

**Database Design & Development**

**10204282  
A/618/7400**

**Section (3)**

**Developing a Database System**

**Submitted to**

Dr. Raneem Qaddoura

**Submitted on**

June 20th, 2023

**Submitted by**

Marwan Tareq Shafiq Al Farah

**Student ID**

21110011

**Spring 2022 - 2023**

**Table of Content**

[***Report*** 4](#_Toc138103746)

[**Introduction** 4](#_Toc138103747)

[**Physical Schema** 4](#_Toc138103748)

[**Database Development** 5](#_Toc138103749)

[ **Database Overview** 5](#_Toc138103750)

[ **Security** 8](#_Toc138103751)

[ **User Interface** 10](#_Toc138103752)

[**Maintenance** 13](#_Toc138103754)

[ **Database Recovery & Backups** 13](#_Toc138103755)

[ **Database Maintenance in General** 16](#_Toc138103756)

[**Testing** 18](#_Toc138103757)

[ **Data Validation** 18](#_Toc138103758)

[ **Output Validation** 21](#_Toc138103759)

[ **Security Validation** 26](#_Toc138103760)

[ **GUI Validation** 32](#_Toc138103761)

[ **Assess whether Meaningful Data Has Been Extracted** 34](#_Toc138103762)

[ **Assess the Effectiveness of the Testing** 36](#_Toc138103763)

[**Evaluation of Database Solution** 41](#_Toc138103764)

[ **Effectiveness of the Database Solution Based on User and System Requirements** 41](#_Toc138103765)

[ **Suggested Improvements** 43](#_Toc138103766)

[ **Evaluation Based on Improvements Needed** 44](#_Toc138103767)

[***References*** 47](#_Toc138103768)

# ***Report***

**Introduction:**

This technical document outlines in detail the relational database system “IHDB” developed for Ibn Al Haitham Hospital in Amman. The IHDB system includes many aspects of hospital operations, such as patients, doctors, surgeries, operating rooms, and many others. It aims to improve hospital workflows, improve medical record management, ensure secure handling of sensitive information, and provide a dependable platform for generating relevant management reports.

The document will outline the IHDB system’s physical schema, development process, security protocols, and user interface, with a focus on its maintenance and testing strategies. We’ll go over the system’s effectiveness in terms of user and system requirements, potential improvements, and its adaptability to future enhancements. Finally, this document serves as a comprehensive guide to effectively understanding, implementing, and maintaining the IHDB system.

**Physical Schema:**

This document’s Physical Schema explains the database’s structural design, demonstrating how tables and entities are interconnected. It displays an illustration of the data model, which includes tables, relationships, and the various types of data stored in the system. Users can comprehend the relational model and data flow by understanding the physical schema, which is essential for data management, querying, and maintenance in the IHDB system.

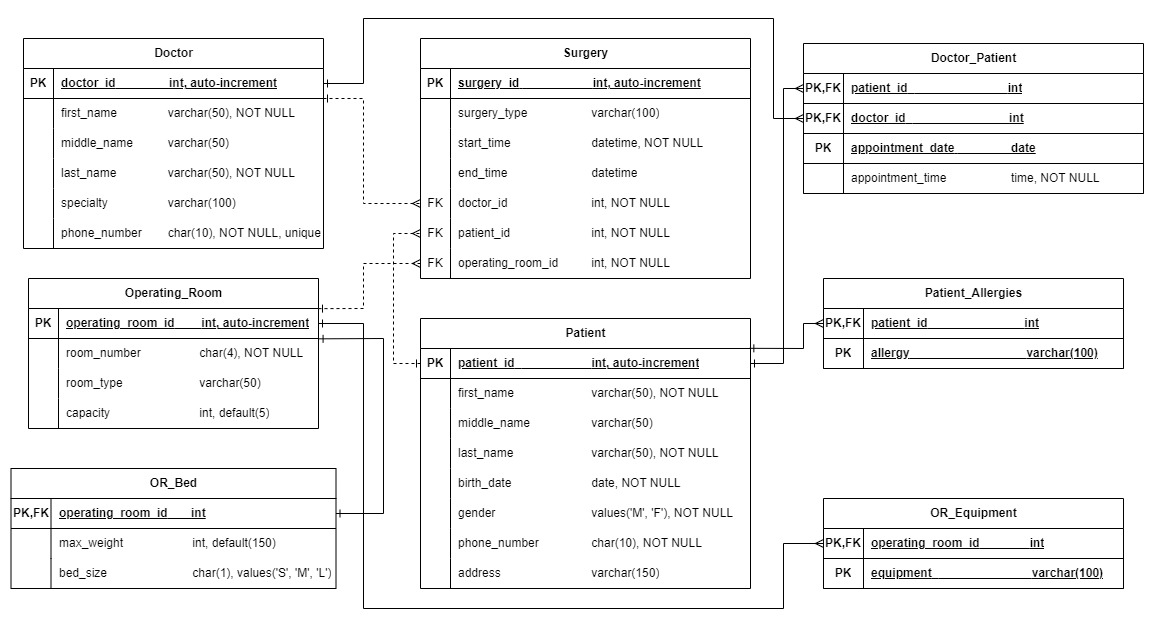


Figure 1 – The Database’s Physical Design

**Database Development:**

IHDB was created with the goal of preserving patient information, doctor details, operating room specifications, surgery records, while keeping the relationship between these entities in mind. Each component of the database contributes to its overall functionality, ensuring that it effectively and efficiently meets the needs of the healthcare system.

* **Database Overview:**

The IHDB database is made up of several tables, views, and procedures, each of which serves a specific purpose. The tables contain the system’s core data, such as information about patients, doctors, surgeries, operating rooms, and so on. Views are simplified views or subsets of the database that are typically designed for specific user roles or operations. The procedures are a collection of SQL statements that perform a specific task in the tables, such as selecting, inserting, updating, or deleting records.

* **Tables:**

The tables in the IHDB database serve as the system’s backbone, storing all critical information. Each table contains specific fields that hold the data, such as Patient, Doctor, Operating\_Room, Surgery, OR\_Bed, Patient\_Allergies, OR\_Equipment, and Doctor\_Patient. These tables are linked to one another using foreign keys, allowing the system to link data for various operations and queries.

|  |  |
| --- | --- |
| Table | Description |
| Patient | Contains patient information such as ID, name, birthdate, gender, phone number, and address. |
| Doctor | Includes doctor information such as ID, name, specialty, and phone number. |
| Operating\_Room | Keeps track of each operating room’s ID, room number, room type, and capacity. |
| Surgery | Contains information about surgeries performed, such as the surgery ID, type, start and end time, and the associated doctor, patient, and operating room IDs. |
| OR\_Bed | Keeps information about each operating room’s bed, such as the operating room ID, bed size, and maximum weight. |
| Patient\_Allergies | It contains information about each patient’s allergies. |
| OR\_Equipment | Keeps track of the equipment available in each operating room. |
| Doctor\_Patient | Keeps track of doctor-patient appointments, including the date and time of each appointment. |

* **Views:**

Views in the IHDB database provide a handy way to access and show organized and relevant data from the underlying tables. The following perspectives have been developed:

|  |  |
| --- | --- |
| View | Description |
| patient\_summary\_view | This view displays a summary of patients’ information, including their ID, name, birthdate, gender, phone number, address, age (estimated based on the birthdate), and a list of allergies (if any) by joining the Patient and Patient\_Allergy tables. |
| patient\_surgery\_details\_view | This view displays information on surgeries linked with patients, such as the patient’s ID, surgery ID, surgery type, start time, finish time, and the first and last names of the performing doctor, as well as the number and type of the operating room, by joining the Surgery, Doctor, and Operating\_Room tables. |
| doctor\_surgery\_details\_view | This view provides information on surgeries linked with doctors, such as the doctor’s ID, surgery ID, surgery type, start time, finish time, the patient’s first and last names, and the number and type of the operating room, by Joining the Surgery, Patient, and Operating\_Room tables. |
| patient\_doctor\_information\_view | This view displays information about doctors who are linked with patients, such as the patient’s ID, the doctor’s ID, the doctor’s first name, middle name, last name, specialization, and phone number, by joining the Doctor and Doctor\_Patient tables. |
| doctor\_surgery\_count\_view | This view calculates the number of operations conducted by each doctor and displays the doctor’s ID, first name, last name, and the number of surgeries performed, by joining the Doctor and Surgery tables. |
| patient\_demographics\_view | This view displays patient demographic information, such as gender distribution, patient count, and average age based on birthday. |
| operating\_room\_equipment\_summary\_view | This view displays an overview of the equipment in each operating room, including the operating room ID, room number, room type, the number of available equipment items, and a concatenated list of the equipment items, by left joining Operating\_Room and OR\_Equipment tables. |

