a) Features of Good Database Design

Characteristics of Good Database

- The database should be strong enough to store all the relevant data and requirements.
- Should be able to relate the tables in the database by means of a relation,

  For example, an employee works for a department so that employee is related to a particular department. We should be able to define such a relationship between any two entities in the database.
- Multiple users should be able to access the same database, without affecting the other user .

  For example, several teachers can work on a database to update learners' marks at the same time. Teachers should also be allowed to update the marks for their subjects, without modifying other subject marks.
- A single database provides different views to different users; it supports multiple views to the user, depending on his role. In a school database.

  For example, teachers can see the breakdown of learners' marks; however, parents are only able to see only their child's report – thus the parents' access would be read only. At the same time, teachers will have access to all the learners' information and assessment details with modification rights. All this is able to happen in the same database.

- **Data integrity** refers to how accurate and consistent the data in a database is. Databases with lots of missing information and incorrect information is said to have low data integrity.

- **Data independence** refers to the separation between data and the application (or applications) in which it is being used. This allows you to update the data in your application (such as fixing a spelling mistake) without having to recompile the entire application.

- **Data Redundancy** refers to having the exact same data at different places in the database. Data redundancy Increases the size of the database, creates Integrity problems, decreases efficiency and leads to anomalies. Data should be stored so that It Is not repeated In multiple tables.

- **Data security** refers to how well the data in the database is protected from crashes, hacks and accidental deletion.

- **Data maintenance** refers to monthly, daily or hourly tasks that are run to fix errors within a database and prevent anomalies from occurring. Database maintenance not only fixes errors, but it also detects potential errors and prevents future errors from occurring.

There are also many people involved with organising a well-run database. These are:

- The **developers**, who design and develop the database to suite the needs of an enterprise
- The **administrator**, who:
- Checks the database for its usages
- Who is checking it
- Provides access to other uses
- Provides any other maintenance work required to keep the database up to date
- The **end user**, who uses the database, for example, teachers or parents.

b) Atomic Domains and first Normal Form

Atomic Domain:

Edgar F. Codd's definition of 1NF makes reference to the concept of 'atomicity'. Codd states that the "Values in the domains on which each relation is defined are required to be atomic with respect to the DBMS." Codd defines an atomic value as one that "cannot be decomposed into smaller pieces by the DBMS (excluding certain special functions)" meaning a column should not be divided into parts with more than one kind of data in it such that what one part means to the DBMS depends on another part of the same column.

Hugh Darwen and Chris Date have suggested that Codd's concept of an "atomic value" is ambiguous, and that this ambiguity has led to widespread confusion about how 1NF should be understood. In particular, the notion of a "value that cannot be decomposed" is problematic, as it would seem to imply that few, if any, data types are atomic:

•A character string would seem not to be atomic, as the RDBMS typically provides operators to decompose it into substrings.

•A fixed-point number would seem not to be atomic, as the RDBMS typically provides operators to decompose it into integer and fractional components.

•An ISBN would seem not to be atomic, as it includes language and publisher identifier.

Date suggests that "the notion of atomicity has no absolute meaning":a value may be considered atomic for some purposes, but may be considered an assemblage of more basic elements for other purposes. If this position is accepted, 1NF cannot be defined with reference to atomicity. Columns of any conceivable data type (from string types and numeric types to array types and table types) are then acceptable in a 1NF table—although perhaps not always desirable; for example, it may be more desirable to separate a Customer Name column into two separate columns as First Name, Surname.

