Journal of Computing and Information Technology - CIT 20, 2012, 4, 257–264
doi:10.2498/cit.1002081

Practical Implications of Real Time Business Intelligence

Dale Rutz, Tara Nelakanti and Nayem Rahman

Intel Corporation, USA

The primary purpose of business intelligence is to improve the quality of decisions while decreasing the time it takes to make them. Because focus is required on internal as well as external factors, it is critical to decrease data latency, improve report performance and decrease systems resource consumption. This article will discuss the successful implementation of a BI reporting project directly against an OLTP planning solver. The planning solver imports data concerning supply, demand, capacity, bill of materials, inventory and the like. It then uses linear programming to determine the correct product mix to produce at various factories worldwide. The article discusses the challenges faced and a working model in which real-time BI was achieved by providing data to a separate BI server in an innovative way resulting in decreased latency, reduced resource consumption and improved performance. We demonstrated an alternative approach to hosting data for the BI application separately by loading BI and solver databases at the same time, resulting in faster access to information.

Keywords: business intelligence, real-time business intelligence, OLTP, data warehouse, planning solver, replication tool

1. Introduction

In today's environment businesses need to make informed business decisions as things are changing in their environments. Companies typically create and collect data in operational data stores (order taking, accounting, procurement, and planning systems). This data is then posted to the enterprise data warehouse for various analytical needs. As the time frame which companies have to react to changes in the market place shrinks, there is a push to access information faster, often as soon as it is created. For decades we have known that reporting against operational data stores should be limited as data layout and indexing needs are different between on line transaction processing (OLTP) which writes, and business intelligence (BI) reporting which reads.

Businesses are facing challenges in today's environment. Volume and complexity of information in enterprises is increasing at the same time that business people are looking to take advantage of that information to gain a competitive edge. Companies face a tradeoff between timeliness of information and cost and operational impact. Generally, companies create data in On Line Transaction Processing (OLTP) systems. They then extract data from these OLTP systems and place it into the enterprise data warehouse (EDW) for various reporting and analytical needs. This framework allows separation of duties (creation vs. reporting) and also facilitates data cleansing efforts while ensuring that different aspects of the business are looking at the same information resulting in a single version of truth. There are, however, significant downsides with this traditional approach.

Generally, enterprise data warehouses are loaded via batch processes on a certain schedule, often every 4 hours or even longer between batches. Data definitions must be negotiated across various business processes and changes require significant planning and testing. The result is often a fairly rigid and bureaucratic process which may not serve all business functions. Additionally, because the environment is shared, the time required to implement changes is often significantly longer (measured in months or quarters) than business units can tolerate in a changing environment.

Alternatives, such as reporting directly against OLTP systems, have been explored. Often called real time analytics, these efforts have their own challenges. OLTP systems are designed for data capture; hence, they are structured, indexed and coded for fast writes. Conversely, reporting and analytic applications require fast reads. Indexes which are created to enhance report performance may adversely affect inserts and updates.

A detailed analysis of the business requirement is needed to determine what level of timeliness best serves the business purpose. This is called right time business intelligence (BI). In some cases, a delay of even a day will not adversely affect the quality of business decisions. In other cases, users may need to inspect and analyze information at various levels of aggregation before finalizing a plan and little or no delay can be tolerated, resulting in a need for real time BI.

In our planning application, business intelligence reports are designed to summarize excess and shortfall to demand across the planning horizon, as well as identify changes which have occurred since the previous planning cycle. There are a wealth of reports which look at various angles and levels of aggregation of information in order to identify patterns and trends which are occurring and affecting supply and demand. These reports are used to judge the quality of the production plan and allow planners to communicate roadblocks to sales and manufacturing organizations before customers are impacted.

Initially, our BI implementation was planned to be on a separate, replicated database. However, due to concerns about latency as well as resource constraints, the initial implementation had both the solver and the BI reporting running on the same server. While this has been shown to work and is in production now, several shortcomings (index would enhance BI performance but negatively impacts the solver, processes blocking each other, tight coupling of OLTP and BI) have caused us to step back and reconsider. Experimentation has shown that by loading both a BI database and a solver database at the same time does not take longer than simply loading the solver database. Updates can be replicated experiencing only 20 or 30 seconds of latency.

