

The Inverted Data Warehouse Based on TARGIT Xbone

How the Biggest of Data Can Be Mined
by “The Little Guy”

Morten Middelfart^(✉)

TARGIT, Tampa, FL, USA
morton@targit.com

Abstract. We present TARGIT’s Xbone memory-based analytics server and define the concept of an Inverted Data Warehouse (IDW). We demonstrate the high-performance analytics properties of this particular design, as well as its resistance to failures. Additionally, we present a large scale solution in which TARGIT Xbone and IDW are implemented incorporating Google search data. The solution is used for so-called Search Engine Optimization (SEO) and can reveal interesting information about Google’s algorithmic behavior on specific searches. Finally, we demonstrate the combined TARGIT Xbone and IDW to be very cost-effective and thus available to small enterprises that would normally not benefit from Big Data analytics.

1 Introduction

Publicly available data, such as social data and open data, play a big part of business decision-making, and the solutions within an organization have to be ready to accommodate that. In this context, it is notable that there are different growth paces for the internal and the external data available. In other words, the data we have in a company will likely grow at a pace close to the growth rate of the company itself. From a numbers perspective, very few companies have growth above 100 % for long; a big company will be considered successful with yearly growth rates of 20–30 %.

According to IDC’s most recent estimate (2011) [3], there are 1.8 zettabytes of digital data in the world, projected to grow to 7.9 zettabytes by 2015. (A zettabyte is 10^{21} or 1,000,000,000,000,000,000 bytes, 1 trillion gigabytes, or 1 quadrillion megabytes.) The pace of this growth is driven chiefly by external data, rather than internal data. Therefore, we are steadily approaching a tipping point at which the majority of data relevant for competitive analytics will be coming from outside the organization. Since at this point we will have to compete on data that doesn’t “belong” to us, it’s a total paradigm shift on the way business decisions are made. Data that’s available today may be gone or made unavailable by its owner tomorrow, and there may never be full knowledge

of data quality. But it’s still pertinent to the decision to be made now, and it’s critical to analyze that data to uncover the useful bits of information that can make that decision more informed.

It is important to note that very few companies will have the ability to download all the external data they need for competitive analytics; this ability will only be available to very few companies with massive resources. In other words, most companies (those below the Fortune 500 level, perhaps) will have to sample the relevant external data and accept the fact that these data represent an incomplete, imperfect, yet relevant slice of information. Despite the imperfection, this data can offer an understanding of trends and early indicators that a business may want to analyze deeper. During such a sampling process, the ability to flexibly add, remove, and analyze data sets is crucial.

Another part of this sampling process is that it allows for an intelligent distinction of useful and non-useful data. For example, a user can identify irrelevant data as early in the process as possible, and thereby drastically reduce the amount of data to be processed. Or instead, focus on pushing as much of the workflow toward the source of the sample before the sample is extracted in order to reduce the workload further.

The inverted data warehouse (IDW) is structured differently from the traditional data warehouse in that:

1. It is physically distributed to multiple machines close to the end-users for optimal analytical performance, and
2. It is deliberately designed for continuous sampling of a big data source.

In a traditional data warehouse all data are stored on a centralized server system where the analytics processing is also done. In the IDW we push all data and analytics processing to the clients, thus we refer to this architecture as *inverted* compared to the traditional.

The TARGIT Xbone server technology [12] is designed to analyze a data set on-the-fly without the traditional Extract Transform Load (ETL) process. In other words, it is optimized for working with sampled data in a very efficient way, as it requires very little effort to assess and sample. Additionally, its in-memory architecture allows it to deliver a very responsive user experience.

The remainder of this paper is structured as follows: in the following section we discuss related work. In Sect. 3, we present the general principles of the Inverted Data Warehouse, along with a case based on Google search data. In Sect. 4, we present the TARGIT Xbone architecture as well as the architecture of TARGIT Decision Suite in general. In Sect. 5, we present an assessment of the data volumes analyzed, as well as an experimental evaluation of the performance of our IDW case from Sect. 3. Finally, in Sect. 6, we conclude and present future work.

2 Related Work

The building of a “classic” data warehouse with an Extract-Transform-Load (ETL) process has been very well documented in [4]. This work describes the

continuous learning process from data acquisition to presentation as a cycle of improvement.

In [5,6], a decision-centric cycle is described, in which a user travels through Observation Orientation Decision Action (OODA) with as few interactions as possible. This work primarily deals with user interaction with data in a data warehouse, and furthermore, how to render a user capable of making very fast fact based decisions.

