A Data warehouse Testing Strategy for Banks

Donia Ali Hamed, Neveen ElGamal, Ehab Hassanein

Abstract: Data Quality, Database Testing, and ETL Testing are all different techniques for testing Data Warehouse Environment. Testing the data became very important as it should be guaranteed that the data is accurate for further manipulation and decision making. A lot of approaches and tools came up supporting and defining the test cases to be used, their functionality, and if they could be automated or not. The most trending approach was the automating of testing data warehouse using tools, the tools started firstly by supporting only the automation of running the scripts helping the developers to write the test case just once and run it multiple times, then the tools developed and modified to automate the creation of the testing scripts and offer their service as a complete application that supports the creation and running of the test cases claiming that the user can work without the need of expertise and high technicality and just by being an end user using the tool's GUI. Banking sector differs completely than any other industry, as data warehouse in banking sectors collects data from multiple sources and multiple branches with different data formats, and quality that should then be transformed and loaded in the data warehouse and classified into some data marts to be used in different dashboards and projects that depend on high quality and accurate data for further decision making and predictions. In this paper we propose a strategy for data warehouse testing, that automates all the test cases needed in banking environment

Keywords*: *Data warehouse testing, testing strategy.

I. INTRODUCTION

Data warehouse is responsible for collecting the data and storing it for further manipulation to support decision makers such as bank heads. They could decide the next service to be offered according to the transactions of the customers and the data stored in the data warehouse, or even detect money laundering suspicious acts, predict profits, amortizations, etc... Therefore, testing the data is vital and effective to guarantee the quality of data supporting the right decisions and deciding the future of any organization. Every organization considers their data as the most valuable and expensive asset, as it does not only tracks and organizes their work but also it supports and define their future curve. Data warehouse testing guarantees the quality of data used for reporting and decision making. Therefore there should be a well-planned testing strategy that supports all the teams and the phases through the work flow of data warehouse. Building a strategy does not depend only on running scripts and automating them. It should act as a structure covering everything affects the data and its rapid changes, validity, integrity, dependency etc…However, testing the data requires a lot of time, effort, and money. It needs experts on both technical and business fields to cover all the testing cases that could happen according to their experience that is only gained by trials and errors. Therefore, many tools are rising up trying to automate this job and a lot of researchers are developing solutions to ease the process. Banking environment is a rich example for data warehouse testing process, as it contains an enormous amount of data that is created from multiple branches and ATMs. The data itself is changing frequently on a daily basis and is integrated from different systems, and applications. Therefore accuracy is a must and no errors are accepted.In the next section, we will introduce the most famous testing tools in the marketplace, also we will introduce the latest techniques, frameworks and models applied by different authors for automating data warehouse testing. Then in section 3, a comprehensive comparison will be presented that covers all studied frameworks, methods, and testing tools to check their coverage for the banking environment challenges. In section 4, a criteria will be defined to specify the needs required in the banking environment and a proposed strategy will be presented along with the definition of the conceptual logic of all test cases needed. Then, in section 5 we will compare the results of applying the proposed strategy on a banking environment with the traditional method currently used. Finally in section 6, we will conclude our work and discuss some possible future works in the area of data warehouse testing.

II. LITERATURE REVIEW

There is no doubt , that many trends have been applied and many authors have spent a huge effort digging on testing area and trying to find out the best solutions and ways to be sure that the data is correct. There are two trends for testing data warehouse, the first trend authors are proposing the concepts of how to test data warehouse, what to test in data warehouse, and when to apply each test case scenario. The second trend focuses on automating the test cases for data warehouse testing aiming to offer the end user an experience to test their data without the need of being experts or learning code.Giving examples on the first trend, the author in [2] proposed a testing framework to automate testing data quality at the stage of ETL process. They sorted the test cases according to the quality parameters.

The following test cases applied in each quality parameter:

- Completeness: duplicate values, integrity constraints, out of boundaries value, SCD, record count validation

Manuscript published on November 30, 2019.
***** Correspondence Author

Donia Ali Hamed*, Faculty of Computers and Artificial Intelligence, Cairo University, Cairo, Egypt.
Neveen Elgamal, Faculty of Computers and Artificial Intelligence, Cairo University, Cairo, Egypt.
Ehab Hassanein, Faculty of Computers and Artificial Intelligence, Cairo University, Cairo, Egypt.

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

- Consistency: filed mapping , measure values,
- Uniqueness, duplicate values, integrity constraints
- validity :data type , field length
- Timeliness: data freshness
- Accuracy: truncated values, out of boundaries value

However, the proposed test routines did not satisfy the fulfillment of data quality completely.

Also, the author in [5] proposed an approach for testing the ETL process in Data Warehouse by firstly generating mapping documents (one to one or one to many) to structure the tables that needs to be tested then they applied the generated balancing tests and test assertions that automate the creation of the test cases. They categorized the test cases as follows:

- Completeness: record count match, distinct record count match
- Consistency: attribute value match, Attribute constraint match, outliers match, average match
- Syntactic validity: Attribute data type match, Attribute length match, Attribute boundary match

However, the covered test cases couldn't fulfill the required percentage of data quality coverage and the method used once you add any column it generates all the test cases mentioned even if not needed.

Other authors such as [10] relied on offering an approach for Regression testing, they defined regression testing as the test cases that should be re-executed frequently to build confidence that the up and running project maintains the same quality as in its pre-live cycle without being affected by any changes.

The authors created database tables, one for storing the test cases, other for capturing the result of each test case whether pass or fail and delta tables that are created for capturing unmatched records with the same structure of the table with fail result.