By hosting the BI implementation separately, overall latency was actually reduced because reports which were taking more than 10 minutes now take seconds by taking advantage of better indexing and precooked tables. IO and CPU consumption for the solver in no way impact reporting performance and vice versa. Locks or other server issues can be root caused to one application only.

Hosting BI applications inside Operational Data Stores is possible, but the benefits of hosting it separately have been shown to far outweigh the benefits of combining them. In fact, the largest perceived benefit, real time BI, is actually better achieved by separating the two functions.

2. Literature Research

There is strong evidence of the importance of business intelligence [17]. Many business as well academic publications describe different ways companies are using and benefiting from business intelligence [2 and 17]. Data warehousing and data management have been identified as one of the six physical capability clusters of IT-infrastructure services required for strategic agility [19]. Data warehousing and business intelligence are closely connected to each other. Significant research has been done on different aspects of data warehousing [1, 6, 8, 11, 15 and 20] and business intelligence [5, 7 and 24] over the last one decade. Previous work on business intelligence has focused on design issues [4, 9, 10, and 25], BI tools [16, 22, 23 and 24], BI data maintenance and collection strategies [12], and implementation issues and best practices [21].

Wu et al. [24] propose a service-oriented architecture for business intelligence that makes a seamless integration of technologies into a coherent business intelligence environment. They suggest that this kind of architecture enables a simplified data delivery and low-latency analytics. They compared our service oriented approach with traditional business oriented architectures, and presented the advantages of the service oriented paradigm. They assert that the proposed approach is the best way to reduce the total development and maintenance cost, and to minimize the risk and impact across an entire enterprise when introducing business intelligence solutions [24]. The Oracle [10] white paper addresses the business reasons to move to real-time data warehousing and describes some of the common data integration approaches, with an emphasis on using real-time CDC capabilities [10]. Sandu [13] and Davis [3] provide an overview of operational and real-time BI and how they optimize the decision making process by reducing to eliminating latency. Given real-time BI is costly, they suggest that companies do not always need to reduce latency to zero and do not always need to take and implement decisions in real time. They argue that companies should define optimum frame time, the right-time for any decision process, an interval that should reflect the business needs and that should offer the best risks-costs ratio [13].

Watson et al. [18] assert that "to be successful with real-time BI, organizations must overcome both organizational and technical challenges." They suggest that on the technical side, new hardware and software must be acquired and implemented, processes and procedures for supporting and managing real-time data feeds from source systems must be established. They also state that the purpose of real-time BI is to increase revenues and decrease costs. Companies that successfully implement real-time BI can dramatically improve their profitability [18]. Ramakrishnan et al. [12] examine how external pressures influence the relationship between an organization's business intelligence (BI) data collection strategy and the purpose for which BI is implemented. They provide managers with a mental model on which to base decisions about the data required to accomplish their goals for BI [12]. Steiger [14] suggests that BI techniques can be applied to knowledge creation as an enabling technology. The author proposes a business intelligence design theory for DSS as knowledge creation, and that indicates how BI can be focused internally on the decision maker to discover and enhance his/her mental model and improve the quality of decisions [14]. The author also suggests that BI is an appropriate enabling technology for knowledge creation.

So, most of the above research work focused on how BI could be used for different purposes, best practices, and what benefits businesses could achieve by effectively using BI. In this article, we talk about the back-end side of BI tools. We present a novel approach of hosting BI applications and solver database separately to achieve maximum benefits in terms of efficiency, resource usage and performance. The article discusses how real-time BI could be achieved by providing data to a BI server in an innovative way while decreasing latency and without impacting performance and resources consumption. We propose hosting data to BI application separately by loading BI and solver databases separately and at the same time.

3. Using Replication Technology for Data Movement

Our initial approach of using the OLTP database to support BI functionality has been available in production for nine months. It is fully functional and users access the data to support their business processes real time. Challenges have been encountered with processes blocking each other, causing response time concerns on both the OLTP solver aspect of the application as well as BI reports. Additionally, some reports run slower than they need to because indices which could be used to speed up data retrieval would have the opposite effect on OLTP and slow those processes down. As the application matures and grows, solutions have been sought.

We initially considered using triggers to capture data changes and write those from the transactional database to another server for use in reporting. This approach is not ideal in that it would require custom code within the OLTP application. Embedding business rules in complex SQL (structured query language) is never a good choice in production applications. Also, performance implications of triggers are widely understood. The more ideas were discussed, the more it became clear that a method of data transfer which is integral to the database engine itself would provide speed and flexibility needed while minimizing customization.

