DATABASE ADMINSTRATION Level III

Based on August 2011, Version 3 Occupational Standards (OS) and Curriculum

Module Title: **Testing Physical Database Implementation**

LG Code: EIS DBA3 M10 1220 Lo (1-3) LG (37-39)

TTLM Code: EIS DBA3 TTLM10 1220v1

December, 2020
Bishoftu, Ethiopia

Table of Contents

L #37	Lo#1. Undertake database management system modeling

Instruction sheet

This learning guide is developed to provide you the necessary information regarding the following content coverage and topics:

- Reviewing Database prototype to determine acceptance criteria and performance standards
- Loading test data according to the technical sequence detailed in documentation
- Generating a test schedule for the database of tasks

This guide will also assist you to attain the learning outcomes stated in the cover page. Specifically, upon completion of this learning guide, you will be able to:

- Review Database prototype to determine acceptance criteria and performance standards
- Load test data according to the technical sequence detailed in documentation
- Generate a test schedule for the database of tasks

Learning Instructions:

Read the specific objectives of this Learning Guide.

1. Follow the instructions described below.
2. Read the information written in the “Information Sheets”. Try to understand what are being discussed. Ask your trainer for assistance if you have hard time understanding them.
3. Accomplish the “Self-checks” which are placed following all information sheets.
4. Ask from your trainer the key to correction (key answers) or you can request your trainer to correct your work. (You are to get the key answer only after you finished answering the Self-checks).
5. If your performance is satisfactory proceed to the next learning guide,

1.1 Reviewing Database prototype to determine acceptance criteria and performance standards

Introduction

A Database model defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system. While the Relational Model is the most widely used database model, there are other models like Hierarchical Model, Network Model, Entity-relationship Model, and Relational Model.

Hierarchical Model: It organizes data into a tree-like-structure, with a single root, to which all the other data is linked. The hierarchy starts from the Root data, and expands like a tree, adding child nodes to the parent nodes. In this model, a child node will only have a single parent node. In this model data is organized into tree-like structure with one-to-many relationship between two different types of data, for example, one department can have many courses, many professors and many students.

In this model each entity has only one parent but can have several children. At the top of hierarchy there is only one entity which is called Root. The tables of this model take on a child-parent relationship. Each child table has a single parent table, and each parent table can have multiple child tables. Child tables are completely dependent on parent tables; therefore, child tables can exist only if its parent table does. It follows that any entries in child tables can only exist where corresponding parent entries exist in parent tables. The result of this structure is that the hierarchical database model supports one-to-many relationships. Every task is part of a project, which is part of a manager, which is part of a division, which is part of a company. So, there is a one-to-many relationship between companies and departments because there are many departments in every company. The disadvantages of the hierarchical database model are that any access must originate at the root node. You cannot search for an employee without first finding the company, the department, the employee's manager, and finally the employee.

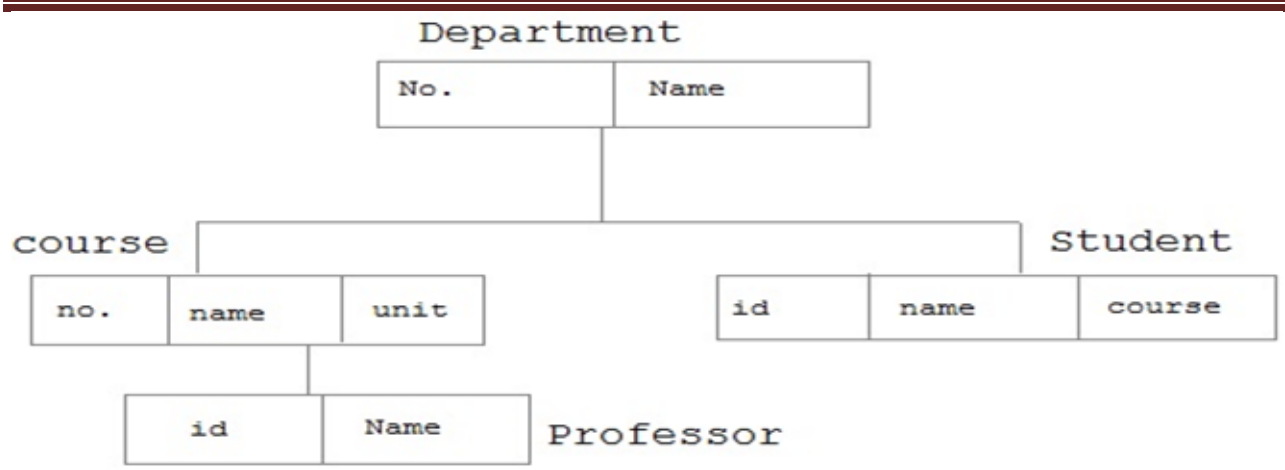


Figure 1: Hierarchical Model

Network Model: Network Model is an extension of the Hierarchical model. In this model data is organized more like a graph, and are allowed to have more than one parent node. In this database model data is more related as more relationships are established in this database model. Also, as the data is more related, hence accessing the data is also easier and fast. This database model was used to map many-to-many data relationships. This was the most widely used database model, before Relational Model was introduced.

In the network model, entities are organized in a graph, in which some entities can be accessed through several paths. The network database model is essentially a refinement of the hierarchical database model. The network model allows child tables to have more than one parent, thus creating a networked-like table structure. Multiple parent tables for each child allows for many-to-many relationships, in addition to one-to-many relationships. In an example network database model there is a many-to-many relationship between employees and tasks. In other words, an employee can be assigned many tasks, and a task can be assigned to many different employees. Thus, many employees have many tasks, and vice versa. The managers can be part of departments and companies. In other words, the network model is taking into account that not only does each department within a company have a manager, but also that each company has an overall manager (in real life, a Chief Executive Officer, or CEO). Employees can be defined as being of different types (such as full-time, part-time, or contract employees).

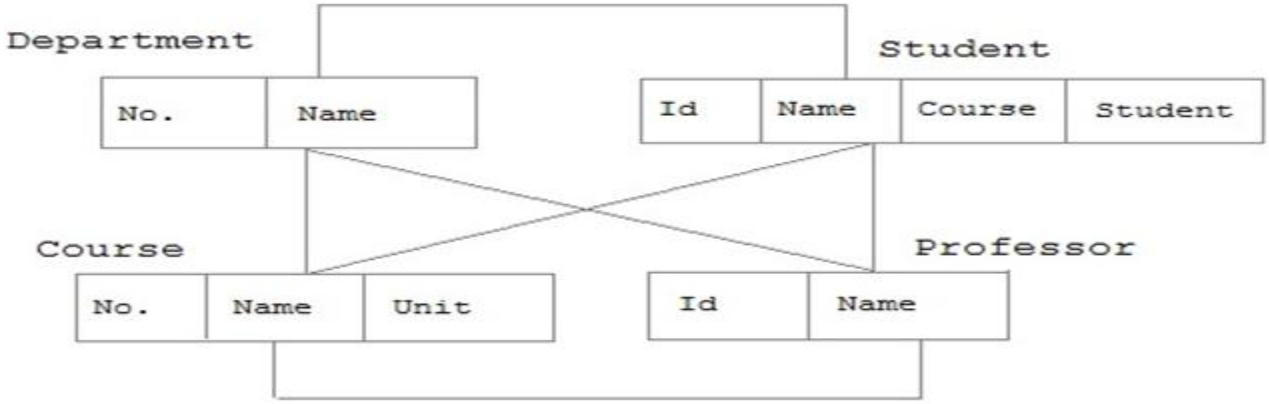


Figure 2: Network Model

Entity-relationship Model: Entity-relationship database model, relationships are created by dividing object of interest into entity and its characteristics into attributes. Different entities are related using relationships. E-R Models are defined to represent the relationships into pictorial form to make it easier for different stakeholders to understand. This model is good to design a database, which can then be turned into tables in relational model. Let's take an example, If we have to design a School Database, then Student will be an entity with attributes name, age, address etc. As Address is generally complex, it can be another entity with attributes street name, pincode, city etc, and there will be a relationship between them.

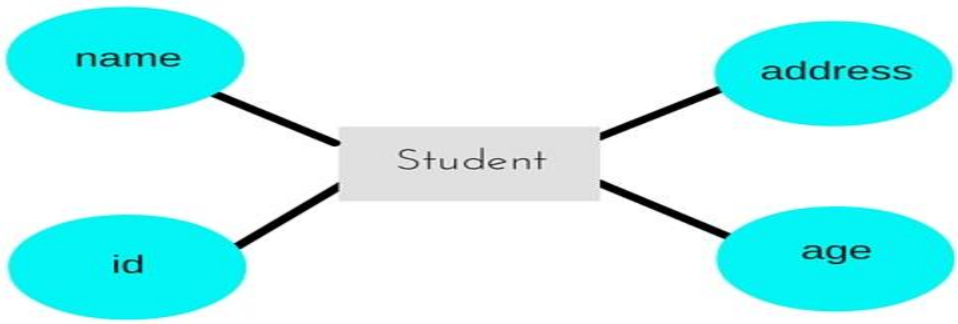
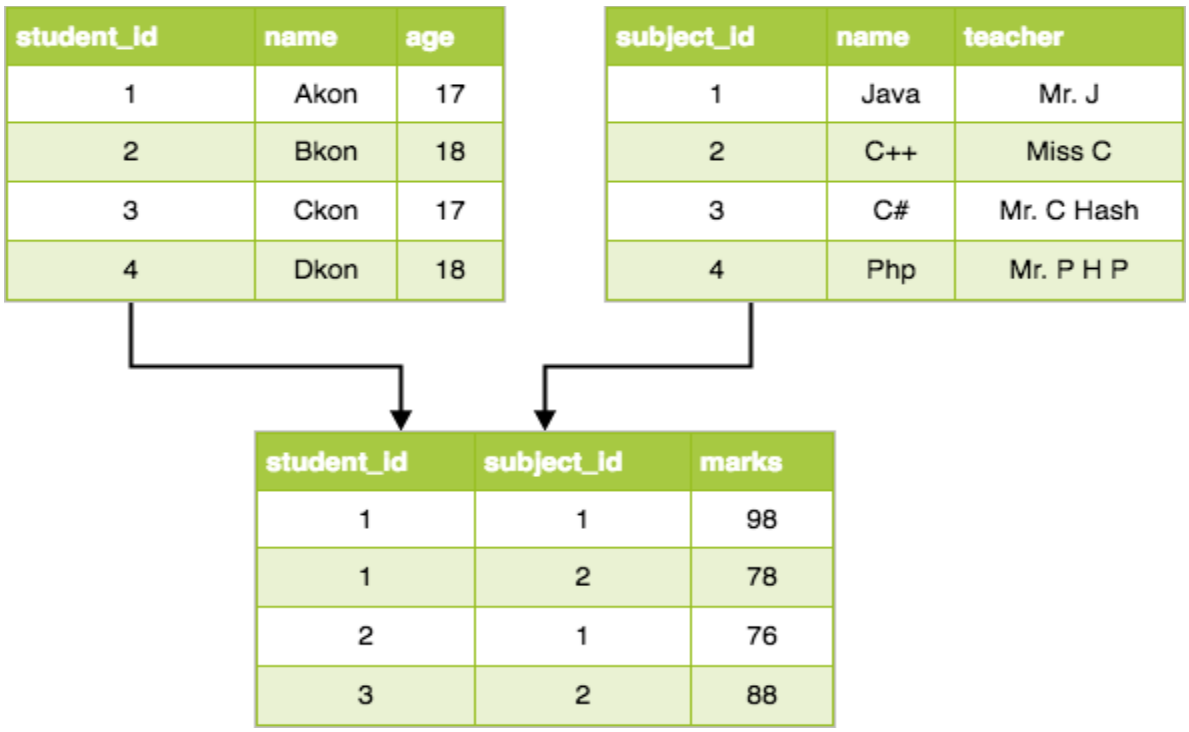


Figure 3: Entity-relationship model

Relational Model: In this model, data is organized in two-dimensional tables and the relationship is maintained by storing a common field. This model is the most widely used

database model, in-fact, we can say **the only database model used around the world**. The basic structure of data in the relational model is tables. All the information related to a particular type is stored in rows of that table. Hence, tables are also known as relations in relational model. Design and normalize a table to reduce data redundancy and how to use Structured Query language to access data from tables.



student_Id	name	age
1	Akon	17
2	Bkon	18
3	Ckon	17
4	Dkon	18

subject_Id	name	teacher
1	Java	Mr. J
2	C++	Miss C
3	C#	Mr. C Hash
4	Php	Mr. P H P

student_Id	subject_Id	marks
1	1	98
1	2	78
2	1	76
3	2	88

Figure 4: Relational model

The relational database model improves on the restriction of a hierarchical structure, not completely abandoning the hierarchy of data. Any table can be accessed directly without having to access all parent objects. The trick is to know what to look for if you want to find the address of a specific employee, you have to know which employee to look for, or you can simply examine all employees. You don't have to search the entire hierarchy, from the company downward, to find a single employee. Another benefit of the relational database model is that any tables can be linked together, regardless of their hierarchical position. Obviously, there should be a sensible link between the two tables, but you are not restricted by a strict hierarchical structure; therefore, a table can be linked to both any number of parent tables and any number of child tables.

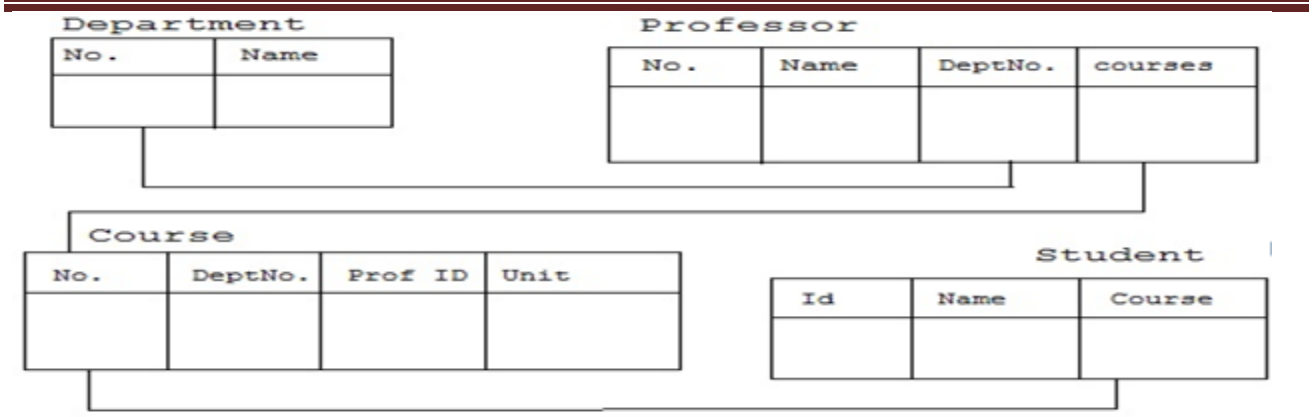


Figure 5: Relational model

Database prototype

Database prototype is a simulation or sample version of a final database project, which is used for testing prior to launch. The goal of a prototype is to test project and project ideas before sinking lots of time and money into the final project. Software prototyping is the activity of creating prototypes of software applications, i.e., incomplete versions of the software program being developed. It is an activity that can occur in software development and is comparable to prototyping as known from other fields, such as mechanical engineering or manufacturing. A prototype typically simulates only a few aspects of, and may be completely different from, the final product.

Prototyping has several benefits

- The software designer and implementer can get valuable feedback from the users early in the project.
- The client and the contractor can compare if the software made matches the software specification, according to which the software program is built.
- It also allows the software engineer some insight into the accuracy of initial project estimates and whether the deadlines and milestones proposed can be successfully met.

The purpose of a prototype is to allow users of the software to evaluate developers' proposals for the design of the eventual product by actually trying them out, rather than having to interpret and evaluate the design based on descriptions. Software prototyping provides an understanding of the software's functions and potential threats or issues.

Prototyping can also be used by end users to describe and prove requirements that have not been considered and that can be a key factor in the commercial relationship between developers and their clients. Interaction design in particular makes heavy use of prototyping with that goal.

In software development cycle of building the entire program first and working out any inconsistencies between design and implementation, which led to higher software costs and poor estimates of time and cost. Success of any project depends on the ability of a development team to meet their client's needs. The communication between the client and the development team plays a vital role in delivering a solution that fits product and market requirements. The issues arise if customers explain their needs too vaguely and the team can't understand clear requirements and eventually the business problem behind them.

Imagine that you ask your team to enable users to search for a product in an online bookstore by categories. You expect to have a clear interface with category links to click on them (e.g. fantasy, non-fiction, historic, etc.) After two weeks of development, you receive a search bar feature where users must type in the category they interested in, instead of browsing pre-listed categories. While this also works, your initial goal was to expose all available categories and let users explore further.

Acceptance criteria

Acceptance criteria (**AC**) are the conditions that a software product must meet to be accepted by a user, a customer, or other system. They are unique for each user story and define the feature behavior from the end-user's perspective. Well-written acceptance criteria help avoid unexpected results in the end of a development stage and ensure that all stakeholders and users are satisfied with what they get.

The main purposes of Acceptance criteria

- **Feature scope detail:** AC defines the boundaries of user stories. They provide precise details on functionality that help the team understand whether the story is completed and works as expected.

- **Describing negative scenarios:** Your AC may require the system to recognize unsafe password inputs and prevent a user from proceeding further. Invalid password format is an example of a negative scenario when a user does invalid inputs or behaves unexpectedly. AC defines these scenarios and explains how the system must react on them.
- **Setting communication:** Acceptance criteria synchronize the visions of the client and the development team. They ensure that everyone has a common understanding of the requirements. Developers know exactly what kind of behavior the feature must demonstrate, while stakeholders and the client understand what's expected from the feature.
- **Streamlining acceptance testing:** AC is the basis of the user story acceptance testing. Each acceptance criterion must be independently testable and thus have a clear pass or fail scenarios. They can also used to verify the story via automated tests.
- **Feature estimation.** Acceptance criteria specify what exactly must be developed by the team. Once the team has precise requirements, they can split user stories into tasks that can be correctly estimated.

Acceptance criteria types and structures

AC can be written in different formats. There are two most common ones, and the third option is to devise your own format:

- scenario-oriented (Given/When/Then)
- rule-oriented (checklist)
- custom formats

As the first and the second formats have very specific structures, we will mostly focus on them. However, you may find that other formats fit your product better.

Scenario-oriented acceptance criteria

Scenario-oriented format of writing AC is known as the *Given/When/Then* (GWT) type.

