HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

ASSIGNMENT 2

REPORT

DATABASE SYSTEMS

RESTAURANT MANAGEMENT SYSTEM

Lecturer: Dr. Prof. Trần Minh Quang

Group 9:

Vũ Lê Thiện Minh 1852590

Vũ Hoàng Anh Khôi 1811013

Đặng Gia Lệ 1812791

Dương Đình Trung 1814498

Contents

c) Role-Based Access Control (RBAC)

3.2) Security in the database design

4. Using tools for tuning and manipulating the DB
   a. Query
   b. Trigger, Stored procedures
   c. Indexing

5. Develop an application

   a. Input forms

6. References

PART 1: ASSIGNMENT 1 REQUIREMENT

1. Detailed requirement description

a) Function requirement

The restaurant system is organized into **BRANCHs**. Each branch has a unique number, a name, an address, and a particular employee who **<manages>** the branch. We keep track of the start date when that employee began managing the branch.

A branch must **<have>** one or more **EMPLOYEEs**. An employee must **work for** one and only one branch. The branch wants to keep track of the employees unique number, name, gender, address, phone, salary, shift. An employee may have more than one shift.

A branch must **<controls>** one or more **DISHes**,and a dish must be contained in one and only one menu.Attribute of dishes includes a unique number, a name, description, price.

There are 3 different types of employees: **waiters, chefs, cashiers**. For waiters, they have rating attributes. About chefs, they have role attributes.

There are employees other than waiters, chefs, cashiers, admins, so an employee does not have to belong to any of these groups. A person can only belong to one of these groups.

A waiter may **<receives>** one or more **ORDERs** or may not receive any orders. An order must have one and only one waiter receive. Attributes of orders include a unique number, a datetime and the total price.

A cashier may **<takes>** one or more **ORDERs** or may not take any orders. An order must be taken by one and only one cashier. Attributes of orders include a unique number, a datetime and the total price.

A chef may **<prepare>** one or more orders or may not prepare any orders. An order may have any numbers of chefs prepare, and must have at least one chef prepare.

An order must **<includes>** one or more dishes which are controlled by a branch. And, a dish may be included in one or more orders or may not be included in any orders. We keep track of the quantity that dishes include in order.

An order must be **<made_by>** one and only one **CUSTOMER**, and a customer must make one or more orders. Each customer has a unique phone number, name, address.

We want to keep track of the **RESERVATIONs** of each customer, a customer may **<make>** many reservations or may not make any reservation, a reservation must be made by one and only one customer. Attribute of reservations include date-time and number of seats.

b) Data requirement

BRANCH

Unique number (BID): string with at most 10 characters. (not null)

Name: string with at most 30 characters. (not null)

Address: string with at most 50 characters. (not null)

EMPLOYEE

Unique number (EID): string with at most 10 characters. (not null)

Name: string with at most 50 characters. (not null)

Gender: 2 value : 0 (female) and 1 (male)

Address: string with at most 50 characters.

Phone: 10 digits

Salary: float number

Date of birth (DOB): YYYY/MM/DD (not null)

Shift: have 3 value : morning, afternoon, evening

Waiter: Rating: float number with 1 digits after decimal point, range from 0 to 5. (not null)

Chef: Role : string with at most 15 characters (not null)

DISH

Unique number: string with at most 10 characters. (not null)

Name: string with at most 50 characters. (not null)

Description: string with at most 100 characters.

Price: float number (not null)

ORDERS

Unique number (OID): : string with at most 10 characters. (not null)

Datetime: YYYY/MM/DD HH:MM:SS

Total price: float number. (default 0)

Quantity: integer number (number of a dish in order) (default 1)

CUSTOMER

Unique phone number: 10 digits (not null)

Name: with at most 50 characters.

Address: with at most 50 characters.

RESERVATION

Date-time : YYYY/MM/DD HH:MM:SS (not null)

Number of seats: integer number (not null)

2. Conceptual DB design

a. Conceptual DB design

Based on the requirement, we design ER model for Restaurant management system.

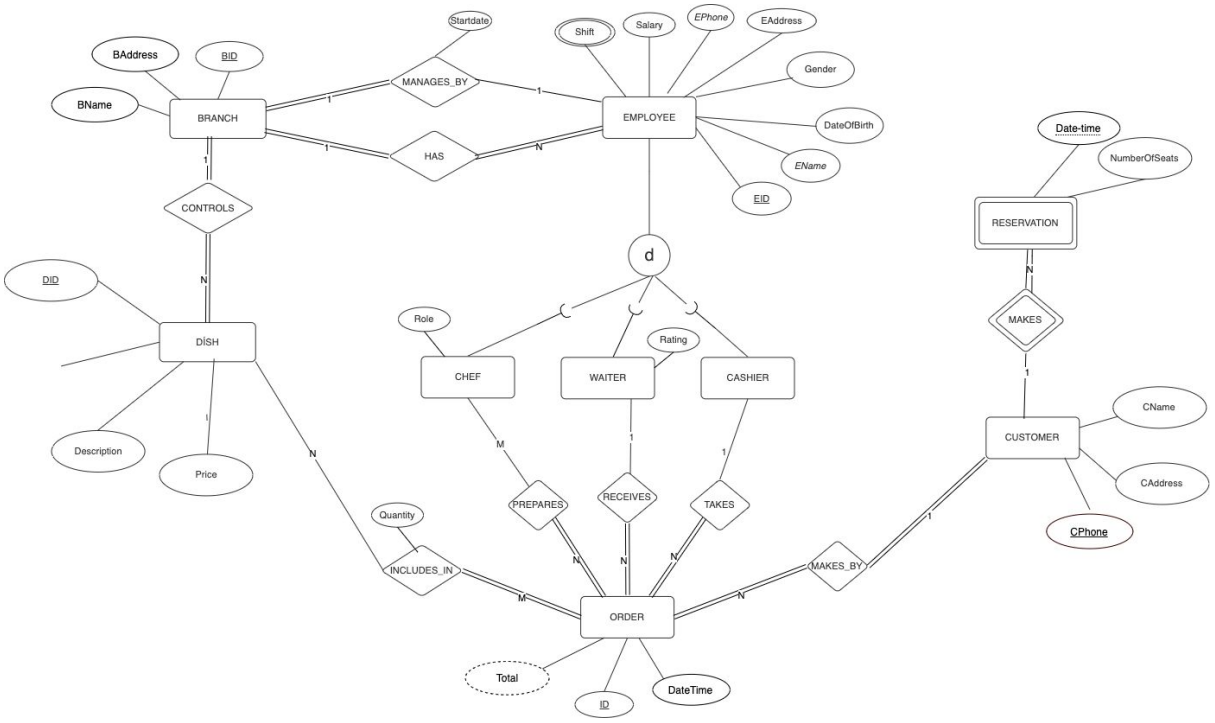


Fig1. ER model for Restaurant Management System

b. Tool for conceptual design

To design ER model we use **diagrams.net** (formerly **draw.io**)

Fig2. diagrams.net (draw.io)

diagrams.net (formerly **draw**.**io**) is free online diagram software. You can use it as a flowchart maker, network diagram software, to create UML online, as an ER diagram tool, to design database schema, to build BPMN online, as a circuit diagram maker, and more. **draw**.**io** can import .vsdx, Gliffy™ and Lucidchart™ files .

3. Logical DB design

We use the ER model to relational mapping method in order to design logical DB.

