

APU3F2211CS(DA)

HAND OUT DATE : 13 May 2023

HAND IN DATE : 30 July 2023, 12 Midnight

**Group Number: 12**

**Group Members:**

|  |  |
| --- | --- |
| **Name** | **Student Number** |
| **Ferdian Marcel** | **TP058072** |
| **Ferdinand Wilson** | **TP062635** |
| **Marcell Agung Wahyudi** | **TP058650** |
| **Michael Henry** | **TP058088** |

Table of Contents

[Workload Matrix 5](#_Toc142606266)

[1. Relational Database with Integrity and Redundancy Controls 6](#_Toc142606267)

[**1.1** **Group Work (Data Dictionary)** 6](#_Toc142606268)

[**Table name**: Equipment 6](#_Toc142606269)

[**Table name**: Transaction 8](#_Toc142606270)

[**Table name**: Membership 9](#_Toc142606271)

[**Table name**: OrderItem 11](#_Toc142606272)

[**Table name**: Category 13](#_Toc142606273)

[**1.2** **Individual Work – Part A (10 Marks)** 14](#_Toc142606274)

[**1.2.1** **Ferdian Marcel [TP058072]** 14](#_Toc142606275)

[**1.2.2** **Ferdinand Wilson [TP062635]** 15](#_Toc142606276)

[**1.2.3** **Marcell Agung Wahyudi [TP058650]** 17](#_Toc142606277)

[**1.2.3** **Michael Henry [TP058088]** 19](#_Toc142606278)

[**1.3** **Individual Work – Part B (10 Marks)** 21](#_Toc142606279)

[**1.3.1** **Ferdian Marcel [TP058072]** 21](#_Toc142606280)

[**1.3.2** **Ferdinand Wilson [TP062635]** 23](#_Toc142606281)

[**1.3.3** **Marcell Agung Wahyudi [TP058650]** 25](#_Toc142606282)

[**1.3.4** **Michael Henry [TP058088]** 28](#_Toc142606283)

[2 Database Encryption 33](#_Toc142606284)

[**2.1** **Group Work** 33](#_Toc142606285)

[**2.1.1** **Decrypted Member Detail Views** 33](#_Toc142606286)

[**2.1.2** **Hidden Member Detail Views** 35](#_Toc142606287)

[**2.1.3** **Database Encryption/Backup Encryption** 36](#_Toc142606288)

[3 User Permission Management 38](#_Toc142606289)

[**3.1** **Authorization Matrix** 38](#_Toc142606290)

[**3.2** **Individual Work (10 Marks)** 40](#_Toc142606291)

[**3.2.1** **Ferdian Marcel [TP058072]** 40](#_Toc142606292)

[**3.2.2** **Ferdinand Wilson [TP062635]** 40](#_Toc142606293)

[**3.2.3** **Marcell Agung Wahyudi [TP058650]** 42](#_Toc142606294)

[**3.2.4** **Michael Henry [TP058088]** 44](#_Toc142606295)

[4 Database Auditing 45](#_Toc142606296)

[**4.1.1** **Login/Logout** 45](#_Toc142606297)

[**4.1.2** **Database Structural Changes** 47](#_Toc142606298)

[**4.1.3** **Data Changes** 48](#_Toc142606299)

[**4.1.4** **User Permission Changes** 49](#_Toc142606300)

[5 References 50](#_Toc142606301)

Table of Figures

[Figure 1 Code snippet for Transaction and Transaction View 14](#_Toc142609154)

[Figure 2 Populate Transaction Table 14](#_Toc142609155)

[Figure 3 Last 20 Days Transaction Query and Output 14](#_Toc142609156)

[Figure 4 Code snippet for Equipment and Equipment View 15](#_Toc142609157)

[Figure 5 Code snippet for Category 15](#_Toc142609158)

[Figure 6 Populate Category table 15](#_Toc142609159)

[Figure 7 Populate Equipment table 16](#_Toc142609160)

[Figure 8 Last 7 Days Transaction Query and Output 16](#_Toc142609161)

[Figure 9 Member Table and View Creation 17](#_Toc142609162)

[Figure 10 Member Decrypted Details View Creation 17](#_Toc142609163)

[Figure 11 Populate Member Table 18](#_Toc142609164)

[Figure 12 Last 30 Days Transaction Query 18](#_Toc142609165)

[Figure 13 Last 30 Days Transaction Output 18](#_Toc142609166)

[Figure 14 Order Item Table and View Creation 19](#_Toc142609167)

[Figure 15 Populate Order Item Table 19](#_Toc142609168)

[Figure 16 Last 15 Days Transaction Query and Output 20](#_Toc142609169)

[Figure 17 User Active or Inactive Trigger 21](#_Toc142609170)

[Figure 18 Examples of the Trigger 21](#_Toc142609171)

[Figure 19 Example of Trigger 22](#_Toc142609172)

[Figure 20 Delete Prevention Trigger 23](#_Toc142609173)

