1. Introduction

Databases are used with every application, whether it is commercial or scientific. Therefore it is essential that these applications work correctly. Testing these database applications correctly and effectively is a challenge because the task of testing is very much tough for even a simple application. In case of database applications the testing is very difficult and time consuming process. The testing must be effective and it is much more important to know where we have to stop, which can be termed as adequacy criteria for testing.

Many testing methods and tools are available for testing SQL driven applications, also several authors have suggested various adequacy criteria for the same.

1.1 Overview of Testing:

Software testing is an activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. Although crucial to software quality and widely deployed by programmers and testers, software testing still remains an art, due to limited understanding of the principles of software. The difficulty in software testing stems from the complexity of software: we cannot completely test a program with moderate complexity. Testing is just more than debugging. The purpose of testing can be quality assurance, verification and validation or reliability estimation. Testing can be used as a generic metric as well. Correctness testing and reliability testing are two major areas of testing. Software testing is tradeoff between budget and quality.

Software testing is the process of executing the program with the intent of finding errors or it involves an activity aimed at evaluating an attribute or capability of a program or system and determining that it meets required result.

Software bugs will almost occur in any software module with moderate size, not because programmers are careless or irresponsible, but because the complexity of software is generally intractable and human have only ability to manage complexity. Discovering the design defects in software is equally difficult for the same reason of complexity. Because the software are not continuous, testing boundary values are not sufficient to guarantee correctness. All the possible
values need to be tested and verified, but complete testing is infeasible. A further complication has to do with the dynamic nature of programs. If a failure occurs during preliminary testing and the code is changed, the software may now work for a test case that it didn't work for previously. But its behavior on pre error test cases that it passed before can no longer be guaranteed.

An interesting thing about software testing is that, "Every method you used to prevent or find the bugs leaves a residue of subtler bug against which those methods are ineffectual". But this alone will not guarantee to make the software better, because the Complexity Barrier states that, Software complexity grows to the limits of our ability to manage that complexity.

Software testing is not mature. It still remains an art, because we still cannot make it a science. We are using the testing methods which were invented before 10 to 15 years. Some of them are crafted methods or heuristics rather than good engineering method. Software testing is quite costly, but not testing software is even more expensive. We can never be sure that a piece of software is correct or a specification is correct.

Regardless of various limitations testing is an integral part in software development. it is broadly deployed in every phases in the software development cycle. Typically more than 50% of the development time is spent in testing.

Testing is usually performed:

- To Improve Quality
- For Verification And Validation
- For Reliability Estimation

There is a plethora of testing methods and testing techniques, serving multiple purposes in different life cycle phase. Classified by purpose, software testing can be divided into: Correctness Testing, Performance Testing, Reliability Testing and Security Testing. Classified by lifecycle phase, software testing can be classified in to following categories: Requirements Phase Testing, Design Phase Testing, and Program Phase Testing, evaluating test results, Installation Phase Testing, Acceptance Phase Testing and Maintenance Testing. By scope, software testing can be categorized as: Unit Testing, Component Testing, Integration Testing, and System Testing.
For performing above mentioned test the testing techniques are divided into following two main types as,

1.3 **Manual Testing:**

In this type of testing, the programs are tested manually by checking the flow of control in the module. Providing various inputs to the program and check the output. Test cases are written after testing the module, depending on which the programmer can make the corrections in the programs. A different tester uses different methods and techniques to write the test cases. The manual testing mainly divided in to:

I. **Black Box Testing:**

The black box approach is a testing method in which test data are derived from the specified functional requirements without regard to the final program structure. It is also termed data driven or requirement driven testing. Since only the functionality of the software module is concern, black box testing also mainly referred to functional testing - a testing method that emphasis on executing the functions and examination of their input and output data. The tester treats the software under test as a black box only the inputs, outputs and specifications are visible and the functionality is determined by observing the outputs to corresponding inputs. In this type of testing various inputs are exercised and the outputs are compared against specification to validate the correctness. All test cases are derived from the specification. No implementation details of the code are considered. It is obvious that more we have covered in the input space, the more problems we will find and therefore we will be more confident about the quality of the software,

Combinatorial explosion is the major problem in functional testing. In this we can never be sure whether the specification is either correct or complete. Due to limitations used in language used in the specification, ambiguity is often inevitable. The research in black box testing mainly focuses on how to maximize the effectiveness of testing with minimum cost usually the number of test cases.
II. **White Box Testing:**

Contrary to black box testing, software is viewed as a white box or glass box in white box testing, as the structure of the software under test are visible to the tester. Testing plans are made according to the details of the software implementation, such as programming language, logic and style. Test cases are derived from the program structure. White box testing is also called glass box testing, logic driven testing.

Control flow testing, loop testing and data flow testing, all maps the corresponding flow structure of the software into a directed graph. Test cases are carefully selected based on the criterion that all the nodes or paths are covered or traversed at least once. By doing so we may discover unnecessary "dead" code that is of no use or never get executed at all, which can not be discovered by functional testing.

1.4 **Automated Testing:**

We know that the testing can be very costly. Automation is a good way to cut down time and cost. Software testing tools and techniques usually suffer from a lack of generic applicability and scalability. The reason is straight forward. In order to automate the process, we have to generate some oracle from the specification and generate the test cases to have a full scale system that has achieved this goal. In this type tools are available in the market for the testing various types of applications, tools like Winruner, Loadruner, QTP, Test Director are general tools, Agenda for DBMS application. Also now a day the large scale companies are using their own testing tools for their own.