In traditional Business Intelligence software, users perform two tasks to create the report desired. First, the user defines a query to examine a subset of data stored within the organization's data warehouse. And second, the user defines a graphical representation of the data that was retrieved. Typically, the presentation means are selected from a report generator or "chart wizard", from which default layout properties (or user-specified layout properties) are identified before making the presentation.

Conventional Business Intelligence software requires users to adhere closely to formal syntax, as the underlying databases require syntactically correct queries to properly retrieve information. Conventional Business Intelligence solutions are frequently configured with user interfaces that guide a user directly to a presentation of retrieved data, often through a wizard. In addition, a user is expected to understand the correct principles of this utilizing the data found [1].

Close conformance to formal syntax places a burden on the user to know the rules of database syntax and to be able to work quickly within them to retrieve the data desired. US Patents 7,783,628 [8] and 7,779,018 [7] identify two methods for making a presentations of data using so-called meta-morphing. The technology allows users to enter a business question through the user interface:

1. I would like to see 'cost' grouped by 'time, month'
2. I would like to see 'revenue' grouped by 'time, month', 'customer, group' and 'product, name'
3. I would like to see 'revenue'
4. I would like to see 'country'

Example: A user submits the question "I would like to see customers".

Step 1: parses all words in the sentence representing the questions (Fig. 1(b)), and these words are subsequently matched against the meta-data of the data warehouse (Fig. 1(a)). If a word in the sentence is not matched in the meta-data it is simply thrown away. The output from Step 1 is a set of dimensions and/or measures; and if the set is empty, the meta-morphing process is simply terminated. In our example, the only word that will remain from this parsing is "customers".

Step 2: compensates for the problem of the user's question containing only measures or dimensions. In a traditional system, asking incomplete questions like "I would like to see customers" or "Show me revenue" would at best return a list of customers (the members of the customer dimension) or the sum of revenue for all data in the data warehouse. By creating an association between

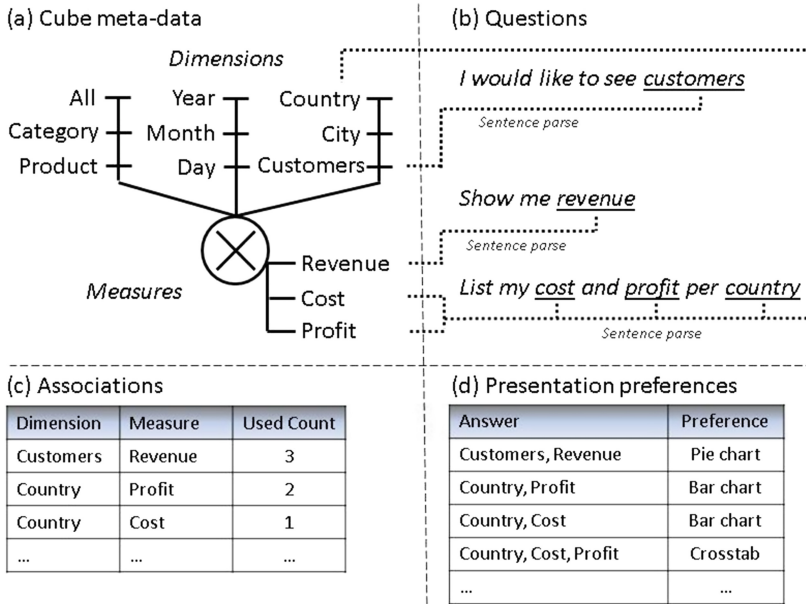


Fig. 1. The meta-morphing model

measures and dimensions (Fig. 1(c)), the system will learn the *individual* user's behavior based on what he clicks, e.g., if he clicks to select revenue and customers at the same time, then an association between revenue and customers will be created. Therefore, the answer to both questions will be a list of customers with their respective revenue. Associations are also created while the user loads any analysis, meaning that he does not need to actively pose a question including the association. This means that the user will simply feel that he is receiving information in the way he is used to. In the event that a user has never seen or asked for a given relationship, the meta-morphing process will look into which dimension or measure the user most often uses, and then create an association that is used most often, i.e., the measure or dimension from the association with the highest Used Count (see Fig. 1(c)). The output of Step 2 is a combination of measures and dimensions.

