What Is Database Testing And Why Is It Important?

## What Is The Scope Of Database Testing Or What Are The Areas To Cover While Testing A Database?

Testing a database requires checking the following areas are working as expected.

* Database objects which include tables, views and stored procedures.
* Validation of data being entered and getting stored in the database.
* Making sure the system is honoring the constraints and data changes (insert/delete/update) are reflecting correctly.
* Ensure the system can execute end to end database transactions and support concurrency.
* Testing of database migration to ensure compatibility across different versions.
* Verify performance of database indices, triggers, and procedures.
* Security compliance testing to rule out any unauthorized access or any threats to data.

## What Are The Known Database Validations?

Testing a database involves following four validations.

* Data mapping.
* ACID (Atomicity, Consistency, Isolation, Durability) properties validation.
* Data Integrity.
* Business Rule Conformance.

## What Is Data Mapping In Database Testing?

Data mapping is a part of the database testing scope. It focuses on validating the data traversing back and forth from the application to the backend database. A software tester can look at it with the following two aspects.

He can check whether the application’s user interface or the input screen has a one-to-one mapping with the target table in the database. This type of information is usually available in software design documents.

Whenever the user submits a form on the application UI, it triggers a CRUD (Create/Retrieve/Update/Delete) event at the backend. So here the tester should ascertain if the right event gets fired and finished successfully or not.

## What Is ACID Properties Validation In Database Testing?

ACID is an acronym for Atomicity, Consistency, Isolation, and Durability. It refers to confirming the four properties (as mentioned) against each database transactions.

* **Atomicity –** It means that all Database Transactions are atomic. They can end in one of the two i.e. Success or Failure.
* **Consistency –** It indicates that the database state will remain valid after the transaction gets completed.
* **Isolation –** Multiple transactions shall run without impacting each other and won’t hinder the database state.
* **Durability –** Committing a transaction shall preserve the change and will not lose it due to any power loss or crash later.

## What Are The Steps Involved In Database Testing?

For successful testing of a database, we need to adopt a structured approach. And a tester should have at least a basic level of understanding of database concepts like SQL commands including DDL (Create/Drop), DML (Insert/Delete/Update), and DCL (Grant/Revoke).

* **DDL =>** Data Definition Language.
* **DML =>** Data Manipulation Language.
* **DCL =>** Data Control Language.

Now, we are laying down the steps to carry out the testing of a database.

### 1. Set Up Testing Pre-Requisites.

First of all, the tester has to create the setup for testing the database. He can do this by cloning a production database or creating a stand-alone database using demo data. He should keep in mind the following points.

The system running the database (Physical or virtual) should have the same hardware characteristics as does the production system.

The database software (Oracle/SQL Server/MySQL or any) should have the same version as running on the production system database.

### 2. Execute The Tests.

A software tester can run the tests from the application UI. Or he can write SQL scripts to invoke the CRUD actions. He should monitor the operations, track database changes and verify whether they are ending with success or not.

Better come with both positive and negative tests covering views, triggers, and stored procedures. Make sure all the functions/blocks are getting tested.

### 3. Verify Test Status.

After running the tests, the tester should analyze all failures or tests that get ignored from execution. It’s wise to come up with a summary of all types of cases and then focus on failures and skipped scenarios.

### 4. Validate Results.

In this step, we’ll concentrate if the tests approached the right action after executing the SQL queries or not. Also, he has to ensure the data changes affected the right tables and columns.

### 5. Consolidate And Publish Report.

The final step is to gather all the results and capture all of them into a report. You need to share this data with all stakeholders. Please make sure that all failures or skipped tests have given proper reasons.

## How To Write Test Cases For Database Testing?

A software tester should prefer writing a separate set of test scenarios and test cases for database testing. That’s how you can make sure your test cases are independent and won’t mix with the UI related use cases.

**Database testing is a type of grey-box testing**. And if you don’t’ know, then note that it’s a mixture of both black box & white box testing methods. So the tester would need to know about the internal functioning of the application and also about the database structure used.

At times, an application could be using multiple databases so you should be aware of hot they relate to each other. Also, make a data mapping sheet which should tell you the UI actions and the tables (and columns) getting affected when the action takes place. **For example**, sometimes it would intend adding a row in one table and update an another row in a different table. While writing the test cases, you need to keep an eye on all of these actions to successfully test a database.

And as we’ve said earlier in this post that you need good SQL skills for testing databases. It will help you write efficient SQL queries that extract data from a table without returning thousands of rows from the database table.

### Database Testing Guidelines.

You can use the below **guidelines** to prepare good test cases for database testing.

* Get clarity on the functional requirements.
* Make a list of all the tables used and find out-
  + Joins used between tables
  + Cursors used, triggers used
  + Stored procedures used
  + Input/Output parameters used.
* Create test cases with multiple input data and try to cover all the paths.

### Database Testing Checklist.

After writing the test cases, refer to the following **checklist** and see if any information is missing.

* The tests are taking care of all the backend tables used for each requirement.
* If application/database is using status flags, then tests should verify each of them.
* Tests cover the triggers/stored procedures with combinations of input and expected output parameters.
* Tables might have columns with default values, tests should check them too.

## What Are The Possible Test Scenarios For Database Testing?

Here, we are listing down three types of scenarios possible for testing a database.

### 1. General Test Scenarios.

General tests should capture the following scenarios.

* Name of the database.
* Name of the log file.
* Disk space allocation for databases.
* Names of all tables, columns, and their types.
* Null value checks.
* Verify keys (primary/foreign), indexes, and data types of columns used.

### 2. Functional Test Scenarios.

* Identify events causing the triggers
* Functions inside stored procedures and possible combinations.
* End to end data flow starting from the front end to the backend.

### 3. Non-Functional Test Scenarios.

* Create test scripts for major features and use them for regression testing at regular intervals.
* Write tests that track errors (OOM/deadlocks/exceptions) in log files.
* Change data in backend tables and watch the effect on the frontend.
* Insert invalid values from the backend and observe the effect.