Data Warehouse and Business Intelligence Testing: Challenges, Best Practices & the Solution

Prepared by

datagaps

http://www.datagaps.com

http://www.youtube.com/datagaps

http://www.twitter.com/datagaps

Contact

contact@datagaps.com

Contributors

Narendar Yalamanchilli

Adi Buddhavarapu

Table of Contents

EXECUTIVE SUMMARY

Testing data warehouse and business intelligence projects is challenging due to the heterogeneous nature of data sources and large volumes of data and reports. Based on our experience in implementing various data warehouse and business intelligence projects, we have come up with a set of best practices/patterns in testing to minimize the risks associated with data intensive projects. Our methodology conceptually divides testing into three areas.

- Data Entity Testing: This is to ensure that the data in every source (flat files, tables etc) or target entities (fact, dimension etc) is as expected.
- ETL Testing: This is to ensure that the data moved from the source systems to the target/warehouse as expected.
- BI Testing: This is to ensure that teams have a firm grasp on reports and dashboards from a regression, performance and stress testing stand point.

This paper walks the reader through the typical issues which are encountered in testing warehouses and suggests best practices for automation

THE PROBLEM

Data warehouse projects often involve extracting large volumes of data from multiple data sources before transforming and loading. Testing of data warehouse requires a data-centric testing approach which is different from testing of traditional transactional applications. Most of the current test automation tools are focussed on transaction application testing and are not designed for testing of data warehouses. Hence it is a common practice for Business Analysts, Engineers and QA Teams to manually test the ETL by copying a handful of records from source and target tables into an excel spread sheet and compare the data. This is a tedious and error prone process when done manually for large number of ETL. Testing is further complicated because the ETL process can run in an incremental mode or full mode often making it necessary to create separate sets of test cases for these two modes. Every incremental change to the source or target system increases the scope for regression and these manual tests have to be repeated every time.

When we extend this problem to BI environments, testing becomes even more challenging as there are hundreds of dashboards and reports to be verified. Testing is further complicated due to the abstraction provided by the BI tool metadata model because the scenarios for adhoc report requests are not fixed and testing all the scenarios is not practical. As the business intelligence application matures, it becomes very difficult to assess the impact of the changes in the metadata model on existing reports because of the effort involved in testing all the reports. Most of the current forms of testing involve manual spot checking by running a few reports and comparing it to the data queried from the physical data sources (e.g. databases). Often there is very little or no regression testing performed for any changes being made for these applications since there is no easy way to automate this testing. The impact of not regression testing reports can vary from throwing errors, showing wrong data or bad performance which is usually detected only when the changes go to production. In addition, system upgrades such as upgrades to BI tool, database or hardware may result in report accessibility issues, errors in rendering reports or performance issues.

These issues often result in increased cost of ownership, inaccurate insights due to bad data/reports and the loss of trust in enterprise data and reporting capabilities. The below section identifies the kinds of issues we see based on the above categories and the best practices to address them.

We believe that there are three areas to ensure proper data testing.

- First, testing has to be done at each data entity level to ensure that the data coming into the warehouse is as expected.
- Second, as data moves from various sources to the warehouse, we need to ensure that data is exactly as expected.
- Finally, the reporting system itself has to be tested for accuracy, regression, performance and scalability.

Issues with Data Entities

Data can enter an enterprise from various sources such as transactional systems, XML files, flat files, web services etc and it is possible for data to take up unexpected forms in this step itself. Few examples are below:

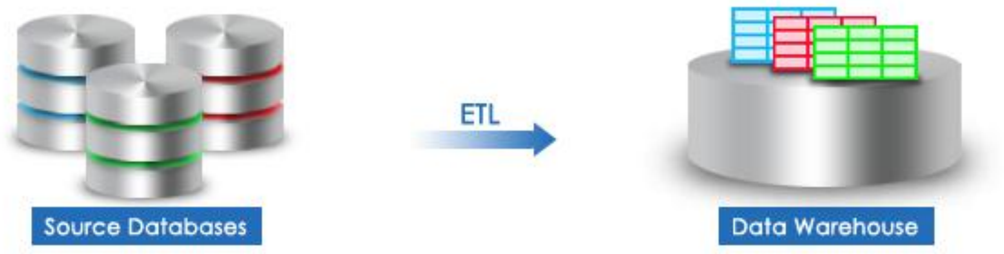
1. An application does not have proper data validation and thus allows inaccurate data to be entered into the system:
    a. Invalid email addresses without an "@" or not following the email address pattern
    b. A column to store country names has values that do not fall in the encoded list of values defined for it
    c. SSN is a required data attribute but has null values or less than 9 numbers
    d. Orphan order item (child) records are persisted even though the corresponding order (parent) record has been deleted
    e. Duplicate records are created in the system for the same customer
    f. Data for start and end dates is not consistent because end date is less than start date
    g. Address entered by the customer is a not a valid one or postal code is null for US addresses

