULE	1
S_0RT_SEASON	1
S_0RT_SEAYR	1
S_0RT_SUPS	1
S_0STD_DESCR	1
S_0LICERTIETY	1

Figure 97: Cardinality information for material group

Material group has cardinality 11, which means that it will not filter well, unless the data is skewed.

If the data in a column in a dimension table or master data table is skewed, and the skew corresponds to the distribution of data in the fact table, creating an index on the column will cause DB2 to gather FREQVAL statistics, and DB2 will be able to detect the skew.

Create an index on S_0MATL_GROUP column in /BI0/XMATERIAL, and run RUNSTATS. Use the “Col dist.” button to display distribution statistics.

Column group	Column value	Type	Cardinality	Frequency [%]
S_0MATL_GROUP	€	Freq.	1-	2.003644
S_0MATL_GROUP	€	Freq.	1-	2.013444
S_0MATL_GROUP	€	Freq.	1-	2.025245
S_0MATL_GROUP	€	Freq.	1-	2.027845
S_0MATL_GROUP	€	Freq.	1-	79.999560
S_0MATL_GROUP	€	Freq.	1-	1.985044
S_0MATL_GROUP	€	Freq.	1-	1.987644
S_0MATL_GROUP	€	Freq.	1-	1.992044
S_0MATL_GROUP	€	Freq.	1-	1.992644
S_0MATL_GROUP	€	Freq.	1-	1.993244

Figure 98: Material group column distribution

Here, we see that through there are only 11 values in material group, the distribution is very uneven. Ten of the material groups each have only 2% of materials. If one of the 10 material groups should be chosen, then few materials would be chosen.

Run the query again.

QBLOCKNO	PLANNO	METHOD	TNAME	TABNO	ACCESSTYPE	MATCHCOLS	ACCESSNAME	PAGE_RANGE	JOIN_TYPE
1	1	0	/BIC/DZSD_C01P	7	R	0			
1	2	1	DSN_DIM_TBLX (03)	10	R	0			
1	3	1	/BIC/EZSD_C01	2	I	1	/BIC/EZSD_C01-050	Y	
1	4	1	/BIC/DZSD_C01T	3	I	1	/BIC/DZSD_C01T-0		
1	5	1	/BIC/DZSD_C01U	6	I	1	/BIC/DZSD_C01U-0		
1	6	1	DSN_DIM_TBLX (04)	11	R	0			
1	7	3		0		0			
3	1	0	/BI0/XMATERIAL	5	I	1	/BI0/XMATERIAL-ZMG		
3	2	1	/BIC/DZSD_C012	4	I	1	/BIC/DZSD_C012-010		
4	1	0	/BI0/XCUSTOMER	9	I	1	/BI0/XCUSTOMER-ZCO		
4	2	2	/BIC/DZSD_C011	8	R	0			

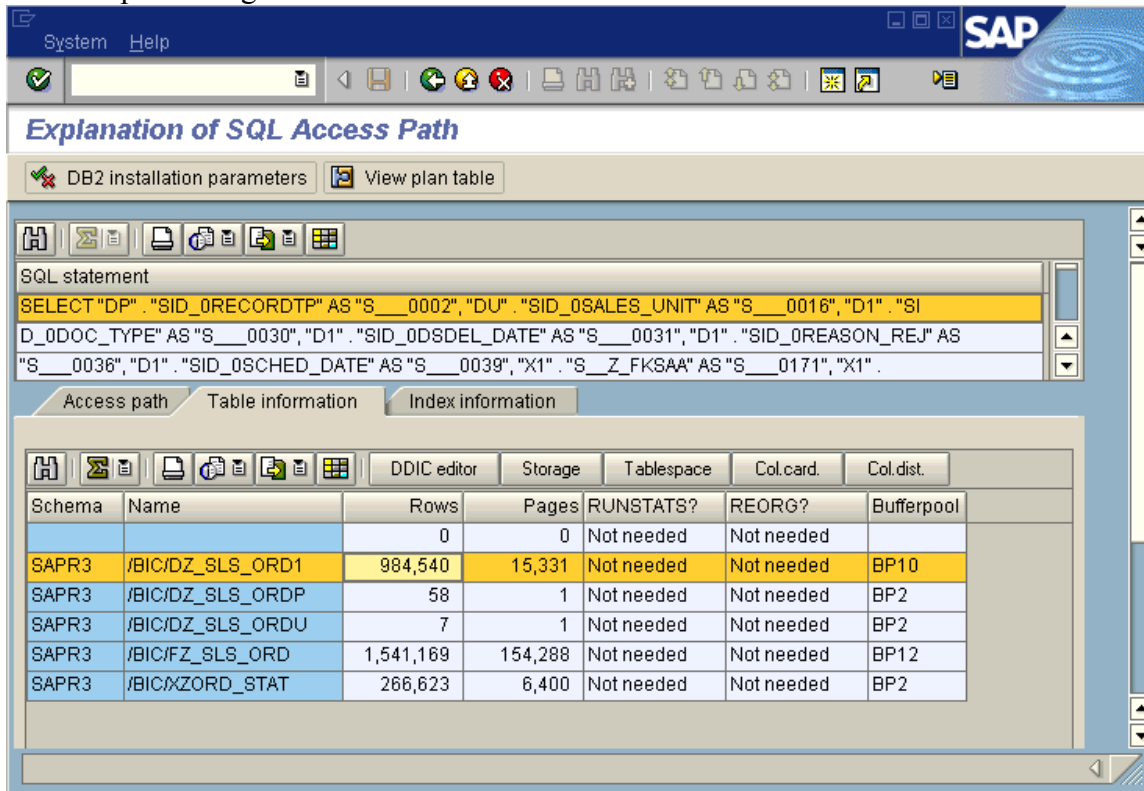
Figure 99: ST04 plan table with material group index

In Figure 99, after the material group index is created, DB2 applies its dimension, D2, before the fact table. This, in conjunction with PAGE_RANGE=Y access on the fact table allows DB2 to filter by two dimensions when accessing the fact table.

In order for column distribution statistics to work to the benefit of a query, then the skew on the indexed column in the dimension table or master data table should reflect skew in the fact table. If the master data material / material group distribution was skewed, but all the item sales were in one of the small material groups, then creating the index on the master data table would be counter-productive.

11.4.5. Check data model

In the following example, dimension 1 (table /BIC/DZ_SLS_ORD1) is nearly as large as the fact table. SAP's recommendation is that dimension tables should be less than 15% as large as the fact table, for efficient processing.



The screenshot shows the SAP DB2 installation parameters window. The 'Table information' tab is selected, displaying a table with the following data:

Schema	Name	Rows	Pages	RUNSTATS?	REORG?	Bufferpool
		0	0	Not needed	Not needed	
SAPR3	/BIC/DZ_SLS_ORD1	984,540	15,331	Not needed	Not needed	BP10
SAPR3	/BIC/DZ_SLS_ORDP	58	1	Not needed	Not needed	BP2
SAPR3	/BIC/DZ_SLS_ORDU	7	1	Not needed	Not needed	BP2
SAPR3	/BIC/FZ_SLS_ORD	1,541,169	154,288	Not needed	Not needed	BP12
SAPR3	/BIC/XZORD_STAT	266,623	6,400	Not needed	Not needed	BP2

The 'Access path' tab is also visible, showing the SQL statement:

```
SELECT "DP"."SID_0RECORDTP" AS "S__0002", "DU"."SID_0SALES_UNIT" AS "S__0016", "D1"."SI
D_0DOC_TYPE" AS "S__0030", "D1"."SID_0DSDEL_DATE" AS "S__0031", "D1"."SID_0REASON_REJ" AS
"S__0036", "D1"."SID_0SCHED_DATE" AS "S__0039", "X1"."S_Z_FKSAA" AS "S__0171", "X1".
```

Figure 100: Dimension 1 data model problem

This infocube is not using SAP compression, so all the rows are in the F fact table. Dimension 1 is about 65% as large as the F fact table.

The characteristics in the dimension should be evaluated, to determine if dimension 1 should be split into multiple dimensions, or whether there is a characteristic with very high cardinality that should be separated into a line-item dimension (a configuration setting in RSA1).

11.4.6. Not a data model problem

Since the same dimension tables are used with both the F fact table and E fact tables of an infocube, if SAP compression is activated and the F fact tables contain very few rows, the dimension tables may be relatively large when compared to the F fact tables. *When SAP compression is activated, compare the ratio of dimension size to the E fact tables when checking the data model.*

In Figure 101, the ratio of the size of the dimension tables to the F fact table looks bad; dimension 2 is about 25% as large as the fact table. Next, check the E fact table, seen in Figure 102, and we see that the dimension table cardinality is less than 10% of fact table cardinality, which is within the SAP guideline. It does, however, exceed the 1 to 25 ratio (dimension to fact table) specified via DB2

STARJOIN=0, so queries using dimension 2 on this infocube may not be eligible for star join optimization.

The screenshot shows the SAP DB2 installation parameters window. The title bar includes 'System Help' and the SAP logo. The main window title is 'Explanation of SQL Access Path'. Below the title bar, there are buttons for 'DB2 installation parameters' and 'View plan table'. The 'SQL statement' field contains the following query:

```
SELECT "S_0039", "S_0006", "S_0031", "K_0031", "S_0002", "S_0004", "S_0012",
"S_0022", "1ROWCOUNT", "ERECQTYCS", "EISSQTYCS", "ERECQTYM3", "EISSQTYM3" FROM ( SELE
CT "D2"."SID_OMATL_TYPE" AS "S_0039", "DT"."SID_0CALDAY" AS "S_0006", "D1"."SID_0PLANT" AS
```

Below the SQL statement, there are tabs for 'Access path', 'Table information', and 'Index information'. The 'Table information' tab is active, showing a table with the following data:

Schema	Name	Rows	Pages	RUNSTATS?	REORG?	Bufferpool
		0	0	Not needed	Not needed	
SAPR3	/BI0/SPLANT	244	2	Not needed	Not needed	BP2
SAPR3	/BI0/XMATERIAL	153,014	5,508	Not needed	Not needed	BP2
SAPR3	/BIC/DMMEUPINVM1	290	2	Not needed	Not needed	BP2
SAPR3	/BIC/DMMEUPINVM2	113,520	1,402	Not needed	Not needed	BP2
SAPR3	/BIC/DMMEUPINVM3	5	1	Not needed	Not needed	BP2
SAPR3	/BIC/DMMEUPINVMT	373	3	Not needed	Not needed	BP2
SAPR3	/BIC/FMMEUPINVM	499,408	41,618	Not needed	Not needed	BP2

