# Business Intelligence (BI)

Business Intelligence (BI) is a terminology representing a collection of processes, tools and technologies helpful in achieving more profit. Business Intelligent System has the ability to measure performance by key performance indicators.

##### Evolution of BI

BI began with the early mainframe reports, called system outputs, decades ago. Printed out on stacks of paper, they were periodically distributed to managers. The managers were expected to pick their way through them to identify pertinent information they could use in their tactical and strategic decision

making. From there organizations proceeded to custom reports, which required skilled programmers and took weeks to produce. Early ad-hoc queries and reporting speeded up the process and made it possible for technically skilled managers to create their own queries and reports, but few managers had the time and skills to do that. The emergence of the data warehouse gave a big boost to BI by aggregating all the data in a single location, where it could be queried interactively without impacting production applications. Online query and reports tools with increasingly easy-to use graphical interfaces made BI accessible to more managers and enabled those managers to get critical information and answers more quickly. Data warehouses were followed by data marts specialized data stores that further accelerated the process of getting information to managers for the purposes of informed decision making. Then there were Online Analytical Processing (OLAP) and other multi-dimensional analytical tools, which allowed managers to dice and splice the data in a variety of ways and mine it for otherwise hidden insights. At that point, BI began merging with business analytics (BA).

Today BI and BA are delivered as applications that run on top of an infrastructure of databases, data management systems, Extract, Transform and Load (ETL) capabilities, and more. The BI infrastructure may include executive dashboards, scorecards, and other tools that make it easy for managers to find and understand the information and proactively use it in decision making. This decades-long evolution did not come without a price. During this time companies purchased and deployed a wide variety of BI-related products. They employed different deployment models, different user interfaces, and different management

interfaces as well as having different integration requirements. By now companies may have eight, ten, or more different BI products deployed. The cost of maintaining this proliferation of tools and technologies is already high and will get higher. Business Intelligence (BI) is the most comprehensive portfolio of technology and applications.

# **Tools for Implementing a Successful Business Intelligent**

Some of the tools necessary for implementing a successfully modern day business intelligent system include:

##### Data Warehousing

Basically a data warehouse is a single, complete, and consistent store of data obtained from various sources. It is a database that contains data that has been cleansed and transformed into an informational format**.** Data warehouses are tools for organized information systems. They are usually relational databases, in other words they can compare, match and relate similar attributes of data. A Data Warehouse is a repository of integrated information, available for queries and analysis. Data and information are extracted from heterogeneous sources as they are generated. This makes it much easier and more efficient to run queries over data that originally came from different sources.

Some of the basic characteristics for implementing a successful data warehouse include:

* A set of programs that extract data from an operational environment.
* A database that maintains data warehouse data.
* A defining characteristic of a data warehouse is its separation of support functionality.

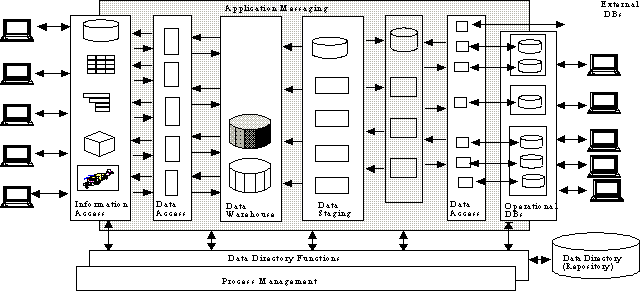
##### Framework of the Data Warehouse

One of the reasons that data warehousing has taken such a long time to develop is that it is actually a very comprehensive technology. In fact, data warehousing can be best represented as an enterprise-wide framework for managing informational data within the organization.(W. Ken et al 2010). In order to understand how all the components involved in a data warehousing strategy are related, it is essential to have a Data Warehouse Architecture.

***Data Warehouse Architecture***

A Data Warehouse Architecture is a way of representing the overall structure of data, communication, processing and presentation that exists for end-user computing within the enterprise. (Data Warehousing Technology a White Paper by Ken Institute, 2010).