2. A number of advertising companies get customer data on a daily basis in flat files from various stakeholders. It is possible that the column with the "First Name" of the first one million incoming customer records is missing. If a campaign is sent out using this data set to all the one million customers, they get emails without their first names in the campaign and may negatively impact the brand value.
    a. Flat file does not contain expected number of data columns
    b. Data in a column has invalid type or length eg. Invalid date format or text in a number column
    c. Date of birth is more than 150 years old
    d. Address fields are null when country is US when they are expected to be available
    e. The product codes passed in the flat file does not exist in the list stored in the product code table

Prior to moving this data into other systems within the enterprise, it is important that we identify such discrepancies early on. The current Data Quality tools primarily focus on data discovery using data profiling and cleaning of data but do not provide a good means to measure the quality of the data by providing capability to define data rules. These data quality tools are often very expensive and difficult to implement.

Issues with Data Warehouses

Data warehouses (and Data Integration) projects typically involve movement of large volumes of data from multiple data sources.

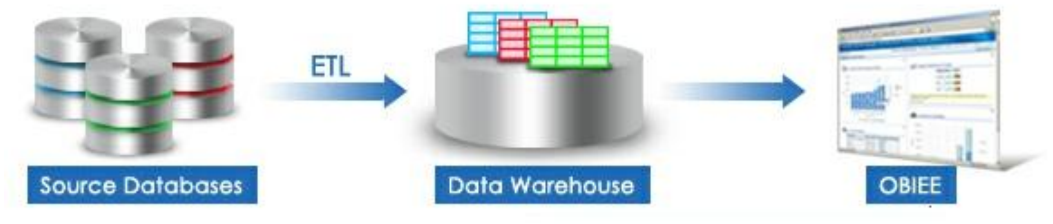


A small sample of issues noticed in data warehouses are listed below:

1. Sales fact does not have all the sales records from the source because of a bad join in the query for extracting source data extract
2. Customer Last Name is truncated because the column data lengths are not matching between the source and target
3. One of the columns in Product dimension is empty due to a defect in ETL
4. Revenue fact has duplicate records due to which the revenue numbers in the report are off
5. Issues with data transformations result in unexpected data. For example:
    a. Lookup transformation for getting country name based on country code results in null values
    b. Foreign keys for Customer dimension are not mapping as expected in the Sales fact
    c. Order amount in the aggregate table does not match the amounts from the Order fact
    d. Customer Status value derived in the ETL is shown as 'Inactive' for a specific record when the expected value is 'Active'
6. Incremental ETL is not updating the values in the de-normalized address columns of the Customer dimension with the latest address when the address changes
7. Incremental ETL did not process changes for the Case data in the source because the issues with the change capture logic or defect in ETL update strategy
8. Incremental ETL did not create a new record for a Type-2 slowly changing dimension
9. Historical Sales data that is expected to be unchanged is different because of an ETL defect
10. Data is missing in the target table because they are wrongly tagged as error records
11. Duplicate records are loaded into Customer dimension because of two different source (ERP and CRM) system resulting in bad data in the reports

Issues with Business Intelligence Applications

Business Intelligence applications present a different set of challenges in testing because they empower the end user to create their own reports and dashboards on the fly.



Some of the issues noticed during in business intelligence applications are:

1. Defect in ETL for Sales fact results in wrong sales numbers to be shown in Sales Dashboard
2. Adding a new conformed Product Category dimension in the BI tool metadata results in extra unwanted joins for reports based out of Sales subject area thus skewing the sales numbers
3. Business Analyst does not trust the Revenue numbers shown in the revenue dashboard because they do not match numbers from the source
4. Incomplete changes to Sales subject area are migrated to the next environment as part of a fix made for Revenue subject area because of common metadata model (RPD)
5. Developer modifies the measure in Order Trend report without notifying the business analyst or quality analyst which goes untested to production
6. BI tool upgrade results in bad report performance and errors in some of the dashboard pages
7. Database upgrade results in performance issues because the queries are executing with a new plan
8. Rollout of the new Customer Intelligence application to 400 new users results in all dashboards to run slow because of a network bottleneck
9. A developer modifies column formatting for date column for all subject areas without realizing the impact
10. A developer renames 'Employee Name' to 'Owner Name' without realizing that there are existing reports using this attribute
11. ETL developer increases the length of lst_name column or renames it to last_name without notifying the report developer
12. A change in the initialization block for authorization results in unwanted loss of access to Customer Management dashboard to business analysts
13. Aggregate numbers shown in the summary report do not match the numbers shown in the drilldown report

Best practices

Testing data warehouses and business intelligence applications requires a data-centric testing approach. This section provides a list of the type of tests and the strategy/best practice for doing the test

Data Entity Testing

The focus of data entity testing is to identify the issues with data that is not following the data rules defined for it.