Figure 101: F fact table - dimension table cardinality 25% of fact cardinality

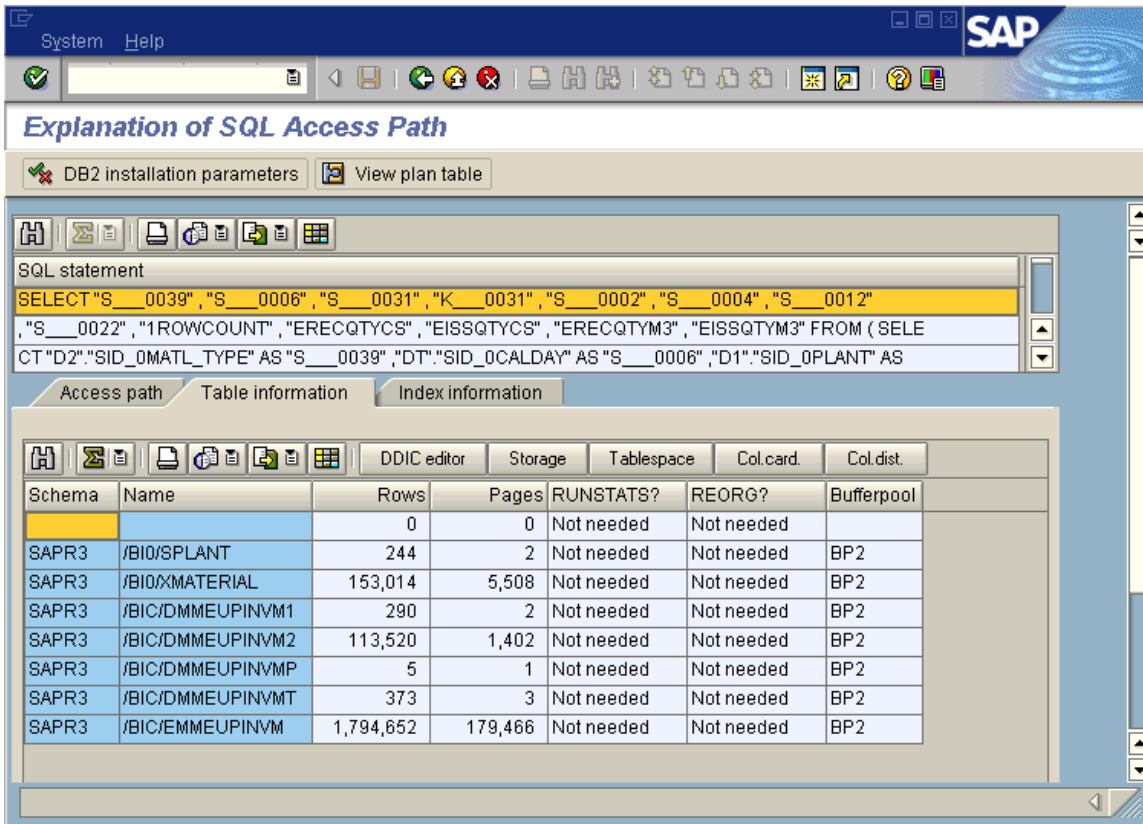


Figure 102: E fact table - dimension cardinality less than 10% of fact cardinality

11.4.7. Evaluate new indexes on fact table

If aggregates, indexes on dimensions and master data, or additional catalog statistics do not help performance, then one may be able to improve query performance with multi-column indexes on the fact table. Due to the space requirements, and difficulty in planning the index requirements, this is generally a last resort.

For ad-hoc queries against infocubes, one can improve SQL performance with multi-column indexes on the fact table. Adding a multi-column index is most useful when the dimensions filter the result set well, and when the column with the highest filter-factor (that is, the column that will choose the fewest rows out of the fact table) is the first column in the multi-column index.

As shown in Figure 3, there can be one to three tables joined to the infocube fact table for each characteristic or navigation attribute in the predicate:

- One table (dimension) when the SAP ODBC interface can substitute the SID in the query,
- Two tables (dimension and SID table) when master data IDs are used, and
- Three tables (dimension, SID table and attribute SID table) when navigation attributes are used.

This makes it rather difficult to accurately determine which dimensions will filter best, since one would need to check column cardinality and column distribution data for several tables.

11.4.7.1. Searching for filtering dimensions and characteristics

The cardinalities of the characteristics and navigation attributes can often be used to estimate how well each dimension will filter.

For example, when reviewing an SQL statement and the business hierarchy, we can assume that material may filter well, but that marketing company will not (since there will be many materials in a marketing company), and sales person may filter well, but sales region will not (since there will be many sales people in a sales area). This can be confirmed using table cardinality statistics, as shown in Figure 104.

We can examine the statement predicates using ST04 statement cache.

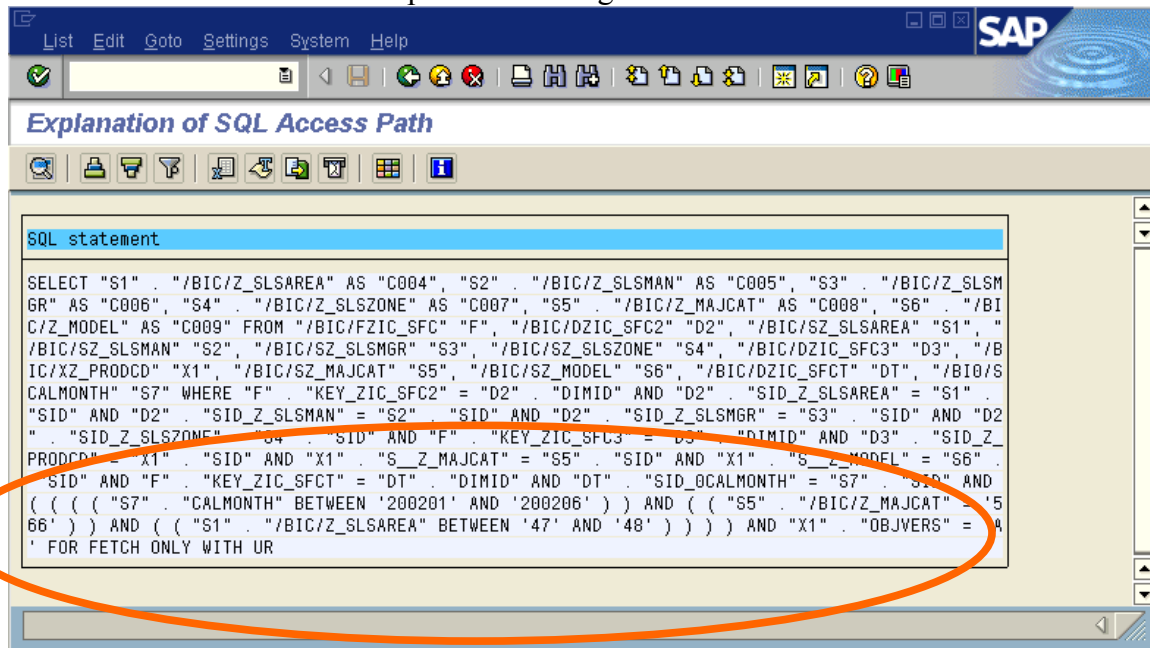


Figure 103: ST04 statement cache - examine predicates

In this case, CALMONTH, /BIC/SZ_MAJCAT, /BIC/Z_SALESAREA, and OBJVERS are the predicates. The DB2 Administration Guide has details on how range predicates (such as the BETWEEN used above) affect optimization. OBJVERS=A is for SAP management of active/inactive objects. We can ignore it.

Since ST04 explain will gather information on all the tables and indexes used by the statement, explain the statement, and check the cardinality of the tables in the predicates. Check the master data ID tables (/BIC/S*) for cardinality of the CALMONTH, /BIC/Z_MAJCAT, and /BIC/Z_SLSAREA characteristics.

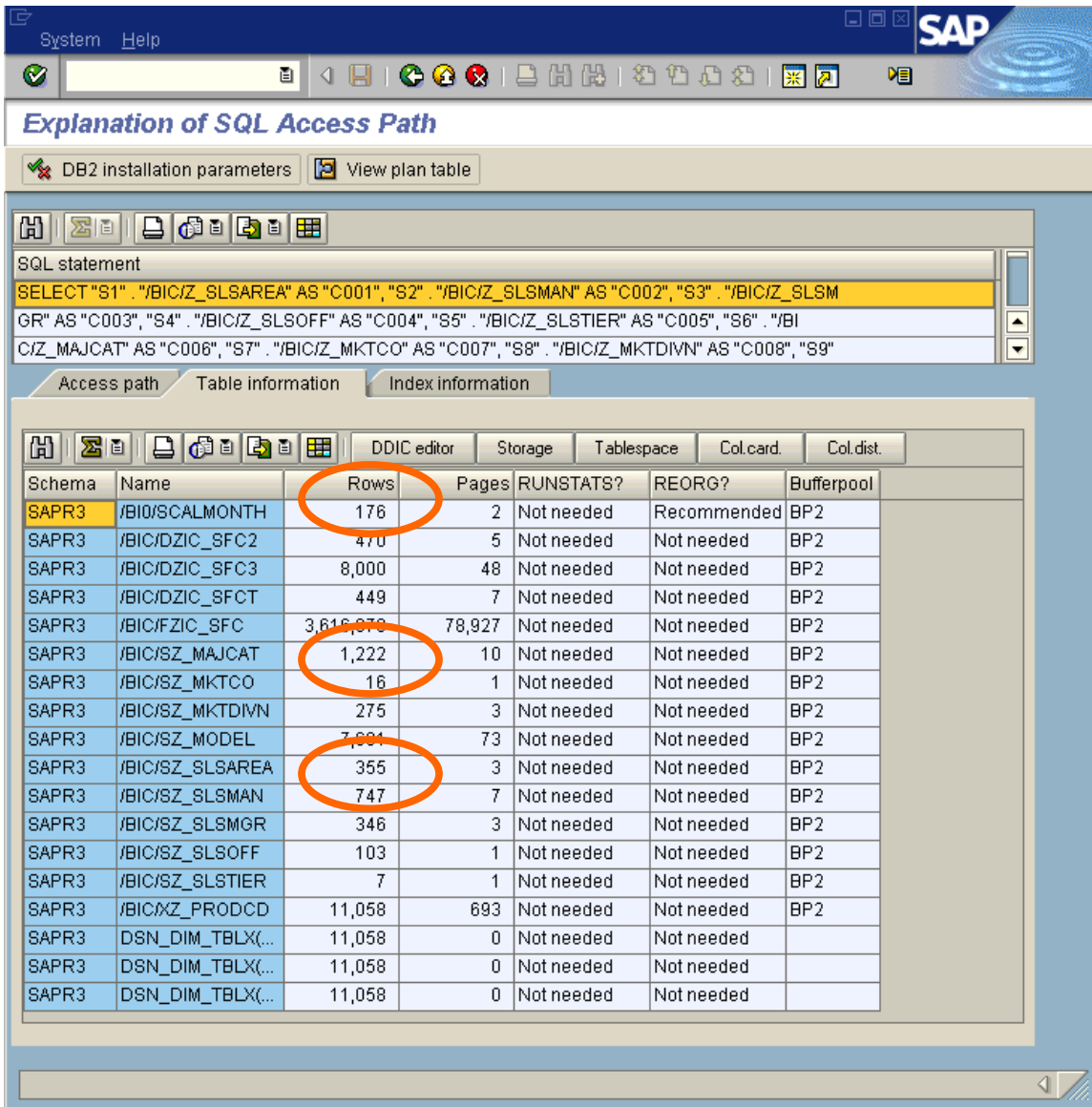


Figure 104: ST04 explain - table cardinality information

Since the cardinality of /BIC/SZ_MAJCAT (1,222) is greater than /BIC/SZ_SLSAREA (355), we can start with the assumption that selection by major category will filter better than selection by sales area, and so a new multi-column index should have the column for the dimension containing major category before the column for the dimension containing sales area. In order to have a good cluster ratio on the index, multi-column fact table indexes should usually start with dimension T or dimension P.

Next, determine which dimension contains which characteristic, since the indexes on the fact table are usually dimension indexes, not characteristic indexes. Line-item dimensions are an exception – their keys are based on a single characteristic. Find the dimension for a characteristic by following the join conditions in the statement (Figure 105), or by using ‘find’ in the RSA1 infocube data model (Figure 106).

First, we will find the dimension that contains the sales area characteristic (/BIC/Z_SLSAREA).