If a table has data redundancy and is not properly normalized, then it will be difficult to handle and update the database, without facing data loss. It will also eat up extra memory space and Insertion, Update and Deletion *Anomalies are very frequent if database is not normalized*.

c) Decomposition using Functional Dependencies

Decomposition

A functional **decomposition** is the process of breaking down the functions of an organization into progressively greater (finer and finer) levels of detail.
In decomposition, one function is described in greater detail by a set of other supporting functions.
The decomposition of a relation scheme R consists of replacing the relation schema by two or more relation schemas that each contain a subset of the attributes of R and together include all attributes in R.
Decomposition helps in eliminating some of the problems of bad design such as redundancy, inconsistencies and anomalies.
There are two types of decomposition :

1. Lossy Decomposition
2. Lossless Join Decomposition

Lossy Decomposition:

"The decomposition of relation R into R1 and R2 is **lossy** when the join of R1 and R2 does not yield the same relation as in R."
One of the disadvantages of decomposition into two or more relational schemes (or tables) is that some information is lost during retrieval of original relation or table.
Consider that we have table STUDENT with three attribute roll_no , sname and department.

STUDENT:

Roll_no	Sname	Dept
111	Ankit	COMPUTER
222	Ankit	ELECTRICAL

This relation is decomposed into two relation no_name and name_dept :

RollNo_Name

Roll_no	Sname
111	Ankit
222	Ankit

Name_Dept

Sname	Dept
Ankit	COMPUTER
Ankit	ELECTRICAL

In lossy decomposition ,spurious tuples are generated when a natural join is applied to the relations in the decomposition.

stu_joined :

Roll_no	Sname	Dept
111	Ankit	COMPUTER
111	Ankit	ELECTRICAL
222	Ankit	COMPUTER
222	Ankit	ELECTRICAL

The above decomposition is a bad decomposition or Lossy decomposition.

Lossless Join Decomposition:

"The decompositio of relation R into R1 and R2 is **lossless** when the join of R1 and R2 yield the same relation as in R."

A relational table is decomposed (or factored) into two or more smaller tables, in such a way that the designer can capture the precise content of the original table by joining the decomposed parts. This is called lossless-join (or non-additive join) decomposition.

This is also refferd as non-additive decomposition.

The lossless-join decomposition is always defined with respect to a specific set F of dependencies.

Consider that we have table STUDENT with three attribute roll_no , sname and department.

STUDENT :

Roll_no	Sname	Dept
111	Ankit	COMPUTER
222	Ankit	ELECTRICAL

This relation is decomposed into two relation Stu_name and Stu_dept :

Stu_name:

Roll_no	Sname
111	Ankit
222	Ankit

Stu_dept :

Roll_no	Dept
111	COMPUTER
222	ELECTRICAL

Now ,when these two relations are joined on the comman column 'roll_no' ,the resultant relation will look like stu_joined.

stu_joined :

Roll_no	Sname	Dept
111	Ankit	COMPUTER
222	Ankit	ELECTRICAL

In lossless decomposition, no any spurious tuples are generated when a natural joined is applied to the relations in the decomposition.

d) Functional Dependency Theories

What is a Functional Dependency?

Functional Dependency (FD) determines the relation of one attribute to another attribute in a database management system (DBMS) system. Functional dependency helps you to maintain the quality of data in the database. A functional dependency is denoted by an arrow →. The functional dependency of X on Y is represented by X → Y. Functional Dependency plays a vital role to find the difference between good and bad database design.

Example:

In this example, if we know the value of Employee number, we can obtain Employee Name, city, salary, etc. By this, we can say that the city, Employee Name, and salary are functionally depended on Employee number.

Employee number	Employee Name	Salary	City
1	Ankit Shah	1000	Mumbai
2	Sagar Parab	1000	Pune
3	Ravindra Gaikwad	1000	Nashik

Key terms

Here, are some key terms for functional dependency:

Key Terms	Description
Axiom	Axioms is a set of inference rules used to infer all the functional dependencies on a relational database.
Decomposition	It is a rule that suggests if you have a table that appears to contain two entities which are determined by the same primary key then you should consider breaking them up into two different tables.
Dependent	It is displayed on the right side of the functional dependency diagram.
Determinant	It is displayed on the left side of the functional dependency Diagram.
Union	It suggests that if two tables are separate, and the PK is the same, you should consider putting them. together

