Seamless Integration of Archiving Functionality in OLTP/OLAP Database Systems Using Accelerator Technologies

Knut Stolze* Oliver Köth* Felix Beier† Carlos Caballero* Ruiping Li‡

Abstract:
The recent version of the IBM DB2 Analytics Accelerator introduces the High Performance Storage Saver as a new product feature. It paves another part of the way towards integrating OLTP and OLAP into a single database system. We present the technical details of this approach, which integrates archiving functionality into the DB2 relational database systems with seamless and transparent access to the archive data. The IBM DB2 Analytics Accelerator for DB2 for z/OS offers the storage area for the archive and also delivers exceptional performance for querying the data online, archived and non-archived data alike.

In this paper, we describe the administrative interfaces controlling which table partitions shall be archived (or restored) and the associated monitoring interfaces. Enhancements in the DB2 optimizer provide control whether archive data shall be considered for query processing or not. Strong focus was laid on using simple interfaces, and we present our approach taken during product design and development.

1 Introduction

The ever-growing size of data warehouse systems requires new and innovative approaches to address performance issues. Star and snowflake schemas [Leh03] are not always the best data model for reporting and analytical systems. An example is the IBM® data warehouse industry model for banking and financial markets [IBM12]. That raises the bar for solving performance problems in real-world analytical products and systems, and new technologies have to be adopted into the database system kernel for solving them.

Customers desire to run analytical queries directly on OLTP systems to base business decisions on the most recent data and to discover future trends. Thus, we see a tendency in the market to gradually merge OLTP and OLAP systems – not only by using the same software products, but also at the data level itself [Inm99]. The benefits are reduced maintenance and operations overhead as well as hardware consolidation opportunities. However, this trend increases the data volume in OLTP systems quite significantly – although the majority of the data is now read-only and kept as history for analytical workload. The OLTP system becomes an operational data store (ODS) [Inm99] with an emphasis on analytics.

* IBM Germany Research & Development, Böblingen, Germany  
† Ilmenau University of Technology, Germany  
‡ IBM Silicon Valley Lab, San Jose, USA
So far few customers of relational database systems have adopted such an integrated approach because available database system products have a hard time to cope with the diverse requirements in an integrated fashion. The IBM DB2 Analytics Accelerator (IDAA) [BBF+12] is a system that delivers extremely high performance for analytical and reporting queries. IDAA is a hardware and software appliance, which comes with its own internal storage structures. It is tightly integrated with DB2® for z/OS® already. While its primary use case is for data warehouses, it can also be applied to other database schemas. The integration into the DB2 optimizer offers the base line to add a new use case for the accelerator: serving as a high performance online archive solution.

In most analytical systems in ODSs and enterprise data warehouses, tables with significantly large amounts of data are horizontally partitioned to accommodate parallel processing and to support very easy roll-in and roll-out of whole partitions. If a date/time-based partitioning column is used to establish the partition ranges, there are typically only a few partitions on which data modification activities occur, i.e., partitions that hold current data. Data in the other partitions does not change at all or only extremely rarely due to business reasons or legal requirements. Such static data is a very good candidate for archiving, especially if it is archived into a system that provides high performance, transparent, and online access to the archived data.

Archiving data into IDAA results in a reduction of the data volume in the DB2 table, which implies smaller indexes (potentially saving multiple levels in a B-Tree) and smaller materialized query tables1. There is even the potential to do away with some access paths (indexes) – either because they are no longer needed and a table scan may be sufficiently fast now, or because multiple access paths with (partially) overlapping columns can now be combined. Durability of the archive data is still guaranteed based on backup/recovery strategies already established in customer environments. Another benefit of the solution is the reduction of disk storage required for the DB2 table, which means that less storage on high quality disks is needed. Hence, the IDAA online archiving functionality is called High Performance Storage Saver (HPSS), a term which emphasizes the latter user case.

Figure 1 illustrates that some table partitions remain in DB2 and in the accelerator for operational workload, i.e., the recent or non-archived data. A copy of the non-archived data exists in IDAA as well to facilitate query processing. Other partitions with history data reside on the accelerator only and that data was purged from DB2, i.e., it was archived to the accelerator. The DB2 optimizer will always direct any queries that touch the archived data to the accelerator, while queries against non-archived data only follow the usual query routing criteria of IDAA.

One of the main requirements for the IDAA HPSS feature was to avoid changes or rewrites of an application’s SQL statements. By default, all queries access only the non-archived data, and such queries may be routed to the accelerator if the DB2 optimizer determines that it is more beneficial to do so. In case an application wants to include archived and non-archived data for the processing of a query, it has to set a special register to convey this fact to the DB2 optimizer. The optimizer will then always route the query to the accelerator. The SQL statement texts of queries do not have to be changed at all – only the setting of the special register has to be triggered by the application.