- *Given* some precondition
- *When* I do some action
- *Then* I expect some result

This approach provides a consistent structure that helps testers define when to begin and end testing a particular feature. It also reduces the time spent on writing test cases as the behavior of the system is described upfront. Each acceptance criteria written in this format has the following statements:

1. **Scenario** – the name for the behavior that will be described
2. **Given** – the beginning state of the scenario
3. **When** – specific action that the user makes
4. **Then** – the outcome of the action in "When"
5. **And** – used to continue any of three previous statements

When combined these statements cover all actions that a user takes to complete a task and experience the outcome.

Example 1: *User story, a*s a user, I want to be able to recover the password to my account, so that I will be able to access my account in case I forgot the password.

Scenario: Forgot password

1. **Given:** The user has navigated to the login page
2. **When**: The user selected *forgot password* option
3. **And**: Entered a valid email to receive a link for password recovery
4. **Then**: The system sent the link to the entered email
5. **Given:** The user received the link via the email
6. **When:** The user navigated through the link received in the email
7. **Then:** The system enables the user to set a new password

Example 2: User story, as a user I want to be able to request the cash from my account in **ATM**. so that I will be able to receive the money from my account quickly and in different places.

Acceptance criteria 1:

Given: that the account is creditworthy

And: the card is valid

And: the dispenser contains cash

When: the customer requests the cash

Then: ensure the account is debited

And: ensure cash is dispensed

And: ensure the card is returned

Acceptance criteria 2:

Given: that the account is overdrawn

And: the card is valid

When: the customer requests the cash

Then: ensure the rejection message is displayed

And: ensure cash isn't dispensed

Rule-oriented acceptance criteria format

In some cases, it's difficult to fit acceptance criteria into the Given/When/Then (GWT) structure. For instance, GWT would hardly be useful for the following cases:

- You're working with user stories that describe the system level functionality that needs other methods of quality assurance.
- The target audience for acceptance criteria doesn't need precise details of the test scenarios.
- GWT scenarios don't fit to describing design and user experience constraints of a feature. Developers may miss a number of critical details.

You can address these cases with the rule-oriented AC format. The rule-oriented form entails that there is a set of rules that describe the behavior of a system. Based on these rules, you can draw specific scenarios.

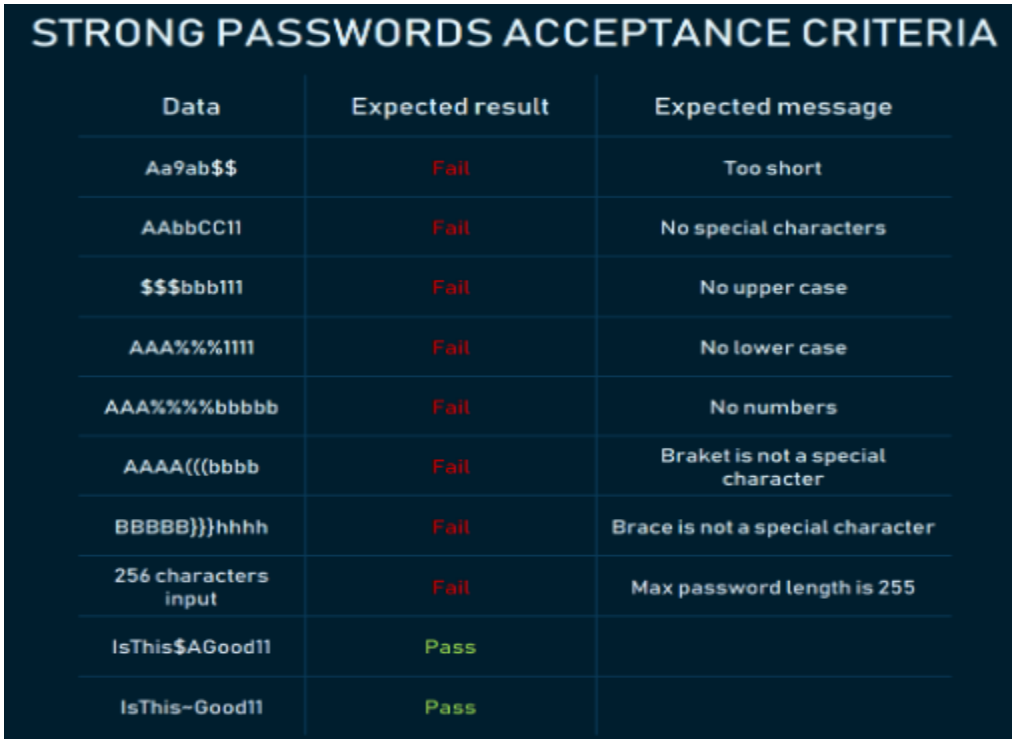
Example*:* User story**,** as a user I want to use a search field to type a city, name, or street, so that I could find matching hotel options.

Basic search interface acceptance criteria

- The search field is placed on the top bar
- Search starts once the user clicks "Search"
- The field contains a placeholder with a grey-colored text: "Where are you going?"
- The placeholder disappears once the user starts typing
- Search is performed if a user types in a city, hotel name, street, or all combined
- Search is in English, French, German, and Ukrainian
- The user can't type more than 200 symbols
- The search doesn't support special symbols (characters). If the user has typed a special symbol, show the warning message: "Search input cannot contain special symbols."

Other formats

Also you can invent your own acceptance criteria given they serve their purposes are written clearly in plain English and can't be misinterpreted. Some teams even use plain text.



STRONG PASSWORDS ACCEPTANCE CRITERIA

Data	Expected result	Expected message
Aa9ab$$	Fail	Too short
AAbbCC11	Fail	No special characters
$$$bbb111	Fail	No upper case
AAA%%%1111	Fail	No lower case
AAA%%%bbbbb	Fail	No numbers
AAAA(((bbbb	Fail	Braket is not a special character
BBBBB}}}hhhh	Fail	Brace is not a special character
256 characters input	Fail	Max password length is 255
IsThis$AGood11	Pass	
IsThis~Good11	Pass	

Figure 6: A simple set of AC for strong passwords *provides clear guidelines to password feature testing.*

Roles responsible and how acceptance criteria are created

Some of the criteria are defined and written by the *product owner* when he or she creates the product backlog. And the others can be further specified by the team during user stories discussions after sprint planning. There are no strict recommendations to choosing the person responsible for writing the criteria. The client can document them if he or she has ample technical and product documentation knowledge. In this case, the client negotiates the criteria with the team to avoid mutual misunderstandings. Otherwise the criteria are created *by a* product owner, business analyst, requirements analyst, *or a* project manager. The creation process *of Acceptance criteria are* starts with user story prioritization and ends with negotiating details with the whole team.

Main challenges and best practices of writing acceptance criteria

The best practices that help to avoid common mistakes are.

- **Document criteria before development.** Acceptance criteria have to be documented before the actual development starts. This way, the team will likely capture all customer needs in advance. In the beginning, it's enough to set the criteria for a small number of user stories to fill the backlogs for two sprints. They must be agreed by both parties. Then the documented acceptance criteria are used by developers to plan the technical process.
- **Don't make AC too narrow.** Acceptance criteria can be way too specific living little to no maneuver options for developers. To avoid this, remember that AC must convey the intent but not a final solution. Moreover, narrow AC may be bereft of multiple user behaviors that aren't covered.
- **Keep your criteria achievable.** This point closely intersects with the previous one. Effective acceptance criteria define the reasonable minimum chunk of functionality that you're able to deliver. But in case you succumb to describing all little details, there's a risk that your team will get stuck with hundreds of small tasks.
- **Keep AC measurable and not too broad.** Broad acceptance criteria make a user story vague. Effective acceptance criteria must outline the scope of work so that the developers can plan and estimate their effort properly.
- **Avoid technical details.** As we mentioned, acceptance criteria must be written in plain English. This will make them clear and easy to understand for everyone: Your stakeholders or managers may not have technical background.
- **Reach consensus.** The same problem may be able to solve differently by a team and stakeholders, depending on their vantage points. Make sure that you've communicated your AC to stakeholders and reached a mutual agreement. The same applies to team members. Everyone must review the AC and confirm they understand and agree with each line.
- **Write testable AC.** This will allow testers to verify that all requirements were met. Otherwise, developers won't understand if the user story is completed.

Final word: Don't neglect the acceptance criteria as they being simple and approachable solve multiple problems at once. Document customer expectations, provide an end-user perspective, clarify requirements and prevent ambiguity, and eventually help quality assurance verify if the development goals were met. Regardless of whether you use agile methods or not, make sure to choose the best format or experiment with your own ones. Different types of user stories and eventually features may require different formats and testing the new ones that work for you is a good practice.

Use Codd's rule to test DBMS's

E.F Codd was a Computer Scientist who invented Relational model for Database management. Based on relational model, Relation database was created. Codd proposed 13 rules popularly known as Codd's 12 rules to test DBMS's concept against his relational model. Codd's rules actually define what quality a DBMS requires in order to become a Relational Database Management System (RDBMS). Still now, there is hardly any commercial product that follows all the 13 Codd's rules.

Rule zero: This rule states that for a system to qualify as an **RDBMS**, it must be able to manage database entirely through the relational capabilities.

Rule 1-Information rule: All information including metadata is to be represented as stored data in cells of tables. The rows and columns have to be strictly unordered.

Rule 2-Guaranteed Access: Each unique piece of data (atomic value) should be accessible by: **Table Name + primary key (Row) + Attribute (column)**. Ability to directly access via POINTER is a violation of this rule.

Rule 3- Systematic treatment of NULL: Null has several meanings; it can mean missing data, not applicable or no value. It should be handled consistently. Primary key must not be null. Expression on **NULL** must give null.

Rule 4-Active Online Catalog: Database dictionary (catalog) must have description of **Database**. Catalog to be governed by same rule as rest of the database. The same query language to be used on catalog as on application database.

Rule 5-Powerful language: One well defined language must be there to provide all manners of access to data. Example: **SQL**. If a file supporting table can be accessed by any manner except SQL interface, then it's a violation to this rule.

Rule 6 - View updating rule: All view that is theoretically updatable should be updatable by the system.

Rule 7- Relational Level Operation: There must be Insert, Delete, and Update operations at each level of relations. Set operation like Union, Intersection and minus should also be supported.

Rule 8 - Physical Data Independence: The physical storage of data should not matter to the system. If say, some file supporting table were renamed or moved from one disk to another, it should not affect the application.

Rule 9 - Logical Data Independence: If there is change in the logical structure of the database the user view of data should not change. Say, if a table is split into two tables, a new view should give result as the join of the two tables. This rule is most difficult to satisfy.

Rule 10 - Integrity Independence: The database should be able to converse its own integrity rather than using other programs. Key and Check constraints, trigger etc should be stored in Data Dictionary. This also makes **RDBMS** independent of front-end.

Rule 11- Distribution Independence: A database should work properly regardless of its distribution across a network. This lays foundation of distributed database.

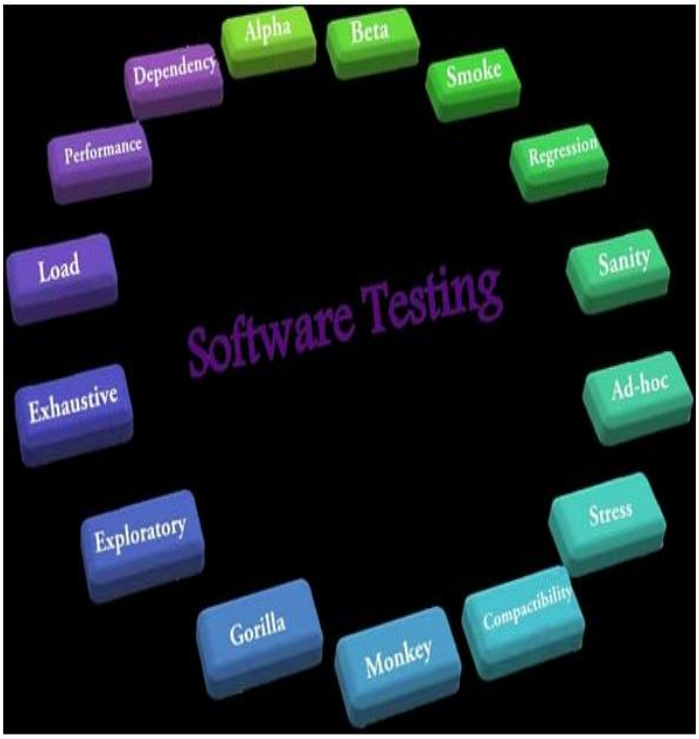
Rule 12 - Non subversion rule: If low level access is allowed to a system it should not be able to subvert or bypass integrity rule to change data. This can be achieved by some sort of looking or encryption.

The different types of Software Testing

As a testers you should aware of the various types of Software Testing such as Functional Testing, Non-Functional Testing, Automation Testing, Agile Testing, and their sub-types, etc. Each type of testing has its own features, advantages, and disadvantages.

Common types of Software Testing includes

- **Functional Testing types include:**
  1. Unit Testing
  2. Integration Testing
  3. System Testing
  4. Sanity Testing
  5. Smoke Testing
  6. Interface Testing
  7. Regression Testing
  8. Beta/Acceptance Testing



- **Non-functional Testing types include:**
  1. Performance Testing
  2. Load Testing
  3. Stress Testing
  4. Volume Testing
  5. Security Testing
  6. Compatibility Testing
  7. Install Testing
  8. Recovery Testing
  9. Reliability Testing
  10. Usability Testing
  11. Compliance Testing
  12. Localization Testing

Functional Testing

Functional Testing ignores the internal parts and focuses only on the output to check if it is as per the requirement or not. It is a Black-box type testing geared to the functional requirements of an application.

1. **Unit Testing:** Testing of an individual software component or module is termed as Unit Testing. It is typically done by the programmer and not by testers, as it requires detailed

knowledge of the internal program design and code. It may also require developing test driver modules or test harnesses.

2. **Integration Testing:** Testing of all integrated modules to verify the combined functionality after integration is termed as Integration Testing. Modules are typically code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/server and distributed systems.
3. **System Testing:** Under System Testing technique, the entire system is tested as per the requirements. It is a Black-box type testing that is based on overall requirement specifications and covers all the combined parts of a system.
4. **Sanity Testing:** Sanity Testing is done to determine if a new software version is performing well enough to accept it for a major testing effort or not. If an application is crashing for the initial use then the system is not stable enough for further testing. Hence a build or an application is assigned to fix it.
5. **Smoke Testing:** Whenever a new build is provided by the development team then the Software Testing team validates the build and ensures that no major issue exists. The testing team ensures that the build is stable and a detailed level of testing is carried out further. Smoke Testing checks that no show stopper defect exists in the build which will prevent the testing team to test the application in detail. If testers find that the major critical functionality is broken down at the initial stage itself then testing team can reject the build and inform accordingly to the development team. Smoke Testing is carried out to a detailed level of any Functional or Regression Testing.
6. **Interface Testing:** The objective of this GUI Testing is to validate the GUI as per the business requirement. The expected GUI of the application is mentioned in the Detailed Design Document and GUI mockup screens. The GUI Testing includes the size of the buttons and input field present on the screen, alignment of all text, tables and content in the tables. It also validates the menu of the application, after selecting different menu and menu items, it validates that the page does not fluctuate and the alignment remains same after hovering the mouse on the menu or sub-menu.
7. **Regression Testing:** Testing an application as a whole for the modification in any module or functionality is termed as Regression Testing. It is difficult to cover all the system in Regression Testing, so typically Automation Testing Tools are used for these types of testing.

8. **Acceptance Testing:** It is performed by the client and verifies whether the end to end the flow of the system is as per the business requirements or not and if it is as per the needs of the end-user. Client accepts the software only when all the features and functionalities work as expected. It is the last phase of the testing, after which the software goes into production. This is also called User Acceptance Testing (UAT).
9. **Beta Testing:** It is a formal type of Software Testing which is carried out by the customer. It is performed in the Real Environment before releasing the product to the market for the actual end-users. Beta Testing is carried out to ensure that there are no major failures in the software or product and it satisfies the business requirements from an end-user perspective. Beta Testing is successful when the customer accepts the software. Usually, this testing is typically done by end-users or others. It is the final testing done before releasing an application for commercial purpose. Usually, the Beta version of the software or product released is limited to a certain number of users in a specific area. So end-user actually uses the software and shares the feedback to the company. Company then takes necessary action before releasing the software to the worldwide.

Non-Functional Testing

Non-Functional Testing is a type of testing for which every organization having a separate team which usually called as Non-Functional Test (NFT) team or Performance team. Non-Functional Testing involves testing of non-functional requirements such as Load Testing, Stress Testing, Security, Volume, Recovery Testing, etc. The objective of NFT testing is to ensure whether the response time of software or application is quick enough as per the business requirement. It should not take much time to load any page or system and should sustain during peak load.

Performance Testing: This term is often used interchangeably with ‘stress’ and ‘load’ testing. Performance Testing is done to check whether the system meets the performance requirements. Different performance and load tools are used to do this testing.

Load Testing: It is a type of Non-Functional Testing and the objective of Load Testing is to check how much load or maximum workload a system can handle without any performance

degradation. Load Testing helps to find the maximum capacity of the system under specific load and any issues that cause software performance degradation.

Stress Testing: This testing is done when a system is stressed beyond its specifications in order to check how and when it fails. This is performed under heavy load like putting large number beyond storage capacity, complex database queries, and continuous input to the system or database load.

Volume Testing: Volume Testing is a type of Non-Functional Testing performed by the Performance Testing team. The software or application undergoes a huge amount of data and Volume Testing checks the system behavior and response time of the application when the system came across such a high volume of data. This high volume of data may impact the system's performance and speed of the processing time.

Security Testing: It is a type of testing performed by a special team of testers. A system can be penetrated by any hacking way. Security Testing is done to check how the software or application or website is secure from internal and external threats. This testing includes how much software is secure from the malicious program, viruses and how secure and strong the authorization and authentication processes are. It also checks how software behaves for any hackers attack and malicious programs and how software is maintained for data security after such a hacker attack.

Compatibility Testing: It is a subtype of Compatibility Testing and is performed by the testing team. Compatibility Testing is performed for web applications and it ensures that the software can run with the combination of different browser and operating system. This type of testing validates whether web application runs on all version of all browsers or not.

Install/Uninstall Testing: Installation and Un-installation Testing is done on full, partial, or upgrade install/uninstall processes on different operating systems under different hardware or software environment.

Recovery Testing: It is a type of testing which validates how well the application or system recovers from crashes or disasters. Recovery Testing determines if the system is able to

continue the operation after a disaster. Assume that application is receiving data through the network cable and suddenly that network cable has been unplugged. Sometime later, plug the network cable; then the system should start receiving data from where it lost the connection due to network cable unplugged.

Usability Testing: Under Usability Testing, User-friendliness check is done. The application flow is tested to know if a new user can understand the application easily or not, Proper help documented if a user gets stuck at any point. Basically, system navigation is checked in this testing.