Types of Functional Dependencies

- **Multivalued dependency:**
- **Trivial functional dependency**:
- **Non-trivial functional dependency**:
- **Transitive dependency:**

Let us discuss all the dependencies with examples

Multivalued dependency:

Car_model	Maf_year	Color
H001	2017	Metallic
H001	2017	Green
H005	2018	Metallic
H005	2018	Blue
H010	2015	Metallic
H033	2012	Gray

- In this example, maf_year and color are independent of each other but dependent on car_model. In this example, these two columns are said to be multivalue dependent on car_model.
- This dependence can be represented like this:
- car_model -> maf_year
- car_model-> colour

Trivial Functional dependency:

The Trivial dependency is a set of attributes which are called a trivial if the set of attributes are included in that attribute.

So, X -> Y is a trivial functional dependency if Y is a subset of X.

For example:

Emp_id	Emp_name
AS555	Ankit
AS811	Sagar
AS999	Ravindra

Consider this table with two columns Emp_id and Emp_name.

{Emp_id, Emp_name} -> Emp_id is a trivial functional dependency as Emp_id is a subset of {Emp_id,Emp_name}.

Non trivial functional dependency in DBMS

Functional dependency which also known as a nontrivial dependency occurs when A->B holds true where B is not a subset of A. In a relationship, if attribute B is not a subset of attribute A, then it is considered as a non-trivial dependency.

Company	CEO	Age
Microsoft	Satya Nadella	51
Google	Sundar Pichai	46
Apple	Tim Cook	57

Example:

(Company} -> {CEO} (if we know the Company, we knows the CEO name)

But CEO is not a subset of Company, and hence it's non-trivial functional dependency.

Transitive dependency:

A transitive is a type of functional dependency which happens when t is indirectly formed by two functional dependencies.

Example:

Company	CEO	Age
Microsoft	Satya Nadella	51
Google	Sundar Pichai	46
Alibaba	Jack Ma	54

{Company} -> {CEO} (if we know the compay, we know its CEO's name)

{CEO } -> {Age} If we know the CEO, we know the Age

Therefore according to the rule of rule of transitive dependency:

{ Company} -> {Age} should hold, that makes sense because if we know the company name, we can know his age.

Note: You need to remember that transitive dependency can only occur in a relation of three or more attributes.

Advantages of Functional Dependency

- Functional Dependency avoids data redundancy. Therefore same data do not repeat at multiple locations in that database
- It helps you to maintain the quality of data in the database
- It helps you to defined meanings and constraints of databases
- It helps you to identify bad designs
- It helps you to find the facts regarding the database design

Summary

- Functional Dependency is when one attribute determines another attribute in a DBMS system.
- Axiom, Decomposition, Dependent, Determinant, Union are key terms for functional dependency
- Four types of functional dependency are 1) Multivalued 2) Trivial 3) Non-trivial 4) Transitive
- Multivalued dependency occurs in the situation where there are multiple independent multivalued attributes in a single table
- The Trivial dependency occurs when a set of attributes which are called a trivial if the set of attributes are included in that attribute
- Nontrivial dependency occurs when A->B holds true where B is not a subset of A
- A transitive is a type of functional dependency which happens when it is indirectly formed by two functional dependencies
- Normalization is a method of organizing the data in the database which helps you to avoid data redundancy

e) Normalization

What is Normalization?

Normalization is a method of organizing the data in the database which helps you to avoid data redundancy, insertion, update & deletion anomaly. It is a process of analysing the relation schemas based on their different functional dependencies and primary key.

Normalization is inherent to relational database theory. It may have the effect of duplicating the same data within the database which may result in the creation of additional tables.

There are various level of normalization.