[Figure 21 Testing the Delete Prevention Trigger 23](#_Toc142609174)

[Figure 22 Output for the Member table after implementing the Delete Prevention Trigger 24](#_Toc142609175)

[Figure 23 Return Item Trigger 25](#_Toc142609176)

[Figure 24 Return Item Trigger User Testing 25](#_Toc142609177)

[Figure 25 Return Item Trigger User Testing [Exceeding Return Time] 26](#_Toc142609178)

[Figure 26 Transaction from User '1001' 26](#_Toc142609179)

[Figure 27 Member Unable to Return [Exceed Return Time] 26](#_Toc142609180)

[Figure 28 Return Item Success 27](#_Toc142609181)

[Figure 29 [Equipment] Table Before Returning Item 27](#_Toc142609182)

[Figure 30 [Equipment] Table After Returning Item 27](#_Toc142609183)

[Figure 31 Equipment Quantity Update Trigger 28](#_Toc142609184)

[Figure 32 Insert into Transaction 28](#_Toc142609185)

[Figure 33 Transaction Table 29](#_Toc142609186)

[Figure 34 Member Decrypted Details View 33](#_Toc142609187)

[Figure 35 Member Decrypted Views Output 33](#_Toc142609188)

[Figure 36 Row Level Security Implementation 34](#_Toc142609189)

[Figure 37 Row Level Security User Testing 34](#_Toc142609190)

[Figure 38 Member Hidden Details View 35](#_Toc142609191)

[Figure 39 Granting Member Hidden Details to Management 35](#_Toc142609192)

[Figure 40 Granting Member Hidden Details to Store Clerk 35](#_Toc142609193)

[Figure 41 Member Hidden Details Output 35](#_Toc142609194)

[Figure 42Create a new Master Key and Certificate for the backup 36](#_Toc142609195)

[Figure 43 Alter the Database to turn on the Encryption 36](#_Toc142609196)

[Figure 44 Backup the database 37](#_Toc142609197)

[Figure 45 Store Clerk Login, User, and Role Creation 40](#_Toc142609198)

[Figure 46 Grant CONTROL and GRANT privileges 40](#_Toc142609199)

[Figure 47 Give role to the user 40](#_Toc142609200)

[Figure 48 Member Login, User, and Role Creation 40](#_Toc142609201)

[Figure 49 Grant Permission to Member 41](#_Toc142609202)

[Figure 50 Member Role add Member 41](#_Toc142609203)

[Figure 51 Member successfully view Transaction table and the decrypted Member tables 41](#_Toc142609204)

[Figure 52 Implementation of member update their own details with Row-Level security 42](#_Toc142609205)

[Figure 53 Management Login, User, and Role Creation 42](#_Toc142609206)

[Figure 54 Grant Permission to ManagementRole 43](#_Toc142609207)

[Figure 55 Management Testing 43](#_Toc142609208)

[Figure 56 Management Testing [Denied Access] 43](#_Toc142609209)

[Figure 57 Database Administrator Login, User, and Role Creation with Grant Permission to Database Administator 44](#_Toc142609210)

[Figure 58 Database Administrator User Testing 44](#_Toc142609211)

[Figure 59 Database Administator User Testing Output 45](#_Toc142609212)

[Figure 60 Database Administrator User Testing 45](#_Toc142609213)

[Figure 61 Database Adminsitrator View Member Table [Denied Access] 45](#_Toc142609214)

[Figure 62 Login/Logout Auditing 45](#_Toc142609215)

# Workload Matrix

Put down the percentage contribution by each member.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Group Task** | **<Ferdian Marcel>** | **<Ferdinand Wilson>** | **<Marcell Agung>** | **<Michael Henry>** |
| Relational Database with Integrity and Redundancy Controls | 30% | 25% | 20% | 25% |
| User Administration and Authorization Matrix | 15% | 35% | 15% | 35% |
| Database Auditing | 15% | 35% | 15% | 35% |
| Database Encryption | 40% | 5% | 50% | 5% |

# Relational Database with Integrity and Redundancy Controls

## **Group Work (Data Dictionary)**

### **Table name**: Equipment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Data Type (Length) | Default Value | Note | Owner |
| Product Code | Integer (20) | - | PK (Primary Key) | * Member (View Data) * Store Clerk (Add, View, Remove Data) * Management (View Data) * DBA (Control Data) |
| Equipment Name | Varchar (100) | - |  | * Member (View Data) * Management (View Data) * Store Clerk (Add, View, Remove Data) * DBA (Control Data) |
| Price Per Unit | Float (50) | - |  | * Member (View Data) * Management (View Data) * Store Clerk (Add, View, Remove Data) * DBA (Control Data) |
| Category ID | Bigint | - | FK (Foreign Key) | * Member (View Data) * Management (View Data) * Store Clerk (Add, View, Remove Data) * DBA (Control Data) |
| Quantity In Stock | Integer (20) | 0 |  | * Member (View Data) * Store Clerk (Add, View, Remove Data) * Management (View Data) * DBA (Control Data) |
| Producing Country | Varchar (100) | - |  | * Member (View Data) * Management (View Data) * Store Clerk (Add, View, Remove Data) * DBA (Control Data) |

