

# Database Testing

Lesson1: Testing Databases  
and DBFit tool

## Lesson Objectives

- Understanding basics of Testing
- Testing RDBMS
  - Why, What, When, Who, and How
- DBFit Tool
- DBFit Tool command reference



1.1: Understanding the basics of Testing

## Why is Testing necessary?

- To test a program, implies adding value to it
  - Testing means raising the reliability and quality of the program.
  - One should not test to show that the program works rather that it does not work.
  - Therefore testing is done with the intent of finding errors.
- Testing is a costly activity

### Why Testing?

#### Why is Testing Necessary?

In SDLC, testing plays a vital role. When one tests a program one adds value to the program, in turn raising the quality and reliability of the program.

When we say “reliable”, it implies finding and removing errors. Hence one should not test a program to show that it works, but to show that program does not work.

Testing cannot guarantee against software problems or even failures but it can minimize the risks of faults developing once the software is put to use.

Typically when testing, one should start with assumptions that the program contains errors and the test the program to find as many errors as possible.

Testing is a costly activity. A test, which does not find an error is a waste of time and money. “A test case that finds an error is a valuable investment”.

1.1: Understanding the basics of Testing

## What is Unit Testing?

- The process of testing the individual subprograms, subroutines, or procedures to compare the function of the module to its specifications is called unit testing
  - Unit testing is a relatively inexpensive and an easy way to produce better codes
  - Unit testing is done with the intent that a piece of code does what it is supposed to do

### Why Testing?

#### What is Unit Testing?

There are various phases in testing. However, in this course we are mainly concentrating on unit testing. Testing individual subprograms or procedure to compare the functions of the module to its required specifications is Unit Testing.

This is an inexpensive activity and a very easy way to produce better code. In other words, unit test is a small piece of code that is written by the developer that will exercise a specific area of functionality of the code being tested.

For example: You write a functionality to sort a list and then check if the sort works. You then modify the functionality to accept the sort order, as well, and then you test this newly added functionality.

The point to note here is that while doing Unit testing, the developers are not worried about verification and validation of the program. It is just that the functionality should be running as required.

Unit Testing is also called as Test-driven Development (TDD). A significant advantage of TDD is that it enables you to take small steps while writing software. Most of you will be already doing some amount of unit testing in an ad hoc manner.

1.1: Understanding the basics of Testing

## What is Test-driven Development( TDD)?

- The Test-driven development, also called Test-first development, is a technique in which you write tests before writing the application functionality.
- TDD in database involves 3 steps
  - Database refactoring
  - Database regression testing
  - Database integration

### What is Test-driven Development?

Developers can use a test-driven development with database schema just as they would use it with any application code. Implementing test-driven database development involves three simple steps: database refactoring, database regression testing, and continuous database integration.

Database refactoring: without changing its semantics developers make a simple change to a database to improve the design.

Database regression testing : developers run a comprehensive test suite that validates the database regularly-ideally, whenever developers change the database schema or access the database in a different way.

Database integration: developers rebuild and retest the database schema whenever it changes.

1.2: Understanding RDBMS Testing

## Why Perform RDBMS Testing?

- Testing of database is done because:
  - Quality data is an important asset
  - Databases are used to implement mission critical business functionality
  - Current approaches for testing RDBMS are inefficient
  - Testing provides a concrete feedback

### Why Perform RDBMS Testing?

RDBMSs often persist mission critical data, which is modified by many applications and in the very likelihood have thousands of users. The users may access, update, delete or append to the database. The modified database should be error free. To make the database error free and to deliver the quality product, testing the database is necessary. RDBMS are not only tabular databases that are used to store data that can be easily queried but they also implement important functionality in the form of databases objects i.e procedure, functions, triggers.

When testing database the key factors to remember is testing actual data, integrity of data, functionality testing of database application.

The reasons for Database testing are as follows:-

Quality data is an important asset :- In a survey approximately 95% believe that data is an important asset. But less than 50% had a test suite for data validation. Hence it makes sense to invest the effort required to validate the quality of data via effective testing.

Databases are used to implement mission critical business functionality.


Current approaches for testing RDBMS are inefficient: Presently we develop a database by setting up database, write code to access the database, run code, and do SELECT operation to find the query results. Although visual inspection is a good start, it may help us to find problems but not prevent them.

Testing provides a concrete feedback :having a good test suite in place provides a feedback and ensures quality of the source data

1.2: RDBMS Testing

## Factors to test in a Database

- Black box testing
  - I/O validation
  - Error handling
  - Table structure validation
  - Testing stored and modified data
- White box testing
  - Testing structure of stored procedures and functions
  - Testing views
  - Testing data constraints :

 Capgemini  
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 7

### Factors to test in a Database:

Once you have understood the reason behind testing databases. The next question is what should we test in a database. On the slide you can see that testing for databases is also categorized into Black Box Testing and White Box Testing.

