Business Intelligence / Data Warehouse ETL Testing: Best Practices

By:

Anand Singh
email: anand.x.singh@accenture.com
Phone: +91 96633 79871

Accenture Services Pvt. Ltd.,
Divyasree Tech Park, Sy 36/2,
Kundalahalli Village, Whitefield,
Bangalore – 560066
Karnataka
Abstract

Testing the so called Business "Intelligence", also involves some extra-ordinary "Intelligence". Testing here is designed to test and measure the ability of a business system to reason in a business or commercial occupational setting rather than demonstrating mental and intellectual agility alone. But internally, it all comes down to the testing of real data in the warehouses which is a manifold process in itself, covering the ETL part.

The biggest myth is that ETL testing is just verifying and validating the data that flows through the process, and just passing the test scripts. On the reality part instead, time after time we come across elusive requirements or business rules when confronted against actual production data, which can never be met.

Moreover, the data warehouse systems, which are specifically meant for decision making, are meant to change with changing requirements with course of time. This white paper shares some of the best practices inferred from experiences of testing the data in a data warehouse.

- The paper describes the milestones and flow of testing process which can help a data warehousing ETL testing process to be more efficient and less time consuming.

- The paper proposes the best practices of how and when the tests should be performed for a data warehouse testing to be qualitative and quantitative.

- The best practices help in outlining better and more appropriate test plans and test strategies for any data warehouse testing projects.

Business Intelligence / Data Warehousing: ETL Testing

Innumerable BI / Data Warehousing testing projects can be conjectured to be divided into ETL (Extract – Transform - Load) testing and henceforth the report testing.

Extract Transform Load is the process to enable businesses to consolidate their data while moving it from place to place (i.e.) moving data from source systems into the data warehouse. The data can arrive from any source.
• **Extract** - The process of reading data from a source file.
• **Transform** - The process of converting the extracted data from its previous form into the form it needs to be in so that it can be placed into another database. Transformation occurs by using rules or lookup tables or by combining the data with other data.
• **Load** - The process of writing the data into the target database.

The ETL part of the testing mainly deals with how, when, from where and what data we carry in our data warehouse from which the final reports are supposed to be generated. Thus, ETL testing spreads across all and each stage of the data flow in the warehouse starting from the source databases to the final target warehouse. The data-flow might also include temporary staging databases or ODS (Operational Data Sources).

The process of populating the data warehouse with data is the most important aspect of a BI/DW implementation, and testing will focus upon the validation of the data. The objective is to ensure that the data in the warehouse is accurate, consistent, and complete in each subject area and across each layer.

Data validation answers the following questions:

- Is all qualified data from the source being loaded into the warehouse?
• Are the business rules readily apparent in the data?
• Are all data relationships present and correctly joined across layers and subject areas?
• Will query results be consistent?

There are at least two approaches to data validation that can be considered:

- **Approach I**: follow the data from the data sources directly to the data warehouse. This approach validates that the data in the source data stores appears in the data warehouse according to the business rules.

- **Approach II**: follow the data from the data sources through each step in the ETL process and into the data warehouse. Validate the data at each transformation:
  - Source data stores to staging tables
  - If an ODS is used, staging tables to the ODS
  - Staging tables or ODS to the warehouse tables.

Available resources and established timelines may drive the approach that is used for the validation process. Approach I can take less time to script and execute. However, since this approach does not offer logical validation points; if issues are uncovered, it will be more difficult and time consuming to determine their origin. If time and resources are available, Approach II is the more comprehensive practice and should be applied. This approach will enable to more easily determine when and what data has been lost or incorrectly manipulated.

In validating each step of the ETL process, testing confirms that data is being moved and transformed as intended from source to target. Each phase of testing will include tasks to confirm data at the field level. Tasks are also included to reconcile data from the previous steps through record counts, reasonability checks, and basic referential integrity checks.