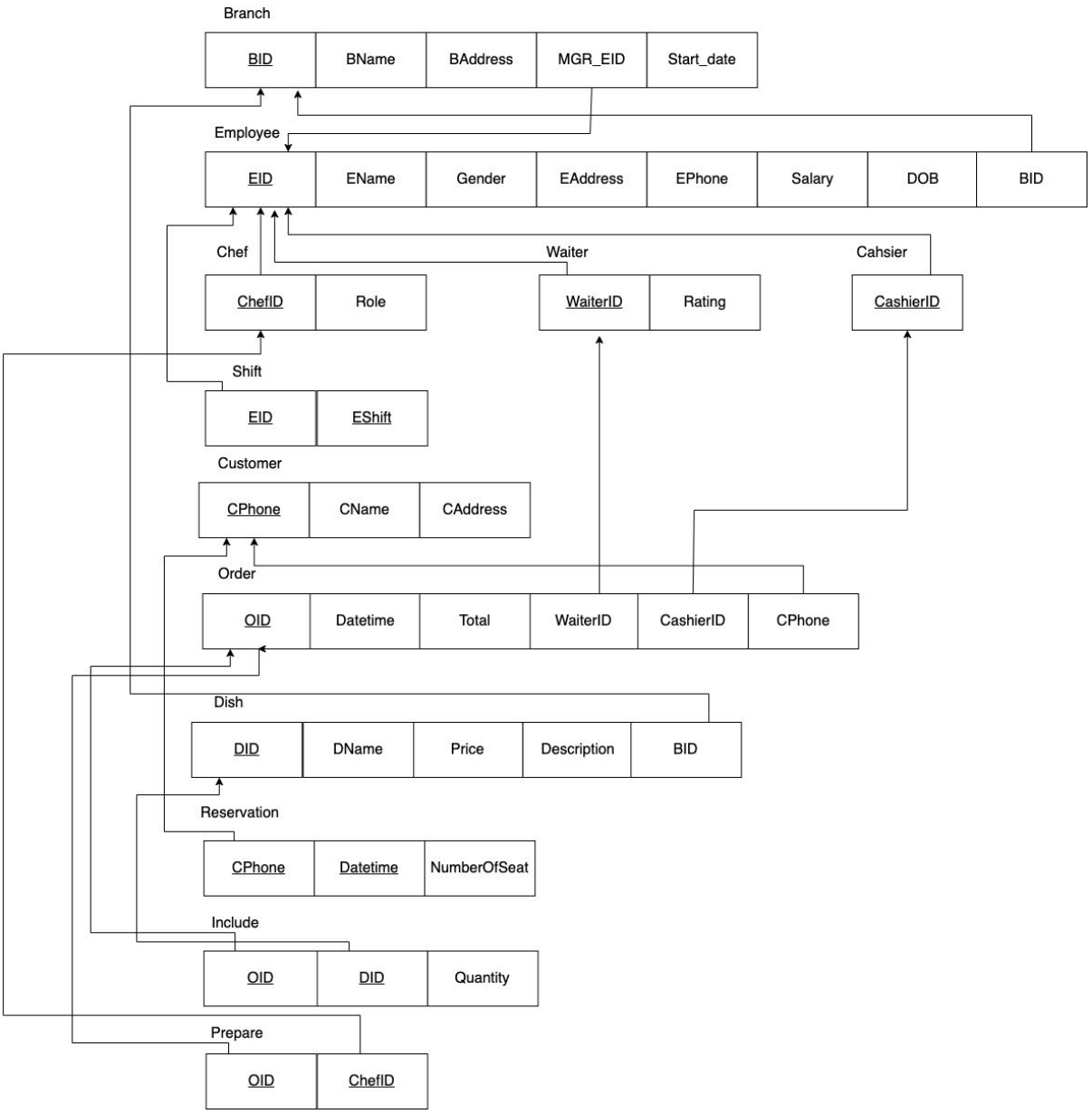


Fig3. Relational mapping for Restaurant Management System

4. Database management system

After discussing, we choose **MySQL** as our DBMS to manage DB.

Fig4. MySQL

MySQL is an open-source relational database management system. A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data. SQL is a language programmers use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups. It can run on many system platforms such as LINUX, macOS, Microsoft Windows, etc.

Advantages of MySQL:

- **Flexibility and easy to use:** MySQL is a fast and stable DBMS and can run on many operating systems.
- **Secure:** MySQL is suitable for applications accessing database and has many features that highly secure the software.
- **High performance:** a wide array of cluster servers backs MySQL. Whether you are storing massive amounts of big e-Commerce data or doing heavy business intelligence activities, MySQL can assist you smoothly with optimum speed.
- **An industry standard:** industries have been using MySQL for years, which means that there are abundant resources for skilled developers. MySQL users can expect rapid development of the software and freelance experts willing to work for a smaller wage if they ever need them.

Disadvantages of MySQL:

- **Limitation:** Based on design, MySQL limits some certain features that a software probably needs.

- **Limited storage capacity:** MySQL does not support a very large database size as efficiently.

5. Physical DB design

Table	Column	Type	Attributes	Index	NULL
Branch	BID	Varchar(10)		Primary Key	No
	Bname	Varchar(30)			No
	BAddress	Varchar(50)			No
	MGR_EID	Varchar(10)			No
	Start_date	Date			
Employee	EID	Varchar(10)		Primary Key	No
	EName	Varchar(50)			No
	Gender	Tinyint(1)			
	EAddress	Varchar(50)			
	EPhone	Varchar(10)			
	Salary	Double			
	DOB	Date			No
	BID	Varchar(10)			No
Chef	ChefID	Varchar(10)		Primary Key	No
	Role	Varchar(15)			No
Waiter	WaiterID	Varchar(10)		Primary Key	No
	Rating	Float			No
Cashier	CashierID	Varchar(10)		Primary Key	No

Customer	CPhone	Varchar(10)		Primary Key	No
	CName	Varchar(50)			No
	CAddress	Varchar(50)			
Orders	OID	Varchar(10)		Primary Key	No
	DateTime	DATETIME			
	Total	Float			
	WaiterID	Varchar(10)			No
	CashierID	Varchar(10)			No
	CPhone	Varchar(10)			No
Dish	DID	Varchar(10)		Primary Key	No
	DName	Varchar(50)			No
	Price	Float			No
	Description	Varchar(100)			
	BID	Varchar(10)			No

Reservation	CPhone	Varchar(10)		Primary Key	No
	Datetime	Datetime		Primary Key	No
	NumberOfSeat	Int			No
Include	OID	Varchar(10)		Primary Key	No
	DID	Varchar(10)		Primary Key	No
	Quantity	Int			No
Prepare	OID	Varchar(10)		Primary Key	No
	ChefID	Varchar(10)		Primary Key	No
	Quantity	Int			No
Shift	EID	Varchar(10)		Primary Key	No
	Eshift	Varchar(10)		Primary Key	No

6. Implement DB into Database management system

Branch table structure:

```
CREATE TABLE IF NOT EXISTS `ass1`.`BRANCH` (
  `BID` VARCHAR(10) NOT NULL,
  `BName` VARCHAR(30) NOT NULL,
  `BAddress` VARCHAR(50) NOT NULL,
  `StartDate` DATE DEFAULT NULL,

  `MGR_EID` VARCHAR(10) NOT NULL,

  PRIMARY KEY (`BID`),

  CONSTRAINT `MGR_BY`
    FOREIGN KEY (`MGR_EID`)
    REFERENCES `ass1`.`EMPLOYEE` (`EID`)
);
```