However, the proposed model covered the logging of test results for regression testing only.

In [3], Author Proposed multi perspective data warehouse testing framework (MPDWT) The authors started by defining the data warehouse architectures and built their approach with respect to the generic (kim-mon) architecture as it covers all possible stages / layers in order to cover all the approaches. The authors also introduced data warehouse testing coverage matrix that could be used to evaluate any testing approach. The matrix consists of:

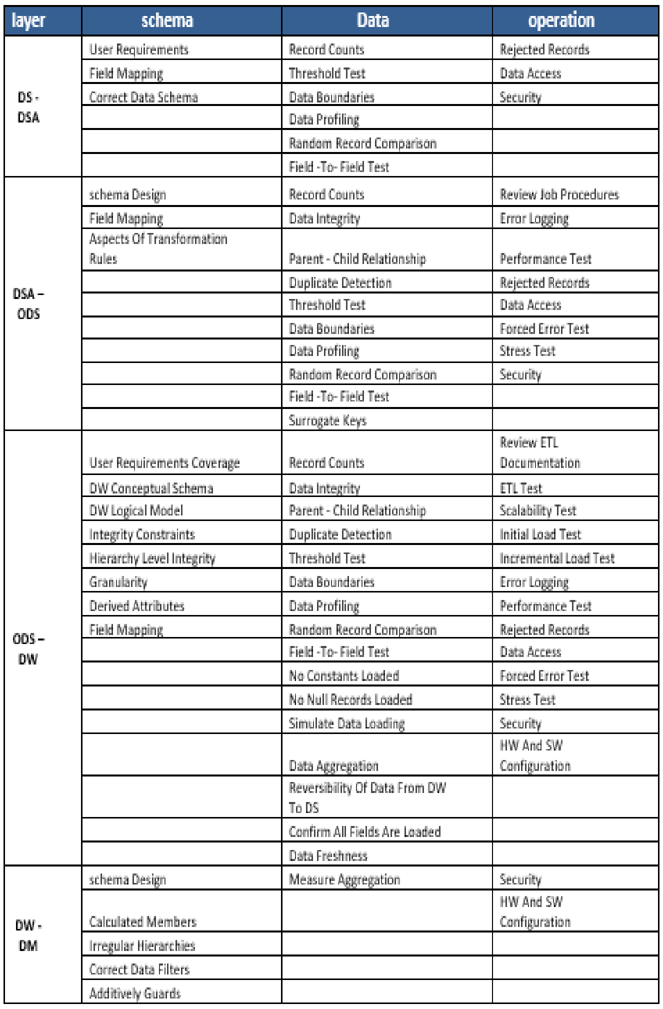
What: represents the test routines that will be tested in the targeted layer in DW component. Whether it is targeting the data or the schema (the structure of the tables) or the operation.

Where: represents which layer or stage in the DW that this test routine targets.

When: represents when this testing routine will take place before system delivery or after system delivery.

The following table represents the testing routines per each layer according to the author's generic framework in [3]:

Table- I: Test Routines in MPDWT Framework



layer	schema	Data	operation
DS - DSA	User Requirements	Record Counts	Rejected Records
	Field Mapping	Threshold Test	Data Access
	Correct Data Schema	Data Boundaries	Security
		Data Profiling	
		Random Record Comparison	
		Field -To- Field Test	
DSA – ODS	schema Design	Record Counts	Review Job Procedures
	Field Mapping	Data Integrity	Error Logging
	Aspects Of Transformation Rules	Parent - Child Relationship	Performance Test
		Duplicate Detection	Rejected Records
		Threshold Test	Data Access
		Data Boundaries	Forced Error Test
		Data Profiling	Stress Test
		Random Record Comparison	Security
		Field -To- Field Test	
		Surrogate Keys	
ODS – DW	User Requirements Coverage	Record Counts	Review ETL Documentation
	DW Conceptual Schema	Data Integrity	ETL Test
	DW Logical Model	Parent - Child Relationship	Scalability Test
	Integrity Constraints	Duplicate Detection	Initial Load Test
	Hierarchy Level Integrity	Threshold Test	Incremental Load Test
	Granularity	Data Boundaries	Error Logging
	Derived Attributes	Data Profiling	Performance Test
	Field Mapping	Random Record Comparison	Rejected Records
		Field -To- Field Test	Data Access
		No Constants Loaded	Forced Error Test
		No Null Records Loaded	Stress Test
		Simulate Data Loading	Security
		Data Aggregation	HW And SW Configuration
		Reversibility Of Data From DW To DS	
		Confirm All Fields Are Loaded	
		Data Freshness	
DW - DM	schema Design	Measure Aggregation	Security
	Calculated Members		HW And SW Configuration
	Irregular Hierarchies		
	Correct Data Filters		
	Additively Guards		

The same author in [15] presented a test routine description template describing the common name used for each test case in the testing field, in which layer should the test case be applied, what is tested whether the structure or the data, the objective of each test case and the logic to build it, its severity and periodicity.

This framework covered most of the layers of DW however, it didn't cover the BI layer or the operational phase.

For the second trend, we categorized the most famous data warehouse testing automating tools into two types:

- **Test case management tools**: such as TestRail or JIRA which is the most famous test case management tool as it organizes the test cases, assigns test cases for users and give complete status for each test case with comments , this category is very important as one of the major problems is the miscommunication between the testers and the developers. Not to forget that a lot of other tools claim that they are compatible with JIRA which shows how powerful is this tool.