Alpha Testing: It is the most common type of testing used in the Software industry. The objective of this testing is to identify all possible issues or defects before releasing it into the market or to the user. Alpha Testing is carried out at the end of the software development phase but before the Beta Testing. Still, minor design changes may be made as a result of such testing. Alpha Testing is conducted at the developer's site. In-house virtual user environment can be created for this type of testing.

Ad-hoc Testing: The name itself suggests that this testing is performed on an Ad-hoc basis i.e. with no reference to the test case and also without any plan or documentation in place for such type of testing. The objective of this testing is to find the defects and break the application by executing any flow of the application or any random functionality. Ad-hoc Testing is an informal way of finding defects and can be performed by anyone in the project. It is difficult to identify defects without a test case but sometimes it is possible that defects found during ad-hoc testing might not have been identified using existing test cases.

Accessibility Testing: The aim of Accessibility Testing is to determine whether the software or application is accessible for disabled people or not. Here, disability means deaf, color blind, mentally disabled, blind, old age and other disabled groups. Various checks are performed such as font size for visually disabled, color and contrast for color blindness, etc.

Back-end Testing: Whenever an input or data is entered on front-end application, it stores in the database and the testing of such database is known as Database Testing or

Backend Testing. There are different databases like SQL Server, MySQL, and Oracle, etc. Database Testing involves testing of table structure, schema, stored procedure, data structure and so on. In Back-end Testing GUI is not involved, testers are directly connected to the database with proper access and testers can easily verify data by running a few queries on the database. There can be issues identified like data loss, deadlock, data corruption etc during this back-end testing and these issues are critical to fixing before the system goes live into the production environment

Backward Compatibility Testing: It is a type of testing which validates whether the newly developed software or updated software works well with the older version of the environment or not. Backward Compatibility Testing checks whether the new version of the software works properly with file format created by an older version of the software; it also works well with data tables, data files, data structure created by the older version of that software. If any of the software is updated then it should work well on top of the previous version of that software.

Black Box Testing: Internal system design is not considered in this type of testing. Tests are based on the requirements and functionality. Detailed information about the advantages, disadvantages, and types of Black box Testing can be seen *here*.

Boundary Value Testing: This type of testing checks the behavior of the application at the boundary level. Boundary Value Testing is performed for checking if defects exist at boundary values. Boundary Value Testing is used for testing a different range of numbers. There is an upper and lower boundary for each range and testing is performed on these boundary values.

Branch Testing: It is a type of White box Testing and is carried out during Unit Testing. Branch Testing, the name itself suggests that the code is tested thoroughly by traversing at every branch.

Comparison Testing: Comparison of a product's strength and weaknesses with its previous versions or other similar products is termed as Comparison Testing.

Component Testing: It is mostly performed by developers after the completion of unit testing. Component Testing involves testing of multiple functionalities as a single code and its objective is to identify if any defect exists after connecting those multiple functionalities with each other.

End-to-End Testing: Similar to system testing, End-to-End Testing involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.

Equivalence Partitioning: It is a testing technique and a type of Black Box Testing. During this Equivalence Partitioning, a set of the group is selected and a few values or numbers are picked up for testing. It is understood that all values from that group generate the same output. The aim of this testing is to remove redundant test cases within a specific group which generates the same output but not any defect. Suppose, the application accepts values between -10 to +10 so using equivalence partitioning the values picked up for testing are zero, one positive value, one negative value. So the Equivalence Partitioning for this testing is -10 to -1, 0, and 1 to 10.

Example Testing: It means real-time testing. Example Testing includes the real-time scenario; it also involves the scenarios based on the experience of the testers.

Exploratory Testing: Exploratory Testing is informal testing performed by the testing team. The objective of this testing is to explore the application and looking for defects that exist in the application. Sometimes it may happen that during this testing major defect discovered can even cause a system failure. During Exploratory Testing, it is advisable to keep a track of what flow you have tested and what activity you did before the start of the specific flow. An Exploratory Testing technique is performed without documentation and test cases.

Gorilla Testing: Gorilla Testing is a testing type performed by a tester and sometimes by the developer the as well. In Gorilla Testing, one module or the functionality in the module is tested thoroughly and heavily. The objective of this testing is to check the robustness of the application.

Happy Path Testing: The objective of Happy Path Testing is to test an application successfully on a positive flow. It does not look for negative or error conditions. The focus is only on the valid and positive inputs through which application generates the expected output.

Incremental Integration Testing: Incremental Integration Testing is a Bottom-up approach for testing i.e. continuous testing of an application when new functionality is added. Application functionality and modules should be independent enough to test separately. This is done by programmers or by testers.

Monkey Testing: Monkey Testing is carried out by a tester assuming that if the monkey uses the application then how random input, values will be entered by the Monkey without any knowledge or understanding of the application. The objective of Monkey Testing is to check if an application or system gets crashed by providing random input values/data. Monkey Testing is performed randomly and no test cases are scripted and it is not necessary to. Monkey Testing is performed randomly and no test cases are scripted and it is not necessary to be aware of the full functionality of the system.

Mutation Testing: Mutation Testing is a type of white box testing in which the source code of one of the program is changed and verifies whether the existing test cases can identify these defects in the system. The change in the program source code is very minimal so that it does not impact the entire application, only the specific area having the impact and the related test cases should able to identify those errors in the system.

Negative Testing: Testers having the mindset of "attitude to break" and using Negative Testing they validate that if system or application breaks. A Negative Testing technique is performed using incorrect data, invalid data or input. It validates that if the system throws an error of invalid input and behaves as expected.

Risk-Based Testing (RBT): In Risk-Based Testing, the functionalities or requirements are tested based on their priority. Risk-Based Testing includes testing of highly critical functionality, which has the highest impact on business and in which the probability of failure is very high. The priority decision is based on the business need, so once priority is set for all functionalities then high priority functionality or test cases are executed first

followed by medium and then low priority functionalities. The low priority functionality may be tested or not tested based on the available time. The Risk-Based Testing is carried out if there is insufficient time available to test entire software and software needs to be implemented on time without any delay. This approach is followed only by the discussion and approval of the client and senior management of the organization.

Static Testing: Static Testing is a type of testing which is executed without any code. The execution is performed on the documentation during the testing phase. It involves reviews, walkthrough, and inspection of the deliverables of the project. Static Testing does not execute the code instead of the code syntax, naming conventions are checked. Static Testing is also applicable for test cases, test plan, design document. It is necessary to perform static testing by the testing team as the defects identified during this type of testing are cost-effective from the project perspective.

Vulnerability Testing: The testing which involves identifying weakness in the software, hardware and the network is known as Vulnerability Testing. Malicious programs, the hacker can take control of the system, if it is vulnerable to such kind of attacks, viruses, and worms. So it is necessary to check if those systems undergo Vulnerability Testing before production. It may identify critical defects, flaws in the security.

White Box Testing: White Box Testing is based on the knowledge about the internal logic of an application's code. It is also known as Glass box Testing. Internal software and code working should be known for performing this type of testing. Under these tests are based on the coverage of code statements, branches, paths, conditions, etc.

Self-Check 1	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

1. Which model is the most widely used database model, in-fact, we can say the only database model used around the world?

   A. Network model B. Relational model C. Hierarchical model D. All

2. Which of the following is a simulation of a final database project, which is used for testing prior to launch?

   A. Database prototype B. Acceptance criteria C. custom formats D. rule-oriented

3. One is not true about Null meanings in database;

   A. it can mean missing data, B. not applicable C. no value D. Applicable

4. Which one is the type of test that ignores the internal parts and focuses only on the output to check if it is as per the requirement or not?

   A. Non-Functional Testing B. Acceptance testing

   C. Functional Testing D. Load testing

5. The objective of _____testing is to ensure whether the response time of software is quick enough as per the business requirement.

A. Non-functional B. Functional C. Unit testing D. All

Note: **Satisfactory rating - 5 points** **Unsatisfactory - below 5 points**

You can ask you teacher for the copy of the correct answers.

Score = _____
Rating: _____

Information sheet 2: Loading test data according to the technical sequence

1.2 Loading test data according to the technical sequence detailed in documentation

Load Testing is a subset of Performance Testing, where we test the system's response under varying load conditions by simulating multiple users accessing the application concurrently. This testing usually measures the speed and capacity of the application. Thus whenever we modify the load, we monitor the behavior of the system under various conditions.

Example: Let's assume that our client requirement for a Login page is 2-5 sec and this 2-5 sec should be consistent all throughout until the load is 5000 users. So what should we observe hear? Is it just the load handling capability of the system or is it just the response time requirement?

The answer is both. We want the system which can handle a load of 5000 users with the response time of 2-5 seconds for all the concurrent users. So what is meant by a concurrent user and a virtual user? Concurrent users are those who log in to the application and at the same time, perform a set of activities together and log off the application at the same time. On the other hand, virtual users just hop in and hop out of the system irrespective of the other user activities.

Load Test Architecture

In the below diagram we can see how different users are accessing the application. Here each user is making a request over the internet, which is later passed through a firewall. After firewall, we have a Load balancer which distributes the load to any of the web servers, and then passes to the application server and later to the database server where it fetches the necessary information based on the user request.

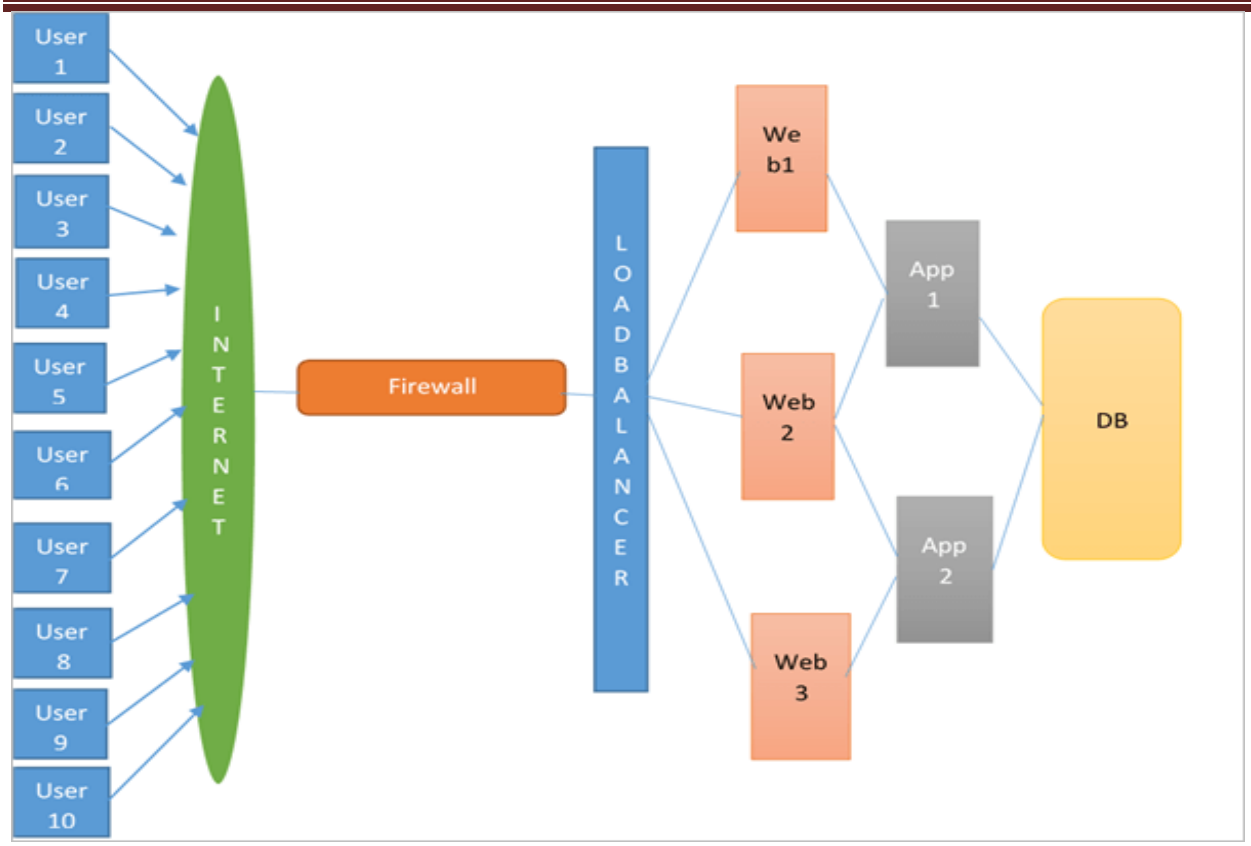


Figure 1.2: Load Test Architecture

Load testing can be done manually as well as by using a tool. But manual load testing is not advised as we don't test the application for a lesser load.

Example: Let's assume, that we want to test an online shopping application to see the response time of the application for each user click i.e. Step1 –Launch URL, the response time, Login to the application and note the response time and so on like selecting a product, adding to the cart, making payment and logging off. All these have to be done for 10 users.

So, now when we need to test the application load for 10 users then we can achieve this by manually putting load by 10 physical users from different machines instead of using a tool. In this scenario, it is advisable to go for a manual load test rather than investing in a tool and setting up an environment for the tool.

Whereas imagine if we need to load test for 1500 users then we need to automate the load test using any of the available tools based on the technologies in which the application is built and also based on the budget that we have for the project.

If we have a budget, then we can go for commercial tools like Load runner but if we don't have much budget then we can go for open source tools like JMeter, etc. Whether it is a commercial tool or an open source tool, the details have to be shared with the client before we finalize the tool. Usually, a proof of concept is prepared, where we generate a sample script using the tool and show the sample reports to the client for approval of the tool before finalizing it. In automated load testing, we replace the users with the help of an automation tool, which mimics the real-time user actions. By automating load we can save resources as well as the time.

Let's assume that there is an online shopping website which is doing pretty good during normal business days i.e. users are able to login to the application, browse through the different product categories, select products, add items to the cart, check out and log off within an acceptable range and there are no page errors or huge response times.

Meanwhile, there comes a peak day i.e., let's say the Thanks Giving day and there are thousands of users who are logged in to the system, the system is crashed all of a sudden and the users experience a very slow response, some couldn't even log in to the site, a few failed to add to cart and some failed to check out.

Hence on this big day, the company had to face a huge loss as it lost many customers and much business too. All this happened just because they didn't predict the user load for peak days, even if they would have predicted there was no load test done on the company website, hence they don't know how much load the application will be able to handle on the peak days.

Thus to handle such situations and in order to overcome huge revenue, it's advisable to conduct load test for such type of applications.

- Load Testing helps to build strong and reliable systems.
- The bottleneck in the system is identified well in advance before the application goes live.
- It helps in identifying the capacity of the application.

With a proper Load test, we can have an exact understanding of the following:

1. The number of users the system is able to handle or is capable of scaling to.
2. The response time of each transaction.
3. How does each component of the entire system behave under Load i.e. application server components, web server components, Database components, etc.
4. What server configuration is best to handle the load?
5. Whether the existing hardware is enough or is there any need for additional hardware.
6. Bottlenecks like CPU utilization, Memory Usage, Network delays, etc., are identified.

Before we start the Load test we need to understand if any Load test is already done on the system or not. If there was any load testing done earlier, then we need to know what was the response time, client and server metrics collected, how much was the user load capacity etc. Also, we need information on how much is the current application handling capability. If it's a new application we need to understand the requirements, what is the targeted load, what is the expected response time and if it is really achievable or not.

If it is an existing application, you can get the load requirements and the user access patterns from the server logs. But if it is a new application then you need to reach out to the business team to get all the information. Once we have the requirements, we need to identify how we are going to execute the load test. Is it done manually or using tools? Doing a load test manually needs a lot of resources and is very expensive too. Also repeating the test, again and again, will be tough as well. Hence, to overcome this we can either use Open source tools or commercial tools. Open source tools are available for free, these tools may not have all the features like the other commercial tools but if the project has a budget constraint, then we can go for open source tools. Whereas commercial tools have many features, they support many protocols and user-friendly.

Load Test approach will be as follows:

1. **Identify the Load test Acceptance Criteria**

   For Example:

   - The response time of the Login page shouldn't be more than 5 sec even during the max load conditions.
   - CPU utilization should not be more than 80%.
   - The throughput of the system should be 100 transactions per sec.

2. **Identify the Business scenarios that need to be tested.** Don't test all the flows try to understand the main business flows which are expected to happen in production. If it is an existing application we can get his information from the server logs of the production environment. If it is a newly build application then we need to work with the business teams to understand the flow patterns, application usage etc. Sometimes the project team will conduct workshops to give an overview or details about each component of the application. We need to attend the application workshop and note all the required information to conduct our load test.
3. **Work Load Modeling:** Once we have the details about the business flows, user access patterns and the number of users, we need to design the workload in such a way in which it mimics the actual user navigation in production or as expected to be in the future once the application will be in production. The key points to remember while designing a workload model is to see how much time a particular business flow will take to complete.
4. **Design the Load Tests:** The Load Test should be designed with the data that we collected so far **i.e.** the Business flows, Number of users, user patterns, Metrics to be collected and analyzed. The tests should be designed in a much realistic way.
5. **Execute Load Test**: Before we execute the Load test, make sure that the application is up and running. The Load test environment is ready. The application is functionally tested and is stable. Check the configuration settings of the Load test environment. It should be the same as the production environment. Ensure all the test data is available. Make sure to add necessary counters to monitor the system performance during test

execution. Always start with a low load and gradually increase the load. Never start with the full load and break the system.

6. **Analyze the Load Test Results:** Have a baseline test to always compare with the other test runs. Gather the metrics and server logs after the test run to find the bottlenecks. Some projects use Application Performance Monitoring (APM) Tools to monitor the system during the test run, these APM tools help to identify the root cause more easily and save a lot of time. These tools are very easy to find the root cause of the bottleneck as they have a wide view to pinpoint where the issue is. Some of the APM tools in the market include DynaTrace, Wily Introscope, App Dynamics etc.
7. **Reporting:** Once the Test Run is complete, gather all the metrics and send the test summary report to the concerned team with your observations and recommendations.

The best practices of Load Testing are:

1. Always check the application stability before starting a Load test. The application should be signed functionally stable by the Functional testing team and all major defects should be fixed and tested before the build is copied to the Load Test environment.
2. Ensure that the Load test environment is a replica or is close to the production environment, including the number of servers, Load balancers, server configurations, and firewalls.
3. Check if the test data is unique and we have the entire test data copied to the load environment before conducting a load test.
4. Design the test scenarios in such a way that they mimic the real-time user action happening in the production.
5. Design the workload based on the production user loads and business flows and in case of an old application, see if it is a new talk to the business team regarding the business flows and the user load.
6. Collect all the important metrics like Response Time, hits per sec, Throughput, CPU, Memory, Network, and Running users.