### **Table name**: Transaction

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Data Type (Length) | Default Value | Note | Owner |
| Product Code | Integer (20) | - | FK (Foreign Key) | * Store Clerks (View Data) * DBA (Control Data) * Management (View Data) |
| Member ID | Integer (50) | - | FK (Foreign Key) | * Store Clerks (View Data) * DBA (Control Data) * Management (View Data) |
| Transaction Code | Integer (20) | - | PK (Primary Key) | * Store Clerks (View Data) * DBA (Control Data) * Management (View Data) |
| Transaction Date | Date | - |  | * Store Clerks (View Data) * DBA (Control Data) * Management (View Data) |
| Items Purchase | Varchar (50) | - |  | * Store Clerks (Add, View Data) * DBA (Control Data) * Management (View Data) |
| Quantity Purchase | Integer (10) | - |  | * Store Clerks (Add, View Data) * DBA (Control Data) * Management (View Data) |
| Total Before Discount | Decimal (7, 2) | - |  | * Store Clerks (Add, View Data) * DBA (Control Data) * Management (View Data) |
| Total After Discount | Decimal (7, 2) | - |  | * Store Clerks (Add, View Data) * DBA (Control Data) * Management (View Data) |

### **Table name**: Membership

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Data Type (Length) | Default Value | Note | Owner |
| Member ID | Integer (100) | - | PK (Primary Key), | * Store Clerks (Add Data) * DBA (Control Data) * Management (View Data) |
| National registration ID or Passport Number | Varbinary (max) | - | Confidential | * Members (Edit) * Store Clerks (Add Data) |
| Name | Varchar (100) | - |  | * Members (Edit) * Store Clerks (Add Data, Update Data) * Management (View Data) * DBA (Control Data) |
| Address | Varbinary (max) | - | Confidential | * Members (Edit) * Store Clerks (Add Data, Update Data) |
| Phone Number | Varbinary (max) | - |  | * Members (Edit) * Store Clerks (Add Data, Update Data) * Management (View Data) * DBA (Control Data) |
| Member Status (Active, Expired) | Varchar (20) | - |  | * Clerks (Add Data, Update Data) * Management (View Data) * DBA (Control Data) |
| Login ID | Varbinary (max) | - | Confidential | * Members (Edit) * Store Clerks (Add Data) |
| Login Password | Varbinary (max) | - | Confidential | * Members (Edit) * Store Clerks (Add Data) |

### **Table name**: OrderItem

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Data Type (Length) | Default Value | Note | Owner |
| Product Code | Bigint | - | FK (Foreign Key) | * DBA (Control Data) * Management (View Data) * Store Clerk (Add, View, Remove Data) |
| Order Code | Integer | - | PK (Primary Key) | * DBA (Control Data) * Management (View Data) * Store Clerk (Add, View, Remove Data) |
| Order Date | Date | - |  | * DBA (Control Data) * Management (View Data) * Store Clerk (Add, View, Remove Data) |
| Quantity Purchase | Integer | - |  | * DBA (Control Data) * Management (View Data) * Store Clerk (Add, View, Remove Data) |

### **Table name**: Category

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Data Type (Length) | Default Value | Note | Owner |
| Category | Bigint | - | PK (Primary Key) | * DBA (Control Data) * Management (View Data) * Store Clerk (Add, View, Remove Data) |
| Category Name | Varchar (100) | - |  | * DBA (Control Data) * Management (View Data) * Store Clerk (Add, View, Remove Data) |
| Discount | Decimal (7, 2) | - |  | * DBA (Control Data) * Management (View Data) * Store Clerk (Add, View, Remove Data) |

## **Individual Work – Part A (10 Marks)**

### **Ferdian Marcel [TP058072]**

#### **1.2.1.1 SQL queries that create tables of Transaction and Transaction views**

The code snippet below creates tables and views as planned in the data dictionary.

A screen shot of a computer code

Description automatically generated

Figure 1 Code snippet for Transaction and Transaction View

#### **1.2.1.2 SQL queries that populate the table**

A white background with black text

Description automatically generated

Figure 2 Populate Transaction Table

#### **1.2.1.3 SQL query/queries that can produce details of transactions that happen in the last n days where n = {1,2,…., 7}**

User may change the @Days variable integer to change to the user needs.

A white background with black and pink text

Description automatically generated

Figure 3 Last 20 Days Transaction Query and Output

### **Ferdinand Wilson [TP062635]**

#### **SQL queries that create tables and views**

The code snippet below creates tables and views as planned in the data dictionary for Equipment.

A screenshot of a computer

Description automatically generated

Figure 4 Code snippet for Equipment and Equipment View

Next, the code snippet below is for creating the table category in the data dictionary.

