Contents

[The Business Intelligence Concept 2](#_Toc1551493)

[Business Intelligence Systems 2](#_Toc1551494)

[The Data Warehouse concept 2](#_Toc1551495)

[Extract, Transform, Load (ETL) 3](#_Toc1551496)

[Evolution of ETL Architecture 4](#_Toc1551497)

[Towards real-time Business Intelligence 5](#_Toc1551498)

[First generation of Business Intelligence Systems 5](#_Toc1551499)

[Second generation of Business Intelligence Systems 6](#_Toc1551500)

[Third generation of Business Intelligence Systems 6](#_Toc1551501)

[Towards Real-time ETL 7](#_Toc1551502)

[ETL techniques - A comprehensive review 7](#_Toc1551503)

[Querying directly the source system for reporting 7](#_Toc1551504)

[Materialized views 8](#_Toc1551505)

[Differential Snapshots 8](#_Toc1551506)

[Full load of the data source 9](#_Toc1551507)

[Process of Elimination 9](#_Toc1551508)

[Change Data Capture 9](#_Toc1551509)

[Timestamps 9](#_Toc1551510)

[Triggers 10](#_Toc1551511)

[Log scraping 11](#_Toc1551512)

[Enterprise Application Integration (EAI) 12](#_Toc1551513)

[Service-Oriented Architecture as an EAI approach 12](#_Toc1551514)

[Data Streaming Platforms 13](#_Toc1551515)

[Traditional ETL vs Streaming Platforms 14](#_Toc1551516)

[Big Data 14](#_Toc1551517)

[A business intelligence system for healthcare monitoring 16](#_Toc1551518)

[The two questions this report is attempting to answer are the following: 17](#_Toc1551519)

## The Business Intelligence Concept

(Connolly and Begg, 2015) defines Business Intelligence as *“an umbrella term that refers to the processes for collecting and analyzing data, the technologies used in these processes, and the information obtained from these processes with the purpose of facilitating corporate decision making”.*  The term Business Intelligence is a relatively new one as it appeared around mid - 90s. However, systems that provide Business Intelligence exist since early 1980s – they used to be called Enterprise Information systems (EIS) (Sharda et al., 2015). EIS were the first systems that had the capability of providing advanced data analytics such as forecasting, prediction and ad-hoc reporting. EIS replaced the Management Reporting Systems (MRS) that showed up in the 1960s and had very limited capabilities. As the EIS continued evolving, they were enhanced with additional features such as artificial intelligence and data mining. These systems are now called Business Intelligence systems (BIS) and are a very important element of a modern business in terms of supporting decision making.

## Business Intelligence Systems

(Sharda et al., 2015) defines a Business Intelligence System as system that has four components. A Data Warehouse, Business Analytics, a User Interface and Business Performance Management (BPM). The data warehouse is the central repository of data that have been collected from different internal or external sources. Business Analytics refer to tools that are used for querying, mining or analyzing data from the data warehouse such as OLAP tools or Data Mining tools. The User Interface refers to the dashboards or reports that can be produced by modern reporting tools. Finally the business performance management (BPM) is tightly connected with Business Intelligence as the latter can help BPM to identify strategy requirements, drive performance and monitor achievement. (Fidler, 2016). This project will focus on the data warehouse component, and more specifically of the methods that are used to extract data from data sources and the load the data into the data warehouse, after performing any necessary transformations and data cleansing.

## The Data Warehouse concept

A data warehouse is the main component of a business intelligence system. It is another database, technically similar to the operational databases, but serves a completely different purpose. While an operational database is optimized to process transactions efficiently and assist the knowledge workers to their day to day tasks, a data warehouse is focused on analytical processes and has the goal to improve decision making. A data warehouse is a decision support system that provides clean and credible data. Businesses are using operational systems to get data in and data warehouses to get data out. (Kimball and Ross, 2013). There are many data warehouse design approaches but two of them are the most prevalent. Bill Inmon, known as the “father of the data warehouse” suggests a centralized data warehouse architecture with dependent data marts while Ralph Kimball advocates the data mart bus architecture with linked dimensional data marts. (paper-43). (figures appendix). The data warehouse is built in the 3d normal form, after an extensive process where the business requirements across all departments of the enterprise are defined. Then data marts that support business processes or specific subjects/functions of the enterprise are built on top of the enterprise data warehouse. On the contrary, Kimball’s architecture follows a bottom-up approach and a different philosophy. The data warehouse is built step by step by creating data marts. Each of the data marts supports a different business process and is modelled by a specific data modelling technique called dimensional modelling. (see appendices). The data marts are linked via conformed dimensions and form the data warehouse. The design approach of the data warehouse affects also the design of the ETL process, as the loading stage is dependent on the underlying data model. The designers of the ETL process should be aware of the architecture of the data warehouse and adjust the design of the loading stage accordingly.

## Extract, Transform, Load (ETL)

ETL stands for Extract, Transform and Load and it’s a term widely used in data warehousing. The ETL system is the backbone of a data warehouse as it is responsible for the extraction of the data from external sources, the cleaning of the data and the loading of the data into the data warehouse. Behrend and Jörg research (as cited in paper 32) found that seventy percent (70%) of the design, implementation and maintenance of a data warehouse is allocated to the ETL system, which is a complex project divided into many subtasks. There are many different methodologies, tools and technologies for ETL development and implementation. ETL design is a significant part of the Business Intelligence lifecycle (Moss and Atre Shaku, 2003). Before implementing any data pipelines , The ETL Team should take into consideration the business requirements and plan the ETL solution accordingly.

