ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (625-631), Month: April - June 2015, Available at: www.researchpublish.com

Bigdata and Bigdata Analytics

[1]Sunita Kanadikar, [2]Navaneetha A R, [3]Mahesh Kaluti

[1, 2, 3] Visvesvaraya Technological University, ALVAS Institute of Engineering and Technology, Mangalore, India

Abstract: **The need of 'big data analytics' in the IT organizations is to ingest, process, and extract business insights from ever larger volumes of data that arrives so much earlier than before, as well as from new sources such as social media, mobile devices, and sensors device ,Research data, Image data are from different sources . However, so as to extract various data feeds from multiple sources, usually these are unrelated sources, these first would like to be correlative to a common level of granularity .so this can be done by using incremental data convergence and soft-key correlation .The aim is process the big data by using the ETL processing and how big data technology allows to store and process the Big data.**

Keywords: **Bigdata, Bigdata analytics, ETL-processing, Incremental data convergence, soft-key correlation, Bigdata Technology.**

I. INTRODUCTION

The Big data refers to datasets whose size is beyond the ability of typical database software tool to capture, store, managed and analyse. Big data is data that goes beyond the traditional limits of data along three dimensions: i) Volume, ii) Variety, iii) Velocity i) volume: Data Volume can be measured by quality of transactions, events and amount of history. Big Data isn't just a description of raw volume. The real challenge is identifying or developing most cost-effective and reliable methods for extracting value from all the terabytes and peta bytes of data now available. That's where Big Data analytics become necessary. ii) Variety: It is the assortment of data. Traditionally data, especially operational data, is structured as it is put into a database based on the type of data (i.e., character, numeric, floating point, etc.)Wide variety of data: internet data (Social media, Social Network-Twitter, Face book), Primary Research (Surveys, Experiences, Observations), Secondary Research (Competitive and Market place data, Industry reports, Consumer data, Business data), Location data (Mobile device data, Geospatial data), Image data (Video, Satellite image, Surveillance), Supply Chain data(vendor Catalogs, Pricing etc), Device data (Sensor data, RF device, Telemetry).

Structured Data: They have predefined data model and fit into relational database. Especially, operational data is structured as it is put into a database based on the type of data (i.e., character, numeric, floating point, etc.) .*Semi-structured data:* These are data that do not fit into a formal structure of data models. Semi-structured data is often a combination of different types of data that has some pattern or structure that is not as strictly defined as structured data. Semi-structured data contain tags that separate semantic elements which includes the capability to enforce hierarchies within the data. *Unstructured data*: Do not have a predefined data model and /or do not fit into a relational database. Oftentimes, text, audio, video, image, geospatial, and Internet data (including click streams and log files) are considered unstructured data. iii) Velocity Data velocity is about the speed at which data is created, accumulated, ingested, and processed. The increasing pace of the world has put demands on businesses to process information in real-time or with near real-time responses. This may mean that data is processed on the fly or while streaming by to make quick, real-time decisions or it may be that monthly batch processes are run inter-day to produce more timely decisions. Bigdata analytics It is the process of Examining Large data sets Containing variety of Data. Here Examining refers to Searching pattern, Finding co-relation, Customer Preference etc.

However, in order to extract valuable business insights from such diverse information feeds using deep analytics, data from multiple, often unrelated sources, first needs to be correlated and reduced to a common level of granularity. Further,

the many valuable insights result only when data from 'new', external 'big data' sources is fused with internal data from within the enterprise [1]. Moreover, the latter currently resides mainly in traditional relational databases. As a result, business intelligence (BI) platforms based on relational databases are being increasingly challenged by the need to ingest and process vast volumes of data from new 'big data' sources together with more traditional data from inside the enterprise. Consequently, BI is becoming dominated by Extraction-Transformation-Load (ETL) processes that consume about 70% of BI resources and effort [2], [3] when engineered using a traditional BI technology stack. The paper is organized as follows: We begin with an overview of enterprise BI, and formally describe the data harmonization problem in Section II. In Section III we describe Big Data Technology and extraction Transformation Load (ETL) processing. In Sections IV and V we describe our technique to address the data Convergence problem and incremental data convergence. Then in Section VI we describe about the soft key co-relation problem. Finally we conclude in Section VIII.

