

Design Tip #134 Data Warehouse Testing Recommendations

By Joy Mundy

A data warehouse / business intelligence system is challenging to test. Standard testing methodology tests one little thing at a time, but a DW/BI system is all about integration and complexity, not to mention large data volumes. Here are my top five recommendations for building and executing a testing environment for your DW/BI project.

1. Create a small static test database, derived from real data.

You want it to be small so tests can run quickly. You want it to be static so that the expected results are known in advance. And you want it to be derived from real data because there's nothing like real data to give you a realistic combination of scenarios, both good and bad. You will need to "cook" additional rows into the test database to test any branch of your ETL code that covers a data scenario not included in the original test data.

2. Test early and often.

Start testing as soon as you write a line of code (or connect two boxes in your ETL tool's user interface). Developers do this all the time, of course, developing and running unit tests to ensure their code does what it's supposed to do. Many developers aren't as good about keeping track of all those tests, and running them often. Daily. Every time you check in code. If you run your tests daily, and prioritize fixing any tests that broke yesterday, it will be easy to determine what went wrong.

Unit testing assures that a developer's code works as designed. System testing ensures that the entire system works, end to end, according to specifications. System testing should also begin early. There's an official test phase before you go live; this test phase is for running tests and fixing problems, not for identifying what the tests should be and how to run them. Start system testing early in the development process, so all the kinks are worked out long before the pressure-cooker system testing phase begins.

3. Use testing tools and automate the test environment.

The suggestion to test early and often is practical only if you automate the process. No developer is going to spend the last hour of the work day babysitting unit tests! And few teams can afford a full time tester to do that work on the developers' behalf.

To automate testing, you need tools. Many organizations will already have system quality assurance testing tools in place. If you don't, or if you're convinced your tools won't meet the needs of the DW/BI system testing, try googling "software quality assurance tools" for an overwhelming list of products and methodologies available at a wide range of costs.

All commercial software test tools will allow you to enter tests, execute tests, log the results of test runs, and report on those results. For unit testing and data quality testing, define tests to run a query in the source and target data warehouse. You're looking for row counts and amounts to match up.

A testing tool used for DW/BI testing must be able to run a script that sets up the test environment before the tests are run. Tasks you may need to execute include:

- Restoring a virtual machine environment with clean test data
- Modifying the static test data with special rows to test unusual conditions
- Running your ETL program

After the tests are executed and logged, end with a cleanup script, which may be as simple as dropping the VM environment.

Standard testing methodology has you change one thing, run a test, and log results. In the DW/BI world, you should expect to group together many tests into a test group. Even with a tiny test database, you don't want to execute your ETL code for each of the hundreds of unit tests that you should be running.

4. *Enlist the business users to define system tests.*

We need the business users' expertise to define good system tests. How do we know the data is correct? How do we know that query performance meets their expectations? Enlisting business users in the test specification process will ensure better testing than if the DW/BI team just made up tests based on what they think is interesting. Engaging key business users in the quality assurance process also provides a huge credibility boost.

5. *The test environment must be as similar as possible to the production environment.*

It is vitally important that the test environment be similar to production. Ideally, it's exactly the same hardware, software, and configuration. In the real world, relatively few organizations have the budget for two big DW servers. But any organization can, and should, make the following elements matchup:

- *Drive configuration* (relative names for drives). Disk is cheap and you should be able to duplicate your disk for test. But if you can't, at least make the drive letters and database file layout the same. Many people have whined at me that this is so much work to change the environment and make them the same. Yes it is! And so much better to do it now than in the final testing phase of your project.
- *Software versions* from the operating system to the database to the users' desktops, and everywhere in between.
- *Server layout*. If the reporting system software will be on its own server in production, test it that way.
- *Security roles and privileges* for back room service accounts. Your deployment is virtually guaranteed to fail if you don't test security roles first. I don't know why, but it always seems to go wrong.

If you follow these suggestions, especially the suggestion for continuous testing, you are likely to have a smooth, crisis-free test phase and move into production on schedule. If not, you're running a serious risk of having an otherwise fabulous project be delayed interminably in QA hell, with business users and management rattling the doors.