Step 3: is, given Step 2 output, a trivial query in the data warehouse retrieving "revenue" for each member on the "customers" dimension. The output of this step is a dataset as well as the dimensions and measures used to provide it.

Step 4: preferences are created for each user given the way they would normally see an answer given its dimension/measure combination, e.g., customer/revenue is usually displayed as a pie-chart (see Fig. 1(d)), profit per country is usually displayed as a bar-chart, etc. Given the "experience" with the user's preferences that are collected whenever the user sees or formats a piece of information, the dataset received from Step 3 is formatted and displayed to the user. In the

event that no preferences have been collected for the user, an expert system will inspect the dataset and make a call for the best presentation object (pie, bar, geographic map, table, etc.), this expert system is based on input from a TARGIT BI domain expert. In our example, the returned revenue for each customer will be presented as a pie chart based on the data in Fig. 1(d).

Although providing some degree of user freedom, the options provided in US Patents 7,783,628 [8] and 7,779,018 [7] in above example are limited to possible combinations of measures, dimensions and criteria (where “I would like to see” was part of a rigid syntax).

US Patent 8,468,444 [9] introduces so-called Hyper Related OLAP which leverages the meta-morphing functionality, and thus allows a user to apply meta-morphing in one interaction (typically mouse click) on any report or dashboard in a Business Intelligence implementation. The visualizations In this case are aligned with best practices [1]. This functionality adds a lot of analytical freedom to a decision process since it essentially moves the user from the observation to the orientation phase in one interaction. However, the freedom is not complete since the analytics is limited to the dashboard or reporting context from which the Hyper Relation was initiated.

The idea to disregard the vast majority of results in a streaming set of Big Data has been presented by European Organization for Nuclear Research (CERN) in relation to their experiments with their Large Hadron Collider (LHC). In [2], it is stated that they “throw away” 99 percent of what is recorded by the sensors in the LHC.

The novel contribution of this paper is to apply the same principles as CERN to a Big Data source (Google) in a way such that it is achievable for a small or medium-sized enterprise. In this context, we break with the traditional ETL principles in data warehousing by running data sample queries continuously while at the same time, synchronizing the resulting data set with multiple analytics client nodes.

We benefit from the same user-friendliness achieved through US patents [7–9], however, we operate on a metadata layer that is dynamically created based on reading the raw data is synchronized using TARGIT Xbone server.

3 The Inverted Data Warehouse IDW

3.1 Architecture

As shown in Fig. 2 the architecture of the Inverted Data Warehouse (IDW) is significantly different from a traditional data warehouse. In the traditional data warehouse, there are a number of data sources that are queried with a certain frequency (this is known as the extraction part of ETL). From this point, the data are transformed in a staging area before being loaded into cubes. The entire ETL process runs end-to-end and its three steps are synchronized. In addition, a traditional data warehouse would normally not be operational in the event that the ETL process fails, since users typically query the cubes loaded on a centralized server.