II. OVERVIEW

A. Big Data Technologies:

Hadoop is an open-source platform for storage and processing of diverse data types that enables data-driven enterprises to rapidly derive value from all their data. Advantages of Hadoop : i) The scalability and elasticity of free open-source Hadoop running on standard. ii) hardware allow organizations to hold onto more data and take advantage of all their data to increase operational efficiency and gain competitive edge. iii) Hadoop supports complex analyses across large collections of data at one tenth the cost of traditional solutions. iv)Hadoop handles a variety of workloads, including search, log processing, recommendations systems, data warehousing and video/image analysis.

Apache Hadoop is an open-source project by the Apache Software foundations. The software was originally developed by the world's largest Internet companies to capture and analyse the data that they generate. Unlike traditional, structured platforms Hadoop is able to store any kind of data in its native format and to perform a wide variety of analyses and transformation on that data. Hadoop stores terabytes and even peta bytes of data inexpensively. It is robust and reliable and handles hardware and system failures automatically without losing data analyses. Hadoop runs on clusters of commodity servers and each of those servers has local CPUs and disk storage that can be leveraged by the system.

Components of Hadoop: The two critical components of Hadoop are i) The Hadoop Distributed File System (HDFS):HDFS is the storage system for a cluster. When data lands in the cluster, HDFS breaks it into pieces and distribute those pieces among the different servers participating in the cluster. Each server stores just a small fragment of the complete data set and each piece of data is replicated on more than one server ii) MapReduce : Because Hadoop stores the entire dataset in small pieces across a collection of servers, analytical jobs can be distributed in parallel to each of the servers storing part of the data. Each server evaluates the question against its local fragment simultaneously and reports its result back for collation into a comprehensive answer.

Map Reduce is the agent that distributes the work and collects the results. Both HDFS and Map Reduce are designed to continue to work even if there are failures. HDFS continuously monitors the data stored on the cluster. If a server becomes unavailable, a disk drive fails or data is damaged due to hardware or software problems, HDFS automatically restores the data from one of the known good replicas stored elsewhere on the cluster. Map Reduce monitors the progress of each of the servers participating in the job, when an analysis job is running. If one of them is slow in returning an answer or fails before completing its work, Map Reduce automatically starts another instance of the task on another server that has a copy of the data. Because of the way that HDFS and Map Reduce work, Hadoop provides scalable, reliable and fault-tolerant services for data storage and analysis at very low cost.

III. EXTRACTION TRANSFORMATION LOAD PROCESSING(ETL)

We envision the general architecture of a near real time data warehouse consisting of the following elements: (a) Data Sources hosting the data production systems that populate the data warehouse, (b) an intermediate Data Staging Area (DSA) where the cleaning and transformation of the data takes place and (c) the Data Warehouse (DW). The architecture is illustrated in Figure 1.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (625-631), Month: April - June 2015, Available at: www.researchpublish.com

