ISSN:0975-9646

Automation in ETL Testing

Omkar Sanjay Kulkarni

Computer Science Graduate (2013-2017)
CSE Walchand College Of engineering, Sangli, India

***Abstract* — ETL which stands for (Extract Transform Load) is a process used to extract data from different data sources, transform the data and load it into a Data Warehouse System. ETL testing is done to identify data defects and errors that occur prior to processing of data for analytical reporting. ETL testing is mainly done using SQL scripts and manual comparison of gathered data. This approach is slow, resource intensive and error-prone. Automating ETL testing allows testing without any user intervention and supports automatic regression on old scripts after every new release. In this paper we will examine the challenges involved in ETL testing and subsequently need for automation in ETL testing to deliver better and quicker results.**

***Keywords*— ETL, Data Warehouse, Automation, SQL, regression, scripts.**

I. Introduction

In managing databases, extract, transform and load (ETL) refers to three separate functions combined into a single programming tool. Firstly, the extract function reads data from heterogeneous data sources and extracts a desired subset of data. Next, the transform function works with the acquired data - using rules or lookup tables, calculations, joining fields, keys, removing incorrect data fields, creating combinations with other data - to convert it to the desired state. Finally, the load function is used to write the resulting data; either all of the subset data or just the changes; to a target database, which may or may not be previously created.

ETL testing is done to ensure that the data that has been loaded from a source to the destination after business transformation is accurate. It also involves the verification of data at various middle stages between source and destination. ETL testing covers all the steps involved in an ETL lifecycle. It starts with understanding the business requirements till the generation of a summary report.

ETL testing is performed in following stages:

- Identifying data sources and validation of the business requirement.
- Data acquisition.
- Implement business logics and dimensional Modelling.
- Creating test scenarios and test cases.
- After pre-execution check, execute all the test-cases.
- Generate a complete summary report and file a closure process.[2]

One Might ask that how the ETL testing differs from database testing? Both ETL testing and database testing involve data validation, but they are not the same. ETL testing is normally performed on data in a data warehouse system, whereas database testing is commonly performed on OLTP (transactional systems) where the data comes from different applications into the transactional database. ETL testing emphasizes on data extraction, transformation and loading for BI reporting, while Database testing stresses more on data accuracy, validation and integration [1]. ETL Testing involves validation of data movement from the source to the target system, verification of data count in the source and the target system, verifying data extraction, transformation as per requirement and expectation, verifying if table relations are preserved during the transformation. On the other hand, Database testing involves verification of primary and foreign keys, validating data values in columns, verifying missing data and to check if there are null columns which actually should have a valid value.

However, in spite of the increased use and preference of ETL, ETL testing reflects the state of testing, which is too slow and too manual, and yet allows an unacceptably high amount of errors. As an alternative, an automation based approach is set out, considering how it might make ETL testing far more efficient, effective, and systematic[11].

II. Challenges faced by traditional ETL testing system

One may have to face different types of challenges while performing ETL testing which is different from database testing or any other conventional testing. We will narrow down the challenges by grouping them according to principle attributes of ETL testing cycle.

A) The Cardinal Issue of complexity & testability:

The issue behind manual validation is that ETL testing rapidly becomes highly complex. Probably the most obvious test is to regulate data completeness which checks whether all records have been loaded into the data warehouse or not. For some ETL's, this can be as easy as comparing a record count of the source and the target table. But unfortunately it's not always that easy, data might be transformed to a completely different and convoluted structure following some data modelling technique to maintain historical information or to improve reporting performance by storing aggregated results. As the business grows, and the variety and volume of data it accumulates increases, the ETL rules grow in order to handle it. In this rapidly this growth is happening faster than traditional testing methods can handle. In fact, the sheer amount of information being collected by data driven organizations has grown so fast that 90% of the data in the world was collected in the last two years alone [3], while the amount of data collected by the average organization is Doubling each year[4]. We need more sophisticated ETL tests to further ameliorate the complex issue.

