



GROUP ASSIGNMENT
TECHNOLOGY PARK MALAYSIA
CT069-3-3-DBS
DATABASE SECURITY

HAND OUT DATE : 11st May 2021

HAND IN DATE : 6th August 2021

WEIGHTAGE : 60%

Group Member : CHAN JIA LE (TP049952)

SOONG HIM HOY (TP053242)

THONG KYN HUI (TP053489)

Intake Code : UC3F2011CS(DA)

ASSIGNMENT TITLE : WELLMEADOWS HOSPITAL

Lecturer Name : ABDALLAH S.M.ALNATSHA

Table of Contents

1.0 Wellmeadows Hospital Database	4
1.1 ERD & Logical Design.....	4
1.2 Business Requirement.....	6
1.3 Data Dictionary	7
2.0 Database Auditing Environment.....	13
2.1 Database.....	13
2.2 Audited Entities	15
2.3 People.....	16
2.4 Objectives	17
2.5 Procedures.....	17
3.0 Authorization Matrixes	18
3.1 Chan Jia Le (TP049952)	18
3.2 Soong Gim Hoy (TP053242).....	34
3.3 Thong Kyn Hui (TP053489).....	44
4.0 Authenticated User.....	58
4.1 Chan Jia Le (TP049952)	58
4.2 Soong Gim Hoy (TP053242).....	62
4.3 Thong Kyn Hui (TP053489).....	64
5.0 Logon Trigger	67
5.1 Chan Jia Le (TP049952)	67
5.2 Soong Gim Hoy (TP053242).....	68
5.3 Thong Kyn Hui (TP053489).....	69
6.0 Modification Trigger.....	70
6.1 Chan Jia Le (TP049952)	70
6.2 Soong Gim Hoy (TP053242).....	75
6.3 Thong Kyn Hui (TP053489).....	77

7.0 Encryption.....	81
7.1 Chan Jia Le (TP049952)	81
7.2 Soong Gim Hoy (TP053242)	86
7.3 Thong Kyn Hui (TP053489)	87
8.0 Backup and Restore	91
9.0 References	93
10.0 Appendix.....	94
10.1 Workload Matrix.....	94

1.0 Wellmeadows Hospital Database

1.1 ERD & Logical Design

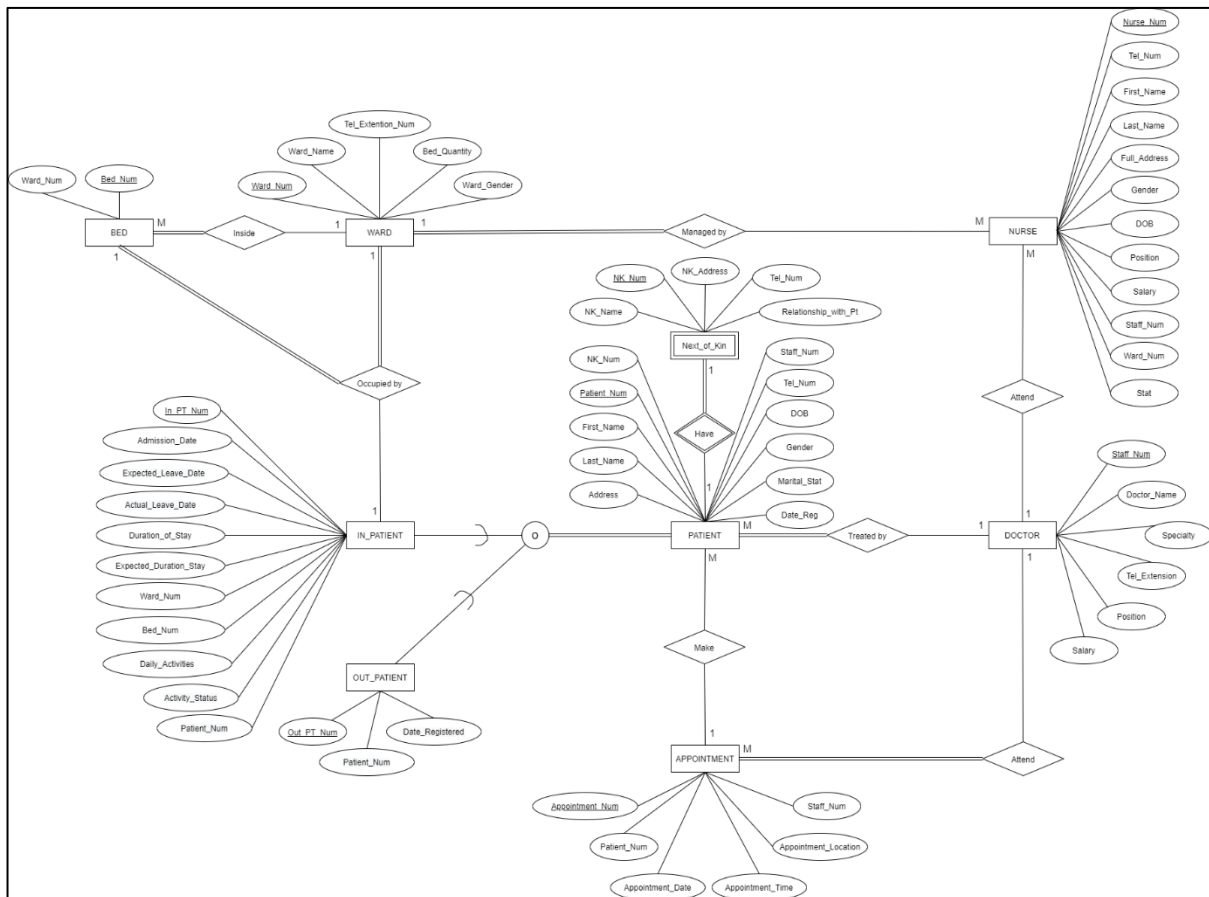


Figure 1.1.1 ERD Design of Wellmeadows Hospital Database

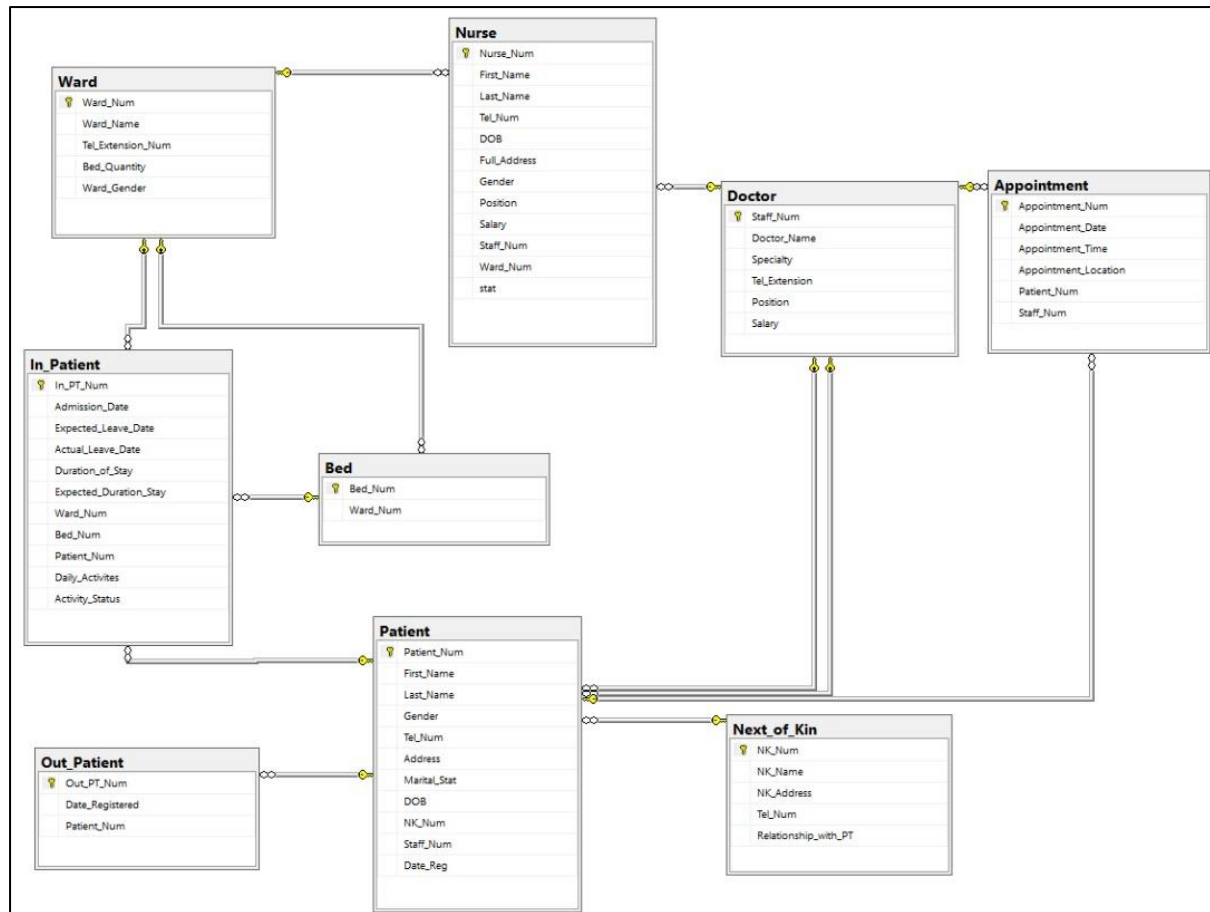


Figure 1.1.2 Logical Design of Wellmeadows Hospital Database

1.2 Business Requirement

1. 17 wards with a total of 240 beds
2. Ward uniquely identified by a number, name, total number of beds, gender and extension number.
3. Only head nurse can update daily activities of patients.
4. Nurse information include staff number, telephone number, first name, last name, address, gender, date of birth, position, stats, and salary.
5. Doctors are identified by their staff number.
6. Doctor information includes staff number, full name, their specialties and telephone extension, position and salary.
7. All salary is stated in monthly basis.
8. Many appointments can be assigned to a doctor, a doctor can be assigned to many appointments.
9. Appointment details include time, date, location and patient name.
10. Patients are identified by their unique patient number.
11. New patient must record their information including the name (first and last name), address, telephone number, date of birth, gender, marital status, date registered, and the details of the next of kin.
12. Details of the next of kin must be recorded.
13. One patient can only have one next of kin registered.
14. Details of the next of kin includes the (full name, relationship and telephone number).
15. Patients make appointment through phone or visit hospital.
16. Each appointment has a unique appointment number.
17. Patient's appointments are recorded, and it includes name and staff number of doctors performing the examination, date and time of examination and the examination room.
18. Patient is either recommended to attend the out-patient clinic or assigned to a bed in an appropriate ward, not both.
19. In-patients will have additional information such as admission date, expected leave date, actual leave date, duration of stay, expected duration of stay, ward number, bed number, daily activity and activity status.
20. Director doctor has the highest authorisation in the hospital system with access over all database tables.
21. All organization staff are assumed to have good ethics and in compliance with company rules.

1.3 Data Dictionary

Name:	Patient		
Primary Key:	Patient_Num		
Foreign Key:	Staff_Num, NK_Num		
Column Name	Description	Data Type	Size
Patient_Num	ID of patient	Int	3
First_Name	First name of patient	Varchar	20
Last_Name	Last name of patient	Varchar	20
Gender	Gender of patient	Varchar	10
Tel_Num	Telephone number of patients	Varchar	20
Address	Full address of patient	Varchar	100
Marital_Stat	Marital status of patient	Varchar	20
DOB	Date of birth of patient	Date	8
NK_Num	Patient's next of kin ID	Int	3
Staff_Num	ID of doctor	Int	3
Date_Reg	Date register of patient	Date	8

Name:	Next_of_Kin		
Primary Key:	NK_Num		
Foreign Key:	-		
Column Name	Description	Data Type	Size
NK_Num	ID of Next of Kin	Int	3
NK_Name	Name of Next of Kin	Varchar	25
NK_Address	Address of Next of Kin	Varchar	100
Tel_Num	Telephone number of Next of Kin	Varchar	20
Relationship_with_PT	Relationship of Next of Kin with patient	Varchar	35

Name:	Out_Patient		
Primary Key:	Out_PT_Num		
Foreign Key:	Patient_Num		
Column Name	Description	Data Type	Size
Out_PT_Num	ID of Out_Patient	Int	3
Date_Registered	Date registered of Out_Patient	Date	8
Patient_Num	ID of patient	Int	3

Name:	In_Patient		
Primary Key:	In_PT_Num		
Foreign Key:	Patient_Num, Ward_Num, Bed_Num		
Column Name	Description	Data Type	Size
In_PT_Num	ID of In_Patient	Int	3
Admission_Date	Admission date of In_Patient	Date	8
Expected_Leave_Date	Expected leave date of In_Patient	Date	8
Actual_Leave_Date	Actual leave date of In_Patient	Date	8
Duration_of_Stay	Actual stay duration of In_Patient	Int	5
Expected_Duration_Stay	Expected stay duration of In_Patient	Int	5
Ward_Num	Ward number of In_Patient	Int	3
Bed_Num	Bed number of In_Patient	Int	3
Patient_Num	ID of patient	Int	3
Daily_Activities	Daily activity of In_Patient	Varchar	200
Activity_Status	Activity status of In_Patient	Bit	1

Name:	Appointment		
Primary Key:	Appointment_Num		
Foreign Key:	Patient_Num, Staff_Num		
Column Name	Description	Data Type	Size
Appointment_Num	ID of appointment	Int	3
Appointment_Date	Date of appointment	Date	8
Appointment_Time	Time of appointment	Time	8
Appointment_Location	Location of appointment	Varchar	10
Patient_Num	ID of patient	Int	3
Staff_Num	ID of doctor	Int	3

Name:	Doctor		
Primary Key:	Staff_Num		
Foreign Key:	-		
Column Name	Description	Data Type	Size
Staff_Num	ID of doctor	Int	3
Doctor_Name	Name of doctor	Varchar	25
Specialty	Specialty of doctor	Varchar	30
Tel_Extension	Telephone extension number of doctor	Int	4
Position	Position of doctor	Varchar	15
Salary	Salary of doctor	Int	6

Name:	Nurse		
Primary Key:	Nurse_Num		
Foreign Key:	Staff_Num, Ward_Num		
Column Name	Description	Data Type	Size
Nurse_Num	ID of nurse	Int	3
First_Name	First name of nurse	Varchar	20
Last_Name	Last name of nurse	Varchar	20
Tel_Num	Telephone number of nurse	Varchar	20
DOB	Date of birth of nurse	Date	8
Full_Address	Full address of nurse	Varchar	100
Gender	Gender of nurse	Varchar	10
Position	Position of nurse	Varchar	20
Salary	Salary of nurse	Int	6
Staff_Num	ID of doctor	Int	3
Ward_Num	Ward number	Int	3
Stat	Status of nurse	Varchar	10

Name:	Ward		
Primary Key:	Ward_Num		
Foreign Key:	-		
Column Name	Description	Data Type	Size
Ward_Num	ID of ward	Int	3
Ward_Name	Name of ward	Varchar	20
Tel_Extension_Num	Telephone extension number of ward	Int	4
Bed_Quantity	Quantity of bed available in ward	Int	3
Ward_Gender	Gender of ward	Varchar	10

Name:	Bed		
Primary Key:	Bed_Num		
Foreign Key:	Ward_Num		
Column Name	Description	Data Type	Size
Bed_Num	ID of bed	Int	3
Ward_Num	ID of ward	Int	3

2.0 Database Auditing Environment

2.1 Database

Table Name	Descriptions	Key
Patient	The patient database records all the patient's data including the inpatient, outpatient and next of kin.	Primary Key: Patient_Num
In_Patient	The inpatient database records only inpatient data. Each inpatient is assigned with a unique inpatient identity.	Primary Key: In_PT_Num
Out_Patient	The outpatient database records only outpatient data. Each outpatient is assigned with a unique outpatient identity.	Primary Key: Out_PT_Num
Doctor	The doctor database records all the doctor's data. Each doctor is assigned with a unique staff identity.	Primary Key: Staff_Num
Nurse	The nurse database records all the nurse's data such as gender, address, contact number, birth date, name and position. The position is used to identify whether the role is head nurse or general nurse. Each nurse is assigned with a unique nurse identity.	Primary Key: Nurse_Num
Appointment	The appointment database records details of the appointment slot such as appointment date, appointment time, appointment location, patient attending the appointment and doctor in charge. Each appointment is assigned with a unique appointment number.	Primary Key: Appointment_Num
Next_of_Kin	The next of kin database records details of next of kin like name, address, telephone number and relationship with the patient. Each next of kin is assigned with a unique next of kin number.	Primary Key: NK_Num
Bed	The bed database records only the number of beds available in each ward.	Primary Key: Bed_Num

Ward	The ward database records ward data. The ward data includes ward name, number of beds, ward gender, telephone extension number and. Each ward is assigned with a unique ward number.	Primary Key: Ward_Num
------	--	--------------------------

2.2 Audited Entities

The audited entity for this audited environment includes the users of the database, the database itself, the process, and the documentation of the database. These are the included entities since these entities could affect the performance and integrity of the database.

1. Users

The authorities and activities of the users will be audited to ensure that users perform tasks in accordance with the business requirement and to ensure that no harm or irregular events are caused by the user. The authorization prevents the user to access confidential data while the audition documents attempt on insertion, modification, and deletion for auditing.

2. Database

The database is the main element of the whole auditing environment, different elements of the database will be audited like the design of the database is audited to make sure that data storage and process is according to the business requirement. Additionally, ensuring that the database follows the data privacy law and policy of the country. Furthermore, auditing the maintenance procedures to ensure that the database is maintained regularly and performing as expected.

3. Documentation

The documentation on the database will be audited as well using details like the database maintenance logbook. The documentation allows the auditor to keep track of the changes as well as tracing malicious activities that has been conducted in the database. Additionally, the authentication matrix documented will outline the granted and denied privileges for the database role user up until column level.

4. Data

The data that are repositied inside the database will be audited, details like data format, encryption process, modification, insertion, and deletion will be recorded. These elements are audited to ensure that the data format is in accordance to the business standard, ensuring that data encryption process has no potential weakness, and making sure that any tempering of data will be recorded or audited for the sake of ensuring that data is not tempered in a malicious manner.

2.3 People

1. Director Doctor

The Director Doctor has the highest authority over the database. His role is like a database administrator, he will have all access over details of the database. This role will also have the privilege to grant option to any users.

2. General Doctor

The General Doctor is one of the higher priority staffs in the Hospital which has access almost all the details of the database including appointment, patient, nurse, etc. Doctor will supervise numerous nurses and will be appointed to attend appointments and to take care of specific patients that requires their specialization. However, the doctor role will not have access to salary column of the doctors and no insertion for doctors are allowed.

3. Head Nurse

Head Nurse is the nurse that has a higher authority than general nurse, being able to insert, modify and delete the daily activities detail in the inpatient table. The head nurse will the authority to insert the daily task that is required for specific inpatients.

4. General Nurse

General Nurses are like Head Nurse but lacks a few authorities over the table. Whereby, the General Nurse will not have permission to grant privileges to other users, insertion on the Nurse table is not available and lastly, only updates on status of daily task are available for the General Nurse to temper. Head Nurse will have grant privileges and insertion to Nurse table.

5. Inpatient

Inpatient is a specialization of patient, where a stay in the hospital is needed, once the inpatient is admitted, details about expected leave date, date of entrance, etc. will be recorded inside the table. Inpatient will have no authority to view these details but only will be available to view the name of the patient inside the inpatient table.

6.Outpatient

Outpatient is another specialization of patient; a patient can be both inpatient and outpatient if necessary. The outpatient will have no access over the inpatient list of names and ward details, bed details, nurse in charge of wards will not be available for the outpatient. The outpatient

will only have the privilege to view appointment dates, doctor in charge of the appointment and the outpatient list.

