

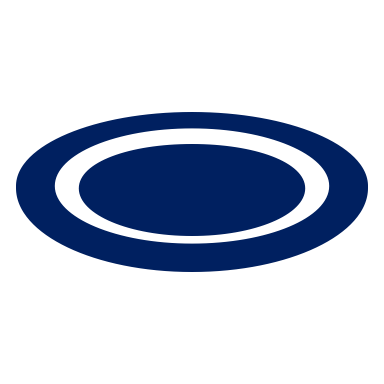
***Computer Science & Engineering Department***

**INTERNATIONAL UNIVERSITY**

**Ho Chi Minh City, Vietnam**

***Principle of Database Management***

***REPORT***



**〜〜〜〜〜〜〜〜**

**〜〜〜〜〜〜〜〜**

Instructor: Mrs. Loan Nguyen

Topic: Restaurant Management System

Semester II - Academic year: 2018-2019

RESTAURANT MANAGEMENT SYSTEM

**INDEX**

|  |  |
| --- | --- |
| Title | Page |
| *Index* | 2 |
| Members and Roles | 3 |
| Abstraction & Introduction | 4- 5 |
| Entity Relationship Diagram | 6- 8 |
| Relational Database Schema | 9- 12 |
| Database | 13- 19 |
| Programming | 20- 23 |
| Conclusion | 24- 25 |

**I. MEMBERS & ROLES**

|  |  |  |
| --- | --- | --- |
| NAME | ID | ROLE |
| ĐOÀN Ý NHI | ITITIU17025 | * Programming to connect database with java. * Drawing ERD. * Writing ERD part and Programming part of report |
| NGUYỄN THẾ ANH | ITITIU17024 | * Creating database * Writing initial PROPOSAL and MIDTERM report * Drawing ERD. * Writing Database part in report. |
| HỒ ĐẶNG PHƯƠNG NGỌC | ITITIU17058 | * Convert ERD into Relational Database Schema. * Writing Relational Database Schema part of report. * Writing Introduction + Conclusion part of report. |

**II. ABSTRACTION & INTRODUCTION**

**A. ABSTRACTION**

Since the technology has made a great expansion in our world and has delivered a numerous influence on our human being lives, our expectation is getting increasingly high. Human being has reached to a new level in which we can rely on computers and technology from normal tasks to complicated computation in real life. Together with that is the growth of hospitality industry, especially restaurant services, which demands a new innovation in processing orders and serving customers. For that reason, by implementing technology into the process of enhancing dining experience for customers, our system comes into life with the expectation of convenience as well as accuracy in receiving orders, serving meals to customers, processing bill payment, and managing restaurant’s information easily, since the old-fashioned style of taking orders brings about a high probability of paper wastes as well as misinterpretation of handwriting orders.

**B. INTRODUCTION**

Nowadays, hospitality is on a rise and in this 4.0 Industrialize Era, service industry demands more than just serving and pleasing customers with old- fashioned style.

Human beings are increasingly dependent on tablets and wireless technology. Having taken this into notice, restaurants have been a prime initiator to implement such techniques and devices associated with that hi-tech revolutionization into management system. Especially, this would bring on a bargain to restaurants since food nowadays is the service with highest consumption counted every second.

Although there are still many restaurants with out of serving style such as using pen and papers to take orders, which is such a waste of time as well as a cause of the inaccuracy of order elements (misinterpretation) or the rise of paper loss- waste of paper which leads to pollution, the others have been prepared carefully for the innovation in the service industry, which helps the business to harmonize perfectly with the market, and provides customers with new experiences such as ordering food via tablets or wireless devices reducing the time spent on this duration and also minimizing the misinterpretation in ordering food and transferring orders, and the process of paying for a meal or altering restaurant menu or system would also no longer be a limitation.