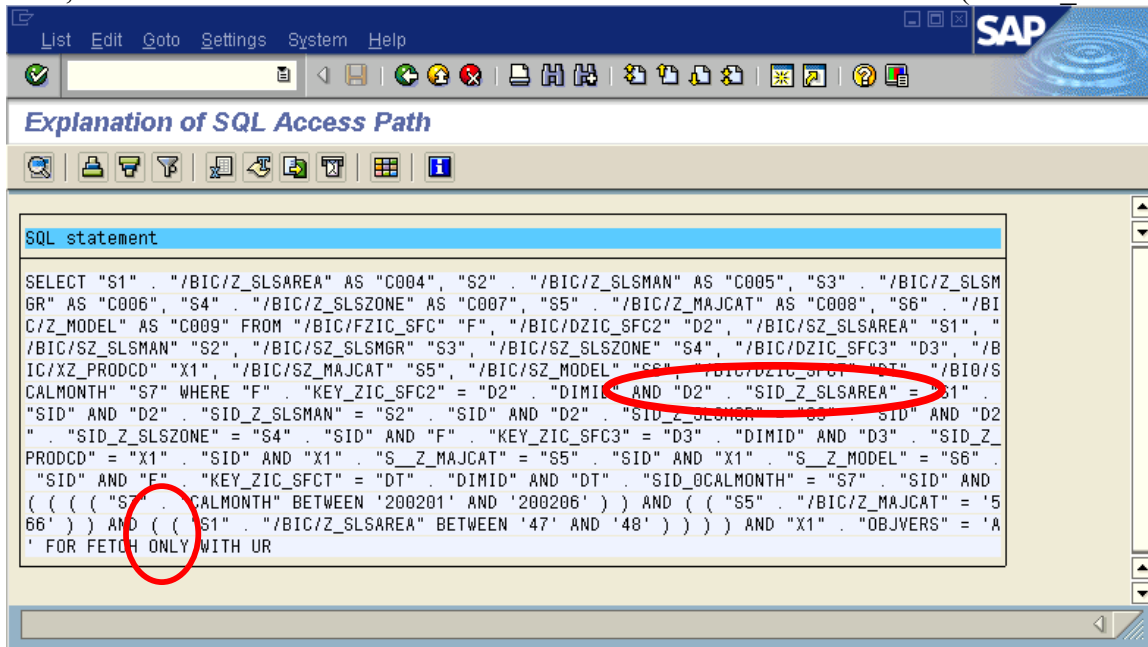


Figure 105: Find dimension from join conditions

In the where clause, S1 is the correlation ID for the master data ID table containing /BIC/Z_SLSAREA. Next, check the join conditions for S1 – it is joined to D2. So, sales area is a part of dimension 2, which is table /BIC/DZIC_SFC2 in Figure 104.

In another example, use RSA1 to find the dimension (**RSA1 > modeling > data targets > choose infocube > display data model > find**).

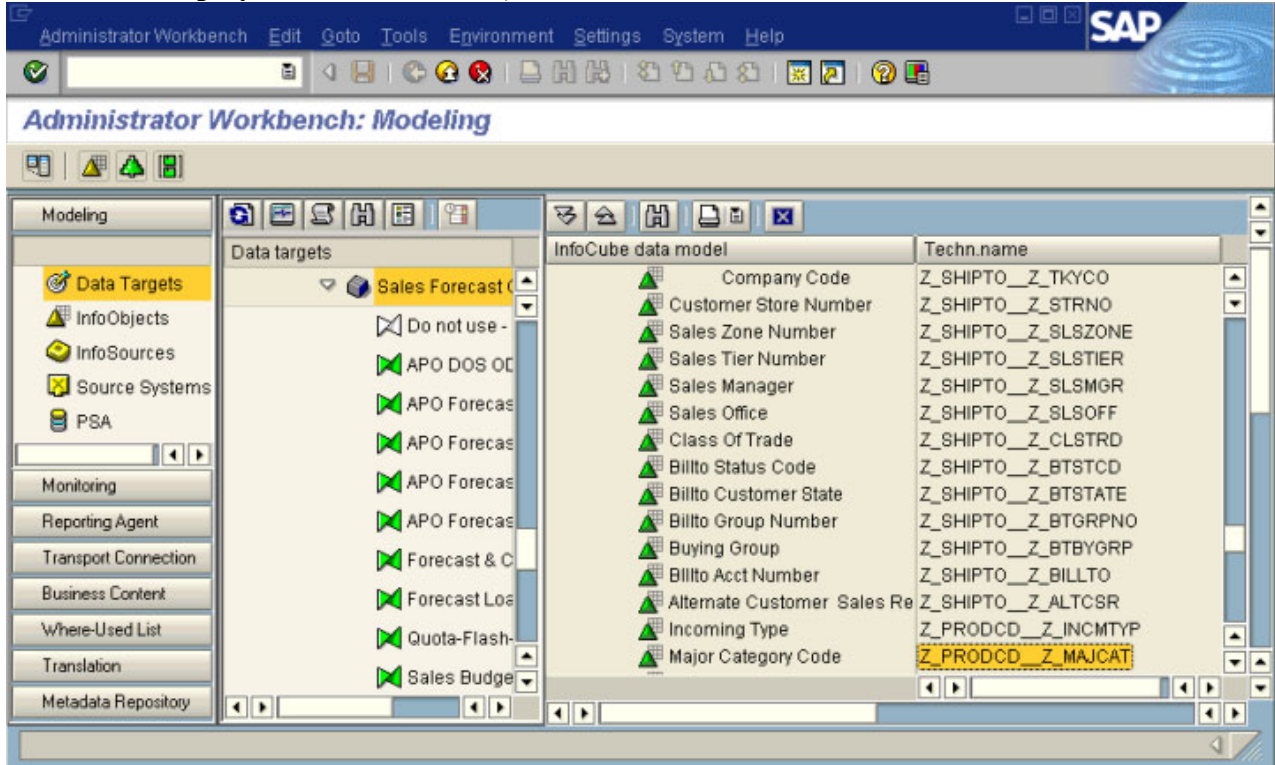


Figure 106: RSA1 find characteristic

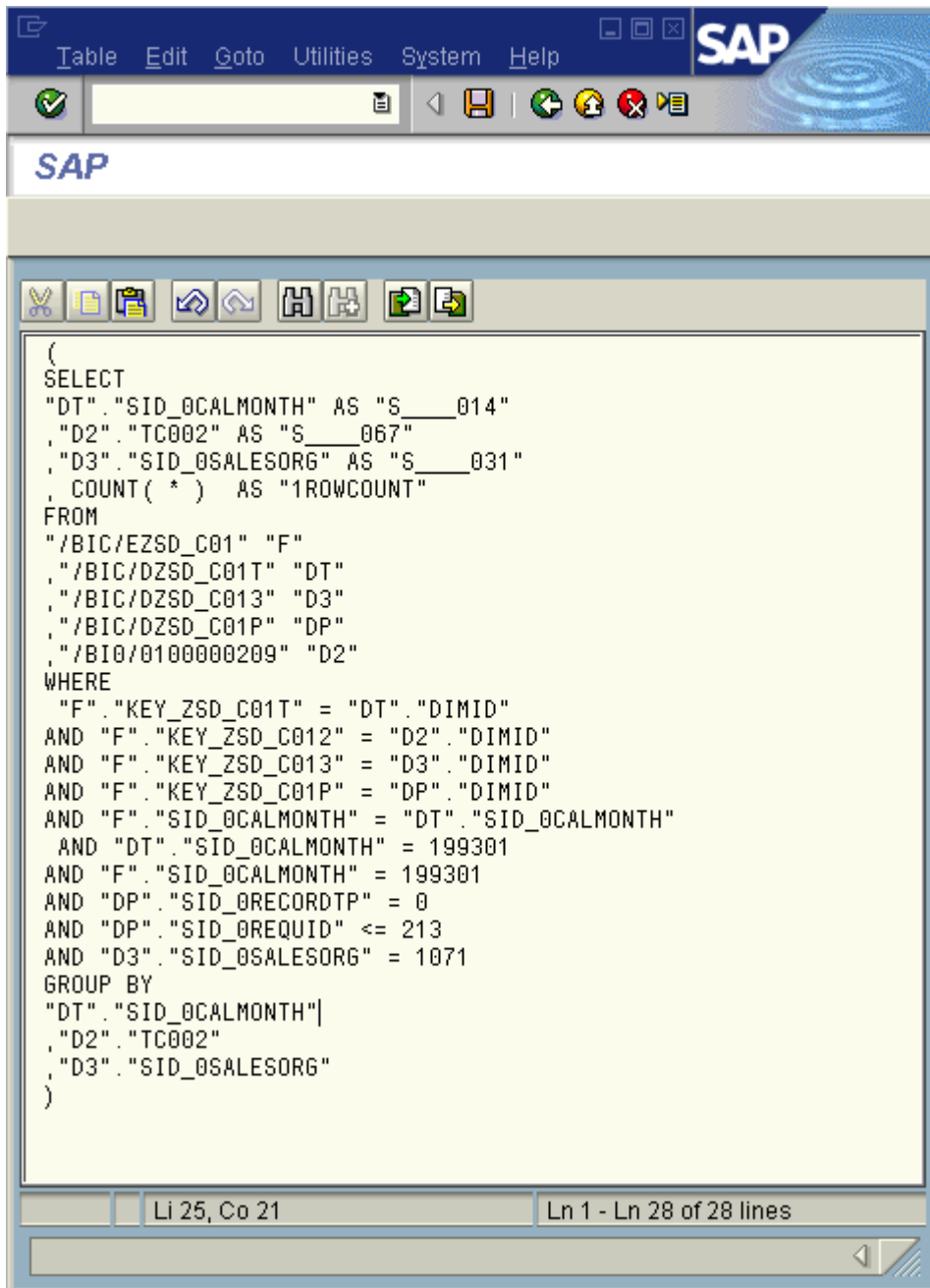
In Figure 106 major category is a navigation attribute on the product code characteristic (Z_PRODCD). Navigation attributes have concatenated names – the name of the characteristic (e.g. Z_PRODCD), two underscores, then the name of the navigation attribute on the characteristic (Z_MAJCAT). Page back up in the data model (not shown here) and major category was part of the product detail dimension, which is dimension 3, the table /BIC/DZIC_SFC3.

Now we have seen above that the characteristics for the product (Z_PRODCD, Z_MAJCAT, etc) have higher cardinality than the sales characteristics (Z_SLSMAN, Z_SLSAREA), and have also determined that dimension 3 contains the product characteristics. Now we can use SE11 to define a multi-column index on the fact table to support queries on these dimensions, specifying dimensions T, 3, and 2.

Cardinality of the master data ID tables (/BIC/S*) does not always predict how well a dimension will filter rows. The distribution may be skewed, where some characteristic values have many rows in the fact table, and some have few rows, or if not all the master data values are present in the cube. If there are several characteristics specified for a single dimension, their correlation, or lack of correlation, will also affect the filtering. The process above is meant to offer a simple way (that will often be correct) to determine which dimension columns should be in the index, and what column order should be chosen.

11.4.7.2. Using listcube to determine dimension filtering

One can use the LISTCUBE transaction to determine the actual filtering of predicates on dimensions.



```

(
SELECT
"DT"."SID_0CALMONTH" AS "S____014"
,"D2"."TC002" AS "S____067"
,"D3"."SID_0SALESORG" AS "S____031"
, COUNT( * ) AS "1ROWCOUNT"
FROM
"/BIC/EZSD_C01" "F"
,"/BIC/DZSD_C01T" "DT"
,"/BIC/DZSD_C013" "D3"
,"/BIC/DZSD_C01P" "DP"
,"/BI0/0100000209" "D2"
WHERE
"F"."KEY_ZSD_C01T" = "DT"."DIMID"
AND "F"."KEY_ZSD_C012" = "D2"."DIMID"
AND "F"."KEY_ZSD_C013" = "D3"."DIMID"
AND "F"."KEY_ZSD_C01P" = "DP"."DIMID"
AND "F"."SID_0CALMONTH" = "DT"."SID_0CALMONTH"
AND "DT"."SID_0CALMONTH" = 199301
AND "F"."SID_0CALMONTH" = 199301
AND "DP"."SID_0RECORDTP" = 0
AND "DP"."SID_0REQUID" <= 213
AND "D3"."SID_0SALESORG" = 1071
GROUP BY
"DT"."SID_0CALMONTH"|
,"D2"."TC002"
,"D3"."SID_0SALESORG"
)

```

Li 25, Co 21 Ln 1 - Ln 28 of 28 lines

Figure 107: sample SQL statement

We will use LISTCUBE to determine whether D3 (SID_0SALESORG) or DT (SID_0CALMONTH) filter better.