A close up of a computer screen

Description automatically generated

Figure 5 Code snippet for Category

#### **SQL Queries that populate the tables**

Populate Category Table:

A screenshot of a computer code

Description automatically generated

Figure 6 Populate Category table

Populate Equipment Table:

A computer code with red text

Description automatically generated

Figure 7 Populate Equipment table

#### **SQL query/queries that can produce details of transactions that happen in the last n days where n = {1,2,…., 7}**

A screenshot of a computer

Description automatically generated

Figure 8 Last 7 Days Transaction Query and Output

The user here can change the @Days variable integer to change to the user needs. In this case, the user set the @Days to 7 which is a week as to produce details of transaction from a week prior to this date. The output of the transaction is blank, as there is no data for transaction from 7 days prior.

### **Marcell Agung Wahyudi [TP058650]**

#### **SQL queries that create tables and views**

The code snippet below creates tables and views as planned in the data dictionary for member.

A screenshot of a computer code

Description automatically generated

Figure 9 Member Table and View Creation

The next view is for member’s decrypted details, for other roles who are denied access to view members’ crucial information.

A screenshot of a computer code

Description automatically generated

Figure 10 Member Decrypted Details View Creation

#### **SQL Queries that populate the tables**

Populate Member Table (with encryption):

A screenshot of a computer code

Description automatically generated

Figure 11 Populate Member Table

#### **SQL query/queries that can produce details of transactions that happen in the last n days where n = {1,2,…., 7}**

User may change the @Days variable integer to change to the user needs.

A white background with black and pink text

Description automatically generated

Figure 12 Last 30 Days Transaction Query

Output: A screenshot of a computer

Description automatically generated

Figure 13 Last 30 Days Transaction Output

### **Michael Henry [TP058088]**

#### **SQL queries that create tables and views**

The code snippet below creates tables and views as planned in the data dictionary.

A screen shot of a computer code

Description automatically generated

Figure 14 Order Item Table and View Creation

#### **SQL Queries that populate the tables**

Populate OrderItem Table (with encryption):

A screenshot of a computer code

Description automatically generated

Figure 15 Populate Order Item Table

#### **SQL query/queries that can produce details of transactions that happen in the last n days where n = {1,2,…., 7}**

User may change the @Days variable integer to change to the user needs, in here it is 15 days.

A screenshot of a computer

Description automatically generated

Figure 16 Last 15 Days Transaction Query and Output

## **Individual Work – Part B (10 Marks)**

### **Ferdian Marcel [TP058072]**

**User Active or Inactive Trigger**

A screenshot of a computer code

Description automatically generated

Figure 17 User Active or Inactive Trigger

As we can see from the figure 17, it writes a query about the user active and inactive trigger. So basically, will trigger member table (MemberStatus) will change to their transaction, if they had a transaction between 1 month, it will change into Active, but if they did not do any transaction in entire month, it will change into Inactive.

For example,

A screenshot of a computer

Description automatically generated

Figure 18 Examples of the Trigger

In this example, I will use number 4 which the MemberID 1004 it said the MemberStatus is Inactive because he did not do any transaction on entire months.

A screenshot of a computer

Description automatically generated

Figure 19 Example of Trigger

As we can see that when I insert the new transaction on member 1004 on number 8 on transaction table, the status of the Member will change into Active.

### **Ferdinand Wilson [TP062635]**

**Delete Prevention Trigger**

A screenshot of a computer

Description automatically generated

Figure 20 Delete Prevention Trigger

The code in the above figure is showing the Trigger function of row deletion from the member table. The Trigger is going to prevent the unauthorized accesses in deleting the row from the member table. The code that is provided is going to raise an error notification for the user when they want to delete the row from the member table. The “RAISERROR” function is used for showing the error message called “No one allowed to delete member row” when running a delete function.

**A screenshot of a computer

Description automatically generated**

Figure 21 Testing the Delete Prevention Trigger

The code snippet above is used for testing the new Delete Prevention Trigger that is implemented in the system. In this case, the user here wants to delete the row which has ‘1001’ from the member table, which is not allowed. In return, the system raises an error message showing that the user is not able to delete the table.

**A screenshot of a computer

Description automatically generated**

Figure 22 Output for the Member table after implementing the Delete Prevention Trigger

To test the Trigger further, the select all function is used as to view that the Trigger is working fine. In a nutshell, the row which contains member id of ‘1001’ still exists with the other details untouched and unbothered. In conclusion, the Trigger is working perfectly for the data base system of the project team.

### **Marcell Agung Wahyudi [TP058650]**

**Return Item Trigger**

A screenshot of a computer program

Description automatically generated

Figure 23 Return Item Trigger

The code above is a return item trigger, meaning when a member deletes an item in their transaction table, the trigger will add the amount purchased back to the equipment table, but this does not always come true, there is a specific condition, which is if the executing role is ‘Member’ and also if the TransactionDate is three days from current date, if these two conditions are not met, an error message will be printed out saying “Transaction cannot be deleted. It has exceeded the return time OR executing as Non-Member Role.” This is to ensure accidental deletion by other roles are not possible.