The extract step of the ETL process should be planned based on the business needs. The business requirements set by end users define the data sources or specific entities and attributes of an operational system that need to be considered for integration into the data warehouse. Data sources or other database objects that are not useful for analysis should not be considered. The Transform step is also driven by the business needs. Specific business rules are applied at this step. The data are cleaned, conformed and ready to be imported in the data warehouse. The transform step is very important as it is responsible for providing a high level of data quality to the end users. After the data have been transformed, they are loaded to the data warehouse.

The end users want to have easy access to information and they should be able to understand the underlying data model. The Business Intelligence Team should chose a data model that is simple, scalable and efficient and the ETL Team needs then to design the ETL processes to load the data efficiently in the data structure that has been implemented based on the chosen data model.

An additional business requirement that affects all of the steps above is the data latency requirement (Kimball Ralph, 2004). The frequency the data warehouse needs to be updated with fresh data is one of the most important aspects to consider by the design of an ETL solution. The update frequency of a data warehouse / data mart is driven by the business needs and can vary from once per month to real-time. The data latency requirement can have a huge impact in the design of an ETL solution as it can define the ETL Software/Hardware Architecture. Real-time ETL solutions require modern ETL tools that use a different architecture.

## Evolution of ETL Architecture

(Paper 44) identifies three distinct generations of ETL tools. The ETL tools of the first generation were written in the native language of the operating system and did not have their own ETL engine. Most of them were used in mainframes and were written in COBOL or C. These tools were single threaded and didn’t support parallelism. In addition, they required developers with strong programming skills.

The ETL tools of the second generation appeared in the mid -1990s. These tools did not generate code but had their own internal ETL engines. In most cases these tools were using a dedicated ETL server, which acts as a hub server between the data sources and the target databases. The advantage was that the data processing was taking place in the ETL server and not in the source systems. In addition, the developers did not have to know how to programming in different languages (the native language of the operating system) but only in one programming language, the one of the ETL tool. The tools of this generator were supporting limited parallel processing and had a GUI.

The ETL tools of the third generation, are either code generators or engine based but are very powerful and advanced. These tools support multiple data sources and have pre-build connectors for hundreds of applications and APIs. In addition they offer advanced transformations and support specific functions of data warehousing and dimensional modeling such as the automatic creation of a slow changing dimensions, surrogate keys as well as easy deployment and scheduling. These modern ETL tools are able to provide advanced parallel processing and a very user friendly visual environment.

As Business Intelligence evolves and the user community becomes more energetic, the requirement of near real-time ETL has led the ETL vendors to enhance the ETL tools with advances features that support real-time or near real-time ETL. Therefore, in addition to paper 44 work, which was published a decade ago, an emerging fourth generation of ETL systems should be identified. These systems have kept the advanced features of their predecessors but are using advanced techniques for real time extraction, transformation and loading of data, either as standalone products or as add-ons to existing ETL technology. Log based Change Date Capture and Streaming data are some terms that are tightly connected to the fourth generation of ETL tools. These technologies will be extensively described in the following paragraphs.

# Towards real-time Business Intelligence

## First generation of Business Intelligence Systems

In the early days of Business Intelligence Systems, data warehouses were serving a very specific purpose: the storage of historical information across different data sources in a central database. Users would analyze then the historical data with the assistance of Business intelligence applications and take decisions based on their findings. The analysis of the data would answer the question “What happened?” For a local super market this would be translated to “How many products did we sale last week?” or “What was the product with the highest sales on Christmas Eve?”. The update frequency of a data warehouse with fresh data would vary from monthly to weekly basis. Later this would change to once per day, most of the times overnight. The ETL processes could cause a performance impact to the OLTP systems but during the night, the production databases were operating with a low load or they could also go offline if needed. However, users and medium level management had also the need to analyze operational data on a daily basis. Standard reports e.g., total sales of yesterday were generated by SQL queries that were running directly on the operational databases. As this was a bad practice that had a performance impact on the OLTP systems, the Business Intelligence Systems were enhanced with an extra layer: The Operational Data Store (Kimball and Ross, 2013). The Operational Data Store was a staging area in the data warehouse or another database that was hosting a copy of the data sources tables. This database was populated either by ETL tools or by data replication software. Ad-hoc SQL queries or Reporting used as a source the data in the operational data store instead of the OLTP systems. By doing this, the knowledge workers could get the operational data quickly without placing a burden in the production systems. To summarize, the first generation of Business Intelligence Systems had an Operation Data Store layer for operational reporting and a data warehouse that was updated on monthly/weekly/daily basis for historical analysis. As aptly described by the white paper of (GoldenGate, 2009), data warehouses of that generation were *strategic* with emphasis on *reporting.*

## Second generation of Business Intelligence Systems

As more and more businesses are becoming data-driven and the amounts of data are growing at a rapid pace, the business intelligence systems should adapt to the new conditions. The business requirements of the users have changed - the user community demands now not only valid, accurate and cleansed data but also fresh data- a low data latency from the target databases to the data warehouse that allows the decision makers to analyze data quickly and effectively. Many data warehouses as of today belong to this generation. The data warehouse refresh rate has been increased up to hourly basis. In addition, both OLTP and OLAP systems are more powerful than before. This had led to significant changes in the Business Intelligence Architecture – the operational data store is gone and the data warehouses is populated directly from the production systems. Data Warehouse Experts, such as Ralph Kimball (Kimball and Ross, 2013) have removed the operational data store from the architecture they propose. Simple or ad-hoc operational reports use a source either directly the powerful OLTP systems or the data warehouse which is up to date. Dashboards and advanced analytics are now easier and more meaningful to generate, not only because the low data latency but also because the high-end commercial Business Intelligence tools that are now available. GoldenGate (2009) names the data warehouses of this generation as *strategic and predictive*. Business Intelligence is now able to answer the questions “What is happening?” and “What will happen?”

