B. Mitschang et al. (Hrsg.): Datenbanksysteme für Business, Technologie und Web (BTW 2017),
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 531

Autonomous Data Ingestion Tuning in Data Warehouse Accelerators

Knut Stolze[1], Felix Beier[1], Jens Müller[1]

Abstract: The IBM DB2 Analytics Accelerator (IDAA) is a state-of-the art hybrid database system that seamlessly extends the strong transactional capabilities of DB2 for z/OS with very fast processing of OLAP and analytical SQL workload in Netezza. IDAA copies the data from DB2 for z/OS into its Netezza backend, and customers can tailor data maintenance according to their needs. This copy process, the *data load*, can be done on a whole table or just a physical table partition. IDAA also offers an *incremental update* feature, which employs replication technologies for low-latency data synchronization.

The accelerator targets big relational databases with several TBs of data. Therefore, the data load is performance-critical, not only for the data transfer itself, but the system has to be able to scale up to a large number of tables, i. e., tens of thousands to be loaded at the same time, as well. The administrative overhead for such a number of tables has to be minimized.

In this paper, we present our work on a prototype, which is geared towards efficiently loading data for many tables, where each table may store only a comparably small amount of data. A new load scheduler has been introduced for handling all concurrent load requests for disjoint sets of tables. That is not only required for a multi-tenant setup, but also a significant improvement for attaching an accelerator to a single DB2 for z/OS system. In this paper, we present architecture and implementation aspects of the new and improved load mechanism and results of some initial performance evaluations.

1 Introduction

The IBM® DB2® Analytics Accelerator for z/OS (IDAA, cf. Fig. 1) [Ba15] is an extension for IBM's® DB2® for z/OS® database system. Its primary objective is the extremely fast execution of complex, analytical queries on a snapshot of the data copied from DB2. Many customer installations have proven that the combination of DB2 with a seamlessly integrated IDAA delivers an environment where both, transactional workload and analytical queries, is supported without impacting existing and new applications. The achieved query acceleration for analytical workloads is at least an order of magnitude, often even exceeding that.

After the initial version of IDAA based on a Netezza backend [Fr11] became available in 2011, functional enhancements were added to expand the product's scope. The first steps were more fine-granular data synchronization mechanisms like *partition load* and *incremental update*. A completely new use case was introduced with the *high performance storage saver*, which transformed the accelerator into a cost-efficient archiving solution [St13]. More recently, IDAA was enhanced to directly support analytical workload. Such workload

[1] IBM Germany Research & Development GmbH, Schönaicher Straße 220, 71032 Böblingen, Germany, {stolze,febe,jens.mueller}@de.ibm.com

Fig. 1: System Overview of the IBM DB2 Analytics Accelerator (IDAA)

often involves complex data transformation, which can now be processed completely within the accelerator by means of accelerator-only tables [BSM16]. DML operations like `INSERT`, `UPDATE`, and `DELETE` statements can be send from a customer's application to the DB2 for z/OS server, which sends the complete SQL statement on to the accelerator for execution. No exchange of the data between DB2 for z/OS and the IDAA server is necessary and, thus, frees resources on System z. Furthermore, analytics stored procedures of the IBM® Netezza® Analytics (INZA) product are enabled directly on the accelerator [IB14b] to add support for data mining workloads as well.

Nevertheless, data movement and copying the data from DB2 for z/OS to the accelerator is still, and will remain, one of the most important aspects of the appliance. Applications are working with tables in DB2 for z/OS, and changes made there need to be made available on the accelerator. IDAA's customers have many tables on the system – 30,000 tables or more is not a rare situation. Many of these tables are rather small in size, i. e., empty or just a few rows up to a few MBs worth of data.

In this paper, we present our work to improve IDAA in order to optimize batch-loading of large table sets. We developed a prototype, which considers all currently pending load requests of a table batch and autonomously determines an optimal schedule for the next time slice. The degree of parallelism is adjusted, based on the current system utilization. That takes into account the currently running loads as well as any other workload on the Netezza system, such as analytic queries or maintenance operations.

The remainder of the paper is structured as follows. The architecture of the IBM DB2 Analytics Accelerator is revisited in Sect. 2. Sect. 3 explains the new architecture for IDAA's data load. It covers the new load scheduler and its interactions with other components in the system. Some initial performance results are presented in Sect. 4, and the paper concludes with a summary in Sect. 5.