Run LISTCUBE and select the cube to be queried.

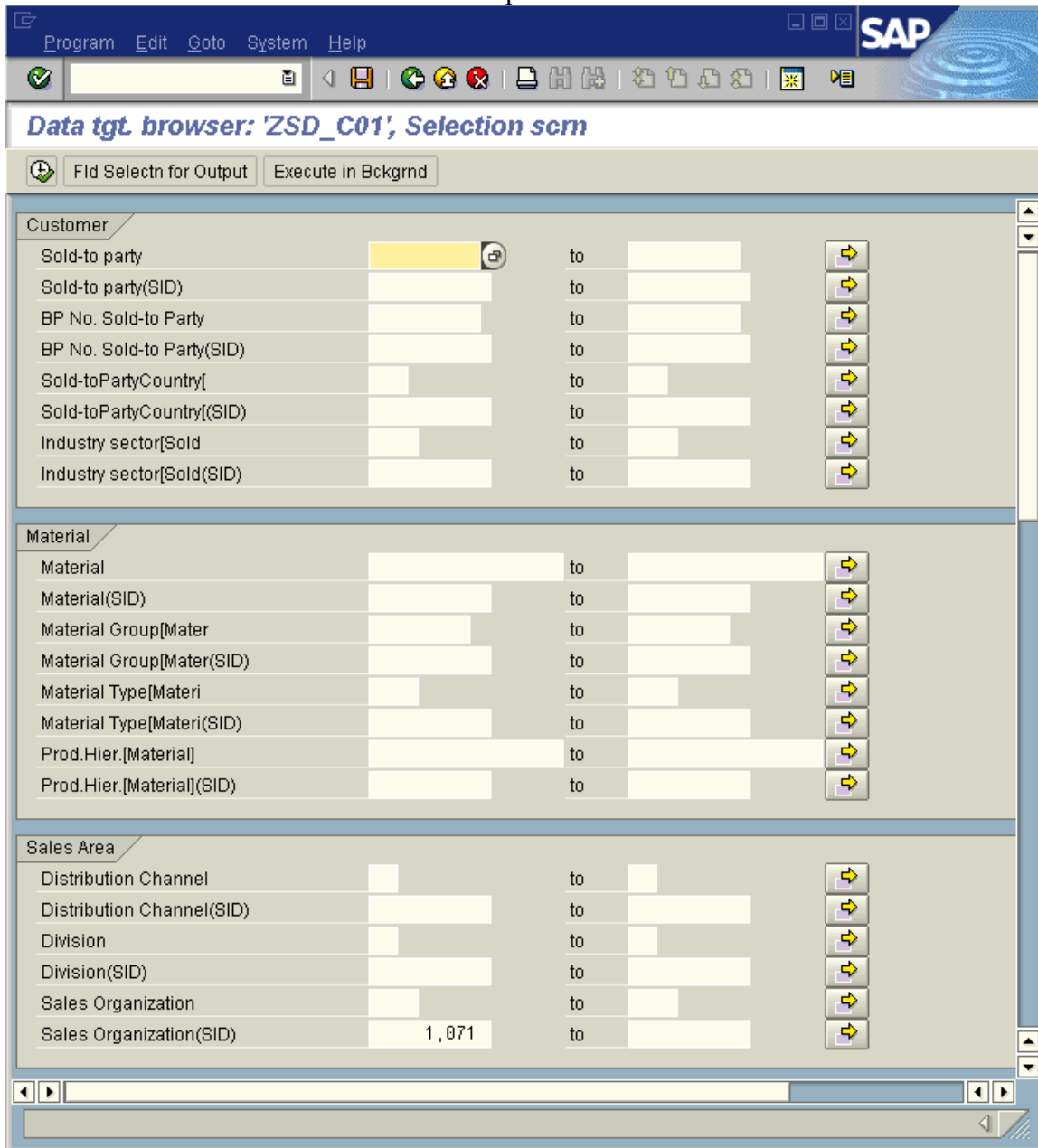


Figure 108: LISTCUBE - enter selection criteria

Since there is only one predicate on dimension 3, and it is a SID value, use the SID value from the SQL statement as the only selection criterion.

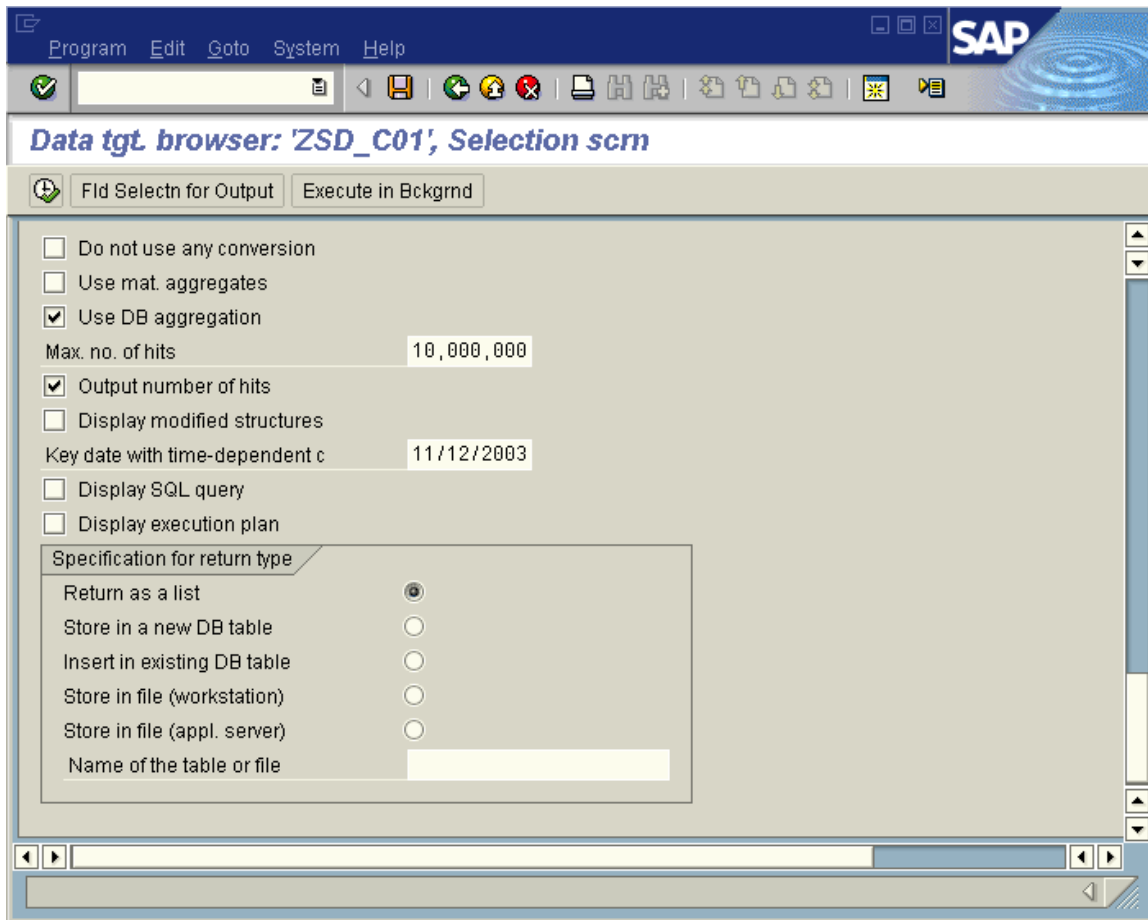


Figure 109: LISTCUBE options

Turn off 'Use mat. Aggregation', so that aggregates will not be used, and turn on 'Use DB aggregation', to summarize the rows. Press 'Fld Selectn for Output'.

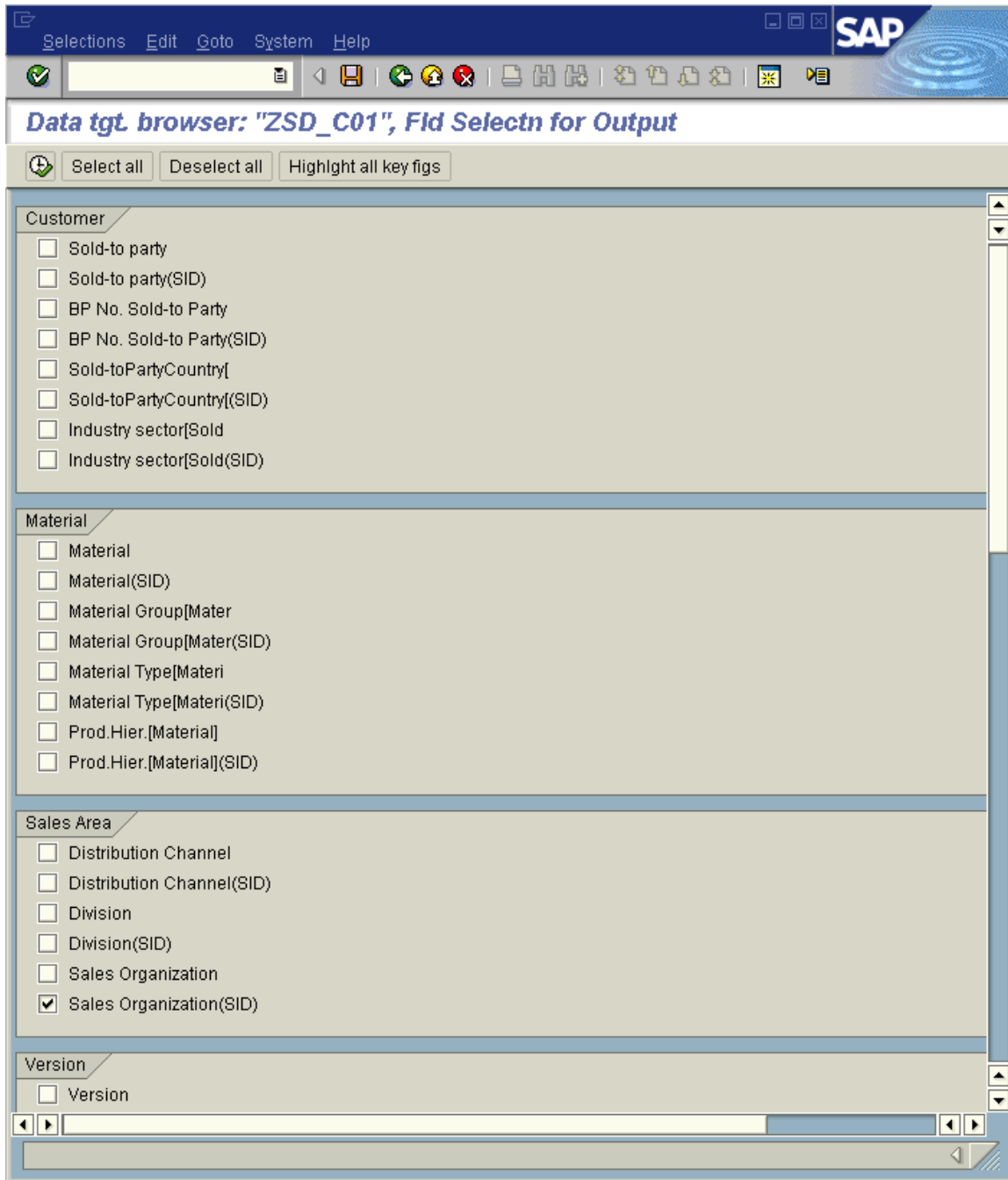


Figure 110: LISTCUBE output field selection

Select only the field that is being used in the selection conditions.