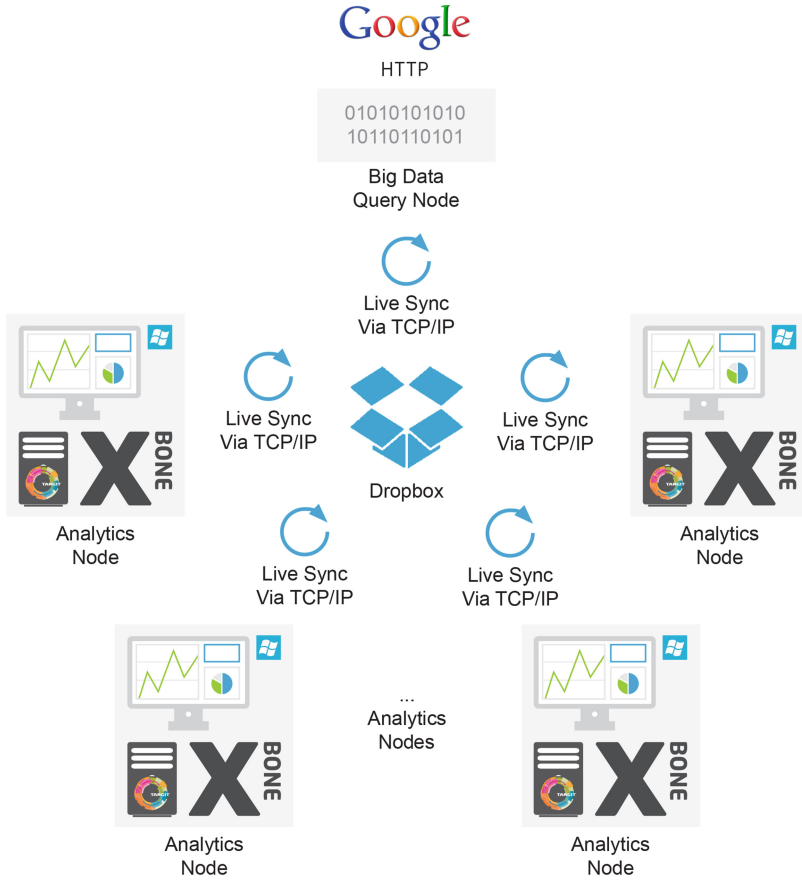


Fig. 2. An Inverted Data Warehouse with four analytics nodes

In an IDW, we run continuous queries as close to the data source as possible, since we intend to push as much of the workload to the source as possible. In this case meaning that we run a query on Google and only concentrate on reading the result of this query. In addition we attempt to ignore all undesired data as early as possible; ideally at the source. It is important to note that the work done on the node interacting with the data source (Google in this case) is not synchronized with any of the nodes analyzing the output. Synchronization of data with the analytics nodes is done using a separate process (in our example in Sect. 5 we use the cloud-based service Dropbox www.dropbox.com). As we will see in Sect. 5, the number of relevant records from a business perspective is eight orders of magnitude smaller than the records processed on Google, and thus the volume needed to be synchronized between the nodes is completely manageable for this type of architecture.

Since all the data selected for analysis are mirrored to all analytics nodes, the IDW has no single point of failure in regards to users being able to analyze data. In the event that the node with the continuous query fails, all users will simply experience that data does not continuously update until the node is back online. In the IDW architecture it is of course possible to compensate for this single point of failure by adding more nodes to query the data source. However, given the fact that end-users are unaffected from a business standpoint in our example (Sect. 5), we have not explored this variant further in this article.

The IDW in this article dynamically works with the following formats: NoSQL records on source (Google internal storage), HTML documents as output from Google queries, flat-file structured .txt files with selected data from HTML, and column-store format data on TARGIT Xbone that are presented in dimensions and measures to the analyst user in TARGIT Decision Suite.

Example: we test the query “analytics dynamics ax” on Google to see if certain named vendors appear in the first 100 results and in which order.

Query Node

Step 1: the query is sent from the query node to Google as a http request: www.google.com/search?hl=en&lr=&q=analytics+dynamics+ax&btnG=Search This step repeats until the first 10 pages are read since we are interested in the first 100 Google results in this as our sample.

Step 2: the relevant links in the resulting document from Step 1 are identified by the tag `<h3 class="r">`. Furthermore, the links are matched against patterns where one or more identifies the link as belonging to a certain vendor, e.g., the pattern `targit.com` identifies TARGIT as the vendor associated with the link www.targit.com/en.

Step 3: only the links that belong to vendors of interest, defined by the patterns in Step 2, are stored by appending a semicolon separated text file in a Dropbox folder with the following format: `day;vendor;page;search;rank`

Analytics Node

Step 4: upon each update of the text file from Step 3, the TARGIT Xbone server loads the updated dataset into memory and identifies all data items that can be used as measures, dimensions, or both. In addition, TARGIT Xbone will identify hierarchies based on identifying date/time or members that fully include subsets of other members (the latter can be done effectively using the fast distinct count abilities). In this example, only the data item “day” in the dataset from Step 3 has the potential for becoming a hierarchy; all other data items become flat dimensions. In addition, “rank” also becomes a measure (in addition to a dimension). Once a new dataset is fully loaded, any previous version of the dataset is flushed from memory. By doing loads in this order, the analytics user does not feel the update as downtime.

Steps 1–3 are run continuously on the query node in a cycle covering all the queries we want to test. Step 4 is run continuously on the analytics node.

In our experience with running the IDW for a four-month period, we never saw the “query source” node fail, and users experienced close to real-time performance when answering business questions. We therefore note that we had a robust and high performing analytical environment based on commodity hardware and software. The data volumes and the analytical performance are further discussed in Sect. 5. The analytical performance optimizations using in-memory storage, solid-state drives (SSD), and column-store architecture on the individual nodes will be described in the following section.

3.2 Business Perspective

For nearly any company in business, a share of their customers will discover them through search engines on the Internet. Therefore it is highly relevant for a company to understand and optimize its position in search results delivered to potential customers looking for products or services.

