https://doi.org/10.48009/1_iis_2013_331-338
Issues in Information Systems
Volume 14, Issue 1, pp.331-338, 2013

SQL SERVER TABULAR MODEL: A STEP TOWARDS AGILE BI

Stevan Mrdalj, Eastern Michigan University, smrdalj@emich.edu

ABSTRACT

As data volumes continue to increase the organizations are under constant pressure to deliver Business Intelligence (BI) faster with more flexibility. That requires faster data warehouse performance to support rapid business analytics as well as a shorter time to respond to the changing business environment. In the past few years, in-memory column oriented databases made it possible to store very large volumes of data in the main memory providing high-speed processing and increased flexibility in providing ad hoc analytics. This paper presents the technological advances of the SQL Server Tabular Model that utilizes in-memory columnar storage for its Analysis Services. It also examines if the Tabular Model provides the so-called agile BI. For such an examination we establish the criteria for agile BI. We conclude that the Tabular Model exhibits many characteristics of an agile BI while we understand that it is an evolving technology and therefore we regard it as a step towards agile BI.

Keywords: Tabular Model, In-Memory Analytics, Columnar Databases, Agile BI and SQL Server

INTRODUCTION

Conventionally delivering Business Intelligence (BI) is quite a lengthy process. It starts by extracting data from operational systems, transforming and loading them into the data warehouse. This process usually concludes by creating an Online Analytical Processing (OLAP) cube, in order to deliver a fast response by pre-aggregating data needed by frequently used queries. Since this process had to be repeated periodically, the data in OLAP cubes was frequently a day or week (or longer) out of date. Such a specially designed OLAP cube could only answer the questions that it was built to answer because it relied on materialized views. Every new analytical request, not originally anticipated, required a lengthy process of inserting new data and re-computation of the cube by a handful of BI professionals [1, 13]. Basically, such a lengthy process cannot keep pace with today's BI needs for fast and agile analytics. Simultaneously, the rapidly increasing data volumes, rising needs for data integration, and more than ever diverse sources create additional barriers for traditional BI solutions [7]. Knabke and Olbrich [3] summarize "that current BI systems behave rigid and inflexible in terms of readjustment to fulfill changed or new requirements".

Advances in speed, cost and sophistication of memory technology combined with 64-bit multicore processors provided a possibility that BI data can be managed entirely in the main memory [10, 11, 12]. Based on today's technology it is theoretically possible to store a database as large as 16 exabytes in the main memory a with seven time higher throughput and almost instantaneous response [1]. There are several in-memory databases available on the market such as TimesTen (Oracle), solidDB (IBM), SQL Server 2012 (Microsoft), and HANA (SAP) [7].

Back in the seventies the first attempts were made to store data column-wise [6]. Recently, the columnar storage became essential to in-memory databases since it greatly improves storage capacity using data compression and query performance of the typical BI queries by two orders of magnitude [6, 10, 14]. Whereas row-oriented data storage was designed and optimized for transactional processing, the column-oriented data storage is especially well suited for data warehouses and BI where data is largely analyzed by attributes (columns) and analysis obtained by aggregations and operations such as sum, min, max, average, etc. [8]. Typically, BI queries tend to read only a few attributes from many records making columnar storage preferable, since irrelevant columns do not need to be accessed which greatly increases performance [2, 15].

The SQL Server is a general-purpose database that provides in-memory columnar storage for its analysis server called the Tabular Model. In this paper we provide a detailed description of the technology used in the Tabular Model and we examine if the Tabular Model delivers a platform for agile BI.

With assumptions that agile BI focuses on flexible creation and change of BI systems in terms of fulfilling changed or new requirements [3], the aim of this paper is to answer the following (among others) research questions:

- What are the novel technological concepts used in the Tabular Model?

- What are the potentials for the Tabular Model to enable agile BI?
- What are the current limitations for the Tabular Model?