It has been for a period of time until now that such kinds of restaurant system have been discovered and developed into abundant versions that can be easily come across in the nowadays hospitality market since this area is also a competent industry. Compared to those existing versions, our system is slightly less modern and professional. However, this project mainly aims to provide our group members with fundamental database and database management, so we focused mainly on manage the restaurant database following these steps respectively: drawing ERD, convert ERD into Relational Database Diagram, create database (create tables and query data), and finally connect the database to java to display query by java.

The steps will be conducted based on the initial idea of Steve’ s demands for his restaurant management system:

* Customer chooses an empty table in the restaurant to have his meal; after having entered, the guest will make order(s) directly from a computer placed right on the table; the menu consists of all COURSE of the restaurant.
* After the customer finishing ordering, the order(s) will be immediately transferred into the kitchen. Now the Chefs or COOK will be assigned to those orders and COOK must compute logical time span for each order as well as the appropriate orders of them.
* After the order(s) has been cooked, the COOK deliver the order(s) to the Customer table via a deliver system.
* After Customer finishing his meals, he sends a signal to the cashier so that he will be sent a bill via computer (a bill may consist of many orders since he will pay the bill last and there may be one after another order during the meal) and use his card account to pay the meals.

**III. ENTITY RELATIONSHIP DIAGRAM**

This entity relationship diagram is based on the demand of Steve restaurant management system. Basically, Steve wants a restaurant which is mostly automatically managed by a computer system and takes advantages of as less human labor as possible. We have developed a system consisting of five entities and five relationships respectively to satisfy his requirements. Considering the entity relationship diagram below:

A close up of a piece of paper

Description automatically generated

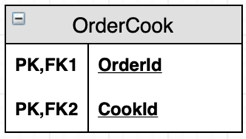
* There are five entities in our system: CUSTOMER, COURSE, COOK, BILL, ORDER and five relationships: MAKES, CONSISTS OF, RECEIVES, COSTS, PAYS.
* In each entity:
* **CUSTOMER:** CUSTOMER consists of **CusID**, which is the primary key to recognize each customer account whenever he/she enters the restaurants; besides there are CusName, CusCardInfo, and Password, which gives more assistance for account verification for customer in the restaurant database. If the customer hasn’t had an account in our system, he/she will input the information as above (includes CusName, CusCardInfo, and Password; **CusId** will be automatically assigned for customers by the system according to the number of customers having entered the restaurant) and will be automatically added into the system of the restaurant.
* **COOK:** COOK consists of **CookId**, which is the primary key to recognize each cook in the database of the restaurant; besides there are CookName and Schedule for specific information of cook in the restaurant database.
* **COURSE:** COURSE consitss of **CourseId**, which is the primary key of Course for distinguishing different courses in the database of the restaurant; besides there are CourseName, CourseDescription, CoursePrice as well as CourseDuration as normal attributes of COURSE.
* **BILL:** BILL comprises of **BillId** to take note of the bill payment at which time and for which customer, which order
  + - It has the relationship (1, 1) to (1,1) with Customer via the relationship (PAYS) which means that a customer just can pay only one bill and one bill just can belong to one customer.
* **ORDERS:** ORDERS consists of **OrderId** functioned as primary key to ORDER for specification of which order has been computed, to differentiate among different orders since there are a lot of orders held in the database; besides are normal attributes including OrderDate, OrderDuration, OrderPrice, OrderDescription.
  + - It has the relationship (1, 1) to (1, n) with Customer via the relationship (MAKES) which means one customer can make at least one order and at most n (many) orders and one order just can belong to one customer.
    - It has the relationship (1, n) to (1, n) with Course via the relationship (CONSISTS OF) which means on order can have at least one course and at most n (many) course and vice versa, one course can belong to at least one order and at most n (many) orders.
    - It has the relationship (1, n) to (1, n) with Cook via the relationship (RECEIVES) which means that one order can be cooked by at least one cook and at most n (many) cooks and vice versa, one cook can cook receive at least one order and at most n (many) orders.
    - It has the relationship (1, n) to (1,1) with Bill via the relationship (COSTS) which means that one bill can be the cost of at least one order and at most n (many) orders and one order just can cost (or belongs to) one bill.