Black Box Testing comprises of:

I/O validation :- Testing will help in validating both incoming and outgoing data values from stored procedures, queries.

Error handling:- Testing helps in identifying if error handling has been done appropriately

Table Structure Validation:- Validating the relationships between rows in different tables.

Testing Stored and modified data:- Testing for existing data stored. Modification of data within tables may introduce new errors, which can be easily detected by regression testing.

White Box Testing comprises of:

Testing structure of Stored Procedures and functions: Entire schema can be tested by various testing methods. We can refactor the database into structures that give optimum performance without changing the semantics. We can also do regression testing, which helps in addressing new needs of customers and fix legacy database design problems



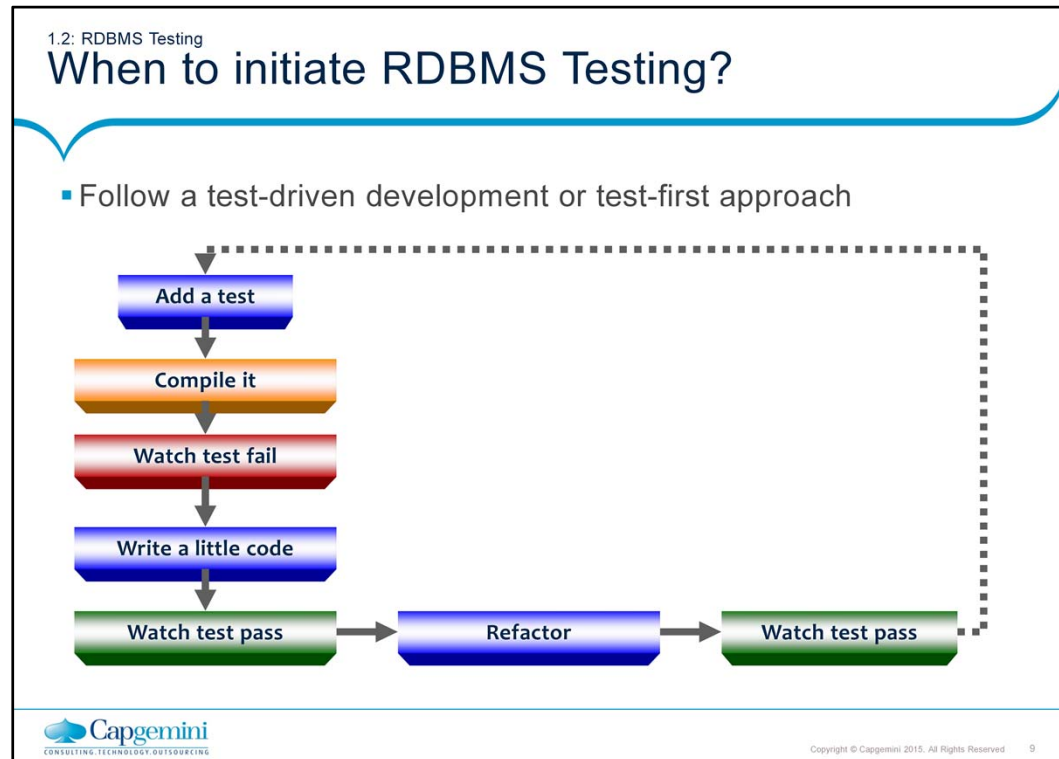
Factors to test in a Database (contd.):

Testing Views: This means testing the 3 layers of the database i.e Physical, Conceptual and logical.

Testing Data Constraints: Testing Data constraints would involve testing for null values, handling single quote in string field, wrong datatype values and so on..

Overall we can say that mostly we do apply regression testing mechanism on database, but that helps in unit testing of stored procedures, function and triggers. Improving data quality is the main goal, which involves correctness, completeness and consistency.





### When to Initiate the RDBMS Testing?

Database testing involves initial testing and database refactoring. This strategy can be applied simultaneously to both the application code and the database schema. Testing of databases is an ongoing process, which is done not only during the release but also during the development.

While database testing follows the Test-First Approach wherein a test is written first and then the code is written, which will fulfill this test.

As depicted in the diagram on the slide the step by step approach is:

Step 1: A test is written/added for just enough code to fail

Step 2: Compile

Step 3: Execute the test to make sure that the test does fail

Step 4: Functional code is then updated so that the test passes.

Step 5: Tests are run again. Tests should pass


Step 6: If tests fail, update functional code again and run the test again

Step 7: Once the test passes, start from Step 1 again

1.2: RDBMS Testing

## Who should test?

- During Development
  - Application developers
  - DBAs
- During release cycle
  - Testers

 Capgemini  
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 10


### Who is best suited to test the Database?

The primary responsibility of doing database testing during the development cycles lies with application developers and DBAs. They typically have to work together and if they are following a TDD approach the implication is that they will be doing unit testing on a continuous basis. During the release cycle the testers will be responsible for the final testing of the overall system, hence will also be doing the database testing.