* **Procedures:**

In the IHDB database, stored procedures encapsulate groups of SQL statements that may be performed repeatedly with a single command. The stored processes listed below have been defined:

|  |  |
| --- | --- |
| Procedures | Description |
| get\_patient\_information | Based on the patient ID, this operation collects extensive information about the patient. It returns the ID, name, birthdate, gender, phone number, address, age, and a list of allergies (if any). |
| get\_patient\_surgeries | Based on the patient’s ID, this procedure finds the surgeries linked with them. It returns the surgery ID, type, start time, end time, and names of the doctors that performed the surgery, and operating room information. |
| get\_doctor\_surgeries | Based on the doctor’s ID, this procedure finds the surgeries linked with a given doctor. It returns the surgery ID, type, start time, end time, patient names, and operating room information. |
| get\_doctor\_information | The procedure obtains information on the doctors who are affiliated with a given patient. It provides the doctor’s ID, name, specialty, and contact information based on the patient’s ID. |
| update\_patient\_contact\_info | This procedure updates the contact information (phone number and address) of a specific patient depending on their ID. |
| update\_doctor\_contact\_info | This procedure updates a specific doctor’s contact information (phone number) depending on their ID. |
| reschedule\_appointment | This procedure reschedules a patient’s appointment with a certain doctor. Based on the patient ID, doctor ID, and prior appointment data, it changes the appointment date and time. |
| get\_available\_operating\_rooms | This procedure returns the operating rooms that are available on a particular day and time. It returns the available rooms’ room number and type. |
| get\_doctor\_patient\_count | This procedure determines how many patients are affiliated with each doctor. It returns the ID, name, and number of patients linked with the doctor. |

* **Security:**

|  |  |  |  |
| --- | --- | --- | --- |
| Username | Privilege Command | Description | Screenshot |
| patients | Execute | Allows user to run the get\_patient\_information stored procedure. |  |
| Allows user to run the get\_patient\_surgeries stored procedure. |
| Allows user to run the reschedule\_appointment stored procedure. |
| Allows user to run the get\_doctor\_information stored procedure. |
| Allows user to run the stored procedure update\_patient\_contact\_info. |
| doctors  doctors | Execute | Allows user to run the get\_patient\_information stored procedure. |  |
| Allows user to run the get\_patient\_surgeries stored procedure. |
| Allows user to run the get\_doctor\_surgeries stored procedure. |
| Allows user to run the update\_doctor\_contact\_info stored procedure. |
| Allows user to run the reschedule\_appointment stored procedure. |
| Allows user to run the get\_available\_operating\_rooms stored procedure. |
| Allows user to run the get\_doctor\_information stored procedure. |
| Select On (Views) | Allows user to select from the patient\_summary\_view view. |  |
| Allows user to select from the patient\_surgery\_details\_view view. |
| Allows user to select from the doctor\_surgery\_details\_view view. |
| Allows user to select from the patient\_doctor\_information\_view view. |
| Allows user to select from the operating\_room\_equipment\_summary\_view view. |
| All Privileges | All privileges (select, insert, update, delete and so on) to PATIENT table. |  |
| All privileges (select, insert, update, delete and so on) to SURGERY table. |
| All privileges (select, insert, update, delete and so on) to DOCTOR\_PATIENT table. |
| All privileges (select, insert, update, delete and so on) to PATIENT\_ALLERGIES table. |
| Select On (Tables) | Allows user to select from the OR\_BED table. |  |
| Allows user to select from the OPERATING\_ROOM table. |
| Allows user to select from the OR\_EQUIPMENT table |
| admin | All Privileges | Gives user full access to the ihdb database. |  |

* **User Interface:**
* **Flowchart and Data Movement Diagrams:**

Before getting into the exact flowchart and data flow diagram (DFD) discussed in the report, it’s crucial to first define these tools. A flowchart is a graphical representation of a process that highlights the stages and their sequence, while a data flow diagram (DFD) is a graphical depiction of the flow of data inside an information system that shows how data is processed and transported between entities and processes.

**Flowchart Explanation:**

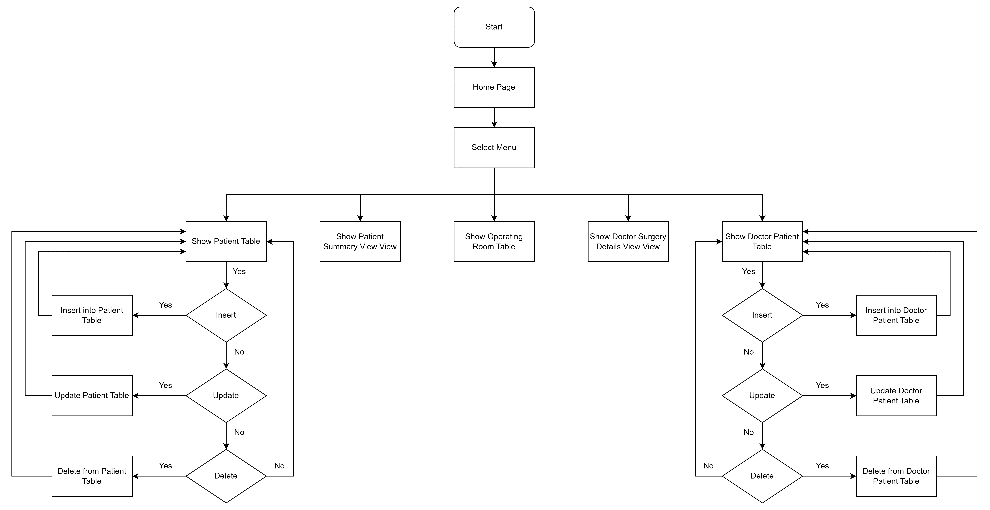
The flowchart shows the series of actions performed by a doctor user on the IHDB database system. The doctor enters the system and navigates to the main page. The doctor has several choices from the menu:

* Show Patient Table
* Show Doctor Patient Table
* Show Operating Room Table
* Show Patient Summary View
* Show Doctor Surgery Details View

What the doctor aims to achieve dictates the kind of process used. For instance, the doctor may choose the Table or View they like in order to view patient information.

There are several actions that may be taken for each table view (such as “Patient Table” or “Doctor Patient Table”):

1. **Insert:** If a new record has to be added to the database, the doctor might opt to insert. If “yes,” the system will proceed to Insert into Patient Table or Insert into Doctor Patient Table, and the modified table will be displayed again. If no the doctor has to choose whether they want to update patient records or not.
2. **Update:** If an existing record has to be updated, the doctor can opt to edit it. If “yes,” the system will move to Update Patient Table or Update Doctor Patient Table, and the revised table will be shown again. If no the doctor has to choose whether they want to delete patient records or not.
3. **Delete:** If a record has to be deleted the doctor can choose to delete it. If “yes,” the system will delete from the appropriate table, and the modified table will be presented again. If no, the system will show the selected table view again.



Flowchart

**Data Flow Diagram (DFD):**

The data flow diagram (DFD) depicts how data in the IHDB system is moved and processed.