A close up of a map

Description automatically generated**IV. REALATIONAL DATABSE SCHEMA**

A screenshot of a cell phone

Description automatically generated



From the initial diagram or this diagram, we can easily transfer into Relational Database Schema:

**Customer- Schema**= (**CusId**, CusName, CusCardInfo, Password)

**Cook- Schema**= (**CookId**, CookName, Schedule)

**Course- Schema**= (**CourseId**, CourseName, CourseDescription, CoursePrice, CourseDuration)

**Bill- Schema**= (**BillId**, CusId, BillDate, Tax, Discount, Total)

From Bill- Schema. CusId to Customer-Schema. CusId

**Orders- Schema**= (**OrderId**, CusId, BillId, OrderDate, Tax, Discount, Total)

From Orders- Schema. CusId to Customer- Schema. CusId

From Orders- Schema. BillId to Bill- Schema. BillId

**OrderCourse- Schema**= (**OrderId, CourseId**)

From OrderCourse- Schema. OrderId to Order- Schema. OrderId

From OrderCourse- Schema. CourseId to Course- Schema. CourseId

**OrderCook- Schema**= (**OrderId, CookId**)

From OrderCook- Schema. OrderId to Order- Schema. OrderId

From OrderCook- Schema. CookId to Receives- Schema. CookId

**EXPLANATION:**

1. **STEP 1: Write down all the regular entities and their simple attributes**

We have five strong entities which are CUSTOMER, COOK, COURSE, ORDER, BILL and their following simple attributes that have been listed in former part ( III. ENTITY RELATIONSHIP DIAGRAM)

**RESULT:**

**Customer- Schema**= (**CusId**, CusName, CusCardInfo, Password)

**Cook- Schema**= (**CookId**, CookName, Schedule)

**Course- Schema**= (**CourseId**, CourseName, CourseDescription, CoursePrice, CourseDuration)

**Bill- Schema**= (**BillId**, BillDate, Tax, Discount, Total)

**Orders- Schema**= (**OrderId**, OrderDate, Tax, Discount, Total)

1. **STEP 2: Write down all weak entities and their simple attributes**

We skip this step since we don’t have any weak entity.

1. **STEP 3: Mapping Binary 1-1**

We have the relationship PAYS between CUSTOMER and BILL is 1-1 relationship => Add Customer.CusId as Foreign Key to Bill (Bill.CusId)

**RESULT:**

**Bill- Schema**= (**BillId**, CusId, BillDate, Tax, Discount, Total)

From Bill- Schema. CusId to Customer-Schema. CusId

1. **STEP 4: Mapping 1- N**

We have two 1- N relations BILL- ORDER and CUSTOMER- ORDER

* BILL- ORDER:

Choose bill and customer to be N side, the other is bill =>

* Add primary key of CUSTOMER (CusId) to BILL (CusId) as foreign key.
* Add primary key of BILL (BillId) to Order (BillId) as foreign key.

**RESULT:**

**Orders- Schema**= (**OrderId**, CusId, BillId, OrderDate, Tax, Discount, Total)

From Orders- Schema. CusId to Customer- Schema. CusId

From Orders- Schema. BillId to Bill- Schema. BillId

1. **STEP 5: Mapping M- N**

We have two M- N relations: ORDER- COOK (RECEIVES) and ORDER- COURSE(CONSISTS\_OF) (as having explained in part (III. Entity Relation Diagram)) => TWO NEW RELATIONS HAVE TO BE CREATED.