Specifically, testing will focus on the following risk points in the process:

- Extraction of source data
- Including, Change data capture
- Population of staging tables
- Transformation (including cleansing) of data from staging tables to ODS and to data warehouse tables

**ETL Testing: Best Practice Flowchart**

Following is a testing flowchart which describes the milestones and the flow of testing process which can help an ETL testing process to be more efficient and less time-consuming.

Putting into place the various prepositions of testing a dimension or a fact in an ETL process like matching the count of unique records in database or column level data validations or matching the source refresh to the extracted data etc., it actually proposes how and when these tests should be performed for testing to be qualitative and quantitative. Not having the process in place can prove the overall efforts to be time-consuming as well as redundant in many cases.
**Benefits of the flowchart:-**

- Outlines all tests in a progressive manner as to which type should be carried in which condition.
- **Time-saver**: Helps in saving both efforts and time that are required for exhaustive testing of ETL process.
- If the flowchart pertaining to the ETL process is outlined early in testing, it can help in better and more appropriate test plans & test strategies for any data-warehousing testing project.

---

*Figure 1.2*  
ETL Testing Best Practice Flowchart
The process:-

1. The process of ETL starts by matching the count of records from the Source database with those in the first staging database. This is the first step in ETL testing because if all records are not flowing into the first staging database, they won’t flow further and hence there’s no point testing anything further. Also, the source refreshes regularly, so it should be made sure that whenever the count is being taken, it should be taken till the latest source refresh when the records were extracted out of it. Usually a simple subtraction is good enough to carry out this match of count. This mathematical calculation gives birth to 3 possibilities:

   - **Case 01 [ Output is 0 ]**
     This signifies that the count is exactly matching from Source to the first staging database. And hence we can go forward with other tests on dimension.

     In this case, the next step is to go ahead with all column level data validations based on different logics, aggregations etc. If all column level validations result in matching values, the dimension / fact test is a PASS.

   - **Case 02 [ Output is a positive integer ]**
     A positive integer signifies that we have more records in Source database than what are going forward. So, in this case there’s no point testing anything further and it can be declared that the dimension is a FAIL.

   - **Case 03 [ Output is a negative integer ]**
     If the output of the count matching subtraction is a negative integer, this means that more records have flown into the staging database than what were present in the Source database, but technically logically this is not possible in any ETL process. The root cause of such situation is the source refresh rate. A situation like this arises when a source has been refreshed again after the records were extracted from Source and some of the older records were updated in that refresh. In a case like this, we need to go further before we go down the bottleneck into the column level validations. The steps followed are:

     2. Identify all the Unique Ids of all discrepant records from Source which are supposedly present in the staging database but have been updated in Source database.
3. Identify the record which was updated earliest as compared to all other such records and mark its update time as B.

4. Simultaneously, find out the maximum time at which the source database was refreshed from Control Table / Unix Box for that particular dimension and mark it as A.

5. Compare the 2 timestamps, i.e A and B. This comparison again gives birth to 2 possibilities:

   - **Case 01 [ A < B ]**
     This signifies that all discrepant records have been updated in the source after they were extracted for the ETL process. In this case, the next step is to go ahead with all column level data validations based on different logics, aggregations etc. If all column level validations result in matching values, the dimension / fact test is a PASS.

   - **Case 02 [ A = B or A > B ]**
     This means update of few records in Source database made them to fail to go through the ETL process, and hence it can be declared that the dimension being tested is a FAIL.

### Automating ETL testing process

The entire ETL testing process can be automated as per the flow depicted in the flowchart. For usual database systems like Oracle, we can use the SPOOLING process to spool in all the test SQL queries together.

The testing flow can be implemented in the single spooled test SQL query by using CASE and WHEN statements which decide on what test step should be followed in what condition. These spools of SQL queries store the result in a single output file which makes it easier for the tester to analyze all the test results.

- Automating the ETL testing process by using the SPOOL method helps in saving up to 50% of the execution time because all the test queries are based on the result of the previous query which helps in choosing the exact step to be followed and what steps to be left out in that case.