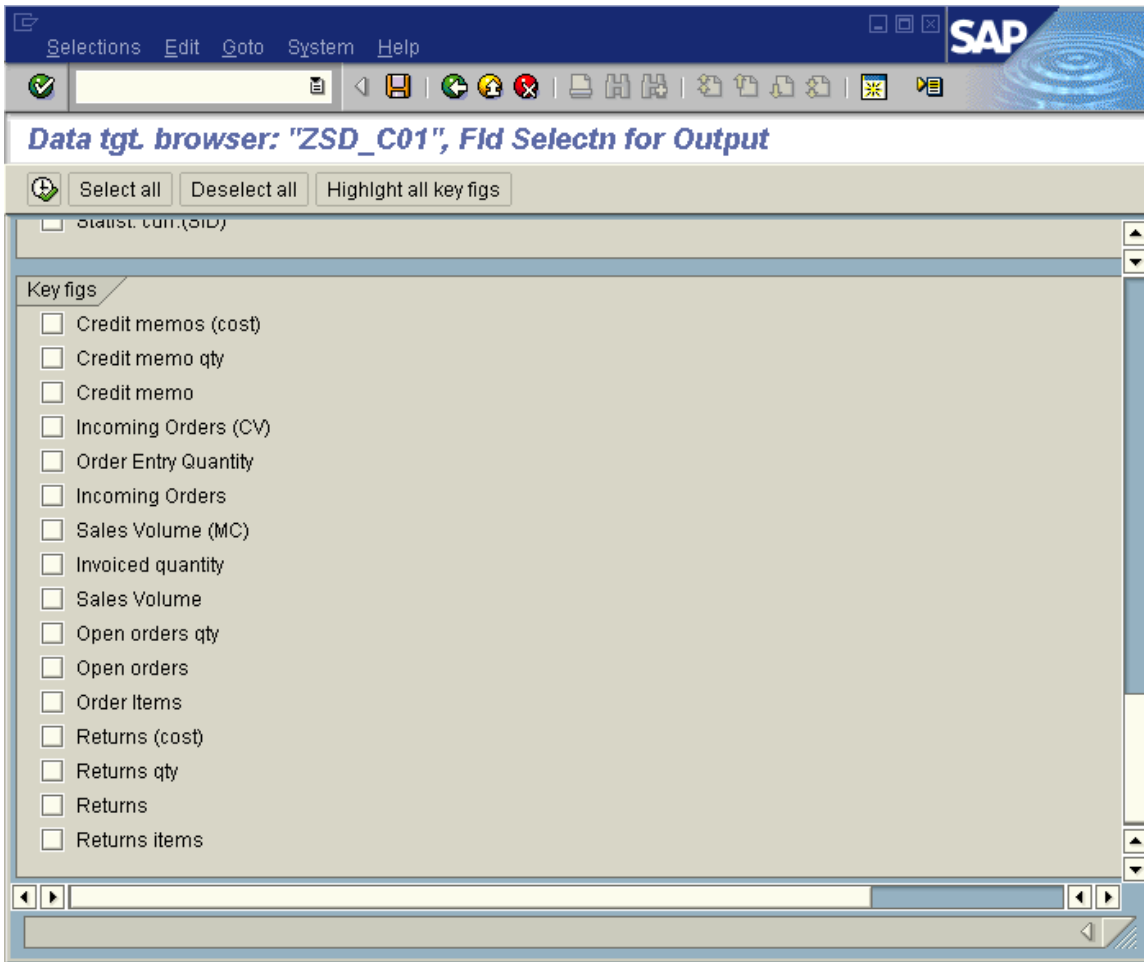


Figure 111: LISTCUBE output selection

De-select the key figures, since we only want the row count.

Execute.

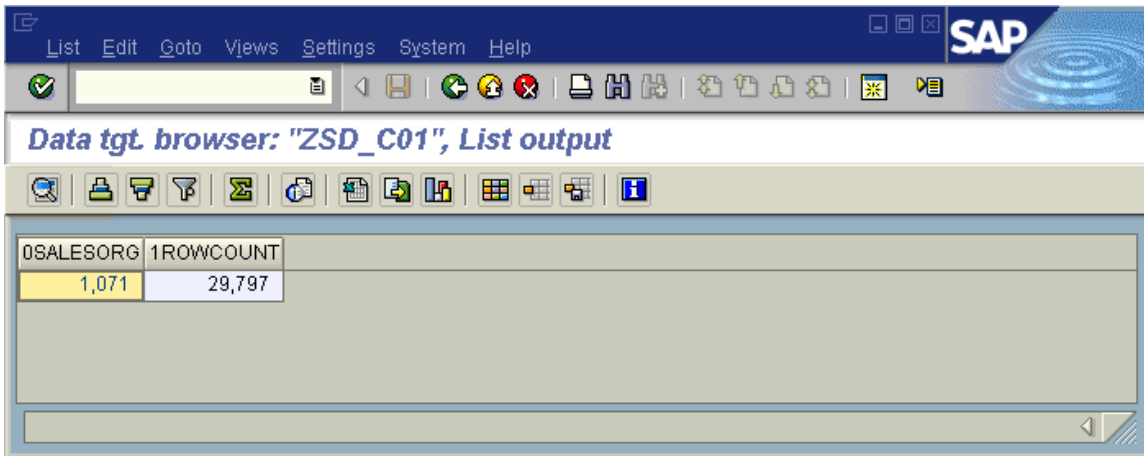
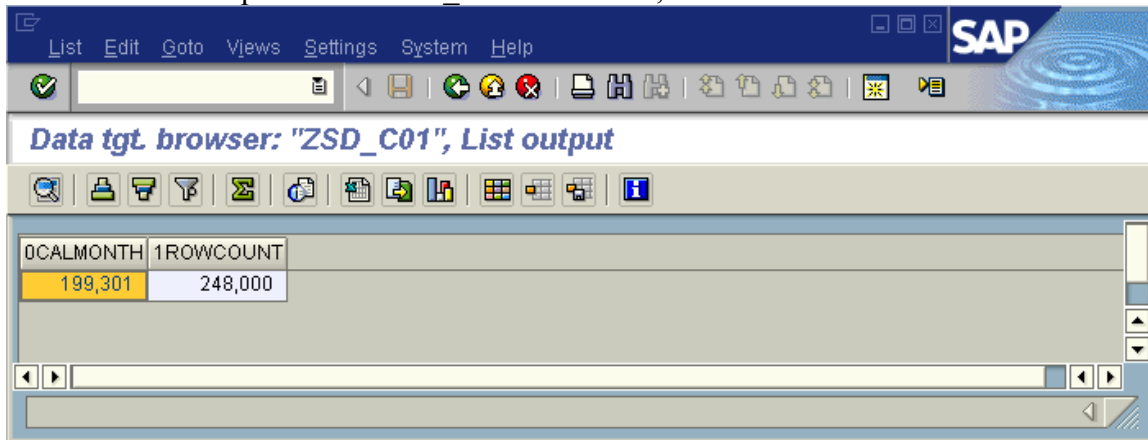


Figure 112: LISTCUBE 0SALESORG count

There are nearly 30,000 rows in the fact table that match SID_0SALESORG = 1071.

If we do the same process for SID_0CALMONTH, we find more rows.



The screenshot shows the SAP Data Target Browser interface. The title bar reads "Data tgt. browser: 'ZSD_C01', List output". Below the title bar is a toolbar with various icons. The main area displays a table with two columns: "0CALMONTH" and "1ROWCOUNT". The first row of data shows "199,301" under "0CALMONTH" and "248,000" under "1ROWCOUNT".

0CALMONTH	1ROWCOUNT
199,301	248,000

Figure 113: LISTCUBE SID_0CALMONTH row count

Different indexes will have different cluster ratios, and thus will be more or less efficient for table access, so the count of rows is not the only factor to evaluate. However, it can give you an idea as to which dimension(s) might offer the best performance, when joined before the fact table.

11.4.8. Check for symptoms of I/O constraints

From within SAP, ST04 ‘global times’ can be used as an indicator of whether there are system-wide I/O constraints.