The next section presents a background of the SQL Server BI semantic model followed by the section which details the underlining technological concepts answering the first research question. The following section establishes criteria for agile BI which is used to answer the remaining two research questions. In answering our research questions we use rational deductive arguments.

SQL Server BI Semantic Model

Back in 1998, Microsoft was first to include an OLAP server in its DBMS [9]. At that time, Microsoft developed OLE DB for OLAP and Multidimensional Expressions (MDX) which later became the industry standard for accessing OLAP servers [9]. The next major advancement was with the release of SQL Server 2005 when the term Unified Dimensional Model (UDM) became introduced. It was the beginning of the Multidimensional Model as a foundation for Analysis Services to host end-user based tools capable of recognizing dimensions and measures such as Microsoft Excel PivotTables and PivotCharts. In 2010 Microsoft introduced the PowerPivot for Excel add-in to deliver Tabular based self-service BI. It utilizes in-memory and column-based data storage to load, explore, relate and derive data [9]. Simultaneously, a new expression language named Data Analysis Expressions (DAX) was developed to define the calculated columns and measures. The Multidimensional and Tabular technologies were combined in the single BI Semantic Model (BISM) in SQL Server 2012.

The BI Semantic Model integrates multidimensional and tabular data models under one roof (Figure 1). It can integrate data from a variety of traditional data sources (databases, LOB applications) or non-traditional data sources (OData feeds, flat files, cloud services). This model also can deliver BI results into a variety of end-user tools.