### ETL testing Checkpoints

Following is a checklist or testing standards that should be followed to ensure complete and exhaustive testing of ETL process in a data-warehousing project. The overall testing process is divided into the testing for data completeness, data quality, data transformation and meta-data and the points to be taken care of in these are as follows:

**A. Data Completeness**
One of the most basic tests of data completeness is to verify that all expected data loads into the data warehouse. This includes validating that all records, all fields and the full contents of each field are loaded. Strategies to consider include:

1. Comparing record counts between Source database data, staging table data and data loaded to target DW during testing for full load testing.

2. Comparing unique values of key fields between Source database, staging database and target DW. This is a valuable technique that points out a variety of possible data errors without doing a full validation on all fields.

3. Populating the full contents of each field to validate that no truncation occurs at any step in the process. For example, if the source data field is a string (30) make sure to test it with 30 and more characters.

4. Testing the boundaries of each field to find any database limitations. For example, for a decimal (3) field include values of -99 and 999, and for date fields include the entire range of dates expected. Depending on the type of database and how it is indexed, it is possible that the range of values the database accepts is too small.

**B. Data Transformation**

Validating that data is transformed correctly based on business rules can be the most complex part of testing an ETL application with significant transformation logic.

1. One typical method is to pick some sample records and "stare and compare" to validate data transformations manually. This can be useful only if we have sampling done for 1 particular dimension.

2. Create a spreadsheet of input data and expected results and validate these with the output of our test scripts.

3. During Incremental Testing - create perfect test data that includes all scenarios which can ever occur in source stage.

4. Validate correct processing of ETL-generated fields such as all control columns in dimension tables as well as the control tables present in all the stages.

5. Validate that data types in the warehouse are as specified in the Technical Design Documents.

6. Set up data scenarios that test referential integrity between tables. For example, what happens when the data contains foreign key values not in the parent table, or when a parent table is populated after child table.

7. Validate parent-to-child relationships in the data. Set up data scenarios that test how orphaned child records are handled.
C. Data Quality

Data quality deals with "how the ETL system handles Staging table data rejection, substitution, correction and notification without modifying data."

1. Reject the record if a certain decimal field has nonnumeric data.
2. Substitute null if a certain decimal field has nonnumeric data. NVLs should be used in such cases. For example, in calculation of some contracts for their costs, no. of days etc.
3. Compare accurate values to values in a lookup table, and if there is no match load anyway but report it somehow.
4. Determine and test exact points where to reject the data and where to send it for error processing.

D. Meta-data Testing

1. All Table and Column names should be as per the Technical design document.
2. All target columns should have datatype as given in the Technical design document or same as the corresponding source column in Source Database.
3. All target columns should have length equal or more than the corresponding source column in Source Database table.

E. Performance and Scalability

1. Load the database with peak expected production volumes to ensure that this volume of data can be loaded by the ETL process within the agreed-upon window.
2. Compare these ETL loading times to loads performed with a smaller amount of data to anticipate scalability issues. Compare the ETL processing times component by component to point out any areas of weakness.
3. Monitor the timing of the reject process and consider how large volumes of rejected data will be handled.
4. Perform simple and multiple join queries to validate query performance on large database volumes. Work with business users to develop sample queries and acceptable performance criteria for each query.

F. Integration Testing

1. Typically, system testing only includes testing within the ETL application. The endpoints for system testing are the input and output of the ETL code being tested. Integration testing shows how the application fits into the overall flow of all upstream and downstream applications. When creating integration test scenarios, consider how the overall process can break and focus on touchpoints between applications rather than within one application. Consider how process failures at each step would be handled and how data would be recovered or deleted if necessary.

2. Most issues found during integration testing are either data related to or resulting from false assumptions about the design of another application. Therefore, it is important to integration test with production-like data. Real production data is ideal, but depending on the contents of the data, there could be privacy or security concerns that require certain fields to be randomized before using it in a test environment. As always, don't forget the importance of good communication between the testing and design teams of all systems involved. To help bridge this communication gap, gather team members from all systems together to formulate test scenarios and discuss what could go wrong in production. Run the overall process from end to end in the same order and with the same dependencies as in production. Integration testing should be a combined effort and not the responsibility solely of the team testing the ETL application.