The following is the examples of the possible outputs:

1. Executing as Management:

A screenshot of a computer

Description automatically generated

Figure 24 Return Item Trigger User Testing

Although TransactionCode = 11 is able to be returned since it is in range for returning time (3 days from GETDATE()).

A screenshot of a computer

Description automatically generated

Figure 25 Return Item Trigger User Testing [Exceeding Return Time]

It is unable to be deleted since the executing user is not a member, rather a management role.

1. Executing as Member (Out of return time):

Below are the transactions by member ‘1001’

A screenshot of a computer

Description automatically generated

Figure 26 Transaction from User '1001'

If the member were to the transaction with [TransactionCode] = 1, they are not able to because the transaction date is not between 3 days the current date(8/6/2023) at the time documented.

A screen shot of a computer

Description automatically generated

Figure 27 Member Unable to Return [Exceed Return Time]

1. Executing as Member (In Return Time):

However, if the date is in between 3 days of the TransactionDate (TransactionCode = 11):

A screenshot of a computer

Description automatically generated

Figure 28 Return Item Success

The deletion is success.

[Equipment] Table before returning the item:

A screenshot of a computer

Description automatically generated

Figure 29 [Equipment] Table Before Returning Item

[Equipment] Table after returning the item:

A screenshot of a computer

Description automatically generated

Figure 30 [Equipment] Table After Returning Item

It is seen that the current quantity in stock is 11 in difference, just as the quantity previously purchased by the member.

### **Michael Henry [TP058088]**

**Equipment Quantity Update Trigger (Transaction)**

A screenshot of a computer code

Description automatically generated

Figure 31 Equipment Quantity Update Trigger

The above SQL trigger depicts an automated procedure in which the equipment table is updated when a new transaction is added. This method is based on the assumption of a continually accessible supply of equipment. The trigger enables the deduction of equipment quantity based on the specifics of the new transaction for each incoming transaction. This dynamic approach ensures that equipment records are correct and up to date, in accordance with the changing transactional scenario. This trigger accelerates data maintenance and improves overall system performance by smoothly integrating amount revisions into the transaction process, all while retaining the assumption of continually accessible equipment quantities.



Figure 32 Insert into Transaction

A white rectangular table with numbers and text

Description automatically generated with medium confidence

Figure 33 Transaction Table

A white rectangular table with numbers and text

Description automatically generated with medium confidence

A white paper with numbers and text

Description automatically generated

As seen in the result, when a new transaction is added to the transaction table, involving the purchase of a product labelled 3003 and totalling three purchases, the quantity of the equipment table decreased from 20 to 17 units.

**Equipment Quantity Update Trigger (Item Order)**

A computer screen shot of a code

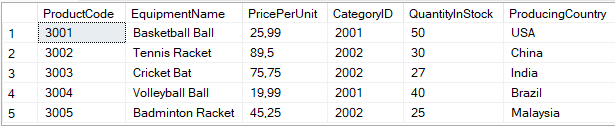
Description automatically generated

The trigger described above, like the one mentioned previously, modifies the equipment number within the designated equipment table. This operation is triggered when a new order is added to the order item database. Any addition of an order triggers an automatic modification in the equipment quantity, ensuring precise and synchronized records. This streamlined approach not only improves data consistency but also optimizes operational efficiency, continuing the trigger's former duty of keeping records up to date within the system.



A white paper with numbers and text

Description automatically generated with medium confidence



A white paper with numbers and numbers

Description automatically generated

The shown output shows that when new order data with a product code of 3003 and a quantity of 10 units is introduced into the order item database, a corresponding update is triggered in the quantity field of the equipment table to appropriately reflect the change. The initial equipment quantity with product code of 3003 in the equipment table is 17 and after the insertion it turns into 27.

**Tax and Discount Trigger**

A screenshot of a computer code

Description automatically generated

The tax and discount trigger computes transaction pricing post-tax and post-discount with automated precision. When a new transaction is entered, the trigger begins by referencing the equipment table to gather essential details such as country and price. Concurrently, it searches the category table for discount information. Provided with this information, the trigger methodically calculates the final transaction price. The starting price is the pure transaction total, free of any tax or discount factors. Following that, the original amount is taxed and discounted, resulting in the ultimate transaction price. The trigger conducts a symphonic symphony of data from several tables, delivering correct post-processing statistics that account for both tax and discount effects. This technique exhibits the incorporation of dynamic pricing aspects into transactions, improving the financial accuracy and operational efficiency of the system.



A screenshot of a table

Description automatically generated

The displayed result above is an example of an automated process in which the addition of a new transaction causes automatic computations of pre-discount and post-discount totals. The sum after taxation is shown in the "Total Before Discount" column, whereas the figure after taxation and discount application is shown in the "Total After Discount" column. The discount value comes from the category table's "Discount" column, while the tax value comes from the equipment table's "Country" column. This streamlined solution harmonizes data from different sources, allowing for precise and dynamic pricing estimates that improve transactional clarity while taking discount and tax implications into account.