1. First Normal Form (1NF)
2. Second Normal Form (2NF)
3. Third Normal Form (3NF)
4. Boyce-Codd Normal Form (BCNF)
5. Forth Normal Form (4NF)
6. Fifth Normal Form (5NF)

We will study types of normalization and how to bring table in normalized form after discussion of few points that are necessary to be aware of before proceeding with normalization.

f) Decomposition

- When a relation in the relational model is not in appropriate normal form then the decomposition of a relation is required.
- In a database, it breaks the table into multiple tables.
- If the relation has no proper decomposition, then it may lead to problems like loss of information.
- Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies, and redundancy

Properties of Relational Decomposition

When a relation in the relational model is not appropriate normal form then the decomposition of a relation is required. In a database, breaking down the table into multiple tables termed as decomposition. The properties of a relational decomposition are listed below :

1. **Attribute Preservation:**

   Using functional dependencies the algorithms decompose the universal relation schema R in a set of relation schemas D = { R1, R2, ….. Rn } relational database schema, where 'D' is called the Decomposition of R.

   The attributes in R will appear in at least one relation schema Ri in the decomposition, i.e., no attribute is lost. This is called the *Attribute Preservation* condition of decomposition.

2. **Dependency Preservation:**
   If each functional dependency X->Y specified in F appears directly in one of the relation schemas Ri in the decomposition D or could be inferred from the dependencies that appear in some Ri. This is the *Dependency Preservation*.

   If a decomposition is not dependency preserving some dependency is lost in decomposition. To check this condition, take the JOIN of 2 or more relations in the decomposition.

   For example:

   ```
   R = (A, B, C)

   F = {A ->B, B->C}

   Key = {A}


   R is not in BCNF.

   Decomposition R1 = (A, B), R2 = (B, C)
   ```

R1 and R2 are in BCNF, Lossless-join decomposition, Dependency preserving.
Each Functional Dependency specified in F either appears directly in one of the relations in the decomposition.
It is not necessary that all dependencies from the relation R appear in some relation Ri.
It is sufficient that the union of the dependencies on all the relations Ri be equivalent to the dependencies on R.

3. **Non Additive Join Property:**
   Another property of decomposition is that D should possess is the *Non Additive Join Property*, which ensures that no spurious tuples are generated when a NATURAL JOIN operation is applied to the relations resulting from the decomposition.

4. **No redundancy:**
   Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies, and redundancy.If the relation has no proper decomposition, then it may lead to problems like loss of information.

5. **Lossless Join:**
   Lossless join property is a feature of decomposition supported by normalization. It is the ability to ensure that any instance of the original relation can be identified from corresponding instances in the smaller relations.
   For example:
   R : relation, F : set of functional dependencies on R,
   X, Y : decomposition of R,
   A decomposition {R1, R2, …, Rn} of a relation R is called a lossless decomposition for R if the natural join of R1, R2, …, Rn produces exactly the relation R.

   A decomposition is lossless if we can recover:
   R(A, B, C) -> Decompose -> R1(A, B) R2(A, C) -> Recover -> R'(A, B, C)
   Thus, R' = R
   Decomposition is lossless if:
   X intersection Y -> X, that is: all attributes common to both X and Y functionally determine ALL the attributes in X.
   X intersection Y -> Y, that is: all attributes common to both X and Y functionally determine ALL the attributes in Y
   If X intersection Y forms a superkey of either X or Y, the decomposition of R is a lossless decomposition.

g) Normal Form Discussion : First normal to fifth normal forms

Let us try to understand all the normalization forms with an example

Assume, a video library maintains a database of movies rented out. Without any normalization, all information is stored in one table as shown below.

Salutation	Full Name	Physical Address	Movies Rented
Ms	Ram Singh	Ayodhya	Baghban, Gangs Of Waseypur
Mr	Pawan Kumar	Bihar	Prince,Harry Potter
Mr	Ravi Teja	South	Drishyam, VIP
Mr	Norrulla Shafillula	Delhi	Airlift

First Normal Form (1NF)

If a relation contain composite or multi-valued attribute, it violates first normal form, or a relation is in first normal form if it does not contain any **composite** or **multi-valued attribute**. A relation is in first normal form if every attribute in that relation is singled valued attribute..