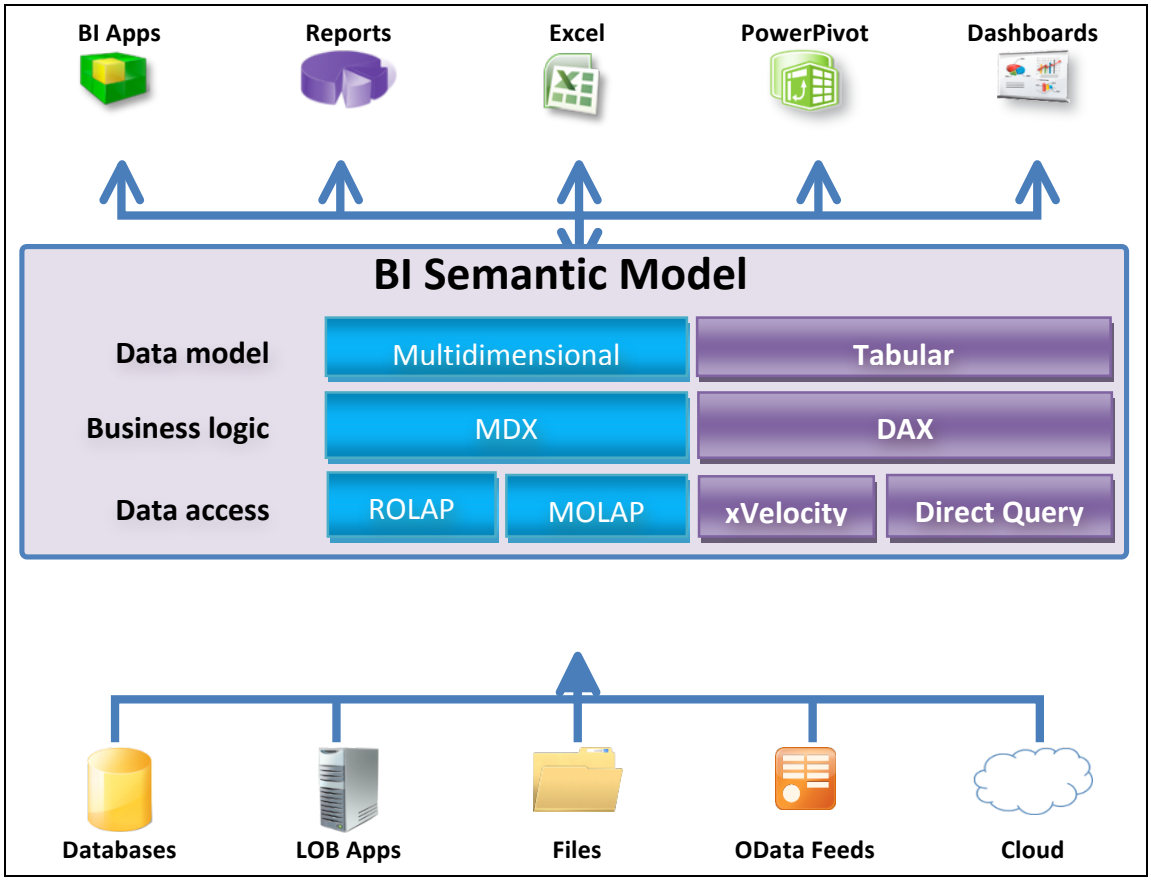

Figure 1. BI Semantic Model Architecture (Adapted from Microsoft Corporation)

The BI Semantic Model itself can be viewed in three layers (data model, business logic and queries, and data access). Figure 1 indicates corresponding technologies used in each layer for both multidimensional and tabular models. In essence, it replaces the UDM, but continues to support the existing multidimensional model. Since the scope of this paper is to examine if and to which extent the Tabular model contributes towards achieving agile BI, the Tabular model will be detailed in the next section.

Although BISM combines both models, at this time the SQL Server Data Tools (using Visual Studio 2010 shell) comes with two project templates: Analysis Services Multidimensional and Data Mining Project, and Analysis Services Tabular Project. Consequently, those projects cannot be deployed on the same server. Therefore, the SQL Server 2012 lets you create an instance of the Analysis Services using either the Multidimensional Mode or the Tabular Mode. Both types of Analysis Services can reside on the same machine, but it is not recommended due to different hardware requirements discussed in the next section.

Tabular Model

The Tabular Model uses the in-memory storage mode which is powered by xVelocity and DirectQuery. DirectQuery is a pass-through mode to a single SQL Server relational data source and it will not be discussed in this paper. xVelocity stores all of the data in the main memory using columnar storage and state-of-the-art compression and querying techniques [9] as shown in Figure 2. The first step is to convert rows of every table into columns. Each column is then encoded and compressed independently [6]. All values from a single column are stored in a single block of memory. Since data from a single column tend to be more homogeneous, such data are more compressible. This is especially true for large BI tables.

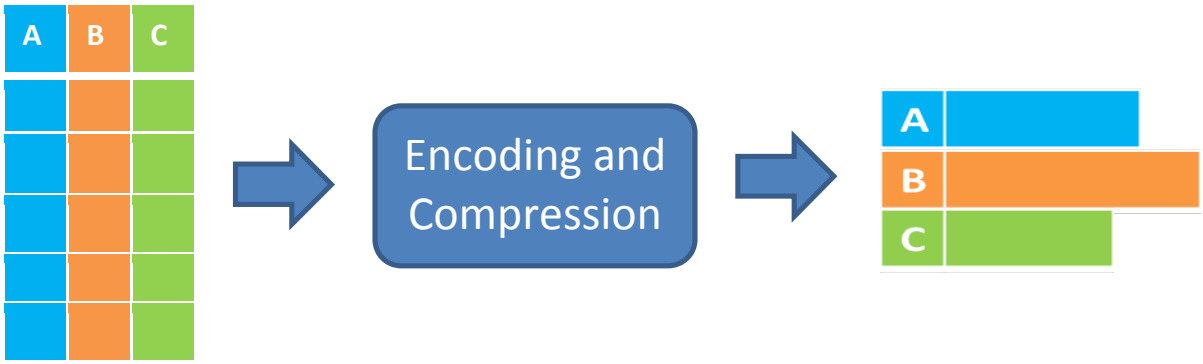

Figure 2. xVelocity Storage Method

xVelocity uses two types of in-memory compression: Run Length Encoding and Dictionary Encoding. The run length encoding replaces segments or runs of consecutive repeated data with a single data and a length of run as shown in Figure 3. This encoding is applied only when the size of compressed data is smaller than original. The compression ration depends on how homogeneous the data is. It is also very dependent on data order and xVelocity automatically selects the best sorting.