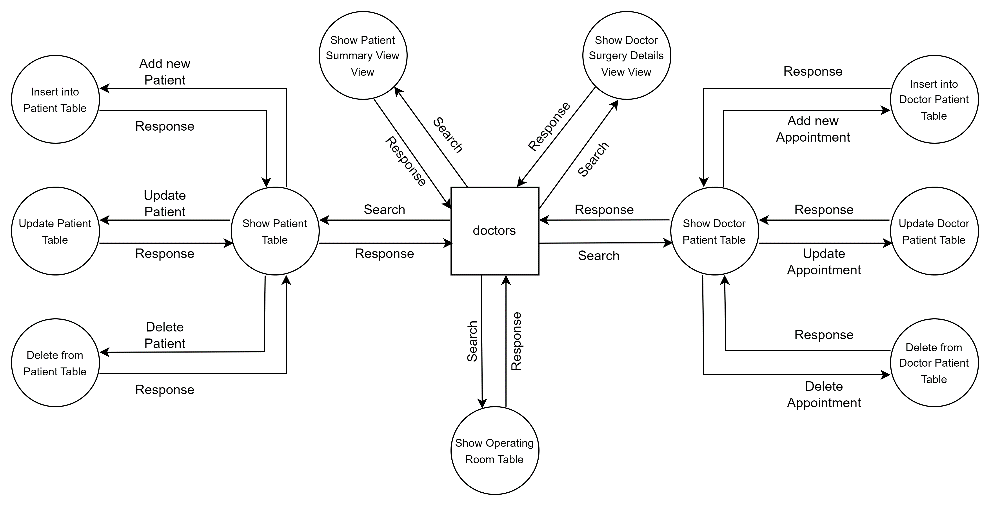
The system begins by sending a search request to one of the five tables or views based on the doctor’s selection from the menu:

* Show Patient Table
* Show Doctor Patient Table
* Show Operating Room Table
* Show Patient Summary View
* Show Doctor Surgery Details View

Depending on the operation selected, the relevant table view will respond to the doctors user and show the table view.

The DFD expands on the multiple forms of data processing for the “Show Patient Table” and “Show Doctor Patient Table” operations: Insert, Update, and Delete. Each action is represented by a data flow between the associated table view and the database table in question.

For example, in the case of “Show Patient Table,” if the doctor selects “Add New Patient,” this initiates a data flow to “Insert into Patient Table.” After the new data has been entered, a response is returned to “Show Patient Table,” indicating the table’s modified status. The same procedure applies to update and delete operations, with each resulting in the IHDB receiving an updated view of the table. The “Show Doctor Patient Table” action follows a similar pattern of data and replies.

****

Data Flow Diagram

* **Interfaces Development:**

|  |  |  |
| --- | --- | --- |
| Title | Description | Screenshot |
| Home Page | This page is the interface’s home page which allows the doctors user to choose one of 5 pages to access (Patient Summary View, Doctor Patient, Operating Room, Patient, Doctor Surgery Details View) |  |
| Patient Summary View | This page is based on the patient\_summary\_view view in the database, as it shows all the data that is in this view which is a result of joining the Patient and Patient Allergy table. It provides a full summary of all of the patients. Since this page is based on a view, it means that it can only be viewed (cannot update, insert, or delete values). |  |
| Doctor Patient | This page is based on the Doctor\_Patient table in ihdb, as it shows all the data that is stored in this table, which displays the IDs of the doctors with their associated patients, along with their appointment dates and times. Since this page is based on a table, and since the doctors user has all privileges to this table, it means that the doctor can select, update, insert, or delete values in this table. |  |
| Operating Room | This page is based on the Operating\_Room table in ihdb, as it shows all the data that is stored in this table, which displays the IDs of the operating rooms with their associated room number, type and capacity. Since this page is based on a table, and since the doctors user has only the select privilege to this table, it means that the doctor can only select from this table. |  |
| Patient | This page is based on the Patient table in ihdb, as it shows all the data that is stored in this table, which displays the IDs of the patients with their associated names, birth dates, genders, phone numbers and addresses. Since this page is based on a table, and since the doctors user has all privileges to this table, it means that the doctor can select, update, insert, or delete values in this table. |  |
| Doctor Surgery Details View | This page is based on the doctor\_surgery\_details\_view view in ihdb, as it shows all the data that is in this view which is a result of joining the Surgery, Patient and Operating\_Room tables together, as it provides a full summary of all of the doctors’ surgery schedules. Since this page is based on a view, it means that it can only be viewed (cannot update, insert, or delete values). |  |

**Maintenance:**

* **Database Recovery & Backups:** (Javatpoint, 2019; Leader *et al.*, 2020; NetApp, 2020; Oracle, 2022)

Database recovery and backups are critical components of database administration, particularly for large and complex systems like the IHDB. They protect against unforeseen events that could result in data loss or corruption. While backups are a preventative measure that creates recoverable copies of data, recovery is a curative action that attempts to restore the database’s state following a failure or corruption event.

* **Data Recovery:**

Database recovery describes the collection of procedures used to return a database to a prior, uncorrupted state after an unforeseen failure or system breakdown. These occurrences can be brought on by a number of things, such as hardware or software malfunctions, power outages, malicious attacks, or human mistake, including unintentional data loss.

Transaction logging is a fundamental mechanism for recovery. Transaction logs record all database modifications, providing a historical record of all actions. Transaction logs are used in two primary recovery techniques: roll-forward and roll-back. The log is used by the system in the roll-forward method to reapply completed transactions after restoring a previous backup, whereas the log is used in the roll-back method to undo the effects of incomplete or erroneous transactions.

Effective recovery depends on atomicity principles being followed. Atomicity, which ensures that each transaction is viewed as a single, indivisible operation that either succeeds entirely or fails completely, is one of the essential properties of database transactions. In order to guarantee data consistency, the database must roll back any changes made during a transaction if it is interrupted, maybe because of a system failure.

A layered recovery strategy frequently works well for a large-scale database. The recovery models used in this tactic could be Full, Bulk-Logged, or Simple, with each one providing a different level of data protection and system resource usage. The criticality of the data, the type and frequency of data updates, and acceptable data loss limits are just a few examples of the variables that influence the choice of the best recovery model.

* **Data Backup:**

Database backups, which are copies of the entire database or a subset of it stored separately from the original database, are the cornerstone of data protection. For the database to be restored to a usable state after incidents involving data loss or corruption, regularly scheduled backups are essential.

There are three main categories of backups, each with a distinct function:

1. **Full Backups:** These are complete copies of the entire database. A full backup includes all data and a portion of the transaction log, allowing the database to be restored to the exact state it was in when the backup finished. The most thorough form of data security is offered by full backups, but they can be time- and resource-consuming to do.
2. **Differential backups:** These backups just record the data that has changed since the previous full backup. Differential backups must be used alongside a full backup during a restore operation even though they are typically faster and use less storage than full backups. Differential backups offer an effective way to safeguard recent database changes by minimizing the backup size and time.
3. **Transaction Log Backups:** These are copies of the transaction log’s active section that have been made. All transactions and database changes since the most recent transaction log backup are recoverable using them. Regular transaction log backups are essential in the Full recovery model because a full transaction log could make it impossible to run your database.