Table: branch

Columns:

BID	varchar(10) PK
BName	varchar(30)
BAddress	varchar(50)
StartDate	date
MGR_EID	varchar(10)

Employee table structure:

```
CREATE TABLE IF NOT EXISTS `ass1`.`EMPLOYEE` (
  `EID` VARCHAR(10) NOT NULL,
  `EName` VARCHAR(50) NOT NULL,
  `Gender` TINYINT(1) DEFAULT NULL,
  `EAddress` VARCHAR(50) DEFAULT NULL,
  `EPhone` VARCHAR(10) DEFAULT NULL,
  `Salary` DOUBLE DEFAULT NULL,
  `DOB` DATE NOT NULL,
  `BID` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`EID`),

  CONSTRAINT `Work_in`
    FOREIGN KEY (`BID`)
    REFERENCES `ass1`.`BRANCH` (`BID`)
);
```

Table: employee

Columns:

EID	varchar(10) PK
EName	varchar(50)
Gender	tinyint(1)
EAddress	varchar(50)
EPhone	varchar(10)
Salary	double
DOB	date
BID	varchar(10)

Chef table structure:

```
CREATE TABLE IF NOT EXISTS `ass1`.`CHEF` (
  `ChefID` VARCHAR(10) NOT NULL,
  `Role` VARCHAR(15) NOT NULL,

  PRIMARY KEY (`ChefID`),
  CONSTRAINT `FK_Chef`
    FOREIGN KEY (`ChefID`)
    REFERENCES `ass1`.`EMPLOYEE` (`EID`)
    ON DELETE CASCADE
);
```

Table: chef

Columns:

ChefID	varchar(10) PK
Role	varchar(15)

Waiter table structure:

```
CREATE TABLE IF NOT EXISTS `ass1`.`WAITER` (
  `WaiterID` VARCHAR(10) NOT NULL,
  `Rating` FLOAT NOT NULL,
  PRIMARY KEY (`WaiterID`),
  CONSTRAINT `FK_Waiter`
    FOREIGN KEY (`WaiterID`)
    REFERENCES `ass1`.`EMPLOYEE` (`EID`)
    ON DELETE CASCADE
);
```

Table: waiter

Columns:

WaiterID	varchar(10) PK
Rating	float

Cashier table structure:

```
CREATE TABLE IF NOT EXISTS `ass1`.`CASHIER` (
  `CashierID` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`CashierID`),
  CONSTRAINT `FK_Cashier`
    FOREIGN KEY (`CashierID`)
    REFERENCES `ass1`.`EMPLOYEE` (`EID`)
    ON DELETE CASCADE
);
```

Table: cashier

Columns:

CashierID	varchar(10) PK

Customer table structure:

```
CREATE TABLE IF NOT EXISTS `ass1`.`CUSTOMER` (
  `CPhone` VARCHAR(10) NOT NULL,
  `CName` VARCHAR(50) NOT NULL,
  `CAddress` VARCHAR(50) DEFAULT NULL,
  PRIMARY KEY (`CPhone`)
);
```

Table: customer

Columns:

CPhone	varchar(10) PK
CName	varchar(50)
CAddress	varchar(50)

Orders table structure:

```
CREATE TABLE IF NOT EXISTS `ass1`.`ORDERS` (
  `OID` VARCHAR(10) NOT NULL,
  `DateTime` DATETIME DEFAULT NULL,
  `Total` FLOAT DEFAULT 0,
  `WaiterID` VARCHAR(10) NOT NULL,
  `CashierID`  VARCHAR(10) NOT NULL,
  `CPhone` VARCHAR(10) NOT NULL,

  PRIMARY KEY (`OID`),

  CONSTRAINT `Waiter_Receive`
    FOREIGN KEY (`WaiterID`)
    REFERENCES `ass1`.`WAITER` (`WaiterID`)
    ,
     CONSTRAINT `Take_By _Cashier`
    FOREIGN KEY (`CashierID`)
    REFERENCES `ass1`.`CASHIER` (`CashierID`)
    ,
  CONSTRAINT `Cus_Make`
    FOREIGN KEY (`CPhone`)
    REFERENCES `ass1`.`CUSTOMER` (`CPhone`)
    ON DELETE CASCADE
);
```

Table: orders

Columns:

OID	varchar(10) PK
DateTime	datetime
Total	float
WaiterID	varchar(10)
CashierID	varchar(10)
CPhone	varchar(10)

Dish table structure:

```
CREATE TABLE IF NOT EXISTS `ass1`.`DISH` (
  `DID` VARCHAR(10) NOT NULL,
  `DName` VARCHAR(50) NOT NULL,
  `Price` FLOAT NOT NULL,
  `Description` VARCHAR(100) DEFAULT NULL,
  `BID` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`DID`),

  CONSTRAINT `Contain_in`
    FOREIGN KEY (`BID`)
    REFERENCES `ass1`.`BRANCH` (`BID`)
    ON DELETE CASCADE
);
```

Table: dish

Columns:

DID	varchar(10) PK
DName	varchar(50)
Price	float
Description	varchar(100)
BID	varchar(10)

Reservation table structure:

```
CREATE TABLE IF NOT EXISTS `ass1`.`RESERVATION` (
  `CPhone` VARCHAR(10) NOT NULL,
  `DateTime` DATETIME NOT NULL,
  `NumOfSeat` INT NOT NULL,
  PRIMARY KEY (`CPhone`, `DateTime`),
  CONSTRAINT `Make_Reservation`
    FOREIGN KEY (`CPhone`)
    REFERENCES `ass1`.`CUSTOMER` (`CPhone`)
    ON DELETE CASCADE
);
```

Table: reservation

Columns:

CPhone	varchar(10) PK
DateTime	datetime PK
NumOfSeat	int

Include table structure:

```
CREATE TABLE IF NOT EXISTS `ass1`.`INCLUDE` (
  `OID` VARCHAR(10) NOT NULL,
  `DID` VARCHAR(10) NOT NULL,
  `Quantity` INT NOT NULL,
  PRIMARY KEY (`OID`, `DID`),

  CONSTRAINT `Dish_Include`
    FOREIGN KEY (`DID`)
    REFERENCES `ass1`.`DISH` (`DID`)
    ON DELETE CASCADE
   ,
  CONSTRAINT `Order_Include`
    FOREIGN KEY (`OID`)
    REFERENCES `ass1`.`ORDERS` (`OID`)
    ON DELETE CASCADE
);
```

Table: include

Columns:

OID	varchar(10) PK
DID	varchar(10) PK
Quantity	int

Prepare table structure:

```
CREATE TABLE IF NOT EXISTS `ass1`.`PREPARE` (
  `OID` VARCHAR(10) NOT NULL,
  `ChefID` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`OID`, `ChefID`),

  CONSTRAINT `Order_Prepare`
    FOREIGN KEY (`OID`)
    REFERENCES `ass1`.`ORDERS` (`OID`)
    ON DELETE CASCADE
  ,
  CONSTRAINT `Chef_Prepare`
    FOREIGN KEY (`ChefID`)
    REFERENCES `ass1`.`CHEF` (`ChefID`)
    ON DELETE CASCADE
);
```

Table: prepare

Columns:

OID	varchar(10) PK
ChefID	varchar(10) PK

Shift table structure:

```
CREATE TABLE IF NOT EXISTS `ass1`.`SHIFT` (
  `EID` VARCHAR(10) NOT NULL,
  `EShift` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`EID`, `EShift`),
  CONSTRAINT `Employee_Shift`
    FOREIGN KEY (`EID`)
    REFERENCES `ass1`.`EMPLOYEE` (`EID`)
    ON DELETE CASCADE
);
```

Table: shift

Columns:

EID	varchar(10) PK
EShift	varchar(10) PK

PART 2: ASSIGNMENT 2 REQUIREMENT

1. Complete implementation of DB to DBMS

Branch table:

BID	BName	BAddress	StartDate	MGR_EID
1	KFC	Ly Thuong Kiet, HCM	2019-06-01	1812121
2	The Coffee House	To Hien Thanh, Ha Noi	2019-12-22	1812791

Employee table:

EID	EName	Gender	EAddress	EPhone	Salary	DOB	BID
1812121	Khang	1	Nguyen Trai HCM	0913223344	1500	1999-12-13	2
1812123	Hai	1	Tran Hung Dao HCM	0938123456	1200	1998-07-08	2
1812791	Le	0	Nguyen Trai Ha Noi	0938919001	1500	1997-06-09	2
1851234	Minh	1	Nga 6 Ha Noi	0903603963	1300	1995-06-30	1
1852345	Khoi	1	Ly Thuong Kiet HCM	0909603963	2200	2000-01-03	1
1854545	Trung	1	3/2 HCM	0913622679	1200	2001-03-16	1

Chef table:

ChefID	Role
1812791	Comis
1851234	Head

2.

Waiter table:

WaiterID	Rating
1812123	5
1852345	4.5

Cashier table:

CashierID
1812121
1854545

Customer table:

CPhone	CName	CAddress
0901111111	Customer A	District 11
0902222222	Customer B	District 10
0903333333	Customer C	District 10
0904444444	Customer D	District 10
0905555555	Customer E	District 3
0906666666	Customer F	District 11
0907777777	Customer G	District 7
0908888888	Customer H	District 3

Orders table:

OID	DateTime	Total	WaiterID	CashierID	CPhone
1230001	2020-12-10 13:17:17	0	1852345	1854545	0901111111
1230002	2020-12-10 13:17:17	0	1852345	1854545	0901111111
1230003	2020-12-10 14:30:17	0	1812123	1812121	0902222222
1230004	2020-12-10 14:40:17	0	1812123	1812121	0902222222
1230005	2020-12-10 13:45:17	0	1852345	1854545	0903333333
1230006	2020-12-10 15:05:17	0	1812123	1812121	0904444444
1230007	2020-12-10 12:55:16	0	1812123	1812121	0905555555
1230008	2020-12-10 12:12:16	0	1812123	1812121	0905555555
1230009	2020-12-10 12:20:16	0	1852345	1854545	0906666666
1230010	2020-12-10 15:15:17	0	1852345	1854545	0907777777
1230011	2020-12-10 18:55:16	0	1852345	1854545	0908888888

Dish table:

DID	DName	Price	Description	BID
1	Fried Chicken	35000	NULL	1
2	Fried Potato	15000	NULL	1
3	Hambergur	10000	NULL	1
4	CocaCola	5000	NULL	1
5	Coffee	20000	NULL	2
6	Milk Tea	25000	NULL	2
7	Cake	40000	NULL	2
8	Juice	30000	NULL	2

Reservation table:

CPhone	DateTime	NumOfSeat
0901111111	2020-12-22 15:50:00	3
0904444444	2020-12-23 18:30:00	2
0906666666	2020-12-22 19:00:00	4

Include table:

OID	DID	Quantity
1230001	1	1
1230002	3	2
1230003	5	2
1230004	6	1
1230005	1	3
1230006	6	1
1230007	5	3
1230008	7	1
1230009	1	1
1230010	3	1
1230011	2	2

Prepare table:

OID	ChefID
1230003	1812791
1230004	1812791
1230006	1812791
1230007	1812791
1230008	1812791
1230001	1851234
1230002	1851234
1230005	1851234
1230009	1851234
1230010	1851234
1230011	1851234

Shift table:

EID	EShift
1812121	Afternoon
1812123	Afternoon
1812123	Evening
1812791	Evening
1851234	Afternoon
1851234	Morning
1852345	Afternoon
1852345	Evening
1852345	Morning
1854545	Morning

3. Discuss about the normal form (NF) and solutions for improving the NF

Normalization is the process of minimizing **redundancy** from a relation or set of relations. Redundancy in relation may cause insertion, deletion and updation anomalies. So, it helps to minimize the redundancy in relations. Normal forms are used to eliminate or reduce redundancy in database tables.

Here are the most commonly used normal forms:

- First normal form(1NF)
- Second normal form(2NF)
- Third normal form(3NF)
- Boyce & Codd normal form (BCNF)

2.1) First normal form (1NF)

a) Definition and solution for 1NF

If a relation contains composite or multi-valued attributes, it violates first normal form. A relation is in first normal form if it does not contain any composite or multi-valued attribute. A relation is in first normal form if every attribute in that relation is singled valued attribute (atomic values)

Example:

Problem:

DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

Solution:

DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATION
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

b) Discuss 1NF in the design

In our database design, we have a multivalued attribute which is Shift of employees. An employee may have more than one shift (morning, afternoon, evening).

In our design, we have created a new relation shift for employee's shift attribute.

Employee

EID	EName	Gender	EAddress	EPhone	Salary	DOB

Shift

EID	EShift

Result: an attribute (column) of a table don't hold multiple values. It holds only atomic values for each shift of employees. As a result, our database design have already been in 1NF.

EID	EShift
1854545	Morning
1852345	Afternoon
1852345	Evening
1852345	Morning
1851234	Afternoon
1851234	Morning
1812791	Evening

2.2) Second normal form (2NF)

a) Definition and solution for 2NF

A relation is said to be in 2NF if both the following conditions hold:

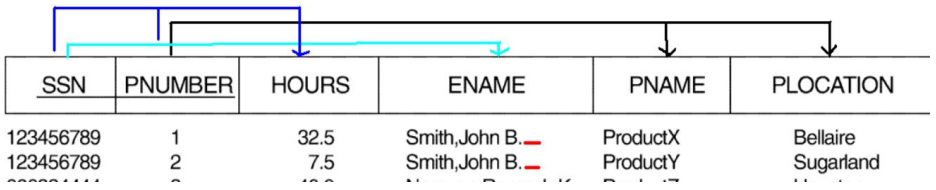
- Relation is in 1NF (First normal form)
- Relation must not contain any partial dependency

Partial dependency : No non-prime attribute (attributes which are not part of any candidate key) is dependent on the proper subset of any candidate key of table