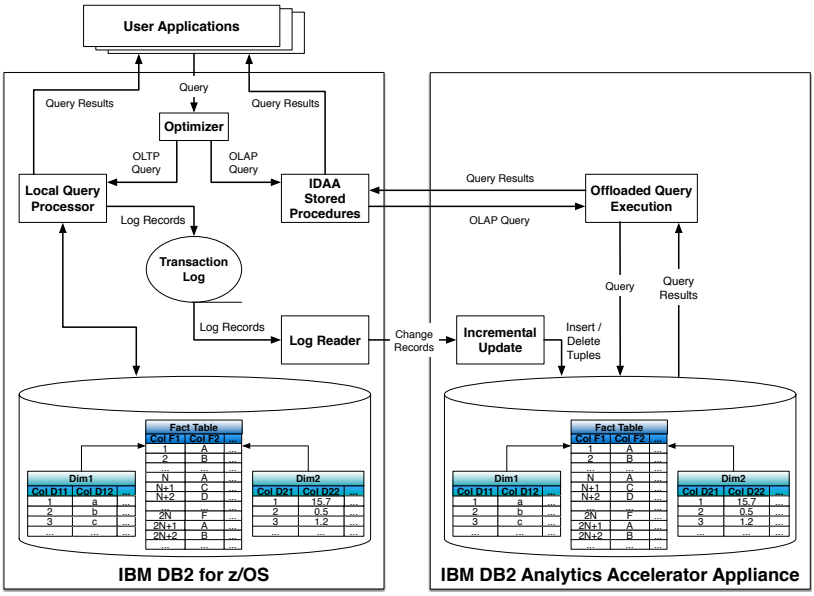

Fig. 2: IDAA System Architecture and its Integration into DB2 for zOS

2 Overview on the IBM DB2 Analytics Accelerator

The IBM DB2 Analytics Accelerator (IDAA) [Ba15] is a hybrid system, which uses DB2® for z/OS® [IB14a] for transactional workload with the well-known excellent performance characteristics. A copy of the DB2 data resides in the accelerator, which is based on Netezza technology [Fr11], and deals with analytical workload in an extremely high-performing way. The DB2 optimizer is responsible for the query routing decision, i. e., whether to run the query in DB2 itself or to offload it to the accelerator.

The benefits of this system are a reduced complexity of the necessary IT infrastructure on customer sites for executing both types of workloads and its tight integration into the existing DB2 for z/OS system, which results in overall cost reductions for customers. A single system is used and not a whole zoo of heterogenous platforms needs to be operated and maintained. Aside from the system management aspects, investments into a business' applications is protected, which is crucial for companies of a certain size. Existing applications can continue to use DB2 unchanged, while additionally exploiting the analytics capabilities of IDAA without any (or only minuscule) changes.

2.1 High-level System Architecture

IDAA provides the data storage and SQL processing capabilities for large amounts of data with exceptional query performance. Fig. 2 illustrates the high-level architecture. The key is the seamless integration of all components in DB2 to leverage the Netezza appliance as analytical backend. SQL statements are passed from DB2 to IDAA, which drives the execution in Netezza. Administrative requests, e. g., to provide a list of accelerated tables, to trigger table load operations, or to setup automatic log-based data replication for offloaded

tables, are handled in IDAA itself in conjunction with the IDAA stored procedures. Either way, the external interface to work with IDAA is DB2. If necessary, SQL queries against Netezza are executed to collect backend-related meta data and/or statistical information which are utilized for the load scheduler presented in Sect. 3.1.

Fig. 3 shows that 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. Appropriate workload balancing is applied by DB2 in case the connected Netezza backends have different hardware and workload characteristics. Similarly, an accelerator implements a multi-tenant environment where different DB2 systems can connect to it, sharing its resources. Another workload balancing layer is applied by the Netezza backend on the SQL level to cover multi-tenancy scenarios.