1.2: RDBMS Testing

## How to test?

- Primarily two activities are involved in database testing:
  - Organizing sandboxes
  - Developing test cases

 Capgemini  
CONSULTING TECHNOLOGY OUTSOURCING


Copyright © Capgemini 2015. All Rights Reserved 11

These two key activities have been discussed in detail in subsequent slides.

1.2: RDBMS Testing

## Database Sandbox Types

- Database sandboxes are of three types
  - Functionality sandbox – Check new functionality or refactor existing functionality
  - Integrated sandbox – integrate all sandboxes
  - QA sandbox – Acceptance testing of sandboxes

Capgemini  
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 12

### Database Sandboxes:

A sandbox is basically a technical environment whose scope is well defined and respected. It is ideal to ensure that developers have their own “sandboxes” to work in. Each sandbox has a copy of database.

Let us take a look at the three types of sandboxes:-

**Functionality Sandbox:** In this we check for new functionality ,refactor the existing functionality of database and validate the changes through testing. The tested sandbox is then passed to integrated sandbox.

**Integrated Sandbox:** In this all the sandboxes are integrated and then the system is tested. In other word the system is rebuild and tested to ensure nothing is broken, if it is so then we go back to the development sandbox.

**QA Sandbox:** Sandboxes are sent for acceptance testing once the system is tested. This will ensure the quality of the database.

The basic advantage of sandboxes are that they help to reduce the risk of technical errors adversely affecting a larger group of people than is absolutely necessary at the time.

1.2: RDBMS Testing

## Developing Test Cases

- Database tests are typically three step processes
  - Setup the test: Put the database into a known state before running tests against it
  - Run the test: Using a testing tool, run database test just like running application tests
  - Check the results: Compare the expected results against current values in the database obtained through “table dumps”

Developing test case involves three steps:

**Setup the test:** To successfully test your database you must know the exact state of the database. Put the database to a known state before running the tests against it. There are two common methods to put database into a known state:

**Fresh Start** – wherein you rebuild the database including both creation of the schema as well as loading of initial test data for every major test run.

**Data re-initialization**-For testing in developer sandboxes, something that you should do every time you rebuild the system, you may want to forgo dropping and rebuilding the database in favor of simply reinitializing the source data. You can do this either by erasing all existing data and then inserting the initial data vales back into the database, or you can simple run updates to reset the data values. The first approach is less risky and may even be faster for large amounts of data

An important part of writing database tests is the creation of test data. The sources of test data are External test data like flat files or XML files, test scripts, test data with known values and real world data.

**Run the test :** The test cases are run similar to running application tests.

You can follow a traditional approach or advanced approach using testing tool. To follow a proper advanced approach of Test Case Execution you need to do a schematic preparation for Database Testing. This can be done by generating a list of database tables, stored procedures, triggers, rules and so on. This helps in determining the scope of testing.

## Developing Test Cases

Analyzing the schema will help us determine the following:

- Can a certain field value be Null?
- What are the allowed or disallowed values?
- What are the constraints?
- Is the value dependent upon values in another table?
- What are user defined data types?
- What are primary key and foreign key relationships among tables?

2. Analyzing how the stored procedures, triggers, defaults and rules work will help us determine the following:

- What is the primary function of each stored procedure and trigger? Does it read data and produce outputs, write data or both?
- What are the accepted parameters?
- What are the return values?
- When is the stored procedure called and by whom?
- When is a trigger fired?

Check the results: Actual database test results and expected test results are compared For eg:

Consider the following function


```
CREATE or REPLACE FUNCTION f_is_leapyear (ai_year number)
RETURN number AS
flag number;
BEGIN
-- if year is illegal (null or -ve), return -1
IF (ai_year IS NULL) OR (ai_year <=0) then
flag:=-1;
end if;
IF (mod(ai_year,4) = 0) AND (mod(ai_year,100) <> 0)
OR (mod(ai_year,400) = 0) then
flag:=1; --leap year
end if;
flag:=0; --Not a leap year
return flag;
END;
```

Following test cases are derived for the above piece of code

1.3 DB Testing Tools

DBFit tool

- Unit testing – DBFit, AnyDbTest, and SQL unit
- Load testing – Mercury interactive and Rational Suite Test Studio
- Test data generator – Data factory and Turbo data

Capgemini  
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 15

#### Types of Database Testing Tools:


There are several critical features, which you need to successfully test RDBMSs. The testing tools should support the language that you're developing in. For example, for internal database testing if you're a Microsoft SQL Server developer, your T-SQL procedures should likely be tested using some form of T-SQL framework. Similarly, Oracle DBAs should have a PL-SQL-based unit testing framework. You also need tools, which help you to put your database into a known state, which implies the need not only for test data generation but also for managing that data.