- **Automated testing tools**: that could be classified into three types, the first type claims/provide automated creation of test cases, second types claims automating the running of test cases and the third type is a hybrid between covering the most test cases that they could automate the creation of their scripts and keeping the testing analyst with an option to write any test case scripts needed just once and then offering to automate the running of the customized script several times.

Firstly, Test case management tools: This category contains a lot of tools that are very trending and competing. However, we focused on the most famous tool of them that most of the automation tools are compatible with as our aim is to automate the scripts itself not only manage the QA process.

Testrail – (JIRA)

TestRail [12] is one of the most powerful test case management tools that tracks the status of the test scripts / cases and organizes the test management. You can update the results whether passed or failed and add comments. Managers could assign test cases to resources and follow the results on a real time insight. It is very easy to use and provides a dashboard for the results. In spite of how helpful this would be for tracking the QA department and creating a connection between teams but still not effective enough as the organization still needs experts and time for writing the testing scripts.

Secondly, automated testing tools: This category discusses the most trending data warehouse automation testing tools in marketplace, the test cases that they cover or claim to automate and all their competitive features.

Informatica Data Quality

In March2019 , Informatica maintained its 12-time position as a Leader in Gartner's Magic Quadrant for Data Quality Tools, this powerful and constant innovative tool considered number one in the marketplace with no competitors to be compared with, it integrates with the power center repository and integration services. Users can create and run mapping in Informatica data quality [7], the users also can use Pre-built rules that can be edited to suite their project objectives, then they can export the mappings for use in the power center. As a power center end user, you ca import the mappings created by Informatica data quality ar run them in power center sessions. Informatica power center transforms your mappings into mapplets if you imported a mapping that contains one or more transformations. Each transformation is converted to a mapplet that contains the expanded configuration.

Informatica Data Validation Tool offers a speed up automated test coverage for both production and development delivering a repeatable, editable test cases in minimum time with no programming skills required.

Also not to forget to mention Data Quality Analyst Tool that allows the business analyst and the managers to take ownership of data quality process. Teams can profile data and perform interactive data analysis with the flexibility to filter and drill down specific records to detect problems better. Business users can read, monitor and share data quality metrics with scorecards and reports by emailing to the team. This means that business stakeholders can participate in and provide the relevant business context for improving the quality of the data.

The QA team will be able to schedule a rule to run and share rules between Informatica data quality and Informatica power center. To improve the process of analyzing, profiling, validating and cleansing the data.

Qualidi- Bitwise

QualiDI [8] ETL Testing Automation tool provides a platform for centralizing testing of one or more ETL tools. QualiDI automates ETL testing from source (any operational system) to target (data warehouse) its main focus is in the quality of data integration through ETL process. It also automate batches for regression testing and re – testing , user can manage the test cycle through dashboards and reports, and even schedule the running of the test cases.

It depends on the mapping document loaded and the correct SQL transformation rules written in that mapping document. And it transfers this transformation rules into set of queries on source table and on target table to check the data quality then visualize the results on dashboard and email the reports to the developers.

Zuzena

Zuzena [14] is an auto-run, schedule-driven data warehouse testing engine which is designed to support agile data warehousing and business intelligence teams.

It can be deployed as a hosted service in hours, or a fully configured and ready to run on a server. It allows agile team to immediately begin enforcing 100% test coverage of all mission-critical ETL routines designed by the team.

Zuzena differs than test case management tools because, the user configures the required test case and configures all the connections that is uses to iterate the defined scenarios and keep them in its staging area for auto managing and running however, test case management tools just view the status of the test cases.

Zuzena is built from open source components, as it integrates into UNIX, Linux, and Microsoft environments and connects to all the SQL database management systems (DBMS).

The user needs to define high level items using its configuration screens, these items are:

- the scenarios that should be tested
- the location of each source data for each scenario
- the location of each ETL workflows to be called
- and finally, the location of the actual and expected results (destination)

After that, Zuzena will automatically iterates through the defined scenarios and repeat them in the staging data area, execute the appropriate ETL, and compare the expected outcomes to the actual results. The detailed results for each run are all archived in a test result repository.

Etl Validator – Datagaps

Data Gaps [4] provides two products that are used for data testing automation ETL validator and BI Validator.

ETL Validator is a data testing tool that simplifies data integration, data warehouse and data migration projects. The most known feature in this tool is the visualization feature as it represents all the test cases in dashboards. ETL validator covers these data integration test cases, as it auto generates and auto run them.

BI validator is the most comprehensive functional, regression, performance, and stress testing tool for business intelligence platforms such as OBIEE, Tableau, etc… It is easy to use and empowers users to create and execute test cases quickly without the need to do any custom programming.

Querysurge

Querysurge [9] is one of the automation tools for data warehouse testing, ETL testing, data migration testing and BI testing.

It is a smart tool for automating and validating the testing of data warehouses. It supports auto generation of some testing cases using the query wizards and still allows power users to write down their own custom code.

Querysurge is designed to be very easy and unique, it supports a huge amount of data stores, allows you to schedule the test scripts, and extracts detailed reports for the failures with email option.

ICEDQ

ICEDQ [6] is an automated software that organizations can use to automate data warehouse testing, data migration testing, database testing, integration testing, and monitor production data issues. Their slogan is “you develop we test, you deploy we monitor”.

ICEDQ testing platform is designed to identify any data issues in structured and semi-structured data. The engine connects with the configured data sources, identifies matching records and missing records, validate the transformations and calculations based on the defined groovy expression, and reports / captures all the missing records and failures in an exception report allowing the users to analyze the issues and take actions according to the results.