* For ORDER- COURSE: we create new relation

OrderCourse= (**OrderID, CourseId**) with:

* OrderId which is the primary key in ORDERS is added to this new relation as Foreign Key.
* CourseId which is the primary key in COURSE is added to this new relation as Foreign Key.
* For ORDER- COOK: we create new relation

OrderCook= (**OrderID, CookId**) with:

* OrderId which is the primary key in ORDERS is added to this new relation as Foreign Key.
* CookId which is the primary key in COOK is added to this new relation as Foreign Key.

**RESULT:**

**OrderCourse- Schema**= (**OrderId, CourseId**)

From OrderCourse- Schema. OrderId to Orders- Schema. OrderId

From OrderCourse- Schema. CourseId to Course- Schema. CourseId

**OrderCook- Schema**= (**OrderId, CookId**)

From OrderCook- Schema. OrderId to Orders- Schema. OrderId

From OrderCook- Schema. CookId to Cook- Schema. CookId

**NOTES:**

We have no multivalued attributes, specialization or generalization, so we skip this part of converting from ERD to Relational Database Schema.

**\*\*\*ADVANTAGES AND DISADVANTAGES OF ERD AND RELATIOBAL DIAGRAM:**

Entity relationship diagram is effective tool for designers to keep up with the relationships between entities and attributes; it is simple as well since the necessary components for it are just entities and attributes. It is also easily converted to any other kind of data models.

However, there are some pay offs. Due to its simplicity, it is more suitable for high level design because it is so general, some information is lost or hidden on the diagram, and there are limitations in relationships compared to relational models. Due to this, it would cause misunderstanding for beginners, and low- level designers in database area.

**V. DATABASE**

**RECALL: RELATIONAL DATABASE SCHEMA**

**Customer- Schema**= (**CusId**, CusName, CusCardInfo, Password)

**Cook- Schema**= (**CookId**, CookName, Schedule)

**Course- Schema**= (**CourseId**, CourseName, CourseDescription, CoursePrice, CourseDuration)

**Bill- Schema**= (**BillId**, CusId, BillDate, Tax, Discount, Total)

From Bill- Schema. CusId to Customer-Schema. CusId

**Orders- Schema**= (**OrderId**, CusId, BillId, OrderDate, Tax, Discount, Total)

From Orders- Schema. CusId to Customer- Schema. CusId

From Orders- Schema. BillId to Bill- Schema. BillId

**OrderCourse- Schema**= (**OrderId, CourseId**)

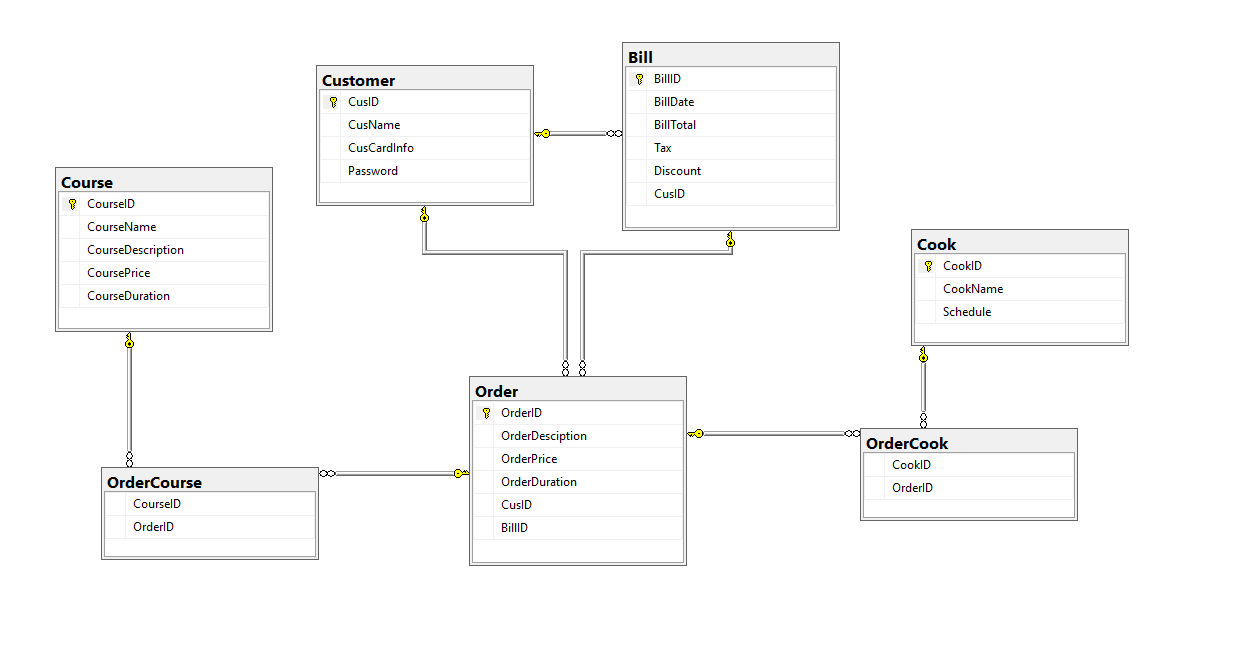
From OrderCourse- Schema. OrderId to Orders- Schema. OrderId