A table is in 1 NF iff:

1. There are only Single Valued Attributes.
2. Attribute Domain does not change.
3. There is a Unique name for every Attribute/Column.
4. The order in which data is stored, does not matter.

Here you see **Movies Rented column has multiple values.**

Now let's move into 1st Normal Forms:

Salutation	Full Name	Physical Address	Movies Rented
Ms	Ram Singh	Ayodhya	Baghban
Ms	Ram Singh	Ayodhya	Gangs Of Waseypur
Mr	Pawan Kumar	Bihar	Prince
Mr	Pawan Kumar	Bihar	Harry Potter
Mr	Ravi Teja	South	Drishyam
Mr	Ravi Teja	South	VIP
Mr	Norrulla Shafillula	Delhi	Airlift

2NF (Second Normal Form) Rules

- Rule 1- Be in 1NF
- Rule 2- Single Column Primary Key

It is clear that we can't move forward to make our simple database in 2nd Normalization form unless we partition the table above.

Membership_ID	Salutation	Full Name	Physical Address
1	Ms	Ram Singh	Ayodhya
2	Mr	Pawan Kumar	Bihar
3	Mr	Ravi Teja	South
4	Mr	Noorulla Shafillula	Delhi

Membership_ID	Movie Rented
1	Baghban
1	Gangs Of Waseypur
2	Prince
2	Harry Potter
3	Drishyam
3	VIP
4	Airlift

We have divided our 1NF table into two tables viz. Table 1 and Table2. Table 1 contains member information. Table 2 contains information on movies rented.

We have introduced a new column called Membership_id which is the primary key for table 1. Records can be uniquely identified in Table 1 using membership id

Database - Foreign Key

In Table 2, Membership_ID is the Foreign Key

Foreign Key references the primary key of another Table! It helps connect your Tables

- A foreign key can have a different name from its primary key
- It ensures rows in one table have corresponding rows in another
- Unlike the Primary key, they do not have to be unique. Most often they aren't
- Foreign keys can be null even though primary keys can not

Why do you need a foreign key?

Suppose, a novice inserts a record in Table B such as

Membership_ID	Movie Rented
110	Dil Bechara

But the entyr of Membership ID 110 is not present in Table 1, this insert will fail.. reason.. Referential Integrity

You will only be able to insert values into your foreign key that exist in the unique key in the parent table. This helps in referential integrity.

The above problem can be overcome by declaring membership id from Table2 as foreign key of membership id from Table1

Now, if somebody tries to insert a value in the membership id field that does not exist in the parent table, an error will be shown!

3NF (Third Normal Form) Rules

- Rule 1- Be in 2NF
- Rule 2- Has no transitive functional dependencies

What are transitive functional dependencies?

A transitive functional dependency is when changing a non-key column, might cause any of the other non-key columns to change

Consider the table 1. Changing the non-key column Full Name may change Salutation.

To move our 2NF table into 3NF, we again need to again divide our table.

Membership ID	Salutation_Id	Full Name	Physical Address
1	3	Ram Singh	Ayodhya
2	1	Pawan Kumar	Bihar
3	1	Ravi Teja	South
4	1	Noorulla Shafillula	Delhi

Membership ID	Movie Rented
1	Baghban
1	Gangs Of Waseypur
2	Prince
2	Harry Potter
3	Drishyam
3	VIP
4	Airlift

Salutation_Id	Salutation
1	Mr
2	Mrs
3	Ms
4	Dr

Advantage of removing Transitive Dependency

The advantage of removing transitive dependency is,

- Amount of data duplication is reduced.
- Data integrity achieved.

h) BCNF

Boyce-Codd Normal Form or BCNF is an extension to the third normal form, and is also known as 3.5 Normal Form.