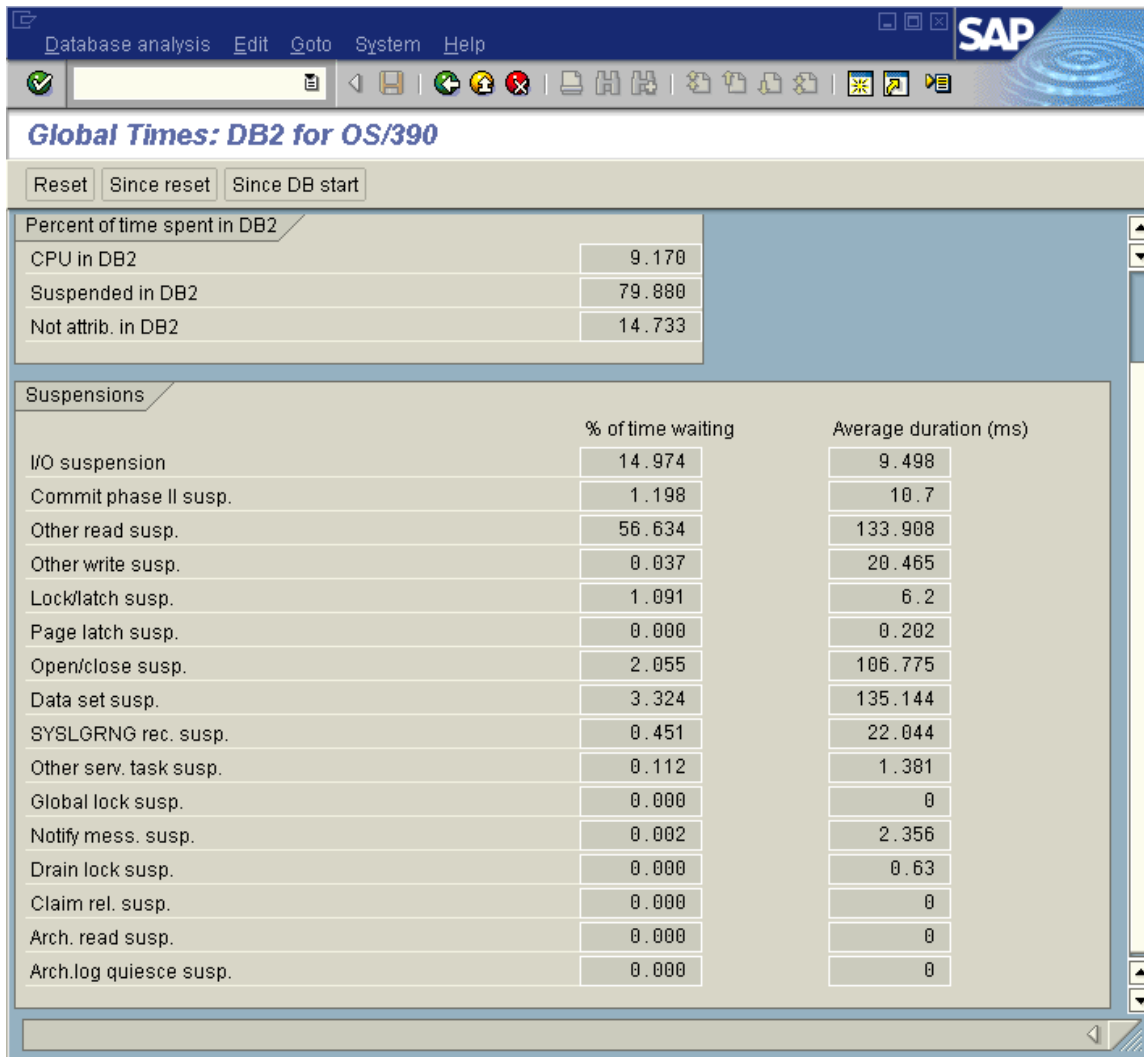


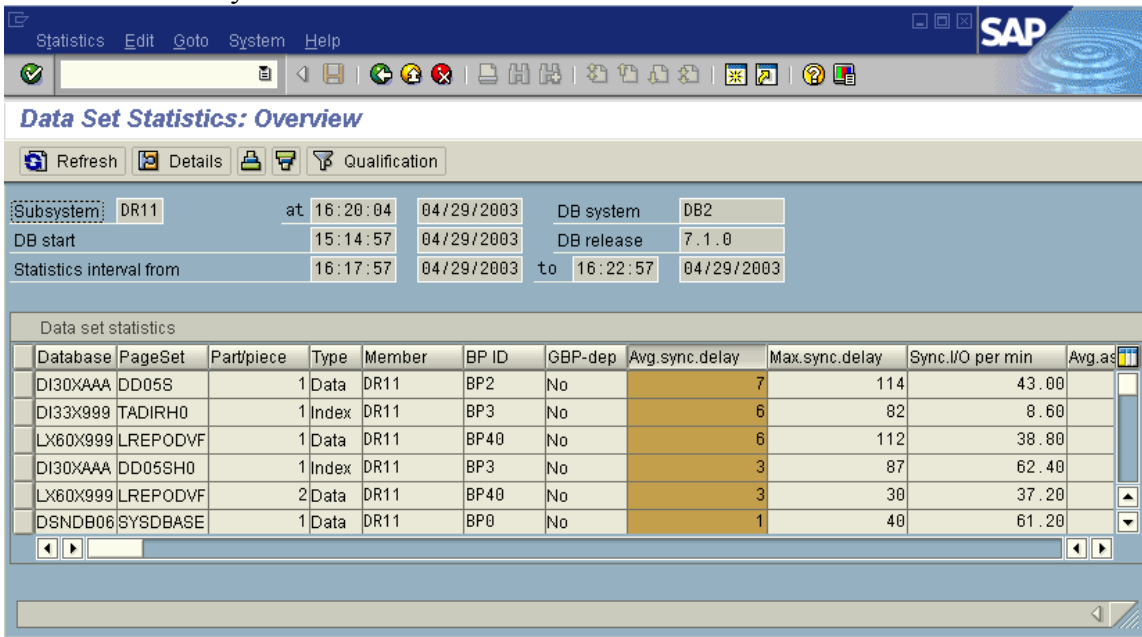
Figure 114: ST04 global times with long 'other read' duration

ST04 “global times” is a summary of the DB2 delay times of active threads in DB2. (See the DB2 administration guide, referenced in section 13.1 for details of delay categories). Since threads terminate and start while SAP is running, the “global times” data cannot be correlated with any specific time period. It is, however, useful as an indicator of recent I/O performance in DB2.

For DB2 subsystems running SAP BW, it is normal for “other read susp” to be the main cause of delay in DB2. “Other read susp” is mostly prefetch I/O for sequential, list, and dynamic prefetch. On a system with good I/O performance, CPU in DB2 will often be 35%-50%, where here it is 9.17%. Together with the high ‘other read susp.’ above, it indicates an I/O constraint.

Checking the ‘Average duration’, note that the average “other read susp” event is 133 ms. The average I/O times for prefetch I/O were 133 ms, which is very long.

For shorter intervals if using the RFCOSCOL-based ST04, one can also check ST04 dataset statistics to review I/O activity and times.



Database	PageSet	Part/piece	Type	Member	BP ID	GBP-dep	Avg.sync.delay	Max.sync.delay	Sync.I/O per min	Avg.as
DI30XAAA	DD05S	1	Data	DR11	BP2	No	7	114	43.00	
DI33X999	TADIRH0	1	Index	DR11	BP3	No	6	82	8.60	
LX60X999	LREPODVF	1	Data	DR11	BP40	No	6	112	38.80	
DI30XAAA	DD05SH0	1	Index	DR11	BP3	No	3	87	62.40	
LX60X999	LREPODVF	2	Data	DR11	BP40	No	3	30	37.20	
DSNDB06	SYSDBASE	1	Data	DR11	BP0	No	1	40	61.20	

Figure 115: ST04 Data Set Statistics

Having seen symptoms of I/O constraints, the next step would be to use OS/390 performance tools such as RMF monitor I or RMF monitor III to examine I/O performance, and find disks with excessive response time.

The solution to I/O performance problems could be:

- Create aggregates, for frequently run queries where QDBSEL/QDBTRANS is high
- Implement PAV functionality on DASD to get more I/O parallelism on active volumes
- Move active volumes to different controllers or storage subsystems, to distribute I/O activity
- Partition active tables (using SAP partitioning), to distribute I/O to more volumes
- Implement DB2 hardware compression, to increase the number of rows per data page and reduce I/O
- Create indexes to enable more efficient SQL access path

11.4.9. Check DB2 buffer pools for signs of constraints

First, it is important to separate the fact tables, and their indexes, out of the default SAP table (BP2) and index (BP3) buffer pools. Infocubes and aggregates are large, and processed in a way that makes it unlikely that their data will be re-referenced frequently, so DB2 caching of these objects in buffer pool or hiper pool is not generally effective.

SAP note 536074 describes a process for implementing buffer pool isolation with BW using the SAP data dictionary. If table and index buffer pool assignments are changed without using the SAP DDIC, and the object is dropped and re-created, then the object will be placed back in its SAP default buffer pool.

The SAP ST04 transaction summarizes several indicators of buffer pool and performance problems into a single field, “Buffer pool shortage”. Monitor this field, and if it shows “YES”, then drill into the buffer pools to examine which one is having the shortage.

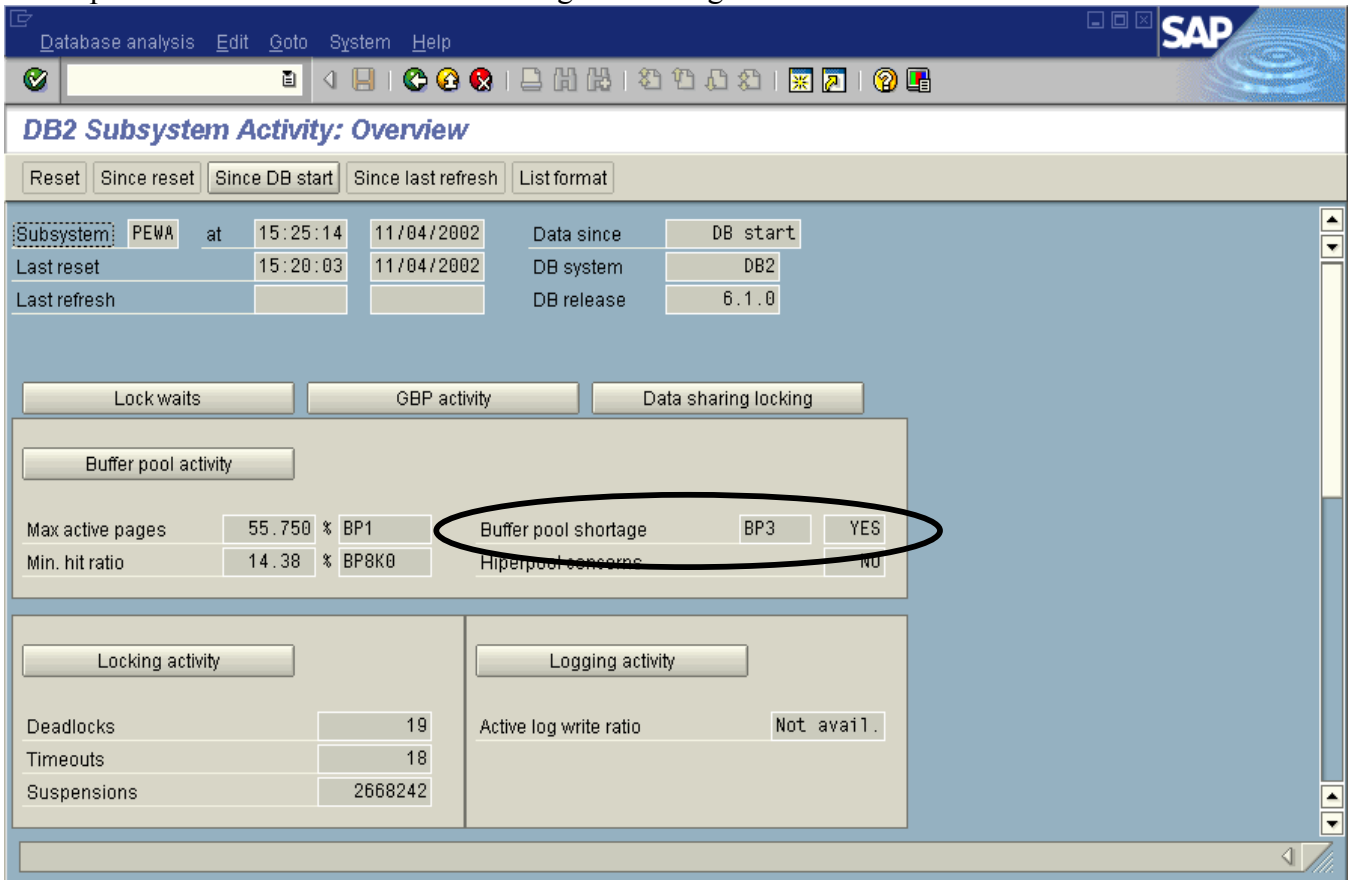


Figure 116: ST04 subsystem activity with bufferpool shortage

The indicators for each bufferpool are summarized in the field “virtual pool shortage”. Use F1 in this field to get a description of the different DB2 indicators summed here.

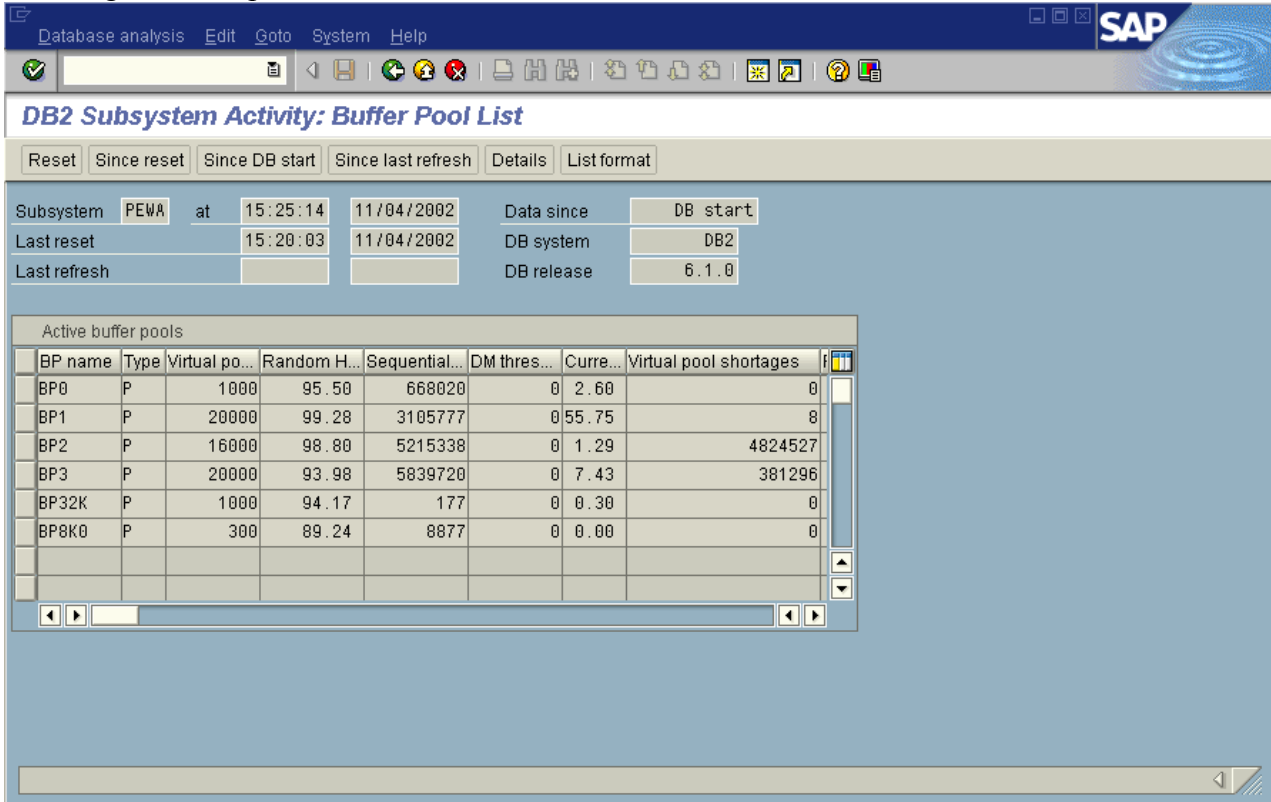


Figure 117: ST04 buffer pool list with virtual pool shortage

Here, BP2 had a very high count, so we use “Details” to examine the cause.