From OrderCouse- Schema. CourseId to Course- Schema. CourseId

**OrderCook- Schema**= (**OrderId, CookId**)

From OrderCook- Schema. OrderId to Orders- Schema. OrderId

From OrderCook- Schema. CookId to Cook- Schema. CookId

**DESCRIPTION:**

**s**

**/\*\*\*\*\*\* OBJECT: TABLE [DBO].[CUSTOMER] Script Date: 5/7/2019 1:16:59 PM \*\*\*\*\*\*/**

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Customer](

[CusID] [nvarchar](50) NOT NULL,

[CusName] [nvarchar](50) NULL,

[CusCardInfo] [nvarchar](50) NULL,

[Password] [nvarchar](50) NULL,

CONSTRAINT [PK\_Customer] PRIMARY KEY CLUSTERED

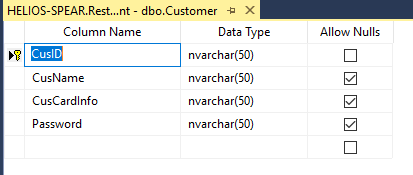
(

[CusID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO



**Table Customer has 4 attributes CusId (PRIMARY KEY), CusName, CusCardInfo, Password; all four of them are in type nvarchar with max length is 50, and the primary key of the table is CusId.**

**/\*\*\*\*\*\* OBJECT: TABLE [DBO].[COURSE] Script Date: 5/7/2019 1:16:59 PM \*\*\*\*\*\*/**

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Course](

[CourseID] [nvarchar](50) NOT NULL,

[CourseName] [nvarchar](100) NULL,

[CourseDescription] [nvarchar](510) NULL,

[CoursePrice] [real] NULL,

[CourseDuration] [int] NULL,

CONSTRAINT [PK\_Course] PRIMARY KEY CLUSTERED

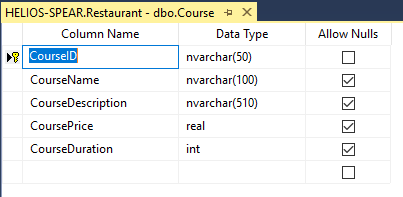
(

[CourseID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO



**Table Course has five attributes: CourseId: nvarchar with max length is 50, CourseName: nvarchar with max length is 100, CourseDescription: nvarchar with max length is 510, CoursePrice: real and CourseDuration: int.**