The architecture is made up of a number of interconnected parts:



*Figure 2.0: Data Warehouse Architecture. Source: Data warehouse (www.datawarehousing.com),(accessed 2012 January 13)*

1. Operational Database / External Database Layer
2. Information Access Layer
3. Data Access Layer
4. Data Directory (Metadata) Layer
5. Process Management Layer
6. Application Messaging Layer
7. Data Warehouse Layer
8. Data Staging Layer
9. **Operational Database / External Database Layer:** Operational systems process data to support critical operational needs. In order to do that, operational databases have been historically created to provide an efficient processing structure for a relatively small number of well-defined business transactions. However, because of the limited focus of operational systems, the databases designed to support operational systems have difficulty accessing the data for other management or informational purposes. This difficulty in accessing operational data is amplified by the fact that many operational systems are often 10 to 15 years old. The age of some of these systems means that the data access technology available to obtain operational data is itself dated.
10. **Information Access Layer:** The Information Access layer of the Data Warehouse Architecture is the layer that the end-user deals with directly. In particular, it represents the tools that the end-user normally uses day to day, e.g., Excel, Lotus 1-2-3, Focus, Access, SAS, etc. This layer also includes the hardware and software involved in displaying and printing reports, spreadsheets, graphs and charts for analysis and presentation.
11. **Data Access Layer:** The Data Access Layer of the Data Warehouse Architecture is involved with allowing the Information Access Layer to talk to the Operational Layer. In the network world today, the common data language that has emerged is SQL. Originally, SQL was developed by IBM as a query language, but over the last twenty years has become the de facto standard for data interchange.
12. **Data Directory (Metadata) Layer:** In order to provide for universal data access, it is absolutely necessary to maintain some form of data directory or repository of meta-data information. Meta-data is the data about data within the enterprise. In order to have a fully functional warehouse, it is necessary to have a variety of meta-data available, data about the end-user views of data and data about the operational databases. Ideally, end-users should be able to access data from the data warehouse (or from the operational databases) without having to know where that data resides or the form in which it is stored.
13. **Process Management Layer:** The Process Management Layer is responsible for scheduling the various tasks that must be accomplished to build and maintain the data warehouse and data directory information. The Process Management Layer can be thought of as the scheduler or the high-level job control for the many processes (procedures) that must occur to keep the Data Warehouse up-to-date.
14. **Application Messaging Layer:** The Application Message Layer has to do with transporting information around the enterprise computing network. Application Messaging is also referred to as "middleware", but it can involve more than just networking protocols. Application Messaging can also be used to collect transactions or messages and deliver them to a certain location at a certain time.
15. **Data Warehouse (Physical) Layer:** The (core) Data Warehouse is where the actual data used primarily for informational uses occurs. In some cases, one can think of the Data Warehouse simply as a logical or virtual view of data. In a Physical Data Warehouse, copies of operational and or external data are actually stored in a form that is easy to access and is highly flexible. Increasingly, Data Warehouses are stored on client/server platforms, but they are often stored on mainframes as well.
16. **Data Staging Layer:** The final component of the Data Warehouse Architecture is Data Staging. Data Staging is also called copy management or replication management, it includes all the processes necessary to select, edit, summarize, combine and load data warehouse and information access data from operational and/or external databases. Data Staging often involves complex programming, but increasingly data warehousing tools are being created that help in this process. Data Staging may also involve data quality analysis programs and filters that identify patterns and data.

***Database Partitioning: The Basics***

Historically, those who have found the most benefit from database partitioning have had one or more of the following characteristics present in their environment:

Database systems supporting mission-critical applications.

Systems characterized by high-volume OLTP workloads.

Database systems supporting OLAP, DSS, data warehousing or data mining activities.

Some proponents of database partitioning see an even broader application for its use. ―In large enterprise environments, as soon as you have single objects getting larger than 1 to 10GB, partitioning can make sense,‖ says Hermann Baer, a principal product manager for data warehousing at Oracle. Baer notes, occasions where customers have been able to use partitioning to break up data in an important table and thus isolate risk that the overall table will become unavailable. Such an approach, he says, also allows for more rapid recovery of isolated partitions from various types of disruption

management, it includes all the processes necessary to select, edit, summarize, combine and load data warehouse and information access data from operational and/or external databases. Data Staging often involves complex programming, but increasingly data warehousing tools are being created that help in this process. Data Staging may also involve data quality analysis programs and filters that identify patterns and data.

##### Database Partitioning: The Basics

Historically, those who have found the most benefit from database partitioning have had one or more of the following characteristics present in their environment:

Database systems supporting mission-critical applications.

Systems characterized by high-volume OLTP workloads.

Database systems supporting OLAP, DSS, data warehousing or data mining activities.