G. User-Acceptance Testing

1. Use data that is either from production or as near to production data as possible. Users typically find issues once they see the "real" data, sometimes leading to design changes.
2. Test database views comparing view contents to what is expected. It is important that users sign off and clearly understand how the views are created.
3. Plan for the system test team to support users during UAT. The users will likely have questions about how the data is populated and need to understand details of how the ETL works.
4. Consider how the users would require the data loaded during UAT and negotiate how often the data will be refreshed.
H. Regression Testing

Regression testing is revalidation of existing functionality with each new release of code. When building test cases, remember that they will likely be executed multiple times as new releases are created due to defect fixes, enhancements or upstream systems changes. Building automation during system testing will make the process of regression testing much smoother. Test cases should be prioritized by risk in order to help determine which need to be rerun for each new release. A simple but effective and efficient strategy to retest basic functionality is to store source data sets and results from successful runs of the code and compare new test results with previous runs.

Key hit-points in ETL testing

There are several levels of testing that can be performed during data warehouse testing and they should be defined as part of the testing strategy in different phases (Component Assembly, Product) of testing. Some examples include

- **Constraint Testing**: During constraint testing, the objective is to validate unique constraints, primary keys, foreign keys, indexes, and relationships. The test script should include these validation points. Some ETL processes can be developed to validate constraints during the loading of the warehouse. If the decision is made to add constraint validation to the ETL process, the ETL code must validate all business rules and relational data requirements.

  In Automation, it should be ensured that the setup is done correctly and maintained throughout the ever-changing requirements process for effective testing. An alternative to automation is to use manual queries. Queries are written to cover all test scenarios and executed manually.

- **Source to Target Counts**: The objective of the count test scripts is to determine if the record counts in the source match the record counts in the target. Some ETL processes are capable of capturing record count information such as records read, records written, records in error, etc. If the ETL process used can capture that level of detail and create a list of the counts, allow it to do so. This will save time during the validation process. It is always a good practice to use queries to double check the source to target counts.

- **Source to Target Data Validation**: No ETL process is smart enough to perform source to target field-to-field validation. This piece of the testing cycle is the most labor intensive and requires the most thorough analysis of the data. There are a variety of tests that can be performed during source to target validation. Below is a list of tests that are best practices:

  - **Threshold testing** – expose any truncation that may be occurring during the transformation or loading of data

    *For example:*

    Source: table1.field1 (VARCHAR40):

    Stage: table2.field5 (VARCHAR25):
Target: table3.field2 (VARCHAR40):

In this example the source field has a threshold of 40, the stage field has a threshold of 25 and the target mapping has a threshold of 40. The last 15 characters will be truncated during the ETL process of the stage table. Any data that was stored in position 26-30 will be lost during the move from source to staging.

- **Field-to-field testing** – is a constant value being populated during the ETL process? It should not be populated unless it is documented in the requirements and subsequently documented in the test scripts. Do the values in the source fields match the values in the respective target fields? Below are two additional field-to-field tests that should occur.

- **Initialization** – During the ETL process if the code does not re-initialize the cursor (or working storage) after each record, there is a chance that fields with null values may contain data from a previous record.

  For example:

  Record 125: Source field1 = Red Target field1 = Red

  Record 126: Source field1 = null Target field 1 = Red

  **Validating relationships across data sets** – Validate parent/child relationship(s)

  ➢ **Transformation and Business Rules**: Tests to verify all possible outcomes of the transformation rules, default values, straight moves and as specified in the Business Specification document. As a special mention, Boundary conditions must be tested on the business rules.

  ➢ **Batch Sequence & Dependency Testing**: ETL’s in DW are essentially a sequence of processes that execute in a particular sequence. Dependencies do exist among various processes and the same is critical to maintain the integrity of the data. Executing the sequences in a wrong order might result in inaccurate data in the warehouse. The testing process must include at least 2 iterations of the end–end execution of the whole batch sequence. Data must be checked for its integrity during this testing. The most common type of errors caused because of incorrect sequence is the referential integrity failures, incorrect end-dating (if applicable) etc, reject records etc.