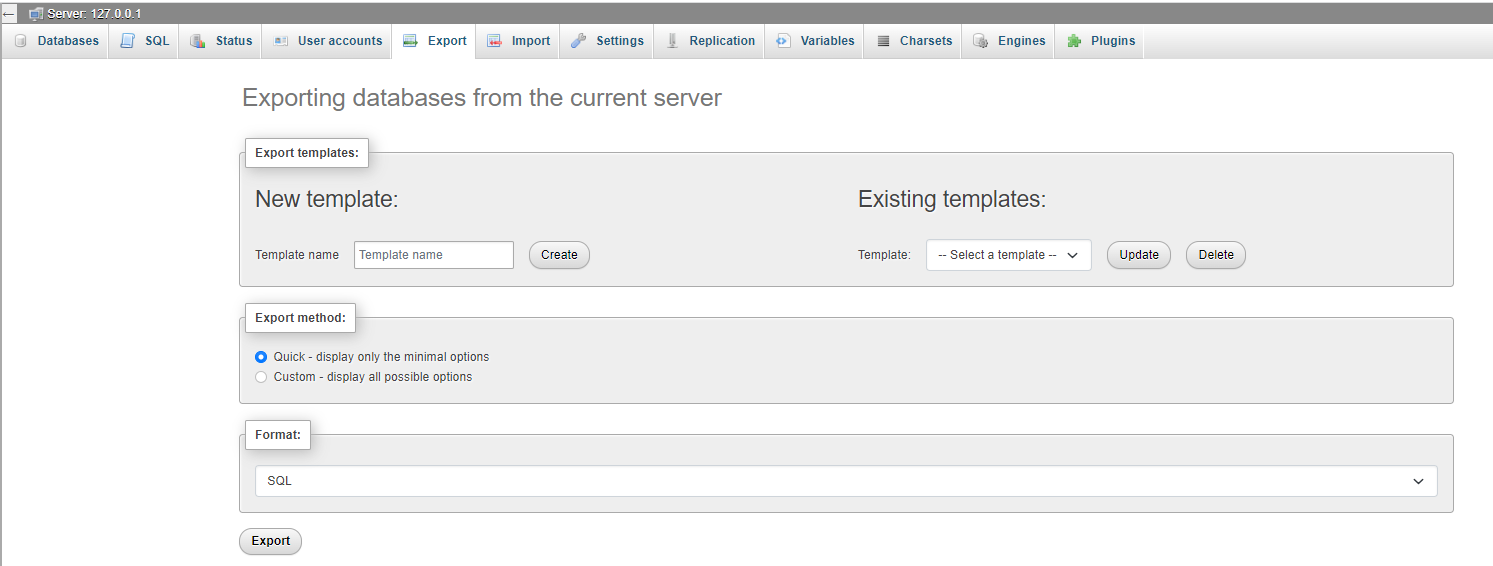
The size, volatility, and business needs of the database determine how frequently backups are performed. Weekly full backups, daily differential backups, and hourly backups of transaction logs could be standard procedures. This multi-layered strategy guarantees a thorough, flexible data protection system that balances resource usage and data security.

A simple and well-supported approach for doing database recovery and backup is to export a database to a .sql format. A number of tools and database management systems on the market can be used to carry out this strategy.

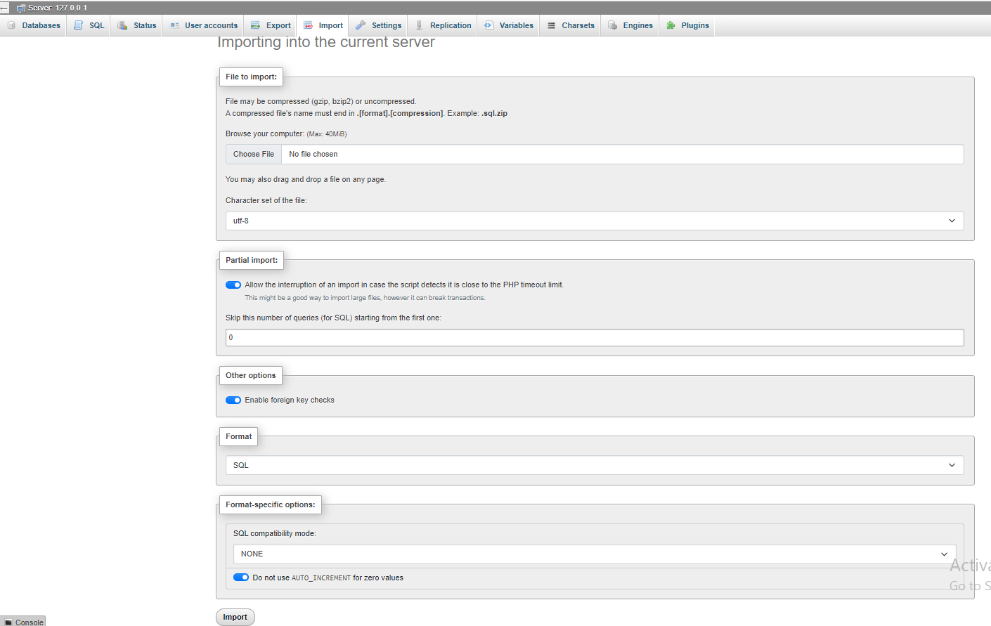
When a database is exported to a .sql format, a file is created that contains all the necessary instructions for recreating the database's full structure and contents. This .sql file acts as a thorough backup that may be kept apart from the original database to ensure the security and integrity of that data. Organizations may successfully protect their sensitive data from potential disasters or system failures by establishing a separate backup file in .sql format.

The database design, table definitions, constraints, indexes, and the actual data kept in the tables are all represented in the .sql file's sequence of SQL statements. With the help of these SQL statements, you may recreate the database precisely as it was at the time of export. This enables a complete restoration of the database to its original condition since not only is the database's structure maintained, but also the data itself.

The .sql import of an exported database is essential for returning the database to its original condition. The import procedure uses the instructions in the .sql file to reconstruct the whole database structure and contents when a database experiences unanticipated failures, corruption, or data loss. The database management system reconstructs the tables, relationships, and records by executing the SQL instructions in order, thus restoring the database to its initial state. This procedure guarantees data consistency and enables healthcare professionals to effortlessly resume their work while accessing the important data kept in the IHDB.



**Database Export**



**Database Import**

* **Database Maintenance in General:** (Office Tools, 2015; CHRIS ODOGWU, 2018; IBM Documentation, 2020; McMahon, 2022)

Effective database maintenance is a multifaceted process that is required to ensure optimal database performance, data integrity, and efficient data management. Given the critical nature of healthcare data and the need for system reliability, routine maintenance is even more important. This procedure consists of several operations, each of which addresses a different aspect of database functionality. Importing and exporting data, database optimization, consistent monitoring, security updates, data validation, and data archiving are all important activities.

1. **Database Optimization:** it entails reorganizing the database and its objects in order to improve performance, efficiency, and response time. Database optimization can include processes such as index rebuilding, using descriptive names, storing data in a single file, updating statistics, defragmentation, and so on. Regular optimization aids in database health maintenance.
2. **Monitoring:** Regular database monitoring aids in the detection and prevention of issues. Monitoring should include tracking database size, error logs, long-running queries, backup status, and so on.
3. **Security Updates:** Updating the database management software on a regular basis ensures that the system is secure against known vulnerabilities. Patches and updates should be applied as soon as they are released by the software provider.
4. **Data Validation:** The database's data is checked often for accuracy, completeness, and compliance. Field validations (such as data type, size, and format) and business rule validations are examples of these checks.
5. **Data Archiving:** The database may accumulate a large amount of historical data that is rarely accessed but still important over time. Data archiving is the process of transferring old data from primary storage to a less expensive storage system. This process helps to maintain database performance while preserving data for future use.

Lastly, general database maintenance is an important aspect of database management, especially for a system like the IHDB. We can ensure the database’s health, performance, and longevity by performing these tasks diligently. Regular maintenance protects the database from potential problems, improves its performance, and ultimately aids in the delivery of high-quality healthcare services. This process emphasizes the importance of the IHDB’s data and our commitment to maintaining its accuracy, accessibility, and relevance to healthcare practitioners.