Datamartist

Datamartist [1] is a data profiling and transformation tool. Its way differs than any other tool as it helps you to drag and drop data from anywhere to be compared with excel sheets or relational databases. You can easily import data into Datamartist and even you can update the SQL statement of selecting the table for further conditions and filtering to be compared with other tables or reports inform of excel sheets as an example.

It is very easy to use and very simple as it gives the user the insight of data inform of statistics that you can write your own transformation rule using the calculate block and add your own calculating column then check the value distribution explorer on the transformed column to find out the results in a form of graphical statistics. What is amazing is that you can drag across and select any amount of the graph to return back to the columns resulting the error and check them. It is more like checking the data by eye but its helping as it points out the existence of an error in form of graphs. It might be great for checking reports with the main data but its tolerance percentage of finding out the errors is very high. It also saves a sample of data at the creation and this sample needs to be refreshed every time you use the table in the tool.

Tricentis - TOSCA

TOSCA [12] Tricentis is a testing platform that won the most innovative company, it embeds the test cases with in the software development life cycle in order to detect and resolve defects faster and increase the code coverage and ensures the quality of each milestone. It is also compatible with JIRA case management tool to improve the communication with developers in real time and to be notified as soon as there is a new code to test or a defect is found and needs fixation.

Talend

Talend [11] is an open studio software integration platform / vendor which makes ETL testing easier, it offers various data integration and data management solutions. Talend includes all ETL testing functionalities, with the help of this tool the user can run ETL jobs on remote servers with variety of operating systems. It ensures that data is transformed from the source system to the target without any data loss.

Talend is developed on top of the eclipse graphical development environment, the user can easily map the data between the source and the target with simple drag and drop. It avails strong connectivity, easy adaptability, and smooth flow of extraction and transformation processes. The user can write java code and java scripts for better performance and results.

III. CHALLENGES AND ANALYSIS

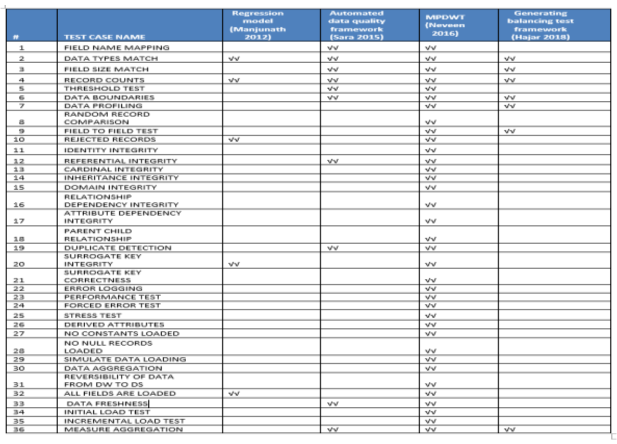
In banking environment critical issues may happen such as:

- Testing issues, covering all the test cases and checks for multiple projects and sources, which needs a huge effort and experienced resources and a lot of time.
- BI Issues, regarding the reconciliation of the dashboard statistics compared with the business team as there might be a gap on the results however, the ETL developed was completely correct.
- Operational issues, operation team that is responsible for running the daily flow of data for different sources and monitor the loading window of the projects faces loading issues as they might find the job ran on the integration tool successfully however zero records inserted or the job is taking too much time however the task has been already done or the session has been cancelled.
- Storage issues, as the huge load of amount of data from multiple sources that cannot be estimated suddenly becomes full, stopping all the flow on the server.
- Live projects that are up and running that business teams are completely depending on their results claiming that there was some codes missing or requested data not found which leads to losing the credibility of the project and its quality rather than its first initiation.

To cover all these issues we started looking for solutions in the frameworks, methods and tools we discovered. So, we started discovering all the test cases mentioned by the authors to solve our main issues in banking environment, the following table is comparing the test cases covered by the already mentioned frameworks and methods and that could be automated from authors' point of view.

As shown on in Table II, the most framework covering data warehouse test cases is MPDWT and the other frameworks or methods are focusing either on regression testing or quality testing. MPDWT framework focused on covering all the concepts and all the stages of data warehouse. However, the other frameworks focused on covering the test cases that could be automated. Yet MPDWT is not covering all the challenges in the banking environment and only proposing the logic of automating the test cases.