  ➢ **Job restart Testing**: In a real production environment, the ETL jobs/processes fail because of number of reasons (say for ex: database related failures, connectivity failures etc). The jobs can fail half/partly executed. A good design always allows for a restart ability of the jobs from the failure point. Although this is more of a design suggestion/approach, it is suggested that every ETL job is built and tested for restart capability.

  ➢ **Error Handling**: Understanding a script might fail during data validation, may confirm the ETL process is working through process validation. During process validation the testing team will work to identify additional data cleansing needs, as well as identify consistent error patterns that could possibly be diverted by modifying the ETL code. Taking the time to modify the ETL process will need to be determined by the project manager, development lead, and the business integrator. It is the responsibility of the
validation team to identify any and all records that seem suspect. Once a record has been both data and process validated and the script has passed, the ETL process is functioning correctly. Conversely, if suspect records have been identified and documented during data validation those are not supported through process validation, the ETL process is not functioning correctly. The development team will need to become involved in finding the appropriate solution. For example, during the execution of the source to target count scripts suspect counts are identified (there are less records in the target table than in the source table). The records that are ‘missing’ should be captured during the error process and can be found in the error log. If those records do not appear in the error log, the ETL process is not functioning correctly and the development team needs to become involved.

- **Views:** Views created on the tables should be tested to ensure the attributes mentioned in the views are correct and the data loaded in the target table matches what is being reflected in the views.

- **Sampling:** Sampling will involve creating predictions out of a representative portion of the data that is to be loaded into the target table; these predictions will be matched with the actual results obtained from the data loaded for business Analyst Testing. Comparison will be verified to ensure that the predictions match the data loaded into the target table.

- **Process Testing:** The testing of intermediate files and processes to ensure the final outcome is valid and that performance meets the system/business need.

- **Duplicate Testing:** Duplicate Testing must be performed at each stage of the ETL process and in the final target table. This testing involves checks for duplicates rows and also check for multiple rows with same primary key, both of which can not be allowed.

- **Performance:** It is the most important aspect after data validation. Performance testing should check if the ETL process is completing within the load window. Check the ETL process for update times and time taken for processing of reject records. At least a year’s data should be present in the data warehouse. Above this, data loads for single days or batch windows should be done to check whether they finish within the provided window. Tools are available for simulating the number of concurrent users accessing the system for Stress/Load and performance testing.

- **Volume:** Verify that the system can process the maximum expected quantity of data for a given cycle in the time expected.

- **Connectivity Tests:** As the name suggests, this involves testing the upstream, downstream interfaces and intra DW connectivity. It is suggested that the testing represents the exact transactions between these interfaces. For ex: If the design approach is to extract the files from source system, we should actually test extracting a file out of the system and not just the connectivity.

- **Negative Testing:** Negative Testing checks whether the application fails and where it should fail with invalid inputs and out of boundary scenarios and to check the behavior of the application.
References

1. [http://www.informationmanagement.com](http://www.informationmanagement.com)

Author’s Biography

Anand Singh was born and brought up in a small town of Mandi (Himachal Pradesh), India. After his schooling from his hometown, he completed his Engineering in Computer Science from Institute of Engineering & Technology Bhaddal, Chandigarh.

Fresh out of college, Anand started his IT career in Bangalore with Accenture Services Pvt. Ltd. Where he has been working on from past 1 and half years. In this time, he has worked for a big client by testing their billion dollars Business Intelligence System which is in turn based out of a big Data Warehouse.

Anand specialized himself as an ETL testing expert within short duration of time to accomplish the needs of testing a big data warehouse project. Through the tough roads of challenges, he gained knowledge and expertise. Through all those experiences and loads of thesis work, he has come out with this white paper called the “ETL Testing Best Practices”.