Type	Description	Strategy
Data Validity	Are the values within the acceptable subset of an encoded list?	For each attribute with a defined list of values, check for values in the data entity to identify the values that are not in the list of valid values
Data Consistency	Are there any unexpected duplicates?	Write queries to check for duplicates based on combinations of columns that are expected to be unique
Referential Integrity	Are there any orphan records or missing foreign keys	Identify parent-child and many-to-many relationships between data entities and write queries to identify inconsistencies
Data Validity	Is the data as per business rules? Email column does not have valid email addresses Birth date has values older than 200 years Postal Code has non-numeric values for US addresses SSN has 'null' values for some records Combination of Customer Name, address columns is unique	For each attribute, run SQL queries to check the data. This may involve using regular expressions.

Data Warehouse Testing

The goal of data warehouse (or data integration) testing is to verify that data moved from source to the target and the transformation rules have been applied as expected.

Type	Description	Strategy
Data Completeness	Are the counts matching between source and the target? For example, source records can be missing in target due to bad filter, rejected records, wrong data types.	While it is ideal to compare the entire source and target data, it is often a very time consuming and resource intensive process. Hence comparing the counts is a good strategy. This will also help identify

		duplicate introduced during the ETL. Compare source and target table row counts. Compare count of rows with data in individual columns between source and target. Compare distinct list of values and count of rows for each value between the source and target.
Data Integrity	Is the target data consistent with the source data?	Data completeness does verify that the data counts are matching but it does not ensure that the data values are consistent with the source. For example, data can get truncated in a column. Hence comparing the data values between the source and target for a sample of data (eg. 10% of the data) can identify these issues. For testing incremental runs, the data that got modified in a specific time period can be verified. This type of test is appropriate for data with little (eg. Expression transformation) or no transformations.
Data Integrity	Is the Fact to dimension table foreign key mapped in the ETL appropriately?	For each fact – dimension foreign key Check the count of mapped rows in source DB and compare it with target DB. For a sample set of fact records, compare the foreign key mapping between the source and target DB
Data Transformations	How to test Lookup transformation?	Write SQL query to compare the list of values in a column with the list of valid values expected
Data Transformations	How to test Aggregate transformation?	Write SQL query to aggregate the data and compare it with data in aggregate target table
Data Transformations	How to test Expression transformation? For eg, are nulls replaced appropriately?	Write SQL query with the expected transformation on the source data and compare it with data in target table
Data Transformations	My transformations are too complex to test using single source and target queries	Setup test data in the source for different conditions. Verify that the data in the target against expected result
Data Transformations	How to perform regression testing of transformations?	Benchmark the target table records for data that is not expected to change (eg. 10000 records) and compare it after the ETL change

Business Intelligence Testing

While data warehouse (ETL) testing is an essential part of ensuring that the data shown in the reports is correct, issues in the BI tool metadata model can cause this data to be reported wrongly. Hence the primary focus of BI testing is to ensure that numbers and data shown in reports are correct when compared to the source and the reports are accessible to the end user from a performance and security standpoint. Listed below are some of the common test scenarios and strategies.

Type	Description	Strategy
Unit Testing	Dashboard prompts are showing wrong data or they are not getting applied to reports appropriately	For each prompt in a dashboard page, check if the list of values shown is correct where applicable. Apply all the prompts and verify the report and the corresponding physical SQL. Verify that all prompts are getting applied appropriately to the drilldowns.
Unit Testing	Dashboard page does not conform to UI Standards	Verify that the report layout, prompts, titles, download and filter display meet the design
Unit Testing	The SQL Query generated by BI Server for a report is complex and shows wrong data (numbers)	Write your own SQL query on the target database and compare the output with the report data
Unit Testing	An RPD change for a new subject area changes the join conditions for an existing report	Compare the SQL Query generated by the report before and after making RPD changes to the RPD
Unit Testing	Mismatches between Summary and detail report data	Aggregate the detail reports output and compare it with the summary report data. Check the links to detail report from charts, data and table headings in the summary report
Unit Testing	A defect in the new ETL results in the corresponding report showing wrong data	Benchmark the report output and compare it. Another option to compare the data shown in reports with the output of a query against the source database
Unit Testing	Webcat merge from local desktop to development environment is unsuccessful	Compare the report data between the local and development environments
Unit Testing	How can we test adhoc reporting since the number of possible test cases can be unlimited? While it is not practical to test all combinations of adhoc reports is there a method to testing adhoc reports?	1) For each dimension folder, pick all attributes and run reports. The objective is to check if any attribute is not mapped properly. Also check the dimension hierarchy. 2) Run report by picking one attribute from each dimension and one measure to verify the joins. Watch for any unwanted joins. If