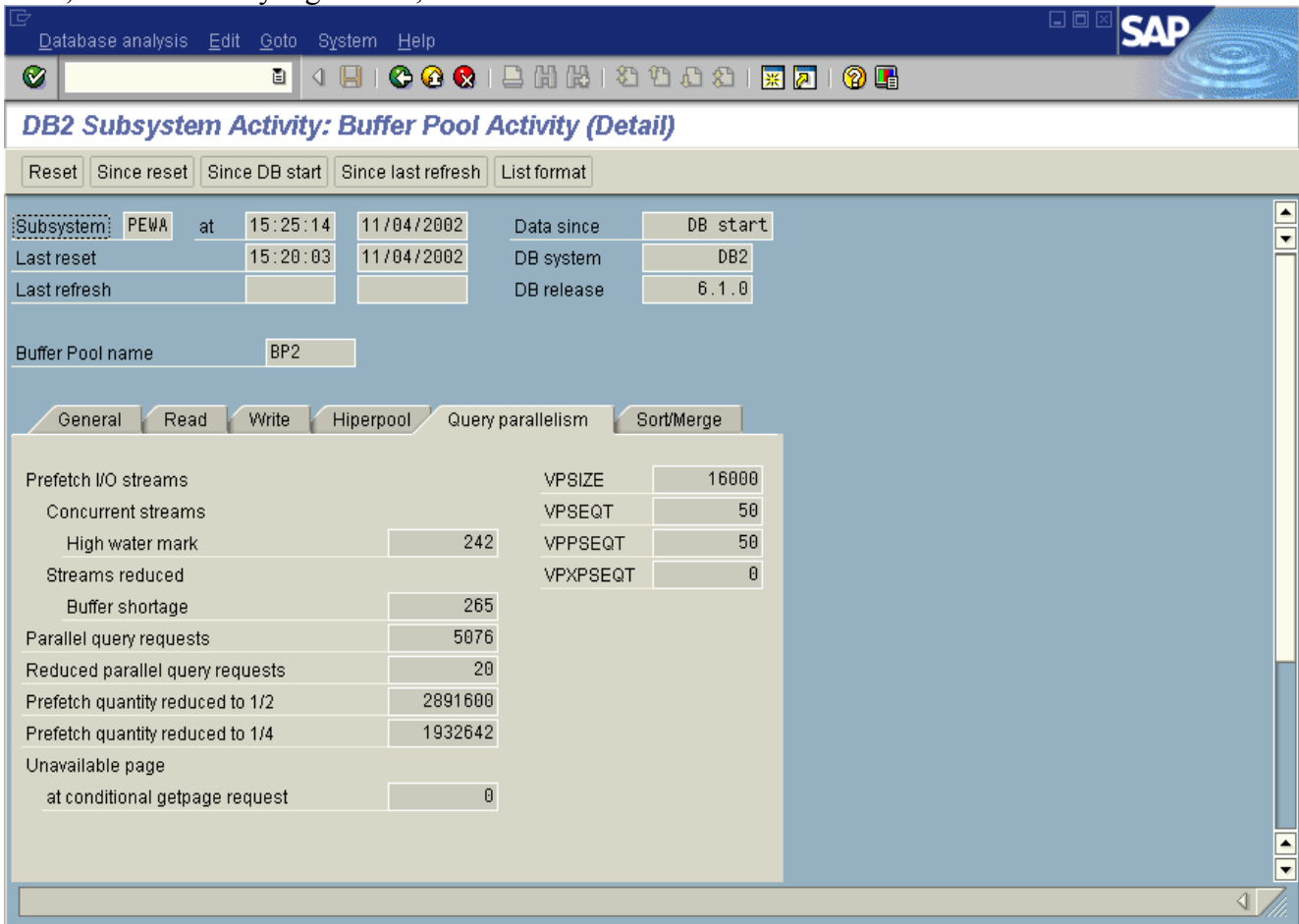


Figure 118: ST04 buffer pool detail for query parallelism reductions

There are several categories of problems that can occur – reduction in parallelism, reduction in prefetch, and reduced streams. See the “DB2 Administration Guide” and “DB2 PM Report Reference” for details.

In Figure 118, the counters for “Prefetch quantity reduced to ½” and “prefetch quantity reduced to ¼” are both very high. When prefetch quantity is reduced, DB2 must do more I/O operations to read the data, which can cause excessive load on the DASD.

There are also indicators for reductions in parallelism, such as “Reduced parallel query requests”. Here, less than one percent of parallel query operations (20/5076) were reduced.

While DB2 may reduce parallelism or prefetch when the bufferpool is too small, hitting these limits can also be a symptom of I/O or query problems. For example, if there are I/O performance problems, then queries will run longer than they should, and there will then be more queries concurrently running. Check that I/O performance is good, before increasing the VPOOL and VPPSEQT parameters. Likewise, if suitable aggregates do not exist, then there will be many queries using infocubes rather than aggregates, and this will cause increased use of DB2 bufferpool and sort areas, compared to queries using the summarized aggregates.

As a rule-of-thumb on sizing the bufferpools to optimize parallelism, check that the VPPSEQT buffer count is larger than (or at least close to) “concurrent streams high water mark” * 32. VPPSEQT is a percentage of VPSEQT, which is a percentage of VPOOL. In Figure 118, $VPOOL * VPSEQT * VPPSEQT$ ($16000 * 0.50 * 0.50$) is 4,000. “Concurrent streams high water mark” * 32 = $242 * 32 = 7744$. In this case, assuming that the limits are not being hit because of I/O or query problems, we would need to enlarge VPOOL, and increase VPPSEQT to enlarge the buffers contained in the VPPSEQT area.

Bufferpool storage requirements will change, and often decrease, as aggregates are defined, since the data can usually be retrieved more efficiently from an aggregate than from an infocube.

The ST04 statistics are available “since startup”, and “since reset”. If the “since startup” counters indicate performance problems, one must review when the problem occurred, and how often it occurs. This can be done using DB2PM, or a similar tool for historical DB2 reporting.

12. Appendix 1: correlating RSDDSTAT and ST04 before 3.0B SP 15

In this example, the query uses the aggregate fact table 100090. Since there is data in both the E and F fact tables for this aggregate, a single query shown in RSDDSTATAGGRDEF will generate two SQL statements.

This query has not been chosen as a performance problem, it has been chosen to show how a single query can generate two SQL statements. QDBSEL/QTIMEDB is about 600, so DB performance is OK. QDBSEL/QDBTRANS is about 10, so an aggregate would speed up this query.

Here, we start from RSDDSTAT and then search for the corresponding statement cache statistics using ST04.

STATUID	INFOCUBE	UNAME	QAGGRUSED	QDBSEL	QDBTRA...	QTIMEDB	STARTTIME	NAV
5883FE9BVXLZESZTD1LE9QEVS	DPEUCC01			84,368	40,352	756.324219	20,021,004,134,319.9770000	850
CJN1QN0P9PLJMX6NQZCPE1C...	SDEUPFOR		100090	66,911	6,275	103.390625	20,021,004,134,258.7200000	8V7
AQ8HQM8PIMYQPOEGC519JOX...	COEUPPR01			22,206	1	61.531250	20,021,004,134,137.5790000	A07
57L78JVVW81FM8NRWQ5113RV...	SDEUPCBIL			7,303	777	37.453125	20,021,004,132,640.6960000	E11
BDJAGB08M2K5C0WLVX6YNM...	COEUPPR01			22,206	1	75.687500	20,021,004,132,615.6960000	CW

Figure 119: Aggregate 100090 RSDDSTAT

STATUID	AGGRNUM	IOBJNM	QUERYCUBE	AGGRST	HIESID	TLEVEL	VALUE	FIXSID
CJN1QN0P9PLJMX6NQZCPE1C7R		0CUSTOMER__EINTCOIND	100090	F	0	02		2
CJN1QN0P9PLJMX6NQZCPE1C7R		0CUST_SALES__ECUSTHR2	100090	F	0	0018501418	10,668	
CJN1QN0P9PLJMX6NQZCPE1C7R		0CUST_SALES__ECUSTHR5	100090	*	0	1310	0	
CJN1QN0P9PLJMX6NQZCPE1C7R		0MATERIAL__EPRODH02	100090	*	0	1310	0	
CJN1QN0P9PLJMX6NQZCPE1C7R		0MATERIAL__EPRODH06	100090	*	0	1310	0	
CJN1QN0P9PLJMX6NQZCPE1C7R		0SALESORG	100090	F	0	1310	3	
CJN1QN0P9PLJMX6NQZCPE1C7R		EINVQTYCA			0		0	
CJN1QN0P9PLJMX6NQZCPE1C7R		EINVQTYM3			0		0	
CJN1QN0P9PLJMX6NQZCPE1C7R		EINVQTYSU			0		0	

Figure 120: Aggregate 100090 RSDDSTATAGGRDEF

Note that QDBTRANS is 6,275 in Figure 119. We will use the QDBTRANS and ST04 “rows processed” to verify that we have the right statements in the statement cache. Check the STARTTIME field, 13:42:58, as it will be used to match the statements in ST04 statement cache.

When analyzing performance problems for a query with both E and F fact tables, there is one gap in the available statistics. There is no ST04 statement cache indicator equivalent to QDBSEL. We can use ST04 'rows processed' to determine how many rows a statement processed after the 'GROUP BY' clause, but cannot see which statement had the most rows that satisfied the predicates (QDBSEL). In the following example, one of the statements runs much longer than the other. The statement may run longer because it had more rows that satisfied the predicates, or it may run longer because of a performance problem. We would have to examine the statement cache statistics and the access path in ST04 'explain plan' to determine where the problem may be.

Go into ST04 statement cache, and use the 'strings' filter to search for SQL referencing the aggregate table. This will make it easier to find the statements referencing the aggregate. Also, check the 'Statement text' (see Figure 122) to look for multiple copies of the same statement. If a query is run against an infocube with E and F fact tables, the same columns will be in the statement text of each statement.