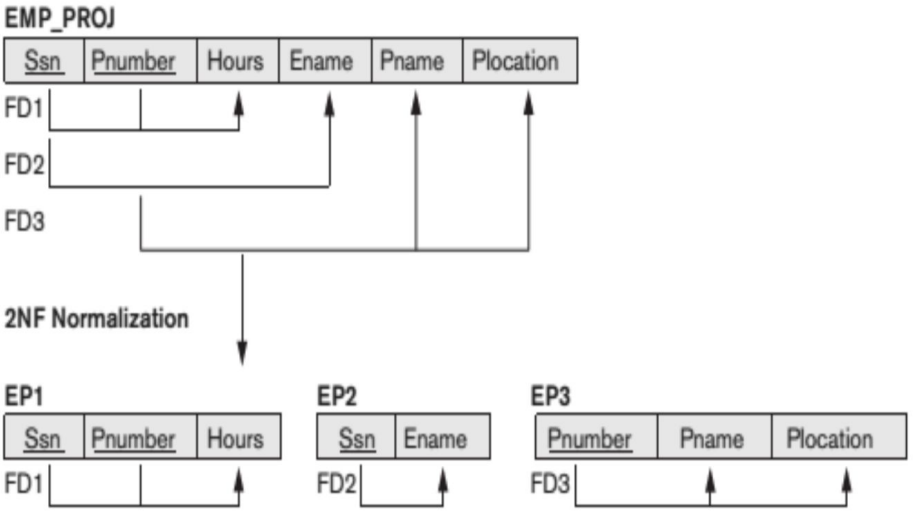
Method: each subset of any candidate determine non-prime attribute will form a separate relation

Example:

Problem:

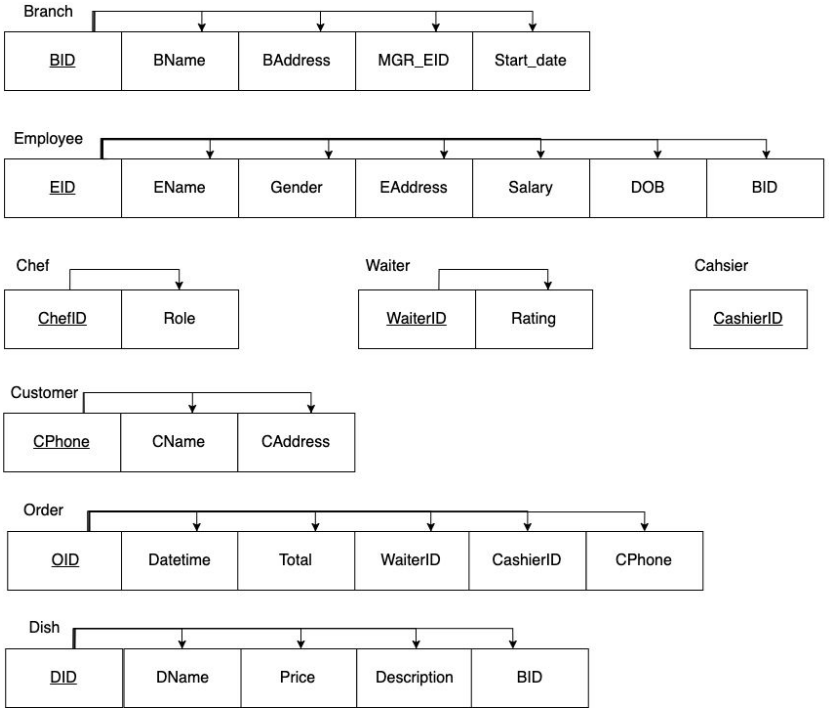


Solution

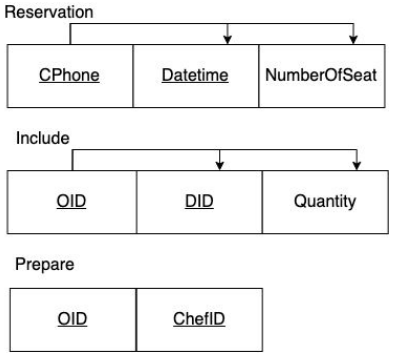


b) Discuss 2NF in the design

In our database design, these relation have only singleton candidate keys(i.e. every candidate key consists of only 1 attribute), so these relations are always in 2NF(because no Partial functional dependency is possible).



We also create new relation prepare, include, reservation. And as you can see, in these relation, there is no non-prime attribute (attributes which are not part of any candidate key) is dependent on the proper subset of any candidate key of table.



For example, you can see in relation reservation, the non-prime attribute number of seats is not dependent on only CPhone attribute or Datetime attribute, it must depend on both CPhone and Datetime. Similarly, with Include relation.

Conclusion, our database design is in 2NF.

2.3) Third normal form (3NF)

a) Definition and solution for 3NF

A relation design is said to be in 3NF if both the following conditions hold:

- Relation must be in 2NF (second normal form)
- There is **no transitive dependency** for non-prime attributes .

Transitive dependency – If A->B and B->C are two FDs then A->C is called transitive dependency.

A relation is in 3NF if at least one of the following condition holds in every non-trivial functional dependency X –> Y

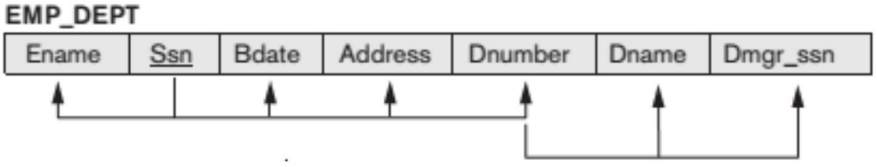
- X is a super key.
- Y is a prime attribute (each element of Y is part of some candidate key).

An attribute that is a part of one of the candidate keys is known as prime attribute.

Method: identify all transitive dependencies and the transitive dependency will form a new relation, and non-prime attributes participating in the transitive dependency with the other attribute which is not in transitive dependency will form another relation.

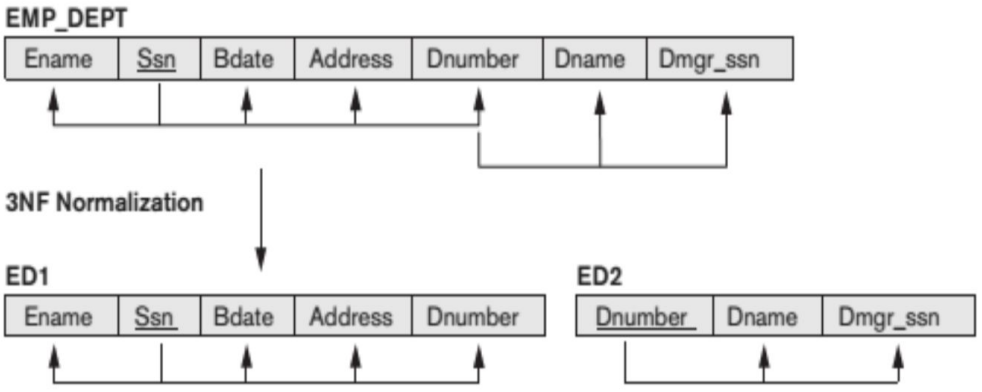
Example:

Problem:



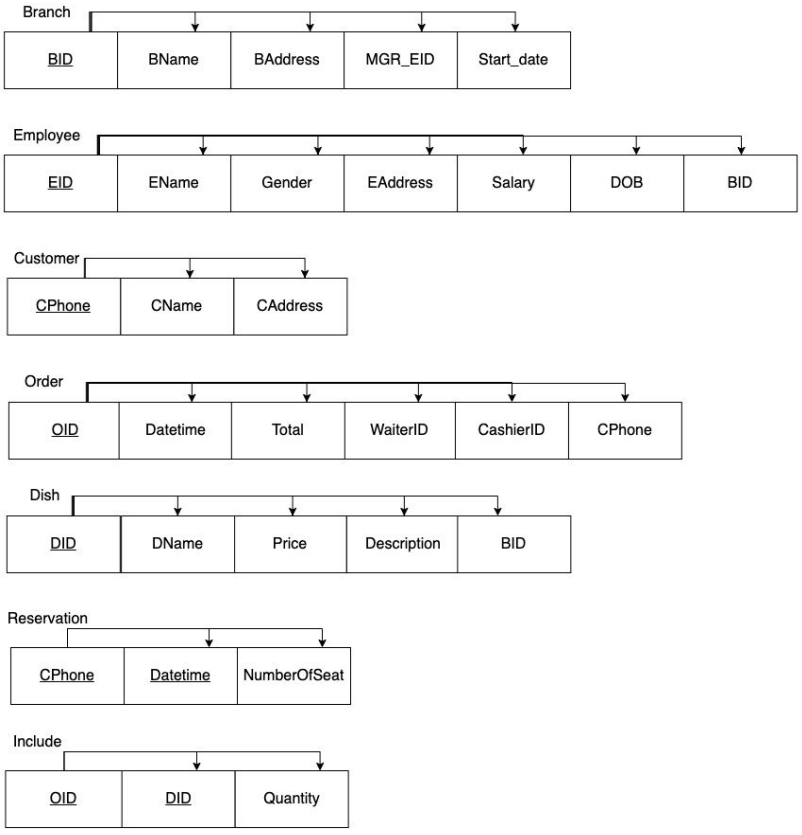
Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

Solution:



b) Discuss 3NF in the design

Similarly, as you can see, our database design, there is no transitive dependency for non-prime attributes .As a results, our database design is in 3NF.



2.4) Boyce-Codd Normal Form (BCNF)

BCNF deals with the anomalies that 3 NF fails to address. For a table to be in BCNF, it should satisfy two conditions:

- The table should be in its 3 NF form
- For any dependency, A à B, A should be a super key i.e. A cannot be a non-prime attribute when B is a prime attributes

4. Discuss the DB security and solutions

3.1) Discuss the DB security and solutions

3.3.1)DB security

- **Types of Security**

Database security is a broad area that addresses many issues, including the following:

• Legal and ethical issues

• Policy issues

• System-related issues

• The need to identify multiple security levels

Three Basic Concepts

Authentication: a mechanism that determines whether a user is who he or she claims to be

Authorization: the granting of a right or privilege, which enables a subject to legitimately have access to a system or a system's objects

Access Control: a security mechanism (of a DBMS) for restricting access to a system's objects (the database) as a whole

- **Threats to databases**

• Loss of integrity

• Loss of availability

• Loss of confidentiality

- **Four main control measures are used to provide security of data in databases:**

  ❖ **Access control**

The security mechanism of a DBMS must include provisions for restricting access to the database system as a whole. This function, called **access control**, is handled by creating user accounts and passwords to control the login process by the DBMS

❖ **Inference control**

The security problem associated with databases is that of controlling the access to a **statistical database**, which is used to provide statistical information or summaries of values based on various criteria.The countermeasures to **statistical database** security problem is called **inference control measures.**

❖ **Flow control**

Flow control prevents information from flowing in such a way that it reaches unauthorized users.

❖ **Encryption**

Data encryption is used to protect sensitive data (such as credit card numbers) that is being transmitted via some type of communication network.

A DBMS typically includes a database security and authorization subsystem that is responsible for ensuring the security portions of a database against unauthorized access.

Two types of database security mechanisms:

- Discretionary security mechanisms
- Mandatory security mechanisms

• Database Security and the DBA

The database administrator (DBA) is the central authority for managing a database system. The DBA's responsibilities include

- Granting privileges to users who need to use the system
- Classifying users and data in accordance with the policy of the organization

The DBA is responsible for the overall security of the database system.

The DBA has a DBA account in the DBMS.Sometimes these are called a system or superuser account. These accounts provide powerful capabilities such as:

- **Account creation:** This action creates a new account and password for a user or a group of users to enable access to the DBMS.

- **Privilege granting:** This action permits the DBA to grant certain privileges to certain accounts.

- **Privilege revocation:** This action permits the DBA to revoke (cancel) certain privileges that were previously given to certain accounts.

- **Security level assignment:** This action consists of assigning user accounts to the appropriate security clearance level.

Action 1 is access control, whereas 2 and 3 are discretionary and 4 is used to control mandatory authorization

- **Access Protection, User Accounts, and Database Audits**

Whenever a person or group of people need to access a database system, the individual or group must first apply for a user account. user must log in to the DBMS by entering account id and password whenever database access is needed.

The database system must also keep track of all operations on the database that are applied by a certain user throughout each login session.If any tampering with the database is suspected, a database audit is performed.A database log that is used mainly for security purposes is sometimes called an audit trail

3.3.2)Solutions

a) Discretionary Access Control (DAC)

The typical method of enforcing discretionary access control in a database system is based on the **granting** and **revoking** of privileges

Types of Discretionary Privileges

- **Account level:** The DBA specifies the particular privileges that each account holds independently of the relations in the database
- **Relation/Table level:** The DBA can control the privilege to access each individual relation or view in the database

Whenever the owner A of a relation R grants a privilege on R to another account B, the privilege can be given to B with or without the GRANT OPTION. If the GRANT OPTION is given, this means that B can also grant that privilege on R to other accounts. If A revokes the privilege granted to B, all the privileges that B propagated based on that privilege should automatically be revoked. There can also be limits on propagation of privileges in horizontal propagation and vertical propagation.

b) Mandatory Access Control(MAC)

Granting access to the data on the basis of users' clearance level and the sensitivity level of the data

Bell-LaPadula's two principles: no read-up & no write-down secrecy

Comparing DAC and MAC:

Discretionary Access Control (DAC) policies are characterized by a high degree of flexibility, which makes them suitable for a large variety of application domains. By contrast, mandatory policies ensure a high degree of protection in a way, they prevent any illegal flow of information.

Mandatory policies have the drawback of being too rigid and they are only applicable in limited environments In many practical situations, discretionary policies are preferred because they offer a better trade-off between security and applicability.

3.2) Security in the database

In our database, we use **Discretionary Access Control** as a database security mechanism.

We create a new 'Manager' user and grant all access rights (SELECT, UPDATE, DELETE, INSERT) of all the tables.

.In addition, to increase security, we also implement some views and procedures for the Manager to use inside the application. And grant him access rights to these views and procedures.

```
GRANT SELECT, UPDATE, DELETE, INSERT ON ManagerView TO Manager WITH GRANT OPTION;
```

We also create many stored procedures. Stored procedures have the following benefits.

- Data logic and business rules can be encapsulated so that users can access data and objects only in ways that developers and database administrators intend.
- Parameterized stored procedures that validate all user input can be used to thwart SQL injection attacks. If you use dynamic SQL, be sure to parameterize your commands, and never include parameter values directly into a query string.

