HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE & ENGINEERNG

Database system

Assignment 2 report

Nguyen Minh Tri - 1752568
Nguyen Hoang Long - 1852164

Ho Chi Minh, 2020

Table of Contents

Introduction

Nowadays, with the development of technology and industrial manufacturing, the economy of the world and individual country is growing faster than ever. And with the growth of the economy comes the birth of new companies and businesses. For these companies and businesses to operate smoothly and successfully, having a database management system to manage employees is crucial. Therefore, in assignment 1 and assignment 2, our team will try to design and implement an effective Human Resource Database Management System for companies and businesses in today's world.

Requirement description for the database system

Company A wants to create a database system for Human Resource Management. They will need to handle three main components, which are Employee, Department and Project. In each Human Resource Department, we will have one supervisor who supervises others' work. Each employee can have train courses and off days.

- Employee: Each employee must work for at least one department. They can work on many projects (which are under control of the department) at the same time. They will also work on client projects which belong to the department (but the department does not have control over client projects due to multilateral cooperation). We also have the supervisor of all employee as the Human Resource Manager.

- Department: Each department must have employees working for them while assigning them to project. Departments control projects of the company. Departments also have outside projects (client projects) which cooperate with the client's side. Company A also wants a department manager for every department. Each department can have more than one location, due to the company being international.

- Client project: A client project is under the authorization of a department. A client project can also change depending on the requirements of the clients.

- Client: Each client can have multiple projects with the company. These projects are called client projects, worked on by the company's employees and managed by individual department.

- Employee release time: Employee can have unpredictable or predictable times of break. The company will need to track starting date, the number of days off of the employee and the type of release. The type of release can be either temporary break or resignation.

- Training course: Each employee can have many training courses, and each course can have many employees. Train course varies in types and have a specific number of hours an employee need to spend to complete the course.

- Dependent: To care for its employees, the company will need extra information from their employees. These information are the employee's gender, birthday and relatives. However, it is optional and the employee can choose to keep secret of their information.

Entity description

Each entity has attributes and primary keys displayed as below:

- Employee:
    - Social security number: Ssn(key)
    - Name includes: f_name, m_name, l_name
    - Birth day: birth_day
    - Salary: salary
    - Gender: gender

- Department:
    - Department Identity: dp_id(key)
    - Department name: dp_name
    - Department type: dp_type
    - Department location: dp_location

- Project:
    - Project Id: Work_id(key)
    - Project hour: hours
    - Project type: type

- Client project:
    - Client project id: Work_id(key)
    - Day started: Started_day
    - Day delivered: Delivery_day

- Client:
    - Cliend Id: Client_id(key)
    - Contact number: Contact

  - Date of birth: Birth_day

- Release Time:
  - Release id: Em_ssn(key)
  - Release date: Date(key)
  - Number of break days: Day_count
  - Type of release: Type

- Train Course:
  - Course id: Ssn(key)
  - Training id: Train_id(key)
  - Hours trained: Hours
  - Course type: Course type

- Dependent:
  - Employee's name: Name(partial key)
  - Employee's genderGender
  - Date of birth: Birth_day
  - Relatives: Relationship

Relationship description

The relationships between entities are below:

- N:1 relationship between Release_time and Employee. This relationship is named <Off>.
- M:N relationship between Train_course and Employee. This relationship is named <Has>.
- N:1 relationship between Dependent and Employee. This relationship is named <Depend_of>.
- N:1 relationship between Employee and Department. This relationship is named <Work_for>.
- M:N relationship between Employee and Project. This relationship is named <Work_on>.
- N:1 relationship between Project and Department. This relationship is named <Control>.
- M:N relationship between Employee and Client_project. This relationship is named <Worked>.

- N:1 relationship between Client_project and Department. This relationship is named <Has>.
- N:1relationship between Client_project and Client. This relationship is named <Work_for>.

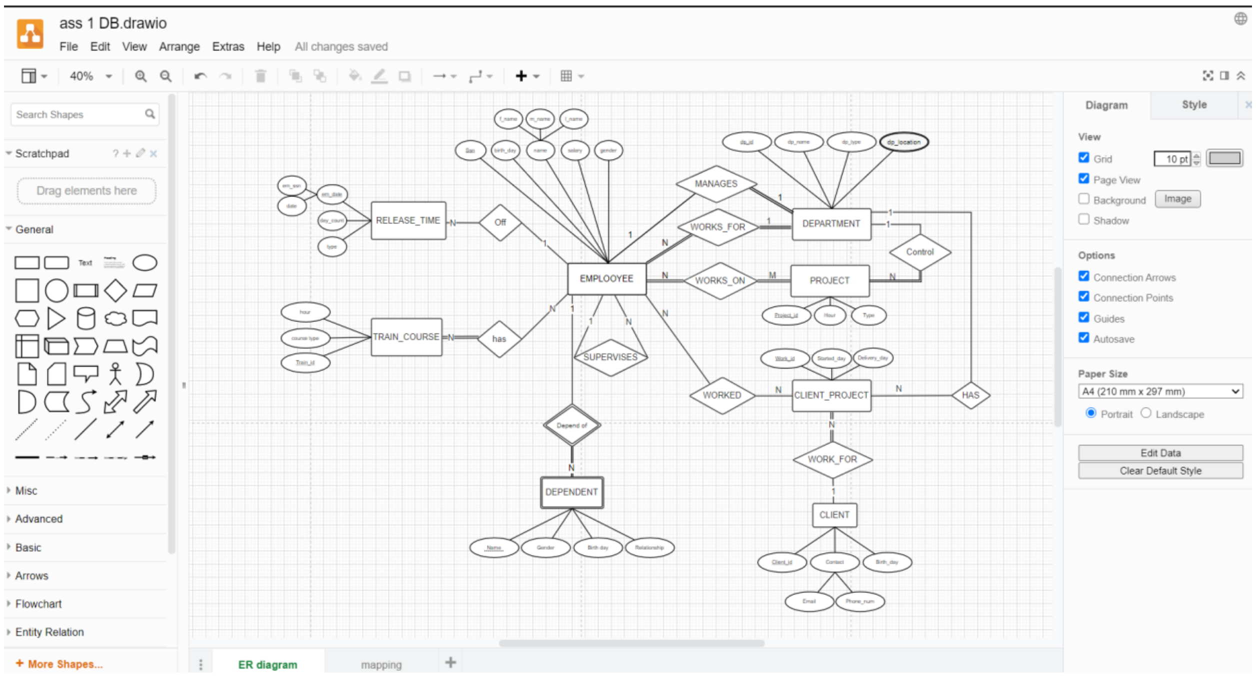
Database Design Tools

To design our database in this Assignment, DrawIO will be our tool of choice.

What is DrawIO?

DrawIO is a tool for drawing UML, ERD, and any designed diagram. We can use either the web base or desktop application.

The main interface of DrawIO:



Pricing

DrawIO is a paid service. Although you can use the free version but it’s limited in some tools and lack of freedom in workplace. For premium options, the online app is 5$ per month or $49 per year for a Personal account, or $25 per month or $249 per year for a Team account (that’s the starting price for up to five members).

Advantages

- DrawIO provides search bar for shape and supports other design tool object like lucid chart.
- We can store our project offline or online. Each online project can be set up to be shared between users of DrawIO.
- DrawIO allows users to load open-source projects. We can download other projects from the internet and load them as a sub factors of our project if suitable.
- Another important feature of the tool is that we can export the design to Jpeg, PNG and even SVG for web application.
- User can make their own element then add to their local library. This is hugely useful for professional users whenever they want to reuse the shapes which belong to their specific work.

Disadvantages

- This is an online tool for multiple user, so you can only use this function when there is an internet connection
- Difficult to access pricing and plan options
- Using lines to link objects is sometimes very finicky.