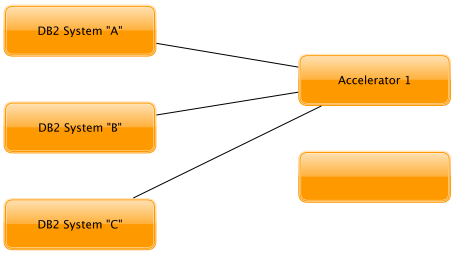

Fig. 3: System Setup with M:N Connections between DB2 z/OS Subsystems and IDAA Appliances

Such a setup is very flexible and allows our customers to slowly grow the exploitation of IDAA, depending on the (changing) workload. However, our experience has shown that the accelerator can easily become overloaded, or the DB2 systems cause contention to such a degree that SLAs can no longer be satisfied. While Netezza's workload management, combined with IDAA's spill-to-disk functionality for buffering query results in slow receiver scenarios, is excellent for concurrent query processing, the data load processing requires more resources and leaves room for improvements which will be presented in the following.

2.2 Data Replication Strategies

IDAA offers three options for refreshing the data that has been shadow-copied from DB2. Entire tables or individual table partitions can be refreshed in the accelerator batch-wise. Fig. 4 and 5 illustrate both bulk-loading scenarios conceptually. The DB2 table is on the left side, and the IDAA table on the right. The sketch above the arrow shows which pieces of the DB2 table are copied to the IDAA table.

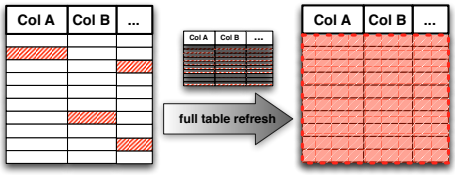

Fig. 4: Full Table Refresh

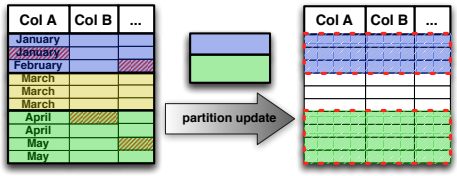

Fig. 5: Partition Update

For tables having a higher update frequency and where a high data currency is desired, the Incremental Update feature (cf. Fig. 6) is suitable. This feature uses IBM® InfoSphere® Change Data Capture (CDC) [IBM13, Be12], which reads DB2 transaction logs and extracts all changes to accelerated DB2 tables. Unlike the bulk-load strategies, CDC replicates only those change data records while unchanged data is not copied.

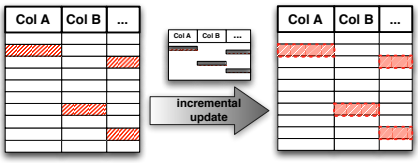

Fig. 6: Refreshing Table Data with Incremental Update

While the bulk-load strategies offer great throughputs but require to copy large data volumes and, hence, are high-latency operations, the incremental update strategy replicates only low data volumes with larger overheads per tuple, but a low latency until change records have been applied in the backend. A latency of about 1 minute is achieved, which is usually fast enough for reporting systems and data warehouses, considering that complex accelerated queries may take several minutes (or hours) in DB2 vs. a few seconds only with IDAA. Furthermore, getting really *the* last committed changes for business reports is not that crucial as long as the report is not based on hours old data. 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) [IB13]. Since many customers have thousands of tables in DB2 and also in IDAA, it is important to understand query access patterns to the individual tables and analyzing the accelerated workload with the help of monitoring tools.

2.3 Load Processing

Right from the very first version of IDAA, the development team focussed on a highly efficient load process for the data synchronization between DB2 for z/OS and IDAA with its Netezza backend. Traditional replication technologies like Q-Rep or CDC [IBM13] were evaluated, which were still separate products in year 2011. We found that replication products were not even close to delivering the desired throughput of $1TB/h$. Therefore, IDAA implemented its own batch-load mechanism, which is depicted in Fig. 7.

On z/OS side, DB2's `UNLOAD` utility is employed to extract the data verbatim from the pages of the database system. It bypasses most of the relational data processing components of DB2. One of the performance improvements for IDAA was to expose the DB2-internal record layout, which avoids conversions to some external format. The UNLOAD utility writes the data to a Unix Systems Services (USS) named pipe so that no persistent storage is needed. The IDAA stored procedure `ACCEL_LOAD_TABLES`, running in DB2 for z/OS, reads the data from the named pipe and sends it directly on to the IDAA. A light-weight variation of the DRDA protocol [DRD03] is used for the communication, in particular to exchange control information.