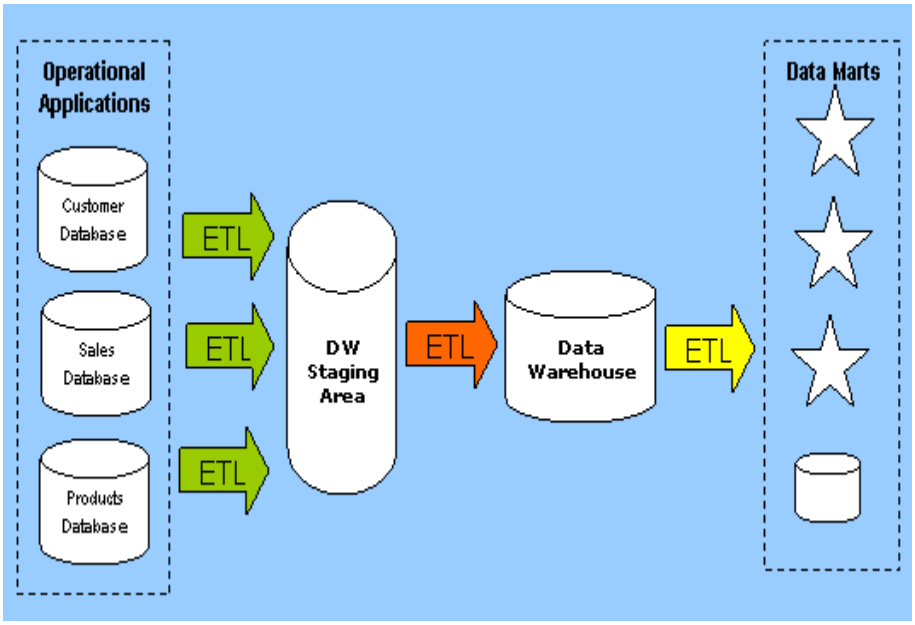

Figure 1: Architecture of near real time data warehouse

Each source can be assumed to comprise a data store (legacy or conventional) and an operational data management system (e.g., an application or a DBMS, respectively.) Changes that take place at the source side have first to be identified as relevant to the ETL process and subsequently propagated towards the warehouse, which typically resides in a different host computer. For reasons that pertain to the QoS characteristics of the near real time warehouse and will be explained later, we envision that each source hosts a Source Flow Regulator (SFlowR) module that is responsible for the identification of relevant changes and propagates them towards the warehouse at periodic or convenient intervals, depending on the policy chosen by the administrators. As already mentioned, this period is significantly higher that the one used in the current state-of-practice and has to be carefully calculated on the basis of the source system's characteristics and the user requests for freshness. A Data Processing Flow Regulator (DPFlowR) module is responsible of deciding which source is ready to transmit data. Once the records have left a certain source, an ETL workflow receives them at the intermediate data processing area.

The primary role of the ETL workflow is to cleanse and transform the data in the format of the data warehouse. In principle, though, apart from these necessary cleansings and transformations, the role of the data processing area is versatile: (a) it relieves the source from having to perform these tasks, (b) it acts as the regulator for the data warehouse, too (in case the warehouse cannot handle the online traffic generated by the source) and (c) it can perform various tasks such as check pointing, summary preparation, and quality of service management. However, it is expected that a certain amount of incoming records may temporarily resort to appropriate Reservoir modules, so that the DSA can meet the throughput for all the workflows that are hosted there. Once all ETL processing is over, data are ready to be loaded at the warehouse. A Warehouse Flow Regulator (WFlowR) orchestrates the propagation of data from the DPA to the warehouse based on the current workload from the part of the end- users posing queries and the QoS "contracts" for data freshness, ETL throughput and query response time. Clearly, this is a load-balancing task, which we envision to be implemented over a tuneable QoS software architecture. The Data Warehouse per se, is a quite complicated data repository. There are different categories of constructs in the warehouse, which we broadly classify as follows: (a) fact tables, containing the records of real-world events or facts, that the users are mainly interested in, (b) dimension tables, which contain reference records with information that explains the different aspects of the facts, (c) indexes of various kinds (mainly, B+-trees and bitmap indexes) which are used to speed-up query processing and (d) materialized views, which contain aggregated information that is eventually presented to the users. In the rest of our deliberations, the term 'materialized view' will be used as a useful abstraction that allows us to abstract all kinds of summaries that are computed once, stored for the users to retrieve and query and regularly updated to reflect the current status of a one or more fact tables (e.g., data marts, reports, web pages, and any possible materialized views per se.) Each of these constructs has a clear role in a traditional ETL setting; in the case of near real time ETL, these constructs are possibly accompanied by auxiliary structures that alleviate the burden of refreshing them very frequently.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (625-631), Month: April - June 2015, Available at: www.researchpublish.com

In an ideal world, all the involved systems (sources, data processing area and warehouse) would be able to process all the data within the given time windows. Clearly, this cannot be the case in practical situation, due to many possible reasons, like the high rate of user queries, the high rate of updates, the high cost of certain parts of the transformation and cleaning stage, or even the failure of a part of the overall architecture at runtime. Practically this results to the necessity of reserving parts of the propagated data for later processing. In other words, a simple selection mechanism in the flow regulators needs to decide which data will be processed by the ETL workflow in near real time and which parts will be reserved in main memory, staged at the hard disk or simply shed in order to be processed later, during an idle period of the warehouse. Similarly, a failure in the near real time processing area or the warehouse can lead to data not propagated to the warehouse on time. These practical considerations lead to the necessity of a compensation scheme that operates in an off-line mode (much like the traditional ETL mechanisms) and completes the missing parts of data in the warehouse and the materialized views.