Self-Check 2	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

1. Which types of testing measures the speed and capacity of the application?
   A. Load test B. Unit test C. Beta test D. System test
2. With a proper Load test, we can have an exact understanding of the following except one.
   A. The number of users the system is able to handle or is capable of scaling to.
   B. The response time of each transaction.
   C. server configuration is best to unit test
   D. Bottlenecks like CPU utilization, Memory Usage, Network delays, etc., are identified
3. Before we start the Load test we need to understand if any Load test is already done on the system or not.
   A. True B. False C. Unknown
4. One of the following is not Load Test approach.
   A. Identify the Load test Acceptance Criteria
   B. Identify the Business scenarios that need to be tested
   C. Work Load Modeling
   D. Documentation of the test
5. **On e is not the benefit of load test.**
   A. Load Testing helps to build strong and reliable systems.
   B. The bottleneck in the system is identified well in advance before the application goes live.
   C. It helps in identifying the capacity of the application.
   D. Load Testing helps to build strong and undependable systems.

Note: **Satisfactory rating - 5 points** **Unsatisfactory - below 5 points**

You can ask you teacher for the copy of the correct answers.

Score = _____
Rating: _____

Information sheet 3: Generating a test schedule for the database of tasks

1.3 Generating a test schedule for the database of tasks

A Test Plan is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product. Test Plan helps us determine the effort needed to validate the quality of the application under test. The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager. Test Plan is a document describing the scope, approach, resources, and schedule of intended test activities.

Follow the seven steps below to create a test plan

1. Analyze the product
2. Design the Test Strategy
3. Define the Test Objectives
4. Define Test Criteria
5. Resource Planning
6. Plan Test Environment
7. Schedule & Estimation
8. Determine Test Deliverables

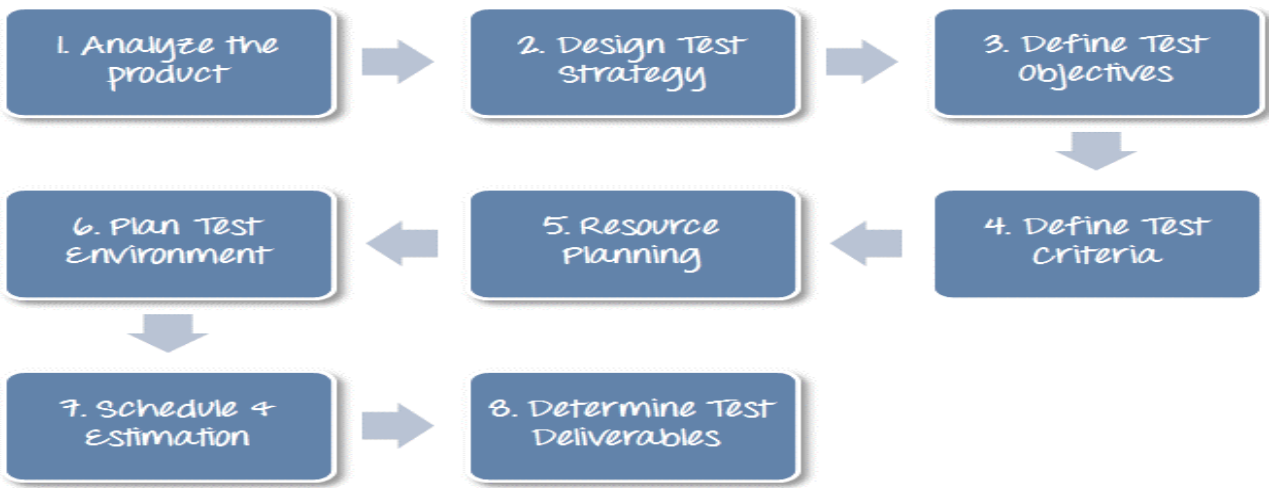


Figure 1.3: steps to create a test plan

A test schedule includes the testing steps or tasks, the target start and end dates, and responsibilities. It should also describe how the test will be reviewed, tracked, and approved.

Sample Template for Test Schedule

Test Schedule ID:	Describe the Test Schedule ID		
Product ID / Name:	Describe the Name of the Product / Project		
Product Version or Build:	Describe the Current Version or Build of the Product		
Present Owner :	Describe the Name of the owner of the Test Schedule Document at present.		
Created On:	Describe the Date on which Test Schedule Document was created initially		
Review On:	Describe the Date on which Test Schedule Document was last Reviewed & Updated		
Review By:	Describe the Name & Position of the Reviewer.		
Review Comments:	Describe the Comments if any, whether comments have been incorporated or Not		
Current Version:	Describe the Current Version of the Test Schedule Document		
Change Details:	Describe the Description of Change, Affected Section of the Test Plan		
Current Status:	Draft / In Process / Approved		
Signing Off Authority:	**Name**	**Position**	**Signature, Date**
	Describe the Name	E.g. QA Manager, Dev. Manager, Product Manager, Release Team Manager	

Test schedule steps template

Test Step	Start Date	End Date	Responsibility
First Spiral			
(A) Information gathering			
a1) Prepare for Interview			
a2) Conduct Interview			
a3) Summarize Findings			
(B) Test Planning			
b1) Build Test Plan			
b2) Define the Metric Objectives			
b3) Review / Approve Plan			
(C) Test Case Design			
c1) Design Function Tests			
c2) Design GUI Tests			
c3) Define the System / Acceptance Tests			
Review / Approve Design			
(D) Test Development			
d1) Develop Test Scripts			
d2) Review / Approve Test Development			
(E) Test Execution/Evaluation			
e1) Setup and Testing			
e2) Evaluation			
(F) Prepare for the Next Spiral			
f1) Refine the tests			
f2) Reassess Team, Procedures, and Test Environment			
f3) Publish Interim Report			
*			
*			
Last Spiral …			
(G) Test Execution/Evaluation			

g1) Setup and Testing			
g2) Evaluation			
*			
*			
(H) Conduct System Testing			
h1) Complete System Test Plan			
h2) Complete System Test Cases			
h3) Review / Approve System Tests			
h4) Execute the System Tests			
(J) Conduct Acceptance Testing			
J1) Complete Acceptance Test Plan			
J2) Complete Acceptance Test Cases			
J3) Review / Approve Acceptance Test Plan			
Execute the Acceptance Tests			
(K) Summarize/Report Spiral Test Results			
k1) Perform Data Reduction			
k2) Prepare Final Test Report			
k3) Review / Approve the Final Test Report			

DBMS Test Schedule

Transactions are set of instructions and these instructions perform operations on database. When multiple transactions are running concurrently then there needs to be a sequence in which the operations are performed because at a time only one operation can be performed on the database. This sequence of operations is known as **Schedule**. Let's take an example to understand what a schedule in DBMS is.

Example of DBMS Schedule

The following sequence of operations is a schedule. Here we have two transactions T1 & T2 which are running concurrently. This schedule determines the exact order of operations that are going to be performed on database. In this example, all the instructions of transaction T1 are executed before the instructions of transaction T2, however this is not always necessary and we can have various types of schedules.

T1	T2
R(X)	
W(X)	
R(Y)	
	R(Y)
	R(X)
	W(Y)

.

Types of Schedules in DBMS

We have various types of schedules in DBMS. Like serial and non-serial (strict, cascade less and Recoverable).

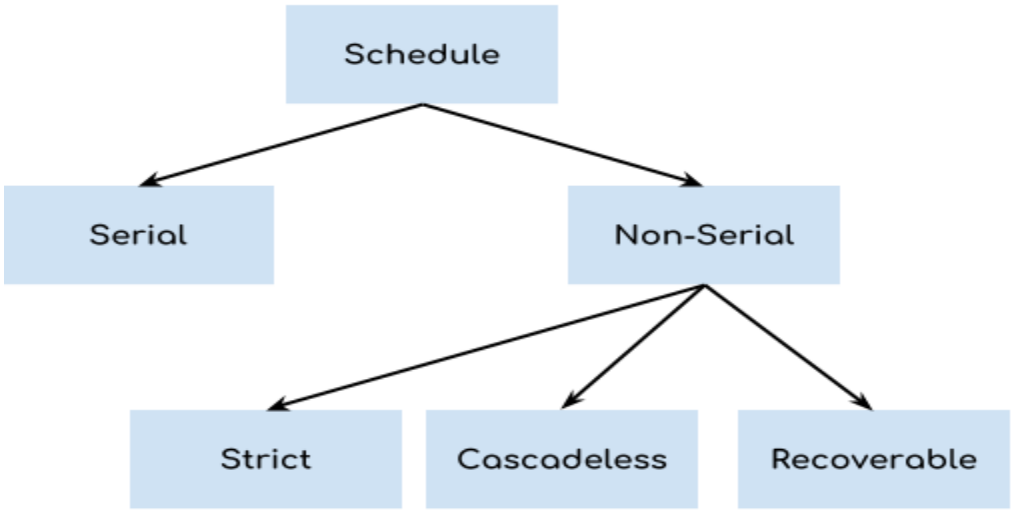


Figure 1.4: Types of Schedules in DBMS

Serial Schedule: In **Serial schedule**, a transaction is executed completely before starting the execution of another transaction. In other words, you can say that in serial schedule, a transaction does not start execution until the currently running transaction finished execution. This type of execution of transaction is also known as **non-interleaved** execution. For example here R refers to the read operation and W refers to the write operation. In this example, the transaction T2 does not start execution until the transaction T1 is finished.

```
T1          T2
----        ----
R(A)
R(B)
W(A)
commit
            R(B)
            R(A)
            W(B)
            commit
```

Strict Schedule: In this schedule, if the write operation of a transaction precedes a conflicting operation (Read/Write operation) of another transaction then the commit/abort operation of such transaction should precede the conflicting operation of other transaction.

For example, let's say we have two transactions Ta and Tb. The write operation of transaction Ta precedes the read or write operation of transaction Tb, so the commit or abort operation of transaction Ta should also precede the read or write of Tb.

```
Ta          Tb
-----       -----
R(X)
            R(X)
W(X)
commit
            W(X)
            R(X)
            commit
```

Here the write operation W(X) of Ta precedes the conflicting operation (Read or Write operation) of Tb so the conflicting operation of Tb had to wait the commit operation of Ta.

Cascadeless Schedule: In Cascadeless Schedule, if a transaction is going to perform read operation on a value, it has to wait until the transaction who is performing write on that value commits. For example, let's say we have two transactions Ta and Tb. Tb is going to read the value X after the W(X) of Ta then Tb has to wait for the commit operation of transaction Ta before it reads the X.

```
Ta          Tb
-----       -----
R(X)
W(X)
            W(X)
commit
            R(X)
            W(X)
            commit
```

Recoverable Schedule: In Recoverable schedule, if a transaction is reading a value which has been updated by some other transaction then this transaction can commit only after the commit of other transaction which is updating value.

For example, Here Tb is performing read operation on X after the Ta has made changes in X using W(X) so Tb can only commit after the commit operation of Ta.

```
Ta          Tb
-----       -----
R(X)
W(X)
            R(X)
            W(X)
            R(X)
commit
            commit
```

DBMS Serializability: When multiple transactions are running concurrently then there is a possibility that the database may be left in an inconsistent state. Serializability is a concept that helps us to check which schedules are serializable. A serializable schedule is the one that always leaves the database in consistent state. A serializable schedule always leaves the database in consistent state. A serial schedule is always a serializable schedule because in serial schedule, a transaction only starts when the other transaction finished execution. However a non-serial schedule needs to be checked for Serializability. A non-serial schedule of n number of transactions is said to be serializable schedule, if it is equivalent to the serial schedule of those n transactions. A serial schedule doesn't allow concurrency, only one transaction executes at a time and the other starts when the already running transaction finished.

Self-Check 3	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

1. Which one is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product?

   A. Test schedule B. Acceptance criteria C. Test plan D. All

2. _____ includes the testing steps or tasks, the target start and end dates, and responsibilities.

   A. Test schedule B. Acceptance criteria C. Test plan D. All

3. When multiple transactions are running concurrently then there needs to be a sequence in which the operations are performed because at a time only one operation can be performed on the database. This sequence of operations is known as _____.

   A. Test plan B. Load test C. DBMS Schedule D. All

4. In which transaction is executed completely before starting the execution of another transaction?

   A. Serial schedule B. Strict schedule C. Cascade less D. Recoverable

Part II: writing

1. List the Test Steps schedule

   a. _____
   b. _____
   c. _____
   d. _____
   e. _____
   f. _____
   g. _____
   h. _____
   i. _____
   j. _____
   k. _____

Note: **Satisfactory rating - 5 points** **Unsatisfactory - below 5 points**

You can ask you teacher for the copy of the correct answers.

Score = _____
Rating: _____

L #38	Lo#2. Test database performance

Instruction sheet

This learning guide is developed to provide you the necessary information regarding the following content coverage and topics:

- evaluating database performance
- identifying discrepancies in results
- identifying and documenting Areas that need enhancement and changes
- Modifying database
- Repeating performance testing until results achieved

This guide will also assist you to attain the learning outcomes stated in the cover page. Specifically, upon completion of this learning guide, you will be able to:

- evaluate database performance
- identify discrepancies in results
- identify and documenting Areas that need enhancement and changes
- Modify database
- Repeat performance testing until results achieved

Learning Instructions:

Read the specific objectives of this Learning Guide.

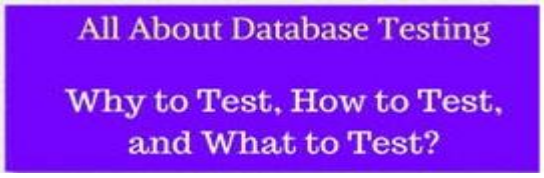
1. Follow the instructions described below.
2. Read the information written in the “Information Sheets”. Try to understand what are being discussed. Ask your trainer for assistance if you have hard time understanding them.
3. Accomplish the “Self-checks” which are placed following all information sheets.
4. Ask from your trainer the key to correction (key answers) or you can request your trainer to correct your work. (You are to get the key answer only after you finished answering the Self-checks).
5. If your performance is satisfactory proceed to the next learning guide,

Information sheet 1: evaluating database performance

2.1 evaluating database performance

Introduction to database performance test

Database performance testing is used to identify performance issues before deploying database applications for end users. Database load testing is used to test the database applications for performance, reliability, and scalability using varying user load. Load testing involves simulating real-life user load for the target database applications and is used to determine the behavior of the database applications when multiple users hit the applications simultaneously. Database testing includes performing data validity, data integrity testing, performance check related to database and testing of procedures, triggers and functions in the database.



The following aspects of a DB should be validated:

1. **Data Mapping:** In software systems, data often travels back and forth from the UI (user interface) to the backend DB and vice versa. So these are some aspects to watch for:
   - Check whether the fields in the UI/frontend forms are mapped consistently with the corresponding fields in the DB table. Typically this mapping information is defined in the requirements documents.
   - Whenever a certain action is performed at the front end of an application, a corresponding CRUD (Create, Retrieve, Update and Delete) action gets invoked at the back end. A tester will have to check if the right action is invoked and whether the invoked action in itself is successful or not.
2. **ACID Properties Validation:** Atomicity, Consistency, Isolation, and Durability. Every transaction a DB performs has to adhere to these four properties.

Figure 2. 1: ACID test

- **Atomicity** means that a transaction either fails or passes. This means that even if a single part of the transaction fails- it means that the entire transaction has failed. Usually, this is called the "all-or-nothing" rule.
- **Consistency**: A transaction will always result in a valid state of the DB
- **Isolation**: If there are multiple transactions and they are executed all at once, the result of the DB should be the same as if they were executed one after the other.
- **Durability**: Once a transaction is done and committed, no external factors like power loss or crash should be able to change it.

3. **Data Integrity:** For any of the CRUD Operations, the updated and most recent values/status of shared data should appear on all the forms and screens. The value should not be updated on one screen and display an older value on another one. When the application is under execution, the end-user mainly utilizes the 'CRUD' operations facilitated by the DB Tool**.**
   - **C: Create**: When user 'Save' any new transaction, 'Create' operation is performed.
   - **R: Retrieve**: When user 'Search' or 'View' any saved transaction, 'Retrieve' operation is performed.
   - **U: Update:** When user 'Edit' or 'Modify' an existing record, the 'Update' operation of DB is performed.
   - **D: Delete:** When a user 'Remove' any record from the system, 'Delete' operation of DB is performed.

Any database operation performed by the end-user is always one of the above four. So devise your DB test cases in a way to include checking the data in all the places it appears to see if it is consistently the same.

4. **Business Rule Conformity:** More complexity in Databases means more complicated components like relational constraints, triggers, stored procedures, etc. So testers will have to come up with appropriate SQL queries in order to validate these complex objects.

What to Test (Database Testing Checklist)

1. **Transactions:** When testing Transactions it is important to make sure that they satisfy the ACID properties. The following are the statements commonly used:
   - BEGIN TRANSACTION TRANSACTION#
   - END TRANSACTION TRANSACTION#

The Rollback statement ensures that the database remains in a consistent state.

   - ROLLBACK TRANSACTION#

After these statements are executed, use a Select to make sure the changes have been reflected.

   - SELECT * FROM TABLENAME <tables which involve the transactions>

2. **Database Schemas:** A Database Schema is nothing more than a formal definition of how the data is going to be organized inside a DB. To test it:
   - **Identify the Requirements based on which the Database operates.**

     Sample Requirements are**:**
     - ✓ Primary keys to be created before any other fields are created.
     - ✓ Foreign keys should be completely indexed for easy retrieval and search.
     - ✓ Field names starting or ending with certain characters.
     - ✓ Fields with a constraint that certain values can or cannot be inserted.
   - **Use one of the following methods according to the relevance:**
     - ✓ SQL Query DESC<table name> to validate the schema.
     - ✓ Regular expressions for validating the names of the individual fields and their values
     - ✓ Tools like SchemaCrawler

3. **Triggers:** When a certain event takes place on a certain table, a piece of code (a trigger) can be auto-instructed to be executed.

   For Example, a new student joined a school. The student is taking 2 classes: math and science. The student is added to the "student table". A Trigger could add the student to the corresponding subject tables once he is added to the student table. The common method to test is to execute the SQL query embedded in the Trigger independently first and record the result. Follow this up with executing the Trigger as a whole. Compare the results.

These are tested in both the Black-box and White-box testing phases.

- **White box testing**: Stubs and Drivers are used to insert or update or delete data that would result in the trigger being invoked. The basic idea is to just test the DB alone even before the integration with the front end (UI) is made.
- **Black box testing**:
  a. Since the UI and DB, integration is now available; we can Insert/Delete/Update data from the front end in a way that the Trigger gets invoked. Following that, Select statements can be used to retrieve the DB data to see if the Trigger was successful in performing the intended operation.
  b. The second way to test this is to directly load the data that would invoke the Trigger and see if it works as intended.