**/\*\*\*\*\*\* OBJECT: TABLE [DBO].[COOK]** **Script Date: 5/7/2019 1:16:59 PM \*\*\*\*\*\*/**

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Cook](

[CookID] [nvarchar](50) NOT NULL,

[CookName] [nvarchar](100) NULL,

[Schedule] [nvarchar](50) NULL,

CONSTRAINT [PK\_Cook] PRIMARY KEY CLUSTERED

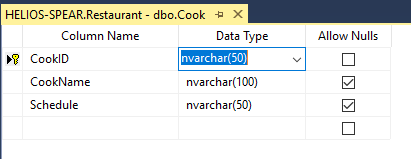
(

[CookID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO



**The table Cook has three attributes: CookId(PRIMARY KEY) : nvarchar with max length is 50, CookName: nvarchar with max length is 100, Schedule: nvarchar with max length is 50.**

**/\*\*\*\*\*\* OBJECT: TABLE [DBO].[BILL] Script Date: 5/7/2019 1:16:58 PM \*\*\*\*\*\*/**

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Bill](

[BillID] [int] NOT NULL,

[BillDate] [nvarchar](50) NULL,

[BillTotal] [real] NULL,

[Tax] [real] NULL,

[Discount] [real] NULL,

[CusID] [nvarchar](50) NULL,

CONSTRAINT [PK\_Bill] PRIMARY KEY CLUSTERED

(

[BillID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

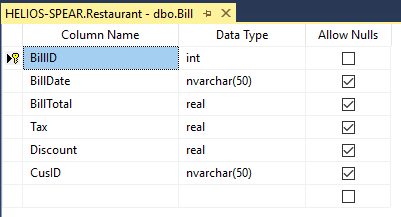
ALTER TABLE [dbo].[Bill] WITH CHECK ADD CONSTRAINT [FK\_Bill\_Customer] FOREIGN KEY([CusID])

REFERENCES [dbo].[Customer] ([CusID])

GO

ALTER TABLE [dbo].[Bill] CHECK CONSTRAINT [FK\_Bill\_Customer]

GO



**The table Bill has five simple attributes: BillId(PRIMARY KEY): int, BillDate: nvarchar with max length is 50, BillTotal: real, Tax: real, Discount: real and CusId, which is the PRIMARY KEY of Customer table, is the FOREIGN KEY in this table.**

**/\*\*\*\*\*\* OBJECT: TABLE [DBO].[ORDER] Script Date: 5/7/2019 1:16:59 PM \*\*\*\*\*\*/**

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Order](

[OrderID] [int] NOT NULL,

[OrderDesciption] [nvarchar](50) NULL,

[OrderPrice] [real] NULL,

[OrderDuration] [int] NULL,

[CusID] [nvarchar](50) NULL,

[BillID] [int] NULL,

CONSTRAINT [PK\_Order] PRIMARY KEY CLUSTERED

(

[OrderID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Order] WITH CHECK ADD CONSTRAINT [FK\_Order\_Bill] FOREIGN KEY([BillID])

REFERENCES [dbo].[Bill] ([BillID])

GO

ALTER TABLE [dbo].[Order] CHECK CONSTRAINT [FK\_Order\_Bill]

GO

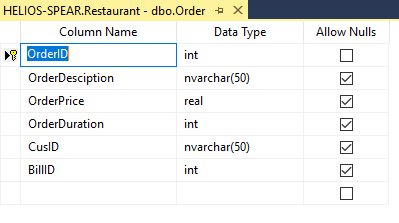
ALTER TABLE [dbo].[Order] WITH CHECK ADD CONSTRAINT [FK\_Order\_Customer] FOREIGN KEY([CusID])

REFERENCES [dbo].[Customer] ([CusID])

GO

ALTER TABLE [dbo].[Order] CHECK CONSTRAINT [FK\_Order\_Customer]

GO



**The table Orders has four simple attributes: orderId(PRIMARY KEY): int, OrderDescription: nvarchar with the max length is 50, OrderPrice: real, OrderDuration: int, and two foreign key which are CusId(reference from Customer.CusId) and BillId(reference from Bill.BillId).**