It is very likely that a company already has a basic understanding of what their customers are looking for, and therefore it is also very likely that they can guess which sorts of questions a potential customer will pose to a search engine. This leads to the first element of reducing the big data at the source, which we call “the shotgun approach”. In the shotgun approach, we simply guess which questions a potential customer would “ask” a search engine. We do this from a very greedy perspective, meaning that we fire off a lot of queries that a user may or may not ask. Our intent is to overshoot the number of queries, such that we are sure that most real end-user queries are a subset.

Philosophically, one could say that we are proposing a hypothesis that a user may ask a certain question, and then we test how relevant our company is in the results delivered by the search engine. We assess the relevancy of the query based on the position in the search results, meaning that the top result is very relevant, whereas position 101 and beyond is completely irrelevant. By capturing the position between 1 and 100 of our company as well as our competitors, we now have a useful tool to improve our own position but also to understand how our competitors are optimizing/communicating.

The shotgun approach allows us a very useful tool for continuously operationally optimizing our “relevancy” in the business space we desire, while at the same time understanding competitors’ communication in the space where we expect potential customers. This presents some very interesting strategic information.

The shotgun approach is therefore very useful from a business standpoint, and it is very good at significantly reducing the data volumes we analyze. In Sect. 5, we present metrics for the Big Data size before and after applying “the shotgun”.

4 TARGIT Xbone and TARGIT Decision Suite

On each analytics node from Fig. 2, we have installed the TARGIT Decision Suite as well as the TARGIT Xbone server [12]. This means that the node is a self-contained analytics client to which data sets can be synchronized. As seen in Fig. 3, the analytics node has three layers: TARGIT Decision Suite, TARGIT ANTserver, and TARGIT Xbone.

TARGIT Decision Suite is the interface where all analytics “questions” are asked and answered to the user. This user interface is designed in accordance with the Computer Aided Leadership and Management (CALM [5]) philosophy in mind, meaning that we seek to reduce the number of interactions needed for a user to travel a complete observation-orientation-decision-action cycle. In order to allow this, we apply the specific patented technologies [7–9] described in Sect. 2. We note that these technologies support both the OODA cycle and the creation of analytics from scratch.

TARGIT ANTserver is an intermediate layer that handles connections to multiple databases, while at the same time storing metadata in multiple languages. In this particular case, we only focus on the connection to TARGIT Xbone (below), but this server would allow access to more than this if needed.

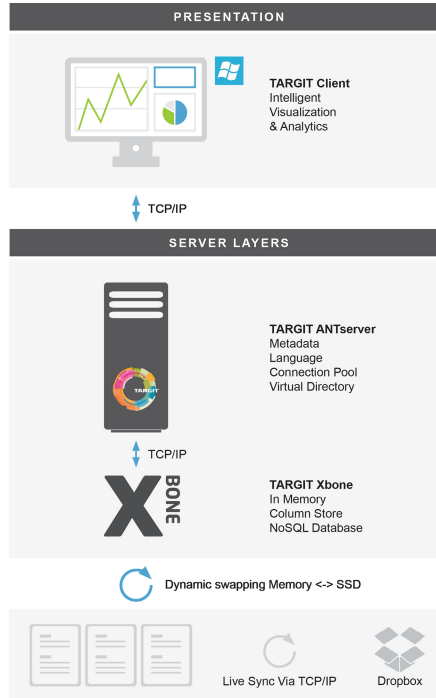


Fig. 3. Analytics Node layer overview

TARGET Xbone is a high-performance analytics server that does all the computations requested from TARGET Decision Suite through TARGET ANTserver. TARGET Xbone has its name because it is a very lean design “stripped to the bone” to reduce overhead, while incorporating large RAM memory stores as well as fast solid-state drives. TARGET Xbone can automatically detect the structure of the dataset and therefore requires no configuration to allow the user to analyze it (see Sect. 5). The data sets in TARGET Xbone are flat files, and the server dynamically swaps these between memory and SSD whenever the data is needed or when memory needs to be freed for the data. Upon loading data from SSD to RAM it is stored in a column store format such that each dimension is compressed to a “column” of unique values under which the references to the original rows are stored. This format is very fast in delivering subset information, especially when one or more measures are presented over one dimension. In addition, we note that this architecture is extremely fast in performing operations like minimum, maximum, and distinct counts since these can be found by reading only one value for a given column (dimension) with no need to access information on a row level. In Sect. 5 we present TARGET Xbone experiments on real-world data that support the expected performance.