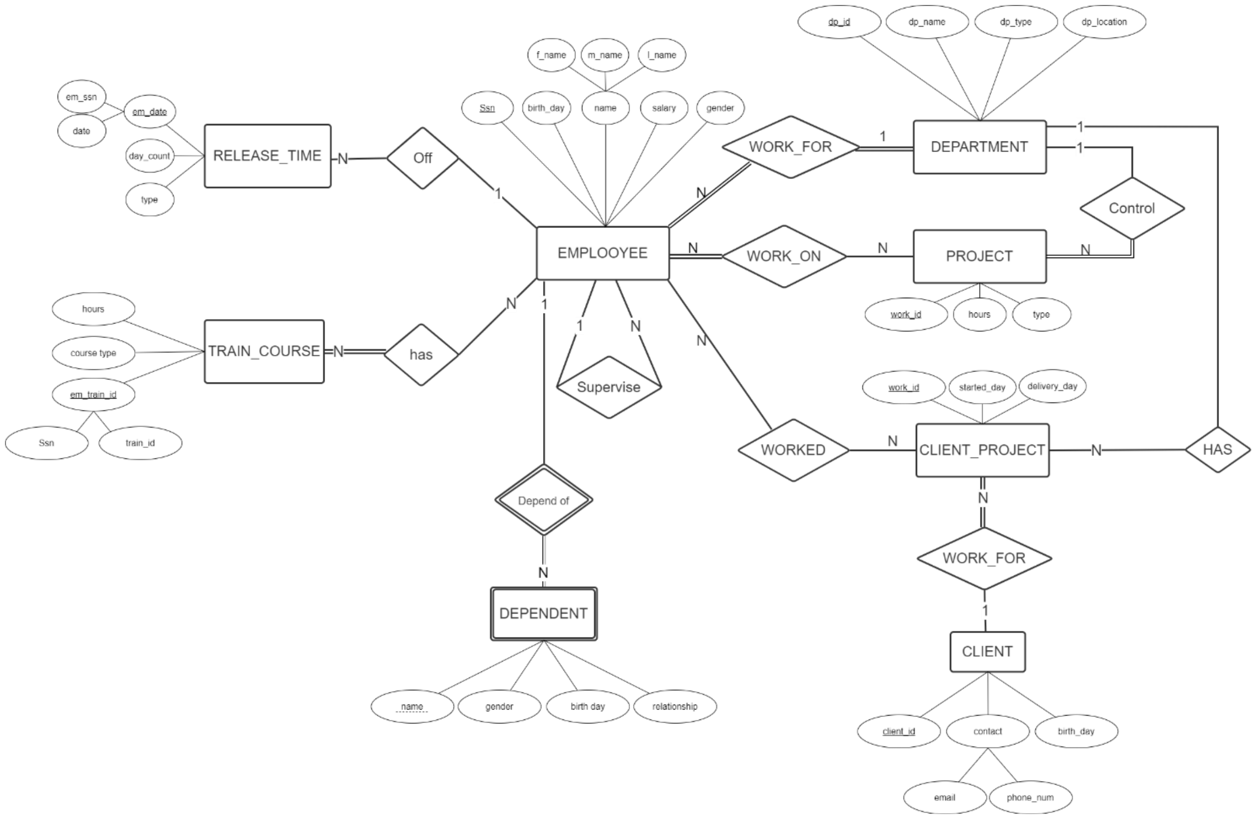
Conceptual database design

Based on the description that we made on part 2: Requirement Description, we divide database system into list of entities and its attributes below:

- EMPLOYEE: **Ssn**, birth_day, t_name, m_name, l_name, salary, gender.
- RELEASE_TIME: **em_ssn**, **date**, day_count, type.
- TRAIN_COURSE: **ssn**, **train_id**, hours, course_type.
- DEPARTMENT: **dp_id**, dp_name, dp_type, dp_location.
- PROJECT: **work_id**, hours, type.
- CLIENT_PROJECT: **work_id**, started_day, delivery_day.
- CLIENT: **client_id**, contact, birth_day.
- DEPENDENT: **name**, gender, birth_day, relationship.

Our conceptual design is described as the ERD Diagram below:

(The entity relation diagram is quite large so I post this on google drive in case that you want to see it more clearly: https://drive.google.com/drive/folders/1-F8FH3YmPAYn3K4o0DROtmaHuMyi-Isl?usp=sharing)



Logical database design

From Conceptual design to Logical design using Relational data model, a several steps have been applied. However, we will list only steps that we use to mapping our design from ERD to relational data model:

Step 1: Mapping of Regular (strong) Entity Type: An entity will become a relation, and its attributes and primary key will be the attributes and primary key of that relation.

Example: EMPLOYEE (Ssn, birth_day, t_name, m_name, l_name, salary, gender)

Step 2: Mapping of Binary 1:1 Relationship Types: For two entities E1 and E2, if E2 is a mandatory member of a 1:1 relationship with E1, then the relation for E2 will include the prime attributes of E1 as a foreign key to represent the relationship.

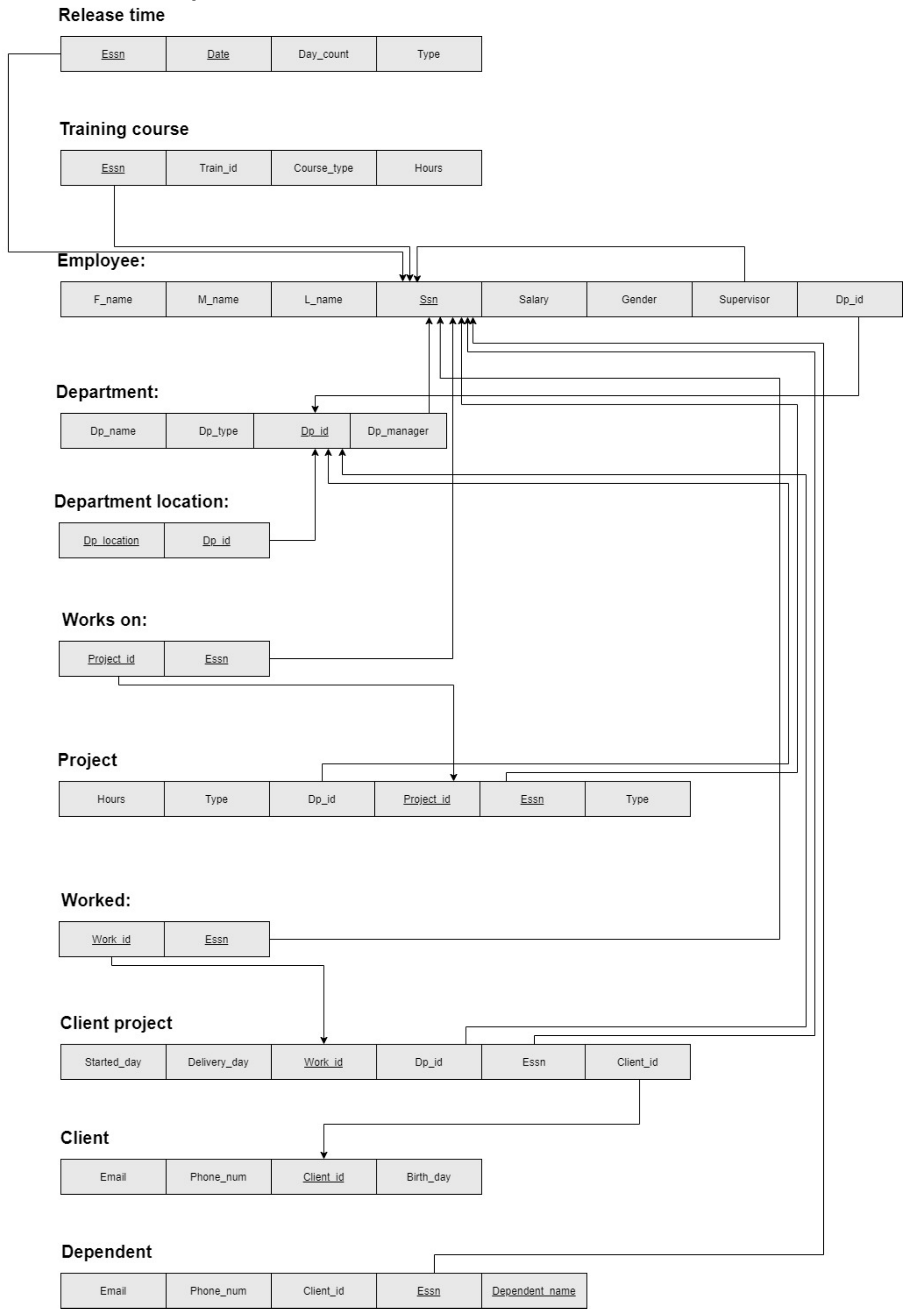
Step 3: Mapping of Binary N:1 Relationship Types: For two entities E1 and E2, if E2 is a mandatory member of an N:1 relationship with E1, then the relation for E2 will include the prime attributes of E1 as a foreign key to represent the relationship. If the membership class of E2, which is at the N-side of the relationship, is optional (i.e. partial), then the above guideline is not applicable.