**/\*\*\*\*\*\* OBJECT: TABLE [DBO].[ORDERCOURSE] Script Date: 5/7/2019 1:16:59 PM \*\*\*\*\*\***/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[OrderCourse](

[CourseID] [nvarchar](50) NULL,

[OrderID] [int] NULL

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[OrderCourse] WITH CHECK ADD CONSTRAINT [FK\_OrderCourse\_Course] FOREIGN KEY([CourseID])

REFERENCES [dbo].[Course] ([CourseID])

GO

ALTER TABLE [dbo].[OrderCourse] CHECK CONSTRAINT [FK\_OrderCourse\_Course]

GO

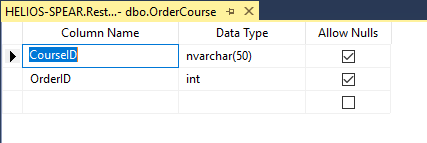
ALTER TABLE [dbo].[OrderCourse] WITH CHECK ADD CONSTRAINT [FK\_OrderCourse\_Order] FOREIGN KEY([OrderID])

REFERENCES [dbo].[Order] ([OrderID])

GO

ALTER TABLE [dbo].[OrderCourse] CHECK CONSTRAINT [FK\_OrderCourse\_Order]

GO



**The table OrderCourse consists of two foreign keys CourseId(reference from Course. CourseId) and OrderId(reference from Orders.OrderId), and those two foreign keys makes up the primary key of this table.**

**/\*\*\*\*\*\* OBJECT: TABLE [DBO].[ORDERCOOK] Script Date: 5/7/2019 1:16:59 PM \*\*\*\*\*\*/**

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[OrderCook](

[CookID] [nvarchar](50) NOT NULL,

[OrderID] [int] NOT NULL

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[OrderCook] WITH CHECK ADD CONSTRAINT [FK\_OrderCook\_Cook1] FOREIGN KEY([CookID])

REFERENCES [dbo].[Cook] ([CookID])

GO

ALTER TABLE [dbo].[OrderCook] CHECK CONSTRAINT [FK\_OrderCook\_Cook1]

GO

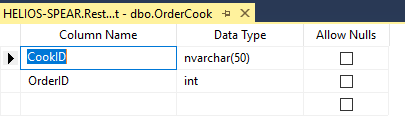
ALTER TABLE [dbo].[OrderCook] WITH CHECK ADD CONSTRAINT [FK\_OrderCook\_Order1] FOREIGN KEY([OrderID])

REFERENCES [dbo].[Order] ([OrderID])

GO

ALTER TABLE [dbo].[OrderCook] CHECK CONSTRAINT [FK\_OrderCook\_Order1]

GO



**The table OrderCookconsists of two foreign keys CookId(reference from Cook. CookId) and OrderId(reference from Orders.OrderId), and those two foreign keys makes up the primary key of this table.**

**VI. PROGRAMMING**

In this program, we defined 2 classes. They are class ResultSetTableModel.java, which is a TableModel that supplies ResultSet data to a JTable, and class DisplayQueryResults.java, which is used to display the contents of the Restaurant database.

1. A screenshot of a cell phone

   Description automatically generated**ResultSetTableModel.java**

* The class ResultSetTableModel extends from AbstractTableModel
* These are the parameters used in the ResultSetTableMode.java in order to create connection object, statement object, result set object, meta data object, number of rows object for this class.
* **A screenshot of a cell phone

  Description automatically generated**The parameter connectedToDatabase is used to keep track of database connection status
* **A screenshot of a cell phone

  Description automatically generated**The public method ResultSetTableModel which takes driver, url, query as parameters and throws SQLException, ClassNotFoundException is used as a constructor which initializes resultSet and obtains its meta data object, determines number of rows.