2.4 Objectives

The objective of having an auditing environment for the database at Wellmeadows Hospital is to ensure data integrity, access of database is controlled so confidential data can only be viewed or managed by certain level of authorities, data is well secured for the avoidance of security breaches or weaknesses, data changes and elimination is recorded for future references, data changes and data structure changes are controlled and recorded to ensure database is in optimal condition. Lastly, to ensure there's backup strategy to avoid data loss due to unexpected events.

2.5 Procedures

1. Ensuring sensitive data like salary, age, etc. is encrypted for security purposes
2. Ensuring that access to database is properly restricted based on the level of authorization
3. Review and Evaluate procedures for creating user account
4. Ensuring that created users and login comply to password policies
5. Ensure that logon trigger is in place to limit concurrent session, login time etc.
6. Check for valid input for specific attributes like salary should be above 2,000
7. Ensuring that insertion, modification, or deletion of any records will be documented
8. Verify granted or revoked database permissions of the individuals appropriately based on the required level of authorization.
9. Review the granted database permission of the individual rather than groups or roles
10. Ensure that database permissions are not granted or revoked incorrectly
11. Ensuring that inserted data complies to the laws and regulations of Malaysia
12. Ensuring backup and recovery strategy is in place to avoid data loss

3.0 Authorization Matrixes

3.1 Chan Jia Le (TP049952)

Password Policy for Users

- Password of the User must contain Uppercase (A-Z), Lowercase (a-z), Numeral Digit (1-10)
- Password does not contain the account name of the specific user
- Password is at least 8 characters long

Authorization table (User View)

User View: In Patient (JiaLeChan, Jane)

	Select	Insert	Update	Delete	Grant
Patient	Y	N	N	N	N
Next_of_Kin	N	N	N	N	N
Appointment	Y	N	N	N	N
In Patient	N	N	N	N	N
Outpatient	N	N	N	N	N
Ward	Y	N	N	N	N
Bed	Y	N	N	N	N
Nurse	Y	N	N	N	N
Doctor	Y	N	N	N	N

Explanation: The inpatient is part of the patient list, hence will have selection permission for patient, appointment, ward, bed, nurse, and doctor table. However, not all of the columns inside the table are made available for the inpatient, details like staff id, nurse id, patient id, address, etc. is confidential data that should not be viewed by user or role outside of the organization, and the inpatient table is not accessible for the inpatient role due to the fact that inpatient table contains many confidential data like staff id, time of admission for patient, etc. that is not otherwise reasonable for an individual outside of the organization to view. Nevertheless, procedures are made to allow inpatient to have temporary access over the inpatient list and other complex queries that requires ids to perform.

User View: Head Nurse (TehSinBei, Rex)

	Select	Insert	Update	Delete	Grant
Patient	Y	Y	Y	N	Y
Next_of_Kin	N	N	N	N	N
Appointment	Y	Y	Y	Y	Y
In Patient	Y	Y	Y	N	Y
Outpatient	N	N	N	N	N
Ward	Y	N	Y	N	Y
Bed	Y	N	Y	N	Y
Nurse	Y	Y	Y	N	Y
Doctor	Y	N	N	N	N

Explanation: The head nurse role users will have more access over the table of the database. The authorization for each table is as shown in the matrix above. It is important to note however that the nurse will have no access over certain columns like doctor's salary as the authorization level of the nurse should not temper with the doctor's table. Other than that, head nurse role's insert privilege on ward and bed will not be available as the business rule has clearly stated that there are 240 beds and 17 wards, tempering with such details is not logical, hence, the privilege is not granted. The doctor table will be allowed to be queried by the nurse except for the salary column. Deletion on any details is not allowed except for the appointment. As the head nurse, the users will have the option to grant privilege to other users as it is safe to assume that if there are new nurses that joined under the head nurse's guidance or team, the grant option should be there to allow the head nurse to provide certain privilege as he or she sees fit.

Detailed user authorization matrix based on object and columns

Role: In Patient (JiaLeChan, Jane)

Object: Patient					
	Select	Insert	Update	Delete	Grant
Patient_Num	N	N	N	N	N
First_Name	Y	N	N	N	N
Last_Name	Y	N	N	N	N
Gender	Y	N	N	N	N
Tel_Num	N	N	N	N	N
Address	N	N	N	N	N
Marital_Stat	N	N	N	N	N
DOB	N	N	N	N	N
NK_Num	N	N	N	N	N
Staff_Num	N	N	N	N	N
Date_Reg	N	N	N	N	N
<p>Explanation: The Select privilege on the patient table for the inpatient is only the view of first name, last name, and gender. Other details like telephone number, marital status will not be allowed to be viewed as it is confidential details of other patients. Furthermore, inpatient is user that is outside of the organization, hence, primary keys like patient number, staff number should not be viewed. Due to the limitation of access over primary keys and foreign keys, procedures are created for temporary access over some queries.</p>					

Object: Next_of_Kin					
	Select	Insert	Update	Delete	Grant
NK_Num	N	N	N	N	N
NK_Name	N	N	N	N	N
NK_Address	N	N	N	N	N
Tel_Num	N	N	N	N	N
Relationship_with_PT	N	N	N	N	N
Explanation: The next of kin details contains confidential details that should not be disclosed to the public. However, procedure is created to temporarily allow inpatient to view the inpatient list along with the next of kin name and doctor in charge.					

Object: Appointment					
	Select	Insert	Update	Delete	Grant
Appointment_Num	Y	N	N	N	N
Appointment_Date	Y	N	N	N	N
Appointment_Time	Y	N	N	N	N
Appointment_Location	Y	N	N	N	N
Patient_Num	N	N	N	N	N
Staff_Num	N	N	N	N	N
Explanation: The level of privilege remained consistent as other tables whereby the patient number and staff number will not be accessible for the inpatient. However, procedure is made to temporarily allow inpatient to perform query over which patient and doctors are to attend the appointment.					

Object: In_Patient					
	Select	Insert	Update	Delete	Grant
In_PT_Num	N	N	N	N	N
Admission_Date	N	N	N	N	N
Expected_Leave_Date	N	N	N	N	N
Actual_Leave_Date	N	N	N	N	N
Duration_of_Stay	N	N	N	N	N
Expected_Duration_Stay	N	N	N	N	N
Ward_Num	N	N	N	N	N
Bed_Num	N	N	N	N	N
Patient_Num	N	N	N	N	N
Daily_Activities	N	N	N	N	N
Activity_Status	N	N	N	N	N
<p>Explanation: The inpatient role has no access over inpatient table as all the details are confidential information that should not be queried or shown to the public. However, because in patient should have the privilege or option to view details about the list of in patients in the hospital, so one procedure is made to perform a complex query to show the inpatient name, next of kin and doctor in charge of the patient.</p>					

Object: Out_Patient					
	Select	Insert	Update	Delete	Grant
Out_PT_Num	N	N	N	N	N
Date_Registered	N	N	N	N	N
Patient_Num	N	N	N	N	N
Explanation: The inpatient and outpatient are two separate entities; hence, no access privilege is given to the inpatient about the outpatient details. No procedures are also constructed to provide temporary access or query.					

Object: Ward					
	Select	Insert	Update	Delete	Grant
Ward_Num	Y	N	N	N	N
Ward_Name	Y	N	N	N	N
Tel_Extension	Y	N	N	N	N
Bed_Quantity	Y	N	N	N	N
Ward_Gender	Y	N	N	N	N
Explanation: All the columns will be granted with the select privilege for inpatient as these details does not pose any harm even if it is disclosed or leaked, and mostly like will bring benefits for the inpatient to understand which ward belongs to which.					

Object: Bed					
	Select	Insert	Update	Delete	Grant
Bed_Num	Y	N	N	N	N
Ward_Num	Y	N	N	N	N

Explanation: Like the Ward table, the bed and ward number will be accessible for the inpatient as it would bring more benefits than harm for disclosing these tables for queries

Object: Nurse

	Select	Insert	Update	Delete	Grant
Patient_Num	N	N	N	N	N
First_Name	Y	N	N	N	N
Last_Name	Y	N	N	N	N
Gender	Y	N	N	N	N
Tel_Num	N	N	N	N	N
Address	N	N	N	N	N
Marital_Stat	N	N	N	N	N
DOB	N	N	N	N	N
NK_Num	N	N	N	N	N
Staff_Num	N	N	N	N	N
Date_Reg	N	N	N	N	N

Explanation: The Select privilege on the patient table for the inpatient is only the view of first name, last name, and gender. Other details like telephone number, marital status will not be allowed to be viewed as it is confidential details of other patients. Furthermore, inpatient is user that is outside of the organization, hence, primary keys like patient number, staff number should not be viewed. Due to the limitation of access over primary keys and foreign keys, procedures are created for temporary access over some queries.

Object: Doctor

	Select	Insert	Update	Delete	Grant
Staff_Num	N	N	N	N	N
Doctor_Name	Y	N	N	N	N

Specialty	Y	N	N	N	N
Tel_Extension	Y	N	N	N	N
Position	N	N	N	N	N
Salary	N	N	N	N	N

Explanation: The Select privilege on the doctor table is only consisted of 3 columns including Doctor name, specialty, and telephone extension number. These details are completely free of risk even if it is disclosed. Details like staff number, position and salary are not authorized to be selected for the inpatient. Since staff number is not made accessible for inpatient complex queries like patient is under which doctor's care is impossible. Hence, procedures are created to allow temporary view over inpatient list with doctors that oversee the patient and next of kin name as stated in previous explanations.

Role: Head Nurse (TenSinBei, Rex)

Object: Patient					
	Select	Insert	Update	Delete	Grant
Patient_Num	Y	Y	Y	N	Y
First_Name	Y	Y	Y	N	Y
Last_Name	Y	Y	Y	N	Y
Gender	Y	Y	Y	N	Y
Tel_Num	Y	Y	Y	N	Y
Address	Y	Y	Y	N	Y
Marital_Stat	Y	Y	Y	N	Y
DOB	Y	Y	Y	N	Y
NK_Num	Y	Y	Y	N	Y
Staff_Num	Y	Y	Y	N	Y
Date_Reg	Y	Y	Y	N	Y

Explanation: As part of the member of the organization, the select, insert, and update privilege of the patient table are all made accessible for the head nurse role. Additionally, grant option is given to the head nurse as their potential cases where head nurse will need to provide some privileges for the general nurse.

Object: Next_of_Kin					
	Select	Insert	Update	Delete	Grant
NK_Num	Y	Y	Y	N	Y
NK_Name	Y	Y	Y	N	Y
NK_Address	Y	Y	Y	N	Y
Tel_Num	Y	Y	Y	N	Y
Relationship_with_PT	Y	Y	Y	N	Y
<p>Explanation: The next of kin details are all made available for the head nurse role, as head nurse is a role with higher authority. Additionally, delete privilege will not be provided as next of kin details are important details that should be retained and should not be deleted for future references or usage.</p>					

Object: Appointment					
	Select	Insert	Update	Delete	Grant
Appointment_Num	Y	Y	Y	Y	Y
Appointment_Date	Y	Y	Y	Y	Y
Appointment_Time	Y	Y	Y	Y	Y
Appointment_Location	Y	Y	Y	Y	Y
Patient_Num	Y	Y	Y	Y	Y
Staff_Num	Y	Y	Y	Y	Y

Explanation: The appointment table is the table that contains details about an appointment. Compared to other tables, the deletion privilege is given to the head nurse for this table as even if the data is disclosed or deleted, it won't have significant impact or usage anytime soon. Hence, deletion privilege is given to the head nurse users.

Object: In_Patient

	Select	Insert	Update	Delete	Grant
In_PT_Num	Y	Y	Y	N	Y
Admission_Date	Y	Y	Y	N	Y
Expected_Leave_Date	Y	Y	Y	N	Y
Actual_Leave_Date	Y	Y	Y	N	Y
Duration_of_Stay	Y	Y	Y	N	Y
Expected_Duration_Stay	Y	Y	Y	N	Y
Ward_Num	Y	Y	Y	N	Y
Bed_Num	Y	Y	Y	N	Y
Patient_Num	Y	Y	Y	N	Y
Daily_Activities	Y	Y	Y	N	Y
Activity_Status	Y	Y	Y	N	Y

Explanation: As explained in the business rule, the head nurse will have the privilege to insert, select and update the details about the daily activities. Hence, select, update, delete and grant option privileges are granted to the head nurse role and for the users. The grant option is given to the head nurse since in certain events, head nurse will need to pass certain privileges for trainee nurses or general nurses.

Object: Out_Patient					
	Select	Insert	Update	Delete	Grant
Out_PT_Num	N	N	N	N	N
Date_Registered	N	N	N	N	N
Patient_Num	N	N	N	N	N
<p>Explanation: Head nurse role will have no privileges over the outpatient table as the outpatient table is directly dealt by doctors and receptionist. Hence, head nurses will have no privilege to access over the outpatient table. However, procedure is created to allow head nurses to check whether is in outpatient based on given patient number (primary key).</p>					

Object: Ward					
	Select	Insert	Update	Delete	Grant
Ward_Num	Y	N	N	N	Y
Ward_Name	Y	N	Y	N	Y
Tel_Extension	Y	N	Y	N	Y
Bed_Quantity	Y	N	Y	N	Y
Ward_Gender	Y	N	Y	N	Y
<p>Explanation: The insert privilege is not given to the head nurse in this case as the business rule has clearly stated that the hospital has 17 wards, making insertion redundant and not reasonable. Additionally, update privilege on the ward number is not given to the head nurse. Other than that, update privilege is given as there will be cases where the ward might have changes in purposes or in bed quantity.</p>					

Object: Bed					
	Select	Insert	Update	Delete	Grant
Bed_Num	Y	N	Y	N	N
Ward_Num	Y	N	Y	N	N
Explanation: The privilege for select and update will be given to the head nurse role, as mentioned, there might be cases where the bed will have slight changes in purpose or delegated to other rooms especially in times of the pandemic.					

Object: Nurse					
	Select	Insert	Update	Delete	Grant
Patient_Num	Y	Y	Y	N	Y
First_Name	Y	Y	Y	N	Y
Last_Name	Y	Y	Y	N	Y
Gender	Y	Y	Y	N	Y
Tel_Num	Y	Y	Y	N	Y
Address	Y	Y	Y	N	Y
Marital_Stat	Y	Y	Y	N	Y
DOB	Y	Y	Y	N	Y
NK_Num	Y	Y	Y	N	Y
Staff_Num	Y	Y	Y	N	Y
Date_Reg	Y	Y	Y	N	Y
Explanation: As the head nurse of the hospital, it is expected that those head nurse will have business ethics, will perform non-miscellaneous tasks and most likely will take charge of					

many general nurses in the hospital. Hence, the select, insert, update and grant option privileges are given to the head nurse role.

Object: Doctor					
	Select	Insert	Update	Delete	Grant
Staff_Num	Y	N	N	N	N
Doctor_Name	Y	N	N	N	N
Specialty	Y	N	N	N	N
Tel_Extension	Y	N	N	N	N
Position	Y	N	N	N	N
Salary	N	N	N	N	N

Explanation: Compared to Inpatient, the select privilege for the doctors table will be consisted of more column except for the salary details as by normal circumstances, even if users are a head nurse, they have no right or authority to view salary of doctors. Furthermore, insertion, update, delete and grant options on the doctor table will not be given.

Creating Role

```
CREATE ROLE In_Patient
CREATE ROLE Head_Nurse
```

Figure 3.1.1 Code Snippet for Creating Roles

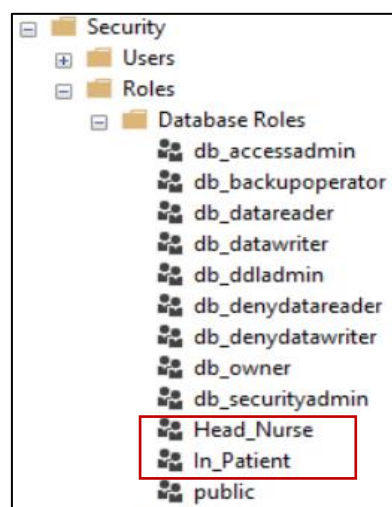


Figure 3.1.2 After Creating Roles

Granting privileges based on authorization matrix to role

```

/* Granting Permissions to In Patient Role */
GRANT SELECT (First_Name,Last_Name,Gender) ON Patient TO In_Patient
GRANT SELECT (Appointment_Num, Appointment_Date, Appointment_Time, Appointment_Location) ON Appointment TO In_Patient
GRANT SELECT ON Ward TO In_Patient
GRANT SELECT ON Bed TO In_Patient
GRANT SELECT (First_Name, Last_Name, Gender) ON Nurse TO In_Patient
GRANT SELECT (Doctor_Name, Specialty, Tel_Extension) ON Doctor TO In_Patient

```

Figure 3.1.3 Granting Designed Authorization to In Patient

```

/* Granting Permissions to In Head Nurse Role */
GRANT SELECT, INSERT, UPDATE ON Patient TO Head_Nurse WITH GRANT OPTION
GRANT SELECT, INSERT, UPDATE ON Next_of_Kin TO Head_Nurse WITH GRANT OPTION
GRANT SELECT, INSERT, UPDATE, DELETE ON Appointment TO Head_Nurse WITH GRANT OPTION
GRANT SELECT, INSERT, UPDATE ON In_Patient TO Head_Nurse WITH GRANT OPTION
GRANT SELECT, UPDATE (Ward_Name, Tel_Extension_Num, Bed_Quantity, Ward_Gender) ON Ward TO Head_Nurse WITH GRANT OPTION
GRANT SELECT, UPDATE ON Bed TO Head_Nurse WITH GRANT OPTION
GRANT SELECT, INSERT, UPDATE ON Nurse TO Head_Nurse WITH GRANT OPTION
GRANT SELECT (Staff_Num, Doctor_Name, Specialty, Tel_Extension, Position) ON Doctor TO Head_Nurse

```

Figure 3.1.4 Granting Designed Authorization to Head Nurse

Based on the authorization matrix that is designed, the privilege is granted into the role inpatient and head nurse.

