**Database Testing – Processes**

The process to perform database testing is similar to testing of other applications. DB testing can be described with key processes given below.

* Set up the environment
* Run a test
* Check the test result
* Validate according to the expected results
* Report the findings to the respective stakeholders

Various SQL statements are used to develop the Test cases. The most common SQL statement, which is used to perform DB testing, is the **Select** statement. Apart from this, various DDL, DML, DCL statements can also be used.

**Example** − Create, Insert, Select, Update, etc.

## **Database Testing Stages**

DB testing is not a tedious process and includes various stages in database testing lifecycle in accordance with the test processes.

The key stages in database testing are −

* Checking the initial state
* Test run
* Outcome validation as per expected result
* Generating the results

**First stage** in DB Testing is to check the initial state of the database before starting the testing process. Then database behavior is tested for defined test cases. In accordance with the results obtained, test cases are customized.

For successful database testing, the workflow given below is executed by every single test.

* **Cleaning up the database** − If there is testable data in the database, it should be emptied.
* **Set up Fixture** − This involves entering the data into the database and check the current state of the database.
* **Perform test, verify results and generate results** − The Test is run and the output is verified. If the output is as per expected results, the next step is to generate the results as per requirement. Otherwise, testing is repeated to find the bugs in database.

# Database Testing – Techniques

As mentioned earlier, it involves testing each object in the Schema.

### **Verifying Databases and devices**

* Verifying the name of database
* Verifying the data device, log device and dump device
* Verifying if enough space allocated for each database
* Verifying database option setting

### **Tables, columns, column types rules check**

Verify the items given below to find out the differences between actual and applied setting.

* Name of all the tables in database
* Column names for each table
* Column types for each table
* **NULL** value checked or not
* Whether a default is bound to correct table columns
* Rule definitions to correct table names and access privileges

### **Key and Indexes**

Verify the Key and indexes in each table −

* Primary key for each table
* Foreign keys for each table
* Data types between a foreign key column and a column in other table Indices, clustered or non-clustered unique or not unique

## **Stored Procedure Tests**

It involves checking whether a stored procedure is defined and the output results are compared. In a Stored Procedure test, the following points are checked −

* Stored procedure name
* Parameter names, parameter types, etc.
* **Output** − Whether the output contains many records. Zero rows are effected or only a few records are extracted.
* What is the function of Stored Procedure and what a stored procedure is not supposed to do?
* Passing sample input queries to check if a stored procedure extracts correct data.
* **Stored Procedure Parameters** − Call stored procedure with boundary data and with valid data. Make each parameter invalid once and run a procedure.
* **Return values** − Check the values that are returned by stored procedure. In case of a failure, nonzero must be returned.
* **Error messages check** − Make changes in such a way that the stored procedure fails and generate every error message at least once. Check any exception scenarios when there is no predefined error message.

## **Trigger Tests**

In a Trigger test, the tester must perform the following tasks −

* Make sure the trigger name is correct.
* Validate the trigger if it is generated for a specific table column.
* Trigger’s update validation.
* Update a record with a valid data.
* Update a record with invalid data and cover every trigger error.
* Update a record when it is still referenced by a row in other table.
* Ensure rolling back transactions when a failure occurs.
* Find out any cases in which a trigger is not supposed to roll back transactions.

## **Server Setup Scripts**

Two types of tests should be performed −

* Setting up the database from scratch, and
* To set up an existing database.

### **Integration Tests of SQL Server**

Integration tests should be performed after you are through with component testing.

* Stored procedures should be called intensively to select, insert, update, and delete records in different tables to find any conflicts and incompatibility.
* Any conflicts between schema and triggers.
* Any conflicts between stored procedures and schema.
* Any conflicts between stored procedures and triggers.

## **Functional Testing Method**

Functional testing can be performed by dividing the database into modules as per functionality. The functionalities are of the following two types −

* **Type 1** − In Type 1 testing, find out the features of the project. For each major feature, find out the schema, triggers, and stored procedures responsible to implement that function and put them into a functional group. Then test each group together.
* **Type 2** − In Type 2 testing, the border of functional groups in a back-end is not obvious. You can check the data flow and see where you can check the data. Start from the front-end.