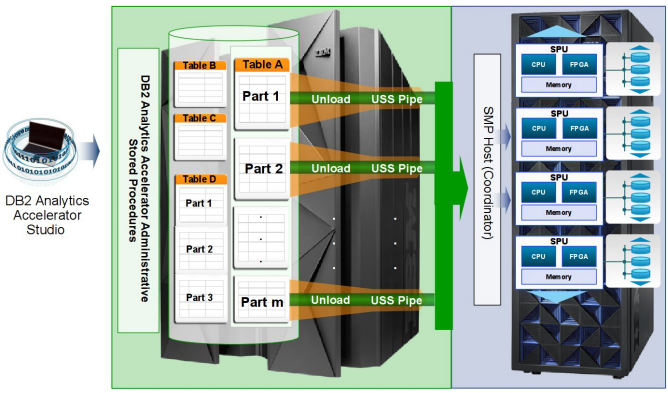

Fig. 7: Traditional Load Architecture

On IDAA server side, the coordinator receives the data and writes it into a named pipe. Again, copying of the data is minimized and no persistent storage is used. The Netezza backend reads from the named pipe when inserting the data into its table. Netezza does the only data conversion from the DB2-internal row format[4] to its own format and to write it to data pages.

Such an end-to-end pipeline is called *load stream*. If a table is partitioned in DB2, multiple partitions are processed in parallel by instantiating multiple load streams. Non-partitioned tables use a single load-stream only. The customer can configure the degree of parallelism, which is the maximum number of concurrent load streams. That configuration applies to a single invocation of the `ACCEL_LOAD_TABLES` stored procedure.

This design delivered a throughput of $1TB/h$ right from the start. The aforementioned support for the DB2-internal row format, and avoiding copying and data transformation steps within DB2 for z/OS and IDAA gave another boost in later versions. Now the bottleneck is the network bandwidth between DB2 and IDAA. Combining (bonding) active and failover network lines resulted in an end-to-end throughput of more than $4TB/h$ under lab conditions.

3 Autonomous Load Performance Tuning

Despite the excellent performance numbers, the load architecture described in Sect. 2.3 does have some caveats. Tables holding a large volume of data are required to fully saturate a load stream. When the data volume is too small, the overhead for establishing a load stream becomes more and more noticeable.

In one situation, a customer wanted to synchronize several thousand, mostly empty or nearly empty tables using a single `ACCEL_LOAD_TABLES` stored procedure call. The total elapsed time exceeded the available time in the nightly batch window. Sequentially loading the tables led to a mostly under-utilized system. The solution was straight-forward: determine

[4] Netezza was extended for this IDAA scenario to understand the DB2-internal row format.

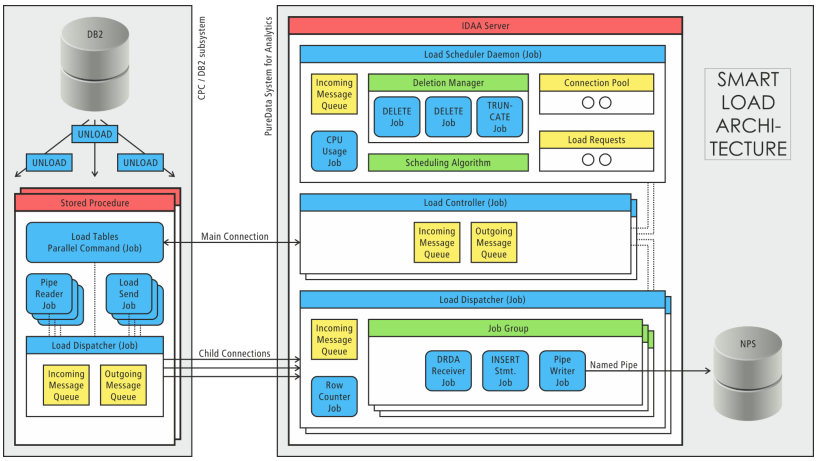

Fig. 8: Smart Load Architecture

clusters of tables to load together, and use multiple concurrent stored procedure calls. Still, such an approach cannot be called *easy* to use. And the customer would still have to take care of managing the parallelism for the concurrent stored procedure calls and, in addition, across different DB2 systems that are connected to the same accelerator.