4. Using tools for tuning and manipulating DB

a) Query

1. Retrieve the name, birth date and address of the employee(s) whose name is “Khoi”

```
SELECT EName, DOB, EAddress
FROM EMPLOYEE
WHERE EName = 'Khoi'
```

2. Retrieve the EID, names of all cashiers in the branch which are located in HCM

```
SELECT EID, EName
FROM EMPLOYEE e, CASHIER c , BRANCH b
WHERE c.CashierID = e.EID AND e.BID = b.BID AND BAddress LIKE '%HCM%'
```

3. Retrieve the average salary of all male(value 1)employees .

```
SELECT AVG(Salary) AS AVERAGE_MALE_SALARY
FROM EMPLOYEE
WHERE Gender=1
```

4. Retrieve the average salary of all female(value 0) head chefs from branch address Ly Thuong Kiet.

```
SELECT AVG(Salary) AS AVERAGE_SALARY
FROM EMPLOYEE e LEFT JOIN (CHEF ch, BRANCH b)
ON e.EID=ch.ChefID AND e.BID=b.BID
WHERE Gender= 0 AND Role= "Head" AND BAddress LIKE "%Ly Thuong Kiet%";
```

5. Retrieve all the information about the customer who ordered Fried Potato.

```
SELECT o.OID, c.CName,c.CPhone,c.CAddress,d.Dname
FROM CUSTOMER c  LEFT JOIN (ORDERS o,INCLUDE i, DISH d)
ON c.CPhone = o.CPhone AND o.OID=i.OID AND i.DID=d.DID
WHERE DName="Fried Potato"
```

6. Retrieve all waiters who work in the evening.

```
SELECT EName,e.EID,EPhone
FROM EMPLOYEE e , WAITER w ,SHIFT s
WHERE e.EID=w.WaiterID AND e.EID=s.EID AND EShift="Evening"
```

7. Retrieve the names of all customer who do not make any order

```
SELECT CName
FROM CUSTOMER c
WHERE NOT EXISTS (   SELECT *
                     FROM ORDERS o
                     WHERE c.CPhone=o.CPhone
                     )
```

8. How many quantities of the dish name ‘Coffee’ in each order ?

```
SELECT o.OID, Quantity
FROM ORDERS o, INCLUDE i ,DISH d
WHERE  o.OID=i.OID AND i.DID=d.DID AND DName="Coffee"
```

9. For each order, retrieve the order id, the name of the customer making this order, name of waiter receiving this order and the branch which the waiter works in.

```
SELECT OID, CName, EName, BName
FROM ORDERS o, CUSTOMER c,EMPLOYEE e, WAITER w , BRANCH b
WHERE o.CPhone = c.CPhone AND o.WaiterID = w.WaiterID AND w.WaiterID = e.EID AND e.BID = b.BID
```

10. For each customer, retrieve the customer phone and the total number of orders made by that customer.

```
SELECT c.CPhone, COUNT(*)
FROM  ORDERS o, CUSTOMER c
WHERE c.CPhone = o.CPhone
GROUP BY c.CPhone
```

11. For each branch, calculate the total sum of salary of all employees which is 'Head' chef for each branch.

```
SELECT b.BID, BName, SUM(Salary)
FROM  BRANCH b , EMPLOYEE e, CHEF c
WHERE e.BID = b.BID AND e.EID =  c.ChefID AND c.Role = 'Head'
GROUP BY b.BID
```

12. Retrieve the names, addresses, and number of orders made for all customers who have more than 2 orders.

```
SELECT CName, CAddress , COUNT(*)
FROM CUSTOMER c, ORDERS o
WHERE c.CPhone = o.CPhone
GROUP BY c.CPhone
HAVING COUNT(*) > 2
```

13. Calculate the sum of total price of orders made by Customer A.. Show the information about the phone number and name of customer.

```
SELECT c.Cphone,c.CName,o.OID,  SUM(Price*Quantity) AS TOTAL
FROM CUSTOMER c LEFT JOIN (ORDERS o,DISH d,INCLUDE i)
ON i.DID= d.DID AND o.CPhone=c.CPhone AND o.OID=i.OID
WHERE CName="Customer A"
```

b) Trigger, Function, stored procedure

1. Employees in the company must be older than 18 years old. Write a trigger to implement this constraint.

```
delimiter $$
DROP TRIGGER IF EXISTS check_age;
CREATE TRIGGER check_age BEFORE INSERT
ON Employee FOR EACH ROW
BEGIN
DECLARE age DATE;
SET @age :=  NEW.DOB ;
IF legalAge(age) = false THEN -- Note: use the function 'legalAge'
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'ERROR: Employee must be 18+';
END IF;
END; $$
```

Example: --Try to insert an employee less than 18 years old

```
INSERT INTO EMPLOYEE (EID, EName, Gender, EAddress,EPhone,Salary, DOB,  BID)
VALUES ('1851235', 'Khoa', '1', 'Nga 6 Ha Noi', '0903603963', '1300','2006-06-30','1');
```

Result:

2. Gender of Employees must be only 1 (male) or 0 (female).. Write a trigger to implement this constraint.

```
delimiter $$
DROP TRIGGER IF EXISTS check_gender_value ;
CREATE TRIGGER check_gender_value BEFORE INSERT
ON Employee FOR EACH ROW
BEGIN
IF NEW.Gender > 1 OR NEW.Gender < 0 THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'ERROR: 0 or 1 only';
END IF;
END; $$
```

Example: --Try to insert an employee with a wrong Gender value

```
INSERT INTO EMPLOYEE (EID, EName, Gender, EAddress,EPhone,Salary, DOB,  BID)
VALUES ('1851235', 'Khoa', '2', 'Nga 6 Ha Noi', '0903603963', '1300','2000-06-30','1');
```

Result:

3. Write a function that checks for legal age for employee (from 18): This function is used by the Trigger *check_age* above

```
delimiter $$
DROP FUNCTION IF EXISTS legalAge;
CREATE FUNCTION legalAge(starting_value DATE)
RETURNS BOOLEAN
BEGIN
    DECLARE today DATE;
    SET today = CURDATE();
    IF YEAR(today) - YEAR(starting_value) > 18 THEN
     RETURN TRUE;
    ELSEIF YEAR(today) - YEAR(starting_value) = 18 THEN
     IF MONTH(starting_value) > MONTH(today) THEN
         RETURN FALSE;
     ELSEIF DAY(starting_value) <= DAY(today) THEN
         RETURN TRUE;
     ELSE RETURN FALSE;
        END IF;
    ELSE RETURN FALSE;
    END IF;
END;$$
```

4. Write a function that returns the average salary when given an branch's ID.
   Input: branch ID
   Output: average salary

```
delimiter $$
DROP FUNCTION IF EXISTS averageSalary;
CREATE FUNCTION averageSalary (BID varchar(10))
RETURNS DOUBLE
BEGIN
    DECLARE result DOUBLE;
    SET result = (SELECT AVG(`Employee`.`Salary`) FROM Employee WHERE `Employee`.`BID` = BID);
    RETURN result;
END; $$
```

Example: --Return average salary of employees from branchID 1

```
SELECT averageSalary(1);
```

Result:

	average_salary(1)
▶	1566.6666666666667

5. Write a procedure that returns reservation information (Datetime, Number of seats) when given a customer's phone number.

```
delimiter $$
DROP PROCEDURE IF EXISTS reservationInfo;
CREATE PROCEDURE reservationInfo (Phone varchar(10))
BEGIN
    SELECT `Reservation`.`DateTime`, NumOfSeat FROM Reservation WHERE Phone = CPhone;
END; $$
```

Example: --Get reservation information from the customer with the phone number 0904444444

```
call reservationInfo('0904444444');
```

Result:

	DateTime	NumOfSeat
▶	2020-12-23 18:30:00	2

6. Write a procedure that increases salary to 10% for all the waiters whose rating is greater than 4.5

```
delimiter $$
DROP PROCEDURE IF EXISTS increaseSalary;
CREATE PROCEDURE increaseSalary()
BEGIN
    DECLARE n INT;
    DECLARE i INT;
    DECLARE id VARCHAR(10);
    DECLARE r FLOAT;
    SELECT COUNT(*) FROM ass1.waiter INTO n;
    SET i=0;
    WHILE i<n DO
    SELECT WaiterID into id FROM ass1.waiter LIMIT i,1;
        SELECT Rating into r FROM ass1.waiter LIMIT i,1;
     IF r > 4.5 THEN
         UPDATE ass1.employee SET Salary = Salary* 1.1 WHERE employee.EID = id;
        END IF;
     SET i = i + 1;
    END WHILE;
END;$$
```