Based on our analysis and research, replication rose as the clear choice to move data seamlessly. It avoids the issue of embedding business logic in SQL and entirely sidesteps the custom code concerns as it is made up of commercially available components. Replication technology is used to move data from server to server in a transactionally consistent state from one instance to another instance. Replication can be transmitted on a schedule or in real time to a single instance or to multiple instances using either pull or push method. Our case study involves real time replication which moves data using push method to a single instance. Replication has a publisher, distributor and a subscriber. The publisher runs on the transactional db instance. The Distributor and distributor db can be on its own server, or it can reside on either the publisher or subscriber if resources allow. The Subscriber runs on the BI server.

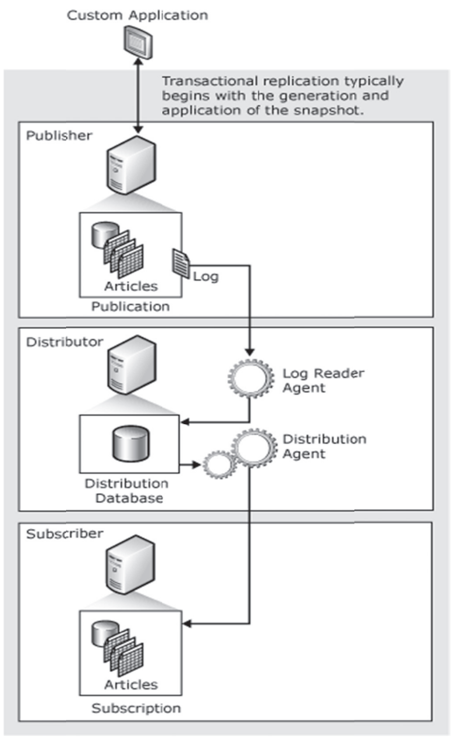
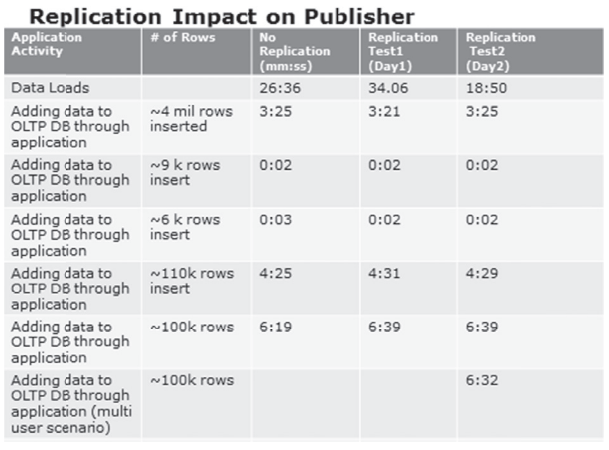


Figure 1. Transactional replication.

The Distributor uses the transaction log to propagate transactions to the Subscriber. In our implementation, as data is transmitted to the Subscriber, all data changes are processed in the order they were made on the publisher. This ensures data consistency on both transactional (OLTP) and BI instances. Our case study transferred/replicated only Table articles. We made schema changes on the transaction database and recorded/observed that the changes were immediately replicated on to the BI server. Any data changes that were made on the transactional database (through User interface) were picked up by the BI server in less than ten seconds. We were concerned about the transaction log size. Without replication, a transaction log is almost always written to, and rarely read from. When replication is configured, the transaction log file will not be truncated until all the transactions are replicated. We wanted to study this, and ensure that the log file growth will not cause bottle neck to the transactional OLTP DB. We monitored the latency that it takes the Log Reader to move transactions from OLTP transaction log to the distribution database. We monitored the latency it takes the Distributor Agent to move transactions from the distribution database to the Subscriber database. The total of these two figures is the amount of time it takes a transaction to get from the publication database to the subscriber database. The counters for these two processes are the DB Server Replication Log Reader: Delivery Latency counter, and the DB Server Replication Distributor: Delivery Latency counter. We concluded that if significant increase in the latency for either of these processes occurred, then this should be a signal to find out what new or different action has happened to cause the increased latency. We also used SQL queries to monitor log file growth.