Table-II: comparing test cases of frameworks and methods

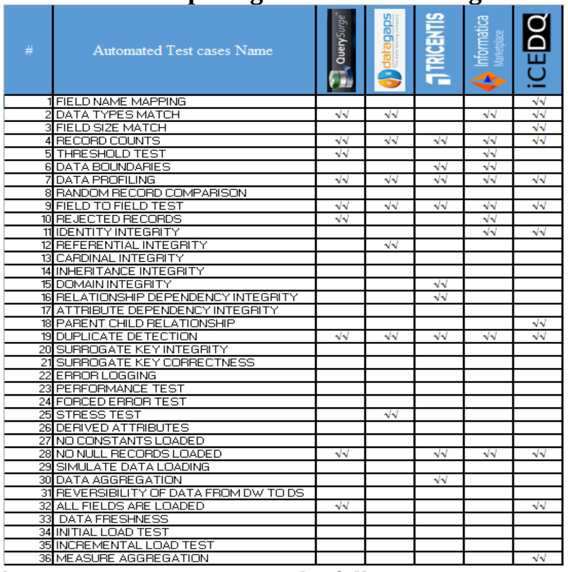


#	TEST CASE NAME	Regression model (Manjunath 2012)	Automated data quality framework (Sara 2015)	MPDWT (Neveen 2016)	Generating balancing test framework (Hajar 2018)
1	FIELD NAME MAPPING		√√	√√	
2	DATA TYPES MATCH	√√	√√	√√	√√
3	FIELD SIZE MATCH		√√	√√	√√
4	RECORD COUNTS	√√	√√	√√	√√
5	THRESHOLD TEST		√√	√√	
6	DATA BOUNDARIES		√√	√√	√√
7	DATA PROFILING			√√	√√
8	RANDOM RECORD COMPARISON			√√	
9	FIELD TO FIELD TEST			√√	√√
10	REJECTED RECORDS	√√		√√	
11	IDENTITY INTEGRITY			√√	
12	REFERENTIAL INTEGRITY		√√	√√	
13	CARDINAL INTEGRITY			√√	
14	INHERITANCE INTEGRITY			√√	
15	DOMAIN INTEGRITY			√√	
16	RELATIONSHIP DEPENDENCY INTEGRITY			√√	
17	ATTRIBUTE DEPENDENCY INTEGRITY			√√	
18	PARENT CHILD RELATIONSHIP			√√	
19	DUPLICATE DETECTION		√√	√√	
20	SURROGATE KEY INTEGRITY	√√		√√	
21	SURROGATE KEY CORRECTNESS			√√	
22	ERROR LOGGING			√√	
23	PERFORMANCE TEST			√√	
24	FORCED ERROR TEST			√√	
25	STRESS TEST			√√	
26	DERIVED ATTRIBUTES			√√	
27	NO CONSTANTS LOADED			√√	
28	NO NULL RECORDS LOADED			√√	
29	SIMULATE DATA LOADING			√√	
30	DATA AGGREGATION			√√	
31	REVERSIBILITY OF DATA FROM DW TO DS			√√	
32	ALL FIELDS ARE LOADED	√√		√√	
33	DATA FRESHNESS		√√	√√	
34	INITIAL LOAD TEST			√√	
35	INCREMENTAL LOAD TEST			√√	
36	MEASURE AGGREGATION		√√	√√	√√

For that reason we needed to compare the automating testing tools and the test cases they cover as in table III.

In table III, we compared the most powerful tools and we found the most test cases covered by all testing tools , yet couldn't solve all the challenges in the banking environment and we also found that there is no generic name for the test cases and multiple naming conventions might mean the same functionality.

Table-III: Comparing automation testing tools



#	Automated Test cases Name	QuerySurge	datagaps	TRICENTIS	Informatica Marketplace	iCEDQ
1	FIELD NAME MAPPING					√√
2	DATA TYPES MATCH	√√	√√		√√	√√
3	FIELD SIZE MATCH					√√
4	RECORD COUNTS	√√	√√	√√	√√	√√
5	THRESHOLD TEST	√√			√√	
6	DATA BOUNDARIES			√√	√√	
7	DATA PROFILING	√√	√√	√√	√√	√√
8	RANDOM RECORD COMPARISON					
9	FIELD TO FIELD TEST	√√	√√	√√	√√	√√
10	REJECTED RECORDS	√√			√√	
11	IDENTITY INTEGRITY				√√	√√
12	REFERENTIAL INTEGRITY		√√			
13	CARDINAL INTEGRITY					
14	INHERITANCE INTEGRITY					
15	DOMAIN INTEGRITY			√√		
16	RELATIONSHIP DEPENDENCY INTEGRITY			√√		
17	ATTRIBUTE DEPENDENCY INTEGRITY					
18	PARENT CHILD RELATIONSHIP					√√
19	DUPLICATE DETECTION	√√	√√	√√	√√	√√
20	SURROGATE KEY INTEGRITY					
21	SURROGATE KEY CORRECTNESS					
22	ERROR LOGGING					
23	PERFORMANCE TEST					
24	FORCED ERROR TEST					
25	STRESS TEST		√√			
26	DERIVED ATTRIBUTES					
27	NO CONSTANTS LOADED					
28	NO NULL RECORDS LOADED	√√		√√	√√	√√
29	SIMULATE DATA LOADING					
30	DATA AGGREGATION			√√		
31	REVERSIBILITY OF DATA FROM DW TO DS					
32	ALL FIELDS ARE LOADED	√√				√√
33	DATA FRESHNESS					
34	INITIAL LOAD TEST					
35	INCREMENTAL LOAD TEST					
36	MEASURE AGGREGATION					√√

Banking Environment requires the following:

- A full automated test cases scripts covering all of the ETL process
- Test cases covering the gap between the BI developer and the ETL developer instead of using the manual approach
- An eye on the storage capacity of the servers to eliminate sudden dropdowns.
- Test cases that could cover the bugs that appears in the data integration tools.
- Test cases that should always cover the up and running projects and maintain its quality.
- A well planned strategy that covers all the phases of data warehouse department in banking environment that integrates the results of all the test cases and log them to be visualized by the managers and the developers.

Accordingly, in the next Section we will propose a Data warehouse testing strategy that is adapted for banking environment and we will mention the test cases need and their logic to be applied and automated.

IV. PROPOSED STRATEGY

There should be a Strategy that plans what to do in such cases and how to act. It should be covering every process happening in data warehouse banking environment to guarantee having automated, well tested accurate data.

In figure 1, we will explain briefly the full cycle testing and logging strategy (FCTL) applied for a banking sector.

As shown, the Flow of data in Banking Environment available, starting from the different data sources then loading the data to data warehouse using integration tools that could automate some jobs and depend on users on the others, the data warehouse might include different data marts and may be contained on just one server of several ones. And finally we use the data from data warehouse for projects and dashboards.