Name	State	City
Susan	MI	Detroit
Steve	MI	Detroit
Mary	MI	Lansing
Bob	MI	Lansing
John	OH	Toledo
Stacy	OH	Toledo
Ann	OH	Toledo

Name
Susan
Steve
Mary
Bob
John
Stacy
Ann

State	Start	Count
MI	1	4
OH	6	3

City	Start	Count
Detroit	1	2
Lansing	3	2
Toledo	5	3

Figure 3. Run Length Encoding

The dictionary encoding creates a dictionary of all distinct values in one column with unique identifiers. This allows that the real storage for the column will be a bitmap index that references the dictionary. Such a bitmap index (identifiers) is further used in the run length encoding to produce xVelocity in-memory storage as displayed in Figure 4. The dictionary encoding makes the tables data type independent and it is always used. The resulting dictionaries are stored separately. For example, when querying a customer table for those with a certain income level, instead of retrieving complete rows it will only retrieve attributes/columns selected by the query. In order to recreate a row, the position of a value in the column indicates to which row it belongs [11].

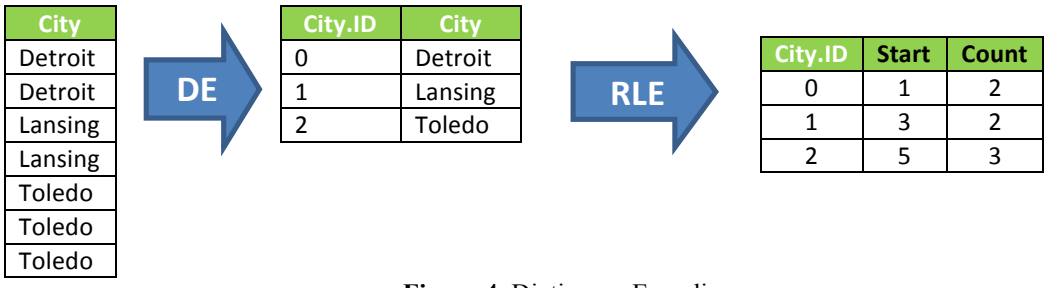

Figure 4. Dictionary Encoding

xVelocity provides enormously fast query performance without the need for indexing or aggregation since it uses brute-force scanning of data in the main memory [9]. It utilizes all of the processor cores and available memory. Therefore, xVelocity requires a very fast multi core CPU and a very fast memory. Number and performance of the disks are not important at all. Consequently, the traditional SQL Engine appliances are not suitable for the Tabular model which requires a new kind of In-Memory appliance.

The business logic and querying of the Tabular Model is achieved using a new scripting language named Data Analysis Expressions (DAX). DAX is a combination of a new query language and a calculation language based on a subset of Excel functions and many new functions. It is quite "relational' but not traditional SQL. DAX is also optimized for multi-core processors. It is very suitable for detail-oriented reporting. Myers [9] lists the following numerous new DAX capabilities and functions:

- Define distinct count measures.
- Define multilevel hierarchies.
- Define and format measures.
- Define KPIs.
- Add descriptions to tables, columns, measures, and KPIs.
- Define a time table with a date column (which must be based on a Date column type) so that integer key values can be used in fact tables.
- Sort column values by another column in the same table.
- Define multiple relationship paths (direct or indirect) between two tables.
- Define perspectives to support subject-specific subsets of the entire model structure.
- Define binary columns (useful for storing images).
- Define reporting properties to optimize the presentation of data in reporting applications.

DAX is used to define calculated columns that extend existing tables and measures that are evaluated at query time. Together with xVelocity, DAX is a driving force behind the Tabular Model technological platform that enables agile BI.