Example: CLIENT_PROJECT (started_date, delivery_date, work_id, *client_id*) with *client_id* as the foreign key

Step 4: Mapping of Binary M:N Relationship Types: An M:N relationship is always represented by a new relation which consists of the prime attributes of both participating entity types together with any attributes of the relationship. The combination of the prime attributes will form the primary key of the new relation.

Example: WORKED (*Essn*, *Work_id*) with *Essn* the prime attribute of Employee and *Work_id* the prime attribute of Client_project

The logical design below is based on the relational data model, this design is also created by DrawIO:



Database Management System Implementation (Assignment 1)

For implementing the database management system, we will use MySQL database management system and MySQL Workbench application.

MySQL

MySQL is the world's most popular database management system backed by open-source code that is based on Structured Query Language. In more detail, this DBMS is quite useful in most cases of applications but it is mostly used in web applications and online publishing due to easy implementation and worldwide usage.

The applications of MySQL are widely known as LAMP and APACHE. They are the two most popular server-side managers connected with PHP, Perl and Python as well. Moreover, MySQL nowadays is at the top of web based applications including Facebook, Twitter, Instagram and YouTube.

We chose MySQL because of it's reliability, high availability, applicable for many services like a Human Resource Management System (our assignment topic) or inventory management, etc. It is designed to process millions of queries and transactions at optimal speed.

The setup process is relatively quick regardless of platform. MySQL offers many self-management features including automatic space expansion, auto restart, etc. It also has a graphical control of many servers from a single station.

How does MySQL work?

MySQL is actually a server management application using the base management system of command. MySQL server is available as a separate program for use in a client-server networked environment, and it is embeddable into applications. Applications use MySQL can be single or multiple locations.

MySQL supports multiple primitive data types. User can invoke simple commands for various purposes like SELLECT, INSERT, DELETE and UPDATE. It is also important to state that MySQL uses TCP but not UDP. In more detail, TCP is Transmission Control Protocol which is safer than UDP.

My SQL Workbench

Workbench nowadays is popular because it is powerful for implementing code and for mapping from entity relationship and relational mapping. It has inverse and reverse engine which helps generate SQL code and physical design quickly. We can manipulate lots of features fast and easily with built in tools.

Implementation of Database on MySQL

Schema

To create a new schema, we first specify its name, character set and collation to be used. After creating the following items in GUI, press Apply for change to take effect.

For example:
CREATE SCHEMA IF NOT EXISTS `humanresource`

Table

To add tables to our newly created schema, we create a table for every relation in the referencing model in Logical Design part. Each attribute will become a column with its own name, datatype and various pre-defined constraints, for example:

- If we set an attribute to be Not Null, that attribute must hold a value.
- If we set Unique, the domain for that attribute cannot have a repeating value.

We can also add a default expression for an attribute, in case the field is not explicitly set during insertion.

When finishing creating a table, we must set its primary key and other constraints of necessary.

For example:

```
CREATE TABLE IF NOT EXISTS `humanresource`.`CLIENT`
CREATE TABLE IF NOT EXISTS `humanresource`.`EMPLOYEE`
CREATE TABLE IF NOT EXISTS `humanresource`.`DEPARTMENT`
```

Foreign key

To add a foreign key, which is in itself a constraint, we have to first name it. The foreign key's name can be anything as long as it hasn't been stated before. Afterwards we need to let MySQL know where the table holding the primary key is. Then we select an attribute column on our current table to become a foreign key and pick the referenced column in the other table (its primary key). The two columns need to have the same datatype, size, otherwise the GUI cannot proceed.

To change the foreign key when the primary key value is modified, you have these actions:

- Set null: Clear the foreign key value
- Cascade: Follow the primary key
- No action: Do nothing
- Restrict: In MySQL, this is similar to No action

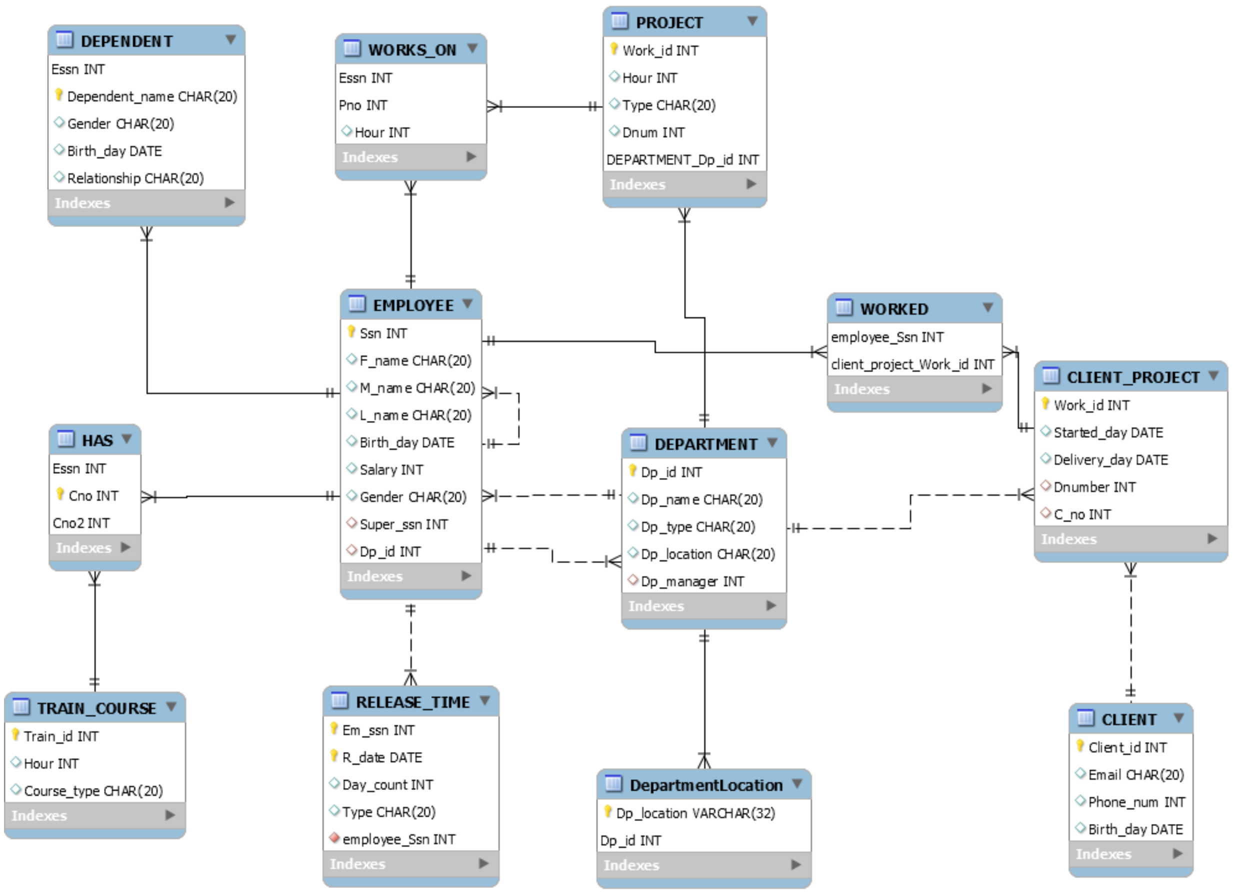
Scripting

To start querying, in MySQL, an SQL script file has the extension .sql, and can be executed on the selected schema. You can also use the command line interface if you don't want the GUI to do everything for you.

Physical design

We design a Physical Design to implement on MySQL DBMS. MySQL has built in tools to create a physical design from the schema we previously created.