		you are seeing a complex query it is usually because of bad model or design. 3) Verify that the list of measures and their names are meaningful
Unit Testing	RPD merge results in duplicate attributes	Compare the list of presentation layer attributes before and after the merge
Regression Testing	A developer knowingly or unknowingly makes a dashboard UI change that the business analyst/tester is not aware of	Take screenshots of the dashboard pages before and after the changes and compare them
Regression Testing	Some of the dashboard pages suddenly start running slow after a recent release or system change	Benchmark and compare the dashboard page run time and corresponding SQL query execution plan before after the change
Regression Testing	An ETL change results in the measure values in the reports to be wrong after the full ETL run	Benchmark and compare the report data and chart before after the change
Regression Testing	Name change of an attribute in the presentation layer adversely impacts user reports in production	Search for impacted reports in production usage tracking data or search in the web catalog
Regression Testing	It is difficult to prioritize which reports or adhoc request to regression test	Regression test most commonly used reports or adhoc requests based on production usage tracking data
Regression Testing	New reports or dashboard pages are running really slow and not meeting SLAs	Capture dashboard page run times after disabling caching. Alternatively capture the average run times by running them multiple times at different times
Regression Testing	Dashboard page runs fine for some roles but is really slow for other roles (eg. CEO)	Capture dashboard page run times after disabling caching for users with different roles
Regression Testing	Drilldown on Hierarchy columns is very slow	Verify the drilldown time for each hierarchy column with caching disabled
Multi-Environment	A recent webcat migration results in access issue due to which some of the users cannot access their reports	Compare the report access for key users with different roles across environments after the migration
Multi-Environment	Uptake of new BI releases (eg. 10g to 11g or 11.1.1.5 to 11.1.1.7) results in report data, security and performance issues	Have a pre and post upgrade environments pointing to the same database and compare the report data and performance between the two environments
Security	Users see data in reports that they are not supposed to have access	Test data security by comparing the data for the same report between two different users with different roles
Security	Users lose access to reports and	Verify the access to reports and dashboards

	dashboards that they previously had access to	for users with different roles before and after the security changes or webcat migration
Stress	Dashboard pages start running slow after rollout to new users or addition of new data	Stress test by simulating concurrent users and compare dashboard page performance to see if the SLA requirements are met
Stress	A software or hardware upgrade adversely impacts the dashboard page performance	Stress test by simulating concurrent users and compare dashboard page performance to see if the SLA requirements are met
Stress	Reports run fine in test environments but run slow in production	Simulate realistic user load in a test environment based on production usage tracking data

Testing Metrics

Data Entity Test Coverage

Table	No. of Rules	% of attributes with rules	No of attributes
User	0	0	12
Customer	5	10%	25
Sales	12	75%	15

Key Relationship tests

Key Relationship	Number of Orphans	% of Orphans
Sales-Customer	0	0
Sales-Product	10	1%

Data Rules

Table	Column	Rule	Number of non-conforming values	% of non-conforming values
Customer	Postal Code	Postal Code is numeric where country code = 'US'	4	0.04%
Customer	Customer Name	Customer Name, Address is unique	0	0%
Customer	SSN	SSN is not null	0	0%
Customer	Email	Email like REGEXP ^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$	1000	10%
Customer	Gender Code	Gender Code in valid List of values	100	1%

Target Table ETL Test Coverage

Target Table	Data Integrity Tests	Data Completeness Tests	Data Consistency Tests	Data Transformation Tests	Total
Costs	1	1	0	0	2
SalesRep	0	0	0	0	0
Customer	0	1	1	0	2
Sales	1	0	0	0	1

Dashboard Test Coverage

Dashboard	Page	Usage Rank	No. of Regression Tests
Sales	Sales Trends	2	1
Sales	Product Profitability	5	1
Sales	Sales Details	10	0

Report Test Coverage

Dashboard	Page	Report	Usage Rank	No. of Unit Tests
Sales	Sales Trends	Sales Trends	2	1
Sales	Product Profitability	Product Profitability	5	1
Sales	Sales Details	Sales Details	10	0

THE SOLUTION

Datagaps offers two solutions that collectively enable comprehensive data warehouse and business intelligence testing automation. Testing teams, Business Analysts and Engineers can leverage our solutions to minimize testing costs, expedite time to market and deliver high quality dashboards and reports to business users. Details of the solutions are below:

ETL Validator: This primarily addresses Data Entity Testing and ETL Testing. Key features include:

- Wizard driven test case creation
- Enterprise grade test case visibility
- Enterprise collaboration
- Flat file testing automation
- Data entity testing automation

BI Validator: This addresses BI Testing Automation. Key features include:

- Dashboard testing
- Report testing
- Multi Environment testing
- Upgrade testing
- Stress testing

Data sheets for both the products are available on our website for your reference.