**Testing:**

* **Data Validation:**

|  |  |  |
| --- | --- | --- |
| Type | Description | Screenshot |
| Primary Key | Inserting a record with an existing primary key |  |
| Inserting a record without specifying a value for the primary key column (Auto-Increment) |  |
| Foreign Key | Inserting a record with a non-existing primary key for the foreign key value |  |
| Updating the primary key and checking what happens to the foreign key |  |
| Deleting the primary key and checking what happens to the foreign key |  |
| Unique | Testing the unique constraint on the doctor.phone\_number by giving a new doctor the same phone\_number as an already registered doctor |  |
| Default | Testing the default constraint in the operating\_room.capacity by not specifying the capacity of the operating room in a new operating room |  |
| NOT NULL | Testing the NOT NULL constraint on patient.first\_name by inserting a patient with a NULL first name |  |
| ENUM and CHECK | Testing the ENUM and CHECK constraints as they work hand in hand to check that the values inserted in the patient.gender column are either ‘M’ or ‘F’ |  |

* **Output Validation:**

|  |  |  |
| --- | --- | --- |
| Query Description | Screenshot (query + result) | Result validation |
| Using Select Statement | **Query:**  **Result:** | As we can see from the screenshot above, the query yielded the expected output as it provided us with all the data from the ‘or\_bed’ table. This was expected as “SELECT \*” yields back all the columns in the table. |
| Using Select Statement with Where | **Query:**    **Result:** | As we can see from the screenshot above, the query yielded the expected output as it provided us with all the data of patient\_id = 1 from the ‘patient’ table. This was expected as “SELECT \*” yields back all the columns in the table, and the “WHERE” yields back the row with the matching value (patient\_id = 1). |
| Using Select Statement with Inner Join | **Query:**    **Result:** | As we can see from the screenshot above, the query yielded the expected output as it joined (inner join) all the data from the ‘patient’ table and ‘patient\_allergy’ table as it provided us with all values where each patient\_id was present in both tables. This was expected as “SELECT \*” yields back all the columns in the table and “JOIN” joins the tables on a specific feature (patient\_id). |
| Using Select Statement with Left Join | **Query:**  **Result:** | As we can see from the screenshot above, the query yielded the expected output as it joined (left join) all the data from the ‘patient’ table and ‘patient\_allergy’ table as it provided us with all values of both tables joined together regardless of whether the patient\_id is present in the right table (patient\_allergy). This was expected as “SELECT \*” yields back all the columns in the table and “LEFT JOIN” joins the tables on a specific feature (patient\_id). |
| Using Select Statement to Select Specific Columns with Having and Group By Statements | **Query:**    **Result:** | As we can see from the screenshot above, the query yielded the expected output as it provided us with the doctors’ IDs, first and last names who have performed two or more surgeries. This was expected as “SELECT” yields back only the columns specified (doctor\_id, first\_name, last\_name), and “GROUP BY” yields back the grouping of both the Doctor and Surgery Table, and finally “HAVING” only shows us the doctors who have performed two or more surgeries. |
| Update Statement | **Query:**  **Result:** | As we can see from the screenshot above, the query yielded the expected output as it only updates the phone number of patient\_id = 1. This was expected as “UPDATE” and “SET” updates values of the phone\_number column, and “WHERE” specifies which rows needs to be updated. |
| Insert Statement | **Query:**    **Result:** | As we can see from the screenshot above, the query yielded the expected output as it only inserted the new row to the data. This was expected as “INSERT INTO” and “VALUES” inserts new values to the table, thus resulting in a new row. |
| Delete Statement | **Query:**    **Result:** | As we can see from the screenshot above, the query yielded the expected output as it only deletes the patient with patient\_id = 11. This was expected as “DELETE FROM” deletes the values, and “WHERE” specifies which rows needs to be deleted. |
| Select Statement on a View | **Query:**    **Result:** | As we can see from the screenshot above, the query yielded the expected output as the view joined between the Surgery, Doctor, and Operating Room tables, as it should have. |
| Procedure that only shows data | **Query:**    **Result:** | As we can see from the screenshot above, the query yielded the expected output as it showed us all the surgeries on a specified patient (patient\_id = 1). This procedure works by selecting the data from the patient\_surgery\_details\_view that joins between Patient, Doctor, and Surgery tables and then specifying only the rows based on the given patient\_id using a “WHERE” statement. |
| Procedure that modifies on the data | **Query:**  **Result:** | As we can see from the screenshot above, the query yielded the expected output as it only updates the phone number and address of patient\_id =1. This was expected as “update\_patient\_contact\_info” procedure updates a specific patient’s phone number and address based on its first input (patient\_id) in order to specify which patient wants to update their contact info, and with the second and third inputs (phone\_number and address) the values of the phone\_number and address in the original table changes to their new values. |

* **Security Validation:**

|  |  |  |
| --- | --- | --- |
| User Name | Description of privilege/no privilege | Screenshot (query + result) |
| patients  patients | **Execute**  **No Privilege:** Calling the get\_doctor\_patient\_count() procedure  **Privilege:** Calling the update\_patient\_contact\_info() procedure | **No Privilege:**    **Privilege:**    Patient table before update    Successful Query    Patient table after update |
| **Select**  **No Privilege:** Selecting values in Patient table  **Privilege:** patients have no direct select privileges to any specific table | **No Privilege:** |
| **Update**  **No Privilege:** Updating values in Patient table  **Privilege:** patients have no direct update privileges to any specific table | **No Privilege:** |
| **Insert**  **No Privilege:** Inserting values into Patient table  **Privilege:** patients have no direct insert privileges to any specific table | **No Privilege:** |
| **Delete**  **No Privilege:** Deleting values from Patient table  **Privilege:** patients have no direct delete privileges to any specific table | **No Privilege:** |
| doctors  doctors  doctors  doctors | **Execute**  **No Privilege:** Calling the get\_doctor\_patient\_count() procedure  **Privilege:** Calling the get\_patient\_information() procedure | **No Privilege:**    **Privilege:**    Original Data    Successful Query    The result of the query matches the original data |
| **Select**  **No Privilege:** Selecting Values in Doctor table  **Privilege:** Selecting values in Patient table | **No Privilege:**    **Privilege:**    Original Data    Successful Query    The result of the query matches the original data |
| **Update**  **No Privilege:** Updating values in the Doctor table  **Privilege:** Updating values in the Patient table | **No Privilege:**    **Privilege:**Original Data  Successful Query    The modifications on the data matches the query |
| **Insert**  **No Privilege:** Inserting values into Doctor table  **Privilege:** Inserting values into Patient table | **No Privilege:**  **Privilege:**  Original Data    Successful Query    The result after inserting into the table |
| **Delete**  **No Privilege:** Deleting values from Doctor table  **Privilege:** Deleting values from Patient table | **No Privilege:**    **Privilege:**    Original Data    Successful Query    The result after deleting the row |
| admin  admin  admin | **Execute**  **No Privilege:** there are no procedures that admins don’t have the execute privilege to  **Privilege:** Calling the update\_patient\_contact\_info() procedure | **Privilege:**    Successful Query    The result from the procedure call |
| **Select**  **No Privilege:** there are no tables or views that admins don’t have the select privilege to  **Privilege:** Selecting values from operating\_room \_equipment\_summary\_view view | **Privilege:**    Original Data    Successful Query    The results of the view |
| **Update**  **No Privilege:** there are no tables or views that admins don’t have the update privilege to  **Privilege:** Updating values in the Patient table | **Privilege:**Original Data  Successful Query    The modifications on the data matches the query |
| **Insert**  **No Privilege:** there are no tables or views that admins don’t have the insert privilege to  **Privilege:** Inserting values into Patient table | **Privilege:**  Original Data    Successful Query    The result after inserting into the table |
| **Delete**  **No Privilege:** there are no tables or views that admins don’t have the delete privilege to  **Privilege:** Deleting values from Patient table | **Privilege:**    Original Data    Successful Query    The result after deleting the row |

* **GUI Validation:**

|  |  |
| --- | --- |
| Description | Screenshot |
| Checking whether the data that is displayed on the interface matches the data on the database  (SELECT \*) | **Patient Summary View**    **Doctor Patient**    **Operating Room**    **Patient**    **Doctor Surgery Details View** |
| Testing the Update Statement on the interface’s tables and whether or not it is reflected on the database, and checking the user’s privileges | **Doctor Patient**      **Operating Room**      **Patient** |
| Testing the Delete Statement on the interface’s tables and whether or not it is reflected on the database, and checking the user’s privileges | **Doctor Patient**        **Operating Room**        **Patient** |
| Testing the Insert Statement on the interface’s tables and whether or not it is reflected on the database, and checking the user’s privileges | **Doctor Patient**        **Operating Room**      **Patient** |

* **Assess whether Meaningful Data Has Been Extracted:**

In this section, we extensively analyzed the extracted data’s quality in terms of realism, accuracy, completeness, consistency, relevance, and coverage of various relationship types (multivalued, many-to-many, one-to-many, one-to-one, and so on). The evaluation seeks to verify whether meaningful data was properly retrieved and inserted into database tables.