The design is shown as the diagram below:



Implementation into DBMS (Assignment 2) – Applying database into phpMyAdmin for lately website building

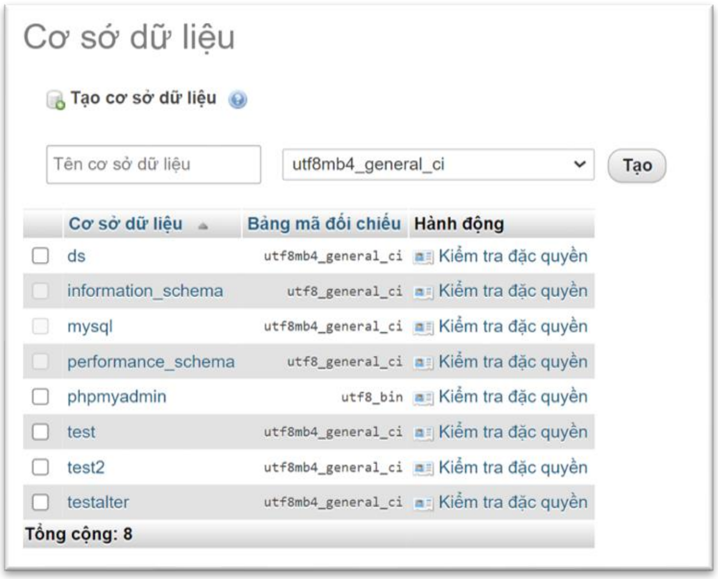
We will first go through how to create Schema, Tables, Foreign keys, etc. similar to what we did in Implementation into DMBS (Assignment 1) previously described. However, this part can be considered an advanced and updated version of Implementation into DMBS (Assignment 1), resulting in a complete implementation of Human Resource Database Management System.

Schema

To begin with the project we create a schema, which is necessary for any SQL database management system.

Within this assignment we choose to use utf8. During the implementation we already created a local server host with Xampp and provided the account for accessing the database. Because our database is based on the conceptual design and the mapping of previous part, after applying, the schema will have the same features as expected.

The below picture is the GUI of phpMyAdmin, after inserting the database name, we hit Make button and the system will create the schema with related tools to help us manipulate the tables inside.



	Cơ sở dữ liệu	Bảng mã đối chiếu	Hành động
☐	ds	utf8mb4_general_ci	Kiểm tra đặc quyền
☐	information_schema	utf8_general_ci	Kiểm tra đặc quyền
☐	mysql	utf8mb4_general_ci	Kiểm tra đặc quyền
☐	performance_schema	utf8_general_ci	Kiểm tra đặc quyền
☐	phpmyadmin	utf8_bin	Kiểm tra đặc quyền
☐	test	utf8mb4_general_ci	Kiểm tra đặc quyền
☐	test2	utf8mb4_general_ci	Kiểm tra đặc quyền
☐	testalter	utf8mb4_general_ci	Kiểm tra đặc quyền

Tổng cộng: 8

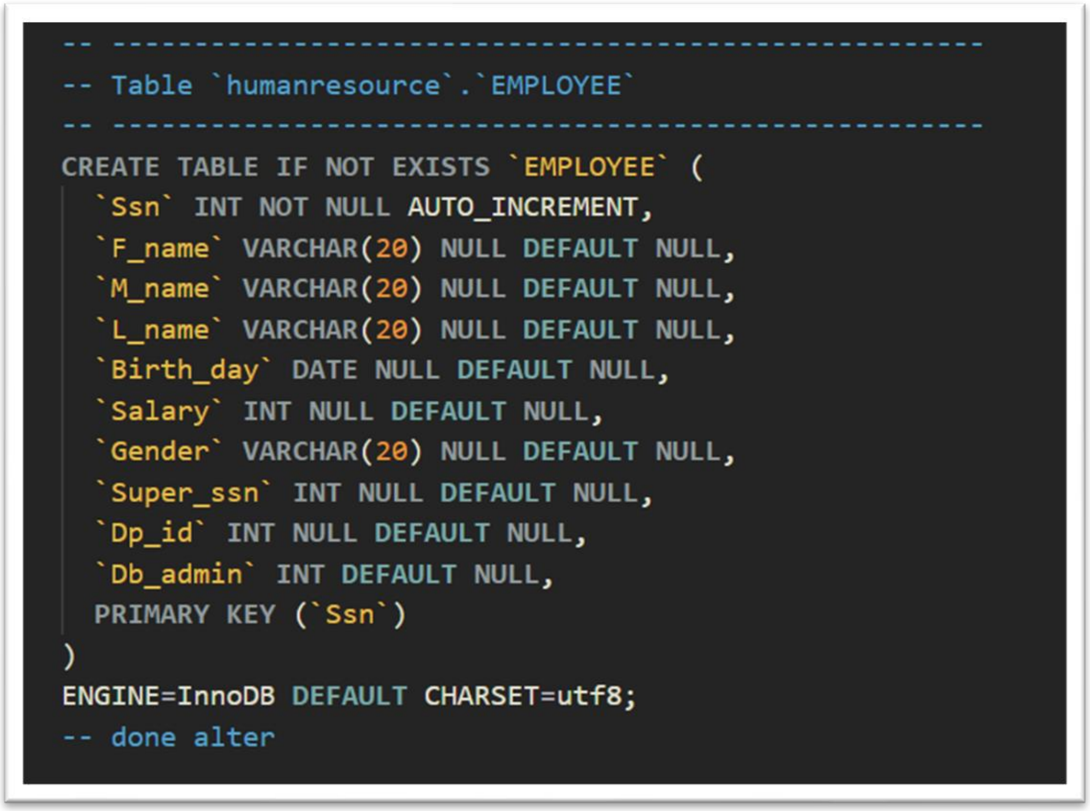
Table

We use InnoDB for storage engine: ENGINE = InnoDB

Each attribute has few properties: name – data type – Nullable – auto increment and so on. For example:

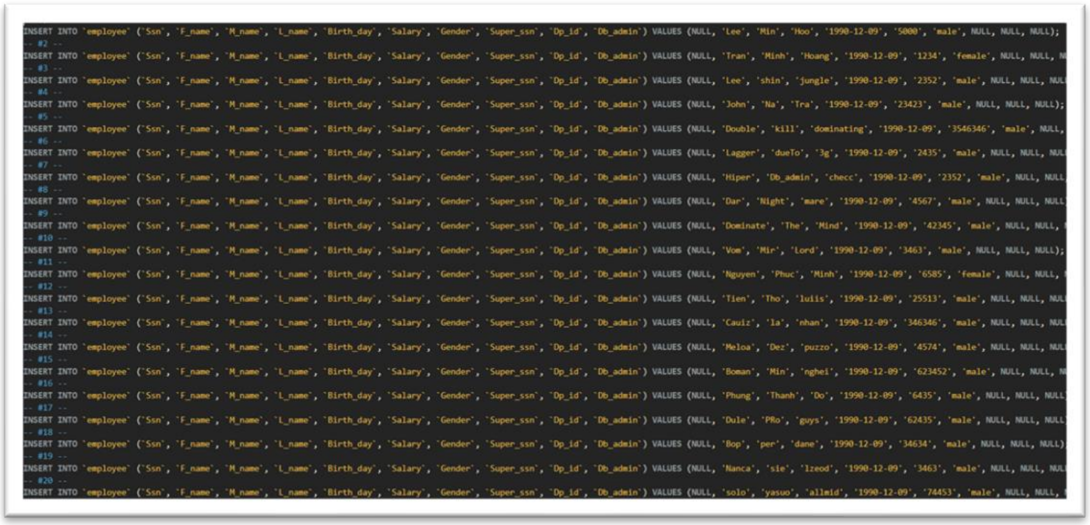
- If we set an attribute with NOT NULL key word then this attribute must have some value.
- If we set the data to integer it cannot be changed to other types (i.e. in case that we try to insert varchar to an integer attribute, the system will raise an error).
- Furthermore, if we set an attribute to auto increment, when giving value to this column it will automatically increase the value compared to the highest value in the column.
- Etc.