Procedures for temporary privilege on queries

```

--TO CREATE PROCEDURE TO CHECK WARD AVAILABILITY
CREATE PROCEDURE check_bed
AS
SELECT Bed.Ward_Num, Ward.Ward_Name , COUNT (Bed.Bed_Num) AS AVAILABLE_BED FROM Bed
LEFT JOIN In_Patient
ON Bed.Bed_Num = In_Patient.Bed_Num
INNER JOIN Ward
ON Bed.Ward_Num = Ward.Ward_Num
WHERE In_Patient.Ward_Num IS NULL
GROUP BY Bed.Ward_Num, Ward.Ward_Name
GO

```

Figure 3.1.5 Creating Procedure check_bed for checking availability of bed in

The procedure check_bed is created for both the inpatient and head nurse to easily access the details over how many beds are available in each ward.

```

CREATE PROCEDURE check_pic_ward @Ward_Name nvarchar(30)
AS
SELECT Nurse.First_Name, Nurse.Last_Name, Nurse.Position, Ward.Ward_Name, Ward.Tel_Extension_Num from Nurse
INNER JOIN Ward
ON Nurse.Ward_Num = Ward.Ward_Num
WHERE Ward.Ward_Name = @Ward_Name
GO

```

Figure 3.1.6 Creating Procedure check_pic_ward for checking nurse in charge of specific ward

The procedure check_pic_ward is created for both inpatient and head nurse to search the person in charge of the ward based on the given ward name. This procedure is set up for easier query over who is taking care of the ward.

```
-- TO CREATE PROCEDURE TO CHECK APPOINTMENTS
CREATE PROCEDURE check_appointment
AS
SELECT Appointment.Appointment_Num, Patient.First_Name, Patient.Last_Name, Appointment_Date, Appointment.Appointment_Time,
INNER JOIN Doctor
ON Appointment.Staff_Num = Doctor.Staff_Num
INNER JOIN Patient
ON Appointment.Patient_Num = Patient.Patient_Num
GO
```

Figure 3.1.7 Creating Procedure check_appointment for checking appointments in hospital

The procedure check_appointment is designed for both inpatient and head nurse to check the appointment made along with details like appointment time and doctor in charge. The procedure is made to create a temporary access and query over appointment details for inpatient cause inpatient has no access over the patient number and staff number. Hence, making this kind of query impossible, so the procedure is created.

```
CREATE PROCEDURE check_in_pt_list
AS
SELECT Patient.First_Name, Patient.Last_Name, Ward.Ward_Name, Ward.Tel_Extension_Num, Doctor.Doctor_Name AS Doc_In_Charge, Next_of_Kin.NK_Name from Patient
INNER JOIN In_Patient
ON Patient.Patient_Num = In_Patient.Patient_Num
INNER JOIN Ward
ON In_Patient.Ward_Num = Ward.Ward_Num
INNER JOIN Doctor
ON Patient.Staff_Num = Doctor.Staff_Num
INNER JOIN Next_of_Kin
ON Patient.NK_Num = Next_of_Kin.NK_Num
GO
```

Figure 3.1.8 Creating Procedure check_in_pt_list to list out the in patients, next of kin name and doctor that oversee the patient

The procedure check_in_pt_list is designed for both inpatient and head nurse to check the list of inpatients along with details like doctor that is in charge, etc. The query has joined 4 tables which requires references of primary keys and foreign keys which is not available for the inpatient. Hence, this procedure is made to provide this complex query which is otherwise impossible for inpatients to make. This procedure is also used for the head nurse to conduct the query easily without many resources.

```
CREATE PROCEDURE search_patient_in_out_pt @Patient_Num int
AS
SELECT Patient.* FROM Patient
INNER JOIN Out_Patient
ON Patient.Patient_Num = Out_Patient.Patient_Num
WHERE Patient.Patient_Num = @Patient_Num
GO
```

Figure 3.1.9 Creating Procedure search_patient_in_out_pt to search patient in outpatient table

The search_patient_in_out_pt is designed specifically for the head nurse to grant temporary access over query for outpatient with a given patient number to check whether the inpatient is also an outpatient.

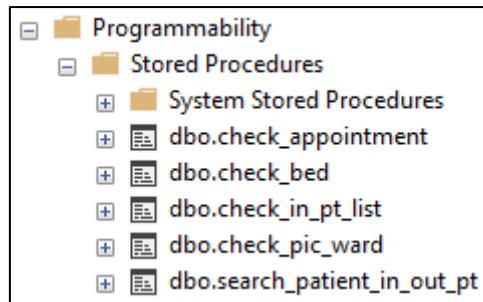


Figure 3.1.10 Stored Procedures after execution of each code

Upon completion of execution for the procedure codes above, the stored procedures will be stored inside the programmability, stored procedures section. However, no access has been granted to any user or role yet so, only the host can access it by now.

```

/* Grant Stored Procedure Execution to In patient and Head Nurse */
GRANT EXECUTE ON dbo.check_appointment TO In_Patient, Head_Nurse
GRANT EXECUTE ON dbo.check_bed TO In_Patient, Head_Nurse
GRANT EXECUTE ON dbo.check_in_pt_list TO In_Patient, Head_Nurse
GRANT EXECUTE ON dbo.check_pic_ward TO In_Patient, Head_Nurse
GRANT EXECUTE ON dbo.search_patient_in_out_pt TO Head_Nurse

```

Figure 3.1.11 Granting stored procedure access to each role

The privilege to access the stored procedure is as stated in the code above. All the procedures will also be made available for head nurse and only one is designed specifically for head nurse. The privilege is granted to head nurse since these queries in the stored procedures can save computational resources, hence, it is provided to the head nurses. However, most procedures are made especially for Inpatient as they have no access over the primary of all the tables and it could cause them some difficulty in queries. Hence, the procedures are made to mitigate those issues.

3.2 Soong Gim Hoy (TP053242)

Password Policy for Users

- Password of the User must contain Uppercase (A-Z), Lowercase (a-z), Numeral Digit (1-10)
- Password does not contain the account name of the specific user
- Password is at least 8 characters long

User View: General Nurse (Kali)

	Select	Insert	Update	Delete	Grant
Patient	Y	Y	Y	N	N
Next_of_Kin	N	N	N	N	N
Appointment	Y	Y	Y	N	N
In Patient	Y	Y	Y	N	N
Outpatient	N	N	N	N	N
Ward	Y	N	N	N	N
Bed	Y	N	N	N	N
Nurse	N	N	N	N	N
Doctor	Y	N	N	N	N
Explanation: The general nurse role can view selected details of the patient, appointment, in-patient, ward, bed, and doctor. These permissions will allow general nurses to handle patients well, along with their appointments. In-patients can be distributed to respective wards and beds. They can also figure out which doctor is in charge of the patient for easier notification.					

User View: Director Doctor (Baw)

	Select	Insert	Update	Delete	Grant
Patient	Y	Y	Y	Y	Y
Next_of_Kin	Y	Y	Y	Y	Y
Appointment	Y	Y	Y	Y	Y
In Patient	Y	Y	Y	Y	Y
Outpatient	Y	Y	Y	Y	Y
Ward	Y	Y	Y	Y	Y
Bed	Y	Y	Y	Y	Y
Nurse	Y	Y	Y	Y	Y
Doctor	Y	Y	Y	Y	Y
<p>Explanation: Director Doctor can be considered as the database administrator. He can perform any database activities on each table to allow a smooth process of managing the whole hospital. The wards and beds are fixed according to the business rules so they will not be tempered even by the director doctor.</p>					

Object View: General Nurse (Kali)

Object: Appointment					
Column	Select	Insert	Update	Delete	Grant
Appointment Num	Y	Y	Y	N	N
Appointment Date	Y	Y	Y	N	N
Appointment Time	Y	Y	Y	N	N
Appointment Location	Y	Y	Y	N	N
Patient Num	Y	Y	Y	N	N
Staff Num	Y	Y	Y	N	N
Explanation: The general nurse can view, insert and update appointment details to assist outpatients with their appointments. They can help outpatients to check and configure their appointments to their needs. They will not be able to delete the appointment however and cannot grant other users with access to the table.					

Object: Bed					
Column	Select	Insert	Update	Delete	Grant
Bed Num	Y	N	N	N	N
Ward Num	Y	N	N	N	N
Explanation: The bed configurations are fixed as according to the business rules. Hence, only viewing permission is provided for general nurse to know the ward and bed numbers of patients.					

Object: Doctor					
Column	Select	Insert	Update	Delete	Grant
Staff Num	Y	N	N	N	N
Doctor Name	Y	N	N	N	N
Specialty	Y	N	N	N	N
Tel Extension	Y	N	N	N	N
Position	N	N	N	N	N
Salary	N	N	N	N	N
Explanation: Doctor's staff number, name, specialty, and telephone extension can be viewed by general nurse so that they can find doctors with ease when questioned by patients. Position and Salary are more confidential details hence general nurse are not allowed to view them. No changing can be done to all values in the table.					

Object: In_Patient					
Column	Select	Insert	Update	Delete	Grant
In PT Num	Y	Y	Y	N	N
Admission Date	Y	Y	Y	N	N
Expected Leave Date	Y	Y	Y	N	N
Actual Leave Date	Y	Y	Y	N	N
Duration of Stay	Y	Y	Y	N	N
Expected Stay Duration	Y	Y	Y	N	N
Ward Num	Y	Y	Y	N	N
Bed Num	Y	Y	Y	N	N
Patient Num	Y	Y	Y	N	N
Daily Activities	Y	Y	Y	N	N
Activity Status	Y	Y	Y	N	N
Explanation: All the inpatient details can be viewed, inserted and updated. These permissions are essential for business daily activities of the hospital to be done. General nurse can update each patient's condition since they are in charge of patients daily activities.					

Object: Next of Kin					
Column	Select	Insert	Update	Delete	Grant
NK Num	Y	Y	Y	N	N
NK Name	Y	Y	Y	N	N
NK Address	Y	Y	Y	N	N
Tel Num	Y	Y	Y	N	N
Relationship with PT	Y	Y	Y	N	N
Explanation: Since general nurses work almost like a receptionist, the access and modification of the patient's next of kin details will help nurses undergo their daily activities better.					

Object: Nurse					
Column	Select	Insert	Update	Delete	Grant
Nurse Num	N	N	N	N	N
First Name	Y	N	N	N	N
Last Name	Y	N	N	N	N
Tel Num	N	N	N	N	N
DOB	N	N	N	N	N
Address	N	N	N	N	N
Gender	Y	N	N	N	N
Position	N	N	N	N	N
Salary	N	N	N	N	N
Staff Num	N	N	N	N	N
Ward Num	N	N	N	N	N
Explanation: General nurses can view each other's name and gender but are not allowed access to other sensitive details. There are also unable to perform modifications as that is the job of the head nurse.					

Object: Out_Patient					
Column	Select	Insert	Update	Delete	Grant
Out PT Num	N	N	N	N	N
Date Registered	N	N	N	N	N
Patient Num	N	N	N	N	N
Explanation: Outpatients details have no connection with general nurses, so it is not required for general nurses to have any permission for this table.					

Object: Patient					
Column	Select	Insert	Update	Delete	Grant
Patient Num	Y	Y	Y	N	N
First Name	Y	Y	Y	N	N
Last Name	Y	Y	Y	N	N
Gender	Y	Y	Y	N	N
Tel Num	Y	Y	Y	N	N
Address	Y	Y	Y	N	N
Marital Status	Y	Y	Y	N	N
DOB	Y	Y	Y	N	N
NK Num	Y	Y	Y	N	N
Staff Num	Y	Y	Y	N	N
Date Reg	Y	Y	Y	N	N
Explanation: Patient details can be viewed and modified by nurses as nurses oversee patients in general. They can help access and identify the patients and their staff in charge to lead them to the right person.					

Object: Ward					
Column	Select	Insert	Update	Delete	Grant
Ward Num	Y	N	N	N	N
Ward Name	Y	N	N	N	N
Tel Extension Num	Y	N	N	N	N
Bed Quantity	Y	N	N	N	N
Ward Gender	Y	N	N	N	N
Explanation: Ward details are not required to change according to the business rule. Hence, general nurses will only be required to view details of the wards.					

Object View: Director Doctor (Baw)

Explanation: All permissions are granted to the director doctor role since it is the highest management role in the whole hospital. The ward and beds, however, will not be modified as according to the business rule where their placements are fixed.

Object: Appointment					
Column	Select	Insert	Update	Delete	Grant
Appointment Num	Y	Y	Y	Y	Y
Appointment Date	Y	Y	Y	Y	Y
Appointment Time	Y	Y	Y	Y	Y
Appointment Location	Y	Y	Y	Y	Y
Patient Num	Y	Y	Y	Y	Y
Staff Num	Y	Y	Y	Y	Y

Object: Bed					
Column	Select	Insert	Update	Delete	Grant
Bed Num	Y	N	N	N	Y
Ward Num	Y	N	N	N	Y

Object: Doctor					
Column	Select	Insert	Update	Delete	Grant
Staff Num	Y	Y	Y	Y	Y
Doctor Name	Y	Y	Y	Y	Y
Specialty	Y	Y	Y	Y	Y
Tel Extension	Y	Y	Y	Y	Y
Position	Y	Y	Y	Y	Y
Salary	Y	Y	Y	Y	Y

Object: In_Patient					
Column	Select	Insert	Update	Delete	Grant
In PT Num	Y	Y	Y	Y	Y
Admission Date	Y	Y	Y	Y	Y
Expected Leave Date	Y	Y	Y	Y	Y
Actual Leave Date	Y	Y	Y	Y	Y
Duration of Stay	Y	Y	Y	Y	Y

Expected Stay Duration	Y	Y	Y	Y	Y
Ward Num	Y	Y	Y	Y	Y
Bed Num	Y	Y	Y	Y	Y
Patient Num	Y	Y	Y	Y	Y
Daily Activities	Y	Y	Y	Y	Y
Activity Status	Y	Y	Y	Y	Y

Object: Next of Kin					
Column	Select	Insert	Update	Delete	Grant
NK Num	Y	Y	Y	Y	Y
NK Name	Y	Y	Y	Y	Y
NK Address	Y	Y	Y	Y	Y
Tel Num	Y	Y	Y	Y	Y
Relationship with PT	Y	Y	Y	Y	Y

Object: Nurse					
Column	Select	Insert	Update	Delete	Grant
Nurse Num	Y	Y	Y	Y	Y
First Name	Y	Y	Y	Y	Y
Last Name	Y	Y	Y	Y	Y
Tel Num	Y	Y	Y	Y	Y
DOB	Y	Y	Y	Y	Y
Address	Y	Y	Y	Y	Y
Gender	Y	Y	Y	Y	Y
Position	Y	Y	Y	Y	Y
Salary	Y	Y	Y	Y	Y
Staff Num	Y	Y	Y	Y	Y
Ward Num	Y	Y	Y	Y	Y

Object: Out_Patient					
Column	Select	Insert	Update	Delete	Grant
Out PT Num	Y	Y	Y	Y	Y
Date Registered	Y	Y	Y	Y	Y
Patient Num	Y	Y	Y	Y	Y

Object: Patient					
Column	Select	Insert	Update	Delete	Grant
Patient Num	Y	Y	Y	Y	Y
First Name	Y	Y	Y	Y	Y
Last Name	Y	Y	Y	Y	Y
Gender	Y	Y	Y	Y	Y
Tel Num	Y	Y	Y	Y	Y
Address	Y	Y	Y	Y	Y
Marital Status	Y	Y	Y	Y	Y
DOB	Y	Y	Y	Y	Y
NK Num	Y	Y	Y	Y	Y
Staff Num	Y	Y	Y	Y	Y
Date Reg	Y	Y	Y	Y	Y

Object: Ward					
Column	Select	Insert	Update	Delete	Grant
Ward Num	Y	N	N	N	Y
Ward Name	Y	N	N	N	Y
Tel Extension Num	Y	N	N	N	Y
Bed Quantity	Y	N	N	N	Y
Ward Gender	Y	N	N	N	Y

Creating Roles

```
-- Create role General_Nurse and Director_Doctor
CREATE ROLE General_Nurse
CREATE ROLE Director_Doctor
```

Figure 3.2.1 Code for creating roles

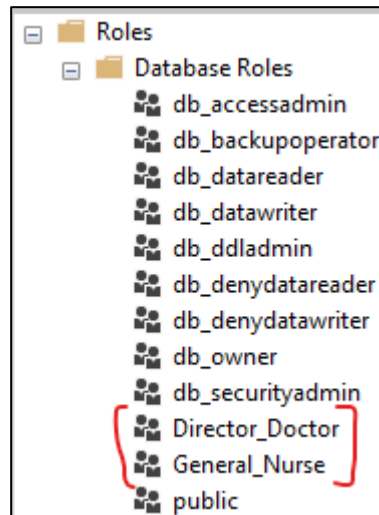


Figure 3.2.2 Database Roles created

Granting Permissions

```
-- Grant General_Nurse permission in server level
GRANT SELECT,INSERT, UPDATE ON Patient TO General_Nurse
GRANT SELECT,INSERT, UPDATE ON Next_of_Kin TO General_Nurse
GRANT SELECT,INSERT, UPDATE ON In_Patient TO General_Nurse
GRANT SELECT ON Appointment TO General_Nurse
GRANT SELECT ON Ward TO General_Nurse
GRANT SELECT ON Bed TO General_Nurse
GRANT SELECT (First_Name, Last_Name, Gender) ON Nurse TO General_Nurse
GRANT SELECT (Staff_Num, Doctor_Name, Specialty, Tel_Extension) ON Doctor TO General_Nurse

-- Grant Director_Doctor permission in server level
GRANT SELECT,INSERT,UPDATE,DELETE ON Patient TO Director_Doctor WITH GRANT OPTION
GRANT SELECT,INSERT,UPDATE,DELETE ON Next_of_Kin TO Director_Doctor WITH GRANT OPTION
GRANT SELECT,INSERT,UPDATE,DELETE ON Appointment TO Director_Doctor WITH GRANT OPTION
GRANT SELECT,INSERT,UPDATE,DELETE ON In_Patient TO Director_Doctor WITH GRANT OPTION
GRANT SELECT,INSERT,UPDATE,DELETE ON Out_Patient TO Director_Doctor WITH GRANT OPTION
GRANT SELECT,INSERT,UPDATE,DELETE ON Nurse TO Director_Doctor WITH GRANT OPTION
GRANT SELECT,INSERT,UPDATE,DELETE ON Doctor TO Director_Doctor WITH GRANT OPTION
GRANT SELECT ON Ward TO Director_Doctor WITH GRANT OPTION
GRANT SELECT ON Bed TO Director_Doctor WITH GRANT OPTION
```

Figure 3.2.3 Granting Permissions to roles according to authorization matrix

3.3 Thong Kyn Hui (TP053489)

Password Policy for Users

- Password of the User must contain Uppercase (A-Z), Lowercase (a-z), Numeral Digit (1-10)
- Password does not contain the account name of the specific user
- Password is at least 8 characters long

Authorization table

User View: Out-Patient (Sofia)

	Select	Insert	Update	Delete	Grant
Patient	Y	N	N	N	N
Next_of_Kin	N	N	N	N	N
Appointment	Y	N	N	N	N
In Patient	N	N	N	N	N
Outpatient	Y	N	N	N	N
Ward	Y	N	N	N	N
Bed	Y	N	N	N	N
Nurse	Y	N	N	N	N
Doctor	Y	N	N	N	N

Explanation: The outpatient is part of the patient list, hence will have selection permission for patient, appointment, outpatient, ward, bed, nurse, and doctor table. However, not all the columns inside the table are made available for the outpatient, details like staff id, nurse id, patient id, address, etc. is confidential data that should not be viewed by user or role outside of the organization.

Object View: Out-Patient (Sofia)

Object: Appointment					
Column	Select	Insert	Update	Delete	Grant
Appointment Num	Y	N	N	N	N
Appointment Date	Y	N	N	N	N
Appointment Time	Y	N	N	N	N
Appointment Location	Y	N	N	N	N
Patient Num	N	N	N	N	N
Staff Num	N	N	N	N	N
Explanation: The level of privilege remained consistent as other tables whereby the patient number and staff number will not be accessible for the outpatient.					

Object: Bed					
Column	Select	Insert	Update	Delete	Grant
Bed Num	Y	N	N	N	N
Ward Num	Y	N	N	N	N
Explanation: All the columns will be granted with the select privilege for outpatient as these details does not pose any harm even if it is disclosed or leaked, and mostly like will bring benefits for the outpatient to understand which ward belongs to which.					

Object: Doctor					
Column	Select	Insert	Update	Delete	Grant
Staff Num	N	N	N	N	N
Doctor Name	Y	N	N	N	N
Specialty	Y	N	N	N	N
Tel Extension	Y	N	N	N	N
Position	N	N	N	N	N
Salary	N	N	N	N	N
Explanation: The Select privilege on the doctor table is only consisted of 3 columns including Doctor name, specialty, and telephone extension number. These details are completely free of risk even if it is disclosed. Details like staff number, position and salary are not authorized to be selected for the outpatient.					

Object: In_Patient					
Column	Select	Insert	Update	Delete	Grant
In PT Num	N	N	N	N	N
Admission Date	N	N	N	N	N
Expected Leave Date	N	N	N	N	N
Actual Leave Date	N	N	N	N	N
Duration of Stay	N	N	N	N	N
Expected Stay Duration	N	N	N	N	N
Ward Num	N	N	N	N	N
Bed Num	N	N	N	N	N
Patient Num	N	N	N	N	N

Daily Activities	N	N	N	N	N
Activity Status	N	N	N	N	N
Explanation: The outpatient role has no access over outpatient table as all the details are confidential information that should not be queried or shown to the public.					