Towards Agile BI

Today's BI systems face a world of agile businesses that require data beyond historical transactional sources such as unstructured mobile feeds, external data sources from remote servers, real time sensor feeds or triggers, website data

sources, cloud services, and many more. Simultaneously, BI systems need to support flexible exploration, integration, aggregation, and multidimensional analysis [3]. Another important requirement is to satisfy the growing need to bring BI closer to end-users. Having such requirements in mind, we extend a similar effort described in [3, 10, 15] to propose the following criteria for agility of BI systems:

- Ability to integrate new and unforeseen data sources,
- Ability to rapidly embrace change, and
- Ability to provide self-service BI.

In what follows, we limit our discussion to the architectural and technological aspects of the Tabular Model as a technology enabler for agile BI while we understand that process and project management plays a significant role in the agility of BI systems. Given the quite early stage of the Tabular Model and its adoption, instead of using an analytical methodology to verify the above criteria, we use rational deductive arguments.

Analysis Server Data Sourcing

As mentioned before, beside historical data, relevant data can be contained in many heterogeneous sources. The multidimensional OLAP model allows only for a relational database (preferably single) as it source, whereas the Tabular Model may acquire data from sources such as relational database, OLAP cubes, flat files, Excel files, and any application that supports Open Data Protocol [5]. The proposed BI architecture in Figure 5 features a heterogeneous landscape that combines both the traditional data warehouse using OLAP and Tabular systems (with the understanding that OLAP and Tabular would reside on a different Analysis Services servers). This is to illustrate the impressive data acquisition capabilities of the Tabular Model.