We also studied the I/O performance of the server. Transactional replication can cause I/O performance issues on databases that experience large numbers of transactions. To address this we moved the transaction log of databases involved in transactional replication on its own dedicated RAID 1 or RAID 10 disk array. This reduced the risk of reading from OLTP transaction log file.

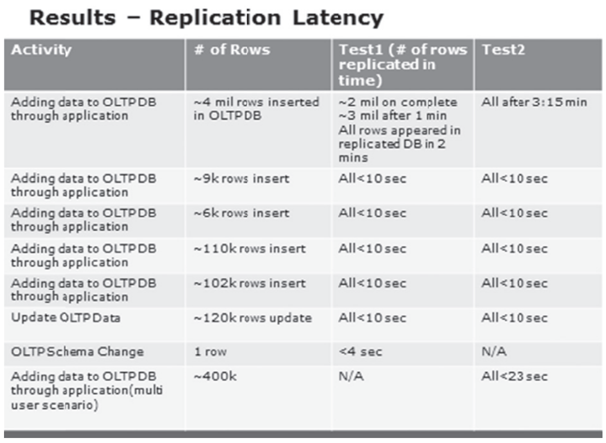
We did several tests for transactional database performance to ensure that replication did not add overhead to the transactional database server. We monitored the health of the application and the database for two months during the peak usage times. We did not observe any significant performance issues at the application level or at the database server level.



Replication Impact on Publisher

Application Activity	# of Rows	No Replication (mm:ss)	Replication Test1 (Day1)	Replication Test2 (Day2)
Data Loads		26:36	34.06	18:50
Adding data to OLTP DB through application	~4 mil rows inserted	3:25	3:21	3:25
Adding data to OLTP DB through application	~9 k rows insert	0:02	0:02	0:02
Adding data to OLTP DB through application	~6 k rows insert	0:03	0:02	0:02
Adding data to OLTP DB through application	~110k rows insert	4:25	4:31	4:29
Adding data to OLTP DB through application	~100k rows	6:19	6:39	6:39
Adding data to OLTP DB through application (multi user scenario)	~100k rows			6:32

Figure 2. Replicaton impact on publisher.



Results – Replication Latency

Activity	# of Rows	Test1 (# of rows replicated in time)	Test2
Adding data to OLTPDB through application	~4 mil rows inserted in OLTPDB	~2 mil on complete ~3 mil after 1 min All rows appeared in replicated DB in 2 mins	All after 3:15 min
Adding data to OLTPDB through application	~9k rows insert	All<10 sec	All<10 sec
Adding data to OLTPDB through application	~6k rows insert	All<10 sec	All<10 sec
Adding data to OLTPDB through application	~110k rows insert	All<10 sec	All<10 sec
Adding data to OLTPDB through application	~102k rows insert	All<10 sec	All<10 sec
Update OLTP Data	~120k rows update	All<10 sec	All<10 sec
OLTP Schema Change	1 row	<4 sec	N/A
Adding data to OLTPDB through application(multi user scenario)	~400k	N/A	All<23 sec

Figure 3. Results – replication latency.

4. Business Context

Our usage model involves factory optimization and loading on a varied product mix. Many products are capacity constrained. Adding to the complexity our bill of materials (BOMs) are many and varied. One assembly product can make literally thousands of different test items and conversely one test item can be made from many different assembly products. Due to these complex business scenarios there is a significant level of difficulty involved in trying to analyze the factory loading levels to ensure optimal output. On the forefront of these challenges is the fact that traditional business measures, such as available to promise, require significant resources to calculate in real time. Let us take available to promise as an example, and explore the challenges faced.

Available to promise (ATP) is defined by APICS (APICS Dictionary, 13th edition) as "the uncommitted portion of a company's inventory and planned production maintained in the master schedule to support customer order processing." The Dictionary defines order promising as "the process of making a delivery commitment." Generally, this value consists of inventory on hand plus the Master Production Schedule (MPS) minus the sum of customer order prior to the next MPS for the near term. However, once a certain time threshold is passed, the forecast of customer orders is substituted for actual orders. Because forecasts are only an estimate, smoothing must be applied to avoid wide variations in signals given to the production floor. Further, different products have different complexities in terms of their bill of materials, yields, throughput times et al. A variety of different subassemblies can be manufactured into the same finished good depending on availability of materials, machinery and manpower.

