Rapid SAP NetWeaver BW ad-hoc Reporting Supported by IBM DB2 Analytics Accelerator for z/OS

Applies to:

SAP NetWeaver BW running on DB2 for z/OS and zEnterprise. For more information, visit the EDW homepage.

Summary

IBM DB2 Analytics Accelerator for z/OS V2.1 powered by Netezza technology (IDAA) just has been released. In this paper we describe how the IDAA can be integrated into SAP NetWeaver BW. Furthermore, we show that SAP BW ad-hoc reporting experiences a drastic decrease in elapsed time when supported by IDAA.

Authors: Joachim Rese, Georg Mayer

Company: IBM Deutschland Research & Development GmbH

Created on: 25 November 2011

Author Bio

Georg Mayer is a senior software engineer at the IBM Böblingen Laboratory in Germany. He is a member of the joint SAP/IBM platform team responsible for DB2 for z/OS. He joined IBM Data Management in 2001 and has over 22 years of experience with ERP software, databases, OLTP, and performance tuning. For the last 14 years Georg has worked as a development expert for a variety of SAP database porting teams in Walldorf and Rot. Prior to working at IBM, he worked at Informix® and Atos. He holds a master degree in Computer Science and Mathematics from the University of Stuttgart / Germany.

Joachim Rese is a senior software engineer at the IBM Böblingen Laboratory in Germany. He is a member of the joint SAP/IBM platform team For many years, Joachim has been the lead developer for SAP BW on DB2 for z/OS. He has 15 years of experience in the field of SAP on DB2 for z/OS. Joachim holds master degrees in Mathematics and Computer Science from the University of Paderborn / Germany.

Table of Contents

Acknowledgement

Thanks to the following people. This documentation would not be possible without their contribution.

Namik Hrle, Distinguished Engineer,
IBM Deutschland Research & Development GmbH

Patric Becker, Data Warehouse on System z Center of Excellence,
IBM Deutschland Research & Development GmbH

Elisabeth Puritscher, Technical Marketing Competence Center,
IBM Deutschland Research & Development GmbH

Uwe Denneler, Technical Marketing Competence Center,
IBM Deutschland Research & Development GmbH

Dr. Bernd Kohler, Development Manager SAP on IBM System z,
SAP AG

Introduction

Data Warehousing and BI queries as found in SAP BW environments are typically complex and often ad-hoc in nature. There is a common concern about the elapsed times of running these very resource intensive workloads in a native DB2 for z/OS environment. Additionally, Data Warehousing and BI applications increasingly require very fast response times irrespective of the complexity of the queries.
IBM DB2 Analytics Accelerator (IDAA) is a new IBM offering that augments IBM DB2 for z/OS into the first class Enterprise Data Warehouse by accelerating selected queries. It is an appliance based on Netezza technology that can be connected to a System z114 or z196 server. The purpose of the accelerator is to store data being heavily accessed by OLAP queries and execute former long-running queries originating from DB2 for z/OS on the accelerator, without any changes to the application.
The IDAA is completely under control of DB2 for z/OS and acts as another (virtual) resource manager that is not visible to the outside world and connecting applications. DB2 for z/OS code is enhanced to deeply integrate the accelerator and intelligently route eligible queries to this appliance.
In addition to providing OLTP-like performance for OLAP-type queries, IDAA can also significantly reduce typical performance tuning activities.
This paper is a hands-on reference for implementing an InfoCube in IDAA. It describes how the IDAA can be integrated into an existing SAP BW environment running on DB2 for z/OS, maintaining data integrity between DB2 for z/OS and the IDAA with minimal administration overhead.
For more information about the IBM DB2 Analytics Accelerator, please see http://www.ibm.com/software/data/db2/zos/analytics-accelerator.

General Concepts in SAP Business Information Warehouse

SAP Business Information Warehouse (SAP BW) offers a wide spectrum of different data containers, call InfoProvider. The most common one is the *InfoCube*, on which we will focus throughout this paper.

InfoCube

An InfoCube is represented by the *extended star schema* (also known as snowflake schema). In the center resides the fact table, which is linked to multiple dimension tables. The latter are each linked to one or many SID tables, which are linked to master data attribute tables.
In addition, data is stored redundantly in attribute SID tables in order to reduce the number of joins for predicates on navigational attributes. Furthermore, some tables might be created and filled on demand and joined to the star schema.

Figure 1 shows only a simplified view of the extended star schema. In particular, attribute SID tables and master data tables might store time dependent data. Thus these tables are split into two tables, one for time dependent data, the other one for data that is not time dependent. An additional view on both tables is used to retrieve all data.
Fact tables and dimension tables contain transaction data and are individual to an InfoCube (yellow box in Figure 1). All other tables represent master data and are shared between InfoCubes. Thus, these tables appear in multiple different star schemas.

SAP BW implements two fact tables. The so-called *e fact table* is optimized for reporting whereas the second one, called *f fact table*, is designed for data maintenance. When data from an InfoCube is requested, two sql queries on the star schema are performed, one on the e fact table and the other one on the f fact table. Both star schema queries are identical except for the involved fact table.