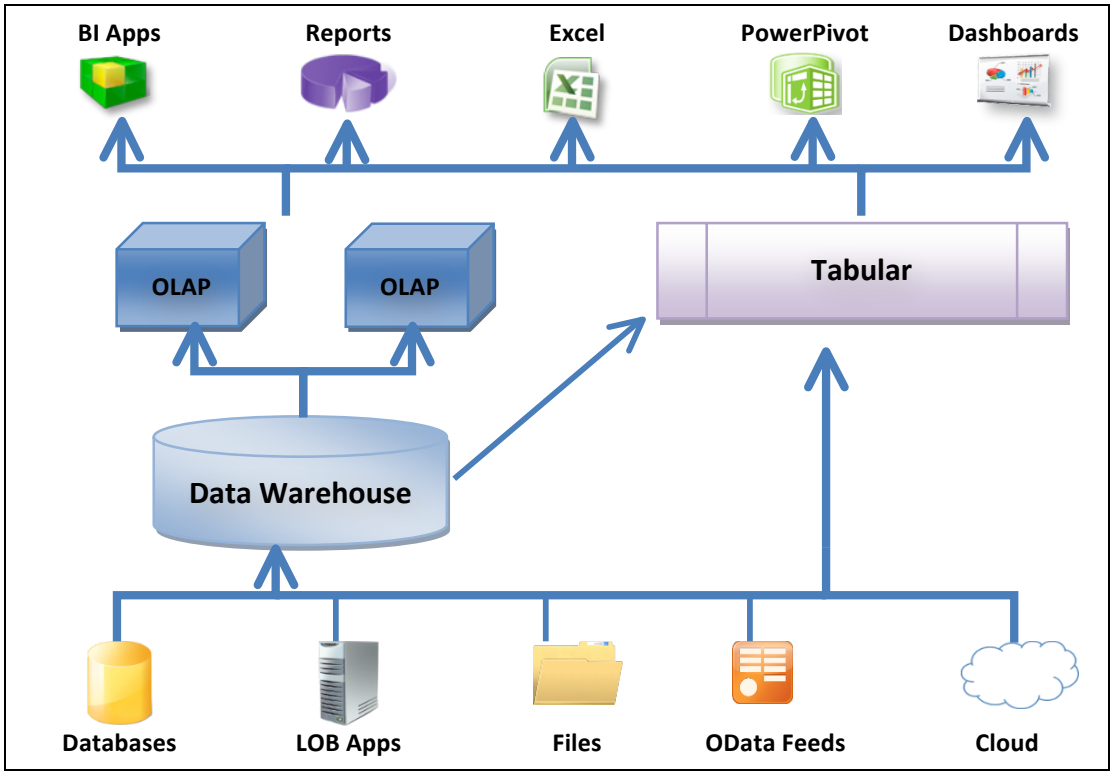

Figure 5. Hybrid BI Architecture

Frequently BI users simply need access to their specific operational data for reporting and analytics. The Tabular Model removes the need to first build complex multidimensional cubes from the data warehouse. Instead, it allows

direct access to almost any data source as indicated in Figure 5. This allows users to source data rapidly and to provide a much greater degree of ad hoc analytics [1].

Therefore, we conclude that the Tabular Model offers the ability to integrate new and unforeseen data sources needed for agile BI systems. In addition, some believe that the in-memory columnar database technology has the potential to integrate OLTP and OLAP databases and to revolutionize the future of BI [7, 11]. This includes a possible end to the ETL as we know it rapidly reducing data entry latency, data analytic latency, and consequently decision making latency [15].

Data carried directly into the Tabular Model still need to be cleaned, transformed and enriched in order to provide reliable analytics. As we will see in the next section, needed transformations can be calculated on the fly to offer consolidated data.

Embracing Change

The increasingly dynamic decision making process requires more and more solutions that enable decision makers to analyze spontaneous aspects of data [15]. Analysis pathways should not be limited by pre-calculated aggregations allowing various perspectives. In the Tabular Model there are no aggregations and the constant need to manage them. New rows can be added in the Tabular Model in real time. Consequently, the queries can contain data from computed and real rows since aggregations and views are not pre-computed. High performance allows all aggregations to be calculated on the fly. Therefore, the Tabular Model exhibits the following characteristics of an agile BI:

- Aggregations are omitted which enables iterative and interactive analytics.
- The pre-processing stage can be reduced and architectural complexity is eliminated.
- Possible analytic pats are not static permitting spontaneous analysis of key performance indicators.

In respect to embracing change, the Tabular Model is a big step forward with high expectations for agile BI. It makes no limitations on how many alternatives or hypothesis should be compared with significantly increased analysis frequency. It is important to indicate that such flexibility provided by the Tabular Model does not reduce response time and in many cases provides significantly better performance compared to OLAP systems.

Self-Service BI

From the BI modeler's perspective the BISM provides a full spectrum of possibilities ranging from Personal BI to Corporate BI as shown in Figure 6. On one end are business users using PowerPivot for Excel to create BI applications that meet their own specific needs. On the other end are corporate developers using SQL Server Data Tools to develop BI solutions for the entire enterprise [9].

Originally, the Tabular Model is developed to support personal BI using PowerPivot for Excel. The Tabular Model uses simple constructs such as tables, columns and relationships to define the BI schema. Business analyst can start with a small (personal) model implemented in Excel. That empowers the business analysts to develop ad hoc BI models and to analyze data by themselves. The personal BI applications can migrate into corporate BI applications since it is powered by a single model.

When the personal model exceeds the Excel capabilities (2GB compressed dataset size), the corporate BI developers can use the SQL Server Data Tools (formerly Business Intelligence Development Studio) to upgrade the BI model for organizational use. Beside the capability to deploy the in-memory analysis server, it provides the following two main features. The first is that the row-level security can be defined to provide data access restrictions. The second is the ability to partition large tables. This allows efficient management and partial or incremental data refresh. The organizational BI has no limit for the size of data set and it based on the available main memory.