Object: Next of Kin					
Column	Select	Insert	Update	Delete	Grant
NK Num	N	N	N	N	N
NK Name	N	N	N	N	N
NK Address	N	N	N	N	N
Tel Num	N	N	N	N	N
Relationship with PT	N	N	N	N	N
Explanation: The next of kin details contains confidential details that should not be disclosed to the public.					

Object: Nurse					
Column	Select	Insert	Update	Delete	Grant
Nurse Num	N	N	N	N	N
First Name	Y	N	N	N	N
Last Name	Y	N	N	N	N
Tel Num	N	N	N	N	N
DOB	N	N	N	N	N
Address	N	N	N	N	N
Gender	Y	N	N	N	N
Position	N	N	N	N	N
Salary	N	N	N	N	N
Staff Num	N	N	N	N	N
Ward Num	N	N	N	N	N

Explanation: The Select privilege on the patient table for the inpatient is only the view of first name, last name, and gender. Other details like telephone number, marital status will not be allowed to be viewed as it is confidential details of other patients. Furthermore, inpatient is user that is outside of the organization, hence, primary keys like patient number, staff number should not be viewed.

Object: Out_Patient					
Column	Select	Insert	Update	Delete	Grant
Out PT Num	N	N	N	N	N
Date Registered	N	N	N	N	N
Patient Num	N	N	N	N	N
Explanation: The outpatient role has no access over outpatient table as all the details are confidential information that should not be queried or shown to the public.					

Object: Patient					
Column	Select	Insert	Update	Delete	Grant
Patient Num	N	N	N	N	N
First Name	Y	N	N	N	N
Last Name	Y	N	N	N	N
Gender	Y	N	N	N	N
Tel Num	N	N	N	N	N
Address	N	N	N	N	N
Marital Status	N	N	N	N	N
DOB	N	N	N	N	N
NK Num	N	N	N	N	N
Staff Num	N	N	N	N	N
Date Reg	N	N	N	N	N
Explanation: The Select privilege on the patient table for the outpatient is only the view of first name, last name, and gender. Other details like telephone number, marital status will not be allowed to be viewed as it is confidential details of other patients. Furthermore,					

outpatient is user that is outside of the organization, hence, primary keys like patient number, staff number should not be viewed.

Object: Ward					
Column	Select	Insert	Update	Delete	Grant
Ward Num	Y	N	N	N	N
Ward Name	Y	N	N	N	N
Tel Extension Num	Y	N	N	N	N
Bed Quantity	Y	N	N	N	N
Ward Gender	Y	N	N	N	N
Explanation: All the columns will be granted with the select privilege for outpatient as these details does not pose any harm even if it is disclosed or leaked, and mostly like will bring benefits for the outpatient to understand which ward belongs to which.					

User View: General Doctor (Ben)

	Select	Insert	Update	Delete	Grant
Patient	Y	N	N	N	N
Next_of_Kin	Y	N	N	N	N
Appointment	Y	Y	Y	Y	N
In Patient	Y	N	N	N	N
Outpatient	Y	N	N	N	N
Ward	Y	N	N	N	N
Bed	Y	N	N	N	N
Nurse	Y	N	N	N	N
Doctor	Y	N	N	N	N
<p>Explanation: The general doctor will have more access to the database compared to outpatient as the doctor are the staff of the organization. General doctor has the authority to select and view the entire database. However, general doctor has no access over certain sensitive personal details like patient telephone number, nurse, and other doctor's salary. The general doctor only has the authority to insert, update and delete on the appointment table. Other than the appointment table, general doctor cannot perform insert, update, delete or grant on other tables.</p>					

Object View: General Doctor (Ben)

Object: Appointment					
Column	Select	Insert	Update	Delete	Grant
Appointment Num	Y	Y	Y	Y	N
Appointment Date	Y	Y	Y	Y	N
Appointment Time	Y	Y	Y	Y	N
Appointment Location	Y	Y	Y	Y	N
Patient Num	N	N	N	N	N
Staff Num	N	N	N	N	N
Explanation: As the general doctor, he or she can perform select, insert, update, and delete any appointment details as they might need to schedule or reschedule an appointment slot with their patients. However, foreign keys like patient number and staff number are not required to be viewed by the general doctors.					

Object: Bed					
Column	Select	Insert	Update	Delete	Grant
Bed Num	Y	N	N	N	N
Ward Num	Y	N	N	N	N
Explanation: As the general doctor, he or she can only perform select and view the bed table. As a part of the organization, general doctors have the access to double check the bed details of the hospital. However, general doctors have no authority to perform any insert or modification to the bed table.					

Object: Doctor					
Column	Select	Insert	Update	Delete	Grant
Staff Num	Y	N	N	N	N
Doctor Name	Y	N	N	N	N
Specialty	Y	N	N	N	N
Tel Extension	Y	N	N	N	N
Position	Y	N	N	N	N
Salary	N	N	N	N	N
Explanation: As the general doctor, he or she can only perform select and view the doctor table except the salary column. Furthermore, insertion, update, delete and grant options on the doctor table will not be given to general doctors.					

Object: In_Patient					
Column	Select	Insert	Update	Delete	Grant
In PT Num	Y	N	N	N	N
Admission Date	Y	N	N	N	N
Expected Leave Date	Y	N	N	N	N
Actual Leave Date	Y	N	N	N	N
Duration of Stay	Y	N	N	N	N
Expected Stay Duration	Y	N	N	N	N
Ward Num	Y	N	N	N	N
Bed Num	Y	N	N	N	N
Patient Num	Y	N	N	N	N

Daily Activities	Y	N	N	N	N
Activity Status	Y	N	N	N	N
Explanation: As the general doctor, he or she can only select and view the patient record. Any modification or delete is not allowed or required to be performed by the general doctor itself. The modification can only be done by the nurses.					

Object: Next of Kin					
Column	Select	Insert	Update	Delete	Grant
NK Num	Y	N	N	N	N
NK Name	Y	N	N	N	N
NK Address	Y	N	N	N	N
Tel Num	Y	N	N	N	N
Relationship with PT	Y	N	N	N	N
Explanation: As the general doctor, he or she can only select and view the patient record. Any modification or delete is not allowed or required to be performed by the general doctor itself. The modification can only be done by the nurses.					

Object: Nurse					
Column	Select	Insert	Update	Delete	Grant
Nurse Num	Y	N	N	N	N
First Name	Y	N	N	N	N
Last Name	Y	N	N	N	N
Tel Num	N	N	N	N	N
DOB	N	N	N	N	N
Address	N	N	N	N	N
Gender	Y	N	N	N	N
Position	Y	N	N	N	N
Salary	N	N	N	N	N
Staff Num	N	N	N	N	N

Ward Num	N	N	N	N	N
Explanation: As the general doctor, he or she can only perform select and view the nurse num, first name, last name, gender, and position of the nurse. They cannot view other personal details such as the telephone number, address, DOB, and address. Furthermore, insertion, update, delete and grant options on the nurse table will not be given to general doctors.					

Object: Out_Patient					
Column	Select	Insert	Update	Delete	Grant
Out PT Num	Y	N	N	N	N
Date Registered	Y	N	N	N	N
Patient Num	Y	N	N	N	N
Explanation: As the general doctor, he or she can only select and view the patient record. Any modification or delete is not allowed or required to be performed by the general doctor itself. The modification can only be done by the nurses.					

Object: Patient					
Column	Select	Insert	Update	Delete	Grant
Patient Num	Y	N	N	N	N
First Name	Y	N	N	N	N
Last Name	Y	N	N	N	N
Gender	Y	N	N	N	N
Tel Num	Y	N	N	N	N
Address	Y	N	N	N	N
Marital Status	Y	N	N	N	N
DOB	Y	N	N	N	N
NK Num	Y	N	N	N	N
Staff Num	Y	N	N	N	N
Date Reg	Y	N	N	N	N

Explanation: As the general doctor, he or she can only select and view the patient record. Any modification or delete is not allowed or required to be performed by the general doctor itself. The modification can only be done by the nurses.

Object: Ward					
Column	Select	Insert	Update	Delete	Grant
Ward Num	Y	N	N	N	N
Ward Name	Y	N	N	N	N
Tel Extension Num	Y	N	N	N	N
Bed Quantity	Y	N	N	N	N
Ward Gender	Y	N	N	N	N
<p>Explanation: As the general doctor, he or she can only perform select and view the ward table. As a part of the organization, general doctors have the access to double check the ward details of the hospital. However, general doctors have no authority to perform any insert or modification to the ward table.</p>					

Creating Roles

```
-- Create role Out_Patient and General_Doctor
CREATE ROLE Out_Patient
CREATE ROLE General_Doctor
```

Figure 3.3.1 Code Snippet for Creating

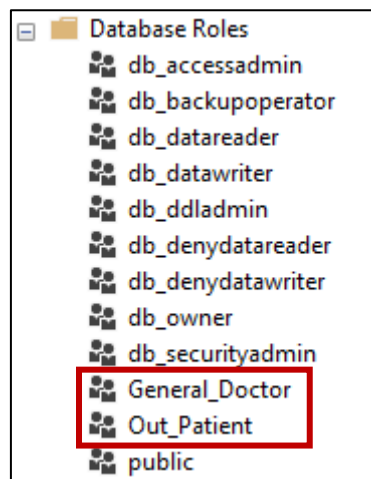


Figure 3.3.2 Databases Roles Created

Granting privileges based on authorization matrix to role

```
-- Grant Out_Patient permission in server level
GRANT SELECT (First_Name, Last_Name, Gender) ON Patient TO Out_Patient
GRANT SELECT (Appointment_Num, Appointment_Date, Appointment_Time, Appointment_Location) ON Appointment TO Out_Patient
GRANT SELECT ON Ward TO Out_Patient
GRANT SELECT ON Bed TO Out_Patient
GRANT SELECT (First_Name, Last_Name, Gender) ON Nurse TO Out_Patient
GRANT SELECT (Doctor_Name, Specialty, Tel_Extension) ON Doctor TO Out_Patient
```

Figure 3.3.3 Granting Authorization to Out-Patient Role

```
-- Grant General_Doctor permission in server level
GRANT SELECT, INSERT, UPDATE, DELETE ON Appointment TO General_Doctor
GRANT SELECT ON Bed TO General_Doctor
GRANT SELECT (Staff_Num, Doctor_Name, Specialty, Tel_Extension, Position) ON Doctor TO General_Doctor
GRANT SELECT ON In_Patient TO General_Doctor
GRANT SELECT ON Next_of_Kin TO General_Doctor
GRANT SELECT (Nurse_Num, First_Name, Last_Name, Gender, Position) ON Nurse TO General_Doctor
GRANT SELECT ON Out_Patient TO General_Doctor
GRANT SELECT ON Patient TO General_Doctor
GRANT SELECT ON Ward TO General_Doctor
```

Figure 3.3.4 Granting Authorization to General Doctor Role

Procedures for temporary privilege on queries

```
-- TO CREATE PROCEDURE TO CHECK APPOINTMENTS
Go
CREATE PROCEDURE check_appointment
AS
SELECT Appointment.Appointment_Num, Patient.First_Name, Patient.Last_Name, Appointment_Date,
Appointment.Appointment_Time, Appointment.Appointment_Location, Doctor.Doctor_Name FROM Appointment
INNER JOIN Doctor
ON Appointment.Staff_Num = Doctor.Staff_Num
INNER JOIN Patient
ON Appointment.Patient_Num = Patient.Patient_Num
GO
```

Figure 3.3.5 Creating Procedure check_appointment for checking appointments in hospital

The procedure check_appointment is designed to check the appointment made along with details like appointment time and doctor in charge. The procedure is made to create a temporary access and query over appointment details for inpatient cause inpatient has no access over the patient number and staff number.

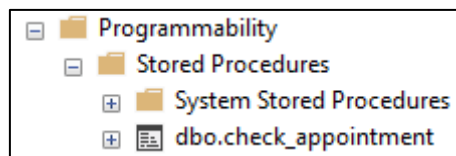


Figure 3.3.6 Stored Procedures after execution

```
-- Grant Stored Procedure Execution to Out_Patient and General_Doctor
GRANT EXECUTE ON dbo.check_appointment TO Out_Patient, General_Doctor
```

Figure 3.3.7 Granting stored procedure access to each role

The privilege to access the stored procedure is as stated in the code above. The stored procedure is granted to Out-Patient and the General Doctor role as they have no access over the primary of all the tables and it could cause them some difficulty in queries. Hence, the procedures are made to mitigate those issues.

4.0 Authenticated User

4.1 Chan Jia Le (TP049952)

In_patient (JiaLeChan, Jane)

```
/* In Patient User */
CREATE LOGIN JiaLeChan WITH PASSWORD = 'J@red3036', CHECK_EXPIRATION = ON, CHECK_POLICY = ON;
CREATE USER JiaLeChan FROM LOGIN JiaLeChan
CREATE LOGIN Jane WITH PASSWORD = 'J@ne8988', CHECK_EXPIRATION = ON, CHECK_POLICY = ON;
CREATE USER Jane FROM LOGIN Jane
```

Figure 4.1.1 Creating server login and creating database user authentication for Inpatient

Head_Nurse (TehSinBei, Rex)

```
/* Head Nurse User */
CREATE LOGIN TehSinBei WITH PASSWORD = 'T$h3036', CHECK_EXPIRATION = ON, CHECK_POLICY = ON;
CREATE USER TehSinBei FROM LOGIN TehSinBei
CREATE LOGIN Rex WITH PASSWORD = 'R$5x1256', CHECK_EXPIRATION = ON, CHECK_POLICY = ON;
CREATE USER Rex FROM LOGIN Rex
```

Figure 4.1.2 Creating server login and creating database user authentication for Head

The create login syntax is used to create a server level login for accessing the server. Upon completion of that, when connected to the specified database perform the create user syntax for creating a user under the database for database access under the server level login. The password policies in SQL are all enabled including checking expiration and checking the password policy which enforces password strength that are stated in the above password policy section.

```
ALTER ROLE In_Patient Add MEMBER JiaLeChan
ALTER ROLE In_Patient Add MEMBER Jane

ALTER ROLE Head_Nurse Add MEMBER TehSinBei
ALTER ROLE Head_Nurse Add MEMBER Rex
```

Figure 4.1.3 Assigning role to the specific users

The SQL code above adds the users into the specific role. Upon completion of the execution, each user will be categorized under the specified role along with the granted privileges that are created during the section grant privilege above based on the designed authorization matrix.

Sample of Login and Access Restriction

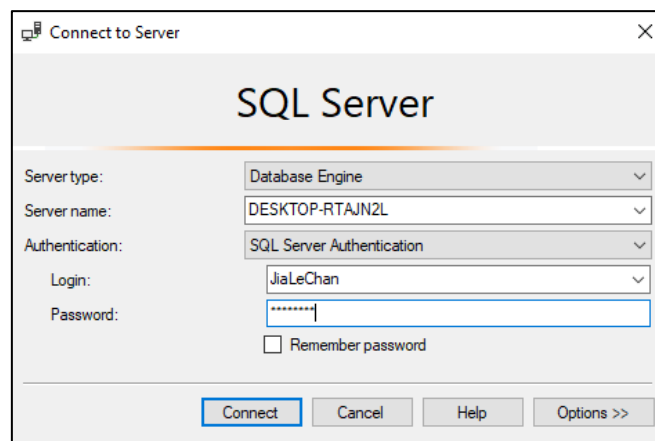


Figure 4.1.4 Logging in into the server

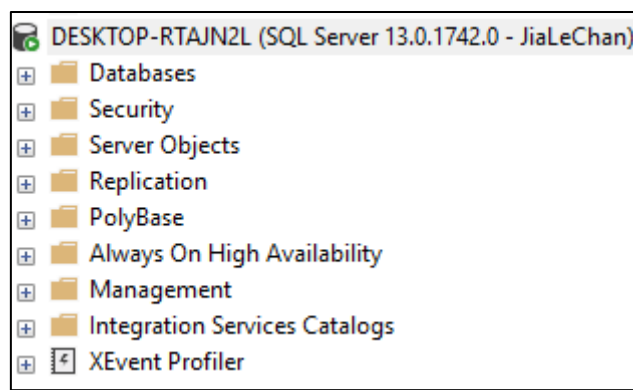


Figure 4.1.5 After successful login

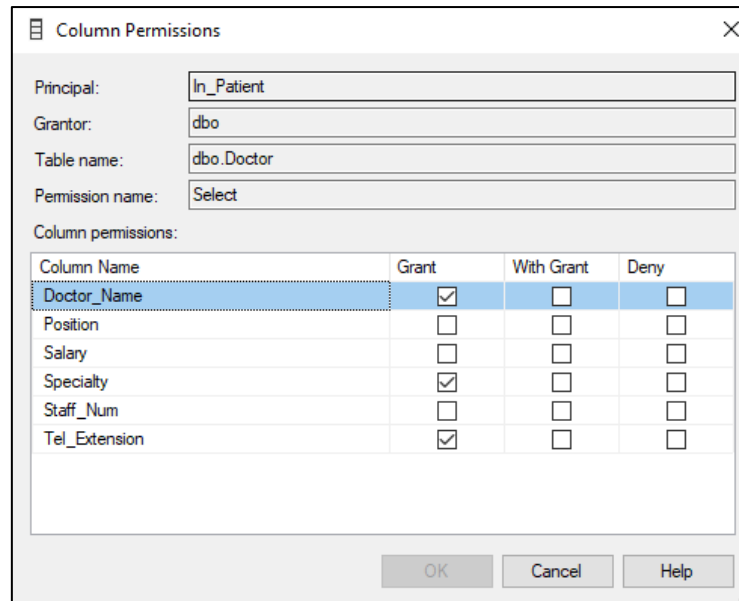


Figure 4.1.6 Column Permission of In_Patient (JiaLeChan)

Upon completion of logging in into the user JiaLeChan which is an inpatient. The principal of the user is under the In_Patient role. Thus, as reflected in the figure above where it shows the granted SELECT permission under the doctors table. Other tables and columns will also be following the privileges that are granted for the inpatient role.

```
SELECT * FROM Doctor
```

Figure 4.1.7 Sample Query on Doctor Table of In_Patient (JiaLeChan)

```
Msg 230, Level 14, State 1, Line 1
The SELECT permission was denied on the column 'Staff_Num' of the object 'Doctor', database 'DBS(Ravenhearst)', schema 'dbo'
Msg 230, Level 14, State 1, Line 1
The SELECT permission was denied on the column 'Position' of the object 'Doctor', database 'DBS(Ravenhearst)', schema 'dbo'.
Msg 230, Level 14, State 1, Line 1
The SELECT permission was denied on the column 'Salary' of the object 'Doctor', database 'DBS(Ravenhearst)', schema 'dbo'.
```

Figure 4.1.8 Error message on query for Doctor Table of In_Patient (JiaLeChan)

Upon query of every element in the doctor's table, because the inpatient has no access or permission over the doctor's staff number, salary, and position. The SQL will prompt the error as shown in the figure above.

```
SELECT Doctor_Name, Specialty, Tel_Extension FROM Doctor
```

Figure 4.1.9 Sample Query on Doctor Table of In_Patient (JiaLeChan)

	Doctor_Name	Specialty	Tel_Extension
1	Helen Cho	Cardiologists	6601
2	Sumaiya Olson	Anesthesiologists	6602
3	Chanel Beech	Emergency Medicine Specialists	6603
4	Annette Feeney	Neurologists	6604
5	Maciej Castro	Ophthalmologists	6605
6	Brent Mclean	Pathologists	6606
7	Antoni Mueller	Pediatricians	6607

Figure 4.1.10 Sample Output on Doctor Table of In_Patient (JiaLeChan)

However, if select is performed under those granted columns, the query will be performed successfully as shown in the output of the figure above.

```
UPDATE Doctor
SET Doctor_Name = 'Helen'
WHERE Doctor_Name = 'Helen Cho'
```

Figure 4.1.11 Sample Update on Doctor Table of In_Patient (JiaLeChan)

```
Msg 229, Level 14, State 5, Line 1
The UPDATE permission was denied on the object 'Doctor', database 'DBS(Ravenhearst)', schema 'dbo'.
```

Figure 4.1.12 Error message on update for Doctor Table of In_Patient (JiaLeChan)

Like the result in the error query. Since inpatient has no granted permission to conduct any update for the doctor table, there will be error stating that UPDATE permission is denied. The same error will occur if inpatient role chooses to perform DELETE in the other tables as there is no granted permission.