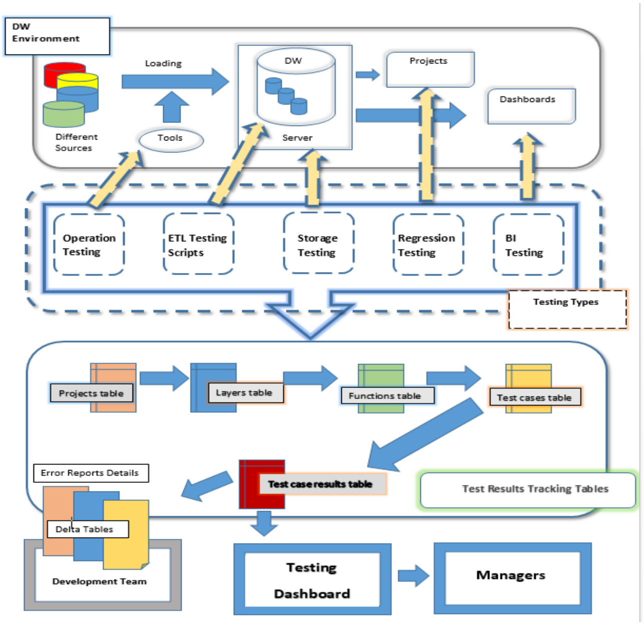


Fig.1.The FCTL Strategy

After explaining the flow of data in the Data Warehouse Environment you will find the testing strategy proposed. it's obvious that we classify our proposed testing strategy components into five major types, these types could be increased or decreased according to the need of the data warehouse environment , we will define each type as follows:

ETL Testing

Contains all the testing cases that could be applied during the development life cycle, guaranteeing the validation of the flow and the accuracy of the data loaded to the data warehouse. As shown on the diagram it is applied on the data inserting the data warehouse

BI Testing

BI Testing definition differs from one author to another and even differs from one tool to another, actually most of the test cases have multiple names with the same functionality which is really one of the major problems in data testing as every author or tool defines their own test cases from their own perspective. For example, ETL Validator tool contains BI testing however it actually applies regression testing, stress testing (checking the performance after increasing the number of users to the dashboard) and functional testing (checking the access privileges across environments)

The manual approach for BI testing used to be as follows:

- Validate the report manually by looking at the report and doing some calculations using the calculator. Which is very difficult because the data maybe a huge number of rows.
- Reconcile the data visually by comparing the data in the report and then manually checking in the database tables.

However, in our case we mean by BI Testing is checking that the logic created by the BI developer is reflecting the required business logic requested by the business team. This process is done by extracting the RBD file and applying some test cases that compares the logic applied by the BI developer while the creation of the aggregates is same as the prepared by the ETL developer and required by the Business Team.

Regression Testing

Regression Testing could be defined as the testing checks that could be applied after the project is up and running to guarantee the quality of the project is same as going live.

The manual approach was waiting to receive and issue regarding the quality of the data of the live project then start to see where is the problem to solve the issues which caused the loss of credibility of business team and of course time and money.

By planning a strategy that automates the test cases that should be re-executed frequently post going live this solves the issue and guarantees the quality of live projects.

Operation Testing

Defined as all the checks that should be applied checking after any automatic tool (Ex. Oracle Data Integrator ODI, Informatica Power Center, Data Stage etc…) load, this type of testing ensures that any bug of the integration tool is solved correctly as it covers the role of the operation support team (testing the integration tool) the major aim of this kind of testing is:

- Checking the dependencies between the automated running jobs
- Identifying the users that started a certain job
- Checking the validity of successful insertions of new data and flagging any problems happened or wrong loads (Ex. Zero records inserted and successful mark by the tool)

Even checking the statistics of each job loading window, this test is applied using the metadata of any integration tool used by the organization as each tool saves its own metadata describing the running jobs and their time of execution with the results (number of insertions, or error messages) creating a full structured environment for the tester to work on.

The manual approach for solving this issue also was after the flow of production batches already finished. The operation team claims finishing their tasks however, the data is missing in the tables or incorrect so we start searching for the reason and find out that it is a bug in the data integration tool and then restart the batches manually.

By planning the test cases that should be automated and applied, we guarantee solving the issues without the need of technical support team and before sending the data to production.

Storage Testing

Storage Testing aims to check the servers storage and send back an alert once the server is about to be full in order to request more storage earlier and avoid table space issues.

The manual approach was sudden downtime in the servers and all the running jobs are failed with the same reason Temp Table Space to find out that the data warehouse server is out of storage and then start the procedures of requesting extra storage and then restart the jobs.

Then all these testing types log their results into test case results tracking tables to be easily presented in dashboards for managers and the rejected tests or the failed ones will be redirected for the developers.