1Materialized query tables are also known as materialized views.
The remainder of the paper is structured as follows. Section 2 gives an overview on related technology, products, and solutions that provide multi-temperature, archiving, or near-line storage (NLS) functionality. We briefly touch on the differences between those solutions and the online archiving functionality in IDAA, for which an overview on the architecture is given in section 3. The concepts and the specific implementation details applicable to HPSS are described in section 4. We cover the set of stored procedures that perform archiving and restore operations, and exposed monitoring interfaces. The behavior and semantics of the new special register and its impact on the DB2 optimizer are outlined. Performance measurements on the archiving process itself and also for query processing with and without archive data have been conducted, and the results are summarized in section 5. Finally, the paper concludes in section 6 with a summary and general outlook to future direction for the development of this new product feature.

2 Related Work

Database archiving [Sch01] is the process of moving records that are not expected to be referenced from an operational database to an archive data store. Basically, the data is partitioned into operational and archive data. The partitioning in different data stores is typically done depending on the “hotness” of the data. This supports the placement of very frequently used data on high-quality, fast (and more expensive) storage, while never or rarely used data is to be put on less expensive but slower storage. Naturally, there can be many different layers in between those extremes.

The technique of automatically moving data between different storage media is commonly referred to as hierarchical storage management (HSM). Such systems monitor data usage and decide what data can be moved to slower/cheaper storage media and which data should be moved to faster/expensive media. A HSM can decide at runtime if data has to be moved
from one level to another and applies necessary actions. HSM is conceptually analogous to cache hierarchies in CPUs where we may have several levels of high speed SRAM for caches, external DRAM, and SSDS, slower hard disk, and even slower tape devices for persistent storage.

HSM usually work at the file level. While this approach works well for individual files, deploying such a solution in the context of a relational DBMS at a higher abstraction level – like relational tables or partitions – is challenging. Originally, HSM could take backups (or copies of the data) of the entire database, but the database had to be taken offline (for consistency reasons) for that time window. In order to avoid this, most RDBMS have special APIs that allow creating backups while the database is kept online. The DBMS ensures the consistency of the backup image by considering the currently running transactions and the state of the buffer pool. These approaches usually use the transaction log for capturing changes that occur while the backup is created. There are various software solutions that perform HSM with such a database integration, for example, IBM Tivoli Storage Manager [HBB+01] and Oracle SAM-QFS [Ora11].

The way to create such database backups is often referred to as archiving. Archived data is typically not directly accessible by users via the relational interface and, thus, cannot be used for analytical evaluation. This is a significant difference with regards to IDAA HPSS where we have an online archive, i.e., data in the archive can be still queried and querying the archive data comes with the exceptional query performance for which IDAA is known.

Many DBMS vendors provide what is called multi-temperature data solutions. These so-called near-line storage repositories have many things in common with archives, but the key difference is, that the data they hold – although being used less frequently – is still accessible for query processing. Querying this data will usually have no penalties for users accessing the online database. As an example of such an approach we have SAP NetWeaver BW NLS. [IBM10] NLS maintains two different databases (possibly in different database systems): (1) the online database with the operational data, and (2) and the near-line storage database. The BW OLAP processor splits queries run against the system into two sub-queries and aggregates the partial results returned by each of them. This approach is based on the fact that clients issue their queries against the SAP BW OLAP processor and not against the underlying database system itself. Queries for the underlying DBMS are generated within SAP. In contrast, IDAA HPSS is deeply integrated into DB2 for z/OS at SQL level. So accessing the archive is transparent to any application that connects to it. In addition we get the full accelerator performance benefits for such queries.

3 IBM DB2 Analytics Accelerator Overview

IDAA [BBF+12] is based on the Netezza appliance [Fra11], which is used as backend. It provides the data storage and querying capabilities to manage large amounts of data and to provide exceptional performance for querying the data.

Figure 2 illustrates the IDAA high-level architecture. An additional process (called DWA) – that implements the integration layer with DB2 for z/OS – runs on the Netezza hardware
and operating system. This integration layer is the entry point for all requests originating either from DB2 or the IDAA stored procedures, which run on System z. The DRDA protocol [DRD03] is used for the communication between both hardware platforms. Requests to execute queries are passed to the Netezza backend by translating DRDA to CLI/ODBC. Administrative requests, e.g., to provide a list of accelerated tables, are handled in DWA itself. If necessary, SQL queries against the Netezza are executed to collect backend-related meta data and/or statistical information.