The following two pictures illustrate how we implement the table and data:



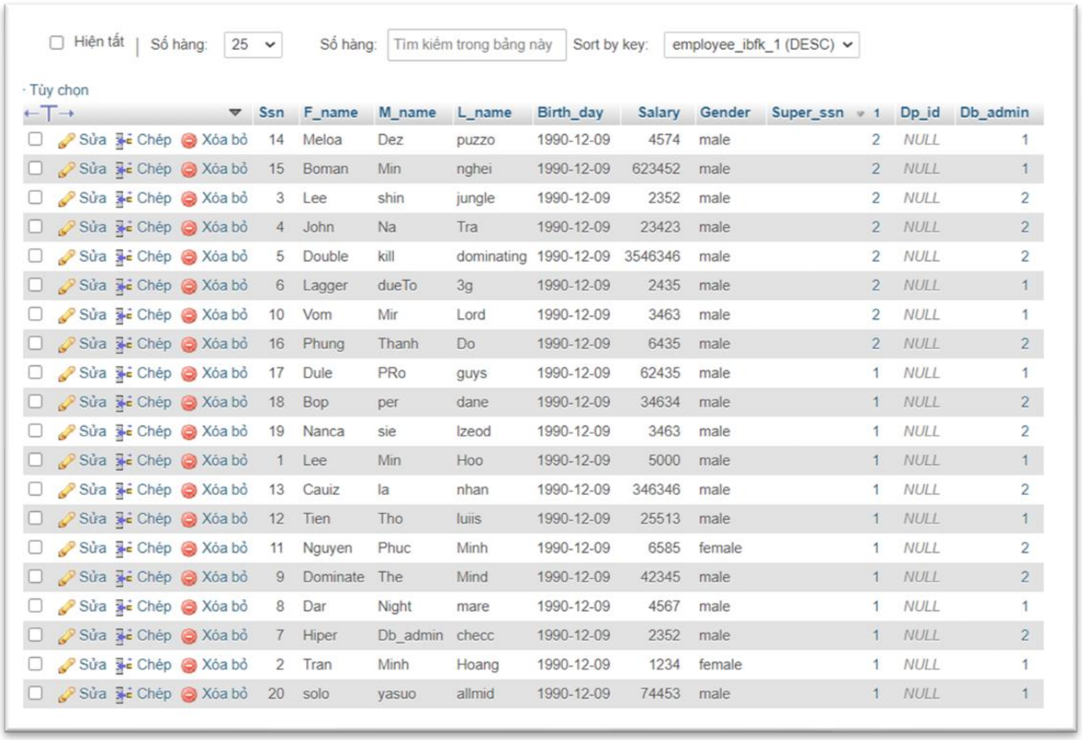
```
-- -----------------------------------------------------
-- Table `humanresource`.`EMPLOYEE`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `EMPLOYEE` (
  `Ssn` INT NOT NULL AUTO_INCREMENT,
  `F_name` VARCHAR(20) NULL DEFAULT NULL,
  `M_name` VARCHAR(20) NULL DEFAULT NULL,
  `L_name` VARCHAR(20) NULL DEFAULT NULL,
  `Birth_day` DATE NULL DEFAULT NULL,
  `Salary` INT NULL DEFAULT NULL,
  `Gender` VARCHAR(20) NULL DEFAULT NULL,
  `Super_ssn` INT NULL DEFAULT NULL,
  `Dp_id` INT NULL DEFAULT NULL,
  `Db_admin` INT DEFAULT NULL,
  PRIMARY KEY (`Ssn`)
)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- done alter
```

Inserting data into the database:



```
INSERT INTO `employee` (`Ssn`, `F_name`, `M_name`, `L_name`, `Birth_day`, `Salary`, `Gender`, `Super_ssn`, `Dp_id`, `Db_admin`) VALUES (NULL, 'Lee', 'Min', 'Hoo', '1990-12-09', '5000', 'male', NULL, NULL, NULL);
-- #2 --
INSERT INTO `employee` (`Ssn`, `F_name`, `M_name`, `L_name`, `Birth_day`, `Salary`, `Gender`, `Super_ssn`, `Dp_id`, `Db_admin`) VALUES (NULL, 'Tran', 'Minh', 'Hoang', '1990-12-09', '1234', 'female', NULL, NULL, N
-- #3 --
INSERT INTO `employee` (`Ssn`, `F_name`, `M_name`, `L_name`, `Birth_day`, `Salary`, `Gender`, `Super_ssn`, `Dp_id`, `Db_admin`) VALUES (NULL, 'Lee', 'shin', 'jungle', '1990-12-09', '2352', 'male', NULL, NULL, NUL
-- #4 --
INSERT INTO `employee` (`Ssn`, `F_name`, `M_name`, `L_name`, `Birth_day`, `Salary`, `Gender`, `Super_ssn`, `Dp_id`, `Db_admin`) VALUES (NULL, 'John', 'Na', 'Tra', '1990-12-09', '23423', 'male', NULL, NULL, NULL);
-- #5 --
INSERT INTO `employee` (`Ssn`, `F_name`, `M_name`, `L_name`, `Birth_day`, `Salary`, `Gender`, `Super_ssn`, `Dp_id`, `Db_admin`) VALUES (NULL, 'Double', 'kill', 'dominating', '1990-12-09', '3546346', 'male', NULL,
-- #6 --
INSERT INTO `employee` (`Ssn`, `F_name`, `M_name`, `L_name`, `Birth_day`, `Salary`, `Gender`, `Super_ssn`, `Dp_id`, `Db_admin`) VALUES (NULL, 'Lagger', 'dueTo', '3g', '1990-12-09', '2435', 'male', NULL, NULL, NUL
-- #7 --
INSERT INTO `employee` (`Ssn`, `F_name`, `M_name`, `L_name`, `Birth_day`, `Salary`, `Gender`, `Super_ssn`, `Dp_id`, `Db_admin`) VALUES (NULL, 'Hiper', 'Db_admin', 'checc', '1990-12-09', '2352', 'male', NULL, NULL
-- #8 --
INSERT INTO `employee` (`Ssn`, `F_name`, `M_name`, `L_name`, `Birth_day`, `Salary`, `Gender`, `Super_ssn`, `Dp_id`, `Db_admin`) VALUES (NULL, 'Dar', 'Night', 'mare', '1990-12-09', '4567', 'male', NULL, NULL, NULL
-- #9 --
INSERT INTO `employee` (`Ssn`, `F_name`, `M_name`, `L_name`, `Birth_day`, `Salary`, `Gender`, `Super_ssn`, `Dp_id`, `Db_admin`) VALUES (NULL, 'Dominate', 'The', 'Mind', '1990-12-09', '42345', 'male', NULL, NULL,
-- #10 --
INSERT INTO `employee` (`Ssn`, `F_name`, `M_name`, `L_name`, `Birth_day`, `Salary`, `Gender`, `Super_ssn`, `Dp_id`, `Db_admin`) VALUES (NULL, 'Vom', 'Mir', 'Lord', '1990-12-09', '3463', 'male', NULL, NULL, NULL);
-- #11 --
INSERT INTO `employee` (`Ssn`, `F_name`, `M_name`, `L_name`, `Birth_day`, `Salary`, `Gender`, `Super_ssn`, `Dp_id`, `Db_admin`) VALUES (NULL, 'Nguyen', 'Phuc', 'Minh', '1990-12-09', '6585', 'female', NULL, NULL,
-- #12 --
INSERT INTO `employee` (`Ssn`, `F_name`, `M_name`, `L_name`, `Birth_day`, `Salary`, `Gender`, `Super_ssn`, `Dp_id`, `Db_admin`) VALUES (NULL, 'Tien', 'Tho', 'luiis', '1990-12-09', '25513', 'male', NULL, NULL, NUL
-- #13 --
INSERT INTO `employee` (`Ssn`, `F_name`, `M_name`, `L_name`, `Birth_day`, `Salary`, `Gender`, `Super_ssn`, `Dp_id`, `Db_admin`) VALUES (NULL, 'Cauiz', 'la', 'nhan', '1990-12-09', '346346', 'male', NULL, NULL, NUL
-- #14 --
INSERT INTO `employee` (`Ssn`, `F_name`, `M_name`, `L_name`, `Birth_day`, `Salary`, `Gender`, `Super_ssn`, `Dp_id`, `Db_admin`) VALUES (NULL, 'Meloa', 'Dez', 'puzzo', '1990-12-09', '4574', 'male', NULL, NULL, NUL
-- #15 --
INSERT INTO `employee` (`Ssn`, `F_name`, `M_name`, `L_name`, `Birth_day`, `Salary`, `Gender`, `Super_ssn`, `Dp_id`, `Db_admin`) VALUES (NULL, 'Boman', 'Min', 'nghei', '1990-12-09', '623452', 'male', NULL, NULL, N
-- #16 --
INSERT INTO `employee` (`Ssn`, `F_name`, `M_name`, `L_name`, `Birth_day`, `Salary`, `Gender`, `Super_ssn`, `Dp_id`, `Db_admin`) VALUES (NULL, 'Phung', 'Thanh', 'Do', '1990-12-09', '6435', 'male', NULL, NULL, NULL
-- #17 --
INSERT INTO `employee` (`Ssn`, `F_name`, `M_name`, `L_name`, `Birth_day`, `Salary`, `Gender`, `Super_ssn`, `Dp_id`, `Db_admin`) VALUES (NULL, 'Dule', 'PRo', 'guys', '1990-12-09', '62435', 'male', NULL, NULL, NULL
-- #18 --
INSERT INTO `employee` (`Ssn`, `F_name`, `M_name`, `L_name`, `Birth_day`, `Salary`, `Gender`, `Super_ssn`, `Dp_id`, `Db_admin`) VALUES (NULL, 'Bop', 'per', 'dane', '1990-12-09', '34634', 'male', NULL, NULL, NULL);
-- #19 --
INSERT INTO `employee` (`Ssn`, `F_name`, `M_name`, `L_name`, `Birth_day`, `Salary`, `Gender`, `Super_ssn`, `Dp_id`, `Db_admin`) VALUES (NULL, 'Nanca', 'sie', 'Izeod', '1990-12-09', '3463', 'male', NULL, NULL, NUL
-- #20 --
INSERT INTO `employee` (`Ssn`, `F_name`, `M_name`, `L_name`, `Birth_day`, `Salary`, `Gender`, `Super_ssn`, `Dp_id`, `Db_admin`) VALUES (NULL, 'solo', 'yasuo', 'allmid', '1990-12-09', '74453', 'male', NULL, NULL,
```