So, the calculation for available to promise must take information at the lowest level (item/day) and calculate through all of the various combinations and permutations of bills of materials to determine available supply as well projected demand and then aggregate these values across product families and weeks/months/quarters to create information which business people can use to plan production facilities and personnel, as well as materials requirements all in the interest of excellent supplier and customer relations.

From this one example of one metric it becomes clear that a variety of stakeholders will need to access this information at different levels of aggregation, using different filters, for a whole host of different business activities. The goal of Business Intelligence is to facilitate quality decisions in the least time possible. In some cases planners are creating production schedules and need to use reports to determine the quality of the schedules immediately before making them plan of record (POR). In other cases business unit managers need to examine product schedules to ensure their products are fully supported and manage tradeoffs between their item groups. Marketing representatives need to understand what they can and cannot promise their clients, and factory managers need visibility into which products can be swapped for which others at various points in the process.

All of this means that data needs to be stored at the lowest possible level. Hierarchies must be created and maintained that allow users to traverse the data without risk of neither double counting nor any other mistake in aggregation which could give a wrong calculation result. Initially these requirements were taken into account and drove a decision in which the data must be real time and reported against at the same level at which it is created. However, after this approach has been implemented, different observations have driven a different conclusion.

5. Performance Advantages of Separate OLTP vs. Reporting Databases

As previously discussed, performance of both OLTP and BI reporting were compromised to some extent by the existence of the other. Initially users felt the performance hits were justified by the immediate access to information created by the application. However, after seeing how quickly data could be replicated to a separate server, we realized that users could have access to information even faster via replication than they could by running reports against the OLTP system.

To understand why this would be, first we must consider the nature of the database itself. The database size is roughly 120 GB. There are 110 tables, the largest of which holds over 32 million rows of data, and the smallest of which holds half a dozen rows of data. All information is stored at the very lowest level of detail, which is individual line items and day. The breakdown is provided in Table 1:

Number of Tables	Number of rows
7	10M – 35M
14	1M – 10M
16	100K – 1M
9	20K – 1K
64	<1K

Table 1. Database tables involved.

The very large tables are accessed consistently for reporting. Often these reports are run at product family and month level, or even at division and quarter. In order to perform dynamic aggregation quickly, we require a significant number of indices, but those indices impede OLTP processes which are trying to access those same tables based on entirely different criteria. It is observed that the large majority of these rows are created before OLTP processes attempt to solve for optimal factory schedules. Therefore, various scenarios have been analyzed.

The fact is that the OLTP system uses data loads to acquire raw data from various other systems prior to solving for optimal factory schedules. This acquired data includes supply, demand, capacity, priority, bill of materials and more. This information can be loaded in a parallel fashion to both the OLTP database and the BI database, eliminating any latency there. Subsequent adjustments made by users are not stored in these huge core tables, but rather in smaller adjustment tables. Replicating these changes as they are made represents a relatively small proportion of data and can be done extremely quickly, and can even continue while solves are run on the OLTP system. Once the OLTP application is triggered and a "solve" is run, rows are stored back into output tables. This information can also be replicated. While the quantity of output rows is large, it does not represent a stumbling block to successful BI reporting. Time spent replicating the solver output to a separate server is more than made up for by the speed of accessing the data which is achieved using indices specifically designed for reads and aggregation.

Other advantages include using the database engine to "pre-aggregate" information which will be needed in reports. Some industry standard calculations can be done either as part of the replication process or in a staging area prior to loading to the core database. This technique allows the database engine to make large calculations once and then the result is accessed many times in different forms by different users. Changes made in the OLTP system (for example if the user solves again and overrides their previous solve) are replicated, any aggregations are made once again and stored in the core area.

Experimentation clearly demonstrates the advantages of aggregation prior to report execution. If we refer back to the previous example of available to promise, we see that we need to calculate forward and backward across the horizon at item level. Measures include average of forward demand, sum of previous demand as well as sum of manufacturing outs from the start of the horizon to present. We created a job to calculate these three values and write the results to a database table. We have shown that this activity takes 10 to 15 seconds depending on the volume of data for that particular version. The report makes calculations at the level of end item and week, as is typical in Master Production Scheduling. It then aggregates this information to product family and month or quarter in order to present out to senior management. When the report is forced to use data in base tables and first aggregate across the horizon, forward and backward, at item level and then traverse up the product and calendar hierarchies it takes upwards of 10 minutes to execute. However, by aggregating the forward and backward horizon calculations ahead of time (taking 10 or 15 seconds) and then using those numbers as a basis to aggregate up the product and calendar, report execution times drops to less than 2 minutes. Multiple scenarios have been tested and demonstrated 80% performance improvement.