**Realism and Correctness:**

The derived data is quite realistic and accurate. The data follows standard naming rules and appropriately depicts typical records seen in a hospital setting. Patient names, doctor specializations, appointment dates, and surgical information all look credible and correspond to real-world events.

**Relevance:**

The extracted data is extremely relevant to the healthcare space and properly corresponds with the database’s objective. It contains vital information including patient demographics, doctor specializations, appointment schedules, and surgical specifics. The data’s relevance permits its use in numerous healthcare management and analytical activities, contributing to the database’s overall efficacy.

**Consistency:**

The data is very consistent in terms of data types, constraints, and relationships. Primary keys, foreign keys, and constraints in the prescribed table architectures ensure data integrity. Table relationships, such as doctor-patient appointments and procedures, are appropriately constructed using the right foreign key references, guaranteeing consistency across linked entities.

**Completeness:**

The collected data is of noteworthy completeness. Multiple records are used to populate the tables, guaranteeing a diversified representation of patients, doctors, appointments, surgeries, operating rooms, bed sizes, patient allergies, and operating room equipment. The dataset includes patients and doctors with and without middle names, patients of different genders, patients with several appointments with different doctors, operating rooms with multiple equipment, doctors with many surgeries, and patients with many surgeries, along with many other cases that ensure the data’s completeness.

**Relationship Types Covered**

Evaluating the presence and correctness of connection types is critical for guaranteeing the database’s integrity and usefulness.

**One-to-Many Relationships:**

There is a notable example of a one-to-many link between patients and procedures in the retrieved data. A patient (one) can have many procedures done on them (many). This link is shown in the “SURGERY” database, where each surgical record corresponds to a certain patient. The patient\_id column in the “SURGERY” database serves as a foreign key, referencing the patient\_id column in the “PATIENT” table. This one-to-many connection appropriately represents the real-world scenario in which a patient may require several procedures.

**One-to-One Relationships:**

The data also includes one-to-one relationships, guaranteeing that entities are uniquely associated. The link between the “OPERATING\_ROOM” table and the “OR\_BED” table is an example of this. Each operating room (one) is linked to a single OR bed (one). The primary key and foreign key relationship between the two tables represents this relationship. The “OR\_BED” table’s operating\_room\_id refers to the “OPERATING\_ROOM” table’s main key operating\_room\_id. This one-to-one link ensures that each operating room has just one OR bed and vice versa.

**Many-to-Many Relationships:**

The extracted data captures many-to-many relationships, which are common in healthcare systems. The interaction between doctors and their patients is a good example. numerous physicians can see numerous patients, and vice versa. The “DOCTOR\_PATIENT” table represents this many-to-many relationship by acting as a bridge table between the “DOCTOR” and “PATIENT” databases. Each record in the “DOCTOR\_PATIENT” database represents an appointment between a single doctor and a specific patient. This style enables for the flexible portrayal of diverse doctor-patient interactions while also ensuring accurate appointment monitoring.

* **Assess the Effectiveness of the Testing:**

Testing is crucial in the complex world of database management and development, acting as the first line of defense against data inconsistency, incorrect outputs, security breaches, and improper user interfaces. This database testing evaluation has four main components: data validation, output validation, security validation, and graphical user interface (GUI) validation. Each component is distinct in its approach and adds significantly to the database’s overall efficacy.

**Data Validation:**

Data validation is critical, serving as the first line of defense against data inconsistencies, incorrect outputs, security breaches, and poor user interfaces. This section discusses validation of primary keys, foreign keys, unique constraint validation, default validation, NOT NULL validation, and ENUM and CHECK validation. Each validation procedure contributes to data consistency and integrity, hence increasing the database’s overall efficacy.

**Primary Keys Validation:**

The primary key is the foundation of every table in a relational database. It serves as a unique identifier for each entry, guaranteeing that there are no duplicate entries. Inserting records with an existing primary key and automatically incrementing primary keys is used to test the primary key validity. This type of validation is critical because it ensures record uniqueness and the absence of NULL values in primary key fields, thus contributing to data consistency and the retrieval process.

**Foreign Key Validation:**

Foreign keys are fields in one table that function as primary keys in another table in a relational database. This interlinking creates a linkage between tables, improving data coherence and avoiding data inconsistencies. Foreign key validation entails running several scenarios, such as entering records with a non-existing primary key and modifying or removing the primary key in order to see how it affects the corresponding foreign keys. This testing guarantees that the database has referential integrity, strengthening linkages between tables and prevents orphan entries from being created.

**Unique Constraint Validation:**

Validating the unique constraint entails determining if a certain column can hold a unique value across several entries. This is accomplished by attempting to insert a duplicate value into a column that has a unique constraint. This type of validation is essential since it guarantees that columns with unique values, such as phone number, preserve data dependability and accuracy.

**Default Validation:**

Checks for the automatic assignment of predefined values to particular columns when they are not explicitly specified during insertion. This prevents NULL or unexpected values from being stored in specific fields, ensuring that data stays consistent and predictable even when certain values are not entered during data entry.

**NOT NULL Validation:**

The NOT NULL constraint validates that important fields in a table do not have NULL values. To test this, we entered a record with a NULL value into a column labeled NOT NULL. This guarantees that data in crucial areas is full and prevents any mistakes or inconsistencies during data retrieval or processing.

**ENUM and CHECK Validation:**

ENUM and CHECK validations provide domain integrity by guaranteeing that only the required values are kept in the database. ENUM is a predefined set of values that a column can accept, and CHECK is a condition that the data in a column must fulfill. In order to ensure that only legitimate and intended data is saved, these constraints are tested by attempting to enter invalid values into the constrained columns.

Data validation is critical to assuring the integrity and dependability of a database. We confirmed the accuracy, integrity, and security of the data contained in the database by rigorous testing. Primary keys, foreign keys, unique constraints, default values, NOT NULL constraints, and ENUM and CHECK constraints were all thoroughly examined to ensure data consistency. This enables enterprises to rely on their databases with confidence for accurate information, safe access, and smooth user interactions.

**Output Validation:**

Output Validation: In a database, output validation is critical for guaranteeing the accuracy and validity of query results. We can evaluate the resilience and efficacy of the database’s operational capabilities by extensively testing and confirming the outcome of various queries.

**SELECT Statements:**

SELECT statements were used to validate the effectiveness of data retrieval methods. Each SELECT command provided the anticipated data from the corresponding tables, confirming the database’s dependability. For example, a simple SELECT operation on the ‘or\_bed’ table yielded the whole contents of the table. Furthermore, a SELECT query with a WHERE clause successfully obtained all data from the ‘patient’ table with patient\_id = 1.

**JOIN Statements:**

JOIN operations like as Inner JOIN and Left JOIN were tested to verify that records from several tables were combined correctly. The Inner JOIN operation performed between the ‘patient’ and ‘patient\_allergy’ databases gave all values when each patient\_id was present in both tables. Similarly, regardless of whether the patient\_id was present in the ‘patient\_allergy’ table, the Left JOIN function returned all values from both tables. This validation validates the data’s integrity and coherence across connected tables.

**SELECT Statements with HAVING and GROUP BY:**

SELECT statements were tested with HAVING and GROUP BY clauses to guarantee correct data aggregation and filtering. A query, for example, provided just the IDs, first names, and last names of doctors who had done two or more procedures. This validator guarantees that complicated searches requiring data aggregation and filtering are handled correctly.

**UPDATE, INSERT, and DELETE Statements:**

The accuracy of UPDATE, INSERT, and DELETE statements was checked to verify that the database was modified as intended. Specific actions, such as altering patient\_id = 1’s phone number, adding new rows, and removing patient\_id = 11, produced the intended effects. This validation ensures that data manipulation procedures work properly.