Rules for BCNF

For a table to satisfy the Boyce-Codd Normal Form, it should satisfy the following two conditions:

1. It should be in the **Third Normal Form**.
2. And, for any dependency A → B, A should be a **super key**.

The second point sounds a bit tricky, right? In simple words, it means, that for a dependency A → B, A cannot be a **non-prime attribute**, if B is a **prime attribute**.

EMP_ID	EMP_COUNTRY	EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developin g	D283	549

In the above table Functional dependencies are as follows:

1. EMP_ID → EMP_COUNTRY
2. EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate key: {EMP-ID, EMP-DEPT}

The table is not in BCNF because neither EMP_DEPT nor EMP_ID alone are keys.

To convert the given table into BCNF, we decompose it into three tables:

EMP_COUNTRY table:

EMP_ID	EMP_COUNTRY
264	India
264	India

EMP_DEPT table:

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

EMP_DEPT_MAPPING table:

EMP_ID	EMP_DEPT
D394	283
D394	300
D283	232
D283	549

Functional dependencies:

1. EMP_ID → EMP_COUNTRY
2. EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate keys:

For the first table: EMP_ID
For the second table: EMP_DEPT
For the third table: {EMP_ID, EMP_DEPT}

Now, this is in BCNF because left side part of both the functional dependencies is a key.

Fourth normal form (4NF)

- A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
- For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

STUDENT

TU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

In the STUDENT relation, a student with STU_ID, **21** contains two courses, **Computer** and **Math** and two hobbies, **Dancing** and **Singing**. So there is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data.

So to make the above table into 4NF, we can decompose it into two tables:

STUDENT_COURSE

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

STUDENT_HOBBY

STU_ID	HOBBY
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

Now the data is in 4th Normal Form

Fifth normal form (5NF)

- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-join normal form (PJ/NF).

Example

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	Hrutuja	Semester 1
Math	Hrutuja	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.

Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.

So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

P1

SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

P2

SUBJECT	LECTURER
Computer	Anshika
Computer	Hrutuja
Math	Hrutuja
Math	Akash
Chemistry	Praveen

P3

SEMSTER	LECTURER
Semester 1	Anshika
Semester 1	Hrutuja
Semester 1	Hrutuja
Semester 2	Akash
Semester 1	Praveen

i) Pitfall in Relational- database design

Creating an effective design for a relational database is a key element in building a reliable system. There is no one "correct" relational database design for any particular project, and developers must make choices to create a design that will work efficiently. There are a few common design pitfalls that can harm a database system. Watching out for these errors at the design stage can help to avoid problems later on.

Careless Naming Practices

Choosing names is an aspect of database design that is often neglected but can have a considerable impact on usability and future development. To avoid this, both table and column names should be chosen to be meaningful and to conform to the established conventions, ensuring that consistency is maintained throughout a system. A number of conventions can be used in relational database names, including the following two examples for a record storing a client name: "client_name" and "clientName."

Lack of Documentation

Creating documentation for a relational database can be a vital step in safeguarding future development. There are different levels of documentation that can be created for databases, and some database management systems are able to generate the documentation automatically. For projects where formal documentation is not considered necessary, simply including comments within the SQL code can be helpful.

Failure to Normalize

Normalization is a technique for analyzing, and improving on, an initial database design. A variety of techniques are involved, including identifying features of a database design that may compromise data integrity, for example items of data that are stored in more than one place. Normalization identifies anomalies in a database design, and can preempt design features that will cause problems when data is queried, inserted or updated.

Lack of Testing

Failure to test a database design with a sample of real, or realistic, data can cause serious problems in a database system. Generally, relational database design is started from an abstract level, using modeling techniques to arrive at a design. The drawback to this process is that the design sometimes will not relate accurately to the actual data, which is why testing is so important.

Failure to Exploit SQL Facilities

SQL has many capabilities that can improve the usability and success of a database system. Facilities such as stored procedures and integrity checks are often not used in cases where they could greatly enhance the stability of a system. Developers often choose not to carry out these processes during the design stages of a project as they are not a necessity, but they can help to avoid problems at a later stage.