## Third generation of Business Intelligence Systems

The third generation of Business Intelligence Systems consist of the so called Active or Real-Time Data Warehouses. Stephen Brosbt research in 2001 (as cited by (Nguyen Manh Tho and Tjoa Min, 2006)) describes the Active Data Warehouse as a decision-making system that can automatically trigger automatic reactions to specific events. The Active Data Warehouse answers the question “What do I want to happen”. This generation of business intelligence systems is able to handle large amounts of data by using a combination of traditional ETL techniques, streaming platforms and other big data technologies. An example of a complex business intelligence system is the work of (Ali Raza Abbas, 2018). A detailed description of this system is out of the scope of this project, however this is a perfect example to illustrate how important is the role of the ETL process within a business intelligence system as the key components of this framework, which are the data sources, the Big Data Warehouse, the Data Analysis platform and the Analytics and Visualization platform are integrated by the several offline and streaming data flows. Many business intelligence systems of the third generation are using real-time or near real-time ETL processes to drive performance and decision making.

# Towards Real-time ETL

The analysis of the Business Intelligence systems of the previous paragraphs revealed also how the ETL techniques have evolved since the early years of the Business Intelligence. The ETL window, which is defined as the duration of an ETL, is shrinking year by year and many business Intelligence projects require real-time data in the data warehouse. There are several reasons for this trend. (Vassiliadis and Simitsis, 2009) have identified the following ones:

* Many data sources are websites that change very often. If the ETL is not fast enough, important information may be lost.
* Due to increased competition, there is a strong requirement for bigger sales, better monitoring of the needs of the customers, and a more accurate monitoring of the stock market.
* Globalization of the economy e.g., a company can have many branches in areas with different time-zones.

According to the authors, nowadays the ETL techniques are extracting less data from the operational system but faster and more frequently.

# ETL techniques - A comprehensive review

## Querying directly the source system for reporting

This is the simplest method of pulling data from the source systems. An SQL query embedded in a reporting tool or applied directly by an SQL client can pull the data needed from the operational system. In addition, SQL has become very powerful and can support many transformation functions such as creating derived fields, pivoting tables, advanced aggregations and calculations between values of different rows. The pulled data can be stored in a flat file or simply can be shown in a report or a dashboard. This method has significant advantages but also limitations. A report or dashboard that queries directly the data source is updated in real-time: Each time a user runs the report, the data are fresh and up to date. In addition, the implementation of this method is easy and can be done by non-technical users who have at least basic SQL skills. However, as (McKenna, 2011) explains in his work, this method can be used only if the data source contains all data needed and there is no requirement for integration with other data sources. In practice, it is possible to link servers in a single SQL query as most database vendors’ supports this functionality, e.g., the dblink for PostgreSQL databases (PostgreSQL, 2019) but this can have a significant impact on the performance of the query .In addition, linking servers in the same query can lead to complex queries that are not easy to understand. A second drawback of querying directly the data source is that these queries can affect significantly the performance of the OLTP databases. Therefore only short queries with a few joins and simple transformations should be used when this method is applied.

## Materialized views

A view defines a set of base tables to a derived table (Gupta and Mumick, 1999). Among other advantages, defining a new database relation as a view makes querying of the database easier, as they users do not have to write the SQL logic to get the information they need. The view contains the schemas and the SQL logic needed to extract data from different tables. A materialized view can be considered as a cache, as the records of the view are physically stored in the database, thus the view definition does not need to be recomputed each time a user executes the view. Using a materialized view as an ETL method has the same limitations as querying directly the data source. The tables that are used in the materialized view should be part of the same database/server- thus the level of data integration that a materialized view can provide is very limited. Some database vendors support materialized views that do not need to be manually refreshed as they are synchronized to the original tables that the view is using as a source (Microsoft, 2018b). For the SQL Server, these views are called *indexed views*. However, this kind of view should be deterministic and should contain expressions that return always the same results for the same input values. As a result there are many limitations as the SELECT statement of indexed view should not contain specific SQL statements such as COUNT, DISTINCT and UNION. Due to this limitation, an indexed view can be used only for simple queries.

## Differential Snapshots

Some systems may allow only the creation of a database snapshot as an extraction method. Specific algorithms need to be used then to compare the current state of the database with the previous stored snapshot in order to extract the deltas (Ram Prabhu and Do Lyman, 2000). This extraction method is very specific and refers only to legacy databases or special purpose systems that do not support other ways of delta extraction.

## Full load of the data source

In this method, known also as Truncate – Insert, the tables in the target databases are truncated and all records of the source tables are re-inserted in each ETL iteration. The biggest advantage of this method is that is very easy to implement. In addition, re-inserting the whole data source ensures that the target tables are up to date and the most recent changes have been successfully captured. However, this method is obviously not suitable for tables with very large amounts of data (Vassiliadis and Simitsis, 2009). Moreover, specific data models such as dimensional modelling in a data warehouse do not allow deletion of data, as the dimension tables preserve historical information that cannot be retrieved from the data sources (Kimball and Ross, 2013). The Truncate-Insert method could be useful in cases where the changed data captures techniques fail to work correctly and a full load of the source tables is necessary to be applied. In addition, when the amounts of data is relatively small, deletion of data in the target tables is allowed and change data capture techniques cannot be applied for any reason, this ETL method can be an effective solution.