Figure 2: IDAA Architecture

It is possible to associate multiple accelerators with a single DB2 system in order to establish an environment that supports high availability and disaster recovery. The data on all accelerators is managed independently. Similarly, a single accelerator can be connected to multiple DB2 systems. The computing resources of the accelerator are then shared by all DB2 systems. It is possible to set a capping for the resources on the accelerator for each DB2 system individually. An appropriate workload balancing is applied by the Netezza backend if resource limits are reached.

It is the responsibility of the database user/administrator to trigger the refresh of the data in each accelerator individually (or to set up incremental update where appropriate). For query processing, DB2 picks any of the available accelerators that has all the tables accessed by the query. Thus, keeping the accelerated data synchronized on all accelerators is important to guarantee consistent and correct query results.

3.1 Query Processing

For each query, the DB2 optimizer decides whether to execute the query locally in DB2, or to pass it to IDAA. Multiple levels influencing this routing decision exist:
1. A system-wide configuration (zparm) can be set to enable query acceleration, i.e., the usage of an accelerator.

2. The connection between DB2 and the accelerator can be started or stopped. Only connected accelerators are considered by the DB2 optimizer.

3. A SQL session-level setting can be used to control query acceleration using the DB2 special register CURRENT QUERY ACCELERATION. The special register value can be changed by a SQL SET statement at any point in time. Possible values for the special register are:

   - **NONE**: Queries will only be evaluated locally by DB2 and no accelerator is considered.
   - **ENABLE**: Eligible queries, i.e., queries that can be executed syntactically and semantically correct by IDAA, are routed to the accelerator if the DB2 optimizer’s heuristics consider the usage of IDAA as beneficial.
   - **ENABLE WITH FAILBACK**: Same as **ENABLE**, but if the query fails on the accelerator for whatever reason at run-time, DB2 executes the query itself to recover.
   - **ELIGIBLE**: Eligible queries are always routed to the accelerator, but no heuristic checks are applied. Non-eligible queries, e.g., queries using unsupported expressions or accessing non-accelerated tables, are executed locally in DB2.
   - **ALL**: All queries are routed to the accelerator. In case a query is not eligible, the query will fail with an appropriate SQL code.

If the decision is made to pass on the query to IDAA, DB2 translates the DB2 SQL dialect to the IDAA/Netezza SQL dialect. A new DRDA connection is established between DB2 and IDAA, and DB2, acting as application requestor, sends the translated query. IDAA, acting as application server, returns the corresponding result set to DB2.

### 3.2 Data Maintenance

The data of accelerated tables in IDAA is a snapshot of the data in DB2. The snapshots have to be refreshed in case the data in DB2 changes. IDAA offers 3 options for the data refresh, which are explained in more detail.

The IBM DB2 Analytics Accelerator comes with an extremely fast loader for whole tables or a set of partitions thereof. Refreshing the data of an entire table (Figure 3) is typically done for rather static data and for non-partitioned tables. Partition-based load (Figure 4) is targeted at partitioned tables, where updates are performed rather infrequently and only to a small subset of partitions. Parallelism is exploited for tables using partition-by-range and also for tables that are defined as partition-by-growth.

If a low latency for the data currency in IDAA is not very important and queries return acceptable results, even if the data is slightly outdated by a few minutes or hours, both
options are viable. Additionally, the initial load of the data into the accelerator is accomplished that way with very good performance, reaching a maximum throughput of 1.5 TB/h. Internally, multiple threads of the ACCEL_LOAD_TABLES unload the data from DB2 in the DB2 internal format and send that to the accelerator, i.e., only a minimum of CPU resources is needed on System z. The accelerator parses the DB2 format, converts it, and inserts it in parallel into a shadow table in the Netezza backend.

For tables with a higher update frequency and where high data currency on the accelerator is desired, a third option for data maintenance is the Incremental Update feature (cf. figure 5). Incremental update is based on replication technology [BNP+12], which reads DB2 logs and extracts all changes to accelerated DB2 tables. Those changes are transmitted to IDAA, where they are applied to the Netezza backend tables. Since Netezza is tailored towards very efficient and high-performing execution of complex queries, and less so for manipulating single rows, batching is used when applying the changes. A latency of about 1 min is achieved for applying the changes to the shadow tables due to this change batching. For reporting systems and data warehouses, 1 min is usually fast enough, especially if complex accelerated queries may take several minutes (or hours) in DB2 and still a few seconds with IDAA.

4 High Performance Storage Saver

4.1 Basic Design Principles

The design of IDAA HPSS was governed by very few principles. DB2 for z/OS is and remains the one and only point to access and administer the data – regardless of the physical storage location. The DB2 optimizer chooses the best access plan, which is either a local execution in DB2, or an execution of the whole query in the accelerator.
DB2 is the owner of the data and all the administrative tasks like backup/recovery strategies are placed there. In particular, no backup/recovery mechanisms were introduced on the accelerator itself so that customers can rely on their already available DB2 skills.