Some proponents of database partitioning see an even broader application for its use. ―In large enterprise environments, as soon as you have single objects getting larger than 1 to 10GB, partitioning can make sense,‖ says Hermann Baer, a principal product manager for data warehousing at Oracle. Baer notes, occasions where customers have been able to use partitioning to break up data in an important table and thus isolate risk that the overall table will become unavailable. Such an approach, he says, also allows for more rapid recovery of isolated partitions from various types of disruption Forrester‘s Yuhanna has seen partitioning used most of the time in data warehousing environments, where the motivation for use is fairly straightforward.

―If you have a large database, you want to manage it in chunks to perform better data management. The major focus of database partitioning over the last decade has been on data warehouses and data marts, where you break down those queries into smaller units so they can run in parallel on different partitions,‖ he says. ―It is divide and conquer. You divide the data and conquer. If you have large amounts of data running on a single processor, that query can be decomposed into smaller units that run concurrently with multiple processors, where each processor goes after a single partition of data‖. Yuhanna says Database vendors have different ways of implementing database partitioning, with some offering more ways to slice and dice the data than others. The most common database partitioning method is to split up a larger, time-based table — like a table containing sales data

— into individual partitions by week, month, quarter or year. This is known as range partitioning because it allows a range of dates or numbers to be represented in the partition, with obvious start and end points for the range. Figure 2.1 shows a typical sales table partitioned by month.

Once the partitions have been defined in a table, queries that run against the table typically are optimized to automatically ―prune‖ unnecessary partitions out of the query process before running the query. For example, if you want to analyze all of the sales made for the last two weeks of April, your query would go to the sales table and automatically process data from just the ―April‖ partition of the table, versus data from all 12 partitions. Users don‘t need to know about the partitioning structure, either. Queries use the same language, whether the table is partitioned or not. What‘s different is the underlying efficiency partitioning now builds into the process. This process can result in significant reductions in processing time. Baer explains it like this: ―In the simplest example of a time- based partitioning strategy partitioned by month, you might only want to look at the last month (or one-twelfth) of data in a table. That means the resources to satisfy the business requests are also one-twelfth as shown in figure 2.1. Staying with this simplistic example, you could then theoretically run 12 times the number of users on that same hardware or make queries run 12 times faster.‖

# Developments and Research in Business Intelligence Technology

From the very first Business Intelligence (BI) systems which began with the early mainframe reports, one the main goal of researchers have been to cut down the costs of business intelligent systems. This can be observed from the evolution of business intelligent system outlined above. Few can argue anymore with the benefits of knowledge and information driven management. Long heralded by such business luminaries as Peter Drucker, information and knowledge have emerged as the key to business success in the 21st century. Although BI plays a central role in enabling organizations to leverage the power of information and data, companies have long struggled with the fragmented nature of BI as it has evolved. The fragmented nature of BI drives up the cost of BI initiatives, limits its application, and reduces its effectiveness in driving business improvement. Another significant problem with a fragmented approach to BI is the potential for delivery of inconsistent and unreliable information. With this and the other shortcomings of a fragmented approach, what is required is a unified approach to BI in which the different pieces are addressed in a single, integrated solution

# ETL modeling concepts

The general framework for ETL processes is shown in Fig. 1. Data is extracted from different data sources, and then prop- agated to the DSA where it is transformed and cleansed be- fore being loaded to the data warehouse. Source, staging area, and target environments may have many different data structure formats as ﬂat ﬁles, XML data sets, relational tables, non-relational sources, web log sources, legacy sys- tems, and spreadsheets.

##### The ETL phases

During the ETL process, data is extracted from an OLTP dat- abases, transformed to match the data warehouse schema, and loaded into the data warehouse database (Berson and Smith, 1997; Moss, 2005). Many data warehouses also incorporate data from non-OLTP systems, such as text ﬁles, legacy sys- tems, and spreadsheets. ETL is often a complex combination of process and technology that consumes a signiﬁcant portion of the data warehouse development efforts and requires the skills of business analysts, database designers, and application developers. The ETL process is not a one-time event. As data sources change the data warehouse will periodically updated. Also, as business changes the DW system needs to change – in order to maintain its value as a tool for decision makers, as a result of that the ETL also changes and evolves. The ETL processes must be designed for ease of modiﬁcation. A solid, well-designed, and documented ETL system is necessary for the success of a data warehouse project.