The following process takes place −

* When a service has a request or saves data, some stored procedures will get called.
* The procedures will update some tables.
* Those stored procedures will be the place to start testing and those tables will be the place to check the test results.

## **Stress Testing**

Stress Testing involves getting a list of major database functions and corresponding stored procedures. Follow the steps given below for Stress Testing −

* Write test scripts to try those functions and every function must be checked at least once in a full cycle.
* Perform the test scripts again and again for a specific time period.
* Verifying the log files to check any deadlocks, failure out of memory, data corruption, etc.

## **Benchmark Testing**

If your database does not have any data problems or bugs, system performance can be checked. A poor system performance can be found in benchmark testing by checking the parameters given below −

* System level performance
* Identify most-likely-used functions/features
* Timing – maximum time, minimum time and average time to perform functions
* Access volume

## **Testing a Database via Front-end**

Back-end bugs can also be found sometimes by doing front-end testing. You can follow the simple steps given below to detect bugs by front-end testing.

* Write queries from the front-end and issue the searches.
* Pick up an existing record, change the values in some fields, and save the record. (It involves the UPDATE statement or update stored procedures and update triggers.)
* Insert a new menu item in the front-end window. Fill in the information and save the record. (It involves the INSERT statements or insertion stored procedures and deletion triggers.)
* Pick up an existing record, click on the DELETE or REMOVE button, and confirm the deletion. (It involves the DELETE statement or deletion stored procedures and deletion triggers.)
* Repeat these test-cases with invalid data and see how the database responds.

## **Structured Database Testing**

Common database scenarios with respect to Structured Database Testing are given below −

* Verifying the name of database, verifying the data device, log device and dump device, verifying if enough space allocated for each database and verifying database option setting.
* Names of all the tables in database, column names for each table, column types for each table, null value check or not. Verify the Key and indexes in each table: Primary key for each table, foreign keys for each table.
* Data types between a foreign key column and a column in other table Indices, clustered or non-clustered unique or not unique.

## **Functional Database Testing**

Common Database Test scenarios with respect to **Functional Database Testing** are −

* Finding out the schema, triggers and stored procedures responsible to implement that function and make them into a functional group and then each group can be tested together.
* Check data flow and see where you can check the data. Start from the front-end.

## **Non-Functional Database Testing**

Common Database Test scenarios with respect to **Non-Functional Database Testing** are −

* Write test scripts to try major functions and every function must be checked at least once in a full cycle.
* Perform the test scripts again and again for a specific time period.
* Verifying the log files to check any deadlock, failure out of memory, data corruption, etc.
* Write queries from a front end and issue the searches. Pick up an existing record, change values in some fields and save the record. (It involves UPDATE statement or update stored procedures, update triggers.)
* Insert a new menu item in a front-end window. Fill in information and save the record. (It involves INSERT statements or insertion stored procedures, deletion triggers.)
* Pick up an existing record, click on the DELETE or REMOVE button, and confirm the deletion. (It involves DELETE statement or deletion stored procedures, deletion triggers.)
* Repeat these test-cases with invalid data and see how the database responds.

**Schemas**, **tables**, **stored procedures**, and **Triggers** are key objects of a database. We have already shared DB testing types and test scenarios for these data base objects.

## **Schemas**

A database schema defines the structure of a database system in a format supported by the database management system. A Schema refers to how a database is structured (composed of database tables in the case of Relational Databases).

The database schema is a set of formulas called integrity constraints imposed on a database. These integrity constraints ensure compatibility between parts of the schema.

In a relational database, the schema consists of tables, fields, views, indexes, packages, procedures, functions, triggers, types, materialized views, synonyms, database links, and other elements.

Schemas are generally stored in a data dictionary. Although a schema is defined in text database language, the term is often used to refer to a graphical depiction of the database structure. In other words, schema is the structure of the database that defines the objects in the database.