B) The convoluting and misguided documentation:

The growing complexity is especially problematic for ETL testing, as the transformation rules are typically stored in poor documentation, lacking explicit expected results. The rules are often designed during the development phase itself, and are frequently stored in written documents or spreadsheets, they might not exist outside the developers and testers' minds. In this instance, no real documentation exists from which test cases can be confidently derived. ETL testers are normally not provided with access to see job schedules in the ETL tool. They hardly have access to BI Reporting tools to see the final layout of reports and data inside the reports. Sometimes the testers are not provided with the source-to-target mapping information [1]. Such incomplete, equivocal documentation makes difficult for testers to accurately understand the ETL routines. ETL testers normally don't have an idea of end-user report requirements and business flow of the information that why often times, testers were left filling in the blanks, but when they got it wrong, defects affect the ETL routines.

C) The Data complexity:

Data warehouse system contains historical data, hence it is tough to generate and build test cases, as data volume is too high and complex. This is another frequent cause of bottlenecks and defects. Manually deriving test cases and data from static requirements is highly unprecedented to change. Another major issue for ETL validation is the availability of data. Source data might be drawn from multitude of heterogeneous sources across an enterprise model. The another problem is that ETL testing is viewed as a series of additive (linear) stages, so that test teams are queued while another team accesses the data. If the data from every data across the enterprise is not available to every team in parallel, then delays will increase and will exacerbate the process aiming the team members inert.

D) Consistency, correctness and quality:

Given the complexity of ETL routines, combined with the poor documentation, it is unfair to expect tester to create every test correctly needed to validate the possible data combinations. Manual derivation can leads to massive under-testing and over-testing, where only a subset of the plausible logic involved in an ETL routine is tested. When incorrect, incomplete or invalid data is copied to the target system, it can jeopardize the integrity of the system. Sometimes a compromise is made, and only a subset of a whole dataset is validated. However, this also compromises the thoroughness of the ETL testing. Given the role of many ETL routines in business critical operations, such a compromise is unacceptable. Quality can be further affected when the expected results are not defined independently of the shadow code used in testing at all. In this instance, testers tend to presume that a test has passed, they are likely to assume that the actual result is the expected result [5], and so cannot confidently determine data validity.

E) The Constraint of Time:

In above mentioned challenge to maintain consistency, correctness of data, various roles need to be fulfilled to ensure integrity of data which in turn is immensely time and labour intensive. This time wasted on manual test case design is made worse by the time which then has to be spent comparing the actual and expected results. Performing data transformations is a bit complex, as it cannot be achieved by writing a single SQL query and then comparing the output with the target. For ETL Testing Data Transformation, you may have to write multiple SQL queries for each row to verify the transformation rules which is time consuming. Comparing the vast individual fields to the expected results is highly time-consuming, given the amount of data produced by a complex ETL test routine, and the fact that the source data will often be stored in multitude of heterogeneous database and file types.

III. Proposed system

As long as manual intervention in involved in ETL testing, one wouldn't be able to keep up with changing complexity of data. Below is a possible strategy for improving the efficiency and efficacy of ETL testing. It is a model-based, requirements based strategy, designed to automate the effort of testing, and build quality into the ETL lifecycle from the very beginning. Such an approach introduces automation across every stage of testing and development, while making ETL testing fully susceptible to constant change.

In the figure below is the outline of the proposed system, in which the arrows below test case generation indicate flow of data in parallel manner from synthetic data generation engine.

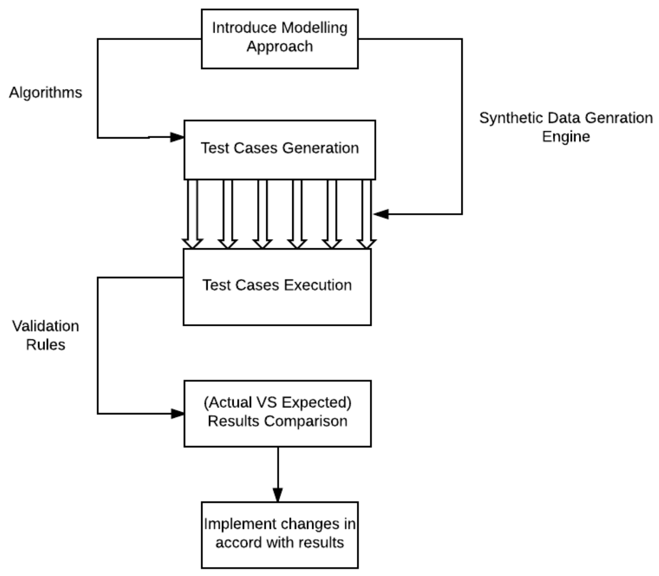
Introduce Modelling Approach
Algorithms
Synthetic Data Genration Engine
Test Cases Generation
Test Cases Execution
Validation Rules
(Actual VS Expected) Results Comparison
Implement changes in accord with results