An ETL system consists of three consecutive functional steps: extraction, transformation, and loading:

##### Extraction

The ﬁrst step in any ETL scenario is data extraction. The ETL extraction step is responsible for extracting data from the source systems. Each data source has its distinct set of charac- teristics that need to be managed in order to effectively extract data for the ETL process. The process needs to effectively inte- grate systems that have different platforms, such as different

During extracting data from different data sources, the ETL team should be aware of (a) using ODBC JDBC drivers con- nect to database sources, (b) understand the data structure of sources, and (c) know how to handle the sources with different nature such as mainframes. The extraction process consists of two phases, initial extraction, and changed data extraction. In the initial extraction (Kimball et al., 1998), it is the ﬁrst time to get the data from the different operational sources to be loaded into the data warehouse. This process is done only one time after building the DW to populate it with a huge amount of data from source systems. The incremental extraction is called changed data capture (CDC) where the ETL processes refresh the DW with the modiﬁed and added data in the source systems since the last extraction. This process is periodic according to the refresh cycle and business needs. It also captures only chan- ged data since the last extraction by using many techniques as audit columns, database log, system date, or delta technique.

##### Transformation

The second step in any ETL scenario is data transformation. The transformation step tends to make some cleaning and con- forming on the incoming data to gain accurate data which is correct, complete, consistent, and unambiguous. This process includes data cleaning, transformation, and integration. It de- ﬁnes the granularity of fact tables, the dimension tables, DW schema (stare or snowﬂake), derived facts, slowly changing dimensions, factless fact tables. All transformation rules and the resulting schemas are described in the metadata repository.

##### Loading

Loading data to the target multidimensional structure is the ﬁ- nal ETL step. In this step, extracted and transformed data is written into the dimensional structures actually accessed by the end users and application systems. Loading step includes both loading dimension tables and loading fact tables.

##### Models of ETL processes

This section will navigate through the efforts done to concep- tualize the ETL processes. Although the ETL processes are critical in building and maintaining the DW systems, there is a clear lack of a standard model that can be used to represent the ETL scenarios. After we build our model, we will make a comparison between this model and models discussed in this section. Research in the ﬁeld of modeling ETL processes can be categorized into three main approaches:

1. Modeling based on mapping expressions and guidelines.

2. Modeling based on conceptual constructs

In the following, a brief description of each approach is presented.

Modeling ETL process using mapping expressions

Rifaieh and Benharkat (2002) have deﬁned a model covering different types of mapping expressions. They used this model to create an active ETL tool. In their approach, queries are used to achieve the warehousing process. Queries will be used to rep- resent the mapping between the source and the target data; thus, allowing DBMS to play an expanded role as a data trans- formation engine as well as a data store. This approach enables a complete interaction between mapping metadata and the warehousing tool. In addition, it addresses the efﬁciency of a query-based data warehousing ETL tool without suggesting any graphical models. It describes a query generator for reus- able and more efﬁcient data warehouse (DW) processing.

Mapping guideline

Mapping guideline means the set of information deﬁned by the developers in order to achieve the mapping between the attri- butes of two schemas. Actually, different kinds of mapping guidelines are used for many applications. Traditionally, these guidelines are deﬁned manually during the system implementa- tion. In the best case, they are saved as paper documents. These guidelines are used as references each time there is a need to understand how an attribute of a target schema has been gener- ated from the sources attributes. This method is very weak in the maintenance and evolution of the system. To keep updating these guidelines is a very hard task, especially with different ver- sions of guidelines. To update the mapping of an attribute in the system, one should include an update for the paper document guideline as well. Thus, it is extremely difﬁcult to maintain such tasks especially with simultaneous updates by different users.

3.1.2. Mapping expressions

Mapping expression of an attribute is the information needed to recognize how a target attribute is created from the sources attributes. Examples of the applications where mapping expressions are used are listed as follows:

Schema mapping (Madhavan et al., 2001): for database schema mapping, the mapping expression is needed to deﬁne the correspondence between matched elements.

Data warehousing tool (ETL) (Staudt et al., 1999): includes a transformation process where the correspondence between the sources data and the target DW data is deﬁned. EDI message mapping: the need of a complex message trans- lation is required for EDI, where data must be transformed from one EDI message format into another.

EAI (enterprise application integration): the integration of information systems and applications needs a middleware to manage this process (Stonebraker and Hellerstein, 2001). It includes management rules of an enterprise’s appli- cations, data spread rules for concerned applications, and data conversion rules. Indeed, data conversion rules deﬁne the mapping expression of integrated data.