6. Conclusion

In this article we discussed the benefits of right time BI and the fact that it is often faster than real time BI for a variety of practical reasons. In our experience we were faced with a need to perform analytics on a dataset which included factory schedules, before finalizing those schedules. And so, initially, it seemed obvious that real time BI was required. However, after implementation, we discovered that performance was hindered by the OLTP aspect of the system, as well as restrictions on indices which enhance performance. In the end we were able to demonstrate that by distributing the data the user can actually be presented with the complete analytical report faster using the data which has been replicated to another server than running the same report directly against the OLTP database.

7. Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments on this paper.

References

[1] S. BROBST, M. MCINTIRE, E. RADO, Agile Data Warehousing with Integrated Sandboxing. *Business Intelligence Journal*, **13**(1) (2008).

[2] T. H. DAVENPORT, Business Intelligence and Organizational Decisions. *International Journal of Business Intelligence Research (IJBIR)*, **1**(1) (January-March 2010), 1–12.

[3] J. R. DAVIS, Right-Time Business Intelligence: Optimizing the Business Decision Cycle. *Sybase, Inc*, (2006). Retrieved February 4, 2012 from: `http://www.computerworld.com/pdfs/sybase_free_realtime_wp.pdf`

[4] M. EMERSON, Embedding BI into Your Software Solution – Best Practices. (2011). A White Paper by Birst^{TM}. Retrieved on February 4, 2012 from: `http://www.birst.com/pdf/birst_wp_embedding_bi.pdf`

[5] Z. JOURDAN, R. K. RAINER, T. E. MARSHALL, Business Intelligence: An Analysis of the Literature. *Information Systems Management*, 25 (2008), 121–131.

[6] K. LAM, V. C. S. LEE, On Consistent Reading of Entire Databases. *IEEE Transactions on Knowledge and Data Engineering*, **18**(4) (2006).

[7] A. LONNQVIST, V. PIRTTIMAKI, The Measurement of Business Intelligence. *Information Systems Management*, **23**(1) (Winter 2006) p. 32.

[8] N. RAHMAN, D. RUTZ, S. AKHTER, Agile Development in Data Warehousing. *International Journal of Business Intelligence Research (IJBIR)*, **2**(3) (2011), pp. 64–77.

[9] N. RAHMAN, J. MARZ, S. AKHTER, An ETL Metadata Model for Data Warehousing. *Journal of Computing and Information Technology – CIT*, **20**(2) (2012), pp. 95–111, doi:10.2498/cit.1002046.

[10] ORACLE CORPORATION, Real-Time Data Integration for Data Warehousing and Operational Business Intelligence. *An Oracle White Paper*, (2010). Updated August 2010.

[11] F. PAYTON, R. HANDFIELD, Strategies for Data Warehousing. *MIT Sloan Management Review*, (2004).

[12] T. RAMAKRISHNAN, M. C. JONES, A. SIDOROVA, Factors influencing business intelligence (BI) data collection strategies: An empirical investigation. *Decision Support Systems*, 52 (2012), 486–496.

[13] D. I. SANDU, Operational and Real-time Business Intelligence. *Revista Informatica Economică*, **3**(47) (2008), pp. 33–36.

[14] D. M. STEIGER, Decision Support as Knowledge Creation: A Business Intelligence Design Theory. *International Journal of Business Intelligence Research (IJBIR)*, **1**(1) (January-March 2010), pp. 29–47.

[15] V. C. STOREY, R. C. GOLDSTEIN, Knowledge-Based Approaches to Database Design. *MIS Quarterly*, **17**(1) (1993), pp. 25–46.

[16] D. VESSET, Competitive Analysis: Worldwide Business Intelligence Tools 2010. *IDC Analyze the Future*, (2011). Retrieved on February 4, 2012 from: `http://www.sas.com/news/analysts/103115_0611.pdf`

[17] H. J. WATSON, Tutorial: Business Intelligence – Past, Present, and Future. *Communications of the Association for Information Systems*, Vol. 25, Article 39 (2009).