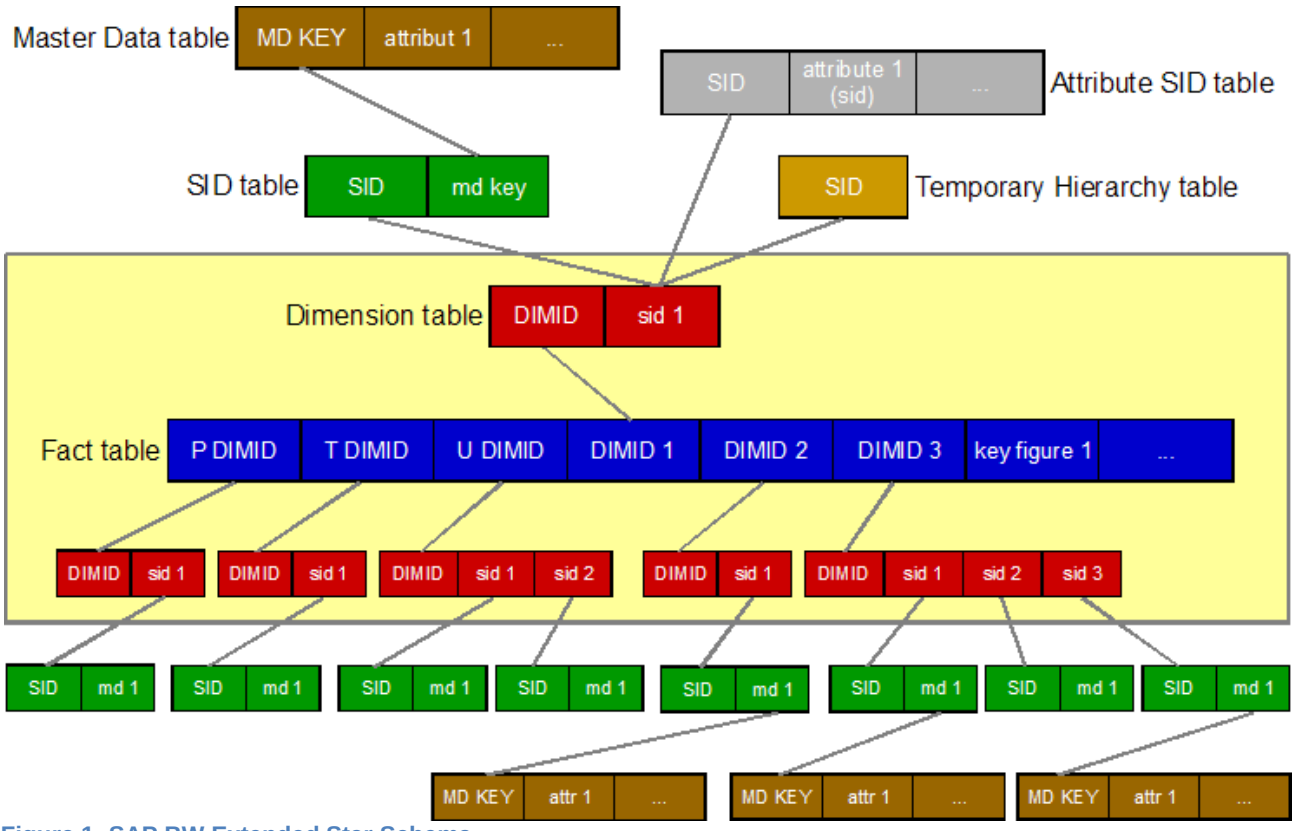

Figure 1: SAP BW Extended Star Schema

InfoCube Compression

The operation of data transfer from f fact table to e fact table is called *InfoCube compression*. InfoCube compression is in particular beneficial if the Infocube is non-cumulative (contains stock data), since the most current stock values are stored in the e fact table, but not in the f fact table. Other benefits of InfoCube compression are potential less data due to aggregation of cancelations and more partition pruning due to partition by time. However, if the data of an Infocube is loaded to IDAA, InfoCube compression becomes much less important and therefore does not need to be performed frequently.

Data Processing with IDAA

In this chapter we explain how we extent the SAP BW ETL processing in order to update the data in IDAA appropriately. The steps described here have been implemented in ABAP reports. See chapter “Integration into SAP BW” for details.

Note: Before implementing any code to your system, please check SAP note 1649284 for latest information about official support of the IBM DB2 Analytics Accelerator by SAP and functionality in SAP products.

SAP BW Parameters for Temporary Tables

First we change some SAP application parameters. SAP BW often stores data temporary in tables (SAP temporary tables) during reporting. These tables are regular tables in DB2 (do not mix up with DB2 global temporary tables). SAP temporary tables are subject to naming scheme /BI0/0t<8-digit-number>, where t is a 1-digit number that indicates the usage type of the table. SAP temporary tables appear in the FROM clause of star schema SQL queries.
Since SAP temporary tables are created and filled on demand by the SAP data manager, i.e. during reporting, they are challenging when a query is supposed to be offloaded to IDAA. Therefore we need to discourage the data manager from using SAP temporary tables. To do so, parameters in table RSADMIN need to be set. Table RSADMIN contains parameter / value pairs. We use ABAP report SAP_RSADMIN_MAINTAIN to update those parameters that are mentioned in the following.

Temporary tables of type 07 (/BI0/07...)

These tables are used by the query splitter functionality. We deactivate the query splitter by setting the according thresholds very high. Thus they are never exceeded. We set the parameter

```
SPLIT_QUERY_TABL_THRES     = 999999
```

to deactivate the query splitter for reporting queries. We do not change the equipollent parameter SPLIT_DATAMART_TABL_THRES for data mart queries, because we assume that queries for data extraction from an InfoCube return a high amount of data. Such queries usually do not benefit from IDAA.

Temporary tables of type 06 (/BI0/06...)

Temporary SID tables are used to store SID values for equal predicates. In order to avoid a huge IN list, SID values are materialized into a temporary table if a given threshold (default 10) is exceeded. The following parameter sets this threshold to a high value and therefore makes the use of temporary SID tables unlikely:

```
DB2_EQSID_THRES = 25000
```

However, if there are more than 25000 SID values in an IN list, a temporary table is created and joined to the star schema. Thus, the query cannot be offloaded to IDAA.

Temporary tables of type 02 (/BI0/02...)

Temporary hierarchy tables are used when a hierarchy is processed. There is no way to avoid or reduce usage of these tables. This documentation explains how hierarchy temporary tables can be loaded to IDAA at reporting time.
Fortunately temporary hierarchy tables are reused and not created anew each time when needed. Only when the amount of temporary hierarchy tables exceeds a given number (called buffer size, default 200), some tables are selected by a least recently used algorithm and then deleted and refilled. We increase the buffer size considerably to reduce the frequency of temporary hierarchy deletion. To do so, the following parameter is set:

```
RSDRH_BUFFER_SIZE = 10000
```

Setup

To load an InfoCube into IDAA, all tables that might be involved in a star schema query must be loaded to IDAA. To determine all those tables, function module RSDU_GET_INFOCUBE_TABLE is called as follows:

```
call function 'RSDU_GET_INFOCUBE_TABLES'
  exporting
    i_infocube      = i_infocube
    i_efacttable    = rs_c_true
    i_ffacttable    = rs_c_true
    i_dimensions    = rs_c_true
    i_sidtables     = rs_c_true
    i_chktables     = rs_c_true
    i_inctables     = rs_c_true
    i_aggregates    = rs_c_false
```

Non-fact star schema tables are all tables that form the star schema except the fact tables. We separate between fact tables and non-fact star schema tables, because we handle them differently.