Common type of Schemas used in a data warehouse are −

* Star Schema
* Snowflakes Schema
* Galaxy Schema

## **Tables in Database**

In a relational database, a table is used to organize the information into rows and columns.

**Example** − A Customer table contains information such as customer id, addresses, phone numbers, and so on as a series of columns.

Each single piece of data is a field in the table. A column consists of all the entries in a single field, such as the telephone numbers of all the customers. Fields are organized as records, which are complete sets of information (such as the set of information about a particular customer), each of which comprises a row.

## **Stored Procedures**

A stored procedure is a series of SQL statements stored in the database in a compiled form and multiple programs can share it. The use of stored procedures can be helpful in maintaining data integrity, data control access and improving productivity.

## **Triggers**

A database trigger is code that is executed in response to certain events on a particular table or view in a database. The trigger is mostly used for maintaining the integrity of the information on the database.

**Database Testing – Data Integrity**

Data Integrity is important in a database. It includes data validation before insertion, updates, and deletion. Triggers must be in place to validate reference table records.

For checking Data Integrity, you need to perform the following operations −

* You need to check major columns in each table and verify if any incorrect data exists. (Characters in name field, negative percentage, etc.)
* Find out inconsistent data and insert them into relevant tables and see if any failure occurs.
* Insert a child data before inserting its parent’s data. Try to delete a record that is still referenced by the data in another table.
* If a data in a table is updated, check whether the other relevant data is updated as well. You need to ensure that replicated servers or databases are in sync and contain consistent information.

# Data Mapping

Data mapping in a database is one of the key concept that needs to be validated by every tester. Usually the testers have to verify the user interface front end field mapping with the corresponding back end database field.

This information is given in Software requirement specification or business requirement specification SRS/BRS document. If mapping is not provided, then you need to check the coding part.

When you take any action in the front end application, there is a corresponding CRUD action get invoked, and tester have to check the every invoked action is successful or not.

## **Key Aspects of Data Mapping**

Given below are the key aspects of Data Mapping −

* To check the fields in the UI/Front end forms and mapped consistently with the corresponding DB table. This mapping information is defined in the requirements documents as mentioned above.
* For any action performed in the front end of an application, a corresponding CRUD ‘Create, Retrieve, Update and delete’ action gets initiated at the back end.
* A tester will have to check if the right action is invoked and the invoked action in itself is successful or not.

## **Steps in Data Mapping Testing**

Given below are the steps followed for Data Mapping Testing −

* **Step 1** − First check for syntax error in each script.
* **Step 2** − Next is to check for table mapping, column mapping, and data type mapping.
* **Step 3** − Verify lookup data mapping.
* **Step 4** − Run each script when records do not exist in destination tables.
* **Step 5** − Run each script when the records already exist in the destination tables.

# Database Testing – Performance

An application with more response time and poor performance can lead to huge problems. Database Load Testing is used to find any performance issues before you deploy your database applications for end users.

Database Load Testing helps you design database application for performance, reliability and scalability. Load Testing of Database applications involves testing the performance and scalability of your Database application with varying user load.

Database Load testing involves simulating real-life user load for the target Database application. It helps you determine how your Database application behaves when multiple users hits it simultaneously.

## **Load Testing**

The primary target of Load Testing is to check if most running transactions have performance impact on the database. In load testing, you need to check the following aspects −

* The response time for executing the transactions for multiple remote users should be checked.
* With normal transactions, you should include one editable transaction to check the performance of the database for these type pf transactions.
* With normal transactions, you should include one non-editing transaction to check performance of database for these type of transactions.
* Time taken by database to fetch specific records should be checked.

## **Stress Testing**

Stress testing is performed to identify the system **breakpoint**. Here the application is loaded in such a way that the system fails at one point. This point is called the breakpoint of the database system. Stress testing is also known as **Fatigue Testing**.

Determining the state of database transactions involves a significant amount of effort. Proper planning is required to avoid any time- and cost-based issues.

The most common stress testing tools are **LoadRunner** and **WinRunner**.