**Procedures and Using SELECT on View:**

The view’s SELECT query successfully obtained joined data from the Surgery, Doctor, and Operating Room databases. Procedures that displayed or changed data, such as presenting surgeries for a specific patient (patient\_id = 1) or changing a patient’s contact information, also worked as expected. This validation assures that sophisticated data retrieval and manipulation activities using views and procedures are reliable.

We can be confident in the quality and correctness of the query results by doing rigorous output validation. This procedure adds to the database’s effectiveness by validating its operational capability and ensuring that data retrieval, manipulation, and aggregation processes work as planned.

**Security Validation:**

Security validation tests were performed to validate the database’s user roles, privileges, and access controls. These tests were designed to check that the database correctly limited or permitted user operations depending on their given roles and privileges, ensuring safe database access.

**Patients:**

Patients were given limited privileges in order to ensure data confidentiality and avoid illegal changes. They lacked direct SELECT, UPDATE, INSERT, and DELETE access to all tables and views. Patients, for example, did not have access to the get\_doctor\_patient\_count() operation, which protects sensitive information. They did, however, have the ability to run the update\_patient\_contact\_info() process, which allowed them to update their own contact information within the specified bounds.

**Doctors:**

Specific privileges were assigned to doctors to allow them to access relevant patient data while respecting their own privacy. They may access patient data by selecting values from the Patient table. They were, however, barred from entering the Doctor table to protect the confidentiality of other doctors’ information. Doctors had the ability to update and insert values into the Patient table, but not the Doctor table, when it came to data modification. Doctors were unable to perform the get\_doctor\_patient\_count() operation, which safeguards critical statistics, but they were able to run the get\_patient\_information() process, which retrieves patient data for their own practice.

**Admins:**

Admins were given broad permissions to help with database management and maintenance. They had complete access to the database and could run whatever process they wanted. Admins also have SELECT, UPDATE, INSERT, and DELETE access to all tables and views. This broad access allowed that administrators could carry out critical administrative operations such data retrieval, update, insertion, and deletion without meeting any visible limitations.

To ensure the integrity and confidentiality of the data, rigorous security validation tests were performed on the database’s user roles, privileges, and access restrictions. This validation process ensures that the security protections in place successfully restrict or enable user actions based on their assigned roles and privileges, guaranteeing secure access and protecting critical information within the database.

**GUI Validation:**

GUI validation tests were performed to ensure that the database and its user interface interacted properly. To provide a smooth user experience, these tests focused on several areas of the GUI, such as data correctness, data updates, and data input and deletion.

**Data Accuracy Check:**

The presented information on the interface was compared to the data recorded in the database to confirm data correctness. Several viewpoints were evaluated, including Patient Summary, Doctor Patient, Operating Room, Patient, and Doctor Surgery Details View. The validation procedure validated that the data displayed on the interface matched the data in the database exactly. As a result, users can trust the interface to give accurate and up-to-date information from the underlying database.

**Data Update Check:**

Data update checks were used to ensure that the interface appropriately reflected changes made to the database. Changes performed via the interface, such as conducting UPDATE operations on tables such as Doctor Patient, Operating Room, and Patient, were investigated. The validation mechanism verified that changes made in the interface were mirrored accurately in the underlying MySQL database. This validation ensures that the interface interacts with the database correctly to edit data.

**Data Insertion and Deletion Checks:**

Data insertion and deletion checks were used to evaluate the interface’s functionality in adding and removing database entries. INSERT and DELETE operations on the interface’s tables were done, and the associated changes in the MySQL database were noticed. This validation guarantees that the interface handles data addition and deletion correctly, keeping the interface and the underlying database consistent.

The relationship between the database and its user interface was carefully evaluated by conducting extensive GUI validation tests. The testing demonstrated that the interface reliably displays data from the database, effectively reflects data changes, and performs data insertion and deletion activities appropriately. This validation method guarantees a dependable and user-friendly interface that interacts with the underlying database flawlessly.

A critical part of determining the efficiency of database testing was the selection of the suitable test data. It covered a wide range of situations and edge cases involving various data types, sizes, and formats that the database is required to manage. To guarantee that the system can handle unforeseen events, realistic inputs and probable faults. The selection of test data takes into account the database’s specific requirements and unique characteristics. Testers may detect possible faults, confirm the database’s performance, and give useful ideas for improvement by carefully choosing test data.

**Evaluation of Database Solution:**

* **Effectiveness of the Database Solution Based on User and System Requirements:**

Based on the defined user and system requirements, the effectiveness of the Ibn Al Haitham Database (IHDB) solution was assessed. The database solution was created and installed to fulfill the unique demands of patients, doctors, and hospital administrators while conforming to technical specifications and performance standards.

**User Requirements:**

**Patients:**

The IHDB solution meets patients’ needs by providing them with quick and safe access to their personal information and healthcare services. Patients may quickly access, add, or update their personal information, such as phone numbers and addresses. The system guarantees that data is comprehensive and accurate, allowing patients to have confidence in the information held in their profiles.

Patients can view personal information such as their name, birthday, age, gender, allergies, and ID. This allows them to be able to access their health data and share it with healthcare providers as needed. Furthermore, the IHDB system allows patients to examine scheduled procedures, including surgical information and the operating room assigned to them. This information allows people to be prepared for and comprehend their future medical procedures.

The system also helps the doctor-patient interaction by informing them about the doctors who are involved in their treatment. Patients may view their physicians’ names and specializations, guaranteeing openness and establishing confidence. Patients may also browse available appointment slots with their designated providers, allowing them to arrange appointments that are convenient for them. Patients may also manage their appointments using the IHDB interface by viewing or updating planned appointments.

**Doctors:**

The IHDB solution successfully adapts to doctors’ individual needs, assisting them with their everyday work and boosting their capacity to offer quality healthcare services. Doctors may readily view, add, or change their personal information, including contact information such as their phone number. By updating their profiles, this service enables doctors to stay in contact with patients and other healthcare providers. A patient's demographics, contact details, and medical history may also be viewed by doctors.This comprehensive viewpoint helps doctors provide individualized and informed healthcare services to their patients.

Doctors may use the IHDB solution to handle upcoming operation information for patients under their care. Doctors may quickly enter or alter surgery information, allocate operating rooms, and update relevant information. The system also gives doctors access to information about operating rooms, such as the equipment accessible in each room. This feature assists surgeons in effectively planning and carrying out procedures, as well as ensuring the availability of required resources.

The IHDB interface allows doctors to view their appointment schedules. This enables doctors to efficiently manage their time and prepare for patient appointments. Doctors can also schedule appointments, change time slots, and schedule breaks or other activities. Doctors may also see their patients’ appointment history, which provides vital insights into their patients’ healthcare journey.

**Hospital Administrators:**

The IHDB system efficiently satisfies hospital administrators’ needs, giving them the tools they need to manage patient data, doctor information, surgical procedures, and operating room resources. Administrators have complete control over the database, since they can quickly add, select, search, edit, or remove any information to guarantee correctness and completeness, hence maintaining the database’s overall integrity. This feature allows administrators to keep an up-to-date directory of all data within the database while also ensuring adequate documentation and monitoring of that data.

**System Requirements:**

The IHDB solution was built and executed to fulfill the system’s requirements, assuring its effectiveness, performance, and security. With a well-defined schema and attribute mapping, the database system integrates conceptual, logical, and physical designs. The normalizing process ensures data completeness, consistency, and integrity up to the third normal form.

The IHDB solution employs best practices for indexing, optimizing query execution plans, and employing appropriate data structures to deliver optimal query and storage performance. This guarantees that consumers have fast responses while obtaining data.

The system is modular, flexible, and scalable, making it easy to maintain and supporting a rising number of users. The database design and documentation provide clear guidelines for maintenance and future expansions.