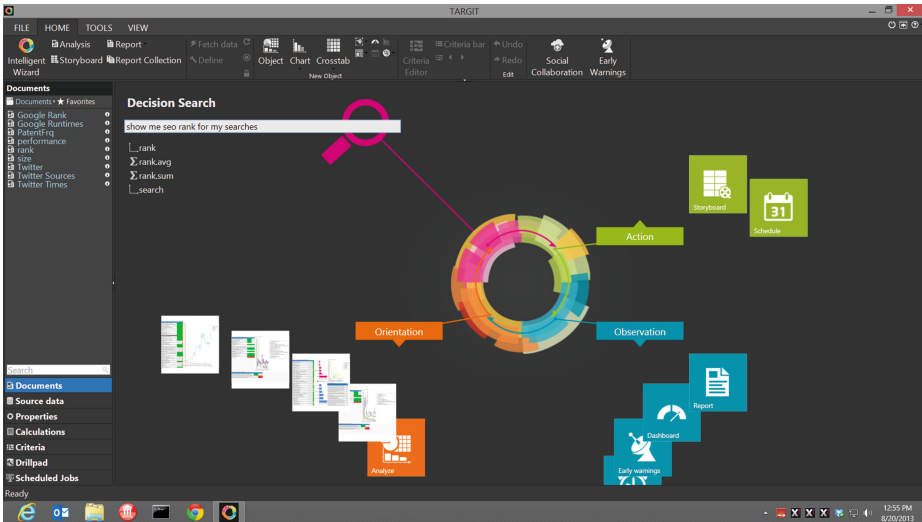


Fig. 4. TARGET Decision Suite with Intelligent Wizard

It is important to note that the end-user only interacts with the TARGET Decision Suite, thus the entire underlying setup is completely transparent. The user will be greeted by TARGET’s Intelligent Wizard as shown in Fig. 4, and from there begin asking “questions” in either human language or by interacting with the interface in a more traditional GUI-way. In Fig. 4, we note that the user simply asks “show me SEO rank for my searches” and the system presents

him with both existing and analysis as well as options to create something new on-the-fly. The Intelligent Wizard has described in detail in [10], so in this paper the user simply clicks the existing most left analysis and the output is shown in Fig. 7.

In the following section we will see how the IDW architecture combined with the TARGIT software delivers an effective and very intuitive analytics experience.

5 Experiments

In the experiments we perform on the IDW, we use one query node and ten analytics nodes. The analytics nodes reside both within and outside the TARGIT organization. As we note from Figs. 2 and 3, the physical location and number of analytical nodes will have no impact on an individual node’s performance. Therefore we limit our tests to one analytics node in this IDW.

The query node is a dual-core AMD Opteron 2 GHz based server with 1 GB RAM, running Windows Server 2008 R2 64 bit, and physically located in Denver, Colorado. The analytics node is a quad-core Intel 3.40 GHz CPU with 12 GB RAM, running Windows 8 64 bit, and physically located in Tampa, Florida. In addition, the analytics node is running TARGIT Decision Suite and TARGIT Xbone.

In Fig. 5, we see the number of records queried and downloaded respectively over a 13-day period. We see that the data found on this server side varies from 24.5 to 32.5 billion records per day, whereas the data actually pushed to the analytics node varies only from 375 to 675 rows per day. In other words, we reduce the data size eight orders of magnitude by limiting the data to a sample relevant to us (in this case top-100). It is important to note that we do not limit ourselves analytically in any way as we still include all pages and competitors;

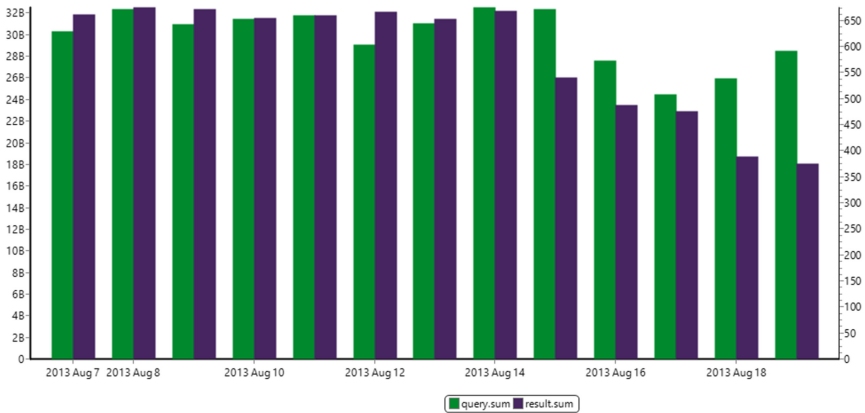


Fig. 5. Number of records queried and rows stored on the Query Node.

rather, we simply eliminate the irrelevant data similarly to [2]. We deliberately use the term records about the format a given website is stored on Google since we assume the source is a NoSQL format, whereas the resulting dataset we produce is a structured flat format with columns and rows.