FIG: Proposed System Architecture

Step I: Introduce Model based approach

Introducing modelling into ETL testing offers to maintain testability in case of the ever-growing complexity of ETL rules, so that testers can quickly understand and visualize exactly the logic which needs to be tested, as the modelling helps complex set of ETL rules to be consolidated into a single, visual diagram [12].

Consider Flowchart Modelling, Modelling ETL routines as a flowchart therefore eliminates ambiguity from the requirements documentation, working to avoid the 56% of defects which stem from it [6]. The flowchart serves as a single point of reference. In contrast to static written documents and diagrams, additional logic can be easily added or omitted to/from the model. In addition to reducing ambiguity, flowchart modelling also helps to combat incompleteness.

Another advantage is that expected results can be defined in the model, independent of the test cases. In flowchart modelling the user can define the model to include the boundary results and diffuse their expected result to various end points in the model. By setting boundary points, one can clearly defines what should be accepted and rejected by a validation rule, so that testers do not wrongly assume that tests have passed when the expected result is not stated explicitly.

Step II: Automatically Derive Test Cases from the Flowchart Model

The introduction of Model-Based Testing can automate one of the major and manual elements of ETL testing: design test case. Tester’s no longer need to manually copy SQL from the source to target database. Due to automation is various constraints are simplified. Automated mathematical algorithms can then be applied, to identify every possible path through the flowchart model, generating test cases which cover every set of inputs and outputs combinations.

A further advantage of this method is that testing becomes measurable. Because every possible test case can be derived, testers can determine exactly how much scope a given set of test cases provides. It also helps in maintaining the principle 'Test More in Fewer Tests'. Automated optimization algorithms help to reduce the number of test cases down to the bare minimum, while retaining maximum functionality coverage. These combinatorial techniques are made possible by virtue of the simplified structure of the flowchart.

Step III: Automatically Create the Data Needed to Execute the Tests and Provision of that data in parallel

Once test cases have been created, testers require data which can cover all of the possible tests in order to execute them. Such data can also be derived directly from the model itself. A synthetic data generation engine like CA Test Data Manager [6] offers multiple ways to create the required data when using Model-Based Testing to drive ETL testing. This is because, in addition to functional logic, a flowchart can also be overlaid with all the data involved in a system.

What’s critical for managing the time constraint for efficient ETL testing is that the 'Utmost dilapidated' data is stored efficiently, so that the same data sets can be requested, cloned and delivered in parallel. This, in turn eliminates the delays caused by data constraints. The first step to efficient data storage is the creation of a Test data Mart, where data is equalled to specific tests. Test matching data can eliminate the time otherwise spent searching for data among large production data source, as data can be retrieved automatically from the Test Data Warehouse; which serves as a central library, where data is stored as re-usable assets, alongside the associated queries needed to extract it.

Data can be fed into multiple systems simultaneously, and is cloned as it is provisioned. This means that data sets from numerous source databases are available to multiple teams in parallel. ETL testing is no longer a linear/additive process, and the long delays created by data constraints are removed. The original data can be maintained as changes are made to the model, enabling teams to work on multiple releases and versions in parallel.

Step IV: Validation of data against the rules and Automatically Compare the Results

Once testers have the tests needed to fully test an ETL routine, and the data needed to execute them, the validation process itself must also be automated if ETL testing is able to keep up with the changing requirements. Using Data-Driven Automation where a test harness created in a data orchestration engine [6] can take each row of an XML file, defined in the flowchart, and execute it as a test, meaning that the data, matched to specific tests and expected results, can be taken, and pumped through a validation rule. This automates both the execution of tests and the comparison of the actual versus expected results. Testers no longer have to manually copy scripts from the data target to the data source, and also avoid the error-prone process of having to compare every individual field produced by the transformation.