3.1 Smart Load Architecture

The new design, called *Smart Load* is shown in Fig. 8. The central change (compared to Fig. 7) is the introduction of the *Load Scheduler Daemon* on the IDAA server. It is responsible for managing and scheduling all concurrently running load operations.

The *Load Controller* handles a single `ACCEL_LOAD_TABLES` stored procedure call and the tables specified therein. The controller determines all its work units. A work unit is an unpartitioned table or a partition of a partitioned table. A work unit descriptor contains (amongst other details) information about the data volume. All work unit descriptors are passed to the load scheduler, who adds them to its pool of load requests.

The scheduler itself implements an optimization algorithm to determine the order in which the remaining work units shall be processed. Different state-of-the-art algorithms can be used for that.We have implemented a simple bin packing approach. The key criteria for the specific algorithm is that it produces an initial result very fast and subsequently improves on that even with a partially changed set of work units.

In addition to defining the sequence in which work units shall be processed, the load scheduler monitors the current system utilization of the accelerator. The following three classic cases are handled:

Under-utilization The scheduler determines that there are still sufficient resources available to start a new load stream. Parallelism increases due to that, and load throughput increases as well.

Over-utilization The accelerator is under pressure in terms of available resources. In order to remedy the situation, the load scheduler reduces the parallelism. That means, when the next work unit finishes, the load stream used for it will not start processing a new work unit. Instead, it is kept as a passive load stream[5] until parallelism can be increased again or some other load controller relinquishes one or more of its load streams at its end.

Normal utilization The system is neither over-utilized nor does it have sufficient free resources to start another load stream (or no work unit requiring another load stream is pending). The parallelism setting remains unchanged.

The system utilization is determined by monitoring CPU, I/O, and network utilization. The resources typically used for a single load stream are determined from currently or previously running load operations. Additionally, the current throughput of active load requests is considered as another factor for reducing the number of active load streams. Hence, the algorithm to determine whether a new load stream can be started, is simple and robust. A condition – like the following for CPU utilization – is applied for all relevant measures.

$$current\_CPU\_Utilization + expected\_Load\_Stream\_CPU\_Utilization < 100\%$$

The decision to increase the parallelism and starting another load stream is made jointly with the *Load Dispatcher* running on the DB2 for z/OS side. The load dispatcher monitors the system utilization there. So if the load scheduler wants to allow a controller to start a new load stream, the controller informs the dispatcher, and only if the dispatcher agrees will the load stream be started. In case the dispatcher disagrees, the load scheduler may pick another controller to use the available resources. Thus, we have implemented a distributed scheduling and workload balancing algorithm specifically tailored to IDAA's needs. Since we measure general performance metrics, our approach also takes concurrently running (non-load) workload into account – without introducing any dependencies on the code level.

Note that it is possible that oscillation effects may appear, i. e., if the scheduler increases parallelism due to free resources, the next calculation may detect an under-utilization, and so on. That is not a problem here, however. First, the load scheduler only makes a decision whether to increase the parallelism by 1 (one), which prevents large spikes. So even if an over-utilization occurs, it only exceeds the available resources by a small amount. Second, changes in the degree of parallelism only become active when new work units are added to the scheduler's pool or when a work unit has finished its data transfer. Therefore, the changes themselves are relatively rare. Third, once scheduled, transfer operations are never aborted due to a change in parallelism which avoids unnecessary restart operations.

This approach not only increases the efficiency for batched table loads but also significantly reduces administration complexity. Customers merely have to state once which tables

[5] In the past, a load stream was created for each partition of table. That includes the creation of the TCP/IP connection (denoted as child connection in Fig. 7) as well as creating and starting the various threads. Creating the TCP/IP connection and threads once and pooling them for reuse gave a significant improvement on its own already for the small tables scenario.

shall be loaded and synchronized without the need to manage multiple stored procedure invocations to exploit parallelism. Additionally, our system automatically tunes itself to the best resource exploitation. There is no need to set some configuration parameter or to do some initial calibration. The load scheduler continuously calibrates itself automatically.

4 Preliminary Performance Evaluation

We used our prototype to run some initial tests to gain a better feel for the improvements. The test used an older (and rather slow) Netezza system as backend.