IV. DATA CONVERGENCE

Now a days enterprises are using the OLAP(online analytical processing)It is the computer processing that enables the user to easily and selectively extract the data from different point of view. For example user may request to show all the data of all the company using OLAP it will be shown as the single point of data. When all the relevant transaction for specific time period is recorded in respective source system then it can be transformed to star schema and loaded to ETL. ETL is the extraction transformation load process.

In ETL during extraction process data is extracted from the various sources then in transformation phase data from the different sources sorted and cleaned then in the load phase the data is loaded in to the data warehouse. Data obtained from different disparate system have different dimension.so it has the following problems i) different data leads to different unit of measurement. For example sales reported daily and sales reported weekly ii) representation refers to different levels of granularity iii) different data needs different encoding schema iv) problem with incompatible data for example when we fuse survey data with the social data.it is therefore important to transform the dimensions and measures to common schema of representation before data of disparate sources can be fused i.e before the join operation is possible.

We define the data convergence as the process of joining different data sets by means of transforming into common schema of representation before the join. The below Figure2 shows the example for the data convergence.

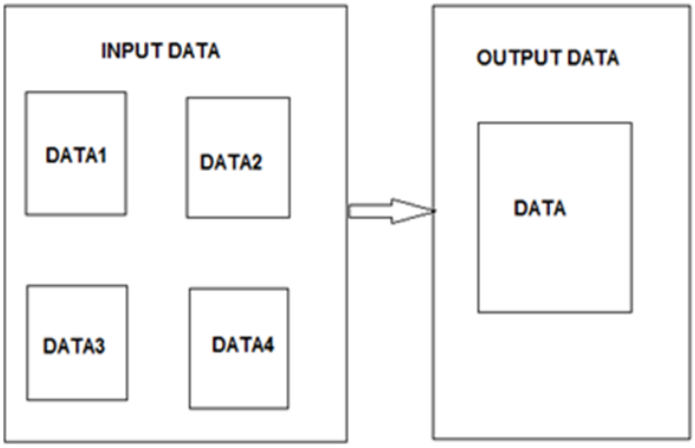

Figure2: Data Convergence

V. INCREMENTAL DATA CONVERGENCE

We first describe data convergence in batch-mode, assuming the data from disparate systems has already arrived; and then motivate need for Incremental Data convergence, followed by a detailed description of how the data can be harmonized in incremental mode. Once data from disparate sources is available, the data can be harmonized following a three phase process. The phases are: Annotation (A), Transformation (T), and Join (J); we also refer to this as the 'ATJ' process. In the annotation phase, we map the columns of the input data with the column specifications given in an 'annotation-file' (pre-created). In the Transformation step, multiple transformations are applied on all input data-sets so as to bring them all

into a common scheme of representation. The transformations include standard ETL processing such as 'cleansing and validation', 'schema alteration (pivot, un-pivot)', and 'dimension' and 'measure' transformations. 'Dimension transformation', performed through a left outer-join with a mapping file, converts the input dimension values to target scheme of representation: The mapping files are created using soft-key-correlation, which leverages machine-learning to create these mappings, and is described in Section VI. Such transformations need not be one-to-one, and are often many-one (e.g. SKU to product-category) or many-one (e.g. approximating selected metrics across multiple 'similar' stores). ' Measure transformation' corresponds to addition of newer derived columns in the data, such as 'weekly gain/loss in sales', which are used for predictive modelling. Once all transformations are complete, a full outer join is performed on the common keys of all the datasets, to obtain the harmonized data. In most enterprise operations, input data for harmonization does not arrive at the same time even for the same time-period. Therefore data needs to be stored in a staging area and can be harmonized after all the data for a specified time-period has arrived. This delay could be avoided if harmonization could be performed incrementally, i.e., by harmonizing data as soon as it arrives. However, there are challenges in doing this; we list those challenges below and then describe our approach for *incremental* data harmonization. If data from only one source arrives for a time-period, we could harmonize and store it in the target database. Later, when data from another source arrives, we might delete the incompletely harmonized data and harmonize both the data-sets all over again, join them, and then store in the target-database. This incremental harmonization is carried out in five phases 1) Annotation, 2) Transformation, 3) Scan, 4) Join, and 5) Index.