A screenshot of a cell phone

Description automatically generated

}//end method

* **A screenshot of a cell phone

  Description automatically generated**The method getColumnCount which takes column as parameter and throws IllegalStateException is used to return number of columns in result set.
* A screenshot of a cell phone

  Description automatically generatedThe method getColumnName which takes column as parameter and throws IllegalStateException is used to get name of a particular column in result set.
* **A screenshot of a cell phone

  Description automatically generated**The method getRowCount which throws IllegalStateException is used to returns the number of rows in result set.

A screenshot of a cell phone

Description automatically generated

* The method getValueAt which throws IllegalStateException is used to obtain value at particular row and column.
* A screenshot of a cell phone

  Description automatically generatedThe method setQuery which throws SQLException and IllegalStateExcption is used to set new database query string

**A screenshot of a cell phone

Description automatically generated**

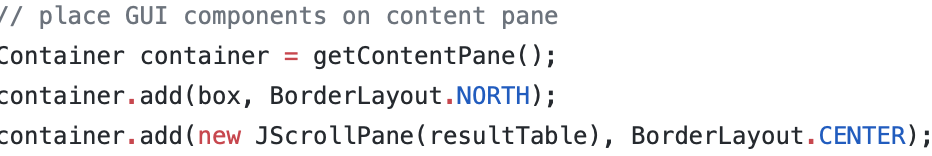
* The method disconnectFromDatabase is used to close statement and connection.

1. **DisplayQueryResults.java**

* **A screenshot of a cell phone

  Description automatically generated**The class DisplayQueryResults.java extends from JFrame
* **A screenshot of a social media post

  Description automatically generated**There are JDBC\_DRIVER object, DATABASE\_URL object, DEFAULT\_QUERY object, table model object, query area object.
* **A screenshot of a social media post

  Description automatically generated**The method displayQueryResult is used to create ResultTableModel and GUI. As this method is executed, if the input query is not a query that displays data from the database, “ResultSet…” will be displayed as notice.

**A screenshot of a social media post

Description automatically generated**

* **A screenshot of a cell phone

  Description automatically generated**The method main is used to execute the program.

**VII. CONCLUSION**

All the duties that have been carried throughout this project of the Restaurant Management System are: getting to know the ideas and demands, drawing ERD, converting relational database schema, creating database and finally make it possible by programming to denotes databases on screen. Every step must be followed strictly to obtain the final satisfactory as well as satisfied result. The primary and important stage that has to be carried carefully is the step of drawing entity relationship diagram, since this step is the illustration of your understanding of the project, how you want your system to work and perform operations; you have to know your system very well to list out all the entities and their relationship to have accurate diagram, which will assist strongly for your process of creating database and reduce as most duplication information as possible. Next primary concern is the relational database schema since the missing of one relation can lead to the inaccuracy of the whole system. We have spent the most time working on those two tasks to bring out the best system to be performed.

The Restaurant Management System is such an efficient tool for storing and retrieving, altering and processing databases that a restaurant should hold for enhancing business as well as boosting quality of customer’s experiences since every operation or performance would be more accelerated, convenient and comfortable; every task would be completed by computer and as less as human impact as possible. This will help the owner to reduce the expense on salary, which will increase the profit gained from the business. Not only the owner, the customers are also offer a bunch of tradeoffs. There will be no more long death time space waiting to be ordered, they will be served the exact food lists they have requested since there no more inaccuracy in paper notes for orders by handwriting. It would be more easily and faster for cooks to take notice of orders and serve the orders as soon as possible. The process of bill payment will not an enormous problem as well thanks to card charging and internet banking assistance; the information of card holder is the only prerequisite for this operation.

To sum up, from this project, what we have learned is that mastering database is a challenging task to be fulfilled in such a short period of time. However, this project has laid a first stone of database and database management system as well as its implementation into real world. Besides, we have gained some foundations on how to perform operation on data as well as manage database system, which will be a great opportunity for us to practice the basic concepts to more sophisticated one which may benefit us in the real cases or in our career path in the future.