Setup of non-fact star schema tables

Non-fact star schema tables are not partitioned. Thus, they need to be reloaded each time when they have been updated. Some of these tables are updated whenever something is loaded into the cube, for example the package dimension table. Others are very rarely updated. In particular, when transaction data is loaded into a cube only for known master data, the according master data tables are not changed.
Thus, we want to reload a non-fact star schema table to IDAA only when needed. To achieve this, we create control table ZIBMDB2AA_STATUS that holds the current data status for all non-fact star schema tables.

```
CREATE TABLE "SAPR3"."ZIBMDB2AA_STATUS"
( "TABNAME" VARGRAPHIC(18) NOT NULL WITH DEFAULT ' ',
  "LASTUPDATE"      VARGRAPHIC(26) NOT NULL WITH DEFAULT ' ',
  "STATSINSERTS"    INTEGER        NOT NULL WITH DEFAULT 0,
  "STATSDELETES"    INTEGER        NOT NULL WITH DEFAULT 0,
  "STATSUPDATES"    INTEGER        NOT NULL WITH DEFAULT 0,
  "STATSMASSDELETE" INTEGER        NOT NULL WITH DEFAULT 0
) CCSID UNICODE;
CREATE UNIQUE INDEX "ZIBMDB2AA_STATUS~0"
  ON "ZIBMDB2AA_STATUS" ("TABNAME")
  CLUSTER NOT PADDED;
```

The data status comprises of the last time a table has been loaded to IDAA (attribute lastupdate) and the number of changes, i.e. inserts, deletes, and updates, that have been performed between that time and a previous execution of the RUNSTATS utility. We can probe this information against the corresponding values in the real time statistics tables to determine if there has been any update of the data since it has been loaded to IDAA.

Real time statistics are not collected for a tablespace unless RUNSTATS has been run on corresponding tables. Therefore, we run the RUNSTATS utility for all tablespaces that contain a non-fact star schema table and do not have statistics.

Next we need to add all non-fact star schema tables to IDAA. We use the following stored procedure call to do so:

```
EXECUTE PROCEDURE SYSPROC.ACCEL_ADD_TABLES (
          IN    <accelerator>,
          IN    :xml_tableSpecifications,
          INOUT :xml_message    )
```

Since an arbitrary number of tables can be specified in xml_tableSpecifications, only one stored procedure call is necessary.
When a table is added to IDAA it has status "InitialLoadPending", thus an initial load is pending. We postpone the initial load of non-fact star schema tables to the point in time after the InfoCube load, because at that moment we need to check all relevant tables in IDAA anyway and (re-)load those tables that have been updated.

Setup of temporary hierarchy tables

Temporary hierarchy tables are created by SAP's data manager. Thus, they need to be added to the accelerator at reporting time.
In addition, though not necessary, we can at setup time add all temporary hierarchy tables that are already existing. To find and add all relevant temporary hierarchy tables to IDAA, proceed as follows:

1. Get all include tables by calling function module RSDU_GET_INFOCUBE_TABLES with parameter i_inctables = rs_c_true. Set all other boolean parameters to rs_c_false.
2. Join each include table to table RSDRHLRUBUFFER in order to retrieve temporary hierarchy table names:

```
SELECT DISTINCT BUF.TABLNM
  FROM <include table> INCLTAB,
       RSDRHLRUBUFFER  BUF
 WHERE INCLTAB.HIESID = BUF.HIESID
```

3. Add temporary hierarchy tables to IDAA like described above for non-fact star schema tables.

Temporary hierarchy tables may be reused, i.e. filled with different data. We create control table ZIBMDB2AA_HIER to store information about the data of a temporary hierarchy table in IDAA:

```
CREATE TABLE "ZIBMDB2AA_HIER"
  ( "TABNAME"     VARGRAPHIC(30) NOT NULL WITH DEFAULT ' ',
    "HIESID"      INTEGER        NOT NULL WITH DEFAULT 0,
    "SVER"        VARGRAPHIC(1)  NOT NULL WITH DEFAULT ' ',
    "SID"         INTEGER        NOT NULL WITH DEFAULT 0,
    "RETURNLEVEL" INTEGER        NOT NULL WITH DEFAULT 0,
    "READLEVEL"   INTEGER        NOT NULL WITH DEFAULT 0
  ) CCSID UNICODE;
CREATE UNIQUE INDEX "ZIBMDB2AA_HIER~0"
  ON "ZIBMDB2AA_HIER" ("TABNAME")
  CLUSTER NOT PADDED;
```

Setup of fact tables

When data is loaded into an InfoCube, it is inserted into the f fact table only. The e fact table is not changed until InfoCube compression is performed.
Data is loaded into an Infocube in packages, called *requests*. Each single request is inserted into a dedicated f fact table partition. In other words, the data of every new loaded request resides in a separate and definite f fact table partition. This fact suits perfectly to IDAA's capability to load a single table partition.

A newly loaded request is not visible for BW queries unless it has been released for reporting. We exploit this functionality to hide requests to SAP BW that have not yet been updated in IDAA. By default, requests are automatically released for reporting after they have been loaded successfully. We need to deactivate this functionality. At the SAP BW Administrator's Workbench we switch to InfoCube maintenance and select menu item "Environment" → "InfoProvider properties" → "Change". We select tab "Roll Up" and de-select "Set Quality Status to 'OK' Automatically", see Figure 2. We will set the quality flag to OK manually after all tables that are part of an InfoCube's star schema have been loaded to IDAA with current data.