Summary

All in all, the authorization matrix is first designed by considering which authorization should be provided into which role to prevent any unauthorized access as mentioned in the audit environment. Upon completion of designing the authorization matrix, roles are created and granted with privileges that are designated in the authorization matrixes. However, considering where some queries might be difficult for certain users like Inpatient where there is no access over primary keys and foreign keys, procedures are made to eliminate such issue to make it more true to life scenario. Once the permissions and procedures are granted to the specific roles, 2 logins and users are created for each role. Afterwards, the developer has tested the login of JiaLeChan (In_Patient) to check the granted permissions and role of the user. Upon completion of that, a query and update are performed to simulate what will occur if user has no access or granted privilege to do so and the results are recorded and documented as shown above.

4.2 Soong Gim Hoy (TP053242)

Creating User and Login

```
-- Create new user
CREATE LOGIN Kali WITH PASSWORD = 'Kali@123', CHECK_EXPIRATION = ON, CHECK_POLICY = ON;
CREATE USER Kali FROM LOGIN Kali
CREATE LOGIN Baw WITH PASSWORD = 'Baw@123', CHECK_EXPIRATION = ON, CHECK_POLICY = ON;
CREATE USER Baw FROM LOGIN Baw
```

Figure 4.2.1 Creating new logins and users for Kali and Baw

Logins are created for Kali and Baw. They are identified as users of the database Wellmeadow. The password given is according to the password policy. They will be able to login into the database server for access to required information in the database.

Assigning Roles

```
-- Assign General_Nurse Role to User
ALTER ROLE General_Nurse Add MEMBER Kali

-- Assign Director_Doctor Role to User
ALTER ROLE Director_Doctor Add MEMBER Baw
```

Figure 4.2.2 Assigning roles to users

The role of general nurse is assigned to Kali and the role of director doctor is assigned to Baw. Baw will now have all privileges of the director doctor role which is access to all tables in the database and able to modify data according to the authorization matrix. Kali will have access to general nurse role available data and are not provided the ability to delete tables or data.

Sample Login

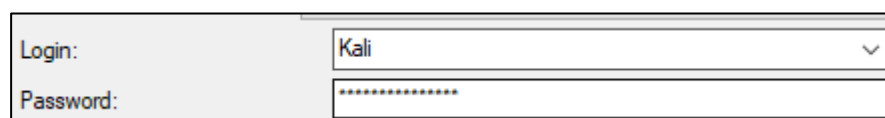


Figure 4.2.3 Login for Kali

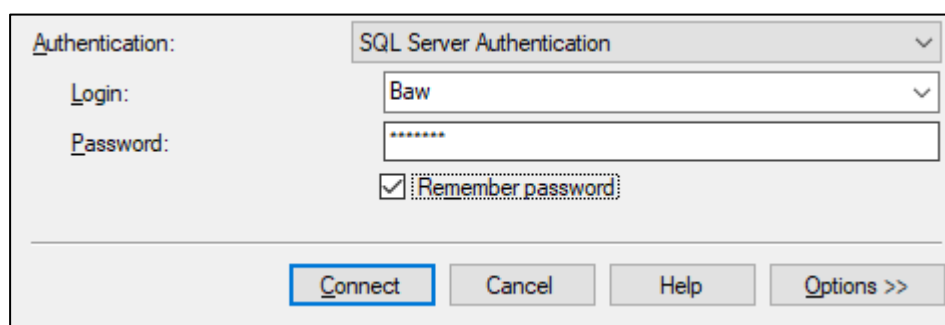


Figure 4.2.4 Login for Baw

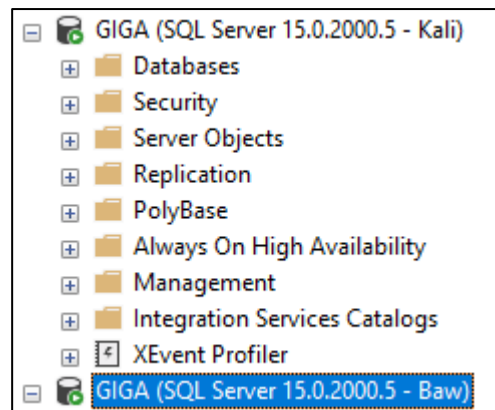


Figure 4.2.5 Successful Login for both Kali and Baw

Figure 4.2.3 and Figure 4.2.4 above shows the login for Kali and Baw. They are given authentication and able to access the database as seen in Figure 4.2.5.

4.3 Thong Kyn Hui (TP053489)

Create server authenticated user & login

Out_Patient (Sofia)

```
-- Create new user
CREATE LOGIN Sofia WITH PASSWORD = 'Sofi@123', CHECK_EXPIRATION = ON, CHECK_POLICY = ON;
CREATE USER Sofia FROM LOGIN Sofia
```

Figure 4.3.1 Creating server login and creating database user authentication for Out_Patient

General_Doctor (Ben)

```
-- Create new user
CREATE LOGIN Ben WITH PASSWORD = 'Ben@123', CHECK_EXPIRATION = ON, CHECK_POLICY = ON;
CREATE USER Ben FROM LOGIN Ben
```

Figure 4.3.2 Creating server login and creating database user authentication for General_Doctor

The create login syntax is used to create a server level login for accessing the server. Upon completion of that, when connected to the specified databased perform the create user syntax for creating a user under the database for database access under the server level login. The password policies in SQL are all enabled including checking expiration and checking the password policy which enforces password strength that are stated in the above password policy section.

```
-- Assign Out_Patient Role to User
ALTER ROLE Out_Patient Add MEMBER Sofia

-- Assign General_Doctor Role to User
ALTER ROLE General_Doctor Add MEMBER Ben
```

Figure 4.3.3 Assigning role to the specific users

The SQL code above adds the users into the specific role. Upon completion of the execution, each user will be categorized under the specified role along with the granted privileges that are created during the section grant privilege above based on the designed authorization matrix.

Sample of Login and Access Restriction

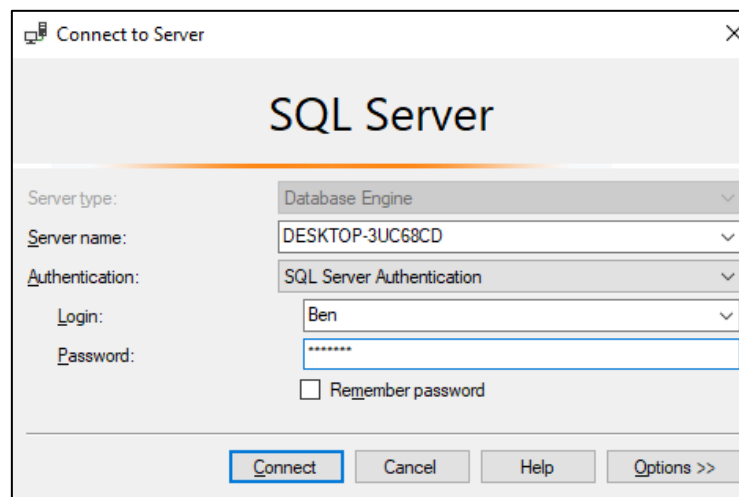


Figure 4.3.4 Logging in into the server

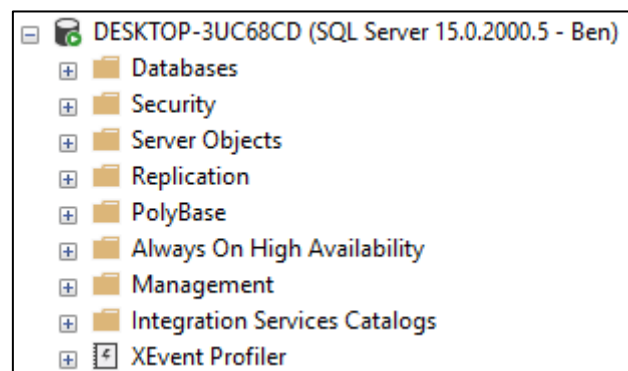


Figure 4.3.5 After successful login

Upon query of every element in the nurse's table as general doctor (Ben), The SQL will prompt the error as shown in the figure below. This is because the general doctor (Ben) has no access or permission over the nurse's telephone number, DOB, full address, salary etc.

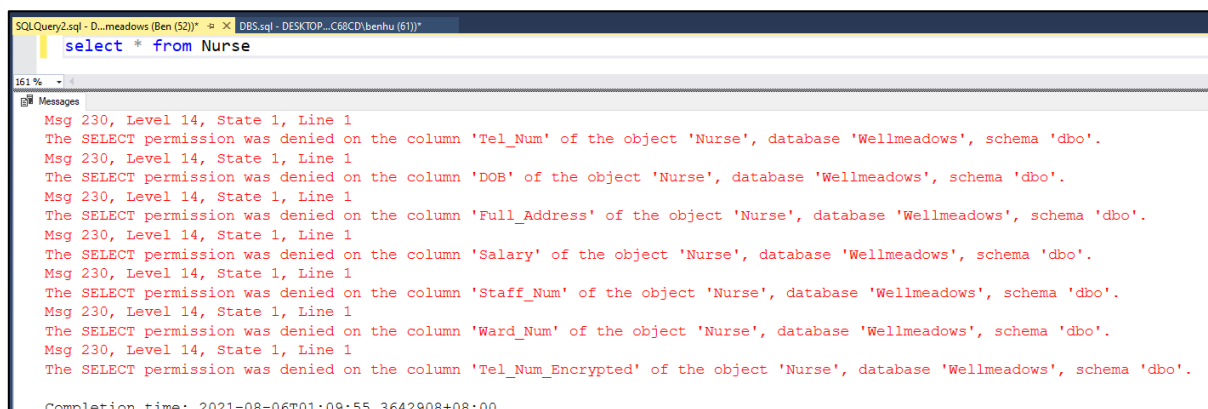
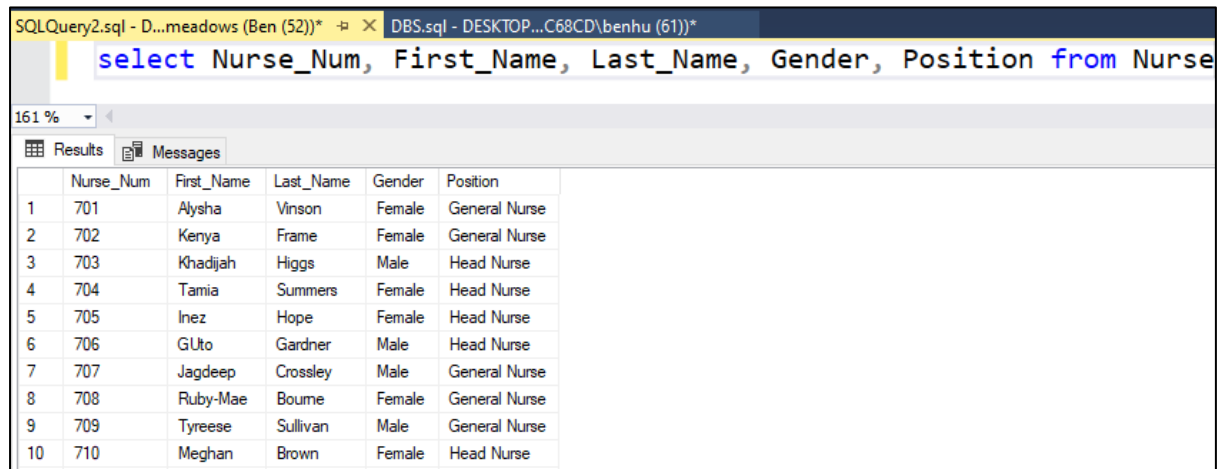


Figure 4.3.6 Error message on query for Nurse Table as General Doctor (Ben)

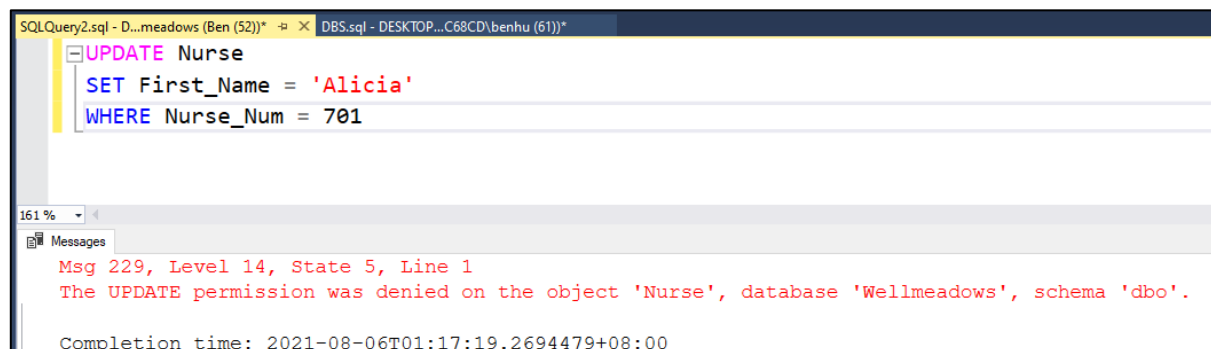
However, if select is performed under those granted columns, the query will be performed successfully as shown in the output of the figure below.



	Nurse_Num	First_Name	Last_Name	Gender	Position
1	701	Alysha	Vinson	Female	General Nurse
2	702	Kenya	Frame	Female	General Nurse
3	703	Khadijah	Higgs	Male	Head Nurse
4	704	Tamia	Summers	Female	Head Nurse
5	705	Inez	Hope	Female	Head Nurse
6	706	GUto	Gardner	Male	Head Nurse
7	707	Jagdeep	Crossley	Male	General Nurse
8	708	Ruby-Mae	Boume	Female	General Nurse
9	709	Tyreese	Sullivan	Male	General Nurse
10	710	Meghan	Brown	Female	Head Nurse

Figure 4.3.7 Sample Query and Output on Nurse Table as General Doctor (Ben)

Since general doctor has no granted permission to conduct any update for the nurse table, there will be error stating that UPDATE permission is denied. The same error will occur if general doctor role chooses to perform DELETE in the other tables as there is no granted permission.



```

UPDATE Nurse
SET First_Name = 'Alicia'
WHERE Nurse_Num = 701
  
```

Msg 229, Level 14, State 5, Line 1
 The UPDATE permission was denied on the object 'Nurse', database 'Wellmeadows', schema 'dbo'.

Completion time: 2021-08-06T01:17:19.2694479+08:00

Figure 4.3.8 Sample UPDATE Query and Output on Nurse Table as General Doctor (Ben)

5.0 Logon Trigger

5.1 Chan Jia Le (TP049952)

```

/* LIMITING USER TO ONLY HAVE 2 QUERY SESSIONS ON THE SERVER Limiting */
CREATE TRIGGER Limit_User_Connection_TR ON ALL SERVER
FOR LOGON
AS
BEGIN
    DECLARE @login NVARCHAR(100);
    SET @login = ORIGINAL_LOGIN();
    IF
    (
        SELECT COUNT(*)
        FROM sys.dm_exec_sessions
        WHERE is_user_process = 1
            AND original_login_name = @login
            AND program_name = 'Microsoft SQL Server Management Studio - Query'
    ) > 2
    BEGIN
        PRINT 'You are Not Allowed to Have More Than 2 SQL Query Connection - Login by' + @login + 'Failed';
        ROLLBACK;
    END;
END;

```

Figure 5.1.1 Logon Trigger Code Snippet for Limiting Query Sections

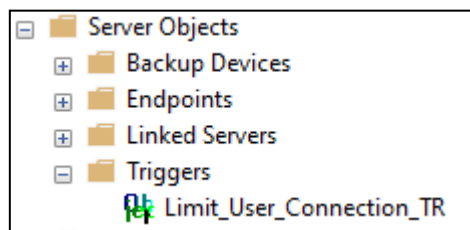


Figure 5.1.2 Logon Trigger Created in Server

The created logon trigger is used to limit one user's query sessions on the server. The system will check the executed query session as per that user login to ensure that query sessions are not created more than 2 sessions.

	session_id	login_time	host_name	program_name	host_process_id	client_version	client_interface
1	54	2021-08-05 19:04:42.073	DESKTOP-RTAJN2L	Microsoft SQL Server Management Studio - Query	19584	7	.Net SqlClient
2	55	2021-08-05 19:44:22.360	DESKTOP-RTAJN2L	Microsoft SQL Server Management Studio - Query	19584	7	.Net SqlClient

Figure 5.1.3 Maximum Query Session Reached

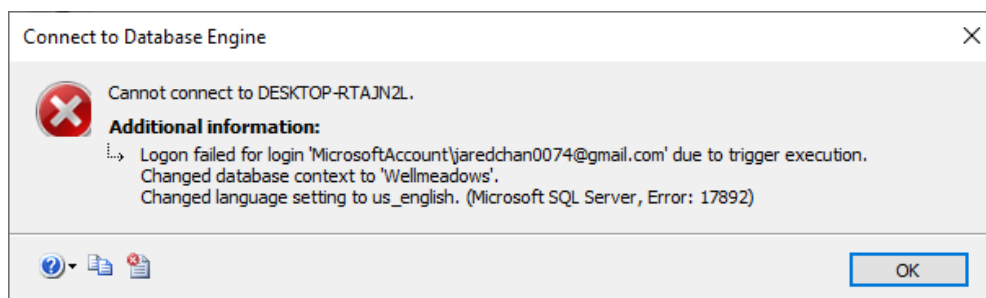


Figure 5.1.4 Error Message Once Additional Query is Opened After 2 Sessions

After 2 sessions of Microsoft SQL Query is opened, if an addition SQL is requested to be opened by the user, the system will prompt an error message stating that the logon for the account failed due to the logon trigger execution. Thus, the new query session won't be allowed for the user unless previous query sessions are terminated.

5.2 Soong Gim Hoy (TP053242)

Logon triggers are triggers that are triggered when a user login to their account. The logon trigger implemented is an audit log trigger where the login of the user is registered, and details of the login is saved in the AuditDB database. The table ServerLogonHistory is created to record the details of user logins. Figure 5.2.1 below shows the code to create the ServerLogonHistory table in the AuditDB database.

```
-- Create Audit Table ServerLogonHistory
CREATE TABLE ServerLogonHistory
(SystemUser VARCHAR(512),
DBUser VARCHAR(512),
SPID INT,
LogonTime DATETIME)

GO
```

Figure 5.2.1: Code to create audit table

Control is then granted to users so that they can automatically update the table whenever they login. Figure 5.2.2 below shows the granting of permissions to the users.

```
GRANT CONTROL SERVER TO Kali
GRANT CONTROL SERVER TO Baw
```

Figure 5.2.2: Code to grant permissions

The logon trigger is then implemented so that all users logging into the database will have their login details recorded. Figure 5.2.3 below shows the code to implement the audit log logon trigger.

```
-- Create Logon Trigger
Go
CREATE TRIGGER Tr_ServerLogon
ON ALL SERVER
FOR LOGON
AS
BEGIN
INSERT INTO AuditDb.dbo.ServerLogonHistory
SELECT SYSTEM_USER,USER,@@SPID,GETDATE()
end
```

Figure 5.2.3: Code to create audit logon trigger

Each user login is recorded in the ServerLogonHistory table and Figure 5.2.4, below shows the login details gathered in the table after the trigger is implemented and triggered.

