Framework For ETL With Hadoop Map Reduce

Jaswender Malik, Kavita

Computer science & engineering, Maharshi Dayanand University, Rohtak, India
Email: jassi15malik@gmail.com

Abstract: Big Data is dealt by every organization which serves large number of users. Efficiently fetching, transferring, storing, cleaning, sanitizing, querying and extracting information from Big Data is a daunting task because a single machine and the traditional algorithms can't handle this staggering amount of data tractably. Now not all data comes in the form that can be directly processed by automated programs. Before feeding the data into huge data processing systems[1]. It is necessary to treat raw data to convert it into a consistent format. This is done using data cleaning, sanitization and transformation operations. In this paper we present a neat framework for data cleaning and transformation operation which can be integrated in existing Map Reduce (Hadoop) infrastructures. This framework can be standardized and be adopted by corporations for their Big Data processing tasks.

Keywords: ETL, Handler, Usage, Conclusion

Introduction

Internet based organizations collect a lot of data from various sources. For example, log files, user click stream, posts, comments, user agent, ip addresses, geo location. In today web scale companies, this amount of data becomes staggeringly huge over time. Building applications on top of this data have their own challenges. Some of them have been discussed in [2]. As the data keeps on growing and the scale increasing, previously written applications may break. This is discussed in [3]. Extract, Transform and Load (ETL) refers to a process in database usage and especially in data warehousing that: Extracts data from homogeneous or heterogeneous data sources Transforms the data for storing it in proper format or structure for querying and analysis purpose Loads it into the final target (database, more specifically, operational data store, data mart, or data warehouse)[8] ETL jobs are becoming popular now a day in the industry which integrate fetching, cleaning and processing of data and storing results. Usually all the three phases execute in parallel since the data extraction takes time, so while the data is being pulled another transformation process executes, processing the already received data and prepares the data for loading and as soon as there is some data ready to be loaded into the target, the data loading kicks off without waiting for the completion of the previous phases. ETL systems commonly integrate data from multiple applications (systems), typically developed and supported by different vendors or hosted on separate computer hardware. The disparate systems containing the original data are frequently managed and operated by different employees. For example a cost accounting system may combine data from payroll, sales and purchasing.

ETL

It is a popular architectural pattern for data warehousing. ETL stands for Extract, Transform, Load. A detailed treatment of ETL can be found in [4]. A brief overview of the steps in ETL follows.

Extract

Extract refers to extracting data from various sources that produce them. Data ware-housing systems collate data from various logically related sources as shown in figure 3.1. For example, a social gaming company that has its products on Android, iOS, Facebook, Blackberry platforms will collect as much useful data as possible by using these platforms' APIs. Different platforms will provide access to the data in different ways. For example, for the version of game running on Facebook platform, user's identity can be the Facebook ID. On the other hand, for the Android platform it can be the handset's IMEI number or the user's Google account, choosen at the discretion of game developer. The extracted data from different data sources is stored in a single format usually JSON or BSON. Storing in a single format makes applying the transformations easy and decouples it with the Extraction process.

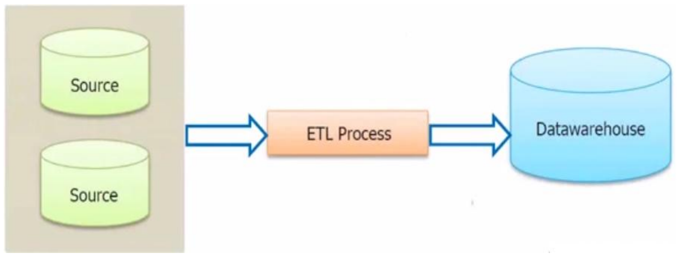

Figure 3.1(a) *Basic ETL process*

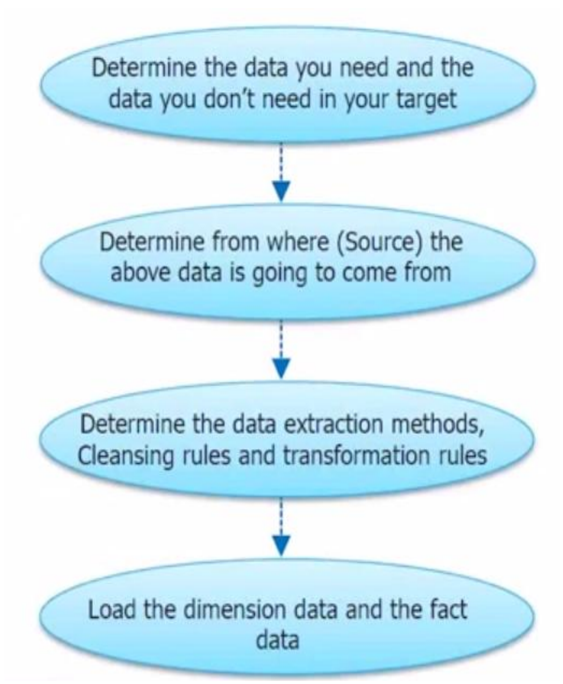
Determine the data you need and the data you don't need in your target
Determine from where (Source) the above data is going to come from
Determine the data extraction methods, Cleansing rules and transformation rules
Load the dimension data and the fact data