**Database Testing – Tools**

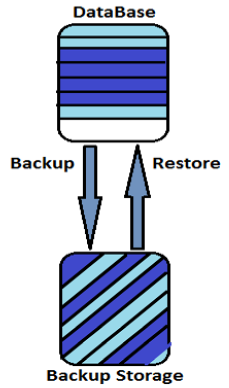
There are various tools provided by vendors that can be used to generate Test data, to manage Test data and perform database testing like Load Testing and Regression Testing.

A few common tools that are used are given below.

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Category & Description** | **Examples** |
| 1 | **Load Testing Tools**  These tools are used to put high usage loads on your database, which enables to determine whether your system's landscape will stand up to your business needs. | Web Performance  Rad View  Mercury |
| 2 | **Data Security Tools**  These tools are used to implement compliance and standards as per the information security regulations. | IBM Optim Data Privacy |
| 3 | **Test Data generator tools**  A tester uses these tools to generate the test data for a database system. These are mostly required when you have huge amount of data and you need sample to perform DB Testing. It is commonly used for Load and Stress testing. | Data Factory  DTM Data Generator  Turbo Data |
| 4 | **Test Data Management Tool**  These tools are used to maintain version control for test data. You have to define the expected results and then you compare it with the actual outcomes of the tests. | IBM Optim Test Data Management |
| 5 | **Tools to perform Unit Testing**  These tools are used to perform regression testing on your database. | SQLUnit  TSQLUnit  DBFit  DBUnit |

# Database Testing – Backup

The most important part of an organizational growth is its data. In case of a system failure, there is a need to restore the data. Back up is an exact copy of the database, which helps you to restore your data in case of any data loss.



Consider a finance company which has data related to its customers such as A/C number, customer names, credit and debits, duration, etc. How would such an organization deal with the pressure of losing such important information in case of a data failure?

This is the reason you back up the data so that in case of any failure of a disk, disk controller, etc. you can rely on the backup to restore it into the database.

## **Types of Data Backups**

There are two types of backup that can be used −

* **Physical Backups** − Physical backup includes taking back up using third-party backup tools like Veritas Net Back, IBM Tivoli Manager or user manager backups using OS utilities.
* **Logical Backups** − Logical backup of database includes taking backups of logical objects like tables, indexes, procedures, etc.

**Example** − One of common tool to take data backup is **Oracle Recovery Manager (RMAN)** that is an Oracle utility to take database backup.

**RMAN** consists of two components −

* **Target database** for which backup is required.
* **RMAN** client is used to run commands to take data backup.

**BACKUP VALIDATE** is used to test if you are able to make a valid backup of database files. It ensures −

* If backup is in place for physical or logical objects of database.
* If regular backups are set up for invaluable data.
* If the backup tool meets the backup requirements of an organization.

# Database Testing – Recovery

**Database recovery testing** is used to ensure that the database is recovered. Recovery testing allows you to find out whether the application is running properly and to check retrieving invaluable data that would have been lost if your recovery method is not properly setup.

You also check if several critical processes are running smooth to ensure that the data recovery will pass smoothly through the testing phase.

You can perform the following checks for database recovery −

* Any errors or mistakes in the backup software and you need to resolve these issues at an earlier stage.
* You need to conduct the recovery testing so that you will know what to do in case of an emergency situation.
* You need to check recovery testing needs so that you can plan for an effective recovery strategy.
* You should also know how you can recover the documents.

You need to run the recovery tests in early phase of the project. This allows you to remove and throw away every type of errors from the system. Here is a list of some of important points, which should be considered at the time of testing −

* Time span when changes or modifications occurs in database system.
* The period by which you want your recovery plan conducted.
* The sensitivity of data in database system. More critical the data is, the more regularly you will need to test the software.

### **Common Steps in Database Backup and Recovery Testing**

In database recovery testing, you need to run the test in the actual environment to check if the system or the data can actually be recovered in case of any disasters and any other unforeseen events in the business environment.

Given below are the common actions performed in Database Recovery Testing −

* Testing of database system
* Testing of the SQL files
* Testing of partial files
* Testing of data backup
* Testing of Backup tool
* Testing log backups