The results after inserting:



Hiện tất | Số hàng: 25 Số hàng: Tìm kiếm trong bảng này Sort by key: employee_ibfk_1 (DESC)

· Tùy chọn

	Ssn	F_name	M_name	L_name	Birth_day	Salary	Gender	Super_ssn 1	Dp_id	Db_admin
Sửa Chép Xóa bỏ	14	Meloa	Dez	puzzo	1990-12-09	4574	male	2	NULL	1
Sửa Chép Xóa bỏ	15	Boman	Min	nghei	1990-12-09	623452	male	2	NULL	1
Sửa Chép Xóa bỏ	3	Lee	shin	jungle	1990-12-09	2352	male	2	NULL	2
Sửa Chép Xóa bỏ	4	John	Na	Tra	1990-12-09	23423	male	2	NULL	2
Sửa Chép Xóa bỏ	5	Double	kill	dominating	1990-12-09	3546346	male	2	NULL	2
Sửa Chép Xóa bỏ	6	Lagger	dueTo	3g	1990-12-09	2435	male	2	NULL	1
Sửa Chép Xóa bỏ	10	Vom	Mir	Lord	1990-12-09	3463	male	2	NULL	1
Sửa Chép Xóa bỏ	16	Phung	Thanh	Do	1990-12-09	6435	male	2	NULL	2
Sửa Chép Xóa bỏ	17	Dule	PRo	guys	1990-12-09	62435	male	1	NULL	1
Sửa Chép Xóa bỏ	18	Bop	per	dane	1990-12-09	34634	male	1	NULL	2
Sửa Chép Xóa bỏ	19	Nanca	sie	Izeod	1990-12-09	3463	male	1	NULL	2
Sửa Chép Xóa bỏ	1	Lee	Min	Hoo	1990-12-09	5000	male	1	NULL	1
Sửa Chép Xóa bỏ	13	Cauiz	la	nhan	1990-12-09	346346	male	1	NULL	2
Sửa Chép Xóa bỏ	12	Tien	Tho	luiis	1990-12-09	25513	male	1	NULL	1
Sửa Chép Xóa bỏ	11	Nguyen	Phuc	Minh	1990-12-09	6585	female	1	NULL	2
Sửa Chép Xóa bỏ	9	Dominate	The	Mind	1990-12-09	42345	male	1	NULL	2
Sửa Chép Xóa bỏ	8	Dar	Night	mare	1990-12-09	4567	male	1	NULL	1
Sửa Chép Xóa bỏ	7	Hiper	Db_admin	checc	1990-12-09	2352	male	1	NULL	2
Sửa Chép Xóa bỏ	2	Tran	Minh	Hoang	1990-12-09	1234	female	1	NULL	1
Sửa Chép Xóa bỏ	20	solo	yasuo	allmid	1990-12-09	74453	male	1	NULL	1

Compared to assignment 1 we have added and changed a lot of attributes and constraints. The first thing that we had to complete is to implement all the tables into the schema. In this case, due to the use of phpMyAdmin then we have to import all the tables then lately.

Foreign key

All tables in the database system have a way to create constrains, one of the most important is foreign key. When we implement the database, we have to make the schema knows which is the key of the table and how can we create relationship between tables via the foreign key.

There are three actions we can set while applying a foreign key:

- Set Null: clear the foreign key
- Cascade: follow the foreign key
- No action: means do nothing
- (Restrict: in MySQL, this is the same as No action)

Table SQL code

The table SQL codes are shown in pictures below:

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+07:00";


-- -----------------------------------------------------
-- Schema humanresource for mySQL
-- -----------------------------------------------------



-- -----------------------------------------------------
-- Table `humanresource`.`CLIENT`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `ADMINS` (
 `id` INT NOT NULL AUTO_INCREMENT,
 `username` VARCHAR(32) NOT NULL,
 `password` VARCHAR(256) NOT NULL,
 PRIMARY KEY (`id`)
)
```

```
ENGINE=InnoDB DEFAULT CHARSET=utf8;


-- -----------------------------------------------------
-- Table `humanresource`.`CLIENT`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CLIENT` (
  `Client_id` INT NOT NULL AUTO_INCREMENT,
  `Email` VARCHAR(20) NULL DEFAULT NULL,
  `Phone_num` INT NULL DEFAULT NULL,
  `Birth_day` DATE NULL DEFAULT NULL,
  PRIMARY KEY (`Client_id`)
)
ENGINE=InnoDB DEFAULT CHARSET=utf8;



-- -----------------------------------------------------
-- Table `humanresource`.`EMPLOYEE`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `EMPLOYEE` (
  `Ssn` INT NOT NULL AUTO_INCREMENT,
  `F_name` VARCHAR(20) NULL DEFAULT NULL,
  `M_name` VARCHAR(20) NULL DEFAULT NULL,
  `L_name` VARCHAR(20) NULL DEFAULT NULL,
  `Birth_day` DATE NULL DEFAULT NULL,
  `Salary` INT NULL DEFAULT NULL,
  `Gender` VARCHAR(20) NULL DEFAULT NULL,
  `Super_ssn` INT NULL DEFAULT NULL,
  `Dp_id` INT NULL DEFAULT NULL,
  `Db_admin` INT DEFAULT NULL,
  PRIMARY KEY (`Ssn`)
)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- done alter


-- -----------------------------------------------------
-- Table `humanresource`.`DEPARTMENT`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `DEPARTMENT` (
  `Dp_id` INT NOT NULL AUTO_INCREMENT,
  `Dp_name` VARCHAR(20) NULL DEFAULT NULL,
  `Dp_type` VARCHAR(20) NULL DEFAULT NULL,
  `Dp_location` VARCHAR(20) NULL DEFAULT NULL,
 `Dp_manager` INT NULL,
 PRIMARY KEY (`Dp_id`)
)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- done alter


-- -----------------------------------------------------
-- Table `humanresource`.`CLIENT_PROJECT`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CLIENT_PROJECT` (
 `Work_id` INT NOT NULL AUTO_INCREMENT,
 `Started_day` DATE NULL DEFAULT NULL,
 `Delivery_day` DATE NULL DEFAULT NULL,
 `Dnumber` INT NULL DEFAULT NULL,
 `C_no` INT NULL DEFAULT NULL,
 PRIMARY KEY (`Work_id`)
)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- done alter




-- -----------------------------------------------------
-- Table `humanresource`.`DEPENDENT`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `DEPENDENT` (
 `Essn` INT NOT NULL AUTO_INCREMENT,
 `Dependent_name` VARCHAR(20) NOT NULL,
 `Gender` VARCHAR(20) NULL DEFAULT NULL,
 `Birth_day` DATE NULL DEFAULT NULL,
 `Relationship` VARCHAR(20) NULL DEFAULT NULL,
 PRIMARY KEY (`Essn`, `Dependent_name`)
)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- done alter


-- -----------------------------------------------------
-- Table `humanresource`.`TRAIN_COURSE`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `TRAIN_COURSE` (
 `Train_id` INT NOT NULL,
 `Hour` INT NULL DEFAULT NULL,
 `Course_type` VARCHAR(20) NULL DEFAULT NULL,
 PRIMARY KEY (`Train_id`)
)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- no alter