## Process of Elimination

The process of elimination, as described by (Kimball Ralph, 2004) compares the source tables with the target tables row by row to identify changes. The source tables should be first bulk loaded in a staging area. After the comparison is over, only the deltas are imported into the data warehouse.

A way to perform the comparison is to use a hashing algorithm. Many ETL tools support this functionality. The hashing algorithm generates a checksum value based on the values of the input fields. This checksum can be stored as an additional column in both source and target datasets. The ETL process can compare then the checksum values and identify which rows have changed since the last iteration. Although this is a very effective method that can identify also deleted rows, its time consuming and not recommended.

## Change Data Capture

Change data capture (CDC) techniques are able to track changes made to data sources so that the ETL software can then process only the changed rows and import them to the data warehouse. There are three change data capture approaches: timestamps, trigger-based CDC and log-based CDC.

### Timestamps

Timestamps or audit fields are used in the data source tables to store the time of the last modification of each row. It’s common that these fields are called “Updated” or “Last\_modified”. These fields can be used by the ETL process so that only the changed rows are further processed. The ETL process queries first the target table and fetches the most recent value of the audit field which is a timestamp. Then this value is used as a reference to query the source table and to fetch only the rows that have a timestamp greater than the reference one. This technique is efficient but there are some factors that should be taken into consideration:

* The audit fields should not contain null or incorrect values (Kimball Ralph, 2004).
* These fields should have an index to improve querying performance (Kimball Ralph, 2004).
* This approach can track only the last change prior to extraction, therefore it should be used only for data sources that have little change activity (Ram Prabhu and Do Lyman, 2000).
* Track of deletions can be lost as the whole record including the timestamp is deleted from the source system (Guera and Andrews A.David, 2011)

It is important to avoid timed extracts when timestamps are used for data extraction (Kimball Ralph, 2004). A timed extract usually includes all records that have been modified since the previous day (GETDATE-1). This method can lead to missing information if one iteration of the ETL fails and the ETL team is not notified about the failure immediately. Even in that case, the re-initiation of the process (e.g. GETDATE-2) is highly likely to extract duplicated rows that have already been inserted to the data warehouse.

### Triggers

A trigger is a type of stored procedure that is executed when an INSERT, DELETE or UPDATE occurs ((Microsoft, 2017). In the case of change data capture, a trigger fires when a new row is added or an existing row is updated and the primary key of the record is stored in a separate log table in the database. This log table can include additional information such as a flag that indicates the type of operation (INSERT,UPDATE,DELETE) or a timestamp. An ETL process can then extract the records of the log table on a periodic basis and join them with the original data source to obtain the rest of the fields (Kimball Ralph, 2004). These records are redirected then to the transformation component of the ETL before they are inserted to the data warehouse. One advantage of triggers, as stated by (Jain, S and Saluja, 2002) is that they offer synchronous data capture, meaning that the changed data are captured immediately. Another advantage of triggers, according to (Ram Prabhu and Do Lyman, 2000) is that they capture all changes and not only the last one prior to extraction.

However triggers come with a significant drawback as their execution has an impact in the performance of the OLTP database. In the work of (Valêncio Roberto Carlos, Marioto Henrique Matheus and Zafalon Donega Francisco Geraldo, 2013), it is stated that an acceptable percentage increase of the processing time of DML statements is 5-13%. The authors run some benchmarks to rate their own proposal that is based on triggers and the result was a 13% percentage increase in comparison with a database without triggers. This overload time is significant but still acceptable, as the impact in the performance of the OLTP system is not severe. This was expected as these triggers, when executed, are performing INSERTS and not UPDATES and therefore they do not place an unacceptable overhead burden (Kimball Ralph, 2004). Therefore the trigger-based CDC could be an effective solution for deltas extraction. According to (Kotopoulis Alex, 2014), Oracle(citation needed) supports triggers as a CDC method as long as the database loads are low to medium. Another disadvantage of the triggers is that they can easily be deleted/disabled (Guera and Andrews A.David, 2011). This is true as the triggers are maintained in the operational databases and the administrator may not be aware of the ETL process. Furthermore, this method can be considered as an invasive solution as the OLTP system should be modified to enable triggers and additional tables should be created. In some OLTP systems there are policies that do not allow these modifications.

### Log scraping

Log scraping, known also as Log sniffing or Log parsing is another ETL method that uses the database log files to extract the changed data. The transaction or redo log files of a relational database contain all the transactions that have been applied to the database since a specific point of time. These log files are very important element of the database and are used to restore the data in case of disaster. Specific tools called Log scrapers can extract the log records and recreate the SQL statements so that these can be applied then to the target database. The advantage of log-based Change Data Capture is that it does not affect the performance of the database, as the log files exist anyway (Shi et al., 2008; Ram Prabhu and Do Lyman, 2000). In addition, the change data capture component is non-invasive, as it does not modify the source system. An example of a non-invasive log-based CDC system is the solution of Oracle Golden Gate, where the CDC component process the log file and stores the data changes externally and not within the source system (GoldenGate, 2009). There are also other commercial products that support log-based CDC, presented as pre-built real-time data warehouses as the RapidDecision system (Guera and Andrews A.David, 2011). The examples of Golden Gate and RapidDecision show how the log scraping has been evolved since the last decade. According to (McKenna, 2011), log scraping is used mainly for real-time data replication. There are many commercial tools that offer data replication, the author claims that most of these tools cannot transform the data. Data replication software can be used as a disaster recovery or to populate the Operational Data Store that is an exact copy of the source tables. In addition, (Kimball Ralph, 2004) claimed that this method is not reliable as the transaction log of the database can be truncated by a database administrator and as a result, all data changes will be lost. The author suggest that log-based CDC should be used only if other ETL techniques are not available. However, according to Microsoft documentation of the CDC component of the SQL Server, *“the log truncation point will not advance until all the changes that are marked for capture have been gathered by the capture process”* (Microsoft, 2018a). This indicates that no data will be lost if the log is truncated.