A primary design goal we had for new interfaces was simplicity and ease of use, e.g., to archive or restore table partitions and to retrieve monitoring information. Since IDAA is an appliance, the amount of information needed for tuning by a database administrator is small to begin with. The accelerator does not have a large tuning layer, which means only distribution and organizing keys can be changed on a table, but no indexes or materialized views or other techniques can be applied. HPSS does not increase the complexity unless absolutely necessary.

In short, the overall guiding rule was to further strengthen the IDAA idea of a fully autonomous appliance.

4.2 Overview

Two new stored procedures establish the main functionality for HPSS (cf. figure 6). The first, ACCEL_ARCHIVE_TABLES, handles all the steps necessary for moving of data from DB2 to the accelerator. This includes the removal of the data from the DB2 table and the maintenance of related indexes. ACCEL_RESTORE_ARCHIVE_TABLES is the second stored procedure. It implements the reverse process, i.e., restoring the data and rebuilding index and support structures in DB2. For monitoring purposes, the output of the ACCEL_GET_TABLES_INFO and ACCEL_GET_TABLESDETAILS stored procedures has been extended to return necessary archiving-related information. The final piece is the newly introduced DB2 special register CURRENT_GET_ACCEL_ARCHIVE. It is set by DB2 client applications and used by the DB2 optimizer – together with the information in the catalog table SYSACCELERATEDTABLES – to decide whether to include archive data in the query (if available) or not.

Inherent to this design is that all interfaces related to HPSS are accessible via SQL. While calling stored procedures is well established in every major relational database system and very beneficial for automation purposes, it is by no means very end-user friendly. The IBM DB2 Analytics Accelerator Studio is the graphical user interface (GUI) for IDAA. Most administrative operations that are necessary for IDAA can be done via the GUI, and moving data to the IDAA online archive has been integrated, too.

Archiving and restoring with HPSS operate on table partitions. Only all the data of a partition can be moved to the accelerator and is pruned from the DB2 table. This granularity was chosen in order to simplify the implementation, improve performance, and to reduce locking and logging overhead that would be inherent to row-level granularity. DB2 utilities are exploited with their low-level access to the DB2 data manager.

DB2 for z/OS supports two types of partitioning: range-based partitioning assigns data rows to partitions based on the values of a partitioning key column (e.g., a column of type DATE) while growth-based partitioning automatically adds storage as new partitions to the table as the volume of data grows. With growth-based partitioning, no fixed association of
Figure 6: HPSS Interfaces

rows to partitions exists. The database system is free to shuffle rows between partitions to fill gaps in the physical layout, e.g., when the table is reorganized. For HPSS, we need an explicit, data-dependent and deterministic decision whether a row should be archived or not. Therefore, support is limited to range-partitioned tables.

In the following, we describe in more detail how the different pieces of the solution play together.

4.3 Moving Data to HPSS

All tables handled by an accelerator must have been previously registered to it. The registration copies schema information (columns, data types, ...) from the DB2 catalog to the accelerator. In fact, the registration triggers the creation of a schema-compatible shadow table in the Netezza backend system that is managed solely by IDAA. So exploiting an accelerator A as online archive for a DB2 table T requires that T has already been registered with A.

The process of archiving data has to prepare for an eventual loss and subsequent recovery of the archived data. Since there are no backup/recovery mechanisms on the accelerator itself, mechanisms in DB2 for z/OS and System z itself are put to the task. Furthermore, the design is such that all archived data is copied from DB2 to the accelerator, even if that data has previously been loaded in the accelerator. This guarantees that the archive data is exactly the same as it was in DB2.
The input for the ACCEL_ARCHIVE_TABLES stored procedure is an XML document with a sequence of tables and a list of partitions for each of them. The partitions list allows abbreviations like “first \( n \) partitions” or “all partitions except last \( m \)”. A typical use case would be a table partitioning based on a single DATE column with ascending key ordering, such that the first partition contains the oldest data and the last partition the newest. In this case, the “first \( n \)” criterion corresponds to a SQL predicate \( \text{WHERE partitioning
column} \leq \text{end date for partition}
\). An example XML document is shown in listing 1. It specifies to archive partitions 1 thru 9 (inclusive) for table HPSS.STORE_SALES.

```xml
<dwa:tableSetForArchiving version="1.0"
    <table schema="HPSS" name="STORE_SALES">
        <partitions>1:9</partitions>
    </table>
</dwa:tableSetForArchiving>
```

Listing 1: XML Document for Archiving

The data for each of the partitions identified in the XML document is archived. If a partition has already been archived before (and was not restored in the mean time), that partition is skipped because the accelerator already has the archive data. This silent tolerance of archived partitions simplifies automation processes, e.g., for a monthly archiving of all partitions, except the partitions for the last 3 by using a fixed range \( 1 : -4 \).