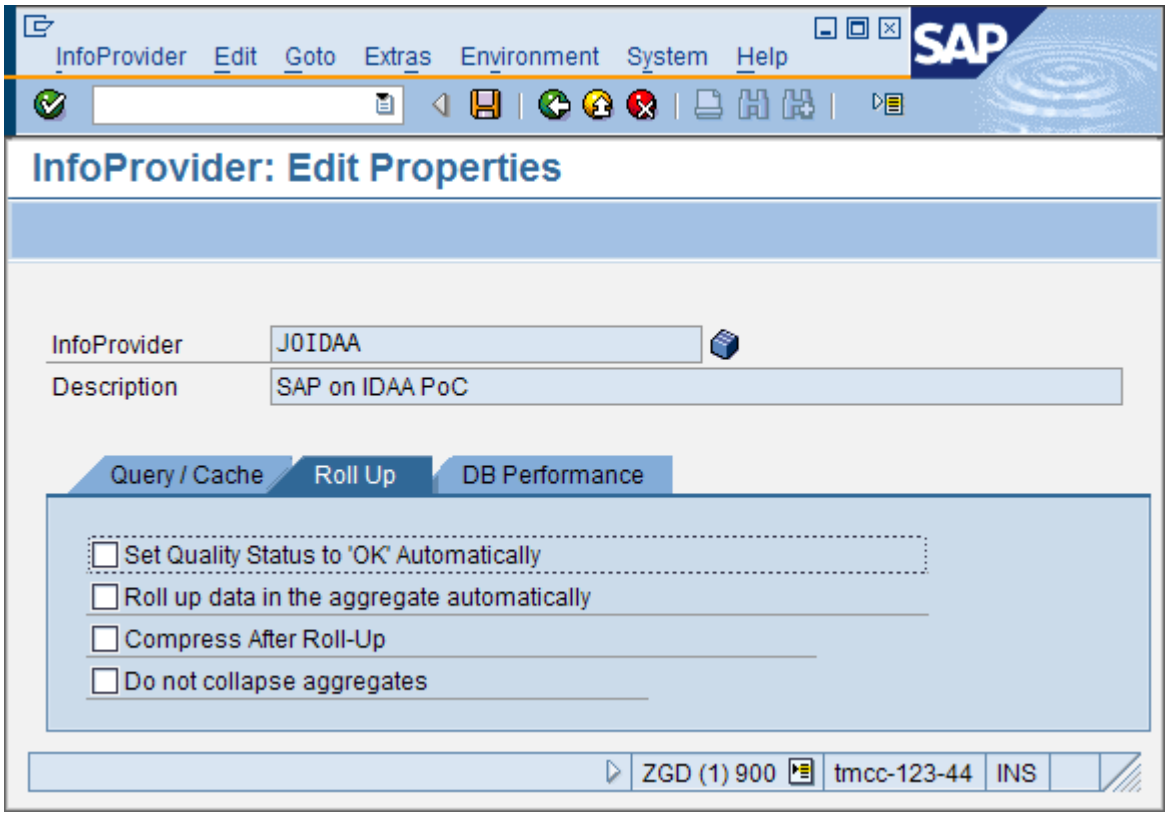

Figure 2: InfoCube Properties

When a request is loaded into an InfoCube, it gets an integer value assigned called *requid* (request id). The requid is always increasing in a sense that each request has a higher requid than all requests loaded before. We create control table ZIBMDB2AA_STATE that we use to track the data status in IDAA based on requid:

```
CREATE TABLE "ZIBMDB2AA_STATE"
  ( "INFOCUBE"  VARGRAPHIC(30) NOT NULL WITH DEFAULT ' ',
    "IDAAOK"    INTEGER        NOT NULL WITH DEFAULT 0,
    "IDAACOMPR" INTEGER        NOT NULL WITH DEFAULT 0
  ) CCSID UNICODE;
CREATE UNIQUE INDEX "ZIBMDB2AA_STATE~0"
  ON "ZIBMDB2AA_STATE" ("INFOCUBE")
  CLUSTER NOT PADDED;
```

The value IDAAOK indicates up to which requid the data of an InfoCube has been loaded to IDAA. The flag IDAACOMPR indicates up to which requid data of the e fact table has been loaded to IDAA. The IDAAOK and IDAACOMPR flags correspond to columns TECHOK and COMPR of table RSMDATASTATE, which holds information about which requests have been released and compressed, respectively.

Update IDAA

After a request has been loaded into an InfoCube, all tables that have been updated need to be reloaded to IDAA. Again, we separate between non-fact star schema tables and fact tables, because the procedure to update IDAA (reload) is different for these two types of tables.

Update of non-fact star schema tables

First we process non-fact star schema tables. Since we access real time statistics, we must ensure that the data in the according tables is current. To materialize the real time statistics data we restart each DB2 tablespace that holds a non-fact star schema table by issuing the following DB2 command:

```
-START DATABASE(<database name>) SPACENAM(<tablespace name>)
```

To determine which tables have been updated since the last load to IDAA, i.e. which table data in IDAA is not in sync with DB2, we compare the information in table ZIBMDB2AA_STATUS with the values in real time statistics table SYSIBM.SYSTABLESPACESTATS. Only those tables for which the following condition is true are eligible for a reload into IDAA:

```
ZIBMDB2AA_STATUS-STATSINSERTS < SYSTABLESPACESTATS.STATSINSERTS or
ZIBMDB2AA_STATUS-STATSDELETES < SYSTABLESPACESTATS.STATSDELETES or
ZIBMDB2AA_STATUS-STATSUPDATES < SYSTABLESPACESTATS.STATSUPDATES or
ZIBMDB2AA_STATUS-STATSMASSDELTE <
                           SYSTABLESPACESTATS.STATSMASSDELETE or
ZIBMDB2AA_STATUS-LASTUPDATE   < SYSTABLESPACESTATS.STATSLASTTIME
```

The last condition is true if the RUNSTATS utility has been run after the last update to IDAA. Since the RUNSTATS utility resets the counters in table SYSTABLESPACESTATS, the actual status of the data in IDAA cannot be determined anymore. Thus, we must reload these tables as well.
In addition, we also load all tables to IDAA which do not have an entry in table ZIBMDB2AA_STATUS at all. To reload all those tables, we execute stored procedure SYSPROC.ACCEL_LOAD_TABLES as follows:

```
EXECUTE PROCEDURE SYSPROC.ACCEL_LOAD_TABLES (
    IN    <accelerator>,
    IN    'TABLE',
    IN    :xml_tablesSetForLoad
    INOUT :xml_msg         )
```

Only one stored procedure call is needed, since an arbitrary number of tables can be specified in xml_tablesSetForLoad.