## Enterprise Application Integration (EAI)

The Enterprise Application Integration (EAI) is a set of technologies and products that aim to solve the “*problem of building enterprise scale information systems, with streamlined, automated internal business processes and web enabled business functions, all across multiple legacy applications*”(Gorton Ian, and Liu Anna, July 2004). There are many EIA technologies and architectures the description of which is out of the scope of this project. In summary, this technologies were intended to be used for application integration and not for database integration as the techniques described in the previous chapters. However, EAI can be used as a very reliable technique for moving data from data sources to the data warehouse. (Kimball Ralph, 2004) defines 3 main components of an EIA system: The adapter, the broker and the application-independent messages that are exchanged in XML format. For every operational system, such as an ERP, there is an adapter that has the role of creating and executing the messages, whenever there is a data change (e.g., a new customer). The role of the broker is to route the messages between the adapters based on publish-subscribe architecture. The broker will forward the customer message only to the adapters that have subscribed to receive customer specific messages. A data mart adapter can receive the message and import the new data to the data warehouse. According to (Kimball Ralph, 2004) EAI technology can offer real-time data integration, can capture all dimensional changes (and not only the latest change that occurred prior to extraction) and can be used also for more sophisticated approaches where the cleansed data are imported back to the operational systems. (GoldenGate, 2009) considers EAI as a very reliable approach that provides guaranteed data delivery and basic transformations. An important drawback however is that EAI has been initially designed to invoke applications and move instructions – a function that does not require large amounts of data. Therefore there can be some constraints regarding the volume of data that EAI can handle when this technology is used as an ETL method for data warehouses. In (Vassiliadis and Simitsis, 2009) work, EAI is described as a method that uses push technology (appendices) to achieve real-time integration, however is a very complicated and expensive solution that a medium size company could not afford.

### Service-Oriented Architecture as an EAI approach

One way to implement EAI is by using the Service Oriented Architecture (SOA). A Service Oriented Architecture is a software design approach that modularizes applications into small services. The services communicate with each other via messages and are platform independent. A fundamental concept of the Service Oriented Architecture is that it provides services that are reusable across business processes. Service Oriented Architecture can be implemented by using different technologies but the preferred one is Web Services (Papazoglou and Van den Heuvel, 2007). There are some academic publications where the authors have designed/implemented a Business Intelligence System based on the Service Oriented Architecture. One of them is the work of (Wu, Barash and Bartolini, Jul 2007) where a traditional Business intelligence system with conventional ETL technology has been replaced by a new system that is using Web Services based on the Service Oriented Architecture approach. All the legacy components/functionalities such as data extraction, cleaning, mapping, transformations, OLAP and loading to data warehouse or other target systems have been re-architectured into *generic smaller servicing modules* that communicate via open standard messaging protocols such as SOAP and XML. The authors claim that this system provides a scalable and efficient ETL solution and a real-time business intelligence environment. However, they also state that this solution has a high level of complexity and it’s difficult to implement. Another drawback that can be identified is that Web services are not designed to transfer very large amounts of data in batches and therefore an implementation of such as solution may be applicable only for specific cases where the volume of stored data is low.

## Data Streaming Platforms

Stream processing is a term that is related to Big Data technology. According to (Perera Srinath, 2018), stream processing is used to query continuous data streams such as streams from a temperature sensor or a transaction log. Stream processing is called also event processing or real-time analytics. Continuous data streams cannot be handled very well by batch processing, as this requires the data first to be collected, then to be processed and then to go for the next batch. On the other hand, stream processing can process the data naturally, as they come. Querying of data and transformations can be applied by a new technology called Streaming SQL - instead of tables, the user creates streams and queries data that are not stored it a table, but as they come through the streaming platform. A data streaming application (or a Business Intelligence Application that reacts on streaming data) can be built on one of the many data stream processing frameworks, such as Apache Flink (), Spark ()or Samza ().The ETL part of such an application can be done by using a message broker such as Apache Kafka, RabbitMQ or Azure Service Bus. It is important here to note that modern message brokers have very powerful capabilities such as storage and transformation of streams. In its official website, Apache Kafka is not described as message broker but as a distributed streaming platform (Apache Software Foundation, 2017). A key difference with a traditional ETL tool is that a streaming platform can handle easily both data from databases and data from applications. A traditional ETL framework would need EAI architecture to achieve this, which is expensive and complex to implement. In addition, a streaming platform can provide the so called ETL with Microservices, as the developers of each application can easily implement their own ETL solution and connect their application to the streaming platform without the need to make a request to central ETL team.(Shapira Gwen, 2017). On the other hand, these streaming platforms do not provide any new ETL techniques to extract data from the data sources. As an example, Apache Kafka supports a connector for the SQL Server that uses Microsoft’s Change data tracking as a change data capture technique. A traditional ETL tool would use exactly the same technology to pull the data from the SQL Server. Both Kafka and traditional ETL tool would do a micro batch processing to extract changed data. To summarize, a data streaming platform is mainly a data integration solution, not an ETL method. However, if an organization has many applications and databases and decides to use a streaming platform to replace its point to point connections, these streaming platforms can replace a traditional ETL as they provide connectors for data extraction and have strong transformation capabilities.