4. **Stored Procedures:** Stored Procedures are more or less similar to user-defined functions. These can be invoked by Call Procedure/Execute Procedure statements and the output is usually in the form of result sets.

These are stored in the RDBMS and are available for applications.

These are also tested during:

- **White box testing:** Stubs are used to invoke the stored procedures and then the results are validated against the expected values.
- **Black box testing:** Perform an operation from the front end (UI) of the application and check for the execution of the stored procedure and its results.

5. **Field Constraints**: The Default value, Unique value, and Foreign key:
   - Perform a front-end operation which exercises the Database object condition
   - Validate the results with a SQL Query.

Checking the default value for a certain field is quite simple. It is part of business rule validation. You can do it manually or you can use tools like QTP(). Manually, you can perform an action that will add value other than the default value of the field from the front end and see if it results in an error.

For the Foreign Key constraint validation use data loads that directly input data that violate the constraint and see if the application restricts them or not. Along with the back end data load, perform the front end UI operations too in a way that will violate the constraints and see if the relevant error is displayed.

Data Testing Activities

Database Tester Should Focus on Following Testing Activities:

1. **Ensure Data Mapping:** Data Mapping is one of the key aspects in the database and it should be tested rigorously by every software tester. Make sure that the mapping between different forms or screens of AUT and its DB is not only accurate but also per the design documents (SRS/BRS) or code. Basically, you need to validate the mapping between every front-end field with its corresponding backend database field. For all CRUD operations, verify that respective tables and records are updated when the user clicks 'Save', 'Update', 'Search' or 'Delete' from GUI of the application.

   What you need to verify:

   - Table mapping, column mapping, and Data type mapping.
   - Lookup Data Mapping.
   - Correct CRUD operation is invoked for every user action at UI.
   - CRUD operation is successful.

2. **Ensure ACID Properties of Transactions:** ACID properties of DB Transactions refer to the '**A**tomicity', '**C**onsistency', '**I**solation' and '**D**urability'. Proper testing of these four properties must be done during the database test activity. You need to verify that every single transaction satisfies the ACID properties of the database.

Figure 2.2: ACID properties

Let us take a simple example through below SQL code:

CREATE TABLE acidtest (A INTEGER, B INTEGER, CHECK (A + B = 100));

The ACID test table will have two columns A & B. There is an integrity constraint that the sum of values in A and B should always be 100.

- **Atomicity test** will ensure any transaction performed on this table is all or none i.e. no records are updated if any step of the transaction is failed.
- **Consistency test** will ensure that whenever the value in column A or B is updated, the sum always remains 100. It won't allow insertion/deletion/update in A or B if the total sum is anything other than 100.
- **Isolation test** will ensure that if two transactions are happening at the same time and trying to modify the data of the ACID test table, then these tractions are executing in isolation.
- **Durability test** will ensure that once a transaction over this table has been committed, it will remain so, even in the event of power loss, crashes, or errors. This area demands more rigorous, thorough and keen testing if your application is using the distributed database.

3. **Ensure Data Integrity**: Consider that different modules (i.e. screens or forms) of application use the same data in different ways and perform all the CRUD operations on the data. In that case, make sure that the latest state of data is reflected everywhere.

The system must show the updated and most recent values or the status of such shared data on all the forms and screens. This is called as Data Integrity.

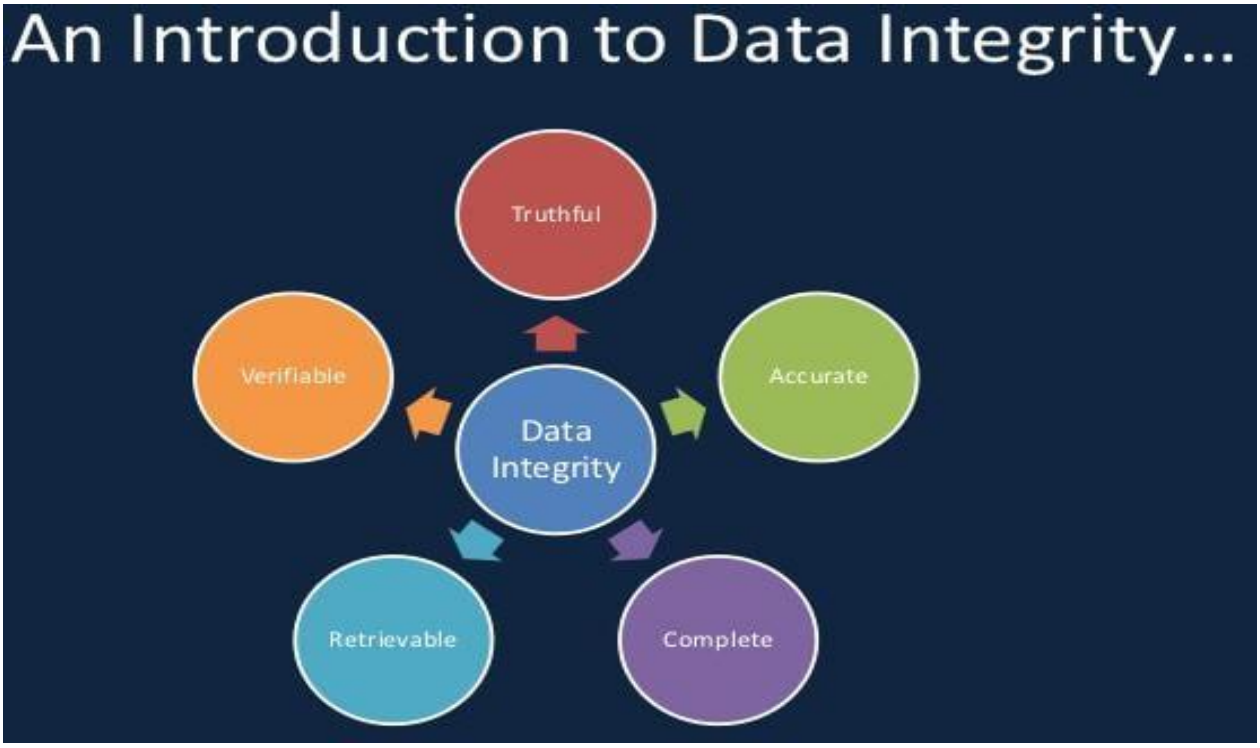


Figure 2.3: Data Integrity

Test cases for validating Database Data Integrity:

- Check if all the Triggers are in place to update reference table records.
- Check if any incorrect/invalid data exists in the major columns of each table.
- Try to insert wrong data in tables and observe if any failure occurs.
- Check what happens if you try to insert a child before inserting its parent (try to play with Primary and foreign keys).
- Test if any failure occurs if you delete a record that is still referenced by data in any other table.
- Check if replicated servers and databases are in sync.

4. **Ensure the Accuracy of the implemented Business Rules:** Today, Databases are not meant only to store the records. In fact, Databases have been evolved into extremely powerful tools that provide ample support to the developers to implement the

business logic at the DB level. Some simple examples of powerful features are 'Referential Integrity', Relational constraints, Triggers, and stored procedures.

So, using these and many other features offered by DBs, developers implement the business logic at the DB level. The tester must ensure that the implemented business logic is correct and works accurately.

The above points describe the four most important 'What To' of testing DB

How to Test the Database (Step-By-Step Process)

The general test process testing database is not very different from any other application. The following are the core steps:

Step 1) Prepare the environment

Step 2) Run a test

Step 3) Check test result

Step 4) Validate according to the expected results

Step 5) Report the findings to the respective stakeholders

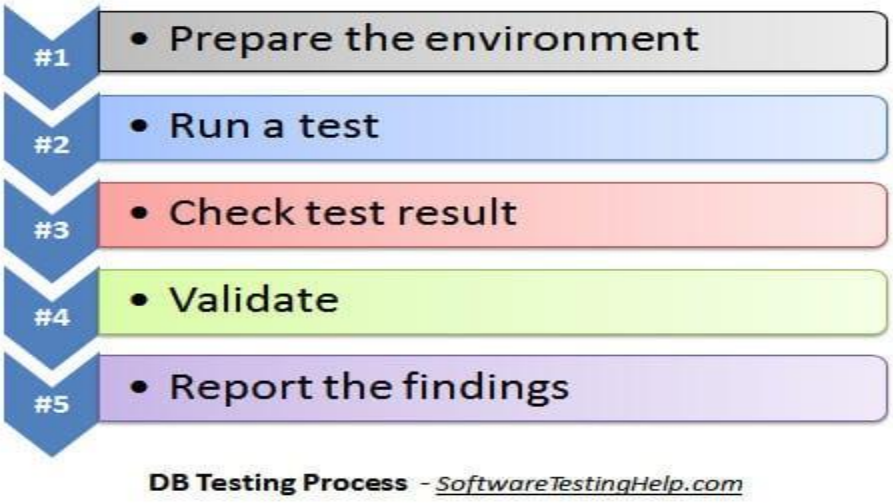


Figure 2.4: DB Testing process shows how to test

Usually, SQL queries are used to develop the tests. The most commonly used command is "Select". Select * from <tablename> where <condition>

Apart from Select, SQL has 3 important types of commands:

1. DDL: Data definition language
2. DML: Data manipulation language
3. DCL: Data control language

The syntax for the most commonly used statements.

Data Definition language: Uses CREATE, ALTER, RENAME, DROP and TRUNCATE to handle tables (and indexes).

Data Manipulation language: Includes statements to add, update and delete records.

Data control language: Deals with giving authorization to users for manipulation and access to the data. Grant and Revoke are the two statements used.

Grant syntax:

Grant select/update

On <table name>

To <user id1, user id2…useridn>;

Revoke syntax:

Revokeselect/update

on <table name>

from<user id1, user id2…useridn>;

Some Practical Tips

1. **Write Queries yourself:** To test the Database accurately, the tester should have very good knowledge of SQL and DML (Data Manipulation Language) statements. The tester should also know the internal DB structure of AUT. You can combine GUI and data verification in respective tables for better coverage. If you are using the SQL server then you can make use of SQL Query Analyzer for writing queries, executing them and retrieving results. This is the best and robust way of testing a database when the application is of a small or medium level of complexity. If the application is very complex then it may be hard or impossible for the tester to write all the required SQL queries. For complex queries, you take help from the developer. I always recommend this method as it gives you confidence in testing and also enhances your SQL skills.

2. **Observe the data in each table:** You can perform data verification using the results of CRUD operations. This can be done manually by using application UI when you know the database integration. But this can be a tedious and cumbersome task when there is huge data in different database tables. For Manual Data Testing, the Database tester must possess a good knowledge of database table structure.
3. **Get queries from the developers:** This is the simplest way to test the Database. Perform any CRUD operation from GUI and verify its impacts by executing the respective SQL queries obtained from the developer. It neither requires a good knowledge of SQL nor requires a good knowledge of the application's DB structure. But this method needs to be used cautiously. What if the query given by the developer is semantically wrong or does not fulfill the user's requirement correctly? The process will simply fail to validate data.
4. **Make use of Database Automation Testing tools:** There are several tools available for the Data Testing process. You should choose the correct tool as per your needs and make the best use of it.

Database performance can be defined as the rate at which a **database** management system (DBMS) supplies information to users. The performance of accessing and modifying data in the database can be improved by the proper allocation and application of resources. Optimization speeds up query performance. Database performance software identifies bottlenecks and points of contention, monitors workload and throughput, and manages system and DBMS resource usage.

Ensuring that database systems perform effectively is a core requirement of modern IT management. There are several types of tools that can help database administrators and other IT professionals monitor, manage and improve the performance of databases and the applications that access them. But before you can consider whether you need database performance software, it's important to first define *database performance*.

At a high level, database performance can be defined as the rate at which a database management system (DBMS) supplies information to users. But to better understand what influences database performance; let's examine the following five factors:

1. **Workload** is the activity the DBMS is requested to perform, which drives the processing demands placed on the system. The workload can include a combination of transactions, batch jobs, ad hoc queries, reporting and analysis applications, database utilities and system commands. Workload can fluctuate drastically by day, hour or even minute. Sometimes it's steady and predictable, but at other times it can spike, such as a heavy transaction workload on Black Friday for a database supporting a retailer's e-commerce site.
2. **Throughput** is a measurement of a system's ability to process data. It's a composite of I/O and CPU speed, and it can be affected by the parallel capabilities of the database server and the efficiency of the operating system and system software.
3. **Resources** are the hardware and software tools at the disposal of the system such as database kernel, disk space, memory, cache controllers and microcode. The performance of accessing and modifying data in the database can be improved by the proper allocation and application of resources.
4. **Optimization** speeds up query performance. All types of systems can be optimized, but relational databases are unique in that query optimization is primarily accomplished internal to the DBMS. Additional factors do need to be considered as well, including SQL formulation and database parameters, to enable the database optimizer to create the most efficient access paths.
5. **Contention** results when the demand for a particular system resource is high, for example, if two or more components of the workload are attempting to use a single resource in a conflicting way, such as dual updates to the same piece of data. As contention increases, throughput decreases.

So, database performance can be more accurately defined as the optimization of resource usage to increase throughput and minimize contention, enabling the largest possible workload to be processed.

Types of database performance management software

Before we delve into the different types of database performance management tools, we need to differentiate between performance monitoring and performance management.

Although they mean different things, it's easy to confuse the two. In fact, performance monitoring is an aspect of performance management, which consists of these three broad components:

- Monitoring a database system to find problems as they occur.
- Analyzing data from the system to determine a corrective action.
- Implementing a fix to alleviate the problem.

Database performance software can aid in all three areas. But some simply monitor systems or fix problems, while others deliver combined functionality.

This software can also be broken down by the type of database performance issues it addresses. For example, managing the performance of database applications involves the following three components:

1. **The DBMS.** This system software enables data to be stored and accessed by programs. It must interact with other system software and hardware, requiring proper configuration to ensure it functions accurately and performs satisfactorily. Additionally, there are many system parameters used to configure the resources to be used by the DBMS, as well as its behavior. This includes important performance criteria such as memory capacity, I/O throughput and locking of data pages.
2. **Database structures.** The design of databases, tables and indexes can also impact database performance. Issues include the physical design of the database, disk usage, number of tables, index design and data definition language parameters. How the data is organized must also be managed. And as data is modified in the database, its efficiency will degrade. Reorganization and defragmentation are required to periodically remedy disorganized data.
3. **SQL and application code.** Coding efficient SQL statements can be complicated because there are many different ways to write SQL that return the same results. But the efficiency and performance of each formulation can vary significantly. DBAs need tools that can monitor the SQL code that's being run, show the access paths it uses and provide guidance on how to improve the code.

When do you need database performance tools?

Overall, database performance software can identify bottlenecks and points of contention, monitor workload and throughput, review SQL performance and optimization, monitor storage space and fragmentation, and view and manage your system and DBMS resource usage. Database performance problems are most commonly brought to light by end users who aren't able to do their jobs effectively. Usually, their concerns are somewhat vague and high-level, such as "the system is sluggish" or "my screen isn't working as fast as it used to." This requires pinpointing the exact problem so that tools can be used to find a solution. Database performance management tools can help to find the problem as well as to formulate and implement a solution to the problem.

Database performance software is essential for organizations that have service-level agreements in place to manage the performance of their applications. These tools can help ensure that adequate levels of service are delivered to all IT users in accordance with business priorities and at acceptable cost.

Many organizations use more than one DBMS, and their DBAs are tasked with ensuring the performance of all of their company's database systems. But each DBMS has different interfaces, parameters and settings that affect how it performs. Heterogeneous database performance tools can simplify the confusion by using intelligent interfaces to make these disparate components look and feel similar from DBMS to DBMS.

Example

Before you begin searching a database, you will want to figure out whether the database actually contains the information you need for your research. Spending a few minutes evaluating the database can save you valuable time and help you leverage your searches to get the best results. The most common method used to decide the most appropriate database for research is disciplinary. Many times you can determine the subject content of a database by its title. For example, Find a list of school Libraries' databases by subject or discipline by clicking on the "Category" tab, and then scrolling through the list of entries in the "Subject Search" box. This only work on Internet;

Besides determining the disciplinary focus of a database, however, there are other elements of evaluating databases you should be aware of. Namely the following:

1) Types of material covered in database: Books? journals? web pages? newspaper articles? Conference proceedings?
2) Publisher of database: Does the database provide any information about the publisher? Is it an academic or commercial publisher?
3) Time coverage: What years of publications does the database cover? Some databases only cover from ~1990 to current. Other databases such as JStor and PsychInfo cover articles back to the 1890s!
4) Update schedule of database: Is the database updated every day? once a month? quarterly?

Database Testing is a type of software testing that checks the schema, tables, triggers, etc. of the Database under test. It also checks data integrity and consistency. It may involve creating complex queries to load/stress test the Database and check its responsiveness.

Importance of database Testing

Database Testing is Important in software testing because it ensures data values and information received and stored into database are valid or not. Database testing helps to save data loss, saves aborted transaction data and no unauthorized access to the information. Database is important for any software application hence testers must have good knowledge of SQL for database testing. The GUI is usually given the most emphasis by the test and development team members since the Graphical User Interface happens to be the most visible part of the application. However, what is also important is to validate the information that is the heart of the application, also known as DATABASE.

Let us consider a Banking application wherein a user makes transactions. Now from Database Testing or DB Testing viewpoint the following, things are important:

1. The application stores the transaction information in the application database and displays them correctly to the user.
2. No information is lost in the process.

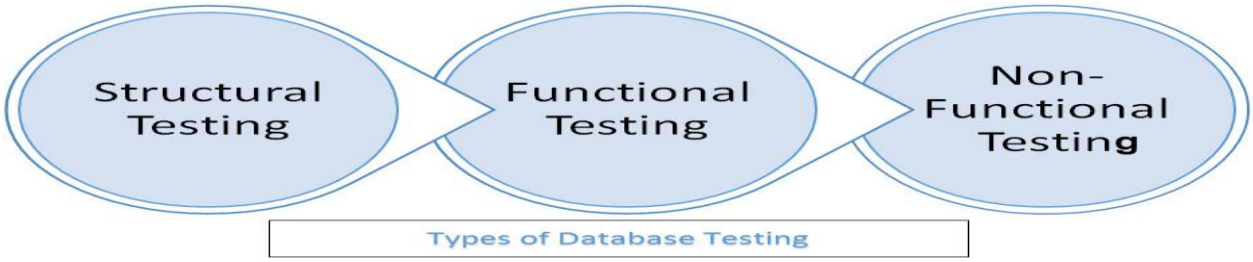
3. No partially performed or aborted operation information is saved by the application.
4. No unauthorized individual is allowed to access the user's information.

To ensure all these above objectives, we need to use data validation or data testing.