For each table that has been loaded to IDAA, a flag indicates whether the table is enabled for query acceleration. If this flag is not set to ON, the table is not used by IDAA and hence any BW query that uses that table cannot be offloaded to IDAA. However, the flag can only be set after the table has been initially loaded. Since we do not load any data when we setup the table for IDAA and thus cannot set the flag at setup time, we need to check the flag after load to IDAA and set it, if necessary. Stored procedure SYSPROC.ACCEL_SET_TABLES_ACCELERATION sets the flag:

```
EXECUTE PROCEDURE SYSPROC.ACCEL_SET_TABLES_ACCELERATION (
    IN    <accelerator>,
    IN    'ON',
    IN    :xml_tablesSet,
    INOUT :xml_message
```

Only one stored procedure call is needed, since an arbitrary number of tables can be specified in xml_tablesSet.

After the data has been loaded to IDAA, we update table ZIBMDB2AA_STATUS with the information read from SYSIBM.SYSTABLESPACESTATS. We specify CURRENT TIMESTAMP for column LASTUPDATE.

Update of fact tables

To check the current data status in IDAA for fact tables, we need to compare columns TECHOK and COMPR of table ZIBMDB2AA_STATE with the corresponding data in table RSMDATASTATE. We need to consider three cases:

1. **ZIBMDB2AA_STATE-IDAACOMPR <> RSMDATASTATE-COMPR**
   The InfoCube has been compressed. Both, e fact table and f fact table must be completely reloaded to IDAA. See update of non fact table for the appropriate stored procedure calls.

2. **ZIBMDB2AA_STATE-IDAACOMPR = RSMDATASTATE-COMPR and**
   ZIBMDB2AA_STATE-IDAAOK > RSMDATASTATE-TECHOK
   Requests that have been released for reporting have been deleted from f fact table. We need to reload the whole f fact table (all data partitions).
   Please note that within this study we actually do not support deletion of requests that have been released for reporting since we do not implement a procedure that updates IDAA after request deletion.

3. **ZIBMDB2AA_STATE-IDAACOMPR = RSMDATASTATE-COMPR and**
   ZIBMDB2AA_STATE-IDAAOK < RSMDATASTATE-TECHOK
   This is the most frequent case. One or several requests have been inserted into the f fact table. The according f fact table partitions need to be reloaded to IDAA.
   To find the partition numbers of the newly loaded requids, we perform the following sql statement:

```
SELECT PRT.PARTITION, PDIM.SID_0REQUID AS "RNR_SID"
  FROM SYSIBM.SYSTABLES TAB,
       SYSIBM.SYSTABLEPART PRT,
       "<pdim table>" PDIM
 WHERE TAB.DBNAME = PRT.DBNAME AND
       TAB.TSNAME = PRT.TSNAME AND
       TAB.CREATOR = CURRENT SQLID AND
       TAB.NAME = "<f fact table>" AND
       INTEGER(PRT.LIMITKEY) = PDIM.DIMID AND
       PDIM.SID_0REQUID >  <zibmdb2aa_state-idaaok> AND
       PDIM.SID_0REQUID <= <rsmdatastate-techok>
```

where <pdim table> is the name of the package dimension table and <f fact table> is the name of the f fact table.
We use stored procedure SYSPROC.ACCEL_LOAD_TABLES to load the f fact table partitions to IDAA like follows:

```
EXECUTE PROCEDURE SYSPROC.ACCEL_LOAD_TABLES (
    IN    <accelerator>,
    IN    'PARTITIONS',
    IN    :xml_tablesSetForLoad
    INOUT :xml_msg           )
```

Since an arbitrary number of tables and table partitions can be specified in xml_tablesSetForLoad, only a single stored procedure call is necessary.
In all cases we check whether the loaded tables are enabled for query acceleration. If not, we execute stored procedure SYSPROC.ACCEL_SET_TABLES_ACCELERATION to set the corresponding flag.
After the fact tables have been loaded to IDAA successfully, we update table ZIBMDB2AA_STATE appropriately:

```
UPDATE "ZIBMDB2AA_STATE"
   SET IDAAOK    = <rsmdatastate-techok>,
       IDAACOMPR = <rsmdatastate-compr>
 WHERE INFOCUBE = '<infocube>'
```

where <infocube> is the technical name of the InfoCube.
Finally, the requests that have been loaded to IDAA must be released for reporting. In the Administrator's Workbench we right-click on the InfoCube and select “Manage” from the context menu. Request that have been successfully loaded but not yet released for reporting appear with a yellow traffic light, which can be clicked to be changed to green, i.e. “Status OK”. Figure 3 shows the corresponding dialog panel.