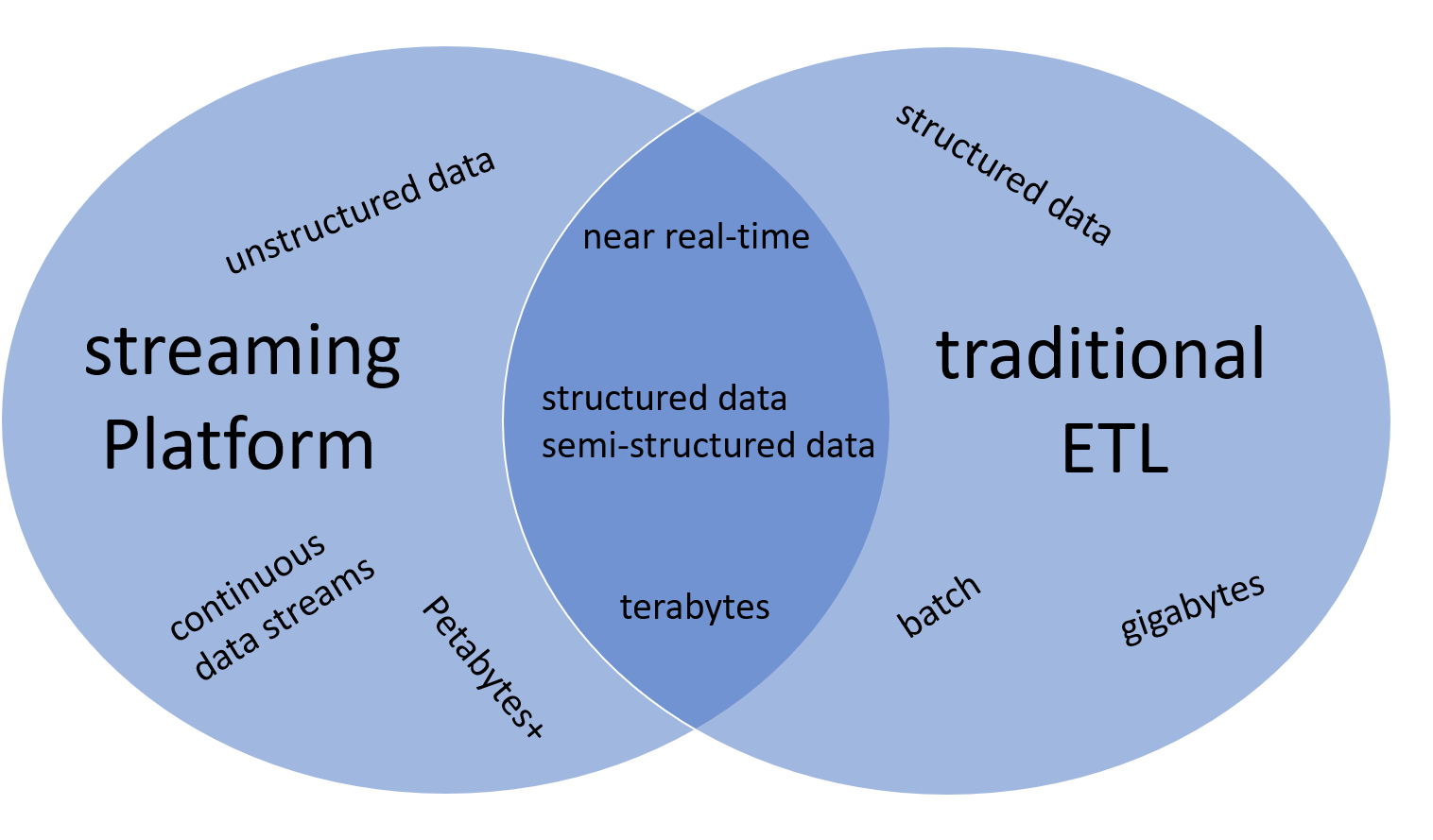
# Traditional ETL vs Streaming Platforms

The review of the ETL methods showed that in terms of data warehousing there are two main approaches. The first approach uses a traditional ETL system to extract, transform and load the data into a data warehouse. The term “traditional” implies that the ETL system belongs to the ETL architecture that was described in the section “ETL Architecture”. A streaming platform could be considered as modern ETL solution as it uses a different technology that can handle more efficient streams of events. The goal of this report is to compare these two approaches and draw conclusions on a set of various criteria that are described in the next sections. This evaluation should not be considered as a direct comparison between two similar products that have been designed for the same purpose. Such an assessment could be done among traditional code-based ETL tools or between two modern streaming platforms. As already described, streaming platform technology does have ETL capabilities but it is considered as a streaming-oriented data/application integration solution for all IT systems and not specifically for data warehousing projects. On the other hand, the term ETL traditionally refers to data warehousing and relational databases. Therefore, it is important to define first the question that this evaluation should answer.