-- -----------------------------------------------------
-- Table `humanresource`.`HAS`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `HAS` (
 `Essn` INT NOT NULL AUTO_INCREMENT,
 `Cno` INT NOT NULL,
 `Cno2` INT NOT NULL,
 PRIMARY KEY (`Essn`, `Cno`, `Cno2`)
)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- done alter





-- -----------------------------------------------------
-- Table `humanresource`.`PROJECT`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `PROJECT` (
 `Work_id` INT NOT NULL,
 `Hour` INT NULL DEFAULT NULL,
 `Type` VARCHAR(20) NULL DEFAULT NULL,
 `Dnum` INT NULL DEFAULT NULL,
 `DEPARTMENT_Dp_id` INT NOT NULL,
 PRIMARY KEY (`Work_id`, `DEPARTMENT_Dp_id`)
)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- done alter





-- -----------------------------------------------------
-- Table `humanresource`.`RELEASE_TIME`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `RELEASE_TIME` (
 `Em_ssn` INT NOT NULL AUTO_INCREMENT,
 `R_date` DATE NOT NULL,
 `Day_count` INT NULL DEFAULT NULL,
 `Type` VARCHAR(20) NULL DEFAULT NULL,
 `employee_Ssn` INT NOT NULL,
 PRIMARY KEY (`Em_ssn`, `R_date`)
)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- done alter




-- -----------------------------------------------------
-- Table `humanresource`.`WORKS_ON`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `WORKS_ON` (
 `Essn` INT NOT NULL AUTO_INCREMENT,
 `Pno` INT NOT NULL,
 `Hour` INT NULL DEFAULT NULL,
 PRIMARY KEY (`Essn`, `Pno`)
)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- done alter
-- -----------------------------------------------------
-- Table `humanresource`.`WORKED`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `WORKED` (
 `employee_Ssn` INT NOT NULL,
 `client_project_Work_id` INT NOT NULL,
 PRIMARY KEY (`employee_Ssn`, `client_project_Work_id`)
)
ENGINE=InnoDB DEFAULT CHARSET=utf8;




-- -----------------------------------------------------
-- Table `humanresource`.`DepartmentLocation`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `DepartmentLocation` (
 `Dp_location` VARCHAR(32) NOT NULL,
 `Dp_id` INT NOT NULL,
 PRIMARY KEY (`Dp_location`, `Dp_id`)
)
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Foreign key SQL code

Below are the codes for SQL foreign keys:

```
ALTER TABLE `EMPLOYEE`
  ADD (CONSTRAINT `employee_ibfk_1` FOREIGN KEY (`Super_ssn`) REFERENCES `EMPLOYEE` (`Ssn`
) ON DELETE NO ACTION ON UPDATE NO ACTION,
      CONSTRAINT `employee_ibfk_4` FOREIGN KEY (`Dp_id`) REFERENCES `DEPARTMENT` (`Dp_id`) O
N DELETE NO ACTION ON UPDATE NO ACTION);

ALTER TABLE `EMPLOYEE`
  ADD (CONSTRAINT `em_db_manager` FOREIGN KEY (`Db_admin`) REFERENCES `ADMINS` (`id`)  ON
 DELETE NO ACTION ON UPDATE NO ACTION);

ALTER TABLE `DEPARTMENT`
  ADD (CONSTRAINT `employee_manager` FOREIGN KEY (`Dp_manager`) REFERENCES `EMPLOYEE` (
`Ssn`)  ON DELETE NO ACTION ON UPDATE NO ACTION,
      CONSTRAINT `dp_location` FOREIGN KEY (`Dp_location`) REFERENCES `DepartmentLocation` (`Dp
_location`)  ON DELETE NO ACTION ON UPDATE NO ACTION);

ALTER TABLE `CLIENT_PROJECT`
  ADD (CONSTRAINT `client_project_ibfk_1` FOREIGN KEY (`Dnumber`) REFERENCES `DEPARTMENT`
 (`Dp_id`)  ON DELETE NO ACTION ON UPDATE NO ACTION,
      CONSTRAINT `client_project_ibfk_3` FOREIGN KEY (`C_no`) REFERENCES `CLIENT` (`Client_id`)  O
N DELETE NO ACTION ON UPDATE NO ACTION);


ALTER TABLE `DEPENDENT`
  ADD (CONSTRAINT `dependent_ibfk_1` FOREIGN KEY (`Essn`) REFERENCES `EMPLOYEE` (`Ssn`)  O
N DELETE NO ACTION ON UPDATE NO ACTION);

ALTER TABLE `HAS`
  ADD (CONSTRAINT `has_ibfk_1` FOREIGN KEY (`Essn`) REFERENCES `EMPLOYEE` (`Ssn`)  ON DELE
TE NO ACTION ON UPDATE NO ACTION,
      CONSTRAINT `has_ibfk_2` FOREIGN KEY (`Cno2`) REFERENCES `TRAIN_COURSE` (`Train_id`)  ON
DELETE NO ACTION ON UPDATE NO ACTION);

ALTER TABLE `PROJECT`
  ADD (CONSTRAINT `fk_PROJECT_DEPARTMENT1` FOREIGN KEY (`DEPARTMENT_Dp_id`) REFEREN
CES `DEPARTMENT` (`Dp_id`)  ON DELETE NO ACTION ON UPDATE NO ACTION);

ALTER TABLE `RELEASE_TIME`
```

```
  ADD (CONSTRAINT `fk_release_time_employee1` FOREIGN KEY (`employee_Ssn`) REFERENCES `EM
PLOYEE` (`Ssn`)  ON DELETE NO ACTION ON UPDATE NO ACTION);

ALTER TABLE `WORKS_ON`
  ADD (CONSTRAINT `works_on_ibfk_1` FOREIGN KEY (`Essn`) REFERENCES `EMPLOYEE` (`Ssn`)  ON
DELETE NO ACTION ON UPDATE NO ACTION,
     CONSTRAINT `works_on_ibfk_2` FOREIGN KEY (`Pno`) REFERENCES `PROJECT` (`Work_id`)  ON D
ELETE NO ACTION ON UPDATE NO ACTION);


ALTER TABLE `WORKED`
  ADD (CONSTRAINT `fk_employee_has_client_project_employee1` FOREIGN KEY (`employee_Ssn`) RE
FERENCES `EMPLOYEE` (`Ssn`)  ON DELETE NO ACTION ON UPDATE NO ACTION,
     CONSTRAINT `fk_employee_has_client_project_client_project1` FOREIGN KEY (`client_project_Wor
k_id`) REFERENCES `CLIENT_PROJECT` (`Work_id`)  ON DELETE NO ACTION ON UPDATE NO ACTION)
;
```

Related points

In the DBMS we have 6 relationship types:

- One-to-one non-identifying relationship
- One-to-many non-identifying relationship
- One-to-one identifying relationship
- One-to-many identifying relationship
- Many-to-many identifying relationship
- Placing a Relationship Using Existing Columns

Some other important keys in MySQL which are applied in the project:

- PRIMARY KEY states that an attribute is a primary key of the table
- CONSTRAINT specifies rules for the data in a table

Normalization

Normalization is the process of trying to keep all data non-redundant. Instead of repeating lengthy values such as names and addresses, assign them with unique IDs, repeat these IDs as needed across multiple smaller tables, and join the tables in queries by referencing the IDs in the join clause. This process comes at the cost of speed but is greatly efficient on storage.

First normalization form

The Conceptual Design of our database system contains no multiple values or composite values. All tables contain only singular values, specified by specific ID or primary key.

At first, we actually had to deal with a lot of hidden multiple values (which couldn't be seen before implementation because of our lack of experience). The process of producing the schema and inserting data showed that there were some multiple attributes that have to be changed into specific tables. For example, at first the table department had location ID, but during implementation, we found that each department could have more than one location, then we had to change the structure of this table.

Second normalization form

The tables at first also had partial key of dependency. Specifically, the table `dependent` of `employee` contained a partial key, then we had to remove it because this attribute itself had no meaning in this case (after considering whether we should use or remove). Furthermore, the 2NF helped reduce the redundancy in the database. If we just apply first normalization form, it will not be sufficient for any kind of small-medium up to larger scale of database system.

Third normalization form

From the conceptual design, we already applied all knowledge to reduce the cost of manipulating and updating the database. For the third normalization form, the conceptual and mapping part already satisfied the factor "no transitive dependency" matching with all the conditions in the first and second normalization form. Then the 3NF produced a database design without second dependent except the primary key.