Differences between User-Interface Testing and Data Testing

User-Interface testing	Database or Data testing
This type of testing is also known as Graphical User Interface testing or Front-end Testing.	This type of testing is also known as Backend Testing or data testing.
This type of testing chiefly deals with all the testable items that are open to the user for viewership and interaction like Forms, Presentation, Graphs, Menus, and Reports, etc. (created through VB, VB.net, VC++,C# Delphi - Front-end Tools).	This type of testing chiefly deals with all the testable items that are generally hidden from the user for viewership. These include internal processes and storage like Assembly, DBMS like Oracle, SQL Server, MYSQL, etc.
This type of testing includes validating the • text boxes • select dropdowns • calendars and buttons • Page navigation • display of images • Look and feel of the overall application	This type of testing involves validating: • the schema • database tables • columns • keys and indexes • stored procedures triggers • database server validations • validating data duplication
The tester must be thoroughly knowledgeable about the business requirements as well as the usage of the development tools and the usage of automation frameworks and tools.	To be able to perform backend testing must the testers have a strong background in the database server and Structured Query Language concepts.

Types of Database Testing



Types of Database Testing

The 3 types of Database Testing are

1. Structural Testing
2. Functional Testing
3. Non-functional Testing

Structural Database Testing: It is a database testing technique that validates all the elements inside data repository that are mainly used for data storage and which are not allowed to be directly manipulated by end-users. The validation of database servers is also an important consideration in structural database testing. A successful completion of this testing need in SQL queries mastery.

Schema Testing: Schema testing in database testing validates various schema formats associated with the database and verifies whether the mapping formats of tables/views/columns are compatible with mapping formats of user interface. The main purpose of schema testing is to ensure the schema mapping between front-end and back-end are similar. Thus, it is also referred to as **mapping testing.**

The most important checkpoints for schema testing are.

1. Validation of the various schema formats associated with the databases. Many times the mapping format of the table may not be compatible with the mapping format present in the user interface level of the application.
2. There is a need for verification in the case of unmapped tables/views/columns.
3. There is also a need to verify whether heterogeneous databases in an environment are consistent with the overall application mapping.

Database Testing tools for validating database schemas.

- DB Unit that is integrated with Ant is very suitable for mapping testing.
- SQL Server allows the testers to be able to check and to query the schema of the Database by writing simple queries and not through code.

For example, if the developers want to change a table structure or delete it, the tester would want to ensure that all the Stored Procedures and Views that use that table are compatible with the particular change. Another example could be that if the testers want to check for schema changes between 2 databases, they can do that by using simple queries.

Database Table, Column Testing

Look into various checks for database and column testing.

1. Whether the mapping of the database fields and columns in the backend is compatible with those mappings in the front-end?
2. Validation of the length and naming convention of the database fields and columns as specified by the requirements.
3. Validation of the presence of any unused/unmapped database tables/columns.
4. Validation of the compatibility of the **Data type** and **Field lengths** of the back-end database columns with that of those present at the front-end of the application.
5. Whether the database fields allow the user to provide desired user inputs as required by the business requirement specification documents.

Keys and indexes testing: The Important checks for keys and indexes are.

1. Check whether the required Primary Key and Foreign Key Constraints have been created on the required tables.
2. Check whether the references for foreign keys are valid.
3. Check whether the data type of the primary key and the corresponding foreign keys are the same in the two tables.
4. Check whether the required naming conventions have been followed for all the keys and indexes.
5. Check the size and length of the required fields and indexes.
6. Whether the required Clustered indexes and Non Clustered indexes have been created on the required tables as specified by the business requirements.

Stored Procedures Testing

Important tests to check stored procedures are:

1. Whether the development team did adopt the required coding standard conventions and exception and error handling for all the stored procedures for all the modules for the application under test.
2. Whether the development team did cover all the conditions/loops by applying the required input data to the application under test?
3. Whether the development team did properly apply the TRIM operation whenever data is fetched from the required tables in the Database?
4. Whether the manual execution of the Stored Procedure provides the end-user with the required result?
5. Whether the manual execution of the Stored Procedure ensures the table fields are being updated as required by the application under test?
6. Whether the execution of the Stored Procedures enables the implicit invoking of the required triggers?
7. Validation of the presence of any unused stored procedures.
8. Validation for Allow Null condition which can be done at the database level.
9. Validation of the fact that all the Stored Procedures and Functions have been successfully executed when the Database under test is blank.
10. Validation of the overall integration of the stored procedure modules as per as the requirements of the application under test.

Trigger Testing

1. Whether the required coding conventions have been followed during the coding phase of the Triggers?
2. Check whether the triggers executed for the respective DML transactions have fulfilled the required conditions.
3. Whether the trigger updates the data correctly once they have been executed?
4. Validation of the required Update/Insert/Delete triggers functionality in the realm of the application under test.

Database Server Validations

1. Check the database server configurations as specified by the business requirements.
2. Check the authorization of the required user to perform only those levels of actions that are required by the application.
3. Check that the database server is able to cater to the needs of the maximum allowed number of user-transactions as specified by the business requirement specifications.

Functional Database Testing: It is a type of database testing that is used to validate the functional requirements of a database from the end-user's perspective. The main goal of functional database testing is to test whether the transactions and operations performed by the end-users which are related to the database works as expected or not.

Following are the basic conditions that need to be observed for database validations.

- Whether the field is mandatory while allowing NULL values on that field?
- Whether the length of each field is of sufficient size?
- Whether all similar fields have the same names across tables?
- Whether there are any computed fields present in the Database?

This particular process is the validation of the field mappings from the end-user viewpoint. In this particular scenario, the tester would perform an operation at the database level and then would navigate to the relevant user interface item to observe and validate whether the proper field validations have been carried out or not.

The vice versa condition whereby, first operation is carried out by the tester at the user interface, and then the same is validated from the back end is should also be done.

Checking data integrity and consistency: The **f**ollowing checks are important

1. Whether the data is logically well organized?
2. Whether the data stored in the tables is correct and as per the business requirements?
3. Whether there are any unnecessary data present in the application under test?

4. Whether the data has been stored as per as the requirement with respect to data which has been updated from the user interface?
5. Whether the TRIM operations performed on the data before inserting data into the Database under test?
6. Whether the transactions have been performed according to the business requirement specifications and whether the results are correct or not?
7. Whether the data has been properly committed if the transaction has been successfully executed?
8. Whether the data has been rolled backed successfully if the transaction has not been executed successfully by the end-user?
9. Whether the data has been rolled backed if the transaction has not been executed successfully and multiple heterogeneous databases have been involved in the transaction in question?
10. Whether all the transactions have been executed by using the required design procedures as specified by the system business requirements?

Login and User Security

The validations of the login and user security credentials need to take into consideration the following things.

1. Whether the application prevents the user from proceeding further in the application in case of a invalid username but valid password, valid username but invalid password and invalid username and invalid password.
2. Whether the user is allowed to perform only those specific operations which are specified by the business requirements?
3. Whether the data is secured from unauthorized access?
4. Whether there are different user roles created with different permissions?
5. Whether all the users have required levels of access on the specified Database as required by the business specifications?
6. Check that sensitive data like passwords, credit card numbers are encrypted and not stored as plain text in Database. It is a good practice to ensure all accounts should have passwords that are complex and not easily guessed.

Non-functional testing: Non-functional testing in the context of database testing can be categorized into various categories as required by the business requirements. These can be load testing, Stress Testing, Security Testing, Usability Testing, and Compatibility Testing, and so on. The load testing, as well as stress testing, which can be grouped under the gamut of Performance Testing, serves two specific purposes when it comes to the role of non-functional testing.

Risk quantification- Quantification of risk helps the stakeholders to ascertain the various system response time requirements under required levels of load. This is the original intent of any quality assurance task. We need to note that load testing does not mitigate risk directly, but through the processes of risk identification and risk quantification, presents corrective opportunities and an impetus for remediation that will mitigate risk.

Minimum system equipment requirement- Minimum system configuration allows the system to meet the formally stated performance expectations of stakeholders. So that extraneous hardware, software, and the associated cost of ownership can be minimized. This particular requirement can be categorized as the overall business optimization requirement.

Load Testing: The purpose of any load test should be clearly understood and documented. The following types of configurations are a must for load testing.

1. The most frequently used user transactions have the potential to impact the performance of all of the other transactions if they are not efficient.
2. At least one non-editing user transaction should be included in the final test suite, so that performance of such transactions can be differentiated from other more complex transactions.
3. The more important transactions that facilitate the core objectives of the system should be included, as failure under a load of these transactions has, by definition, the greatest impact.
4. At least one editable transaction should be included so that performance of such transactions can be differentiated from other transactions.
5. Optimum response time under huge number of virtual users for all the prospective requirements.
6. Effective times for fetching of various records.

Database Stress Testing: It is a testing method used to stress test database system with heavy load such that it fails at some point. This helps in identifying the breakdown point of database system. It requires proper planning and efforts in order to avoid over usage of resources. Data stress testing is also known as torturous testing or fatigue testing. Important stress testing tools are LoadRunner and JMeter.

Most common occurring issues during database testing

1. A significant amount of overhead could be involved to determine the state of the database transactions.
   - **Solution**: The overall process planning and timing should be organized so that no time and cost based issues appear.

2. New test data have to be designed after cleaning up of the old test data.
   - **Solution**: A prior plan and methodology for test data generation should be at hand.

3. An SQL generator is required to transform SQL validates in order to ensure the SQL queries are apt for handling the required database test cases.
   - **Solution**: Maintenance of the SQL queries and their continuous updating is a significant part of the overall testing process which should be part of the overall test strategy.

The above mentioned prerequisite ensure that the set-up of the database testing procedure could be costly as well as time consuming.

- **Solution**: There should be a fine balance between quality and overall project schedule duration.

Myths or Misconceptions related to Database Testing

Database Testing requires plenty of expertise and it is a very tedious job

- **Reality**: Effective and efficient Database Testing in Software Testing provides long-term functional stability to the overall application thus it is necessary to put in hard work behind it.

Database testing adds extra work bottleneck

- **Reality**: On the contrary, database testing adds more value to the overall work by finding out hidden issues and thus pro-actively helping to improve the overall application.

Database testing slows down the overall development process

- **Reality**: Significant amount of database testing helps in the overall improvement of quality for the database application.

Database testing could be excessively costly

- **Reality**: Any expenditure on database testing is a long-term investment which leads to long-term stability and robustness of the application. Thus expenditure on Database Testing or SQL Testing is necessary.

Best Practices in database performance evaluation

- All data including the metadata as well as the functional data needs to be validated according to their mapping by the requirement specification documents.
- Verification of the test data which has been created by / in consultation with the development team needs to be validated.
- Validation of the output data by using both manual as well as automation procedures.
- Deployment of various techniques such as the cause effect graphing technique, equivalence partitioning technique and boundary-value analysis technique for generation of required test data conditions.
- The validation rules of referential integrity for the required database tables also need to be validated.

- The selection of default table values for validation on database consistency is a very important concept Whether the log events have been successfully added in the Database for all required login events
- Does scheduled jobs execute in timely manner?
- Take timely backup of Database.

Self-Check 1	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

1. Any database operation performed by the end-user is always at list one of the following except one.

A. Data mapping B. ACID properties D. Data integrity D. Check list

2. It is a database testing technique that validates all the elements inside data repository that are mainly used for data storage and which are not allowed to be directly manipulated by end-users.

   A. Structural Testing B. Functional Testing C. Non-functional Testing D. All

3. The main purpose of schema testing is to ensure the schema mapping between front-end and back-end are similar. This is referred to as _____

A. Structural Testing B. Functional Testing C. Non-functional Testing D. Mapping testing.

4. Data stress testing is also known as

   A. Torturous testing B. fatigue testing C. A and B D. load test

5. One of the best practice of database performance evaluation is

   A. Unit testing

   B. Take timely backup of Database.

   C. System testing

   D. Timely restore of Database.

Note: **Satisfactory rating - 5 points** **Unsatisfactory - below 5 points**

You can ask you teacher for the copy of the correct answers.

Score = _____
Rating: _____

Information sheet 2: identifying discrepancies in results

2.2 identifying discrepancies in results

Data discrepancies can surface as missing records, incorrect values, or fields not being correctly typed. Pinpointing the cause of a data discrepancy has the potential to require quite a bit of investigation. To increase efficiency, it is recommended using the following resources to perform quick checks on some of the more obvious and common causes.

The steps you should follow in Data Discrepancy Troubleshooting guide:

1. Data Loading
2. Downtime
3. Queries
4. Discrepancy Consistencies
5. Replication Frequency
6. Replication Keys
7. Contacting Support

Data Loading: While Stitch is designed to quickly and efficiently process large amounts of data, it can take some time to replicate and load your data into your data warehouse. What looks like missing data may actually be incomplete processing, meaning Stitch hasn't finished loading all the data. Processing time can be affected by a variety of factors:

- The volume of data being replicated,
- The integration's Replication Frequency, and
- API quotas for SaaS (Software as a service) integrations.

Most data discrepancies can be solved by simply waiting and giving Stitch time to process and load the data.

Downtime: If a SaaS integration provider (ex: Salesforce) is undergoing maintenance or experiencing downtime, Stitch may be unable to replicate data. We recommend checking

the provider's status page for reported outages. Stitch may also occasionally encounter performance issues.

Queries: Before reporting a data discrepancy to Stitch support, we recommend that you double-check how you're querying the data.

- **Are you using a SQL client to directly query your data warehouse?**

  Make sure you're using a SQL client to **directly query** your data warehouse. This will eliminate the possibility of:

  1. Report refresh lags, which can occur in visualization tools like Tableau or Mode,
  2. Third party defects or downtime, and
  3. Any other type of data delay.

- **Are you querying for the same timeframe in the data source and data warehouse?**

  Unless you specify a start date, the majority of SaaS integrations will sync historical data going back **one year** from the Stitch connection date. When querying in your data warehouse, check that the timeframe you're querying for matches up with the integration's start date. If you find that the integration's start date was set incorrectly, you can reset it to queue a full re-sync.

- **Have you accounted for any time zone variation?**

  Keep in mind that your data warehouse may handle timestamp data in a specific way. For example: data warehouses will convert timestamps to UTC even if your data source is configured to report in a specific time zone.

Discrepancy Consistencies: When investigating a data discrepancy, look for consistencies such as records missing over a specific timeframe or issues that only affect certain records or data types. For example: formula fields in Sales force can occasionally cause data discrepancies due to how they're updated.

Replication Frequency: If the missing records were created very recently, you may need to wait for an update of your data to complete before they appear in your data warehouse.

Replication Keys: This section is only applicable to database integrations. When a table in database integration is initially set to use Incremental Replication, a Replication Key must be defined. For Stitch to accurately replicate data, Replication Keys must align with how data in the table is updated. If the table in question is set to use Incremental Replication, keep in mind that:

1. Stitch won't capture hard deletes.
2. Replication Key columns with NULL values are only replicated during an integration's initial replication job.
3. Records that are updated over time should use a modification timestamp to ensure updates are captured.
4. Mongo Replication Keys have additional considerations. For example: multiple data types in the Replication Key column can lead to missing data.
5. BigQuery destinations only support Append-Only Incremental Replication**.** What look like duplicate data may actually be updated records being appended to a table.

Row count discrepancies describe discrepancies that affect the number of records in your data warehouse. Complete records may be missing, duplicated, etc. **Field value discrepancies** describe discrepancies that affect values in individual columns.

Row Count Discrepancies: For row count discrepancies, you should:

1. **3-5 examples of records that exist in the source but not in your data warehouse**. You can check the entire record, but at the very least it is needed:
    - The Primary Key
    - The Replication Key
    - The field with the discrepancy
2. The results for the queries listed below, **run in the data source** for [source_integration_schema].[table_name]:
    - MIN(DATE)
    - MAX(DATE)

- MIN(REPLICATION_KEY)
- MAX(REPLICATION_KEY)
- COUNT(*)

3. The results for the queries listed below, **run in the data warehouse** for [data_warehouse_schema].[table_name]:
   - MIN(DATE)
   - MAX(DATE)
   - MIN(REPLICATION_KEY)
   - MAX(REPLICATION_KEY)
   - COUNT(*)
4. For SaaS integrations, whenever possible, you should check:
   - Raw exports showing row-level data that illustrates the discrepancy
   - Screenshots from the source integration's UI that illustrate discrepancy
   - Exact API calls and full responses that illustrate the discrepancy (**make sure you exclude your API key**)

Field Value Discrepancies: For field value discrepancies,:

1. **3-5 examples of discrepancies between the source integration and your data warehouse**. Please include:
   - The record's id
   - The field with the discrepancy
   - The updated_at value, if applicable
2. **For database integrations:**
   - Confirmation that the Replication Method is appropriately capturing changed values
   - Confirmation that the Replication Key is being appropriately populated and does not contain NULL values
3. **For SaaS integrations**, whenever possible, please provide us with:
   - Raw exports showing row-level data that illustrates the discrepancy
   - Screenshots from the UI that illustrate discrepancy
   - Exact API calls and full responses that illustrate discrepancy (make sure you exclude your API key)

Additional resources

- **Database Integration Table Name Collisions:** In database integrations, if the names of multiple tables cannibalize to the same name even if they're from different source databases or schemas - name collisions and data discrepancies can occur. This applies to any database integration available in Stitch.
- **Missing Columns & NULL Values:** If you've noticed some missing columns or data from your data warehouse, the root cause may be NULL values.
- **Missing Mongo Data Due to Fields with Multiple Data Types:** Missing some Mongo data? The root cause may be multiple data types in the Replication Key or Primary Key (_id) fields.
- **Missing Segment Data & Selective Integration Snippets:** If you've noticed some missing data from your Segment integration, the culprit might be the selective integration snippet on your website.
- **Mongo Fields Missing from Replication Key Menu:** If you don't see all the fields you expect to in the Replication Key field for you Mongo integration, the root cause may be insufficient permissions or a lack of field indexing.
- **MySQL TINYINT(1)/boolean Columns Stored as BIT:** If you've noticed that some MySQL TINYINT(1) columns are displaying as BIT in Stitch, it's usually due to how the MySQL driver converts this data type.
- **Non-Replicating Data & Unsupported Data Types:** If a table isn't replicating into your data warehouse, it may be because one or more of the columns in the table contains an unsupported data type.
- **PostgreSQL Data Types Stored as Strings:** When certain Postgres data types are replicated, they'll be stored as strings in your data warehouse.
- **PostgreSQL Read Replicas and Slow Replication:** If you connected a PostgreSQL read replica as database integration and are experiencing extremely slow replication, the root cause may be the database's standby settings.
- **Stale Salesforce Data & Formula Fields:** If you've noticed some out-of-date Salesforce data in your data warehouse, the root cause may be a formula field.