Figure 3.1(b) *ETL process in simple language*

Transform
The transform phase applies rules and functions to the data to ensure homogeneity by cleaning up and sanitizing data. It is made sure that the records interesting for and expected by the data processing phase remain and others are discarded. We have identified following transformations that are usually performed on the raw data: Projection: Projection operation projects certain columns and discards others. This is because, not all columns collected from the data sources are relevant to every data processing task. Each data processing task requires a certain set of columns as its inputs. Translation: A data collection system may encode attribute values differently than expected by the data warehousing system. For example the source type field can have integral values as 1 for Facebook, 2 for Android, 3 for Blackberry and 4 for IOS. But the data processing system may represent them as characters F for Facebook, A for Android, B for Blackberry and I for IOS. The translation operation performs such translations and adapts incoming data to the expected format.

Derivation:
This operation involves deriving new columns based on the values in existing columns. For example, an age column can be derived from the birthday column which involves a trivial calculation.

Sorting :
The rows can be sorted based on the values in a particular column.

De-duplication:
This operation involves removal of some rows depending on the duplicate values in one or more designated columns.

Aggregation:
This involves applying aggregate functions on a particular column that summarizes several rows and reduces the number of rows to be processed.

Splitting:
This involves splitting a single column into multiple ones. An example can be a column that has values as a comma separated list.

Validation:
This involves performing validation on certain fields. If a particular field value doesn’t pass validation certain actions may be taken to convert it into a valid data item or the entire row may be rejected.

Load
This phase involves transferring the transformed data into the storage system which can be a distributed file system like HDFS[5]or a NoSQL[6] database or a full fledged data warehouse[7] We present handler based architecture for a data cleaning framework that can be used before the map operations. The inheritance hierarchy in Fig. 3.2 illustrates the relations between the handlers and their organisation. We implemented the framework in Java programming language.

Handlers
Transformation Handler is an interface implemented by all handler classes. It exposes a single method named transform that takes as input a list containing individual map of key value pairs where each map represents a record. The method returns a list of trans-formed records. We have identified eight kinds of transformations mostly prevalent now a day whose implementation in our architecture is discussed below. Each transformation kind is specified by an abstract class that implements the Transformation Handler interface.

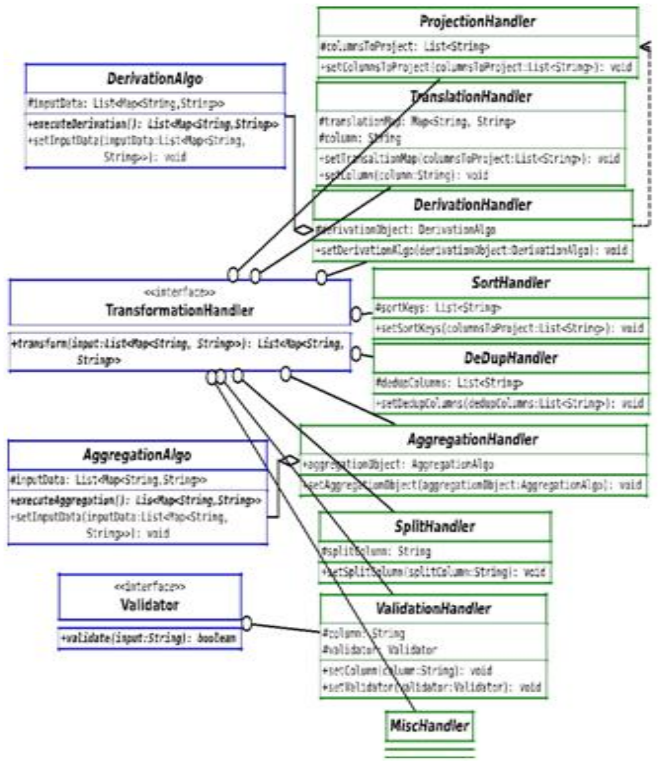
ProjectionHandler
#columnsToProject: List<String>
+setColumnsToProject(columnsToProject:List<String>): void
DerivationAlgo
#inputData: List<Map<String,String>>
+executeDerivation(): List<Map<String,String>>
+setInputData(inputData:List<Map<String, String>>): void
TranslationHandler
#translationMap: Map<String, String>
#column: String
+setTransaltionMap(columnsToProject:List<String>): void
+setColumn(column:String): void
DerivationHandler
#derivationObject: DerivationAlgo
+setDerivationAlgo(derivationObject:DerivationAlgo): void
<<interface>>
TransformationHandler
+transform(input:List<Map<String, String>>): List<Map<String, String>>
SortHandler
#sortKeys: List<String>
+setSortKeys(columnsToProject:List<String>): void
DeDupHandler
#dedupColumns: List<String>
+setDedupColumns(dedupColumns:List<String>): void
AggregationAlgo
#inputData: List<Map<String,String>>
+executeAggregation(): List<Map<String,String>>
+setInputData(inputData:List<Map<String, String>>): void
AggregationHandler
+aggregationObject: AggregationAlgo
+setAggregationObject(aggregationObject:AggregationAlgo): void
SplitHandler
#splitColumn: String
+setSplitColumn(splitColumn:String): void
<<interface>>
Validator
+validate(input:String): boolean
ValidationHandler
#column: String
#validator: Validator
+setColumn(column:String): void
+setValidator(validator:Validator): void
MiscHandler