	SystemUser	DBUser	SPID
1	Baw	Baw	57
2	Baw	Baw	63
3	Kali	Kali	64
4	Kali	Kali	63

Figure 5.2.4: Table ServerLogonHistory showing the login details

5.3 Thong Kyn Hui (TP053489)

```
-- Create Logon Trigger to limit connection after office hours
Go
CREATE TRIGGER LimitConnectionAfterOfficeHours
ON ALL SERVER
FOR LOGON
AS
BEGIN
IF ORIGINAL_LOGIN() = 'Ben' AND
  (DATEPART(HOUR, GETDATE()) < 8 OR
   DATEPART (HOUR, GETDATE()) > 17)
BEGIN
  PRINT 'You are not authorized to login after office hours'
  ROLLBACK
END
END
```

Figure 5.3.1 Logon Trigger Code Snippet for Limiting Query Sections

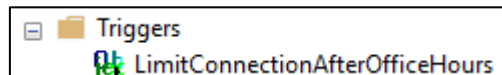


Figure 5.3.2 Logon Trigger Created in Server

The logon trigger created is to limit user connection to the database after normal office hours. The system will check the system time whenever a user login. If the time is before or after office hours, the system will prompt an error message stating that the logon action failed due to trigger execution. The error message is as shown in figure below.

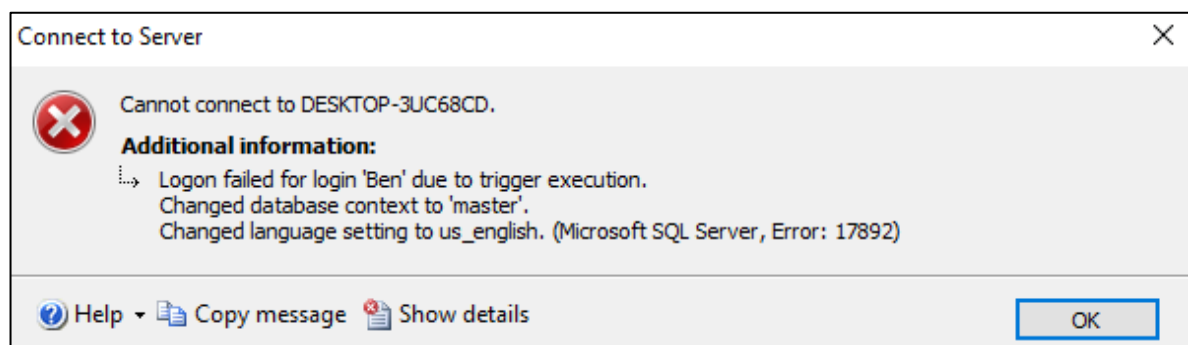


Figure 5.3.3 Error Message When Logon Attempt After Office Hour

6.0 Modification Trigger

6.1 Chan Jia Le (TP049952)

DML Trigger

INSTEAD OF INSERT & UPDATE TRIGGER (check_salary_insert)

```

/* TRIGGER TO CHECK INSERTED AND UPDATED DETAILS OF NURSE SALARY BASED ON ROLE */
CREATE TRIGGER check_salary_insert
ON Nurse
INSTEAD OF INSERT, UPDATE
AS
IF EXISTS (SELECT Salary FROM INSERTED WHERE Salary NOT BETWEEN 2000 AND 4500 AND Position = 'General Nurse')
BEGIN
    PRINT 'Salary of General Nurse Must Be Between 2000 to 4500'
    INSERT INTO NurseAudit
    SELECT Nurse_Num, First_Name, Last_Name, Tel_Num, DOB, Full_Address, Gender, Position, Staff_Num, Ward_Num, Salary, getdate(),
    CONCAT('Unsuccessful Salary Insertion or Modification on ', Nurse_Num, ' Position: ',Position), ORIGINAL_LOGIN() FROM inserted
    Select * FROM inserted
END
ELSE IF EXISTS (SELECT Salary FROM INSERTED WHERE Salary NOT BETWEEN 5000 AND 8500 AND Position = 'Head Nurse')
BEGIN
    PRINT 'Salary of Head Nurse Must Be Between 5000 to 8500'
    INSERT INTO NurseAudit
    SELECT Nurse_Num, First_Name, Last_Name, Tel_Num, DOB, Full_Address, Gender, Position, Staff_Num, Ward_Num, Salary, getdate(),
    CONCAT('Unsuccessful Salary Insertion or Modification on ', Nurse_Num, ' Position: ',Position), ORIGINAL_LOGIN() FROM inserted
END
ELSE IF EXISTS (SELECT Salary FROM INSERTED WHERE Salary BETWEEN 2000 AND 4500 AND Position = 'General Nurse')
BEGIN
    Print 'Successful Insertion or Modification'
    INSERT INTO NurseAudit
    SELECT Nurse_Num, First_Name, Last_Name, Tel_Num, DOB, Full_Address, Gender, Position, Staff_Num, Ward_Num, Salary, getdate(),
    CONCAT('Successful Salary Insertion or Modification on ', Nurse_Num, ' Position: ',Position), ORIGINAL_LOGIN() FROM inserted
    -- CHECK IS DELETE FROM EXISTING RECORD
    IF EXISTS (SELECT * FROM deleted)
    BEGIN
        DELETE FROM Nurse WHERE Nurse_Num = (SELECT Nurse_Num FROM deleted)
    END
    INSERT INTO Nurse SELECT Nurse_Num, First_Name, Last_Name, Tel_Num, DOB, Full_Address, Gender, Position, Staff_Num, Ward_Num,Salary,Stat FROM inserted
END
ELSE IF EXISTS (SELECT Salary FROM INSERTED WHERE Salary BETWEEN 5000 AND 8500 AND Position = 'Head Nurse')
BEGIN
    Print 'Successful Insertion or Modification'
    INSERT INTO NurseAudit
    SELECT Nurse_Num, First_Name, Last_Name, Tel_Num, DOB, Full_Address, Gender, Position, Staff_Num, Ward_Num, Salary, getdate(),
    CONCAT('Successful Salary Insertion or Modification on ', Nurse_Num, ' Position: ',Position), ORIGINAL_LOGIN() FROM inserted
    IF EXISTS (SELECT * FROM deleted)
    BEGIN
        DELETE FROM Nurse WHERE Nurse_Num = (SELECT Nurse_Num FROM deleted)
    END
    INSERT INTO Nurse SELECT Nurse_Num, First_Name, Last_Name, Tel_Num, DOB, Full_Address, Gender, Position, Staff_Num, Ward_Num,Salary,Stat FROM inserted
END
GO

```

Figure 6.1.1 Trigger for checking salary insert and update based on position

```

/* CREATING AUDITING TABLE FOR NURSE*/
CREATE TABLE NurseAudit
(
    Nurse_Num int not null,
    First_Name varchar(20),
    Last_Name varchar(20),
    Tel_Num varchar(20),
    DOB varchar(20),
    Full_Address varchar(100),
    Gender varchar(10),
    Position varchar(20),
    Staff_Num int,
    Ward_Num int,
    Salary int,
    Date_modi date,
    Descrription varchar(100),
    Trigger_acc varchar(50)
)

```

Figure 6.1.2 Creating NurseAudit table for auditing nurse table

The check_salary_insert trigger is a trigger that is consisted of instead of insert and update. It will go through the process of checking the new insertion or updates that is passed to the Nurse table. It will first check whether the nurse salary is valid based on the role either head nurse or general nurse. If it is unsuccessful, the details of the insert and update will be stored in the Nurse Audit table for auditing. Additionally, if the details are correct or valid, the trigger will identify whether this is an update statement or insert statement with a conditional if

```
/* TRY UPDATING WITH ERROR SALARY*/
UPDATE Nurse
SET Salary = 1000
WHERE Nurse_Num = 701
```

Figure 6.1.3 Executing an update statement that is not valid

Salary of General Nurse Must Be Between 2000 to 4500

Figure 6.1.4 Output of the invalid update statement

```
/* CHECK TABLE */
SELECT * FROM Nurse
SELECT * FROM NurseAudit
```

Figure 1.1.5 Query for Nurse and NurseAudit Table

Nurse_Num	First_Name	Last_Name	Tel_Num	DOB	Full_Address	Gender	Position	Staff_Num	Ward_Num	Salary
701	Alysha	Vinson	012-4521253	1987-05-23	8/1,Jln 3/146 Bandar Tasik Selatan Wilayah Persekutuan...	Female	General Nurse	601	2	4000

Figure 6.1.6 Result of Nurse Table

Nurse_Num	First_Name	Last_Name	Tel_Num	DOB	Full_Address	Gender	Position
701	Alysha	Vinson	012-4521253	1987-05-23	8/1,Jln 3/146 Bandar Tasik Selatan Wilayah Perse...	Female	General Nurse
Staff_Num	Ward_Num	Salary	Date_modi	Description	Trigger_acc		
601	2	1000	2021-08-05	Unsuccessful Salary Insertion or Modification on 701 Position: General Nurse	MicrosoftAccount\jaredchan0074@gmail.com		

Figure 6.1.7 Result of NurseAudit Table

statement. If the statement is an update statement (checked based on whether there are details in the deleted memory table), it will first remove the previous details and insert the new information into the table. On the contrary, if deleted table is empty which indicates that its an insert statement, there will not be any removal of details and will only insert the new details.

Upon execution of the update statement as shown in the figure above, the output will state that the salary of the general nurse must be between 2000 to 4500. The Nurse table will not have any updates and there will be a record inserted into the NurseAudit Table with a detailed description of what gone wrong and which account has triggered this execution for auditing.

```
/* TRY UPDATING WITH VALID SALARY*/
UPDATE Nurse
SET Salary = 4500
WHERE Nurse_Num = 701
```

Figure 63.1.8 Executing an update statement that is valid

Successful Insertion or Modification

Figure 2.1.10 Result of Nurse Table

Nurse_Num	First_Name	Last_Name	Tel_Num	DOB	Full_Address	Gender	Position
701	Alysha	Vinson	012-4521253	1987-05-23	8/1,Jln 3/146 Bandar Tasik Selatan Wilayah Perse...	Female	General Nurse
701	Alysha	Vinson	012-4521253	1987-05-23	8/1,Jln 3/146 Bandar Tasik Selatan Wilayah Perse...	Female	General Nurse
Staff_Num	Ward_Num	Salary	Date_modi	Description	Trigger_acc		
601	2	1000	2021-08-05	Unsuccessful Salary Insertion or Modification on 701 Position: General Nurse	MicrosoftAccount\jaredchan0074@gmail.com		
601	2	4500	2021-08-05	Successful Salary Insertion or Modification on 701 Position: General Nurse	MicrosoftAccount\jaredchan0074@gmail.com		

Figure 6.1.11 Result of NurseAudit Table

Upon execution of a valid update, the trigger will then state that the insertion or modification is successful message, and the result is as shown in the figure above where Nurse table's salary for 701 is now updated into 4500 instead of the previous 4000. After that, the result of the update will also be inserted inside the NurseAudit table as shown in the result of NurseAudit Table and the description will be "Successful Salary Insertion or Modification on 701 Position: General Nurse".

```
/* TRY INSERTING WITH INVALID RESULT */
INSERT INTO Nurse VALUES
(759, 'Well', 'Done', '0124168988', '1951-05-23', '32, Try Medan Kurau 2, Where', 'Female', 'Head Nurse', 601, 2, 2000, 'Active')
```

Figure 6.1.12 Executing an insert statement that is not valid

Salary of Head Nurse Must Be Between 5000 to 8500

Figure 6.1.13 Output of the valid update statement

The above insertion will be inserting a new record that is a head nurse, but salary of head nurse is only 2000 which is not valid. The trigger will be executed, and the output will be as shown in the figure above stating that position head nurse must have salary between 5000 to 8500.

Nurse_Num	First_Name	Last_Name	Tel_Num	DOB	Full_Address	Gender	Position
701	Alysha	Vinson	012-4521253	1987-05-23	8/1,Jln 3/146 Bandar Tasik Selatan Wilayah Perse...	Female	General Nurse
701	Alysha	Vinson	012-4521253	1987-05-23	8/1,Jln 3/146 Bandar Tasik Selatan Wilayah Perse...	Female	General Nurse
759	Well	Done	0124168988	1951-05-23	32, Try Medan Kurau 2, Where	Female	Head Nurse
Staff_Num	Ward_Num	Salary	Date_modi	Description	Trigger_acc		
601	2	1000	2021-08-05	Unsuccessful Salary Insertion or Modification on 701 Position: General Nurse	MicrosoftAccount\jaredchan0074@gmail.com		
601	2	4500	2021-08-05	Successful Salary Insertion or Modification on 701 Position: General Nurse	MicrosoftAccount\jaredchan0074@gmail.com		
601	2	2000	2021-08-05	Unsuccessful Salary Insertion or Modification on 759 Position: Head Nurse	MicrosoftAccount\jaredchan0074@gmail.com		

Figure 6.1.14 Result of NurseAudit Table

The insertion will not be executed successfully, and the result of the execution will be stored inside the NurseAudit table stating that the insertion or update is unsuccessful.

```
/* TRY INSERTING WITH VALID RESULT */
INSERT INTO Nurse VALUES
(759, 'Well', 'Done', '0124168988', '1951-05-23', '32, Try Medan Kurau 2, Where', 'Female', 'Head Nurse', 601, 2, 7000, 'Active')
```

Figure 5.1.15 Executing an insert statement that is valid

Successful Insertion or Modification

Figure 6.1.16 Output of the valid insert statement

The above insertion is valid with salary amount of 7000 for head nurse which is within the range of valid inputs. Hence, once the query is executed, the query shows successful insertion

or modification and as always the results will be stored inside the NurseAudit table, and the insertion will be successfully added into the Nurse table in this case.

759	Well	Done	0124168988	1951-05-23	32, Try Medan Kurau 2, Where	Female	Head Nurse	601	2	7000
-----	------	------	------------	------------	------------------------------	--------	------------	-----	---	------

Figure 6.1.17 Result of Nurse Table after insertion

759	Well	Done	0124168988	1951-05-23	32, Try Medan Kurau 2, Where	Female	Head Nurse
601	2	7000	2021-08-05	Successful Salary Insertion or Modification on 759 Position: Head Nurse			MicrosoftAccount\jaredchan0074@gmail.com

Figure 6.1.18 Result of NurseAudit Table

Like the successful update, the result of the successful insertion will be reflected inside the Nurse table and NurseAudit table as shown in the figure above.

INSTEAD OF DELETE TRIGGER (stop_del_nurse)

```

/* TRIGGER TO RESTRICT DELETION OF NURSE ACCOUNT AND SET IT INTO INACTIVE */
CREATE TRIGGER stop_del_nurse
ON Nurse
INSTEAD OF DELETE
AS
BEGIN
    RAISERROR ('Nurse Account Should Not Be Deleted, System will change the status into inactive',14,11)
    INSERT INTO NurseAudit
    SELECT Nurse_Num, First_Name, Last_Name, Tel_Num, DOB, Full_Address, Gender, Position, Staff_Num, Ward_Num, Salary, getdate(),
    CONCAT('Attempted Deletion on ', Nurse_Num, ' Position: ',Position), ORIGINAL_LOGIN() FROM deleted

    UPDATE Nurse
    SET Stat = 'Inactive'
    FROM Nurse as s INNER JOIN deleted as d
    ON s.Nurse_Num = d.Nurse_Num
END

```

Figure 6.1.19 Trigger for restricting deletion and setting nurse status into Inactive

The stop_del_nurse trigger is created to restrict deletion of nurse details and only setting the nurse details from active to inactive. Once an update statement is executed, the trigger will raise an error stating that nurse should not be deleted and will proceed to save the execution attempt into the NurseAudit table and proceed to change the status from active into Inactive.

```

/* TRY DELETING A NURSE RECORD*/
DELETE FROM Nurse
WHERE Nurse_Num = 701

```

Figure 6.1.20 Executing delete statement

```

Msg 50000, Level 14, State 11, Procedure stop_del_nurse, Line 6 [Batch Start Line 244]
Nurse Account Should Not Be Deleted, System will change the status into inactive

```

Figure 6.1.21 Output of delete statement

	Nurse_Num	First_Name	Last_Name	Tel_Num	DOB	Position	Stat
1	701	Alysha	Vinson	012-4521253	1987-05-23	General Nurse	Inactive

Figure 6.1.22 Result of Nurse Table after attempted deletion

Nurse_Num	First_Name	Last_Name	Tel_Num	DOB	Full_Address	Gender	Position	Staff_Num
701	Alysha	Vinson	012-4521253	1987-05-23	8/1,Jln 3/146 Bandar Tasik Selatan Wilayah Perse...	Female	General Nurse	601
Ward_Num	Salary	Date_modi	Description	Trigger_acc				
2	4500	2021-08-05	Attempted Deletion on 701 Position: General Nurse	MicrosoftAccount\jaredchan0074@gmail.com				

Figure 6.1.23 Result of NurseAudit Table

Upon execution of deletion statement, the system as expected will raise an error stating that nurse should not be deleted, and it will update the status of the nurse into inactive as shown in the figure above. Thus, the attempted deletion will be written and stored inside the NurseAudit table.

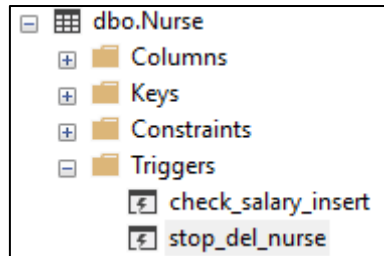


Figure 6.1.24 Triggers for Nurse Table After Execution

After the triggers are created, the trigger will be located under the database and inside the Nurse table. The triggers can then be disabled or enabled by the developer for testing and maintenance.