Self-Check 2	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

1. Write the recommended resources to perform quick checks on some of the more obvious and common causes of data discrepancies troubleshooting guide.
   2. _____
   3. _____
   4. _____
   5. _____
   6. _____
   7. _____
   8. _____

Note: **Satisfactory rating - 5 points** **Unsatisfactory - below 5 points**

You can ask you teacher for the copy of the correct answers.

Score = _____
Rating: _____

2.3 identifying and documenting Areas that need enhancement and changes

Documenting Changes

Effective management of change provides a structured, consistent, and measurable change environment to be utilized across an organization and is a critical component in the success of its daily business. Its goal is to increase awareness and understanding of proposed changes across the organization and ensure that all changes are made in a thoughtful way that minimizes negative impact to services and customers. An organization should have a document that defines the implementation of Change Management procedure. The computing systems, networks, peripherals, and associated facilities are subject to continuous changes driven by new technology, evolving business requirements, changing contractual requirements and growing regulatory policies. Effective change management applies to both systems and supporting infrastructure, and is a necessary component for the continuous success and growth of the organization.

The key steps of change management process are:

- Planning,
- Evaluation,
- Documentation,
- Review,
- Approval,
- Communication,
- Implementation, and
- Post implementation review.

Documenting changes do not only happen during the documentation step. Changes need to be documented and updated throughout the change management process. Nowadays, most organizations use change management software to manage change requests (CR), review, and approval workflows. Smaller organizations with resource constrains may still use simple Change Request Forms (CRF) to manually manage the CRs, review, and approval processes. Important information must be captured for each CR regardless the change management tool (sophisticated software or CRF). The mandated information is required for adequate review by others for approving the requested change.

Self-Check 3	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

1. Write the key steps to change management process.
   a. _____
   b. _____
   c. _____
   d. _____
   e. _____
   f. _____
   g. _____
   h. _____

Note: **Satisfactory rating - 5 points** **Unsatisfactory - below 5 points**

You can ask you teacher for the copy of the correct answers.

Score = _____
Rating: _____

Information sheet 4: Modifying database

2.4 Modifying database

The Database Modification is generation of SQL script that leads your database to the current state of your diagram. Database modification usually causes multiple complex statements for database structure modification. It is possible that some of them may not execute correctly due to some physical reason. Once you have changed your diagram, it's usually necessary to apply these changes to your database. It's easy to do with the Database Modification tool. The Database Modification is generation of SQL script that leads your database to the current state of your diagram.

Database modification usually causes multiple complex statements for database structure modification. It is possible that some of them may not execute correctly due to some physical reason. It's recommended to make a backup of your database before applying structure changes to database.

You can modify database in two ways: Directly execute a modification script on a MySQL server. Please examine Connect to a

- Database section to explore the database connection process;
- Generate a modification script for executing at a MySQL server.

In both cases, the database modification commands are saved in a script file. You must always provide the path to the script file. Sometimes you need to apply changes in your diagram to physical database without leading whole database structure to current state of your diagram. For example if there are some objects in database that are not covered by your diagram or if you want to apply only particular set of diagram changes to database. You can do this using Diagram Revisions and Compare Diagram tool.

Database Modification tool works:

1. Reverse engineers your existing MySQL database.
2. Compares the result with your current database diagram.
3. Creates a list that contains differences between database objects and diagram objects. Then lets you to examine this list and to select changes you want to apply to database.

4. After analyzing the difference list, creates necessary SQL statements, which modify database structure.

If some table is going to be modified, **Database Designer for MySQL** makes a backup copy of that table, so you can restore data and table structure later on if there will be some errors during the table structure alteration. Those backup tables have *_tmp_* string appended before their names. Example `*Cars*` table will have `*_tmp_Cars*` backup copy.

Step by step guide

To modify your database start **Database Modification** tool by selecting the **Modify Database** ( ) item on **Database** tab of the **Ribbon** or pressing **Ctrl+M**. **Database Designer for MySQL** asks you with **Database Connection Manager** what database you want to modify.

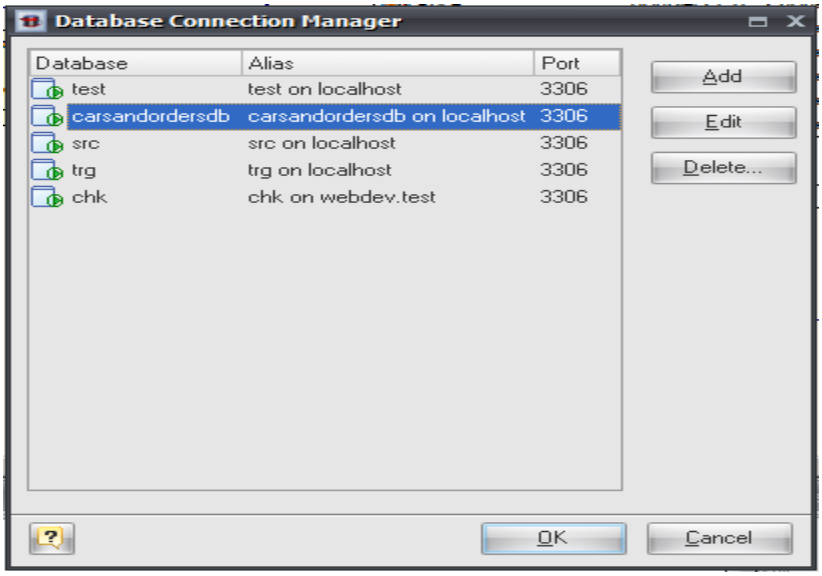
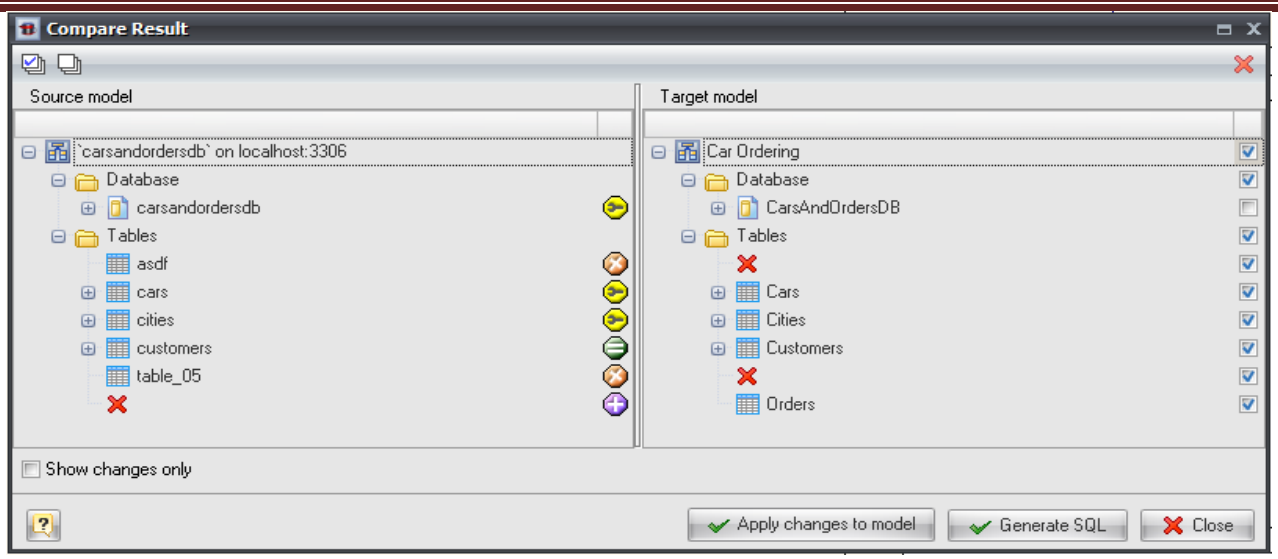
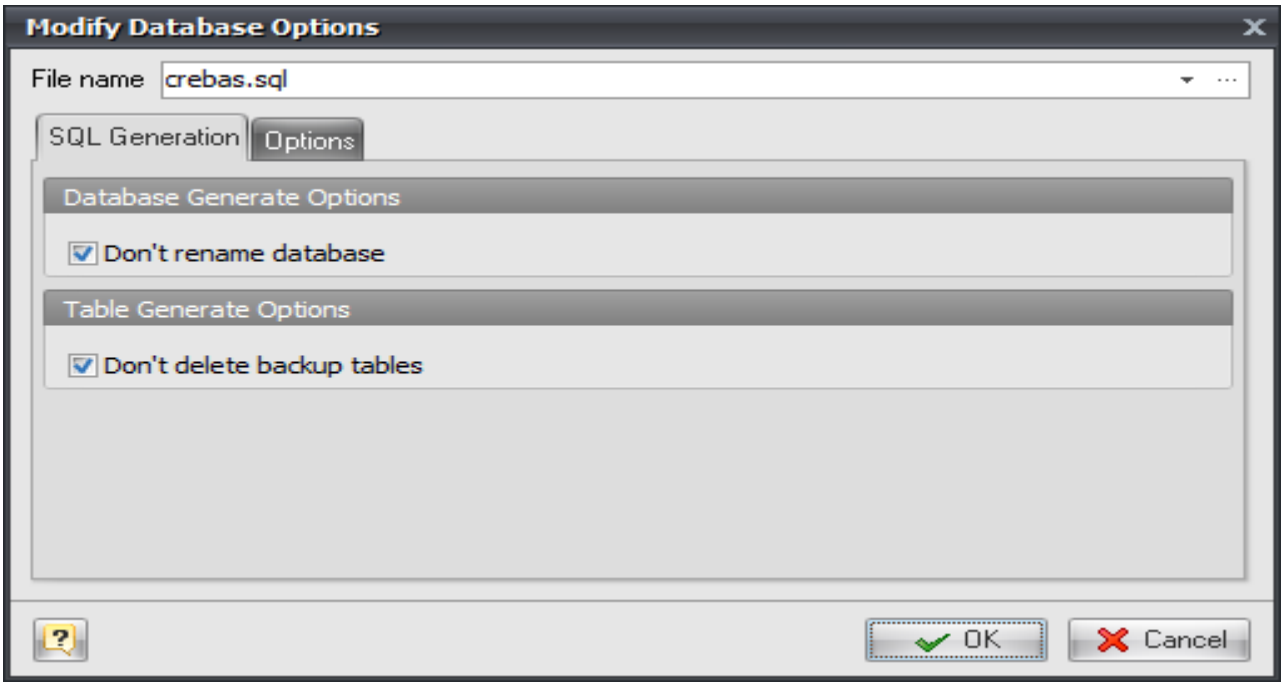


Figure 4.1: sql modification

After connecting to selected database **Database Designer for MySQL** performs Reverse Engineering of this database, compares it with diagram and shows **Compare Result dialog** with all differences found.



You can check or uncheck particular changes you want to generate SQL code for. Then press **Generate SQL** button to open **Modify Database Options** dialog.



There are following options on **SQL Generation** tab:

Don't rename database: Check this option to prevent database renaming even if physical database name differs from one set in diagram.

Don't delete backup tables: If this option is set **Database Designer for MySQL** will not generate DROP TABLE statements for backup table copies (with _tmp_ prefix) after altering tables structure.

Settings on **Options** tab are the same as on **Options** tab of **Database Generation** tool. Press **OK** button to generate SQL script. **SQL Executor** with generated SQL statements for database modifications will appear. You can easily customize statements according to your wants and wishes. And then send them to the database server by clicking on the **Execute SQL** button.

Self-Check 4	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

Part I: Choose the correct answer

1. _____ is generation of SQL script that leads your database to the current state of your diagram.

   A. Database Modification B. Renaming database C. Delete database D. All

2. If some table is going to be modified, Database Designer for MySQL makes a backup copy of that table those backup tables have _____ appended before their names.

   A. *_tmp_* string B. *_bck_* string C. *_res_* string D. *_dba_* string

Part II: write

1. Database Modification tool works:

   a. _____

   b. _____

   c. _____

   d. _____

Note: **Satisfactory rating - 5 points** **Unsatisfactory - below 5 points**

You can ask you teacher for the copy of the correct answers.

Score = _____
Rating: _____

Information sheet 5: Repeating performance testing until results achieved

2.5 Repeating performance testing until results achieved

Repeated tests serve as checks that ensure that everything is working appropriately. If an important part of a test is adjusted while keeping other areas constant, the test as a whole is considered new and must be run. Testing is a critical part of any software development project. Each individual app has a set of assigned test cases and scripts that have been written to specifically evaluate it based on its unique features, functionality and requirements. However, running these tests is not a one-and-done affair; agile testing methodologies have ensured that a program will be evaluated throughout its lifetime, necessitating easy access and repeatability in its test set. If an important part of a test is adjusted while keeping other areas constant, the test as a whole is considered new and must be run. This mutation exercise is used by teams to see if it reveals new behavior and provide better coverage than before. Some teams may also recognize the need to run a test multiple times because one execution may not successfully find a bug. This could occur due to a number of uncontrollable variables, so it's important to run tests a few times to see if any issues emerge. Let's take a look at how to repeat your tests and why doing so is so important to your projects.

Setting up repetitive tests

Teams need to follow a step-by-step strategy for repeating their tests. It's an ongoing method, but one that is essential to ensuring the validity and effectiveness of your test cases throughout a project's lifecycle. First, you'll want to label the type of test it is. There are some test cases that are more suitable for repetition, while others may be run only once. Software Testing Tricks contributor debase Pradhan noted that benchmark tests, sanity tests, performance tests and regression tests are all perfect examples of cases that will need to be run over and over again. Graphic user interface tests and other exploratory evaluations are not compatible with repetition and will likely be run manually by testers.

Once you've labeled each test, you'll want to build out the test cycle based on requirements and configurations. Not all features in a project will need the same tests, so it's important to make these distinctions before you start running evaluations. With agile test management, teams can assign the same test to different cycles, change criteria and make other necessary adjustments to ensure everything is operating effectively.

"Think of a scenario where you need to run certain tests that are important in nature," Pradhan stated. "The tests that verify some very critical functionalities of the application and these tests must be run periodically to make sure those functionalities continue to work without issues."

Repeatable tests take up a lot of time to run. Pradhan also noted that tests become less likely to fail when they run constantly, making it possible to miss out on identifying critical errors. Regularly updating and adding to the test set can mitigate these issues and show the true value of repeatable tests. Industry expert James Bach noted on Satisfies that there are many distinct reasons why repeatable tests will benefit technical and business efforts. Possibly the biggest reason repeatable tests are so useful is that they verify that a problem or behavior has been fixed.

They can also identify if new functionality affected any other parts of the application during the software development process. It's important to understand these types of issues in order to provide a reliable user experience and adjust the deliverable before it's properly released. This reflects well on the team's ability to maintain performance standards, uphold quality expectations and ensure that the project is set up for success. Teams must understand the importance of each test, what it's meant for and how it helps support testing efforts in order to maintain and adjust it properly.

"Be careful not to confuse the importance of a problem with the importance of a test," Bach wrote. "A test might be important for many reasons, even if the problems it detects are not critical ones. Also, don't make the mistake of spending so much effort on one test that looks for an important bug that you neglect other tests that might be just as good or better at finding that kind of problem."

Self-Check 5	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

1. Which types of test is serving as checks that ensure that everything is working appropriately?

   A. Repeated tests B. non-functional test C. Functional test D. All

2. All the following software testing that need to be run over and over again except one.

   A. GUI tests, B. sanity tests, C. performance tests D. regression tests

3. _____ and other exploratory evaluations are not compatible with repetition and will likely be run manually by testers.

   A. GUI tests, B. sanity tests, C. performance tests D. regression tests

Note: **Satisfactory rating - 3 points** **Unsatisfactory - below 3 points**

You can ask you teacher for the copy of the correct answers.

Score = _____
Rating: _____

L #39	Lo#3. Seek client feedback and signoff

Instruction sheet

This learning guide is developed to provide you the necessary information regarding the following content coverage and topics:

- Presenting and providing test results to client for feedback
- Incorporating client change
- Securing client sign-off for testing process

This guide will also assist you to attain the learning outcomes stated in the cover page. Specifically, upon completion of this learning guide, you will be able to:

- Present and providing test results to client for feedback
- Incorporate client change
- Secure client sign-off for testing process

Learning Instructions:

Read the specific objectives of this Learning Guide.

1. Follow the instructions described below.
2. Read the information written in the “Information Sheets”. Try to understand what are being discussed. Ask your trainer for assistance if you have hard time understanding them.
3. Accomplish the “Self-checks” which are placed following all information sheets.
4. Ask from your trainer the key to correction (key answers) or you can request your trainer to correct your work. (You are to get the key answer only after you finished answering the Self-checks).
5. If your performance is satisfactory proceed to the next learning guide,

Information sheet 1: Presenting and providing test results to client for feedback

3.1 Presenting and providing test results to client for feedback

Test report is a document which contains a summary of all test activities and final test results of a testing project. By asking for your customers to provide you with feedback, you're communicating that you value their opinion, and you care about what they have to say. Your customers feel important because you're treating them as such and they feel involved in shaping your project. Constantly improve.

Communication between you and your client is the most important part of your project. It began with your client explaining the project concept. You then created a Concept Paper which explained you understanding of the scope, standards, expected outcomes, intended audience, and explanation of the medium you intend to use. You then completed a set of constraints documents to solidify your understanding of the parameters of the project and this is only for the planning stage.

The only question is that while you have identified how you understand the project to be, does it coincide with your client's vision? The only way to find out is to ask your client. This requires you to meet with your client, share all of your documents and then receive your client's OK to move ahead with your work.

The pivotal document in this verification cycle is the Client Sign-Off. This is the document that your client signs to validate that s/he has read your materials and is in complete agreement with what you are doing. The trick is to create a document that is comprehensive enough to ensure that if your client ever changes her/his mind about what you are doing, the responsibility will rest on her/him rather than on you. This also means that your client will be responsible for paying for the additional resources that are necessary. You are not working on a pay-for-work basis, but it is still important for you to keep your connections with your client documented. This assignment asks you to create a document that, when completed, will verify the date and content of your communications with your client.

For example, if the test report informs that there are many defects remaining in the product, stakeholders can delay the release until all the defects are fixed.