On the slide we have listed some examples for testing tools required for testing various aspects of database.

1.3 DB Testing Tools

About DBFit

- DBFit makes test-driven database development easy
- DBFit is a set of FIT fixtures that enable the tests to execute directly against the database
- DBFit tool is for the:
  - Database developers – a neat unit testing tool for stored procedures and database objects
  - .Net/Java developers – a set of FIT fixtures that enable tests to directly execute against a database

Capgemini  
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 16

#### The DBFit Tool:

DBFit makes test driven development easy. It is a product from gojko.net. It is a set of FIT fixtures that enable the tests to be executed against a database. This allows developers to manipulate database objects in a relational tabular form making database testing and management much easier.

You can look at DBFit in a different perspective depending on whether you are primarily working in a database environment or in a .Net/Java environment.

DBFit for

Database developers – a neat unit testing tool for stored procedures and database objects, which allows to write database tests in a tabular, relational form without the need of learning or using an object oriented language.


.Net/Java developers – a set of FIT fixtures that enable tables to execute directly against a database.



1.3 The DBFit Tool

## Features

- Regression testing for SQL statements and queries
- Functional testing for Stored Procedures and Functions
- Techniques to make writing test scripts easier and more efficient
- Support for Oracle, MySQL, and SQL Server 2000 & 2005

 Capgemini  
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 17

#### Features of the DBFit Tool:

DBFit is reflected in two main project goals i.e support efficient database acceptance and unit testing by providing database developers a good tool to express and manage tests in relational language, without knowledge of .Net/Java.

The features of DBFit are listed on slide. DBFit provides automatic transaction control, building regression tests for queries to make writing test scripts easier.

## 1.3 The DBFit Tool

## Installation and Setup

- DBFit can be executed either through Java or .Net
- The following table shows DBFit supportSupport for Oracle, MySql, and SQL Server 2000 & 2005

| Database   | Java | .Net |
|------------|------|------|
| Oracle     | Yes  | Yes  |
| SQL Server |      | Yes  |
| MySQL      | Yes  |      |

Install and setup DBFit:

DBFit is an extension to FitNesse, so you will use the FitNesse Server to manage and run DBfit tests. We will first now understand the setup procedure for .Net

1.3 The DBFit Tool

## Installation and Setup (contd..)

- For using Java version, JRE 5, or later
- For using .Net version, both JRE & Microsoft's .Net Framework 2, or later
- Download the dbfit-complete.zip package from <http://sourceforge.net/projects/dbfit>
  - This includes DBFit libraries and all dependencies

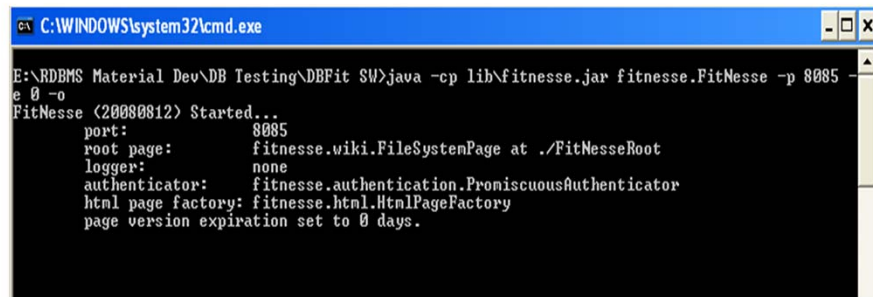
### Install and setup DBFit (contd.):

For working with DBFit you can either choose Java version or .Net version. The choice could also depend on which database you are using. As mentioned on the slide have the appropriate runtime installed on your machine. The DBfit-complete is a zip file, unzip in this package somewhere on your disk. This package includes DBFit libraries and all required dependencies including the .Net and Java test runners for Fitnesse and the FitNesse server itself.