DB2 utilities are employed to achieve very high performance for the archiving process by bypassing much of the relational processing inside DB2 and to reduce the logging and locking overhead. The following major steps are applied for each partition:

1. Lock the table partition in shared mode to prevent concurrent updates on the data.
2. Create a new backup (called image copy) of the partition using DB2’s COPY utility.
3. Copy the data from the partition to the accelerator using the UNLOAD utility.
4. Commit the archive data in the accelerator.
5. Prune the partition in DB2 using the LOAD REPLACE utility with an empty input data set.

DB2 customers have well-established procedures for backing up their data. The introduction of IDAA into the customer environment in general (and HPSS in particular) should not impact these procedures. Backups are always based on the data in DB2. Thus, recovering any data (should that become necessary) is always based on those backups and not on the accelerator. So even after data has been purged from the DB2 table, DB2 for z/OS and System z with its superior security and reliability qualities guarantees that data can be recovered in case of system, storage, or site failures.

While several mechanisms and third-party utilities exist for backing up DB2 databases on System z, the most common mechanism is the COPY utility. So we based our approach.
for HPSS on that. The COPY utility takes a backup of DB2 table data at data set level and logs that action in a DB2 catalog table to allow an automated restore in combination with log replay. Of course, the COPY utility provides a wide variety of options to influence the placement of backups and the processing behavior.

In the context of IDAA, we wanted to avoid cluttering the interface of the stored procedure ACCEL_ARCHIVE_TABLES with a multitude of (possibly never needed) options. Therefore, the only input that has to be specified is a high-level qualifier (HLQ), identifying the location within the file system where the backup data sets are to be placed. The HLQ is specified as a global environment property that must be set by the administrator. A naming convention is automatically applied to merge the names of the DB2 subsystem, the database, the tablespace, and the partition identifier to build the final data set name for the backup image. It is mandatory that the automated disk space management facility of System z, i.e., system managed storage (SMS) with corresponding automatic class selection (ACS) routines, is set up to correctly handle the backup data sets, which will be allocated under the defined HLQ. In particular, to realize the space saving potential of HPSS, SMS must be set up such that these backups are placed on less expensive disk storage.

An important aspect for the administrator is that the backup images created by HPSS must exist independently of backup copies that are created by other automated regular backup processes. Usual backup procedures of customers retain a certain number of backup levels. At some point, old backups are discarded by deleting the data set and purging the DB2 catalog table using the MODIFY RECOVERY utility. The backup images created by HPSS must not be discarded, however. These backups have a different semantics because they are the source for restoring the actual customer data back into DB2, while other backups are only needed to recover the DB2 table to a certain point in time in case of system or storage failures.

After backup creation, the data of an archive partition is transferred to the accelerator. Data transfer uses the same mechanisms as the ACCEL_LOAD_TABLES stored procedure, i.e., the DB2 UNLOAD utility is employed to read the data in parallel using multiple threads. The archive data is read from the backup data set so that the same data can be send to multiple accelerators. Since archiving is a destructive operation on the DB2 table, it is not possible to read the data again from the DB2 table; but the backup data set holds the master data and can be used as primary source.

On accelerator side, the archive data is inserted into a different shadow table in the Netezza backend database than regular, non-archived data. A view combines the rows of both tables, and DB2 uses this view in rewritten queries. This gives IDAA the flexibility to choose a different suitable storage structure in the future. Both shadow tables (cf. figure 7) use a hidden column identifying the DB2 partition from which each row was loaded, thus allowing efficient detection and management of all data that corresponds to a specific DB2 partition.

Finally, archived data is pruned from the DB2 table using the LOAD REPLACE utility to overwrite the partition data from an empty data set. The utility provides a very efficient means of deleting an entire partition, by-passing the DB2 transaction log. An effect of the utility is that the data set for the partition is redefined in the file system, i.e., deleted,
re-allocated, and re-initialized. The re-allocation effectively reduces the size of the data set to the minimum possible allocation (primary extent), so that the disk space for the archived partition is actually freed. The partitioning definition of the table remains unchanged in the DB2 catalog, including the archived partitions, which are now empty. An unchanged partition definition is required for a potentially needed restore of the partition backup image.

The partition is then marked as archived in the accelerator meta-data and subsequent attempts to load or archive this partition in the future will be ignored. The content of archived partitions is effectively frozen. For DB2, those partitions still exist without any data, so SQL statements like UPDATE or DELETE have no effect. If any data is subsequently inserted into these partitions in DB2, it will not be reflected on accelerator side. Setting the partition to read-only state in DB2 prevents such situations. Of course, the policies and ETL logic of the data warehouse should already guarantee that archived data ranges remain unchanged. Otherwise, using HPSS is not appropriate.