The following tables describes each testing type and the test cases per each:

Table- IV: ETL Strategy Test Cases

type	test case	test case definition
ETL	field name mapping	the field names between the data source and the destination attributes are matching
	data types match	the data types between the data source and the destination attributes are matching
	field size match	the field sizes between the data source and the destination attributes are matching
	record counts	the number of records in the source matches the number of records in the target
	field to field test	checks the values in the source matches the values in the target
	rejected records	checks that the rejected records are logged separately and that the job is rejecting the correct excluded criteria
	duplicate detection	confirms that there is no duplicate records existing in the destination table
	stress testing	confirm that the procedures work efficiently given an extraordinary workload

	SCD testing	checking the slowly changing dimension is applied correctly
	no null records	ensure that there is no null records in the destination table
	Trimming testing	ensuring that the data is transformed to the destination table without being trimmed of having added spaces
	Data Freshness testing	confirms that all the tables loaded with the specified fresh date as they should be and after the end of day batch
	Data profiling testing	Utilizing different descriptive statistics like min, max, avg, sum, mean, mode, etc…
	First load testing	ensures that the first load is correct and within the expected performance
	Incremental Load	ensures that the incremental load is correct and within the expected performance
	Case Sensitive	checks that the values especially char codes reading from lookups are correct and not received by case sensitive formats
	Performance testing	ensure the efficiency of the ETL procedures , confirm that the processing time is within the acceptable loading window
	surrogate key testing	verify the correctness of the creation of the surrogate keys
	Identity Integrity	checking that there is no more than one entity having the same primary key and null valued keys
	Referential Integrity	to detect foreign key value that does not exist as a primary key in the relating table

Table-V: BI Strategy Test Cases

type	test case	test case definition
BI	Data profiling testing	Utilizing different descriptive statistics like min, max, avg, sum, mean, mode, etc...
	Transformation rules testing	checking that the transformation rules are applied correctly
	Data Freshness testing	checking the date is as it should be
	record counts	checking that the count is applied as requested (distinct count , count)

Table- VI: Storage Strategy Test Cases

type	test case	test case definition
Storage	storage capacity	checking the available percentage of storage used

Table- VII: Regression Strategy Test Cases

type	test case	test case definition
Regression	Referential Integrity	to detect foreign key value that does not exist as a primary key in the relating table
	record counts	the number of records in the source matches the number of records in the target
	Trimming testing	ensuring that the data is not trimmed or having any added spaces
	no null records	ensure that there is no null records in the destination table
	Case Sensitive testing	checks that the values especially char codes reading from lookups are correct and not received by case sensitive formats
	rejected records	checks that the rejected records are logged separately and that the job is rejecting the correct excluded criteria
	surrogate key testing	verify the correctness of the creation of the surrogate keys
	Date format	checks the correct date format and ensures that the date is seen correctly
	Performance testing	ensure the efficiency of the ETL procedures , confirm that the processing time is within the acceptable loading window

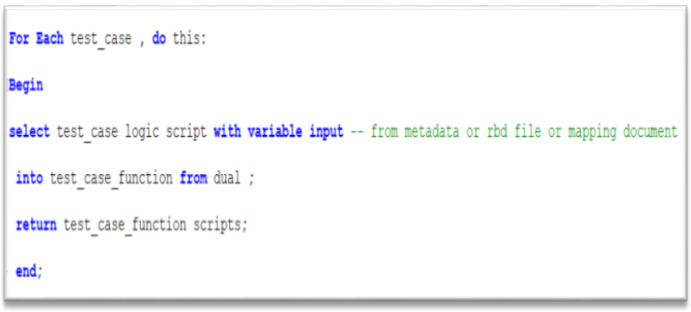
Table- VIII: Operation Strategy Test Cases

type	test case	test case definition
operation	Dependency testing	ensuring the dependency of the jobs (no job starts before confirming that the dependent job has ended successfully)
	Insertion Testing	ensuring that the job worked correctly and inserted data
	Creation testing	ensures whether the table is created correctly or the session has been killed
	User logging	checking whether the job started automatically or re-started manually by a user

As mentioned above, we defined all suitable test cases for each type to solve the issues that we face in banking environment.
Then we started by generating the concepts of each test case and created SQL scripts that achieve the required goal/objective in each test case to automate them.
Then we created PLSQL functions in order to loop multiple variables and auto generated multiple scripts at once. Also, to ease the calling of the scripts and the scheduling of them.
Figure 2 is describing the logic used for auto generated the test cases mentioned above.

V. CASE STUDY

ABC bank is using the traditional way of testing the data warehouse, a team of 5 resources testing 5 source systems flow on a daily basis.
The first source is the core banking system and it contains 108 tables, the second source system contains only 30 tables as it covers the credit cards service, the third source covers the mobile payment service and contains 10 tables, the fourth source system covers the prepaid cards and consists of 22 tables and the fifth source system covers all the trades and consists of 61 tables.
After applying the automated test cases using the logic represented in figure 2 and checking the differences between the traditional ways used versus the automatic testing, we found the following:



```
For Each test_case , do this:

Begin

select test_case logic script with variable input -- from metadata or rbd file or mapping document

 into test_case_function from dual ;

 return test_case_function scripts;

end;
```

Fig. 2. : Logic for implementing auto generated test cases

As per the statistics in figure 3, source one working hours were reduced by average 89 percent than the traditional way.

The average working hours for source two and source three were also reduced by 79 percent.
And for source number four the working hours were reduced by 75 percent. Source five were reduced by 81 percent.

source system	percentage reduced
Source 1	89%
Source 2	79%
Source 3	79%
Source 4	75%
source 5	81%

Fig. 3. : Percentage of reduced working hours per each source system

Figure 4 shows exactly the huge difference in cost of time and effort comparing the traditional way used to test data warehouse sources versus the automated testing strategy applied.