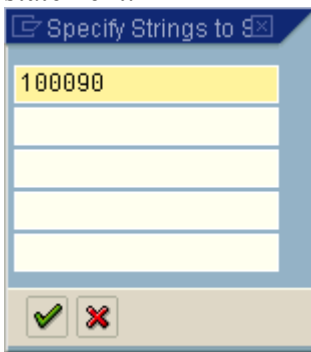


Figure 121: ST04 'String' filter

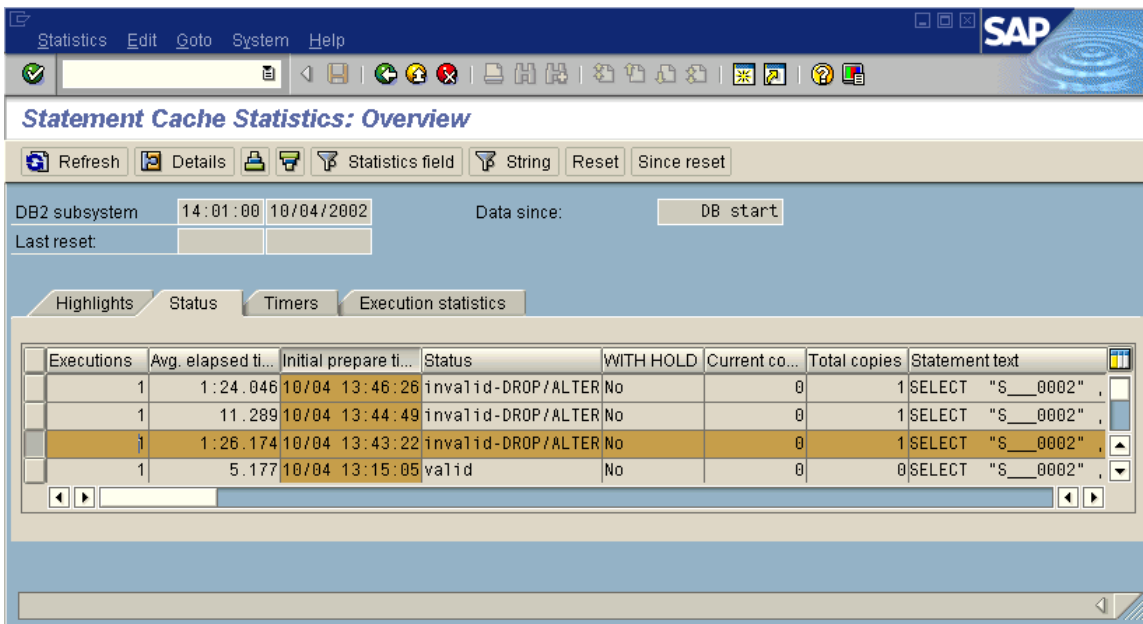


Figure 122: Filtered ST04 statement cache

The statements in the cache nearest to the RSDDSTAT timestamp have elapsed time 11.289 and 1:26.147. Remember that RSDDSTAT is application server time, and GMT. The ST04 'Initial prepare time' is based on DB server time, and in this example is also GMT. The DB server time can be a few seconds off from application server time.

13. Appendix 2: Reference Materials

13.1. *IBM manuals*

Planning Guides, which contain detailed description of architecture of SAP to DB2 connection:

SC33-7966-00 "SAP R/3 on DB2 for OS/390: Planning Guide SAP R/3 Release 4.6A"

SC33-7966-01 "SAP R/3 on DB2 for OS/390: Planning Guide SAP R/3 Release 4.6B"

SC33-7966-02 "SAP R/3 on DB2 for OS/390: Planning Guide SAP R/3 Release 4.6C"

SC33-7966-03 "SAP R/3 on DB2 for OS/390: Planning Guide SAP R/3 Release 4.6D"

SC33-7959-00 "SAP on DB2 UDB for OS/390 and z/OS: Planning Guide" (SAP Web AS 6.10)

SC33-7959-01 "SAP on DB2 UDB for OS/390 and z/OS: Planning Guide" (SAP Web AS 6.20)

DB2 Administration Guides, which contain detailed description of DB2 access paths, prefetch capabilities, buffer pool parameters, and components of DB2 elapsed time:

SC26-9003-02 "DB2 Universal Database for OS/390: Administration Guide" (DB2 V6)

SC26-9931-01 "DB2 Universal Database for OS/390 and z/OS: Administration Guide" (DB2 V7)

DB2 PM Report Reference, which contain details of DB2 PM reporting, and descriptions of events (hit rates, limits exceeded) monitored by DB2

SC26-9164 "DB2 Performance Monitor for OS/390 Report Reference Volume 1"

SC26-9165 "DB2 Performance Monitor for OS/390 Report Reference Volume 2"

Figure 1: RSA1 definition of line item dimensions	9
Figure 2: Naming Conventions	11
Figure 3: Simplified SAP infocube query structure	12
Figure 4: Query designer	13
Figure 5: E fact table access via PAGE_RANGE = Y	17
Figure 6: ODS object key structure	18
Figure 7: ST04 statement statistics - parallelism used	20
Figure 8: ST04 statements invalid due to dropped VIEW	21
Figure 9: Statement using view to access infocube or aggregate	22
Figure 10: ST04 statement cache with BW query correlation	25
Figure 11: ST04 BW analysis	25
Figure 12: ST04 statement cache BW InfoObjects	26
Figure 13: RSDDSTAT slow QDBSEL/second with QDBSEL similar to QDBTRANS	28
Figure 14: RSDDSTAT QDBSEL much larger than QDBTRANS	29
Figure 15: RSDDSTAT to get STATUID	30
Figure 16: RSDDSTATAGGRDEF selection screen	31
Figure 17: RSDDSTATAGGRDEF query information	31
Figure 18: Multicube RSDDSTAT entry	32
Figure 19: Multicube RSDDSTATAGGRDEF display	33
Figure 20: probably incorrect read mode setting	33
Figure 21: RSRREPDIR - search for queries with incorrect read mode	34
Figure 22: Query cache settings	34
Figure 23: RSRT main screen	35
Figure 24: RSRT 'execute and debug' options	36
Figure 25: RSRT statistics	37
Figure 26: Statement cache – specify selection criteria	38
Figure 27: ST04 cache selected by STATUID	38
Figure 28: ST04 statement cache BW statistics	38
Figure 29: statement details	39
Figure 30: ST04 explain	40
Figure 31: Plan table	40
Figure 32: Search RSDDSTAT by name and date for query statistics	41
Figure 33: Choose fields	42
Figure 34: RSDDSTAT query statistics	43
Figure 35: RSDDSTAT Details	43
Figure 36: Query using sparse indexes on workfiles	44
Figure 37: ST03N infocubes	46
Figure 38: ST03N queries by DB time	47
Figure 39: XL spreadsheet to highlight slow queries	48
Figure 40: RSDDSTATAGGRDEF display of query using navigation attributes	50
Figure 41: Explain plan of query on aggregate containing navigation attributes	50
Figure 42: Aggregate fact table information	51
Figure 43: Aggregate fact table SQL statement	52
Figure 44: Aggregate fact table RSA1 display	53
Figure 45: Aggregate fact table SE11 display	54

Figure 46: RSDDIMEIOBJ - dimensions and characteristics	55
Figure 47: aggregate containing 0MATERIAL characteristic.....	56
Figure 48: query contains navigation attributes on 0MATERIAL	57
Figure 49: DB2 join attribute SID table to aggregate for navigation attribute processing.....	57
Figure 50: Time-dependent aggregate.....	58
Figure 51: BW 3.0 Aggregate without time-dependent characteristics.....	59
Figure 52: aggregate containing both characteristic and navigation attribute on characteristic.....	60
Figure 53: SE11 display of customer dimension on aggregate 100131	61
Figure 54: RSA1 modeling data targets.....	63
Figure 55: RSA1 maintain aggregates.....	64
Figure 56: RSA1 parameters for aggregate proposal	64
Figure 57: RSA1 proposed aggregates	65
Figure 58: Z_SLS_ORD roll-up hierarchy part 1	66
Figure 59: Z_SLS_ORD rollup hierarchy part 2.....	67
Figure 60: Z_SLS_ORD roll up hierarchy part 3.....	68
Figure 61: Evaluate aggregate merge - STAT 58 - dimensions 1, 3, 5, and T	69
Figure 62: Evaluate aggregate merge - STAT 60 - dimensions 1, 2, 5	70
Figure 63: Definition of aggregation levels	71
Figure 64: STAT 3 and STAT 4 comparison for merge.....	71
Figure 65: Aggregate hierarchy	72
Figure 66: Deep roll-up hierarchy STAT 191.....	73
Figure 67: Deep roll-up hierarchy STAT 178.....	74
Figure 68: Deep roll-up hierarchy STAT 154.....	74
Figure 69: Custom report from RSDDSTAT RSDDSTATAGGRDEF join.....	76
Figure 70: View joining RSDDSTAT and RSDDSTATAGGRDEF	77
Figure 71: columns in RSDDSTAT RSDDSTATAGGRDEF view.....	78
Figure 72: SE16 using RSDDSTAT/RSDDSTAGAGGRDEF view.....	78
Figure 73: SE16 display with RSDDSTAT/RSDDSTATAGGRDEF view	79
Figure 74: ST03N - query “MSDEUPBIOB_I_PUB_500” with high select / trans ratio.....	80
Figure 75: Sample MIN 0 proposed aggregate.....	81
Figure 76: Rollup hierarchy of proposed aggregate.....	82
Figure 77: RSRT cache monitor.....	83
Figure 78: RSRT cached queries.....	84
Figure 79: ST04 correlation example	85
Figure 80: BW statistics correlation example	85
Figure 81: ST04 details correlation example	86
Figure 82: ST04 correlation example	87
Figure 83: parallelism prompt.....	87
Figure 84: correlation example explain	88
Figure 85: plan table correlation example	88
Figure 86: SE11 display of E fact table.....	89
Figure 87: ST04 statement after correlation stats.....	90
Figure 88: plan table after correlation stats	90
Figure 89: plan table for example of increasing freqval count	91
Figure 90: ST04 times for freqval count example.....	92
Figure 91: SQL for freqval count example.....	93

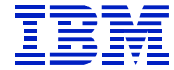


Figure 92: plan table after increasing COUNT in FREQVAL 94

Figure 93: ST04 plan table display for material group example..... 95

Figure 94: Statement for material group example..... 96

Figure 95: check indexes for material group example 97

Figure 96: ST04 Table Information for material group..... 97

Figure 97: Cardinality information for material group..... 98

Figure 98: Material group column distribution..... 99

Figure 99: ST04 plan table with material group index..... 99

Figure 100: Dimension 1 data model problem..... 100

Figure 101: F fact table - dimension table cardinality 25% of fact cardinality 101

Figure 102: E fact table - dimension cardinality less than 10% of fact cardinality 102

Figure 103: ST04 statement cache - examine predicates..... 103

Figure 104: ST04 explain - table cardinality information 104

Figure 105: Find dimension from join conditions..... 105

Figure 106: RSA1 find characteristic 106

Figure 107: sample SQL statement 107

Figure 108: LISTCUBE - enter selection criteria 108

Figure 109: LISTCUBE options 109

Figure 110: LISTCUBE output field selection 110

Figure 111: LISTCUBE output selection 111

Figure 112: LISTCUBE 0SALESORG count 111

Figure 113: LISTCUBE SID_0CALMONTH row count 112

Figure 114: ST04 global times with long 'other read' duration..... 113

Figure 115: ST04 Data Set Statistics..... 114

Figure 116: ST04 subsystem activity with bufferpool shortage 115

Figure 117: ST04 buffer pool list with virtual pool shortage 116

Figure 118: ST04 buffer pool detail for query parallelism reductions..... 117

Figure 119: Aggregate 100090 RSDDSTAT..... 119

Figure 120: Aggregate 100090 RSDDSTATAGGRDEF..... 119

Figure 121: ST04 'String' filter..... 120

Figure 122: Filtered ST04 statement cache 120