## 1.3 The DBFit Tool

## Starting DBFit

- FitNesse works as a web application with its own Web Server
- The FitNesse by default starts on port 8085
- Run startfitness.bat to start the Server



```
C:\WINDOWS\system32\cmd.exe
E:\RDBMS Material Dev\DB Testing\DBFit SW>java -cp lib\fitnesse.jar fitnesse.FitNesse -p 8085
FitNesse (20080812) Started...
port: 8085
root page: fitnesse.wiki.FileSystemPage at ./FitNesseRoot
logger: none
authenticator: fitnesse.authentication.PromiscuousAuthenticator
html page factory: fitnesse.html.HtmlPageFactory
page version expiration set to 0 days.
```

### Starting the DBFit Tool:

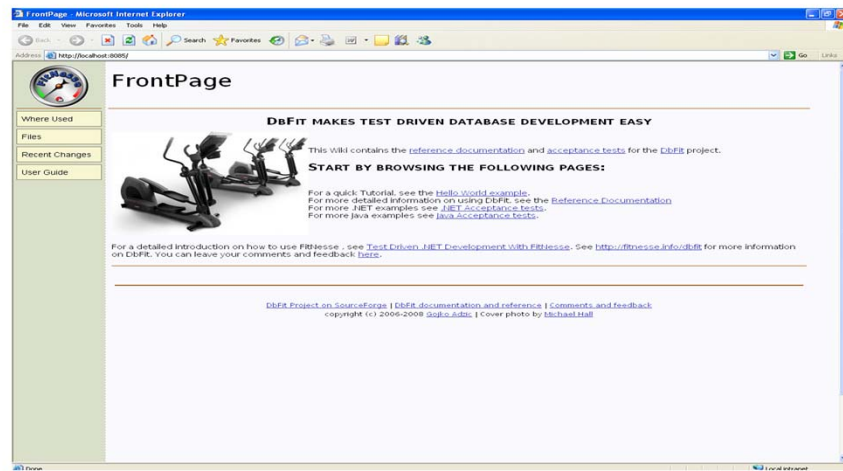
FitNesse works as a Web application with its own web server, which by default runs port 8085. If this port is already in use on your machine, you can edit the startFitNesse.bat and change the port to any other free port.

Once you run the startFitNesse.bat you can see the command window as shown on the slide.

## 1.4 Test Cases

## Creating a Test Case: Steps

- In the browser open <http://localhost:8085>



## Steps to Create a Test Case:

Let us create our first test quickly and check if everything has been setup properly. Open <http://localhost:8085/FirstTest> in the browser. It is the name of the test you are about to create. Since it does not exist you will see the page as shown on the slide. In the text box you can type in the code for your test. Note the page name "FirstTest" is given in CamelCase. FitNesse is really strict about that. All page names have to be provided on similar lines.

Here are some agreeable page names:

HelloWorld

TestPerson

TestEmp

IsPaymentWorking

Here are some not agreeable page names:

helloworld – No capital letters

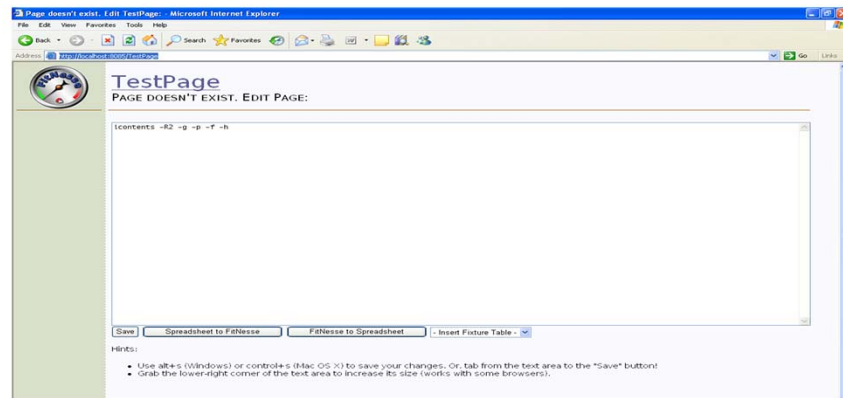
Testperson – only one capital letter

testEmp – starts with lowercase

## 1.4 Test Cases

## Creating a Test Case: Steps (contd..)

- Step 1: Creating a new test page
  - Open <http://localhost:8085/TestPage>



## Steps to Create a Test Case:

Let us create our first test quickly and check if everything has been setup properly. Open <http://localhost:8085/FirstTest> in the browser. It is the name of the test you are about to create. Since it does not exist you will see the page as shown on the slide. In the text box you can type in the code for your test. Note the page name "FirstTest" is given in CamelCase. FitNesse is really strict about that. All page names have to be provided on similar lines.

Here are some agreeable page names:

HelloWorld

TestPerson

TestEmp

IsPaymentWorking

Here are some not agreeable page names:

helloworld – No capital letters

Testperson – only one capital letter

testEmp – starts with lowercase


1.4 Test Cases

## Creating a Test Case: Steps (contd..)

- Step 2: Setting up the environment
  - To load correct libraries
    - For Java version

```
!path lib/*.jar
```
    - For .Net version

```
!define COMMAND_PATTERN {%m %p}
!define TEST_RUNNER {dotnet2\FitServer.exe}
!define PATH_SEPARATOR {:}
!path dotnet2\*.dll
```

 Capgemini  
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 23

### Steps to Create a Test Case (contd..)

In order to load the DBFit extension into FitNesse, the test pages have to load the correct libraries. Paste the content appropriately as mentioned on the slide for using Java version or .Net version.