This scheme is implemented using map-reduce, resulting in what we call the ATSJI procedure in annotation phase One file for each of the input data containing its column specifications, and one file containing column specifications of the harmonized data. These files are used by subsequent stages of the procedure. Transformation: The input data is transformed into the output schema assigning NULL to the measures that are not present in the current data-set. For example, if there area total of 10 measures in the output schema and only 2 of them ensures are reported in the current data-set, even then all 10 measures are included in the transformed data, marking the remaining measures as *NUL* .

scan: We aim to process chunks of data as they arrive ,rather than wait until all the data is available. As a result, some incoming data records may need to be joined with records processed in earlier iterations. Therefore, in ATSJI, once we transform newly incoming data we need to scan previously harmonized data for records that may *need* to be joined with the freshly transformed data. Such a scan is performed using a 'Group Range Index'. Join: Next both the freshly transformed as well as 'matching' subset of previously harmonized data are inputs to the join stage. A full outer join is performed between all the incoming data and the matching subset of harmonized data Note that the values of any measure columns already presenting the matching subset are replaced with new measures as computed by the join. Most importantly, the output of the join stage is always *inserted* into HBase data store. Index phase : After an incoming dataset is harmonized and stored in the warehouse, the 'Group Range Index' is updated using an additional map-reduce phase. Starting with input data in 'T1', we group by k dimensions as in 'T2' (Here $k <$ total number of dimensions in the data), we then find minimum and maximum values of the primary keys for each such group (which requires map-reduce) to create 'T3'. Data of this table 'T3' is then inserted or updated in the 'Group Range Index' table.

VI. SOFT KEY CO-RELATION

Recall the challenge of matching keys when data is represented differently across data sources, such as two different codes 'Lt. Coffee Can 100', and 'Light Coffee Can 100 gms' both referring to the same SKU (stock-keeping-unit).We refer to this problem as *Soft-Key Correlation*, which is similar to the approximate join problem [6]. In contrast to approximate joins, where UDFs (user defined functions) are used to compare entity-names during execution of join, we first generate the mapping files which are then used by the ATSJI framework for data harmonization. Feature Selection: Since some of the features are more important than others in a given problem-set, we drop features that correlate poorly with the labels in training set. For this, we calculate statistical correlation between labels and the features. Subsequently, absolute values of these scores are normalized and are referred as *feature weights*. We traverse these features in descending order of their weights, and drop all un-traversed features if the difference between weights of two consecutive features is above a prior-threshold ($t1$) ,or if the absolute value of a feature-weight is below another prior-threshold($t2$). Classification and Mapping File Generation: We use a *Support Vector Machine* (SVM) to learn a discriminative statistical model, from the training set against the selected features. This model is then used for classification of all the pairs of strings into 'matching' or 'non-matching' categories. These mappings are stored in a file, which has been referred.

VII. RELATED WORK

Why not active ETL or data warehousing? In [16], the term active data warehousing is used in the sense of what we now call near real time data warehousing-i.e., it refers to an environment "where data warehouses are updated as frequently as possible, due to the high demands of users for fresh data". Some years before, the term active data warehouse has also been used in [17], [18] to refer to a warehouse that is enriched with ECA rules that provide automatic reporting facilities whenever specific events occur and conditions are met. According to R. Kimball seventy percent of the resources needed for the implementation and maintenance of a data warehouse are typically consumed by the ETL system [19]. Incremental ETL problem described in this paper is very different from the incremental map-reduce performed on data-set that is pushed again onto a map-reduce based system after some amendments or modifications as described in [10]. Here, by incremental ETL we indicate a process of on arrival data harmonization, which gets called as soon as data from a source arrives, and performs the transformation of the newly received data only, instead of repeated transformation of previously received data along with the new data.