Test report example

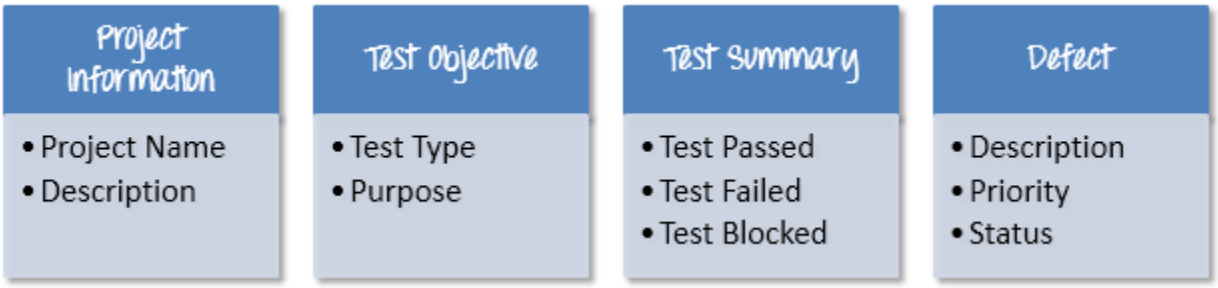
Test Report

Test Cycle: System Test

EXECUTED	PASSED			130	
	FAILED			0	
	(Total) TESTS EXECUTED (PASSED + FAILED)				130
PENDING					0
IN PROGRESS					0
BLOCKED					0
(Sub-Total) TEST PLANNED (PENDING + IN PROGRESS + BLOCKED + TEST EXECUTED)					130

Functions	Description	% TCs Executed	% TCs Passed	TCs pending	Priority	Remarks
New Customer	Check new Customer is created	100%	100%	0	*High*	
Edit Customer	Check Customer can be edited	100%	100%	0	*High*	
New Account	Check New account is added	100%	100%	0	*High*	
Edit Account	Check Account is edit	100%	100%	0	*High*	
Delete Account	Verify Account is delete	100%	100%	0	*High*	
Delete customer	Verify Customer is Deleted	100%	100%	0	*High*	
Mini Statement	Verify Ministatement is generated	100%	100%	0	*High*	
Customized Statement	Check Customized Statement is generated	100%	100%	0	*High*	

A good Test Report contains



Project Information

All information of the project such as the project name, product name, and version should be described in the test report. For example,

Project Overview				
PROJECT BASIC INFORMATION				
Project Name	Guru99 Bank			
Name of product (Product Number)	Banking website www.demo.guru99.com			
Product Description	The banking website			
Project Description	<**Mision of project**> Conduct testing to verify the quality of this website Ensure the website is released without any defects <**Project's output product**> Test Summary Report & Evaluation			
	Project Type	Testing/Verification		
Project Duration	Start date	10/1/2013	End date	10/31/2013

Test Objective: Test Report should include the objective of each round of testing, such as Unit Test, Performance Test, System Test …Etc.

Test Summary: It includes the summary of testing activity in general. Information detailed here includes

- The number of test cases executed
- The numbers of test cases pass
- The numbers of test cases fail
- Pass percentage
- Fail percentage
- Comments

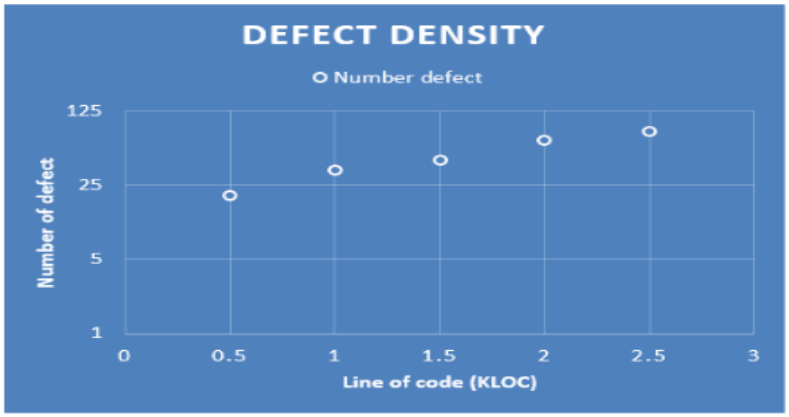
Defect: One of the most important information in Test Report is defect. The report should contain following information.

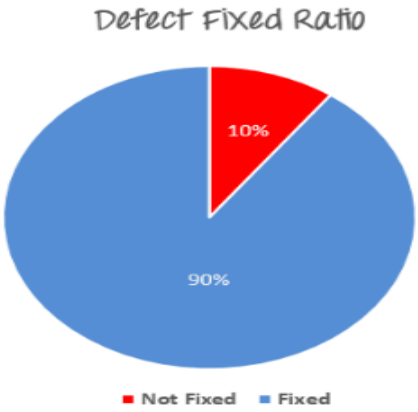
- Total number of bugs
- Status of bugs (open, closed, responding)
- Number of bugs open, resolved, closed
- Breakdown by severity and priority

Like test summary, you can include some simple metrics like Defect density, % of fixed defects. You set the project Defect information like

- Defect density is 20 defects/1000 lines of code average
- 90% defects fixed in total
- The detail of the bugs defect tracker/records

You can represent the data as following graph





Tips to write a good test report

Test report is a **communication** tool between the Test Manager and the stakeholder. Through the test report, the stakeholder can **understand** the project situation, the quality of product and other things.

Test Report	
Project Name	Guru99 Bank
Test Type	Performance Test
Pass	250
Fail	30
Not Executed	30
Total	310

The information of that report is too **abstract**. It does not have any detailed information. The stakeholder who will read it might be slightly **puzzled** when they get it. They might ask or have following sets of questions: -

- Why did they not execute 30 TCs that remains
- What are these failed Test Cases
- Doesn't have any bugs description

To solve that problem, a good Test Report should be detail, clear, standard, and specific

- **Detail**: You should provide a detailed description of the testing activity, show which testing you have performed. Do not put the abstract information into the report, because the reader will not understand what you said.
- **Clear:** All information in the test report should be short and clearly understandable.
- **Standard:** The Test Report should follow the **standard** template. It is easy for stakeholder to review and ensure the **consistency** between test reports in many projects.
- **Specific:** Do not write an essay about the project activity. Describe and summarize the test result specification and focus on the main point.

For example, to correct the above Test Report, the tester should provide more information such as:

- Project information
- Test cycle: (System Test, Integration Test...etc.)
- Which functions have already tested (% TCs executed, % TCs passed or fail…)
- Defect report (Defect description, Priority or status...)

Self-Check 1	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

1. A document that contains a summary of all test activities and final test results of a testing project is called_____.

   A. Test report B. Test type C. Test phase D. Test blocked

2. Project information contains

   A. Project name B. Test type C. Test summary D. All

3. Test summary includes

   A. Test passed B. Test Failed C. Test Blocked D. All

Note: **Satisfactory rating - 5 points** **Unsatisfactory - below 5 points**

You can ask you teacher for the copy of the correct answers.

Score = _____
Rating: _____

Information sheet 2: Incorporating client change

3.2 Incorporating client change

All clients will inevitably change their minds at some point during a project. But, you can set expectations and mitigate these changes so you keep your sanity and save your time.

Information sheet 3: Securing client sign-off f or testing process

3.3 Securing client sign-off for testing process

Introduction to Security Testing

Security testing is a process that is performed with the intention of revealing flaws in security mechanisms and finding the vulnerabilities or weaknesses of software applications. Recent security breaches of systems at retailers like Target and Home Depot, as well as Apple Pay competitor Current, underscore the importance of ensuring that your security testing efforts are up to date.

The prime objective of security testing is to find out how vulnerable a system may be and to determine whether its data and resources are protected from potential intruders. Online transactions have increased rapidly of late making security testing as one of the most critical areas of testing for such web applications. Security testing is more effective in identifying potential vulnerabilities when performed regularly. Normally, security testing has attributes like Authentication, Authorization, Confidentiality, Availability, Integrity, Non-repudiation and Resilience.

System testing, in the current scenario, is a must to identify and address web application security vulnerabilities to avoid any of the following:

- Loss of customer trust.
- Disturbance to your online means of revenue generation/collection.
- Website downtime, time loss and expenditures in recovering from damage (reinstalling services, restoring backups, etc.)

- Cost associated with securing web applications against future attacks.
- Related legal implications and fees for having lax security measures in place.

Classes of Threats

There are different types of classes of threats which can be used to take advantage of security vulnerability.

Privilege Elevation: Privilege elevation is a class of attack where a hacker has an account on a system and uses it to increase his system privileges to a higher level than he/she was not meant to have. If successful, this type of attack can result in a hacker gaining privileges as high as root on a UNIX system. Once a hacker gains super-user privileges, he is able to run code with this level of privilege and the entire system is effectively compromised.

SQL Injection: SQL injection is the most common application layer attack technique used by hackers, in which malicious SQL statements are inserted into an entry field for execution. SQL injection attacks are very critical as an attacker can get critical information from the server database. It is a type of attack which takes the advantage of loopholes present in the implementation of web applications that allows a hacker to hack the system. To check the SQL injection we have to take care of input fields like text boxes, comments, etc. To prevent injections, special characters should be either properly handled or skipped from the input.

Unauthorized Data Access: One of the more popular types of attacks is gaining unauthorized access to data within an application. Data can be accessed on servers or on a network. Unauthorized access includes:

- Unauthorized access to data via data-fetching operations
- Unauthorized access to reusable client authentication information by monitoring the access of others
- Unauthorized access to data by monitoring the access of others

URL Manipulation: URL manipulation is the process of manipulating the website URL query strings & capture the important information by hackers. This happens when the application uses the HTTP GET method to pass information between the client and the server. The information is passed in parameters in the query string. The tester can modify a parameter value in the query string to check if the server accepts it.

Denial of Service: A denial-of-service (DoS) attack is an explicit attempt to make a machine or network resource unavailable to its legitimate users. Applications can also be attacked in ways that render the application, and sometimes the entire machine, unusable.

Data Manipulation: In data manipulation, a hacker changes data used by a website in order to gain some advantage or to embarrass the website's owners. Hackers will often gain access to HTML pages and change them to be satirical or offensive.

Identity Spoofing: Identity spoofing is a technique where a hacker uses the credentials of a legitimate user or device to launch attacks against network hosts, steal data or bypass access controls. Preventing this attack requires IT-infrastructure and network-level mitigations.

Cross-Site Scripting (XSS): Cross-site scripting is a computer security vulnerability found in web applications. XSS enables attackers to inject client-side script into Web pages viewed by other users and trick a user into clicking on that URL. Once executed by the other user's browser, this code could then perform actions such as completely changing the behavior of the website, stealing personal data, or performing actions on behalf of the user. All of the attacks listed above are most critical threat classes but these are not all.

Security Testing Techniques: To prevent all of the above security testing threats/flaws and perform security testing on a web application, it is required to have good knowledge of the HTTP protocol and an understanding of client (browser) server communication through HTTP. Also, basic knowledge of SQL injection and XSS is required. The following techniques will help in performing quality security testing:

Cross Site Scripting (XSS): The tester should additionally check the web application for XSS (Cross site scripting). Any HTML e.g. <HTML> or any script e.g. <SCRIPT> should not be accepted by the application. If it is, the application can be prone to an attack by Cross Site Scripting. Attackers can use this method to execute malicious scripts or URLs on a victim's browser. Using cross-site scripting attackers can use scripts like JavaScript to steal user cookies and information stored in the cookies. Cross Site Scripting Testing can be done for:

- Apostrophe

- Greater-Than Sign
- Less-Than Sign

Ethical Hacking: Ethical hacking means hacking performed by a company or individual to help identify potential threats on a computer or network. An ethical hacker attempts to bypass the system security and search for any vulnerability that could be exploited by malicious hackers aka Black hats. White hats may suggest changes to systems that make them less likely to be penetrated by black hats.

Password Cracking: Password cracking is the most critical part while doing system testing. In order to access the private areas of an application, hackers can use a password cracking tool or can guess a common username/password. Common usernames and passwords are easily available online along with open source password cracking applications. Until a web application enforces a complex password (e.g. a long password with a combination of numbers, letters, and special characters), it is easy to crack the username and password. Another way of cracking the password is if username/password is to target cookies if cookies are stored without encryption.

Penetration Testing: It is an attack on a computer system with the intention of finding security loopholes, potentially gaining access to it, its functionality and data.

Risk Assessment: This is a process of assessing and deciding on the risk involved with the type of loss and possibility of vulnerability occurrence. This is determined within the organization by various interviews, discussions and analysis.

Security Auditing: A security audit is a systematic evaluation of the security of a company's information system by measuring how well it conforms to a set of established criteria.

Security Scanning: This is a program which communicates with a web application through the web front-end in order to identify potential security vulnerabilities in the web application, OS and Networks.

SQL Injection: Entering a single quote (‘) in any textbox should be rejected by the application. Instead, if the tester encounters a database error, it means that the user input is inserted in some query which is then executed by the application. In such a case, the application is vulnerable to SQL injection. SQL injection attacks are very critical as attackers can get vital information from the server database. To check SQL injection entry points into your web application, find out code from your code base where direct MySQL queries are executed on the database by accepting some user inputs.

SQL Injection Testing can be done for:

- Apostrophes
- Brackets
- Commas
- Quotation marks

Vulnerability Scanning: The automated computer program to proactively identify security vulnerabilities of computing systems in a network to determine where a system can be exploited and/or threatened.

Posture Assessment: This describes the overall security posture of an organization; it is a combination of Ethical hacking, Security scanning and Risk Assessment.

URL manipulation through HTTP GET methods: HTTP GET method is used between application client and server to pass on the information. The tester needs to verify if the application is passing vital information in the query string. The information via HTTP is passed in parameters in the query string. To test this, a parameter value can be modified in the query string to check if the server accepts it.

Generally user information is passed through HTTP GET request to the server for either authentication or fetching data. Hackers can manipulate the input of this GET request to the server so that the required information can be gathered or to corrupt the data. Any abrupt behavior of application or web server, in such condition, is the key for a hacker to slip into the application.

Ad hoc Data Testing can also be done as a part of security testing:

- Testing random data which is included in requests.
- Testing random data which is included as parameters.

- Testing encoded random data included as parameters.

Buffer Overflow Testing: Boundary value testing on Lengths of strings e.g. 128 bytes, 256 bytes, 1024 bytes

- Long strings of a single character
- Varied string patterns

Security Testing Approach: We can take the following approach while preparing and planning for Security testing:

- **Security Architecture Study:** The first step is to understand the business requirements, security goals, and objectives in terms of the security compliance of the organization. The test planning should consider all security factors, like the organization might have planned to achieve PCI compliance.
- **Security Architecture Analysis:** Understand and analyze the requirements of the application under test.
- **Classify Security Testing:** Collect all system setup information used for development of Software and Networks like Operating Systems, technology, hardware. Make out the list of Vulnerabilities and Security Risks.
- **Threat Modeling**: Based on above step, prepare Threat profile.
- **Test Planning:** Based on identified Threat, Vulnerabilities and Security Risks prepare test plan to address these issues.
- **Traceability Matrix Preparation**: For each identified Threat, Vulnerabilities and Security Risks prepare Traceability Matrix.
- **Security Testing Tool identification:** All security testing cannot be executed manually, so identify the tool to execute all security test cases faster & more reliably.
- **Test Case Preparation:** Prepare the Security tests case document.
- **Test Case Execution**: Perform the Security Test cases execution and retest the defect fixes. Execute the Regression Test cases.
- **Reports**: Prepare detailed report of Security Testing which contains Vulnerabilities and Threats contained, detailing risks, and still open issues etc.

A project sign-off sheet is essential to formally close a project or acknowledge delivery of a key deliverable. Creating a client sign-off sheet document should be written after all necessary test finished, so that it relieves you of liability if questions begin to surface in the future. Test Report is a document which contains a summary of all test activities and final test results of a testing project. Test report is an assessment of how well the Testing is performed. Based on the test report, stakeholders can evaluate the quality of the tested product and make a decision on the software release.

Self-Check 3	Written Test

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page:

1. _____is a process that is performed with the intention of revealing flaws in security mechanisms and finding the vulnerabilities or weaknesses of software applications.

   A. Security testing B. Authentication C. Authorization D. Confidentiality

2. All are security testing attributes except one.

   A. Authentication, B. Authorization, C. Confidentiality, D. repudiation

3. _____ is performed by a company or individual to help identify potential threats on a computer or network.

   A. Password hacking B. Ethical hacking

   C. Security Scanning D. Security Scanning

4. Cross Site Scripting Testing cannot be done for

   A. Apostrophe B. Greater-Than Sign C. Less-Than Sign D. Equal sign

5. In order to access the private areas of an application, hackers can use a _____tool or can guess a common username/password.

   A. Security Scanning B. Ethical hacking C. Password hacking D. Security Scanning

Note: **Satisfactory rating - 5 points** **Unsatisfactory - below 5 points**

You can ask you teacher for the copy of the correct answers.

Score = _____
Rating: _____

AKNOWLEDGEMENT

We wish to extend thanks and appreciation to the many representatives of TVET instructors who donated their time and expertise to the development of this TTLM.

We would like also to express our appreciation to the TVET instructors and Oromia TVET Bureaus, and Federal Technical and Vocational Education and Training Agency (FTVET) who made the development of this curriculum with required standards and quality possible.

This TTLM developed on December 2020 at Bishoftu Ethiopia.

The trainers who developed the TTLM

No	Name	Organization	Educational background & Level	Profession /Job title	Email address	Mobile
1.	Ayansa Ergiba	Ambo TVETC	Msc.	Instructor	aergiba@gmail.com	0917851343
2.	Ejigu Birhanu	Nekemte TVETC	Msc.	Instructor	ejigubiranu2011@gmail.com	0906566892
3.	Endale Lema	Adama PTC	Msc.	Instructor	endhiywet@gmai.com	0913292212
4.	Keresa Gadisa	Nekemte TVETC	Msc.	Instructor	keresag2010@gmail.com	0920420664
5.	Meseret Tezera	Atilet Kenanisa PTC	Msc.	Instructor	YafetMes@gmail.com	0911751285

REFENCES

Fundamentals of |Database System 4th Edition Ramez Elmasri

Department of Computer Science Engineering University of Texas at Arlington Shamkant B. N avathe

College of Computing Georgia Institute of Technology

Beginning SQL Server 2005 For Developers (2006) Robin Dewson,

Beginning SQL (2005) Paul Wilton and John W. Colby

Database Design for Mere Mortals™, Second Edition

Introduction to SQL: Mastering the Relational Database Language, Fourth Edition/20th Anniversary Edition By Rick F. van der Lans

Answer Key

LO #1- Undertake database management system modeling	
Self-Check 1	**Written Test**

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page (each 1 point):

1. B 2. A 3. D 4. C 5. A

Self-Check 2	**Written Test**

1. A 2. C 3. A 4. D 5.D

Self-Check 3	**Written Test**

1. C 2. A 3. C 4.A

LO #2- Test database performance	
Self-Check 1	**Written Test**

Directions: Answer all the questions listed below. Use the Answer sheet provided in the next page (each 1 point):

1. D 2. A 3. D 4. C 5. B

Self-Check 2	**Written Test**

1. A 2. A 3.B 4.D

LO #3- Seek client feedback and signoff	
Self-Check 1	**Written Test**

Directions: Answer all the questions listed below. Each 1 point