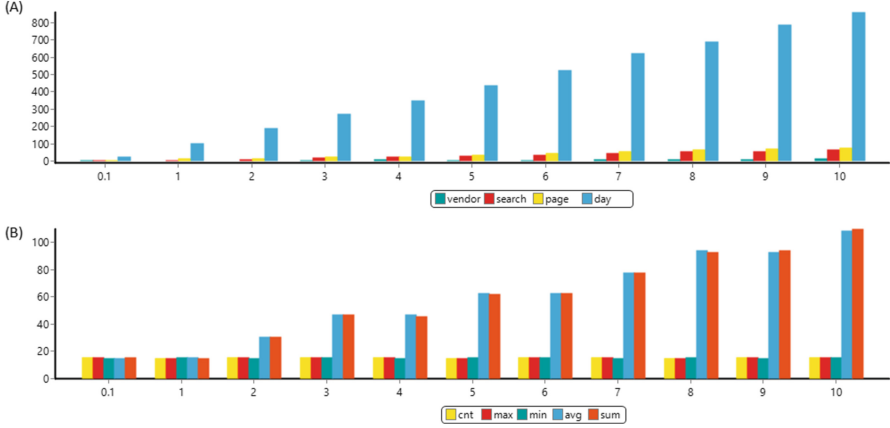


Fig. 6. Performance in milliseconds over millions of rows in TARGIT Xbone

In Fig. 6, we see the performance on the analytics node. Note that the run-times are in milliseconds and the size of data sets are in the millions of records. The smallest data set (about 0.1 million records) is the real size of the data set, thus in these experiments we scale two orders of magnitude bigger than the actual system. In Fig. 6(A), we note that even at the largest synthetic dataset, we still have response times less than one second. In Fig. 6(B), we notice the distinctly different behavior of a column store [11] compared to a traditional table, since performance of count, minimum, and maximum operations are substantially different from the performance of sum and average operations. This is due to the fact that TARGIT Xbone stores data in columns rather than rows, and therefore counting, minimum and maximum can be directly read without even going to the row-level (see Sect. 4).

Referring back to Fig. 4, we now see the output of the “natural language question” in Fig. 7. As mentioned earlier in Sect. 4, we choose one of the existing visualizations in this case. The intelligent wizard supports “answering” most natural language queries with meaningful visualizations [10], but in this case we stick with a predefined visualization as shown above.

As shown in Fig. 7, we see how well our organization’s website (in this case `targit.com`) ranks on a Google search for “analytics dynamics ax”. In other words, a reasonable guess for what a potential customer might ask when searching for analytics software to complement a Microsoft Dynamics AX (ERP) system. On the analytics node, the user is able to explore each guessed query and

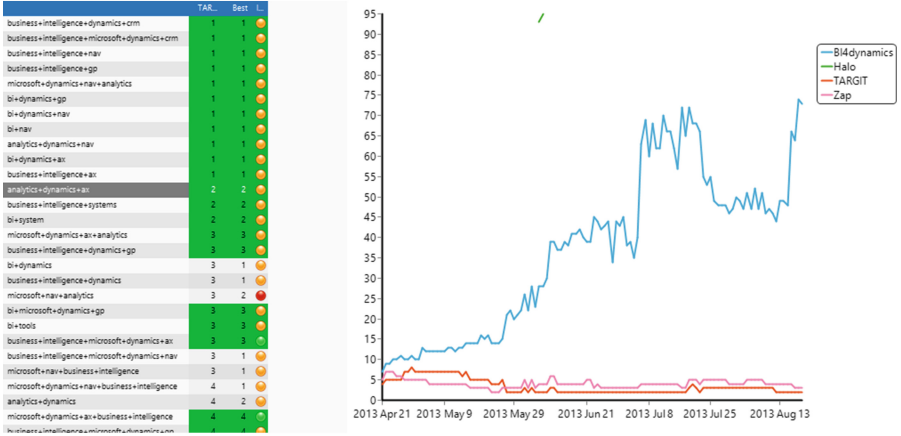


Fig. 7. Example of Intelligent Wizard output on an Analytics Node (Color figure online).