For changing the partitioning of a table, DB2 supports the addition of new partition and the rotation of existing ones. Adding new partitions is straight-forward since any data in new partitions is treated as non-archived data, which can later be archived. Rotating a partition is logically equivalent to deleting the partition (typically the one containing the oldest data) and then adding a new partition at the end of the existing partition key range. If the partition being rotated has previously been archived, a call to the ACCEL_ARCHIVE_TABLES stored procedure synchronizes the data in the accelerator and deletes the matching rows in the archive shadow table.

4.4 Restoring Data from HPSS

The anticipated, typical usage scenario for HPSS is the movement of data from DB2 to the accelerator – not vice versa. Historical and archived data that will not be changed and is not needed any more for most of the query processing (at least for transactional queries). It can be backed up on inexpensive storage and moved to the accelerator where it remains available for occasional analytic processing. However, there are some situations where it may become necessary to restore the archived data back into DB2 tables.
It must be possible to restore data when a system or site failure involving the accelerator has destroyed the archive data. By design, IDAA does not capture backups on accelerator side and rather relies on DB2 for z/OS. Another accelerator may have been used to store a second copy of the archive data to avoid such situations. However, not all customers may use a second IDAA. In addition, it may turn out that the decision to archive some or all partitions of a table in HPSS was premature and direct access in DB2 is required again. For example, business reasons may demand modifications to history data, or queries may have to be applied to the archive data, which are not (yet) supported by IDAA.

The stored procedure ACCEL.RESTORE.ARCHIVE_TABLES can be used to restore data of archived partitions. Its main task is to automate the DB2 utility calls that are needed for recovering the image copy and to purge the archive data in IDAA. This includes executing utilities to check data consistency (i.e. re-validate constraints like unique and referential constraints) and to rebuild indexes. On the accelerator, the archived data is directly moved back into the shadow table for the non-archived data and the corresponding catalog information is updated. The stored procedure is currently in prototype stage and will be made generally available in IDAA in the near future.

4.5 Monitoring Archived Data

For monitoring purposes, the output of stored procedures ACCEL.GET_TABLES_INFO and ACCEL.GET_TABLESDETAILS has been extended to return archiving-related information at table and partition level, respectively. At table level, HPSS provides information whether an accelerated table has an archive, and how much DB2 data it contains, measured by the number of rows and bytes. A sample output is shown in listing 2.