## Big Data

Big Data refers to very large data sets that may have a complex structure and cannot be stored, analyzed and further processed with conventional hardware and tools. Big data can be characterized by three main components: Velocity, Variety and Volume (SAGIROGLU and SINANC, 2013). The velocity refers to the speed of the data –how fast the data are coming in. This can vary from batch data to streams of data. The volume is the size of the data and can be up to huge amounts (e.g, zettabytes). Variety refers to the variety of the different data sources – data can be structured or unstructured (e.g, emails). Understanding first the three Vs’ of the data within an organization is a critical factor to determine the technology and the structure that is going to be used to store and process the data. Very large amounts of unstructured data cannot be handled very well by a relational database system. NoSQL databases would be a better approach as they provide better scalability and flexibility (Madison et al., 2015). In addition, a streaming platform such as Apache Kafka (Apache Software Foundation, 2017) can even replace a NoSQL database, as it has the capability of providing long-term data storage apart from ETL and data processing functionality (Kreps, 2017). However, Apache Kafka and similar streaming platforms cannot be used for data querying. Therefore, a common practice is to extract the data to other systems, such as the Hadoop ecosystem (Madison et al., 2015). Hadoop framework includes ETL technologies that can process the data further such as Apache Flume (Apache, 2019)

This analysis shows that in the world of big data that are characterized by very high Volume, Velocity and Variety a set of new technologies that include NoSQL databases, streaming platforms and data processing ecosystems can extract, transform, load and store that data much more efficiently than a traditional architecture than includes a code-based ETL tool and a relational database system. (Ramu, Hota Kumar Pavan, P. C. and Rao Subba, 2016) considers a traditional ETL system as unable to handle unstructured data. This is mainly because in a traditional ETL system, the schemas and the data structure of the source should be defined before running the ETL. In the unstructured data world though, the data schemas change dynamically and cannot be defined in advance. As a conclusion, a traditional ETL system cannot handle very well unstructured and very large volumes of data. As the purpose of this report it to evaluate a traditional ETL tool, this should be part of a Business Intelligence System that uses structured data with a low Velocity and Volume. However, the amount of data as well as the speed the data that are coming in to the system should justify the selection of a streaming platform as an ETL tool candidate. The amount of data should be large enough and should reflect the current situation many organizations are facing, where the querying performance and data quality are very low due to legacy ETL methods that are still used (e.g, querying directly the source, as mentioned in the first section). In addition, the velocity of the data of the test system should be near-real time as there are many sectors where timely data are crucial for efficient decision making.



This Venn diagram shows the overlapping area where both systems could be used to extract, transform and load data from the source systems to the data warehouse or another target database. This is not to say that a streaming platform cannot be used for batch ETL or a very modern code-based ETL tool is unable to handle unstructured data. However, this graph and the overlapping portion of the circles indicate the data requirements of a Business Intelligence System that can be used as an example in order to perform a fair evaluation of the two systems.

## A business intelligence system for healthcare monitoring

Healthcare sector is among the sectors that generates massive amounts of data such as transactional data (fingerprints, medical images, genetics), human-generated data (prescriptions, emails, electronic medical reports) and machine-to-machine data (sensors, healthcare tracking, devices). Healthcare data have a high level of variety and can be either unstructured or structured. Many authors suggest a Big data framework for Healthcare Business intelligence that includes big data technologies and not conventional ETL methods (Bahri Safa et al., 2018 ; (Gómez Baldominos Alejandro, Rada Fernando and Saez Yago, 2018)). For large scale healthcare data projects, implementing big data technologies could be the only solution, as the velocity, volume and variety is very high. However, for smaller projects that could potentially deal only with the structured data, a traditional ETL system could be enough and handle not only the large amounts of data but also satisfy the requirement for near real time data.

Healthcare monitoring can give not only a detailed description about the condition of patient but also can be life-saving as a real-time monitoring system can detect life-threatening symptoms based on the data that are transmitted by healthcare sensors. Healthcare sensor data are structured, are time sensitive, have a significant size and therefore satisfy the requirements of the Business Intelligence System that were defined in the previous section. In the following scenario, a healthcare company is tracking some basic healthcare metrics from a patient base. The data are transmitted from the healthcare sensors to a MQTT server(mqtt.org, 2019) and then they are pushed to an SQL Server for storage, monitoring and analysis. This system design was initially working efficiently, as the patient base was small, and the sensor data were not considered as time sensitive. There was no live monitoring of the healthcare metrics and the analysts could aggregate the data on the fly -without significant performance issues. However, when this healthcare organization was acquired by a larger enterprise, the new stakeholders decided to expand the patient base and to provide real-time monitoring. The healthcare sensor data would be used now for live monitoring of critical healthcare metrics such as the blood pressure. In addition, new data sources with medical health records should be integrated into the system – combining sensor data with historical information about the condition of the patients could provide valuable insights.

As the requirements of the system changed, the Business Intelligence team had to redesign the system. The first decision that was taken was to build a data warehouse to integrate the sensor and the medical records data. As the data warehouse is used mainly for analysis of historical information, the team decided to create a separate system for real-time monitoring of the data. This system should be able to detect life threating condition based on the values of the sensor data and notify the medical team of the company. The most challenging part of the design is the selection of the ETL tool – as the volume and velocity of the data is now increased, the ETL system should be stable and efficient.

A close up of a sign

Description automatically generated

## The two questions this report is attempting to answer are the following:

* Is a traditional ETL tool

ALI RAZA ABBAS (2018) Real-Time Big Data Warehousing and Analysis Framework. In: *IEEE 3rd International Conference on Big Data Analysis (ICBDA), 2018*: IEEE.

APACHE (2019) *Apache Flume*. [Online] Available from: <https://flume.apache.org/> [Accessed Feb 2019].

APACHE SOFTWARE FOUNDATION (2017) *Apache Kafka*. [Online] [Accessed Jan 2019].