1.4 Test Cases


Creating a Test Case: Steps (contd..)

■ Step 3: Connecting to the database

```
!|dbfit.OracleTest|  
!|Connect|192.168.224.26:1521|testuser|testuser|trgdb|
```

■ Step 4: Testing a Simple Query

```
!|Query| select 'test' as x from dual|  
|X|  
|test|
```

Capgemini  
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 24

#### Steps to Create a Test Case (contd..)

The next step is to connect to the database. DBFit requires two commands to do so. As shown on the slide the first line specifies the database type and second defines connection properties. The snippet on the slide shows how to connect to the Oracle database, which indicates that it is using Oracle database driver. For SQLServer 2005 you should use SQLServerTest and for MySql use MySqlTest.

The next command is the Connect command that can be written in various formats. The default method is to use 4 parameters:

```
!Connect|SERVICE_NAME|USER_NAME|PASSWORD|DATABASE_NAME|
```

SERVICE\_NAME is the host, instance name or service name, depending on the type of driver used. If you are using Oracle in .Net, this can be a TNS name and the fourth argument can be omitted. In Java version, Oracle Thin driver is used hence you have to specify the host name and you will have to specify the database SID as fourth argument. Now to test a simple query. In DBFit that is done with the query command. The second argument after 'Query' should contain a valid query for executing. This is followed by resultset structure continuing on the next line. The resultset includes the names of the columns that we want to inspect. All rows after that contain the expected result.

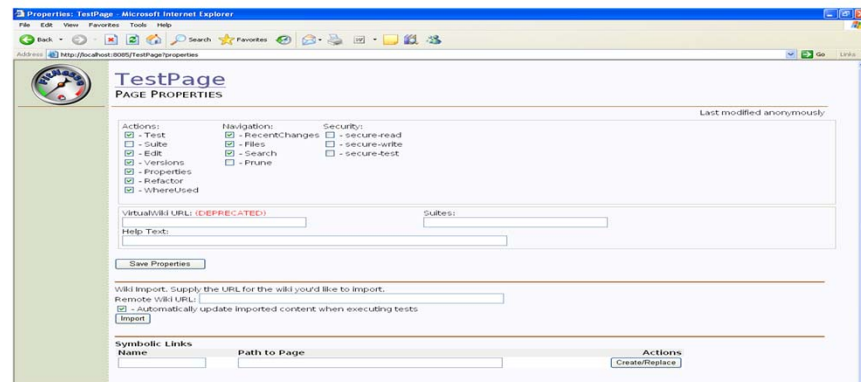
The example on the slide shows a simple Oracle query and its corresponding resultset.



## 1.4 Test Cases

## Creating a Test Case: Steps (contd..)

- Step 5: Running the test case
  - Save the page contents.
  - Next you need to indicate that it is a test page. Click on properties button on the left. Check the Test checkbox and save



## Steps to Create a Test Case (contd..)

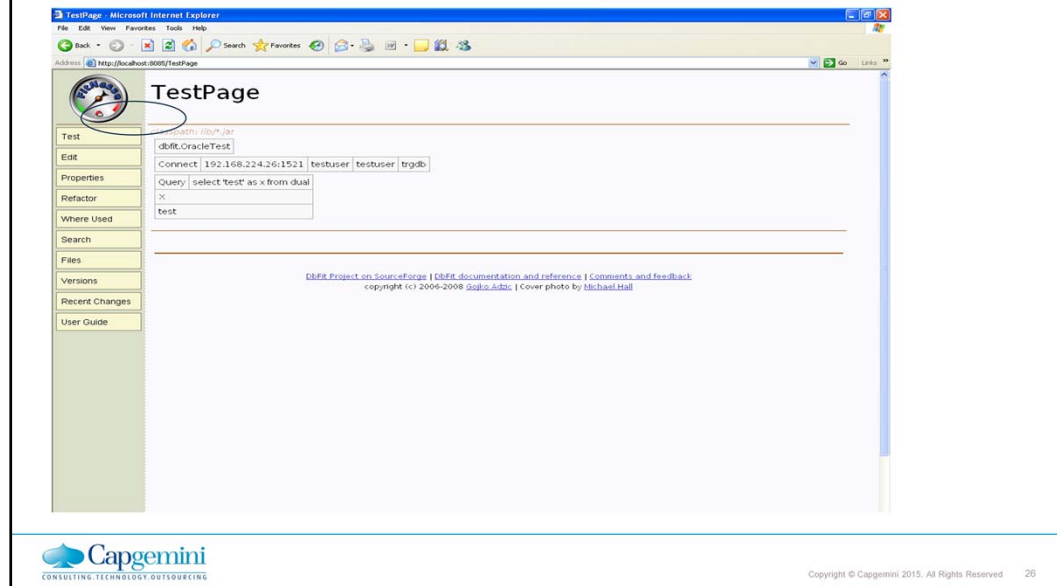
Once the code of the test is written you are ready to run the test case.