rank his organization as well as competitors. As we can see in Fig. 7, the competitors show distinctively different behavior: one competitor seems to be competing for that position as number one in “analytics dynamics ax” whereas to other competitors are not fighting for this space at all. Therefore, the information that we see does not only show our ability to attract customers; it also shows the priorities of the competition. Information that allows an organization to understand not only its current position, but also competitive strategic intent, is highly valuable to most organizations.

Qualitative Assessment: During the four-month period that this showcased IDW implementation has been operational, the information from the IDW has helped TARGIT to improve its average ranking on Google from 44 to 7 within all search *sentences* deemed strategically important. In addition, TARGIT is ranking number one on specific (confidential) search sentences. We note that the information about search ranking at this “atomic level” is not available on any other tool, e.g., Google Analytics, since the information is too specific for those tools to collect. We therefore conclude that the combined IDW with the “shotgun approach” delivered highly valuable and strategically important information. We also note that commodity hardware in very limited amounts was able to produce these results, even though the daily query results at the server end amounted up to 32 billion records.

6 Conclusion

In this article, we presented a so-called Inverted Data Warehouse (IDW) with analytics nodes running on TARGIT’s Xbone memory-based analytics server. We demonstrated a real-world implementation, in which strategic information was generated with each query running in less than 0.1s. In addition, we demonstrated that similar queries could be run on a 2 orders of magnitude larger

synthetic volume in less than one second. All of this was done using commodity hardware and in accordance with the principle of pushing the Big Data workload towards the server, and furthermore by eliminating irrelevant data at an early stage.

We demonstrated in concrete examples how the entire solution is highly relevant for an organization's ability to navigate in a competitive market. In addition, we showed how this specific implementation of IDW can be used for Search Engine Optimization (SEO). In summary, we have demonstrated how an organization with very little financial ability can conduct highly valuable data mining and visualization on a Big Data source.

Acknowledgments. This work was supported by TARGIT and Center for Data-Intensive Systems (Daisy) at Aalborg University.

References

1. Few, S.: Show Me the Numbers: Designing Tables and Graphs to Enlighten. Analytics Press, US (2012)
2. Cern, N.H.: Where the Big Bang meets big data. TechRepublic. www.techrepublic.com/blog/european-technology/cern-where-the-big-bang-meets-big-data/, as of 14 Aug 2013
3. Roe, C.: IDC summary. The Growth of Unstructured Data: What To Do with All Those Zettabytes? Dataversity. www.dataversity.net/the-growth-of-unstructured-data-what-are-we-going-to-do-with-all-those-zettabytes/, 12 Aug 2013
4. Kimball, R.: The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses. Wiley, New York (1998)
5. Middelfart, M.: CALM: Computer Aided Leadership and Management. iUniverse (2005)
6. Middelfart, M.: Improving business intelligence speed and quality through the OODA concept. In: Proceeding of DOLAP, pp. 97–98 (2007)
7. Middelfart, M.: Presentation of data using meta-morphing. US Patent 7,779,018. Issued 17 Aug 2010
8. Middelfart, M.: Method and user interface for making a presentation of data using meta-morphing. US Patent 7,783,628. Issued 24 Aug 2010
9. Middelfart, M.: Hyper related OLAP. US Patent 8,468,444. Issued 18 June 2013
10. Middelfart, M.: Intelligent Wizard for human language interaction in Business Intelligence. To appear in eBISS 2013
11. Wikipedia. Column-oriented DBMS en.wikipedia.org/wiki/Column-oriented_DBMS, 21 Aug 2013
12. TARGIT. TARGIT Xbone - Ad-hoc analytics for everyone www.targit.com/en/experience-targit/products/xbone, 12 Aug 2013

Enabling Real-Time Business Intelligence

International Workshops, BIRTE 2013, Riva del Garda, Italy, August 26, 2013, and BIRTE 2014, Hangzhou, China, September 1, 2014, Revised Selected Papers
Castellanos, M.; Dayal, U.; Pedersen, T.B.; Tatbul, N.
(Eds.)

2015, XX, 175 p. 80 illus., Softcover

ISBN: 978-3-662-46838-8