First, we loaded 50 unpartitioned tables, and each table stored only a single row. A single row is the minimum possible amount of data per table because the IDAA load process has an optimization for empty tables where it skips all data transfer and processing. In particular, no DB2 `UNLOAD` utility is started.

Tab. 1 shows the measured execution times. Even if the times may vary depending on the actual data, the results clearly show that tearing down and re-establishing the load stream for each work unit (as was done in the past) adds a significant overhead for such small-sized tables. Furthermore, loading all tables sequentially looses a lot of potential, showing that inter-table parallelism is important. Extrapolating the improvement of about 20x from our 50 tables to 20,000 tables results in several hours of savings.

Scenario	Load Time
Smart Load	13s
Old (sequential) Load	4min 16s

Tab. 1: 50 Tables with 1 row each

Tables with just a single row (or very few rows) are extreme cases – not uncommon for our customers, but not the norm either. That's why, we also tested a TPC-H scenario with a scale factor of 30 MB. The improvements are not as dramatic as above because data transfer and processing take a larger slice. Still, the results in Tab. 2 show an improvement by factor 6.

Scenario	Load Time
Smart Load	7s
Old (sequential) Load	42s

Tab. 2: TPC-H Workload

5 Summary

In this paper we have presented a prototype that extends the IBM DB2 Analytics Accelerator with smart load scheduling capabilities. It simplifies the usage of the accelerator for data maintenance via batch loading dramatically. Customers no longer have to manage the resources consumed on the accelerator across different connected DB2 systems. Instead,

IDAA actively monitors all systems involved and automatically scales the parallel degree for processing load requests. The net result is an overall improvement for throughput of batch load operations. We expect a lot of satisfaction from the much simplified user experience.

Next steps for our work will be the integration of the prototype into the product. More intensive performance measurements will be forthcoming. We expect that different scheduling algorithms may show some impact, but given the typical pattern of data load, it is unlikely to observe a major impact. Nevertheless, we intend to verify our assumption empirically.

6 Trademarks

References

[Ba15] Baumbach, Ute; Becker, Patric; Denneler, Uwe; Hechler, Eberhard; Hengstler, Wolfgang; Knoll, Steffen; Neumann, Frank; Schoellmann, Guenter Georg; Souissi, Khadija; Zimmermann, Timm: . Accelerating Data Transformation with IBM DB2 Analytics Accelerator for z/OS. IBM Redbooks, 2015.

[Be12] Beaton, A.; Noor, A.; Parkes, J.; Shubin, B.; Ballard, C.; Ketchie, M.; Ketelaars, F.; Rangarao, D.; Tichelen, W.V.: . Smarter Business: Dynamic Information with IBM InfoSphere Data Replication CDC. IBM Redbooks, 2012.

[BSM16] Beier, Felix; Stolze, Knut; Martin, Daniel: Extending Database Accelerators for Data Transformations and Predictive Analytics. In: Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016, Bordeaux, France, March 15-16, 2016, Bordeaux, France, March 15-16, 2016. S. 706–707, 2016.

[DRD03] The Open Group. DRDA V5 Vol. 1: Distributed Relational Database Architecture, 2003.

[Fr11] Francisco, P.: . The Netezza Data Appliance Architecture: A Platform for High Performance Data Warehousing and Analytics. IBM Redbooks, 2011.

[IB13] IBM: Sychronizing data in IBM DB2 Analytics Accelerator for z/OS. Bericht, IBM, 2013. http://www-01.ibm.com/support/docview.wss?uid=swg27038501.

[IB14a] IBM: . DB2 11 for z/OS, 2014. http://www.ibm.com/support/knowledgecenter/api/content/SSEPEK/db2z_prodhome.html.

[IB14b] IBM: . IBM Netezza Analytics – In-Database Analytics Developer's Guide, Release 3.0.1, 2014.

[IBM13] IBM. IBM InfoSphere Data Replication V 10.2.1 documentation, 2013.

[St13] Stolze, Knut; Köth, Oliver; Beier, Felix; Caballero, Carlos; Li, Ruiping: Seamless Integration of Archiving Functionality in OLTP/OLAP Database Systems Using Accelerator Technologies. Datenbanksysteme fÃ¼r Business, Technologie und Web (BTW) 2013 : Tagung vom 11. - 15. MÃ¤rz 2013 in Magdeburg, S. 383–402, March 2013.