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <table schema="HPSS" name="STORE_SALES">
    <status loadStatus="Loaded" accelerationStatus="true" integrityStatus="Unimpaired" archiveStatus="true" />
    <statistics usedDiskSpaceInMB="1" rowCount="2000" archiveDiskSpaceInMB="100" archiveRowCount="10000" skew="0.3" organizedPercent="95.00" lastLoadTimestamp="2012-09-20T11:53:27.997141Z" />
  </table>
</dwa:tableInformation>
```

Listing 2: Sample Output of ACCEL_GET_TABLES_INFO

At partition level, the output lists the timestamp when a partition was archived, how much archive data was transferred from DB2 to IDAA, and the name of the backup data set on System z. Listing 3 shows an example for a single table with just 4 partitions, which represent the quarters of 2012. The partitions with logical partition number 1 and 2 have been archived, while partition 3 holds non-archived data only and partition 4 is actually empty. The GUI visualizes the information provided by both stored procedures (cf. figure 8).
4.6 Query Processing

For each query entering DB2, the DB2 query optimizer checks if all of the following conditions hold true (in the non-archiving context):

- Query acceleration is enabled.
- All tables referenced by the query are loaded on an accelerator.
- The query qualifies for routing (e.g., only constructs supported by the accelerator’s query processor are used).
- Heuristics (cf. section 3.1) indicate that the query will be executed faster on the accelerator (in particular, it’s an analytic query and not a transactional one).
If at least one condition is not satisfied, the query is handled in DB2. If all checks are passed, the query is rewritten into the SQL dialect supported by the accelerator (the Netezza dialect), and the query is passed on to the accelerator and executed there. The accelerator solely relies on the data in the shadow table(s) in Netezza. Query results are passed back to DB2, which returns it as-is to its client application.

In short, where the whole query is executed is a binary decision. Admittedly, those rules are simple, but that makes them very robust and attractive for customers. This is a very light-weight variation of federated technologies [ISO03]. In particular, DB2 – as the federated server – does not attempt to compensate missing functionality. Also, only heuristics are used to come up with a routing decision and no fully-fledged costing is applied. The advantage of this approach is that applying heuristics is much faster and, thus, any impact on OLTP workload is significantly reduced.

With HPSS, query processing needs to take into account whether the query should include archived data of the involved tables. For queries that involve non-archived data only, the query optimizer usually has the freedom to decide whether the query should be routed to the accelerator or not, as outlined above. Assuming the data in the accelerator has been maintained properly, both execution paths return the same result. On the other hand, queries involving archive data must always be executed on the accelerator; running them in DB2 would produce incorrect results due to missing rows.

At first sight, it may seem desirable to let the DB2 optimizer and/or DB2 runtime decide whether a query may involve archived data and handle it accordingly. However, for many cases it cannot be easily determined a-priori that archived data is or is not in scope for the...
query. For example, predicates on functionally dependent columns may apply the necessary filtering, or the scanning of a dimension table may reduce the qualifying rows on a fact table to the non-archived data only, possibly over a cascade of joins. Another aspect are index-only queries, for which no access to the archived partitions of the tablespace occurs. The index itself has no knowledge about values for archived rows, especially if the index is non-partitioning. The consequence is that any query where such doubts arise, would have to be routed to the accelerator. Since this applies to a majority of the queries, the optimizer would essentially lose the freedom to determine the “optimal” place for query execution (DB2 or accelerator). While it would be possible to require specific filtering predicates involving the partitioning key columns, this is not only unrealistic, it also voids the key design principle of transparency for existing applications.

Therefore, the decision whether a query should or should not include archived data is explicitly made by exposing the new special register CURRENT GET ACCEL_ARCHIVE in DB2 with possible values YES and NO. The solution is not fully transparent to client code, but is much more flexible since it allows explicit control at several levels by configuring a system-wide default value, setting it at the connection level (e.g., in the JDBC connection string), or allowing application developers to switch the semantics within an established SQL session at will.

If CURRENT GET ACCEL_ARCHIVE is set to YES, then any query that includes a table which has archive data must be executed on the accelerator. The DB2 optimizer heuristics do not even have to be checked. To easily detect the presence of archive data from DB2 query processing, the DB2 catalog table SYSACCELERATEDTABLES carries a new column ARCHIVE. The values in this column indicate the presence or absence of archive data for this table, and the IDAA stored procedures maintain that. If the query cannot run on the accelerator, e.g., because it references tables that are not accelerated or because it uses SQL constructs that are not yet understood by IDAA, then the query will fail with a dedicated SQL code.

If the query can be routed and needs to access archive data, DB2 rewrites the query against the view (cf. figure 7) in the accelerator, which combines the data from the shadow table holding non-archived data with the shadow table for the archive data. The view employs a UNION ALL operator, which is evaluated in the Netezza backend only. Thus, the physical separation of the data is transparent to DB2. Furthermore, we retain the binary decision for the location of query processing. This avoids the inherent complexity of partial query offloading if, e.g., only the archived data would reside in IDAA while the non-archived data remains in DB2 only.

If CURRENT GET ACCEL_ARCHIVE is set to NO, query processing remains as before without HPSS and only works on the shadow tables containing the non-archived data. That is true even if archive data for the same tables is present on the accelerator.
4.7 Imposed Limitations

Our implementation initially imposes several restrictions for tables that are moved to IDAA HPSS. A fundamental restriction is that DB2 is unable to handle referential constraints if part of the referenced data is not available. Therefore, tables that are referred to by a foreign key in the same or another table may not be archived. Other restrictions are rooted in the backup strategy that we have chosen. Since the “master data” for archived partitions are the backups, tables that have LOB and XML data columns are not allowed. Those objects are stored in separate tablespaces, which are not yet included in backup copies that the ACCEL_ARCHIVE_TABLES procedure creates.