# Database Encryption

## **Group Work**

### **Decrypted Member Detail Views**

Create a view that will show member full details including automatically decrypting the encrypted values if run by a user in the Members role. This view should be accessible by Members only. It must also show only the users’ own details only (implement row level security).

Firstly, a data view with the decrypted member details is created:

A screenshot of a computer code

Description automatically generated

Figure 34 Member Decrypted Details View

The output are as follows:

A screenshot of a computer

Description automatically generated

Figure 35 Member Decrypted Views Output

In order to limit privilege for the members to see only their own data, row-level security is implemented, the following code implements row-level security for Member role:

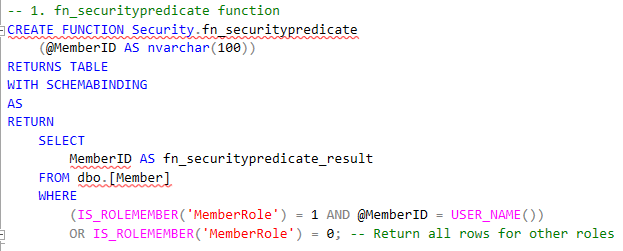


Figure 36 Row Level Security Implementation

The code above will return the MemberID which will be used in the later security policy as input to only show the data in which the current user is executing as, if the current user’s role is not ‘MemberRole’, the function will not work, meaning that higher status roles, such as store clerks and management can see the full Member table.

Example, if the user ‘1001’ executes SELECT \* FROM memberDecryptedDetails and SELECT \* FROM [Transaction], the only information shown is about the MemberID with ‘1001’:

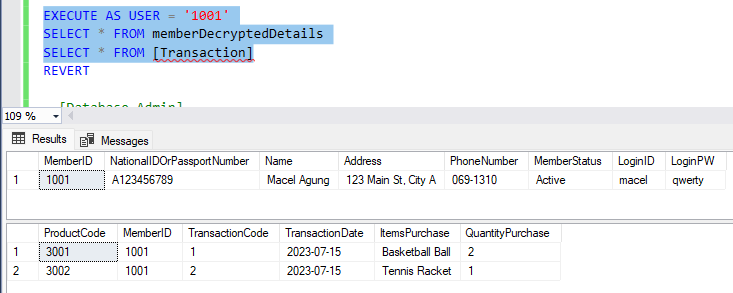


Figure 37 Row Level Security User Testing

### **Hidden Member Detail Views**

Create a view that will show member full details and hides any encrypted values. This view can be accessed by Store Clerks and Management only.

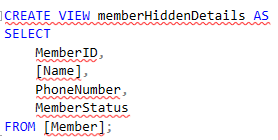


Figure 38 Member Hidden Details View



Figure 39 Granting Member Hidden Details to Management



Figure 40 Granting Member Hidden Details to Store Clerk

The output are as follows:

A screenshot of a computer

Description automatically generated

Figure 41 Member Hidden Details Output

The code ensures that it will only select columns that are not encrypted.

### **Database Encryption/Backup Encryption**

In this part, we work to do the encryption for the database backup.

A computer code with text

Description automatically generated with medium confidence

Figure 42Create a new Master Key and Certificate for the backup

In this database, we use the new master key and new certificate to back up the database. The master key is protected by an Encryption by a password ‘qwert’ and the Certificate just a normal certificate “CertmasterDB”. After that we create the encryption key with algorithm AES\_128 by the server of the certificate “CertMasterDB”. After that we Alter the DB to turn on the encryption using this query below.



Figure 43 Alter the Database to turn on the Encryption

After Creating a Master key, Certificate for the backup and alter it to turn on the Encryption, we backup the database into a disk in the laptop and we Backup the the “CertMasterDB” Certificate and the Private Key.

A screenshot of a computer program

Description automatically generated

Figure 44 Backup the database

Before we do the restore of the backup we need to drop/delete the database first, after dropping or deleting the database, we can easily try to restore the database using the last 5-line query on the figure 44.

# User Permission Management

## **Authorization Matrix**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Account | Type | Object(s) | Privilege(s) | Owner (grup kita yang nanti coding) |
| Member Role | SQL Role | Table: Equipment | SELECT | Ferdinand Wilson (TP062636) |
| Column: Membership (National Registration ID or Passport Number, Name, Address, Phone Number, Login ID, Login Password) | UPDATE |
| Table: Transaction | SELECT |
| Store Clerks | SQL Role | Table: Equipment | CONTROL | Ferdian Marcel (TP058072) |
| Column: Transaction (Product Code, Member ID, Transaction Code, Transaction Date, Items Purchase, Quantity Purchase) | SELECT |
| Table: Membership | INSERT |
| Column: Membership (Name, Address, Phone Number, Member Status (Active, Expired) | UPDATE |
| Database Administrators | SQL Role | Table: Transaction | CONTROL | Michael Henry |
| Column: Membership (Member ID, Name, Member Status) | CONTROL |
| Table: Equipment | CONTROL |
| Management | SQL Role | Database: DBS\_Assignment | CONTROL, GRANT | Marcell Agung W |

## **Individual Work (10 Marks)**

### **Ferdian Marcel [TP058072]**

SQL Code for Management Role:

First, create the user.