[18] H. J. WATSON, B. H. WIXOM, J. A. HOFFER, R. ANDERSON-LEHMAN, A. M. REYNOLDS, Real-Time Business Intelligence: Best Practices at Continental Airlines. *Information Systems Management*, Volume 23, Issue 1, (2006), pp. 7–18, DOI: 10.1201/1078.10580530/45769.23.1.20061201/91768.2

[19] W. WEILL, M. SUBRAMANI, M. BROADBENT, Building IT Infrastructure for Strategic Agility. *MIT Sloan Management Review*, (2002).

[20] J. WIDOM, Research Problems in Data Warehousing. In *Proceedings of the 4th Int'l Conference on Information and Knowledge Management (CIKM)*, (1995).

[21] B. WIXOM, H. WATSON, The Bi-Based Organization. *International Journal of Business Intelligence Research (IJBIR)*, **1**(1) (January-March 2010), pp. 13–28.

[22] C. WHITE, Now is the Right-Time for Real-Time BI. *DM Review*, September 2004.

[23] C. WHITE, The Next Generation of Business Intelligence: Operational BI. *Information Management*, (2006). Retrieved 02/04/2012 from: `http://www.information-management.com/issues/20050501/1026064-1.html`

[24] L. WU, G. BARASH, C. BARTOLINI, A Service-oriented Architecture for Business Intelligence. In *Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications*, (2007). SOCA '07, pp. 279–285, Newport Beach, CA.

[25] I. YERMISH, V. MIORI, J. YI, R. MALHOTRA, R. KLIMBERG, Business Plus Intelligence Plus Technology Equals Business Intelligence. *International Journal of Business Intelligence Research (IJBIR)*, **1**(1) (January-March 2010), pp. 48–63.

Received: July, 2012
Accepted: November, 2012

Contact addresses:

Dale Rutz
Intel Corporation
Supply Planning Decision Technologies (SPDT), SPO, CPLG, TMG
Mail Stop: RNB 4-80
2200 Mission College Blvd
Santa Clara, CA 95054-1549, USA
e-mail: `dale.m.rutz@intel.com`

Tara Nelakanti
Supply Planning Decision Technologies (SPDT), SPO, CPLG, TMG
Intel Corporation
Mail Stop: RNB 4-80
2200 Mission College Blvd
Santa Clara, CA 95054-1549, USA
e-mail: `tara.k.nelakanti@intel.com`

Nayem Rahman
IT Business Intelligence (BI)
Intel Corporation
Mail Stop: AL3-85
5200 NE Elam Young Pkwy
Hillsboro, OR 97124-6497, USA
e-mail: `nayem.rahman@intel.com`

DALE RUTZ is a Software Engineer in the Technology and Manufacturing group working in Supply Planning Decision Technologies. She has been developing software at Intel Corporation for 20 years. Her work has involved all facets of software development including ETL, integration and presentation layer using technologies such as SQL, JAVA, UNIX, C, XML and various proprietary protocols. Ms. Rutz holds an MBA from Santa Clara University and a BS from San Jose State University in MIS and Math. She completed the APICS CSCP (certified supply chain professional) in December 2008. Her focus is on harnessing the power of information.

TARA NELAKANTI is a Software Engineer in the Technology and Manufacturing group working in Supply Planning Decision Technologies at Intel Corporation. She implemented and supported Master Production Scheduling solutions in Supply Planning Operations. She also implemented several key projects at Intel which involved technologies such as TCP/IP, ETL and Publish-Subscribe. She also worked on presentation layer using technologies such as SQL, C, and C-Sharp. Ms. Nelakanti holds a BS in Computer Science from University of Texas at Dallas and a BS in Electronics from Osmania University, India. Her focus is on data modeling and data architecture.

NAYEM RAHMAN is a Senior Application Developer in IT Business Intelligence (BI), Intel Corporation. He has implemented several large projects using data warehousing technology for Intel's mission critical enterprise DSS platforms and solutions. He holds an MBA in Management Information Systems (MIS), Project Management, and Marketing from Wright State University, Ohio, USA. He is a Teradata Certified Master. He is also an Oracle Certified Developer and DBA. His most recent publications appeared in the International Journal of Business Intelligence Research (IJBIR) and Journal of Computing and Information Technology (CIT). His principal research areas are active data warehousing, changed data capture and management in temporal data warehouses, change management and process improvement for data warehousing projects, decision support system, data mining for business analysts, and sustainability of information technology.