CONNOLLY, T. and BEGG, C. (2015) *Database Systems: A Practical Approach to Design, Implementation and Management*. Sixth ed: Pearson Education Limited.

FIDLER, C. (2016) *Lectures from IMAT5166-2017-1 Fundamentals of Business Intelligence Systems*: De Montford University, Leicester,UK. [Accessed 2016].

GOLDENGATE (2009) *Going Real-Time for Data Warehousing and Operational BI*: GoldenGate Software, Inc.

GÓMEZ BALDOMINOS ALEJANDRO, RADA FERNANDO and SAEZ YAGO (2018) DataCare: Big Data Analytics Solution for Intelligent  
Healthcare Management*. International Journal of Interactive Multimedia and Artificial Intelligence,* 4 (7).

GORTON IAN, and LIU ANNA (July 2004) Architectures and Technologies for Enterprise Application Integration. In: *, May 2004*: IEEE.

GUERA, J. and ANDREWS A.DAVID (2011) *Creating a Real Time Data Warehouse*.

GUPTA, A. and MUMICK, I.S. (1999) *Materialized Views : Techniques, Implementations, and Applications*: The MIT Press.

JAIN, T., S, R. and SALUJA, S. (2002) Refreshing Datawarehouse in Near Real-Time*. International Journal of Computer Applications,* 46 (18).

KIMBALL RALPH, C.J. (2004) *The data warehouse ETL toolkit : practical techniques for extracting, cleaning, conforming, and  
delivering data*. 10475 Crosspoint Boulevard Indianapolis, IN 46256: Wiley Publishing, Inc.

KIMBALL, R. and ROSS, M. (2013) *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. New York, UNITED STATES: John Wiley & Sons, Incorporated.

KOTOPOULIS ALEX (2014) *Best Practices for Real-time Data Warehousing*: Oracle Corporation.

KREPS, J. (2017) *It’s Okay To Store Data In Apache Kafka*. [Online] Available from: <https://www.confluent.io/blog/okay-store-data-apache-kafka/> [Accessed Feb 2019].

MADISON, M. et al. (2015) NoSQL Database Technologies*. Journal of International Technology and Information Management,* 24 (1).

MCKENNA, J. (2011) Moving Toward  
Real-Time Data  
Warehousing*. Business Intelligence,* 16 (3).

MICROSOFT (2018a) *About Change Data Capture*. [Online] [Accessed Jan 2019].

MICROSOFT (2018b) *Create Indexed Views*. [Online] Available from: <https://docs.microsoft.com/en-us/sql/relational-databases/views/create-indexed-views?view=sql-server-2017> [Accessed Feb 2019].

MICROSOFT (2017) *CREATE TRIGGER*. [Online] [Accessed Jan 2019].

MQTT.ORG (2019) *MQTT*. [Online] Available from: <http://mqtt.org/> [Accessed Feb 2019].

NGUYEN MANH THO and TJOA MIN, A. (2006) Zero-Latency Data Warehousing (ZLDWH): the   
State-of-the-art and experimental   
implementation approaches. In: *2006 International Conference onResearch, Innovation and Vision for the Future, 2006*: IEEE.

PAPAZOGLOU, P.M. and VAN DEN HEUVEL, W. (2007) Service oriented architectures: approaches, technologies and research issues*. The International Journal on very Large Data Bases,* 16 (3), pp. 389-415.

PERERA SRINATH (2018) *A Gentle Introduction to Stream Processing*. [Online] [Accessed Jan 2019].

POSTGRESQL (2019) *PostgreSQL:dblink*. [Online] Available from: <https://www.postgresql.org/docs/10/contrib-dblink-function.html> [Accessed Jan 02, 2019].

RAM PRABHU and DO LYMAN (2000) Extracting Delta for Incremental Data Warehouse Maintenance. In: : IEEE.

RAMU, Y., HOTA KUMAR PAVAN, P. C. and RAO SUBBA, V.B. (2016) A Relative Study on Traditional ETL and ETL with Apache Hadoop*. International Journal of Advanced Research in Computer Science and Software Engineering,* 6 (3).

SAGIROGLU, S. and SINANC, D. (2013) Big Data: A Review. In: *International Conference on Collaboration Technologies and Systems (CTS), 2013*: IEEE.

SHAPIRA GWEN (2017) *The Future of ETL Isn’t What It Used To Be*. [Online] [Accessed Jan 2019].

SHARDA, R. et al. (2015) *Business Intelligence: A Managerial Perspective on Analytics, International Edition*. Third ed. Harlow, United Kingdom, UNITED KINGDOM: Pearson Education Limited.

SHI, J. et al. (2008) Study on Log-Based Change Data Capture and Handling Mechanism in  
Real-Time Data Warehouse. In: *International Conference on Computer Science and Software Engineering, 2008*: IEEE.

VALÊNCIO ROBERTO CARLOS, MARIOTO HENRIQUE MATHEUS and ZAFALON DONEGA FRANCISCO GERALDO (2013) Real Time Delta Extraction Based on Triggers to Support Data Warehousing. In: *2013 International Conference on Parallel and Distributed Computing, Applications and Technologies, 2014*: IEEE.

VASSILIADIS, P. and SIMITSIS, A. (2009) Near Real Time ETL. In: *New Trends in Data Warehousing and Data Analysis*: Springer, pp. 19-49.

WU, L., BARASH, G. and BARTOLINI, C. (Jul 2007) A Service-oriented Architecture for Business Intelligence. In: *IEEE International Conference on Service-Oriented Computing and Applications, Jun 2007*: IEEE.