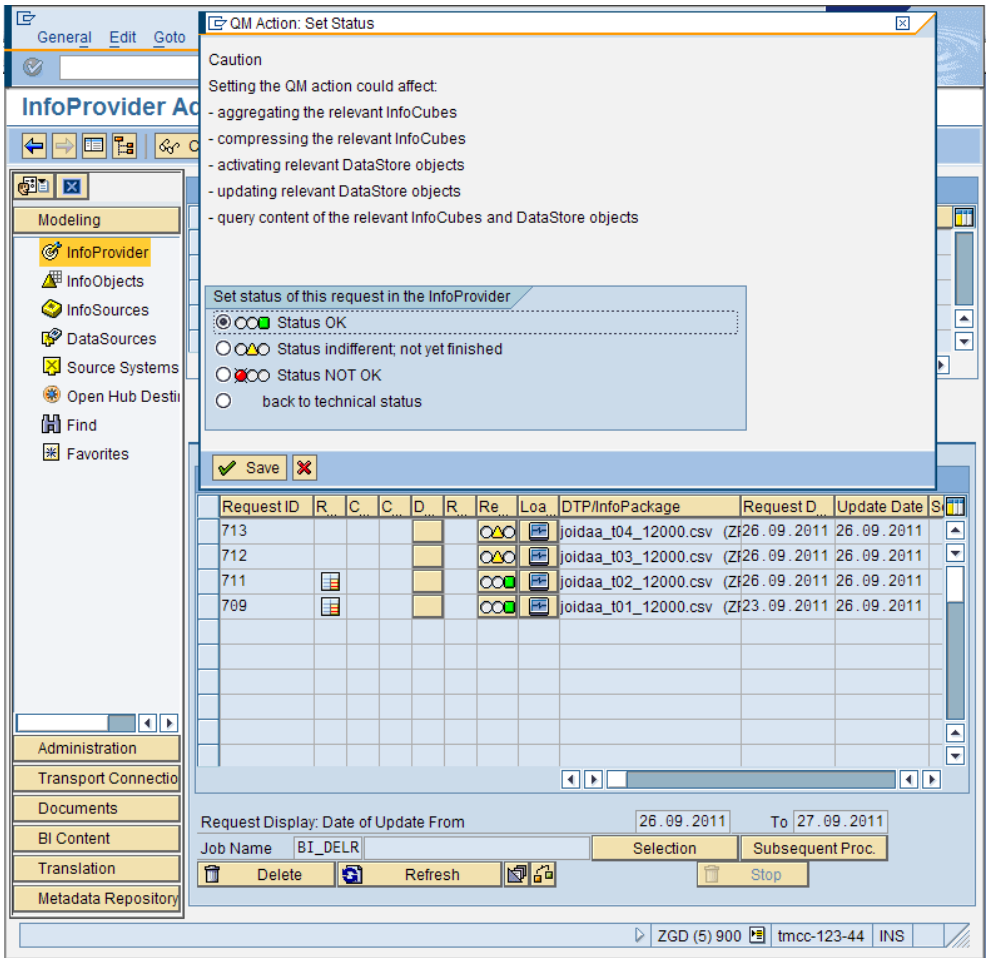

Figure 3: InfoCube Administration

Update of temporary hierarchy tables

Temporary hierarchy tables are created and filled by the SAP data manager. Therefore, these tables must be updated at reporting time. This cannot be done manually. Instead, we implement ABAP routine `update_temp_hierarchy_table(IBMDB2AA_UPDATE)` that adds and loads temporary hierarchy tables into IDAA.
ABAP routine `update_temp_hierarchy_table` must be called after the data manager has created and filled all temporary hierarchy tables that are used for a BW query and before the query is actually executed. To achieve this, we add a call to `update_temp_hierarchy_table` at the very beginning of ABAP routine `build_view(RSDRH_DB2_ROUTINES)`. The latter routine is called by the data manager to generate an appropriate expression that appears later in the FROM clause of the star schema sql query. All temporary hierarchy tables that are used by the query are given as input parameter.

```
report rsdrh_db2_routines .
[…]
form build_view
  using
    i_th_tablnm   type cl_rsdrh_hier_rest=>tn_th_tablnm
    i_tablnm      type rsd_tablnm
    i_seq_nr      type i
  changing
    c_t_tablnm    type rsdrh_t_tablnm
    e_viewnm      type rsd_tablnm
    e_is_hier_tbl type rs_bool.

********************************************** IDAA start *
  data: l_rc type sy-subrc value 0.
  perform update_temp_hierarchy_table(ibmdb2aa_update)
    using i_th_tablnm
          i_tablnm
          i_seq_nr
    changing l_rc
    if found.
  if l_rc <> 0.
    raise exception type cx_rs_program_error
      exporting
        method = 'build_view - update_temp_hierarchy_table'.
  endif.
********************************************** IDAA end *
[…]
```

All temporary hierarchy tables that are used for the executed BW query are in internal table i_th_tablnm. For each of these tables we check if the information on the temporary hierarchy table content that is stored in table ZIBMDB2AA_HIER matches the according information in table RSDRHLRUBUFFER. If there is a match (and the table name is different from parameter i_tablnm, see below), nothing needs to be done. If the information does not match or if an entry in ZIBMDB2AA_HIER for the temporary hierarchy table does not exist, the table must be (re)loaded to IDAA.

We also have to consider the following special case. If parameter i_seq_nr is greater than 0, some records have just been added to the table specified in i_tablnm by the data manager. Thus, this table needs to be reloaded to IDAA in any case.

When to update tables in IDAA

Data in IDAA that is accessed by SAP BW reporting must be consistent with DB2. Thus, whenever a extended star schema table is updated in DB2, it must be (re)loaded into IDAA.

The following operations update star schema tables and therefore require an update of IDAA:
- Load of master data
- Load of transaction data
- InfoCube compression
- Changes to InfoObject design
- Changes to InfoCube design, including repartitioning
  Tables are created anew. Thus, the setup procedure must be rerun as well.
- Deletion of all data from an InfoCube
  The fact tables might be dropped and created anew. Thus, the setup procedure must be rerun as well.
- Selective deletion
- Deletion of a request that has been released for reporting

On the other hand, the following operations do not require an update of IDAA:
- Load of hierarchies
- Rollup of aggregates
- Attribute Change Run
- Deletion of a request that has not been released for reporting

Alternative approach

The way we update tables in IDAA is not the only possible procedure. Alternatively, one can load the e fact table to IDAA, but not the f fact table. The advantage of such an approach is that the data in IDAA must be updated only when an InfoCube gets compressed. This is true for fact tables, non-fact star schema tables. Temporary hierarchy tables must be updated at reporting time as described. On the other hand, access to data in the f fact table is not accelerated and IDAA's capability of updating a single partition is not exploited. Therefore, one should implement this approach only for InfoCubes that get compressed on a regular basis.

Enabling Query Acceleration

We want to ensure that only reporting queries are actual executed by Netezza. Other queries, for example for lookup during data load, must not be executed by Netezza, because the data might not be current at that time. Therefore we do not set the DB2 parameter QUERY_ACCELERATION, which activates query acceleration for all eligible sql queries. Instead, we set the special register `CURRENT QUERY ACCELERATION` to `ENABLE WITH FAILBACK` before a BW reporting query is executed. With this setting a query that fails in IDAA is executed by standard DB2 query processing. With the alternative value `ENABLE` set, each query that fails in IDAA returns an error.
After the query is executed, we set the special register back to `NONE`.
To achieve this, we need to change the SAP code appropriately. We apply the following two modifications:

1. We append the following lines to method SET_CURRENT_DEGREE of class CL_RSDRS_DB2_EXITS:

```
select single value into l_value
  from rsadmin
 where object = 'DB2_QUERY_ACCELERATION'.
if l_value(6) eq 'ENABLE'.
  concatenate
    'SET CURRENT QUERY ACCELERATION =' l_value
    into l_stmt separated by space.
  call function 'RSDU_EXEC_SQL'
    exporting
      i_stmt = l_stmt
    exceptions
```

```
      others = 0.  " ignore any error
endif.
```