One of the goals for BISM was to hide the underlining data storage mode and to provide the end-user with the ability to do what they really need to do, which is analyze data. Consequently, from the BI end-user's perspective the Tabular Model can be consumed as either a tabular or multidimensional model. Such flexibility is available since

the Tabular Model utilizes a multidimensional interface similar to OLAP [9]. Therefore, all existing client tools capable of interfacing with OLAP cubes would interface with the SQL Server Tabular Model.

We join Lachev [4] in the assessment that the Tabular Model allows having "a simple and efficient BI model that promotes rapid BI development … well suited for both personal and organizational BI projects". The Tabular Model with its in-memory storage technology has capabilities to ultimately provide business users with virtually instant self-service business intelligence.

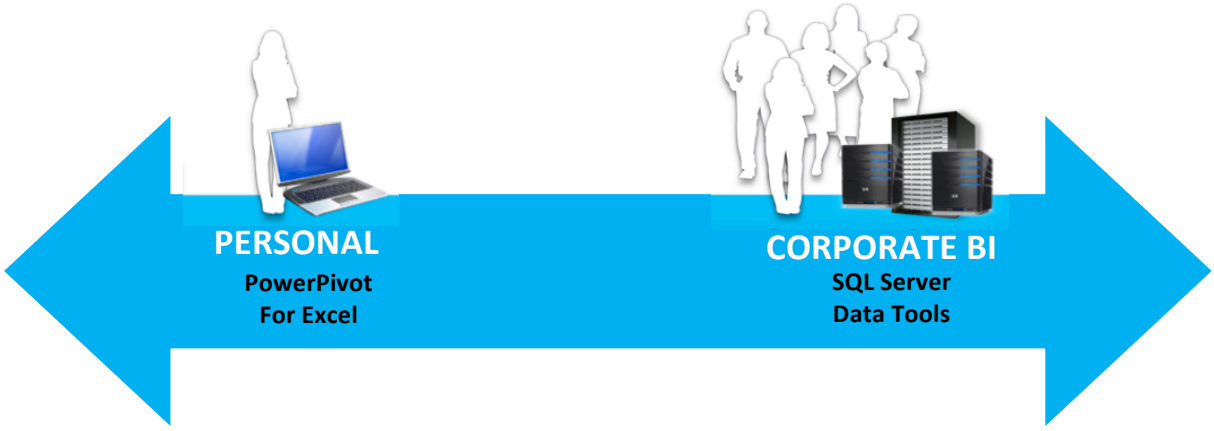

Figure 6. Range of BI Applications (based on [9])

Current Limitations

The Tabular Model is a fairly new technology that is in its first release. Inherently, it has several limitations compared to the multidimensional model that is a mature technology that allows work with advanced concepts. One such area is declarative relationships. The Tabular model only supports regular and role-playing relationships with the limitation that only one role-playing relationship can be active at the time [4]. It does not support the native parent-child hierarchy, many-to-many relationships, and dimension attribute relationships. There is no direct accommodation for slowly changing dimensions. There is no built-in time dimension. In terms of aggregation functions, the Tabular Model does not support powerful calculations such as semi-additive functions.

We still need to keep in mind that main memory is volatile which makes data recovery strategies critical for Tabular projects [3]. Those strategies are beyond the scope of this paper.

At the moment, in-memory databases are not designed with cloud computing in mind. Instead, they are seen as a complement to the cloud and their combination may make real-time BI not only possible but also affordable.

CONCLUSIONS

Business Intelligence is faced with constantly increasing data that need to be analyzed and with a growing BI user population. Confronted with such requirements, developing and managing agile BI systems became a major challenge for organizations. Improved analytical performance and development flexibility are identified as possible solutions.

New technological advances in multi-core processors and the availability of high capacity main memory made in-memory columnar databases a reality. Microsoft used such a technology to add the Tabular Model to its Business Intelligence Semantic Model. Thus, this paper presents the Tabular Model as a technology capable of facilitating agile BI. We also identify criteria for agile BI and use it to examine if the Tabular Model indeed enables agile BI.