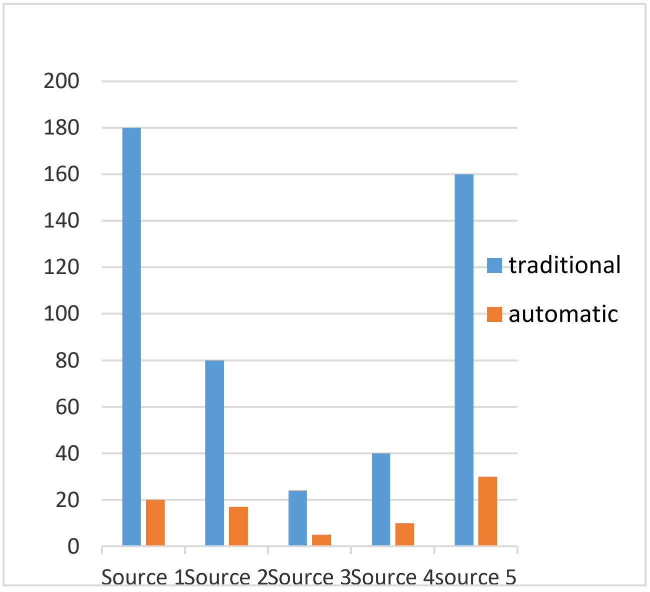


Fig. 4. Traditional Vs. Automatic Testing

Even the support team working hours were reduced 50%, which means reducing their working hours to the half from for example 8 hours to just 4 hours.

VI. CONCLUSION

In this paper, we mentioned the challenges that occur in the banking sectors, mentioned the test cases that could cover these challenges, then we compared between our test cases and the approaches proposed by different authors to check the gap and also compared the gap between our chosen test cases and the automated tools in the marketplace. Finally we proposed the full cycle testing and logging strategy FCTL.

After proposing the previous Strategy (FCTL), we concluded the following:

- Most automated tools are only covering data quality test cases, applying only very simple and not enough automated testing scripts, claiming that for more complex scripts you should write the code and only automating the execution afterwards.
- The naming conventions regarding each tool for the test cases is not unified and depends on the tools own perspective which makes it difficult to be collected or found and might contain the same functionality with multiple names.
- We studied all the approaches proposed by different authors to automate the data warehouse testing and compared them with our strategy.
- We studied the most famous and powerful automated testing tools and compared them with our proposed strategy.

The proposed strategy covered the following:

- Automation of the test cases needed to support the banking environment, Reducing the cost and time and not even needing experts
- Adding new phases for the testing strategy such as BI testing, Storage Testing, Operation testing and Regression testing to cover all the missing test cases.
- Tracking all the results and provided automation logging for all the auto generated test cases.

Tracking the failed tests with their detailed results and auto logged them for re-execution.

REFERENCES

1. Datamartist. Retrieved From (2019, September): Http://Www.Datamartist.Com/
2. Eldakrory, S. B., Mahmoud, T. M., & Ali, A. A. (2015, DEC). Automated ETL Testing On The Data Quality Of A Data Warehouse. International Journal Of Computer Applications, 131, 0975-8887.
3. Elgamal, N., El-Bastawissy, A., & Galal-Edeen, G. (2016). An Architecture-Oriented Data Warehouse Testing Approach. International Conference On Management Of Data (COMAD), (Pp. 24-34). Pune.
4. Etl Validator. Retrieved From Datagaps (2019, September) : Https://Www.Datagaps.Com/Data-Testing-Concepts/
5. Homayouni, H., Ghosh, S., & Ray, I. (2018). Generating Balancing Tests For Validating The ETL Process In DWH Systems. 11th IEEE Conference Of Software Testing, Validation And Verification.
6. ICEDQ. Retrieved From (2019, September): Https://Icedq.Com/
7. Informatica Data Quality. (Informatica) Retrieved From Https://Www.Informatica.Com/Products/Data-Quality/Informatica-Data-Quality.Html#Fbid=I1nfazjdr9o
8. Qualidi.. Retrieved From Bitwise(2019, September): Https://Www.Bitwiseglobal.Com/Innovations/Qualidi/
9. Querysurge. Retrieved From(2019, September) Https://Www.Querysurge.Com/
10. T.N., M., S.Hegadi, R., H.K., Y., R.A., A., & I.M., U. (2012, DEC). A Case Sutdy On Regression Test Automation For Data Warehouse Quality Assurance. International Journal Of Information Technology And Knowledge Management, 5, 239 - 243.
11. Talend. (2005). Retrieved From (2019, September): Https://Www.Talend.Com/Resources/What-Is-Etl/
12. Testrail. (2004). (Gurock Software) Retrieved From JIRA: Https://Www.Gurock.Com/Testrail?Utm_Source=Adwords&Utm_Medium=Cpc&Utm_Campaign=Asia_Afr_Ams_En_Brand&Utm_Content=Testrail&Gclid=Eaiaiqobchmi3l-Nsiuu5qivbvlrch3nyarleaayasaeglcc_D_Bwe
13. Tricentis. Retrieved From Tosca(2019, September): Https://Www.Tricentis.Com/Solutions/Testers/
14. Zuzena. Retrieved From(2019, September): Http://Zu.Socialtheories.Info/
15. Elgamal, Neveen. (2015). Data Warehouse Testing. 10.13140/RG.2.1.2048.7844.

AUTHORS PROFILE

Donia Ali Hamed*, Data Warehouse Specialist, and testing analyst in National Bank of Egypt. Masters Student in the Faculty of Computers and Artificial Intelligence, Cairo University. Email: doniaali92@gmail.com

Neveen Elgamal, Assistant Professor, Faculty of Computers and Artificial Intelligence, Cairo University. 11 research items, Email: n.elgamal@fci-cu.edu.eg

Ehab Hassanein, Chairman, Faculty of Computers and Artificial Intelligence, Cairo University.21 research items, Email: ehabez1@gmail.com