A close-up of text

Description automatically generated

Figure 45 Store Clerk Login, User, and Role Creation

When the Role and user has been created, grant the CONTROL and GRANT privileges.

A screenshot of a computer screen

Description automatically generated

Figure 46 Grant CONTROL and GRANT privileges

Also, after grant the control to the store clerk, add the user to the role using Alter Role



Figure 47 Give role to the user

### **Ferdinand Wilson [TP062635]**

A computer screen shot of a computer code

Description automatically generated

Figure 48 Member Login, User, and Role Creation

To start with, the User Permission Management coding part can be done with the help of the authorization matrix that has been made previously, as to help the preparation of the coding part. Next, the login is created with the login id “1001” with the password called “pisangkeju” as to access as the user named. Following that, the role is being created for the member, called “MemberRole”.

A screenshot of a computer code

Description automatically generated

Figure 49 Grant Permission to Member

Furthermore, the member is going to be able to view the whole Equipment table, based on the authorization table that is made prior to this coding stage. Next, the member table can be viewed and updated by the member. Deny update is used for denying the member role from modifying the member ID in the member table. The grant deletes and select on transaction table is given for member role as well. Next, the grant control on the certificate is used for member role as a security function. Next, the members can view their own decrypted details.



Figure 50 Member Role add Member

The member ‘1001’ is added to the member role list. In this list, there will be more than one member later.

A screenshot of a computer

Description automatically generated

Figure 51 Member successfully view Transaction table and the decrypted Member tables

The code is then tested for the user ‘1001’ to view their own personal member details from the member table. Then, the details like the transaction(s) that is or are under the same member id, which is ‘1001’, can also be seen.

A screenshot of a computer

Description automatically generated

Figure 52 Implementation of member update their own details with Row-Level security

The code allows the member ‘1001’ to update their own details. The output after the update is shown as well. The user ‘1001’ is granted permission to update their own phone number, address, passport number or national id, name, login id, and finally their own login password.

### **Marcell Agung Wahyudi [TP058650]**

SQL Code for Management Role:

Firstly, the user has to be created.

A computer code with text

Description automatically generated with medium confidence

Figure 53 Management Login, User, and Role Creation

Then, next the [Management] role is created and granted the CONTROL and GRANT privileges, also the user is added to the role.

A close up of text

Description automatically generated

Figure 54 Grant Permission to ManagementRole

Finally, to test the role:

A screenshot of a computer

Description automatically generated

Figure 55 Management Testing

A screenshot of a computer

Description automatically generated

Figure 56 Management Testing [Denied Access]

As seen from the output, management is able to read member hidden details and transaction data view but unable to read to encrypted values.

### **Michael Henry [TP058088]**

A screenshot of a computer code

Description automatically generated

Figure 57 Database Administrator Login, User, and Role Creation with Grant Permission to Database Administator

The authorisation of the database administrator is closely linked to a pre-established authorization matrix. The scope of their privileges is defined by this matrix. The administrator, in particular, has the authority to perform Data Definition Language (DDL) queries across all databases. There is, however, one important caveat: access to confidential member data is prohibited. A series of procedures is taken to implement this authorisation setup. First, the administrator's login credentials, and user profile are built, and then a particular role is created. Following that, a granular granting process occurs, allocating control privileges within the scope of the role. Furthermore, customized select rights are methodically provided to certain columns. This stratified method guarantees that only pertinent information is available. Finally, the administrator is assigned to the appropriate role, completing the framework of their approved skills. This rigorous procedure ensures that permissions are not only aligned with the matrix, but also that data security and operational integrity are prioritized.

A group of text on a white background

Description automatically generated

Figure 58 Database Administrator User Testing

A screenshot of a member test

Description automatically generated

Figure 59 Database Administator User Testing Output

A white background with black text

Description automatically generated

Figure 60 Database Administrator User Testing

A close up of text

Description automatically generated

Figure 61 Database Adminsitrator View Member Table [Denied Access]

As shown in the output above, DBA role able to alter the member table but they cannot see the member confidential details. If they are trying to access the member table, the SQL will throw a permission denied message.