VIII. CONCLUSION

Data is coming from different sources at different time period so we need to perform incremental data harmonization using ETL process and Map reduce , approximate soft key co-relation technique can be used for big data harmonization. We have formally defined Data convergence, a commonly arising business scenario where data from different and incongruous sources need to be correlated. We have then shown how data harmonization can be accomplished as a formally defined ETL process using the map- reduce programming paradigm and approximate 'soft-key' correlation technique. We have further shown how this can be achieved incrementally, minimizing staging delays as data arrives in bits and pieces.

REFERENCES

[1] G. Shroff, P. Agarwal, and L. Dey, "Enterprise information fusion for real-time business intelligence," in Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on. IEEE, 2011, pp. 1–8.

[2] S. Chaudhuri and U. Dayal, "An overview of data warehousing and olap technology," SIGMOD Rec., vol. 26, no. 1, pp. 65–74, Mar. 1997.

[3] R. Kimball and J. Caserta, The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data,1st ed. Wiley, 9 2004. [Online]. Available: http://amazon.com/o/ASIN/0764-567578.

[4] M. Stonebraker, D. Abadi, D. DeWitt, S. Madden, E. Paulson, A. Pavlo, and A. Rasin, "mapreduce and parallel dbmss: friends or foes?" Communications of the ACM, vol. 53, no. 1, pp. 64–71, 2010.

[5] P. Agarwal, R. Vaithiyanathan, S. Sharma, and G. Shroff, "Catching the long-tail: Extracting local news events from twitter," in Proceedings of Intl. AAAI Conference on Weblogs and Social Media, ser. ICWSM '12, 2012.

[6] L. Gravano, P. Ipeirotis, H. Jagadish, N. Koudas, S. Muthukr- ishnan, D. Srivastava et al., "Approximate string joins in a database (almost) for free," in Proceedings of the interna- tional conference on very large data bases, 2001.

[7] W. W. Cohen, P. Ravikumar, and S. E. Fienberg, "A com- parison of string distance metrics for name-matching tasks," in Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03), ser. IIWeb-03, 2003, pp.73–78.

[8] T. Jorg and S. Deßloch, "Towards generating etl processes for incremental loading," in Proceedings of the 2008 interna- tional symposium on Database engineering & applications. ACM, 2008, pp. 101–110.

[9] P. Vassiliadis and A. Simitsis, "Near real time etl," New Trends in Data Warehousing and Data Analysis, pp. 1–31,2009.

[10] P. Bhatotia, A. Wieder, R. Rodrigu es, U. A. Acar, and R. Pasquin, "Incoop: Mapreduce for incremental computations," in Proceedings of the 2nd ACM Symposium on Cloud Computing, ser. SOCC '11. New York, NY, USA: ACM, 2011, pp. 7:1–7:14.[Online]. Available: http://doi.acm.org/10.1145/2038916.2038923.

[11] A. Abello´, J. Ferrarons, and O.Romero, "Building cubes with mapreduce," in Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP. ACM, 2011, pp. 17–24.

[12] M. Bilenko and R. Mooney, "Adaptive duplicate detection using learnable string similarity measures," in Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2003.

[13] W. Cohen, P. Ravikumar, S. Fienberg et al., "A comparison of string distance metrics for name-matching tasks," in Proceed- ings of the IJCAI-2003 Workshop on Information Integration on the Web -IIWeb-03, 2003.

[14] S. Tejada, C. Knoblock, and S. Minton, "Learning object identification rules for information integration," Information Systems, vol. 26, no. 8, 2001.

[15] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani, "Robust and efficient fuzzy match for online data cleaning," in Inter- national Conference on Management of Data: Proceedings of the 2003 ACM SIGMOD international conference on Management of data, vol. 9, no. 12, 2003.

[16] Alexandros Karakasidis, Panos Vassiliadis, and Evaggelia Pitoura. ETL queues for active data warehousing. In IQIS, pages 28–39, 2005.

[17] Michael Schrefl and Thomas Thalhammer. On Making Data Warehouses Active. In DaWaK, pages 34–46, 2000

[18] Thomas Thalhammer, Michael Schrefl, and Mukesh K. Mohania. Active Data Warehouses: Complementing OLAP with Analysis Rules. Data Knowl. Eng., 39(3):241–269, 2001

[19] R. Kimball and J. Caserta, The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data, John Wiley & Sons, Inc., 2004.