The IHDB system assures data security and patient privacy in order to comply with healthcare standards such as HIPAA. Robust access control measures are in place, with users being granted appropriate credentials based on their roles and responsibilities. Patient information is protected from unlawful access or disclosure, supporting patient confidence and confidentiality.

The IHDB solution has a user-friendly interface and a user manual, allowing users to easily operate the system.

* **Suggested Improvements:**

The following four significant enhancements have been identified to improve the functionality, performance, and usability of the Ibn Al Haitham Database (IHDB) system. These improvements will help to make the healthcare system more efficient and effective, enhancing patient care and maximizing resource use.

**Improved User Interface**

An improved user interface is critical to providing healthcare professionals engaging with the IHDB system with an intuitive and more user-friendly experience. Usability testing and user input will be critical in identifying problem spots for development. Based on the findings, the user interface may be modified to optimize data entry operations, simplify navigation, and give visual clues for improved information understanding. Clear and logical feature and data arrangement will allow for faster and more accurate access to patient information, appointment scheduling, and surgical management. The IHDB system will facilitate efficient and intuitive interaction by highlighting user-centric design principles, resulting in greater user satisfaction and productivity.

**Performance Optimization**

Optimizing the IHDB system’s performance is critical for effective data retrieval. Regular monitoring and analysis of system performance will assist in identifying and addressing any bottlenecks or inefficiencies. Index tuning, query plan analysis, and database schema refinement are all query optimization methods that may greatly improve response times and overall system performance. Fine-tuning hardware resources such as memory allocation and disk setup will improve data storage and retrieval even more. Healthcare workers will benefit from speedier access to patient information, appointment scheduling, and surgery planning as a result of assuring optimal performance, which will lead to better operational efficiency and rapid decision-making.

**Data Validation and Quality Checks**

To preserve data integrity and consistency inside the IHDB system, strong data validation and quality checks must be implemented. This involves verifying data types, imposing referential integrity requirements, and conducting range checks to guarantee that the database contains only correct and legitimate data. Regular audits and data quality checks will detect and correct any errors or irregularities in the data. The IHDB system will eliminate data entry mistakes, reduce data duplication, and improve the dependability and quality of information stored by applying tight data validation requirements. This will allow for better clinical decision-making, more accurate reporting, and more dependable data analysis. Also during this step, we will monitor the if the database might require any future modifications, or additions, such as the need to store multiple values of the same feature (e.g. phone\_number), then we would have to create a new table for this new multivalued attribte, thus, changing the database’s schema, and modifying its tables.

**Mobile Access and Telemedicine Support**

Enabling mobile access to the IHDB system via a specialized mobile application or a responsive online interface would increase healthcare professionals’ flexibility and accessibility. Doctors will be able to check patient information, arrange visits, and supervise procedures while on the go, increasing productivity and allowing remote care delivery. Integrating with telemedicine systems will allow for secure video consultations, decreasing the need for in-person visits and improving patient convenience. The IHDB system will enhance patient-centered care, increase healthcare provider efficiency, and enable distant cooperation among medical experts by incorporating mobile technologies and telemedicine assistance.

These four identified enhancements will together improve the IHDB system’s functionality, efficiency, and usability. The healthcare system will benefit from simpler processes, improved decision-making, and improved patient care outcomes if these enhancements are implemented.

* **Evaluation Based on Improvements Needed:**

Our database system was created with an emphasis on flexibility, scalability, and a user-centric approach as specified in the system requirements, establishing the groundwork for future updates and enhancements. This well-thought-out design technique supports the system’s strength and adaptability, ensuring its usefulness over time while smoothly incorporating crucial changes and developments.

**Modular Structure and User Interface**

The modular design of our database system is a key feature. This structure enables each component to be updated or modified independently. This strategy not only ensures the system’s long-term maintainability and flexibility, but also ensures that changes, such as user interface modifications, may be readily implemented. Because of the modular nature, we can improve the interface without interfering with the core database functionality, resulting in a more intuitive and simplified user experience. As a result, such changes are not only relevant, but also strategically linked with the general design philosophy of our system.

**Performance Optimization of Scalability**

Our database system is built to scale effectively as data volumes grow and more users utilize the system. A scalable system like this provides a solid foundation for performance improvement. Our system can respond to a variety of optimization strategies, including query optimization, database schema refinement, and hardware resource modifications. Because these improvements are directly tied to system performance, their implementation is not only appropriate but also necessary for sustaining an efficient and responsive system.

**Data Validation, Quality Assurance, and Security Measures**

Recognizing the importance of data integrity in our operations, we created our system to include capabilities like data validation and quality checks. Furthermore, our system will enable proactive security features such as encryption, access control, and intrusion detection systems in the future. These features are not just add-ons, but essential advancements that, if implemented, will protect our system from any dangers and assure the processing of correct and trustworthy data. As a result, such upgrades are not only relevant, but also fundamental to the system’s architecture and will be essential components of its operation. Also due to the flexibility of our system, it shouldn’t be too difficult to change the schema of the database and create or remove tables or columns if needed, as the database’s design is relatively simple and the implementation of these modifications isn’t that difficult.

**Mobile Integration and Telemedicine Support**

The interoperability of our database system is prioritized in its design, which is critical for connecting it with other healthcare applications, medical equipment, and other IT systems. This is especially important for the integration of mobile platforms and telemedicine assistance, both of which are key developments in modern healthcare. Because of the scalable and interoperable nature of our database system, such connections are not only viable but also very useful, increasing the system’s reach, accessibility, and usability.

**Iterative Development and User Feedback**

In line with our dedication to user-centered design, we value user input and iterative development. Regular user input and feedback integration guarantees that our system changes in accordance with user demands and preferences, ensuring its relevance and effectiveness. As a result, these changes are not only appropriate but also helpful in assuring the system’s usability and general performance.

Finally, the inherent flexibility and forward-thinking architecture of our database system allow it to adapt to future advancements while preserving its efficacy. The suggested enhancements are more than just additions; they are well-aligned with the system’s architecture and capabilities, boosting overall usefulness, efficiency, and flexibility to changing needs. We can guarantee that our database system remains a powerful tool for delivering high-quality, efficient, and patient-centered care by prioritizing these enhancements based on urgent requirements and long-term strategic vision.

# ***References***

CHRIS ODOGWU (2018) *What Is Database Maintenance and Why Is It Important?, Make Use Of.* Available at: [https://www.makeuseof.com/what-is-database-maintenance/#:~:text=Database%20maintenance%20is%20a%20set,new%20technology%20and%20security%20risks.](https://www.makeuseof.com/what-is-database-maintenance/%23:~:text=Database%20maintenance%20is%20a%20set,new%20technology%20and%20security%20risks.)

IBM Documentation (2020) *Database maintenance tasks, IBM Documentation.* Available at: [https://www.ibm.com/docs/en/product-master/12.0.0?topic=database-maintenance-tasks](https://www.ibm.com/docs/en/product-master/12.0.0?topic=database-maintenance-tasks%20)

Javatpoint (2019) *Difference between Backup and Recovery, javatpoint.* Available at: <https://www.javatpoint.com/difference-between-backup-and-recovery>

Leader, T. et al. (2020) *Database Backup and Recovery, Tutorials Point.* Available at: <https://www.tutorialspoint.com/Database-Backup-and-Recovery>

McMahon, M. (2022) *What is Database Maintenance? (with picture), Easy Tech Junkie.* Available at: <https://www.easytechjunkie.com/what-is-database-maintenance.htm>

NetApp (2020) *What Is Backup and Recovery?, NetApp.* Available at: [https://www.netapp.com/us/info/what-is-backup-and-recovery.aspx](https://www.netapp.com/us/info/what-is-backup-and-recovery.aspx%20)

Office Tools (2015) *Database Maintenance Explained - OfficeTools, Office Tools.* Available at: <https://www.officetools.com/knowledgebase/database-maintenance-explained/>

Oracle (2022) *Oracle Backup and Recovery, Oracle.* Available at: <https://docs.oracle.com/cd/B19306_01/server.102/b14220/backrec.htm>