Step V: Automate the implementation of changes made

One of the advantages of Model-Based Testing for ETL validation is the ability to react to changes [12]. Because the tests cases, data and requirements are so closely linked, a change made to the model can be automatically reflected in the test cases and associated data. This traceability means that, as ETL routines rapidly grow more and more complex, testing can keep up, and does not create bottlenecks in the Application Delivery pipeline. With flowchart modelling, implementing a change becomes as quick and simple as adding a new block to the flowchart. As the test data is traced back to the model, when the requirements change, the changes are automatically reflected in the relevant data subsets. Data is available parallelly while the data needed for efficient regression testing is preserved in the Test Data Mart.

IV. Future Scope

Although many organizations still use ETL testing tools without automation, there are many alternatives available in market which automate the whole ETL testing process in more meticulous manner.

Datagaps provides ETL validator which comes with the most comprehensive data testing automation platform (DTAP) with features such as visual test case builder, metadata and FLAT files testing capability [7]. Another ETL testing tool that provides complete automation is QuerySurge. It provides creation of test QueryPairs fast & without writing any SQL with Query Wizards, it also schedules tests to run immediately, any date/time or automatically after event ends[8]. Another leading ETL testing automation solution is iCEDQ which comprises of rich features like Database and file connectors, in-memory rule engine and various test suites [9]. And many other automation services are available for ETL testing. One just have to choose the tool which accommodates its volume of data and provides tests according to business model.

V. Conclusion

Introducing a higher degree if automation into ETL testing is of dire importance for any organization striving for extensive data preparation and transformation. Automated ETL will provide a simplified and more transparent experience than manual coding – particularly when working with rapidly changing data for Delivery of high quality, further minimizing manual effort in ETL validation, from manually writing ghost code and static requirements, to sourcing the required data and comparing the results.

The Bottom Line is Coders will always have reservations about any tool that simplifies manual coding but the advantages overweigh the drawbacks [10]. Due to automation, ETL testing no longer creates bottleneck in application delivery, and can keep up with the pace with which data-driven businesses grow. In the end, automating the whole testing process will save a lot of time and users will appreciate the quality of Business Intelligence tools and accept the data from BI solution as the single version of the truth.

Acknowledgment

I would like to thank Tutorialspoint.com for providing in depth knowledge on ETL testing which helped me evaluate the pros and cons of ETL testing. Also I would like to thank the good folks at CA technology, their whitepaper on Automation helped me to study statistics, facts and models related to ETL testing and also it provided references to various articles on ETL testing, Surveys and data warehouse/Data Marts analogy.

References

[1] ETL testing tutorial. Available at https://www.tutorialspoint.com

[2] ETL Testing or Data Warehouse Testing Tutorial. Available at https://www.guru99.com/utlimate-guide-etl-datawarehouse-testing.html

[3] IBM, Available at http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html

[4] Jacek Becla and Daniel L.Wang, Lessons Learned from managing a Petabyte, Available at http://www.slac.stanford.edu/BFROOT/www/Public/Computing/Databases/proceedings/.

[5] Robin F. Goldsmith, Four Tips for Effective Software Testing. Available at http://searchsoftwarequality.techtarget.com

[6] White Paper on Fully Automated ETL Testing a Step-by-Step Guide. Available at https://www.ca.com/us/collateral/white-papers/fully-automated-etl-testing-a-step-by-step-guide.html

[7] Product Page of DATAGAPS ETL VALIDATOR. Available at http://www.datagaps.com/etl-testing-tools/etl-validator

[8] Automate the Testing Effort, product page of QuerySurge. Available at http://www.querysurge.com/business-challenges/automate-the-testing-effort

[9] Product Page of iCEDQ. Available at https://icedq.com/solutions/etl-testing

[10] Adi Azaria, THE CASE FOR AUTOMATED ETL VS MANUAL CODING. Available at https://www.sisense.com/blog/the-case-for-automated-etl-vs-manual-coding/

[11] Davy Knuysen, Automate your ETL testing and deliver quicker and better results. Available at http://www.element61.be/en/resource/automate-your-etl-testing-and-deliver-quicker-and-better-results

[12] Model Based Testing – Stuff You Must Know Available at https://www.guru99.com/model-based-testing-tutorial.html