Before that Save the page contents by clicking on the Save button provided. DBFit saves the page content and displays it in tabular format. After this you need tell FitNesse that it is test page. Click on the properties button on the left. It opens the properties page shown on the slide. Check the Test checkbox and Save the properties.

Page properties define what the user can do with the page.

## 1.4 Test Cases

## Creating a Test Case: Steps (contd..)



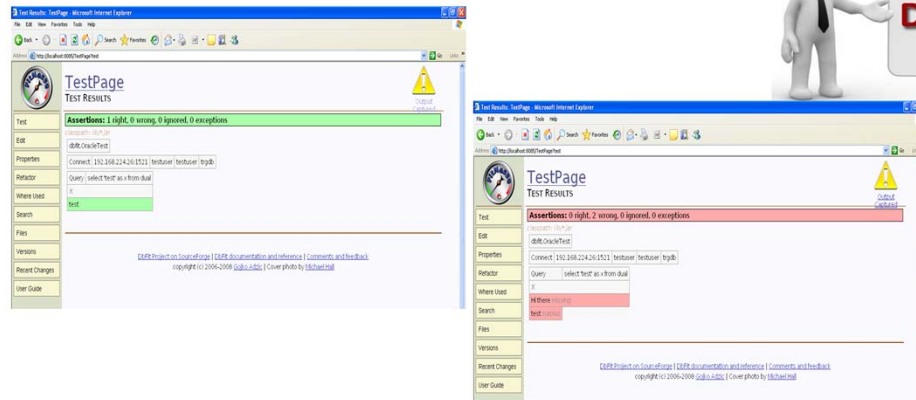
## Steps to Create a Test Case (contd..)

After saving the page properties, the page reloads and you will notice a button 'Test' on the left. Click on it to execute your test. You should see the result as test pass or fail.

## 1.5: Demonstrating the use of the DBFit Tool

## Demo

- Setup of DBFit
- Creating and executing the first DBFit test case



1.6 DBFit Commands

## Writing a Query

- Query: Write query against database and check the SQL query results

```


!!Query| select ename from emp where empno=7788|
|ename|
|SCOTT|

```
- OrderedQuery: Provides order checking for SQL query results

```

!!OrderedQuery| select deptno,dname from dept order by deptno|
|deptno|dname|
|10|ACCOUNTING|
|20|RESEARCH|
|30|SALES|
|40|OPERATIONS|

```


Copyright © Capgemini 2015. All Rights Reserved 28

### Writing a Query:

The Query command is termed as RowFixture in DBFit, which compares rows in the test data to objects in the system under test and uses SQL Query results. The query to be executed is the first fixture parameter, which follows the Query command itself. It is followed by column names on the next line and all the subsequent lines will be the expected results of the query. The example is shown on the slide. Query will report any rows that exist in the actual result set but are not in the FitNesse table and they are marked as surplus. It also reports the rows that are missing in the result set but are present in the FitNesse table and are marked as missing. All the matched rows are then checked for values in columns and any differences will be reported in individual cells. By default the row order is ignored by Query. OrderedQuery is provided for order checking.

Note : Tests are described in FitNesse as some sort of coupling of inputs and expected output. These couplings are expressed as some sort of variation of a decision table. The FitNesse tool supports several of these variations, ranging from literal decision tables to tables that execute queries to tables that express testing scripts (i.e. a literal ordering of steps that must be followed to reach a result). The most generic form is a fully free-form table that can be interpreted in any way the test designers like. All tests are expressed in the shape of some sort of table,


1.6 DBFit Commands

## Writing a Query

- Passing parameters to queries
  - SetParameter : Set the query parameters using this command

```
|Set Parameter|dno|10|  
!!Query| select deptno,dname from dept where deptno=:dno|
```
  - Storing resultset values to be re-used as parameters

```
!!query| select sysdate as newdate from dual|  
|newdate?|  
|>>dateval|  
!!Query| select empno from emp where hiredate=:dateval|
```

 CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 29

### Writing a Query (contd.)

You can use Set Parameter to pass query parameters. DBFit supports Database specific syntax to pass parameters. For Oracle you use :paramname.

You can also store elements of the result set into parameters to re-use them later in stored queries and stored procedures. Use >>parameter to store a cell value into a parameter.

If you use the query just to populate values into parameters, then make sure to mark columns with the question mark to avoid row matching. There will be nothing to match the rows with in this case, so a proper comparison would fail.


Consider the second example shown on the slide where the system date and time is collected to be used as parameter in other query.