No table or tablespace modifications may be performed that prevent restoring the backup images. For example, modifications of physical table space properties like DSSIZE would cause the RECOVER utility to fail. Such modifications require that all archived data is restored first and archived again after the modification is done.

Naturally, any restrictions that IDAA has in general also apply to HPSS. So if a table has columns with data types that cannot be loaded into the accelerator, a projection of the table is used. This applies, for example, to the data types DECFLOAT and ROWID. While the data of such tables can be archived, those columns are omitted and queries including the archive must not involve those columns.

4.8 HPSS and Data Maintenance

Section 3.2 described the options that can be used to maintain data of accelerated tables. Archiving introduces another option, even if it only applies to a part of the table data. For any given table, only one of the data maintenance options can be used concurrently. Thus, if some data shall be archived, it is not possible to load all or parts of the non-archived data using the ACCEL_LOAD_TABLES stored procedure at the same time. Likewise, replication has to be stopped or disabled for the table while an archive or restore operation is in progress.

Incremental update via replication merits a closer look. It is supported to archive some part of a DB2 table to IDAA and to apply replication on the non-archived data. Using replication for the archive data is not necessary since the archive data won’t change. Purging the archive data in DB2 uses the LOAD REPLACE utility. This avoids the logging of deleted rows. The replication setup is configured to ignore log entries for the utility executions. So, after an archiving operation completes, replication can be re-started and will continue to propagate changes on the non-archived data from the DB2 log position where it left off when it was stopped before archiving. That means, changes to non-archived data that occurred while data was being moved to HPSS will be propagated correctly. The purged data will not be propagated, however.
5 Evaluation

We conducted a set of performance tests during the development of HPSS. The new functionality should deliver the same results and acceleration factors that customers have come to expect from IDAA.

An important aspect was the impact of the two shadow tables and the UNION ALL view for query processing. For example, the optimizer of the Netezza backend has to detect the union and push down filtering predicates into both branches of the set operation if possible. It may also have an influence on join order, broadcasting (intermediate) tables, and so on. Some performance issues have been identified in that area, and improvements were made.

As figure 9 illustrates, the resulting query performance is very close to the performance of the same queries without any data being archived, i.e., all the data residing in a single table in the accelerator. The LINEITEM table of the TPC-H benchmark was used, and about half of the table’s data was archived. Since the data is evenly distributed across the table partitions, 50% of all partitions was in the archive. The total TPC-H data volume was 200 GB. The relative differences (cf. figure 10) are comparatively small. Some queries actually run slightly faster because less data resides in the base tables and more parallel processing in the Netezza backend can be exploited. Other queries take a bit longer, e.g., in case the push-down of predicates into both branches of the UNION ALL is not always applied and also due to the the increased query complexity. Overall, the known advantages of IDAA with acceleration factors of up to 2000x can be achieved with HPSS as well.

We also measured the performance for moving data from DB2 to the accelerator. The upper limit for the throughput is defined by \texttt{ACCEL\_LOAD\_TABLES}, which just copies data from DB2 to the accelerator. Typically, a throughput of 1 TB/h can be achieved; in lab environments, we measured an even higher throughput of 1.5 TB/h. For archiving, additional steps need to be performed. A commit point occurs after each partition, which implies that at most 64 GB of data can be streamed and then the data flow pipeline stalls. This has a very minor and negligible impact only. However, additional DB2 utilities are executed in order to create the backup and to prune the data from the DB2 table. The data pruning maintains indexes in DB2, and figure 11 shows the impact of this. Archiving 50% of the LINEITEM table in a TPC-H schema is only slightly slower than loading the data if no indexes are defined. The overhead for the executing the additional utilities is between a
few percent only. Those indexes are maintained by the **LOAD REPLACE** utility. The more indexes are defined on a table the higher the overhead gets. The 200 GB scenario with just 3 indexes – one index is a partitioning index (includes the partitioning key column) while the other two indexes are non-partitioned – shows that the execution time for 50% of the data jumps from 10 min to nearly 30 min. All additional time is spent by the utility.

The archiving performance with a (more typical, but still small) data volume of 1 TB is depicted in figure 12. Loading 50% of the table data scales linearly, and so does archiving. The overhead for committing after each partition and the additional utility execution stays in the single-digit percentage range. The figure does not show the case with indexes being defined on the table because the behavior with indexes is the same as in figure 11. We only want to highlight the small gap between loading and archiving the data.

### 6 Summary and Outlook

In this paper we have presented the High Performance Storage Saver, a new feature of the IBM DB2 Analytics Accelerator. It enhances the product to use IDAA as Online Archive, with extremely good query performance on querying the archived and non-archived data. Thus, a completely new use case for accelerator technology is established, which takes the next step towards an integration of OLTP and OLAP systems, at schema and data level. We described the interfaces to archive data from DB2 for z/OS to IDAA and to restore it back. Query processing relies on a new DB2 special register for the decision whether archive data shall be considered in the query or not. The just released IDAA Version 3 offers this functionality to customers.

Since this is a new feature, there is still room for improvements. The idea is to largely base further enhancements on feedback we already have received and will receive from customers. For example, the restore functionality is rather basic and its scope shall be broadened, e.g., to restore data back into DB2 based on the data in the accelerator. Similarly, the support for multiple accelerators is still in a prototype stage and will be productized. Long term, it is desirable to automate the decision for exploiting the IDAA online archive based on query semantics.
7 Trademarks

IBM, DB2, and z/OS are trademarks of International Business Machines Corporation in USA and/or other countries. Other company, product or service names may be trademarks, or service marks of others. All trademarks are copyright of their respective owners.

References