2. We append the following lines to method UNSET_CURRENT_DEGREE of class CL_RSDRS_DB2_EXITS:

```
select single value into l_value
  from rsadmin
 where object = 'DB2_QUERY_ACCELERATION'.
if l_value eq rs_c_true or l_value(6) eq 'ENABLE'.
  l_stmt = 'SET CURRENT QUERY ACCELERATION = NONE'.
  call function 'RSDU_EXEC_SQL'
    exporting
      i_stmt = l_stmt
    exceptions
      others = 0.  " ignore any error
endif.
```

Special register CURRENT QUERY ACCELERATION is set only if parameter DB2_QUERY_ACCELERATION in table RSADMIN is set appropriately. ABAP report SAP_RSADMIN_MAINTAIN can be used to set this parameter to ENABLE WITH FAILBACK (or ENABLE).

Integration into SAP BW

This chapter explains how we integrate the procedures described before into SAP BW operations.

ABAP Reports

In chapter “Data Processing with IDAA” we have described how we setup and update an SAP BW InfoCube in IDAA. The manual execution of these procedures is very time consuming and error-prone. If, for example, an update of a single table was not performed correctly for any reason, the data in the InfoCube and IDAA are not consistent anymore. Consequently, SAP BW reporting would return an incorrect result set.
For these reasons, we implemented the setup and update procedures in ABAP code.

The following ABAP reports have been implemented and used during our studies.

Report Name	Description
IBMDB2AA_STATUS	Displays the status in IDAA of all tables related to an Infocube. Input parameter is the technical InfoCube name.
IBMDB2AA_SETUP	Performs the following steps to setup an InfoCube in IDAA: 1. Creates table ZIBMDB2AA_STATUS in SAP's data dictionary. 2. Creates table ZIBMDB2AA_STATE in SAP's data dictionary. 3. Deactivates automatic setting of quality status to 'OK'. 4. Add all tables related to an InfoCube to IDAA. 5. Fills table ZIBMDB2AA_STATE appropriately. Input parameter is the technical InfoCube name.
IBMDB2AA_UPDATE	Performs the following steps to update all relevant tables in IDAA with the current data of an InfoCube: 1. Reloads all non-fact star schema tables that have been updated since last load to IDAA according to table ZIBMDB2AA_STATUS and real time statistics. 2. Reloads fact tables or fact table partitions according to table ZIBMDB2AA_STATE. 3. Enables all tables for query acceleration.

	4. Updates tables ZIBMDB2AA_STATUS and ZIBMDB2AA_STATE appropriately. Input parameter is the technical InfoCube name.
IBMDB2AA_CLEANUP	Removes all tables related to an InfoCube from IDAA, if they are not used by another InfoCube. Also updates all ZIBMDB2AA_* control tables. Input parameter is the technical InfoCube name.
IBMDB2AA_INCL	Include; implements forms that are called by the reports above.
IBMDB2AA_XML	Include; contains templates for XML generation.

The ABAP code is provided on request. Contact one of the authors and provide SAP customer number. The latter is needed to grant access to the corresponding SAP pilot note that has the code attached.
You may use the ABAP reports for reference. Please note that the code has been developed within a lab study and has limited functionality. In particular, it implements only basic error handling. Also, it is not optimized for performance; for example it does not at all exploit parallelism.

Note: Code is provided “as is” without any support and warranty.

Automated IDAA Update

Loading a request to an InfoCube and update the corresponding table in IDAA can be automated. A simple process chain has been created to show the feasibility of this approach.
Within the process chain, two InfoPackages are executed in parallel, each is loading a request. When all InfoPackages are finished, regardless whether the load failed or was successful, ABAP report IBMDB2AA_UPDATE is called with a variant that specifies the InfoCube that has been loaded. See Figure 4 for the design of our process chain.

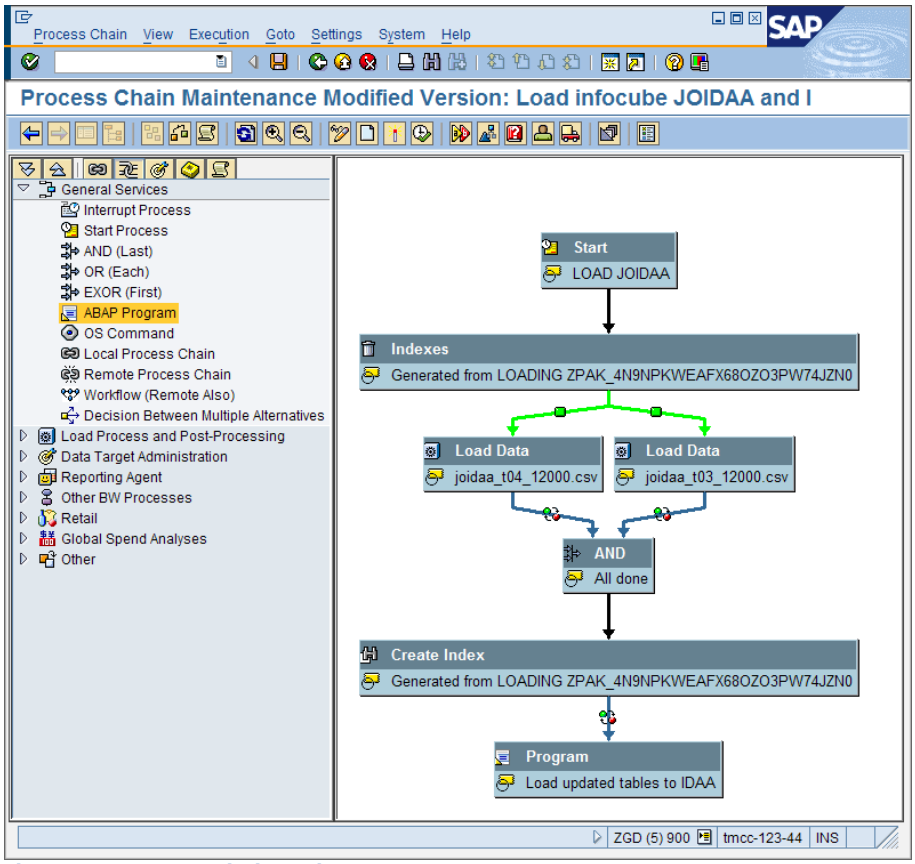

Figure 4: Process Chain Maintenance

Results

We run three different test scenarios in order to show the performance benefit that is gained by exploiting IDAA with SAP BW,.
The first test runs a set of determinate queries. Each query has a dedicated characteristic and therefore represents an individual type of query that might be run at a customer site.
The second test generates a high amount of random queries and runs them against two identical cubes (holding exactly the same data). Only one of these cubes is configured for acceleration by IDAA. For verification the result sets are compared. This mass test demonstrates the robustness of the IDAA.
The third test measures and compares the runtime of a BW random queries stream with and without acceleration by IDAA.