Figure 3.2: *Inheritance hierarchy of handler based architecture*

Usage
This framework can be used by following the steps below: Subclass provided abstract classes and implements the required methods. Also implement the algorithm objects for the derivation handler, aggregation handler and validation handlers. Once all the concrete subclasses have been written, a main method in a separate Main class can be used to instantiate and apply all the handlers in the user defined sequence on the input set of records which should be in the form of a list of records where each record is a map of string, string pairs. A separate class should be responsible for converting the input set of records into the data structure expected by the handlers. After all the handlers have been applied, another class should be used to perform the inverse of previous step. The transformed data can be loaded into the data warehouse or HDFS or a NoSQL database.

Conclusion
Most organizations don’t use a standard technique for data cleaning. The framework proposed by us can be

standardized and used by any company for their data cleaning and sanitizing operations. It scales well, is modular and follows software engineering principles. The framework was proved to work on a large dataset in an industrial setting.

References

[1] Dean, J. and S. Ghemawat (2008). Mapreduce: simplified data processing on large clusters. Commun. ACM, 51(1), 107–113. ISSN 0001-0782. URL http://doi. acm.org/10.1145/1327452.1327492.

[2] Agrawal, D., S. Das, and A. El Abbadi, Big data and cloud computing: current state and future opportunities. In Proceedings of the 14th International Conference on Ex-tending Database Technology, EDBT/ICDT ’11. ACM, New York, NY, USA, 2011. ISBN 978-1-4503-0528-0. URL http://doi.acm.org/10.1145/1951365. 1951432.

[3] Jacobs, A. (2009). The pathologies of big data. Commun. ACM, 52(8), 36–44. ISSN 0001-0782. URL http://doi.acm.org/10.1145/1536616.1536632.

[4] El Akkaoui, Z., E. Zimànyi, J.-N. Mazón, and J. Trujillo, A model-driven framework for etl process development. In Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP, DOLAP ’11. ACM, New York, NY, USA, 2011. ISBN 978-1-4503-0963-9. URL http://doi.acm.org/10.1145/2064676. 2064685.

[5] Shvachko, K., H. Kuang, S. Radia, and R. Chansler, The hadoop distributed file system. In Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), MSST ’10. IEEE Computer Society, Washington, DC, USA, 2010. ISBN 978-1-4244-7152-2. URL http://dx.doi.org/10.1109/MSST. 2010.5496972.

[6] Hecht, R. and S. Jablonski, Nosql evaluation: A use case oriented survey. In Pro-ceedings of the 2011 International Conference on Cloud and Service Computing, CSC ’11. IEEE Computer Society, Washington, DC, USA, 2011. ISBN 978-1-4577-1635-5. URL http://dx.doi.org/10.1109/CSC.2011.6138544.

[7] Chaudhuri, S. and U. Dayal (1997). An overview of data warehousing and olap tech-nology. SIGMOD Rec., 26(1), 65–74. ISSN 0163-5808. URL http://doi.acm. org/10.1145/248603.248616.

[8] http://en.wikipedia.org/wiki/Extract,_transform,_load