Larson [5] states "There may be a day when a Tabular model is the right answer for any and all BI needs. It is also the case that even with the data compression available within a Tabular model, there are still limits to what can be effectively processed in memory". In our opinion the Tabular Model is big step forward towards agile BI and it will have a significant impact on future BI technology. In-memory columnar analytic servers are evolving systems which

can overcome current shortcomings. Therefore, we conclude that the SQL Server Tabular Model is certainly a step towards a true "in-memory analytics" where we would have data required for all operational transactions stored in a single in-memory database that would also handle all BI needs as well as provide virtually real time BI [1].

REFERENCES

1. Acker, Olaf, et al. (2011). In-memory analytics–strategies for real-time CRM. *Journal of Database Marketing & Customer Strategy Management, 18*(2), 129-136.
2. Dhindsa, P. B. S. K. (2012). A Comparative Study of Database Systems. *International Journal of Engineering and Innovative Technology* [online], *1*(6), 267-269. Available: ijeit.com/vol%201/Issue%206/IJEIT1412201206_50.pdf
3. Knabke, T., & Olbrich, S. (2011, December). Towards agile BI: applying in-memory technology to data warehouse architectures. In *Gesellschaft für Informatik eV (GI),* 101-113. Available: subs.emis.de/LNI/Proceedings/Proceedings193/P-193.pdf#page=110
4. Lachev, T (2012). *Applied Microsoft SQL Server 2012 Analysis Services*. Prologica Press.
5. Larson, B. (2012). *Delivering Business Intelligence with Microsoft SQL Server 2010*, 3ed. New York, NY. McGraw Hill.
6. Larson, P. Å., Hanson, E. N., & Price, S. L. (2012). Columnar storage in SQL Server 2012. *IEEE Data Engineering Bulletin*, *35*(1), 15-20.
7. Loos, Peter, et al. (2011). In-memory databases in business information systems. *Business & Information Systems Engineering, 3*(6), 389-395.
8. Matei, G. (2010). Column-Oriented Databases, an Alternative for Analytical Environment. *Database Systems Journal* [online], *1*(2), 3-16. Available : dbjournal.ro/archive/2/1_Gheorghe_Matei.pdf
9. Myers, P. (2012). Introducing the BI Semantic Model in Microsoft SQL Server 2012, *SQL Server Technical Article* [online], Microsoft. Available: http://msdn.microsoft.com/en-us/library/jj735264.aspx
10. Piller, G., & Hagedorn, J. (2011). Business Benefits And Application Capabilities Enabled By In-Memory Data Management. In *Gesellschaft für Informatik eV (GI),* 45-56. Available: subs.emis.de/LNI/Proceedings/Proceedings193/P-193.pdf#page=45
11. Plattner, H. (2009). A common database approach for OLTP and OLAP using an in-memory column database. *Proceedings of the 35th SIGMOD international conference on Management of data* [online]. Providence RI USA, 1-7. Available: http://dl.acm.org/citation.cfm?id=1559846
12. Russom, P. (2009). Next Generation Data Warehouse Platforms. *TDWI Best Practices Report, TDWI*, 5. Available: www.oracleimg.com/us/solutions/datawarehousing/040119.pdf
13. Saxena, G., Narula, R. & Mishra, M. (2013). New Dimension Value Introduction for In-Memory What-If Analysis. *preprint arXiv:1302.0351* [online]. Available: arxiv.org/abs/1302.0351
14. Winsemann, T. Valuation Factors for the Necessity of Data Persistence in Enterprise Data Warehouses on In-Memory Databases. Available: ceur-ws.org/Vol-731/09.pdf
15. Winter, R., Bischoff, S., & Wortmann, F. (2011). Revolution or Evolution? Reflections on In-Memory Appliances from an Enterprise Information Logistics Perspective. In *Gesellschaft für Informatik eV (GI)*. 23-34. Available: subs.emis.de/LNI/Proceedings/Proceedings193/P-193.pdf#page=23