6.2 Soong Gim Hoy (TP053242)

The 2 DML triggers created are to prevent the deletion of next of kin details and patient details. Figure 6.2.1 below shows the code used for creating the deletion prevention trigger in next of kin table. After the query is run, the number of affected rows is shown in Figure 6.2.2. The deleteNK trigger can then be seen in the database triggers as shown in Figure 6.2.3.

```
-- Create DML trigger to prevent deletion of next of kin details
Go
ALTER TRIGGER deleteNK
ON Next_of_Kin
INSTEAD OF delete
AS
BEGIN
    select * from deleted
    RAISERROR('Next of Kin cannot be deleted', 16,10)
    select * from Next_of_Kin
END
```

Figure 6.2.1: Code to create DML trigger to prevent deletion of next of kin details

```
(1 row affected)
Msg 50000, Level 16, State 10, Procedure deleteNK, Line 7 [Batch Start Line 83]
Next of Kin cannot be deleted

(20 rows affected)

(1 row affected)

Completion time: 2021-08-06T21:53:44.0790033+08:00
```

Figure 6.2.2: Result after query is run



Figure 6.2.3: Trigger present in database triggers

The code to prevent patient details to be deleted from the database is shown in Figure 6.2.4. The result of the query implemented is shown in Figure 6.2.5. The trigger can then be seen in the database triggers as shown in Figure 6.2.6.

```
-- Create DML trigger to prevent deletion of patient details
Go
ALTER TRIGGER deletePatient
ON Patient
INSTEAD OF delete
AS
BEGIN
    select * from deleted
    RAISERROR('Patient cannot be deleted', 16,10)
    select * from Patient
END
```

Figure 6.2.4: Code to create DML trigger to prevent deletion of patient details

```
(1 row affected)
Msg 50000, Level 16, State 10, Procedure deletePatient, Line 7 [Batch Start Line 100]
Patient cannot be deleted

(19 rows affected)

(1 row affected)

Completion time: 2021-08-06T21:55:00.4421690+08:00
```

Figure 6.2.5: Result after query is run

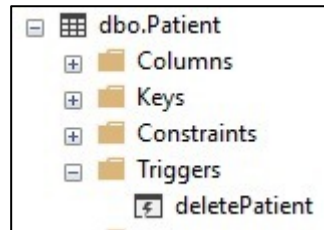


Figure 6.2.6: Trigger present in database triggers

6.3 Thong Kyn Hui (TP053489)

AFTER INSERT & UPDATE TRIGGER (check_appointment_time)

```
-- Create DML trigger for insert & update Appointment Time
Go
ALTER TRIGGER check_appointment_time
ON Appointment
INSTEAD OF INSERT,UPDATE
AS
IF exists ( select Appointment_Time from inserted where Appointment_Time NOT BETWEEN '08:00:00' AND '18:00:00' )
begin
print 'Invalid time inserted. Please ensure appointment time is within office hours.'
INSERT INTO Appointment_Audit
SELECT Appointment_Num, Appointment_Date, Appointment_Time, Appointment_Location, Patient_Num, Staff_Num, getdate(),
'Unsuccessful appointment slot due to appointment time not within office hour' FROM inserted
End
IF exists ( select Appointment_Time from inserted where Appointment_Time BETWEEN '08:00:00' AND '18:00:00' )
begin
print 'Successful appointment time slot booked.'
INSERT INTO Appointment_Audit
SELECT Appointment_Num, Appointment_Date, Appointment_Time, Appointment_Location, Patient_Num, Staff_Num, getdate(),
'Successful appointment time slot booked.' FROM inserted
INSERT INTO Appointment
SELECT Appointment_Num, Appointment_Date, Appointment_Time, Appointment_Location, Patient_Num, Staff_Num FROM inserted
End
```

Figure 6.3.1 Trigger for checking appointment time insert and update

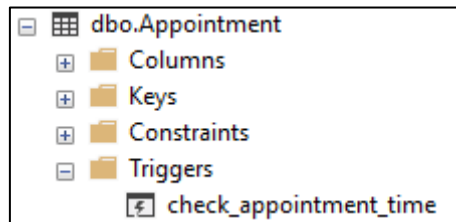


Figure 6.3.2 Trigger created for appointment table

```
-- Create Appointment_Audit Table
CREATE TABLE Appointment_Audit
(
Appointment_Num          Int,
Appointment_Date         Date,
Appointment_Time         Time,
Appointment_Location     Varchar(10),
Patient_Num             Int,
Staff_Num               Int,
Date_Modified           Date,
Description              Varchar(100),
)
```

Figure 6.3.3 Creating appointment audit table for auditing appointment table

The check_appointment_time trigger is a trigger that is consisted of after inserting and update. It will go through the process of checking the new insertion or updates that is passed to the appointment table. It will first check whether the appointment time slot is within office hour. If it is unsuccessful, the details of the insert and update will be stored in the appointment_audit table for auditing.

```
-- Try inserting appointment with valid appointment time
INSERT INTO Appointment VALUES
(543, '2019-05-10', '08:30:00', 'E801', 109, 605)
SELECT * FROM Appointment_Audit
SELECT * FROM Appointment
```

Figure 6.3.4 Executing an insert statement with valid appointment time

	Appointment_Num	Appointment_Date	Appointment_Time	Appointment_Location	Patient_Num	Staff_Num	Date_Modified	Description
1	541	2019-05-10	20:30:00.0000000	E801	109	605	2021-08-06	Unsuccessful appointment slot due to appointme...
2	541	2019-05-10	08:30:00.0000000	E801	109	605	2021-08-06	Successful appointment time slot booked.
3	542	2019-05-10	06:30:00.0000000	E801	109	605	2021-08-06	Unsuccessful appointment slot due to appointme...
4	543	2019-05-10	08:30:00.0000000	E801	109	605	2021-08-06	Successful appointment time slot booked.

	Appointment_Num	Appointment_Date	Appointment_Time	Appointment_Location	Patient_Num	Staff_Num
1	543	2019-05-10	08:30:00.0000000	E801	109	605

Figure 6.3.5 Result of Appointment Audit and Appointment Table

Upon execution of insert query with valid appointment date, the trigger will show a successfully inserted message. The inserted details will be inserted into both the appointment audit and appointment table as shown in figure above.

```
-- Try inserting appointment with invalid appointment time
INSERT INTO Appointment VALUES
(544, '2019-05-10', '20:00:00', 'E801', 109, 605)
SELECT * FROM Appointment_Audit
SELECT * FROM Appointment WHERE Appointment_Num = 543
```

Figure 6.3.6 Executing an insert statement with invalid appointment time

```
Invalid time inserted. Please ensure appointment time is within office hours.

(1 row affected)

(1 row affected)

(5 rows affected)

Completion time: 2021-08-06T15:11:03.3043683+08:00
```

Figure 6.3.7 Output of the invalid insert statement

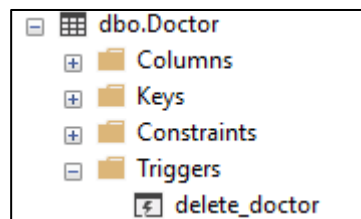
	Appointment_Num	Appointment_Date	Appointment_Time	Appointment_Location	Patient_Num	Staff_Num	Date_Modified	Description
1	541	2019-05-10	20:30:00.0000000	E801	109	605	2021-08-06	Unsuccessful appointment slot due to appointment time not within office hour
2	541	2019-05-10	08:30:00.0000000	E801	109	605	2021-08-06	Successful appointment time slot booked.
3	542	2019-05-10	06:30:00.0000000	E801	109	605	2021-08-06	Unsuccessful appointment slot due to appointment time not within office hour
4	543	2019-05-10	08:30:00.0000000	E801	109	605	2021-08-06	Successful appointment time slot booked.
5	544	2019-05-10	20:00:00.0000000	E801	109	605	2021-08-06	Unsuccessful appointment slot due to appointment time not within office hour

Figure 6.3.8 Result of Appointment Audit

If the appointment time inserted is invalid (not within office hour), the trigger will execute an error message and the appointment will not be inserted into appointment table. Instead, the invalid insertion will be recorded in the appointment audit table.

INSTEAD OF DELETE (delete_doctor)

```
-- Create DML trigger to prevent deletion of doctor details
Go
ALTER TRIGGER delete_doctor
ON Doctor
INSTEAD OF delete
AS
BEGIN
    RAISERROR('Doctor details cannot be deleted', 16,10)
    INSERT INTO Doctor_Audit
    SELECT Staff_Num, Doctor_Name, getdate(), 'Attempted deletion on doctor details' FROM deleted
END
```

Figure 6.3.9 Trigger for restricting deletion of doctor*Figure 6.3.10 Trigger created for doctor table*

```
CREATE TABLE Doctor_Audit
(
    Staff_Num Int,
    Doctor_Name Varchar(25),
    Date_Modified Date,
    Descripton Varchar(100),
)
```

Figure 6.3.11 Creating doctor audit table for auditing doctor table

The delete_doctor trigger is a trigger that is consisted of instead of deleting. The trigger will stop any deletion of doctor details from doctor table. Any attempt of deleting doctor details from the table will be recorded in the doctor audit table.

```
-- Attempt to delete doctor
DELETE FROM Doctor
WHERE Staff_Num = 601
```

Figure 6.3.12 Executing delete statement

```
Msg 50000, Level 16, State 10, Procedure delete_doctor, Line 6 [Batch Start Line 128]
Doctor details cannot be deleted

(1 row affected)

(1 row affected)

Completion time: 2021-08-06T15:28:06.3144062+08:00
```

Figure 6.3.13 Output of delete statement

	Staff_Num	Doctor_Name	Date_Modified	Descripton
1	601	Helen Cho	2021-08-06	Attempted deletion on doctor details

Figure 6.3.14 Result of Doctor Audit Table

7.0 Encryption

7.1 Chan Jia Le (TP049952)

Encryption is a form of protection that encodes information or data with series of algorithms or keys to protect it from unauthorized access or data theft. The receiving party with the decryption key will have the ability to decrypt the information (IBM, 2021). In SQL Server, encryption is conducted with a series of hierarchical encryption options and key management infrastructures, each possessing different elements like certificates, symmetric or asymmetric keys, etc. The hierarchy of the encryption for SQL server is as shown in the figure below:

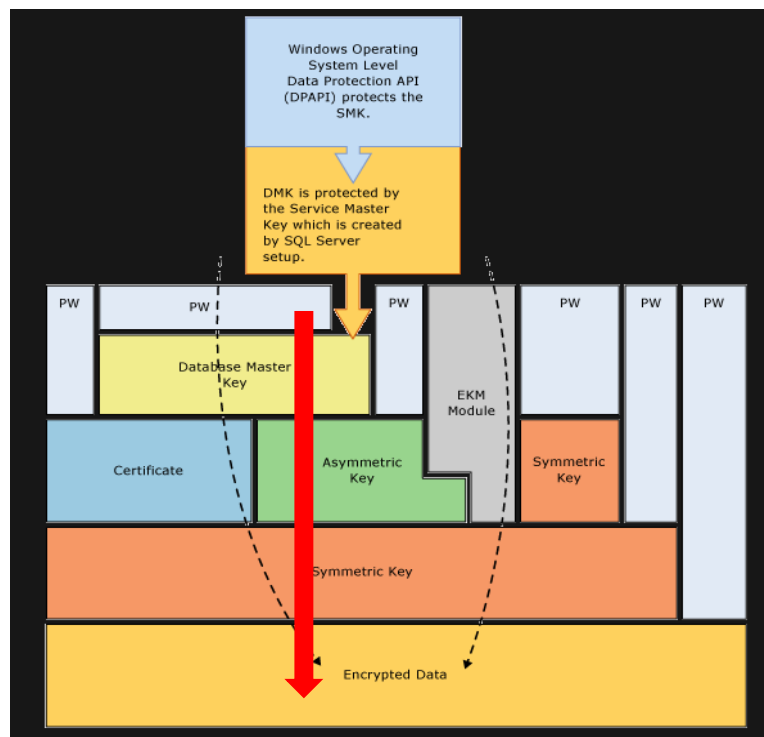


Figure 7.1.1 Encryption Hierarchy *Invalid source specified.*

The selected hierarchy by the developer is from password to Database Master Key to Asymmetric Key, Symmetric Key and Encrypted Data. Different levels of encryption provide different benefits. The reason why asymmetric key is selected instead of the certification is since asymmetric key will not have issues of key distribution issues, additionally, the asymmetric key provides a more secure environment as the private keys will not have to be revealed to anyone (Brush, 2021). Encryption is especially important as a method for keeping data safe and confidential as it is utilized in a server. Whenever, processes are performed, encryption will enable protection that stops unauthorized individuals to see what information is accessed (Beaming, 2017). Furthermore, the selected encrypted data on this scenario is the salary column for both doctor and nurse, the salary column is one of the most confidential details that is stored inside the database, hence, it is crucial that a more secure encryption

combination is selected to protect the salary details which is the reason why developer has chosen the stated combination of encryption.

```
/*CREATING MASTER KEY*/  
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'ENCRYPT@123';  
  
/* CREATING ASYMMETRIC KEY THAT IS PROTECTED BY MASTER KEY */  
Create ASymmetric Key fld_ASymKey1 WITH ALGORITHM = RSA_2048;  
  
/* CREATING SYMMETRIC KEY PROTECTED BY THE ASYMMETRIC KEY */  
CREATE SYMMETRIC KEY Encrypted_Key WITH  
    ALGORITHM = AES_256  
    ENCRYPTION BY ASYMMETRIC KEY fld_ASymKey1
```

Figure 7.1.2 Creating Encryption Combination in SQL

First the master key will be created which is encrypted by a password. The master key is another form of symmetric key. Once the master key is created, the asymmetric key is generated with the RSA algorithm with 2048 bit. Lastly, a symmetric key will be created with AES algorithm which is encrypted by the Asymmetric key. So, the hierarchy will be as described which is Password > DMK (Database Master Key) > Asymmetric Key > Symmetric Key > Encrypted Data and to double check the result a query will be performed as shown in the result below.

```
/*CHECK ENCRYPTION HIEARCHY*/  
select * from sys.symmetric_keys  
select * from sys.asymmetric_keys  
select * from sys.key_encryptions
```

Figure 7.1.3 Query Symmetric, Asymmetric and Key Encryptions.

name	principal_id	symmetric_key_id	key_length	key_algorithm	algorithm_desc	create_date	modify_date	key_guid
##MS_DatabaseMasterKey##	1	101	256	A3	AES_256	2021-08-01 05:06:29.700	2021-08-01 05:06:29.700	607DD800-6636-48FE-B7D2-6A26D62A1B23
Encrypted_Key	1	257	256	A3	AES_256	2021-08-01 05:22:36.030	2021-08-01 05:22:36.030	E8F90F00-069D-4858-BFE5-2A6B72E36224

Figure 7.1.4 Symmetric Key created

	name	principal_id	asymmetric_key_id	pvt_key_encryption_type	pvt_key_encryption_type_desc
1	fId_ASymKey1	1	257	MK	ENCRYPTED_BY_MASTER_KEY

Figure 7.1.5 Asymmetric Key created

	key_id	thumbprint	crypt_type	crypt_type_desc	crypt_property
1	101	0x01	ESKM	ENCRIPTION BY MASTER KEY	0x2A80DA0730FD7231FF9BADDBA0FC0D7E62F7650324044E...
2	101	NULL	ESKP	ENCRIPTION BY PASSWORD	0x30E040E57350913F295D712CA266B8EEB4E9B5734BD0DC...
3	257	0xF6635...	EPUA	ENCRIPTION BY ASYMMETRIC KEY	0xFB415AF3C6CE68A968196FC870FC75992BF2AF49385D8E0...

Figure 7.1.6 Encryption Keys created

As mentioned, DMK (Database master key) is also considered a symmetric key, hence when query for symmetric keys in the database there is 2 keys, one DMK and another one which is the last symmetric key used for encrypting data. The second query conducted is to list the asymmetric key created and as reflected there is one asymmetric key named fId_ASymKey1 which is encrypted by the master key. Lastly, the encryption keys list out the keys that are existent in the database which reflects the level of hierarchy Password > DMK (Database Master Key) > Asymmetric Key > Symmetric Key. Once the keys are created, the developer will proceed to encrypt the salary data of Doctor and Nurse.

```

/* USING THE SYMMETRIC KEY TO ENCRYPT THE SALARY COLUMN */
OPEN SYMMETRIC KEY Encrypted_Key
    DECRYPTION BY ASYMMETRIC Key fId_ASymKey1

UPDATE Nurse
    SET Salary_Encrypted = ENCRYPTBYKEY (Key_GUID('Encrypted_Key'),convert(nvarchar,Salary))
FROM Nurse
GO

UPDATE Doctor
    SET Salary_Encrypted = ENCRYPTBYKEY (Key_GUID('Encrypted_Key'),convert(nvarchar,Salary))
FROM Doctor
GO

```

Figure 7.1.7 Opening the symmetric key and encrypt salary data

```

/* CHECK RESULT OF THE ENCRYPTED OUTPUT*/
SELECT Nurse_Num, First_Name,Last_Name, Salary_Encrypted FROM Nurse
SELECT Staff_Num,Doctor_Name, Salary_Encrypted FROM Doctor

```

Figure 7.1.8 Query Encrypted Data Code

	Nurse_Num	First_Name	Last_Name	Salary_Encrypted
1	701	Alysha	Vinson	0x000FF9E89D065848BFE52A6B72E36224010000002597989C...
2	702	Kenya	Frame	0x000FF9E89D065848BFE52A6B72E3622401000000059F9C7...
3	703	Khadijah	Higgs	0x000FF9E89D065848BFE52A6B72E3622401000000098A9D0...
4	704	Tamia	Summers	0x000FF9E89D065848BFE52A6B72E36224010000008AD40DD...

Figure 7.1.9 Result of salary encryption on Nurse Table

	Staff_Num	Doctor_Name	Salary_Encrypted
1	601	Helen Cho	0x000FF9E89D065848BFE52A6B72E3622401000000635AAE...
2	602	Sumaiya Olson	0x000FF9E89D065848BFE52A6B72E362240100000026C5028...
3	603	Chanel Beech	0x000FF9E89D065848BFE52A6B72E36224010000005837003...
4	604	Annette Fee...	0x000FF9E89D065848BFE52A6B72E36224010000003DC3A2...

Figure 7.1.10 Result of salary encryption on Doctor Table

After opening the symmetric key, the developer can encrypt the salary data into a new column and remove the original salary column later. However, it is important to note that the symmetric must be opened before encryption or decryption or else it will not work.

```
/* CLOSING THE SYMMETRIC KEY TO SEE OUTPUT OF DECRYPTION WITHOUT KEY*/
CLOSE SYMMETRIC KEY Encrypted_Key
```

Figure 7.1.11 Closing the symmetric key to try encryption without opening symmetric key

	Nurse_Num	First_Name	Last_Name	Salary_Encrypted
1	701	Alysha	Vinson	NULL
2	702	Kenya	Frame	NULL
3	703	Khadijah	Higgs	NULL
4	704	Tamia	Summers	NULL
5	705	Inez	Hope	NULL

Figure 7.1.12 Result of salary encryption on Nurse Table without opening symmetric key

	Staff_Num	Doctor_Name	Salary_Encrypted
1	601	Helen Cho	NULL
2	602	Sumaiya Olson	NULL
3	603	Chanel Beech	NULL
4	604	Annette Fee...	NULL
5	605	Maciej Castro	NULL

Figure 7.1.13 Result of salary encryption on Doctor Table without opening symmetric key

As mentioned, without opening the symmetric key, the encryption of the salary data will not be possible, if executed, the value of the encrypted salary will be null as shown in the figure above.

```

/* QUERY RESULT OF DECRYPTED SALARY*/
Select Salary_Encrypted, CONVERT(nvarchar, DECRYPTBYKEY(Salary_Encrypted)) AS decrypted_salary from Nurse
Select Salary_Encrypted, CONVERT(nvarchar, DECRYPTBYKEY(Salary_Encrypted)) AS decrypted_salary from Doctor

```

Figure 7.1.14 Decrypt the Encrypted salary data for query

	Salary_Encrypted	decrypted_salary
1	0x000FF9E89D065848BFE52A6B72E3622401000000BA255B5...	4500
2	0x000FF9E89D065848BFE52A6B72E3622401000000389D057...	3000
3	0x000FF9E89D065848BFE52A6B72E362240100000093CFED2...	5000
4	0x000FF9E89D065848BFE52A6B72E3622401000000ED4D77...	5000
5	0x000FF9E89D065848BFE52A6B72E36224010000006E5AA9A...	5000

Figure 7.1.15 Result of salary decryption on Nurse Table

	Doctor_Name	Salary_Encrypted	decrypted_salary
1	Helen Cho	0x000FF9E89D065848BFE52A6B72E362240100000092FA47...	4000
2	Sumaiya Olson	0x000FF9E89D065848BFE52A6B72E36224010000009A051C0...	3500
3	Chanel Beech	0x000FF9E89D065848BFE52A6B72E36224010000007A3239A...	4000
4	Annette Feeney	0x000FF9E89D065848BFE52A6B72E3622401000000ABAA14E...	3000

Figure 7.1.16 Result of salary decryption on Doctor Table

Similarly, if the symmetric key is opened and decryption execution is triggered, the query for the decryption will perform as expected as shown in the figures above. However, if the symmetric key is closed, the decryption key will be null as displayed in the results below.