Dedicated Query Test

First we run 20 dedicated queries on an InfoCube with 2 million records in the fact table. See Table 1 for the results. Each query has an individual characteristic as indicated in column “Description”. Column “Records read” is the number of records that are read before aggregation. Column “Records returned” is the cardinality of the result set, i.e. after aggregation. Columns “DB2” and “IDAA” contain the elapsed time when executed in DB2 and IDAA, respectively. Both times include the network time. Finally, column “Acceleration factor” gives the factor by which the elapsed time is improved when exploiting the IDAA.

No	Description	Records read	Records returned	DB2 [sec]	IDAA [sec]	Accel. factor
1	Simple mass aggregation	1902006	21	11.6	0.53	22
2	Query #1 + 70% filter	1330351	21	9.56	0.61	16
3	Query #1 + 30% filter	570296	21	6.80	0.59	12
4	Query #1 + 10% filter	196696	21	2.05	0.58	4
5	Screwed data, low filtering	1197856	21	17.1	0.81	21
6	Screwed data, high filtering	2	2	0.53	0.54	1
7	Many restrictions	421639	21	13.9	1.63	9
8	Navigational attributes	91686	21	2.80	1.09	3
9	Navigational attributes + selective condition	97	20	1.56	0.92	2
10	Open value ranges	245	21	0.90	1.66	0.5
11	Hierarchy	183740	21	2.66	0.89	3
12	Hierarchy + selective condition	6183	21	1.42	0.77	2
13	Restricted key figures on 2 dimensions	146412	1515	14.8	1.12	13
14	Query #14 + hierarchy	14679	1015	30.8	0.82	38
15	Calculated key figures (OLAP)	589830	10	5.70	0.77	7
16	OR linked values	690791	13	4.93	0.64	8
17	Non uniform data distribution	112663	13	4.18	0.61	7
18	Selective line item	187	187	0.56	0.57	1
19	Non-selective line item	12878	12101	5.82	0.65	9
20	All together	343259	468	8.70	1.38	6

Table 1: Dedicated Query Test on a 2-million-records InfoCube

Next, we have loaded 18 million records to the InfoCube and rerun the 20 dedicated queries, see Table 2 for the results.

No	Description	Records read	Records returned	DB2 [sec]	IDAA [sec]	Accel. factor
1	Simple mass aggregation	17116647	21	117	0.78	150
2	Query #1 + 70% filter	11980812	21	94.2	0.86	110
3	Query #1 + 30% filter	5133708	21	54.8	0.82	67
4	Query #1 + 10% filter	1710293	21	17.6	0.87	20
5	Screwed data, low filtering	10790019	21	96.8	2.47	39
6	Screwed data, high filtering	24	14	7.28	0.83	9
7	Many restrictions	3805941	21	128	7.65	17
8	Navigational attributes	823646	21	17.1	1.27	13
9	Navigational attributes + selective condition	811	21	15.8	1.17	14
10	Open value ranges	2006	21	19.6	3.52	6
11	Hierarchy	1653981	21	17.6	0.97	18
12	Hierarchy + selective condition	55068	21	38.6	0.98	39
13	Restricted key figures on 2 dimensions	1314964	1948	207	7.22	29
14	Query #14 + hierarchy	132564	1499	> 1000	1.27	> 787
15	Calculated key figures (OLAP)	5321586	10	57.8	2.37	24
16	OR linked values	6212609	13	40.5	0.92	44
17	Non uniform data distribution	11016253	13	31.2	0.99	32
18	Selective line item	1724	1706	0.71	1.17	0.6
19	Non-selective line item	115481	68619	33.8	1.36	25
20	All together	3087692	468	87.7	4.42	20

Table 2: Dedicated Query Test on a 18-million-records InfoCube

With IDAA all our test queries could be run within very little time, mostly around 1 second. Even those queries are executed very fast that are hard for classic RDBS query processing (query #13 and #14).
We are experiencing an overall acceleration factor of 50. We expect an even higher acceleration factor with more data in the InfoCube.
These results prove that IDAA very well supports rapid SAP BW ad-hoc reporting.

Mass Random Query Verification Test

This test implements InfoCubes GMQT1 and GMQT1C. Both cubes are identical as the data of GMQT1 has been loaded into GMQT1C (500,000 records).However, only cube GMQT1C is configured for acceleration by IDAA. A stream of 100 random queries is fired against the two InfoCubes. For each query it is verified that the two result sets are identical. The test is done by calling report

```
RSDRC_DS_BASE_TC_RUN   with parameters
         TESTTYPE     CL_RSDRI_IPRO_CMP_TC
         INFOPROV     GMQT1
         INFOPRO2     GMQT1C
         NUM_TCS      100
```

We run this test more than 10 times for different random streams without any errors.

Mass Random Query Performance Test

This test fires a stream of 100 random queries against an InfoCube which has 6 million records. It compares the total runtime of all 100 queries with and without acceleration. We repeated this test with different random streams. **In the mean the total runtime of the report was 12 times faster with acceleration.**

Related Content

http://www.sdn.sap.com

https://www.sdn.sap.com/irj/sdn/db2

Announcement Letter: IBM DB2 Analytics Accelerator for z/OS V2.1

IBM DB2 Analytics Accelerator

SAP Note 1649284: DB2-z/OS: SAP support for IBM DB2 Analytics Accelerator

For more information, visit the EDW homepage.

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.