1.7 The DBFit Tool

## Using DML Statements

- Insert: Builds an insert command and executes once for each row of the table

```
!|Insert|student|  
|rollno|name|  
|101|ABC|  
|102|XYZ|  
  
!|Query|Select * from student|  
|rollno|name|  
|101|ABC|  
|102|XYZ|
```

 Capgemini  
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 30

### DML Statements:

The Insert command can be executed as shown on the example on the slide. It builds an insert command from the parameters and executes the insert once for each row of the table. The view or the table name is given as the first fixture parameter. The second row contains column names and subsequent rows contain data to be inserted.


1.7 The DBFit Tool

## Using DML Statements

- Update: Builds an update command and executes the update once for each row

```
!|Update|student|
|name|=|rollNo|
|PQR|101|

!|Query|Select * from student|
|rollNo|name|
|101|ABC|
|102|XYZ|
```

 Capgemini  
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 31

### DML Statements (contd.):

Update allows you to quickly script data updates. It builds an update command from the parameters in a data table and executes the update once for each row. Observe the syntax in the example given on the slide. Columns ending with “=” are used to update records, the new data value is specified on the next line and columns without “=” on the end are used to select rows i.e the where clause value. The view or table name is given as the first fixture parameter. The second line contains the column names and all subsequent rows contain data to be updated or queried.

The example on the slide updates the Student name where rollNo=101. The full script is as follows:-

1.8 Procedures and Functions

## Executing Procedures and Functions

- Execute Procedure : Executes a stored procedure or function for each row, binding input/output parameters to columns.

```


!|Execute Procedure|Get_Name|
|empid|name?|
|7369|SMITH|
|7499|ALLEN|

!|Execute Procedure|Get_Name_Func|
|empid|?|
|7369|SMITH|
|7499|ALLEN|

```

Calling Procedure

Calling Function

 CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 32

### Executing Procedures and Functions:

You can use the Execute Procedure command to execute procedure or functions. It executes a stored procedure or function for each row of data table, binding input/output parameters to columns. The procedure name should follow the Execute Procedure command as the next parameter. The next line contains the names of parameters, the output parameters are followed by a question mark. All the subsequent rows are data rows, which contain input parameters values and expected values of output parameters. The first part of the snippet shown on the slide tests a procedure.

The order of parameters or case is not important. You can store output values into parameters with the >> syntax.

To use IN/OUT parameters, you'll need to specify the parameter twice. Once without the question mark, when it is used as the input and one with the question mark when it used as output For eg:

```

|Execute Procedure|format_phone|
|phone_no|phone_no?|
|02027105000|020-2710500|

```

If there are no output parameters for the procedure then ExecuteProcedure command has no effect on the outcome of the test-unless an error occurs during processing.

The second part of the snippet on the slide tests a function. When a function is getting called, then a column containing just the question mark is used for function results.



## Testing Exceptions

- A test would fail if any database exception occur
- To test exceptions you can use 'Execute procedure expect exception' variant of 'Execute procedure'

```
!|Execute Procedure expect exception|Get_Name_Exception|  
|empid|name?|  
|1001|Name not found|
```

The 'Execute procedure expect exception' variant of the 'Execute procedure' command to test exceptions. You can also provide an optional exception code as the third argument. If exception occurs the test should pass. If the exception code is specified than the actual error code is also considered for fail or pass of the test.



#### Executing Procedures and Functions (contd..)

**Execute:** It executes any SQL statement. The statement is specified as the first parameter. There are no additional rows required for this command. Query parameters can be used in the DB-specific syntax. Currently all parameters are used as inputs and there is no option to persist any statement outputs.

Examples of execute command

```
|Execute|Create table student(rollno char(3),name varchar2(20))|
```

```
|Execute|Insert into student values('101','John')|
```

#### Best Practices for DBFit

Once you are ready to start creating the unit tests, you can follow some practices so that your test are easily maintainable

**Initializing tests:** It is a good practice to initialize the database connection on a Setup Page. You can run the '!!ClearParameters' command to clear any symbols or values which are persisted.

## Lab

- Testing Queries
- Testing Stored Procedures



## Summary

- Understanding basics of Testing
- Testing RDBMS
  - Why, What, When, Who, How
- DBFit Tool
- DBFit Tool command reference



## Review Question

- Question1: Which of the following are unit testing tools:
  - Option 1: AnyDBTest
  - Option 2: SQL Unit
  - Option 3: Both the above
- For using Java version of DBFit you need both JRE 5 and Microsoft .Net Framework
  - True/ False
- Testing is a \_\_\_\_\_ activity



## Review Question

- Question1: The FitNesse server by default works on \_\_\_\_\_ port
- Option 1:8080
- Option 2:8085
- Option 3:8090