7. Write a procedure to calculate the total price a customer has to pay for each order.

```
delimiter $$
DROP PROCEDURE IF EXISTS totalCal;
CREATE PROCEDURE totalCal (OrderID varchar(10))
BEGIN
    UPDATE Orders
    SET Total = (SELECT SUM(`Dish`.`Price` * `Include`.`Quantity`)
                                FROM Dish, Include
                                WHERE OID = OrderID AND `Dish`.`DID` = `Include`.`DID`)
    WHERE OID = OrderID;
END; $$
```

c) Indexing

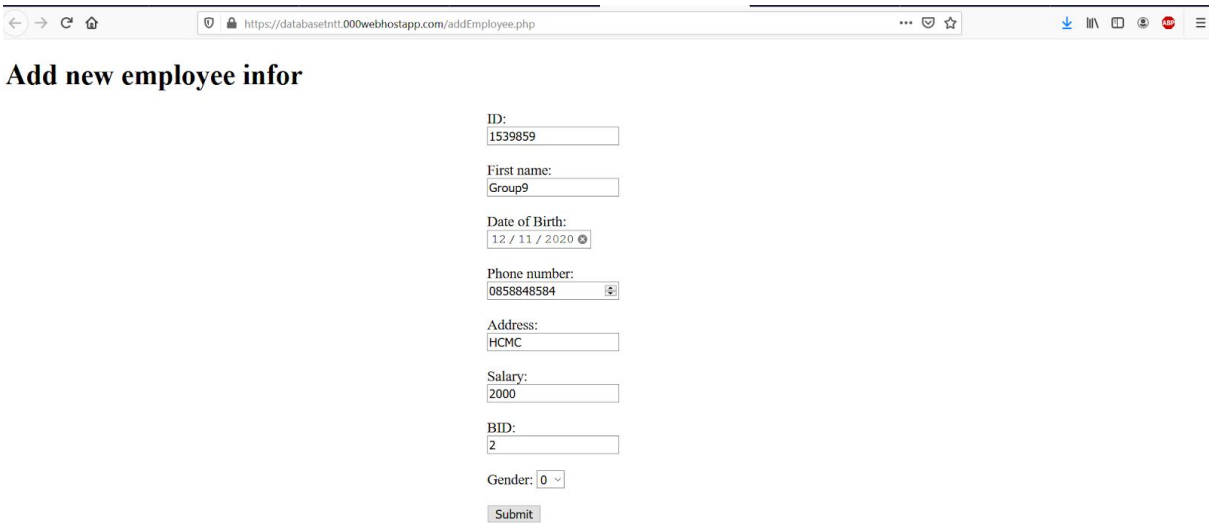
Since the Employee table could be expanded to very big and our queries WHERE clauses rely mostly information from EID and EName, we choose to index these two columns to improve the database performance.

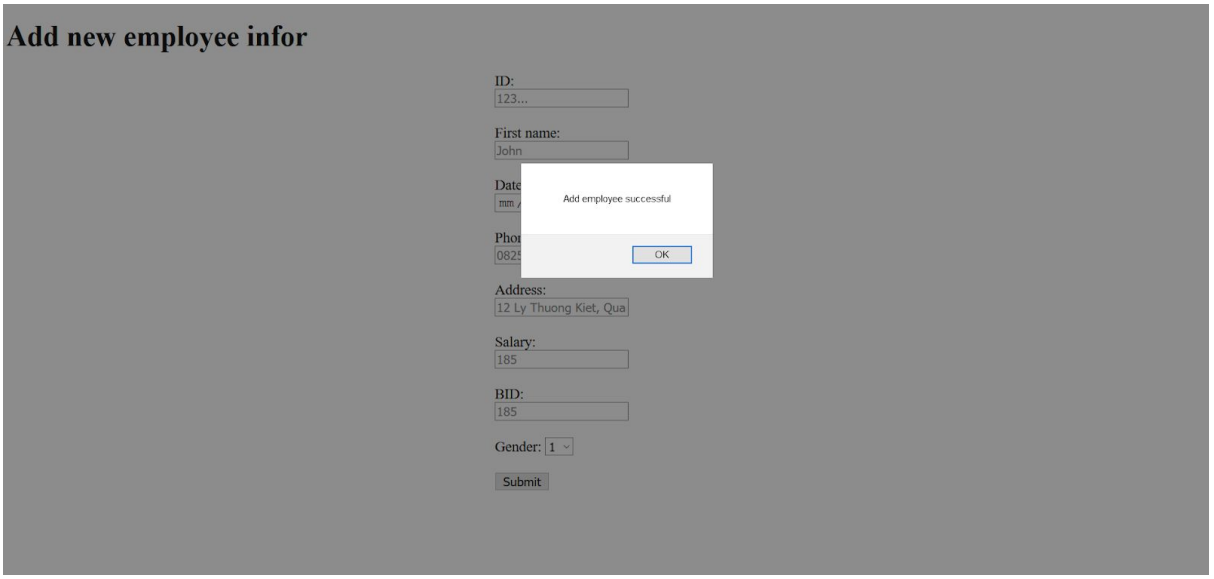
```
CREATE INDEX ID ON Employee (`EID`);
CREATE INDEX Name ON Employee (`EName`);
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible
employee	0	PRIMARY	1	EID	A	5	NULL	NULL		BTREE			YES
employee	1	ID	1	EID	A	5	NULL	NULL		BTREE			YES
employee	1	Name	1	EName	A	5	NULL	NULL		BTREE			YES
employee	1	Work_in	1	BID	A	2	NULL	NULL		BTREE			YES

5. Develop an application

a)Input form





	EID	EName	Gender	EAddress	EPhone	Salary	DOB	BID
Edit Copy Delete	12345	M	1	HCM	88383	12325	2020-12-12	1
Edit Copy Delete	1539859	Group9	0	HCMC	0858848584	2000	2020-12-11	2
Edit Copy Delete	1812121	Khang	1	Nguyen Trai HCM	0913223344	1500	1999-12-13	2
Edit Copy Delete	1812123	Hai	1	Tran Hung Dao HCM	0938123456	1200	1998-07-08	2
Edit Copy Delete	1812791	Le	0	Nguyen Trai Ha Noi	0938919001	1500	1997-06-09	2
Edit Copy Delete	1851234	Minh	1	Nga 6 Ha Noi	0903603963	1300	1995-06-30	1
Edit Copy Delete	1852345	Khoi	1	Ly Thuong Kiet HCM	0909603963	2200	2000-01-03	1
Edit Copy Delete	18525901	M	1	HCM	1231233	12325	2020-12-12	1
Edit Copy Delete	1854545	Trung	1	3/2 HCM	0913622679	1200	2001-03-16	1
Edit Copy Delete	35234653	Khoi	1	sdftgsag	35436457	124123	2020-12-01	1

Employee “Group9” has successfully been added.

6. References

https://www.google.com/

https://www.w3schools.com/

https://www.mysql.com/

https://vi.wikipedia.org/wiki/MySQL