	Salary_Encrypted	decrypted_salary
1	0x000FF9E89D065848BFE52A6B72E3622401000000BA255B5...	NULL
2	0x000FF9E89D065848BFE52A6B72E3622401000000389D057...	NULL
3	0x000FF9E89D065848BFE52A6B72E362240100000093CFED2...	NULL
4	0x000FF9E89D065848BFE52A6B72E3622401000000ED4D77...	NULL

Figure 7.1.17 Result of salary decryption on Nurse Table without opening symmetric key

	Doctor_Name	Salary_Encrypted	decrypted_salary
1	Helen Cho	0x000FF9E89D065848BFE52A6B72E362240100000092FA47...	NULL
2	Sumaiya Olson	0x000FF9E89D065848BFE52A6B72E36224010000009A051C...	NULL
3	Chanel Beech	0x000FF9E89D065848BFE52A6B72E36224010000007A3239...	NULL
4	Annette Fee...	0x000FF9E89D065848BFE52A6B72E3622401000000ABAA14...	NULL

Figure 7.1.18 Result of salary decryption on Doctor Table without opening symmetric key

7.2 Soong Gim Hoy (TP053242)

Master key encryption is used to encrypt the addresses of next of kin, nurses, and patients. It is used to prevent the sensitive address data to be accessed without the master key decryption. The address columns in each table are encrypted with certificate and symmetric key to prevent unauthorised personal from accessing and retrieving the data. Figure 7.2.1 below shows the code used to create the master key, certificate and symmetric key.

```
-- Create encryption on address column with symmetric key
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Encrypt@123';
CREATE CERTIFICATE addresscert WITH SUBJECT = 'Address Certificate';
CREATE SYMMETRIC KEY address_symmkey WITH
    ALGORITHM = AES_256
    ENCRYPTION BY CERTIFICATE addresscert
GO
OPEN SYMMETRIC KEY address_symmkey
    DECRYPTION BY CERTIFICATE addresscert;
GO
```

Figure 7.2.1 Code to create symmetric key, master key and certificate

If the database is being accessed by unauthorised users, the encrypted data cannot be understood without the symmetric key and certificate for decryption. Figure 7.2.2 below shows the code to add in another column in tables Next_of_Kin, Patient, and Nurse for encrypted values of the address column.

```
-- ENCRYPTING address COLUMN FOR Next_of_Kin TABLE
ALTER TABLE Next_of_Kin
ADD NK_Address_Encrypted varbinary(MAX)

UPDATE Next_of_Kin
SET NK_Address_Encrypted = ENCRYPTBYKEY (Key_GUID('address_symmkey'), NK_Address)
FROM Next_of_Kin
GO

-- ENCRYPTING address COLUMN FOR PATIENT TABLE
ALTER TABLE Patient
ADD Patient_Address_Encrypted varbinary(MAX)

UPDATE Patient
SET Patient_Address_Encrypted = ENCRYPTBYKEY (Key_GUID('address_symmkey'), Address)
FROM Patient
GO

-- ENCRYPTING address COLUMN FOR NURSE TABLE
ALTER TABLE Nurse
ADD Address_Encrypted varbinary(MAX)

UPDATE Nurse
SET Address_Encrypted = ENCRYPTBYKEY (Key_GUID('address_symmkey'), Full_Address)
FROM Nurse
GO
```

Figure 7.2.2: Code to add columns for encrypted address values

7.3 Thong Kyn Hui (TP053489)

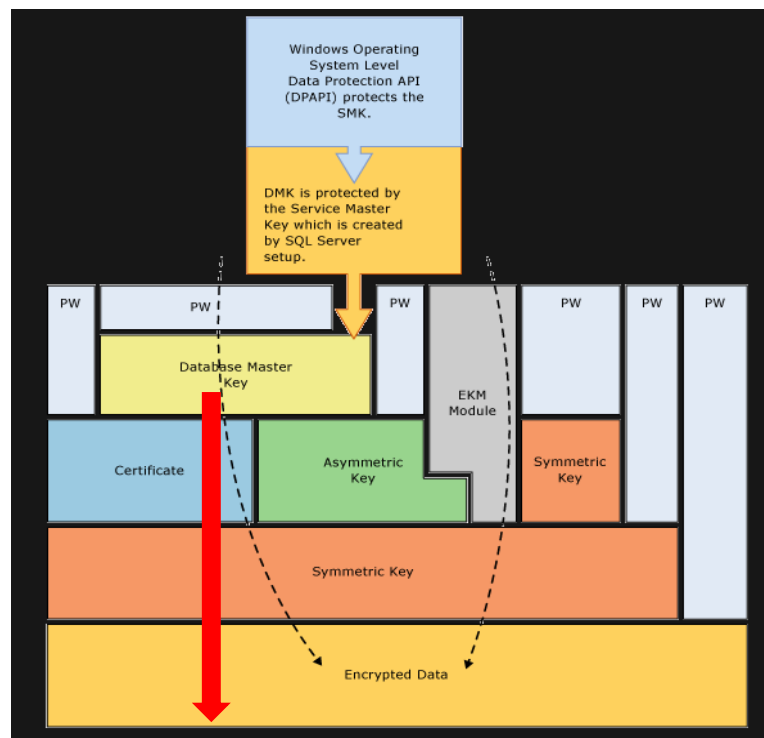


Figure 7.3.1 Encryption Hierarchy *Invalid source specified.*

The selected encryption hierarchy by the developer is from password, Database Master Key, Certificate, Symmetric Key to Encrypted Data. The reason of choosing certificate over asymmetric key is because the execution and implementation of certificate is straightforward and simple compared to asymmetric key. The encrypted data in this case is the telephone number of nurses.

```
-- Create encryption on telephone number column with symmetric key
-- Create Master Key
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Encrypt@123';
-- Create Certificate
CREATE CERTIFICATE telcert WITH SUBJECT = 'Tel Certificate';
-- Create Symmetric Key
CREATE SYMMETRIC KEY tel_symmkey WITH
    ALGORITHM = AES_256
    ENCRYPTION BY CERTIFICATE telcert
GO
```

Figure 7.3.2 Creating Encryption Combination in SQL

First the master key will be created which is encrypted by a password. The master key is another form of symmetric key. Once the master key is created, the certificate is generated with the. Lastly, a symmetric key will be created with AES algorithm which is encrypted by the

certificate. So, the hierarchy will be as described which is Password > DMK (Database Master Key) > Certificate > Symmetric Key > Encrypted Data.

```
-- ENCRYPTING Tel_Num COLUMN FOR NURSE AND PATIENT TABLE
OPEN SYMMETRIC KEY tel_symmkey
    DECRYPTION BY CERTIFICATE telcert;
GO

ALTER TABLE Nurse
ADD Tel_Num_Encrypted varbinary(MAX)

ALTER TABLE Patient
ADD Tel_Num_Encrypted varbinary(MAX)

UPDATE Nurse
    SET Tel_Num_Encrypted = ENCRYPTBYKEY (Key_GUID('tel_symmkey'), Tel_Num)
FROM Nurse
GO

UPDATE Patient
    SET Tel_Num_Encrypted = ENCRYPTBYKEY (Key_GUID('tel_symmkey'), Tel_Num)
FROM Patient
GO
```

Figure 7.3.3 Opening the symmetric key and encrypt telephone number

```
-- Checking result of encryption
select Nurse_Num, First_Name, Tel_Num_Encrypted from Nurse
select Patient_Num, First_Name, Tel_Num_Encrypted from Patient
```

Figure 7.3.4 Query Encrypted Data Code

Results Messages			
	Nurse_Num	First_Name	Tel_Num_Encrypted
1	701	Alysha	0x006EC8560895F5419ADC444D22F0C69202000000E8B43A...
2	702	Kenya	0x006EC8560895F5419ADC444D22F0C69202000000903DA63...
3	703	Khadjah	0x006EC8560895F5419ADC444D22F0C69202000000A12F04D...
4	704	Tamia	0x006EC8560895F5419ADC444D22F0C692020000001C18A7B...
5	705	Inez	0x006EC8560895F5419ADC444D22F0C692020000005A61BD...
6	706	GUto	0x006EC8560895F5419ADC444D22F0C69202000000143E5FB...
7	707	Jagdeep	0x006EC8560895F5419ADC444D22F0C69202000000840BCB6...
8	708	Ruby-Mae	0x006EC8560895F5419ADC444D22F0C69202000000FCB7C8B...
	Patient_Num	First_Name	Tel_Num_Encrypted
1	102	Ariadne	0x006EC8560895F5419ADC444D22F0C69202000000FBFF320...
2	103	Eliana	0x006EC8560895F5419ADC444D22F0C692020000009445E1F...
3	104	Elsie-May	0x006EC8560895F5419ADC444D22F0C6920200000025D0F1...
4	105	Hallie	0x006EC8560895F5419ADC444D22F0C69202000000B25730F...
5	106	Jazmyn	0x006EC8560895F5419ADC444D22F0C69202000000F0C99A...
6	107	Winston	0x006EC8560895F5419ADC444D22F0C6920200000059BEBA...
7	108	Temi	0x006EC8560895F5419ADC444D22F0C69202000000F565DF...
8	109	Albie	0x006EC8560895F5419ADC444D22F0C69202000000AC35EB...

Figure 7.3.5 Result of telephone number encryption on Nurse and Patient Table

	Tel_Num_Encrypted	decrypted_Tel_Num
1	0x006EC8560895F5419ADC444D22F0C69202000000E8B43AD...	012-4521253
2	0x006EC8560895F5419ADC444D22F0C69202000000903DA63...	012-7678311
3	0x006EC8560895F5419ADC444D22F0C69202000000A12F04D...	017-4234451
4	0x006EC8560895F5419ADC444D22F0C692020000001C18A7B...	0112-2343253
5	0x006EC8560895F5419ADC444D22F0C692020000005A61BDA...	018-3356436
6	0x006EC8560895F5419ADC444D22F0C69202000000143E5FB...	019-6843574
7	0x006EC8560895F5419ADC444D22F0C69202000000840BCB6...	013-2749374
8	0x006EC8560895F5419ADC444D22F0C69202000000FCB7C8B...	017-2474932
	Tel_Num_Encrypted	decrypted_Tel_Num
1	0x006EC8560895F5419ADC444D22F0C69202000000FBFF320...	017-59264642
2	0x006EC8560895F5419ADC444D22F0C692020000009445E1F...	019-1453542
3	0x006EC8560895F5419ADC444D22F0C6920200000025D0F1...	016-4636782
4	0x006EC8560895F5419ADC444D22F0C69202000000B25730F...	019-2957382
5	0x006EC8560895F5419ADC444D22F0C69202000000F0C99AF...	012-1048492
6	0x006EC8560895F5419ADC444D22F0C6920200000059BEBA...	017-2345325
7	0x006EC8560895F5419ADC444D22F0C69202000000F565DF3...	0112-2390132
8	0x006EC8560895F5419ADC444D22F0C69202000000AC35EB...	017-5929294

Figure 7.3.6 Result of telephone number encryption on Nurse and Patient Table

After opening the symmetric key, the developer can encrypt the salary data into a new column and remove the original salary column later. However, it is important to note that the symmetric must be opened before encryption or description or else it will not work

```
-- Checking result of encryption after closing symmetric key
CLOSE SYMMETRIC KEY tel_symmkey
select Tel_Num_Encrypted, CONVERT(varchar,DECRYPTBYKEY(Tel_Num_Encrypted)) AS decrypted_Tel_Num from Nurse
select Tel_Num_Encrypted, CONVERT(varchar,DECRYPTBYKEY(Tel_Num_Encrypted)) AS decrypted_Tel_Num from Patient
```

Figure 7.3.7 Trying encryption without opening symmetric key

	Tel_Num_Encrypted	decrypted_Tel_Num
1	0x006EC8560895F5419ADC444D22F0C69202000000E8B43AD...	NULL
2	0x006EC8560895F5419ADC444D22F0C69202000000903DA63...	NULL
3	0x006EC8560895F5419ADC444D22F0C69202000000A12F04D...	NULL
4	0x006EC8560895F5419ADC444D22F0C692020000001C18A7B...	NULL
5	0x006EC8560895F5419ADC444D22F0C692020000005A61BDA...	NULL
6	0x006EC8560895F5419ADC444D22F0C69202000000143E5FB...	NULL
7	0x006EC8560895F5419ADC444D22F0C69202000000840BCB6...	NULL
8	0x006EC8560895F5419ADC444D22F0C69202000000FCB7C8B...	NULL

	Tel_Num_Encrypted	decrypted_Tel_Num
1	0x006EC8560895F5419ADC444D22F0C6920200000009752BD...	NULL
2	0x006EC8560895F5419ADC444D22F0C69202000000436CCF3...	NULL
3	0x006EC8560895F5419ADC444D22F0C692020000004DC72E7...	NULL
4	0x006EC8560895F5419ADC444D22F0C69202000000F4A0379...	NULL
5	0x006EC8560895F5419ADC444D22F0C69202000000F966D85...	NULL
6	0x006EC8560895F5419ADC444D22F0C692020000006E0500D...	NULL
7	0x006EC8560895F5419ADC444D22F0C692020000006A78513...	NULL
8	0x006EC8560895F5419ADC444D22F0C69202000000E4B5262...	NULL

Figure 7.3.8 Result of telephone number encryption without opening symmetric key

8.0 Backup and Restore

The backup and restore strategy implemented in Wellmeadows is with encryption for security and compression for saving space. A master key and certificate are used to authenticate the restoration of the database to ensure the database is protected. Figure 8.0.1 below shows the code for creating the database master key and certificate. Figure 8.0.2 and Figure 8.0.3 below shows that the master key and certificate are both created.

```
--Backup Restore Strategy
-- create Database Master Key
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Backup@123';
GO

select * from sys.symmetric_keys

CREATE CERTIFICATE BackupEncryption
    WITH SUBJECT = 'Backup Certificate';
GO

select * from sys.certificates
```

Figure 8.0.1: Code to create master key and certificate

	name
1	##MS_DatabaseMasterKey##

Figure 8.0.2: Master Key is created

	name	certificate_id	principal_id	pvt_key_encryption_type	pvt_key_encryption_type_desc
1	BackupEncryption	256	1	MK	ENCRYPTED_BY_MASTER_KEY

Figure 8.0.3: Certificate is created

After the master key and certificate are required, they are both backed up first so that restoration processes can be undergone with ease in the future. Database administrators will need both the key and the certificate for the database restoration. Figure 8.0.4 below shows the code to backup both the certificate and master key. Figure 8.0.5 below shows the backup file for both the master key and certificate being created.

```
BACKUP CERTIFICATE BackupEncryption
TO FILE = 'C:\Users\ghsoo\Desktop\Year 3 Sem 2\DBS\4 DBS Assignment\BackupEncryption.cert'
WITH PRIVATE KEY (
FILE = 'C:\Users\ghsoo\Desktop\Year 3 Sem 2\DBS\4 DBS Assignment\BackupEncryption.key',
ENCRYPTION BY PASSWORD = '@P@SS_WORD123')
```

Figure 8.0.4: Code to backup master key and certificate

BackupEncryption.cert	6/8/2021 5:45 PM	CERT File	1 KB
BackupEncryption.key	6/8/2021 5:45 PM	KEY File	2 KB

Figure 8.0.5: Backup files for certificate and master key

The whole Wellmeadows database is then backed up. Two versions of the backup are available to showcase the power of encryption and compression while backing up the database. If they are no encryptions, the database can be easily accessed by anybody, and this impose a database security flaw in our database. Figure 8.0.6 below shows the code to back up the database.

```
--Backup database without encryption and compression
BACKUP DATABASE Wellmeadows
TO DISK = 'C:\Users\ghsoo\Desktop\Year 3 Sem 2\DBS\4 DBS Assignment\Wellmeadows_withoutEncrypt.bak'
--Backup database with encryption and compression
BACKUP DATABASE Wellmeadows
TO DISK = 'C:\Users\ghsoo\Desktop\Year 3 Sem 2\DBS\4 DBS Assignment\Wellmeadows_withEncrypt.bak'
WITH COMPRESSION,
ENCRYPTION (ALGORITHM = AES_256, SERVER CERTIFICATE = BackupEncryption)
```

Figure 8.0.6: Code to back up Wellmeadow database

After both bak files are created, they are compared. The database with compression allows the bak file to be much smaller than the one without. Obviously, the backup that will be continued to use in the backup and restoration strategy is the one with encryption and compression enabled. Figure 8.0.7 below shows the two bak file created.



 Wellmeadows_withEncrypt.bak	6/8/2021 5:53 PM	BAK File	528 KB
 Wellmeadows_withoutEncrypt.bak	6/8/2021 5:52 PM	BAK File	5,236 KB

Figure 8.0.7: Bak files created for Wellmeadow database

For the restoration of the database, it will not be allowed without the access of the master key and the certificate. An error message as shown in Figure 8.0.8 below will be presented and the database restoration will fail.

```
Msg 33111, Level 16, State 3, Line 413
Cannot find server certificate with thumbprint '0x58DD0E10324E749DDFAD80C3F581AAD0563EE0DB'.
Msg 3013, Level 16, State 1, Line 413
RESTORE DATABASE is terminating abnormally.
```

Figure 8.0.8: Error Message while restoring database without credentials

Hence, to proceed, the backup files for the master key and certificate will have to be first created as shown in Figure 8.0.9 below. After the certificate is present, then only can the database be successfully restored as shown in Figure 8.0.10.

```
CREATE CERTIFICATE BackupEncryption
FROM FILE = 'C:\Users\ghsoo\Desktop\Year 3 Sem 2\DBS\4 DBS Assignment\BackupEncryption.cert'
WITH PRIVATE KEY (FILE = 'C:\Users\ghsoo\Desktop\Year 3 Sem 2\DBS\4 DBS Assignment\BackupEncryption.key',
DECRYPTION BY PASSWORD = '@P@SS_WORD123');
GO
```

Figure 8.0.9: Restoring Master Key and Certificate

```
Processed 640 pages for database 'Wellmeadows', file 'Wellmeadows' on file 1.
Processed 2 pages for database 'Wellmeadows', file 'Wellmeadows_log' on file 1.
RESTORE DATABASE successfully processed 642 pages in 0.013 seconds (385.516 MB/sec).
```

Figure 8.0.10: Database is restored

9.0 References

1. Beaming, 2017. *Why do we need encryption?*. [Online]
Available at: <https://www.beaming.co.uk/knowledge-base/why-do-we-need-encryption/>
[Accessed 1 August 2021].
2. Brush, K., 2021. *asymmetric cryptography (public key cryptography)*. [Online]
Available at: <https://searchsecurity.techtarget.com/definition/asymmetric-cryptography>
[Accessed 1 August 2021].
3. IBM, 2021. *What is encryption? Data encryption defined*. [Online]
Available at: <https://www.ibm.com/topics/encryption>
[Accessed 5 August 2021].
4. Oracle, 2021. *Database Security Guide*. [Online]
Available at:
https://docs.oracle.com/cd/B19306_01/network.102/b14266/auditing.htm#CHDJBDHJ
[Accessed 28 July 2021].
5. RUS, I., 2015. Technologies And Methods For Auditing Databases. *4th World Conference on Business, Economic and Management, WCBEM*, 1(1), pp. 991-999.

10.0 Appendix

10.1 Workload Matrix

Area	Chan Jia Le TP049952	Thong Kyn Hui TP053489	Soong Gim Hoy TP053242
ERD, Logical Design, Business Requirements & Assumptions	34%	33%	33%
Audited Environment	33%	34%	33%
Password Policy, Authorization Matrixes	33%	33%	34%
User Login and Authentication, Role Assigning	34%	33%	33%
Logon Trigger (One Per Person)	33%	34%	33%
DML Trigger (Two Per Person)	33%	33%	34%
Encryption Appraisal and Implementation	34%	33%	33%
Backup and Restore Strategy	33%	33%	34%
Signature	<i>Jared</i>	<i>B.F.O.V</i>	<i>PH9L99</i>