# Database Auditing

## **Login/Logout**

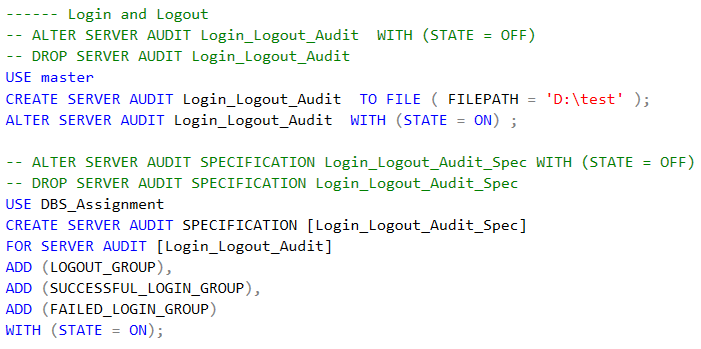
****

Figure 62 Login/Logout Auditing

The provided code snippet demonstrates how to set up an audit system to track both login and logout events in a server environment. This feature improves the system's security and accountability by meticulously monitoring user activities. Let's get into the specifics of this code:

The first step in starting this audit trail is to create a server audit called "Login\_Logout\_Audit." This audit is meticulously connected to a specific file path on the computer's drive. This link to a file system location guarantees that all relevant audit data is persistently maintained, allowing for further analysis and review.

Following the creation of the server audit, the establishment of a server audit specification is the next critical step. This specification outlines the audit's precise scope and characteristics. This standard delineates three distinct groups of events:

* + - 1. **LOGOUT\_GROUP**: This category contains events associated with user logouts. It includes a detailed record of logout operations, revealing when and by whom logout actions are initiated. This is critical for tracking user sessions and identifying illegal access attempts.
      2. **SUCCESSFUL\_LOGIN\_GROUP**: This section is about successful login attempts. It meticulously records occasions in which users get permitted access to the system. This information is critical for maintaining accountability and monitoring user interactions to ensure that security standards are followed.
      3. **FAILED\_LOGIN\_GROUP**: The third group is for unsuccessful login attempts. It logs incidents in which users' attempts to access the system are blocked due to invalid credentials or other circumstances. The extensive logging of failed login attempts can act as an early warning system for future security breaches, allowing administrators to take preventative measures to protect the system.

A computer screen shot of a code

Description automatically generated

Following that, a critical procedure happens in order to access and analyse the audit data's contents. This includes declaring a variable to hold the exact file location linked with the audit records. Following that, there is a purposeful curation of specific bits of information, with the emphasis on carefully selecting significant data points from the audit records. Once the file path is obtained, the following step is to carefully extract essential data pieces from the audit records. This procedure is distinguished by the careful selection of material that is relevant in the context of the audit's aim. Rather to overburdening the study with irrelevant details, the emphasis is squarely on recognizing and extracting significant insights.

A screenshot of a computer program

Description automatically generated

The output shows that the login and logout records were successfully retrieved. "1001" identifies a member, with "LGIS" indicating successful login and "LGO" signaling logout events.

## **Database Structural Changes**

A screenshot of a computer program

Description automatically generated

Same as before, the audit procedure begins with the addition of a server audit. Unlike previous cases, a database audit specification is used over a server specification audit. This customized specification includes the DATASE\_OBJECT\_CHANGE\_GROUP, indicating an intentional focus on monitoring changes within database objects. By selecting this group, the audit becomes more sensitive to changes in database components. This sophisticated approach emphasizes the commitment to gathering exact and relevant data, boosting the system's security and integrity by tracking alterations with greater accuracy within the authorized database.

**A screen shot of a computer code

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

Same as before, to read the audit, a variable to hold the file path is created. Then, some selection of information is chosen from the audit file. Furthermore, as seen in the output, all database structure such as create and alter is recorded in the audit file.

## **Data Changes**

**A screenshot of a computer program

Description automatically generated**

The procedure used for implementing the data changes audit is similar to that of other database audits. Nonetheless, a distinguishing feature develops in the shape of audit specifications. This specification, designed for the data changes audit, cover a wide range of activities, including INSERT, UPDATE, DELETE, and SELECT. This specification covers the entire range of data manipulation, monitoring the insertion, alteration, removal, and retrieval of information from a database. The audit, by embracing these critical functions, carefully monitors every aspect of data dynamics, enhancing security, compliance, and accountability. This complex approach indicates a dedication to thorough control, preserving database integrity and protecting against illegal changes.

A computer screen shot of a code

Description automatically generated

A screenshot of a computer

Description automatically generated

The procedure, as previously stated, remains unchanged. A variable is set up to store the file path in order to access the audit information. Following that, a purposeful extraction of certain data from the audit file occurs. Moreover, as seen in the output, some activities such as select, and insert is recorded in the audit file.

## **User Permission Changes**

A computer screen shot of text

Description automatically generated

The audit specification is adjusted to include the SCHEMA\_OBJECT\_PERMISSION\_CHANGES\_GROUP when auditing changes to user permissions. This option makes it easier to keep track of changes to schema-level object permissions. Notably, the selection for constructing the audit mechanism, including the formation of the server audit and the definition of the file path, is same across all three audits outlined before. This consistent approach emphasizes the methodical character of the auditing process, ensuring that each audit, while unique in focus, follows to a defined framework that simplifies implementation and promotes cohesive audit record management.

A computer code with text

Description automatically generated

A screenshot of a computer

Description automatically generated

As seen from the output above, some action such as granting and denying permission to user is recorded in the audit file.

# References

Backup Location: C:\College Assignment\BackupDatabase