Further discussion on normalization

First of all, we want to state that 3NF is sufficient for our design. The Human Resource Database is not particularly demanding on database requirements (part 2 of the report). Therefore, we find the next level of normalization unnecessary - Boyce-Codd Normal Form. In fact, the design and normalization are close to the Boyce-Codd Normal Form. In our prediction, the scale of the Human Resource Database System will be applied for not just one company, which means there will be more automation in the inserting and updating processes. For the 5NF and 6NF, we do not support improving the design to that high level because those normalizations are really strict and hard to manipulate or change the design.

From application to security and manipulation

This part of the report will explain the next part's structure and how we illustrate the work flow. Instead of explaining everything before implementing the knowledge, we will attempt to explain those concepts with a top-down approach. In more detail, the flow will be:

Building website with Xampp which includes phpMyAdmin ->
-> Database manipulation with website examples ->
-> Discussing about security with hashing and DAC.

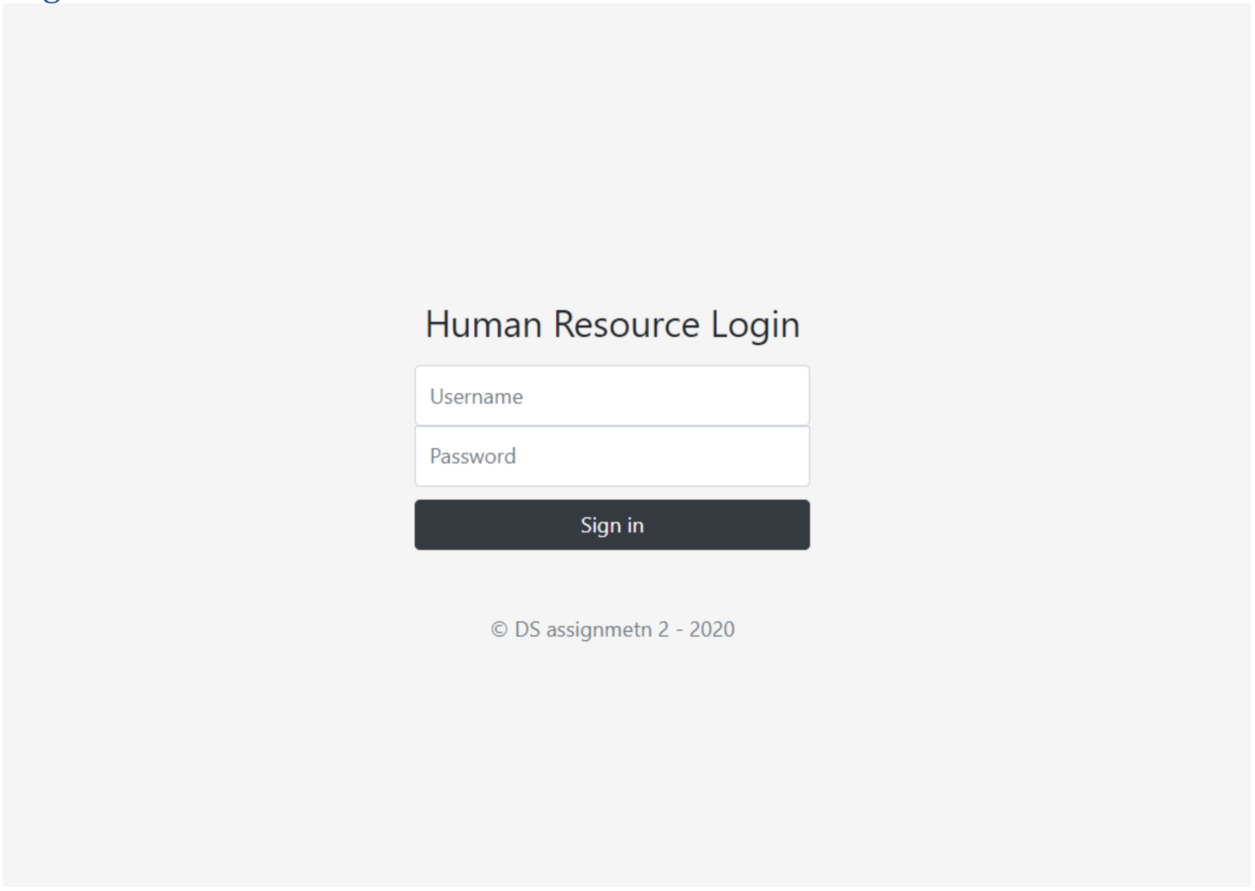
All these parts will be discussed carefully in the following parts.

Web application with phpMyAdmn

The website applies some main components as follow:

- Front-end: HTML, CSS, javascript
- Back-end: PHP applying for procedural querying process

Login form



In this part, the user who is the admin of the page has to sign in, the account is provided by the system. After they input the account information, and hit sign in button, the request will be sent to server and the following part will be processed:

```
if(isset($_POST['username']) and isset($_POST['password'])) {
   $res= $users_model->login($_POST);
}
else $res= false;

if ($res == true) {
   // login success
   echo "Success";
   // Save session
   $_SESSION['username'] = $_POST['username'];
   $_SESSION['ad_id'] = $users_model->getID($_POST);
   $url = 'http://'.$_SERVER['SERVER_NAME'].'/ds/manage;
   header("Location: $url");
}
```

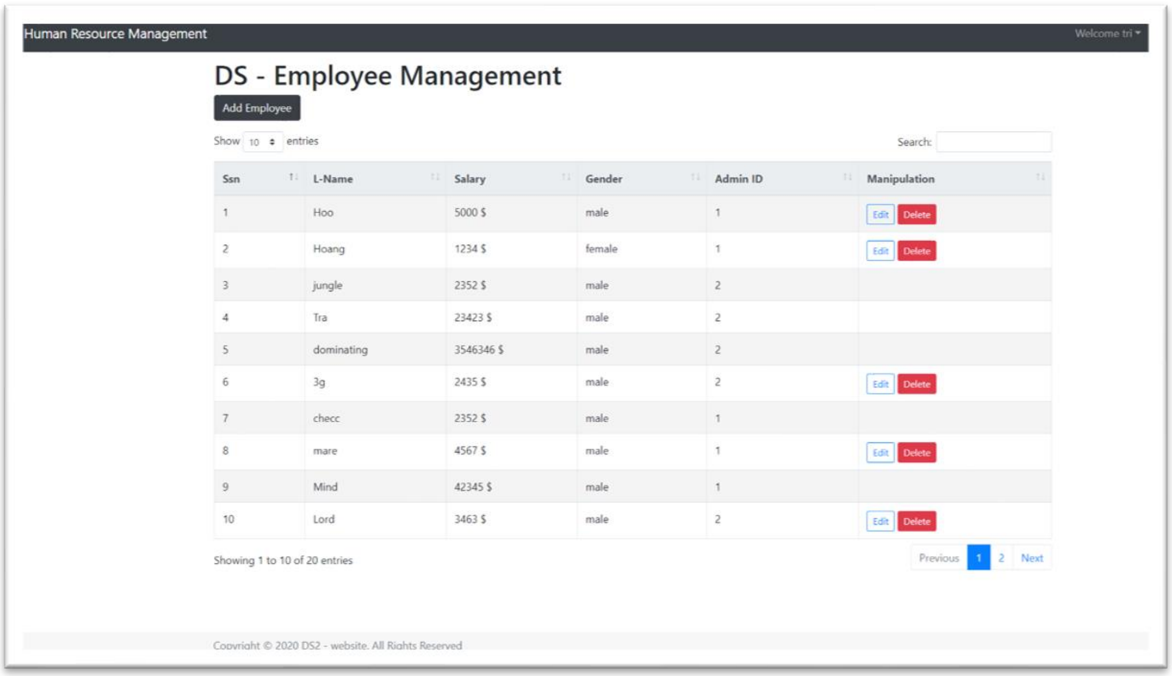
In case the login form fails we get an error message and have to remake the request sign in. The real SQL part happens behind this:

```
function login($post)
{
   $this->connect();
   $username = $post['username'];
   $password = $post['password'];
   $_SESSION['username'] = $username;

   $password_h =  hash("sha3-256", $password); //hash password

   $sql = "SELECT * FROM admins WHERE username='" . $username . "' AND password
='" . $password_h . "'";

   $result = $this->connection->query($sql);

   if ($result->num_rows == 0) {
      //Login failed
      // echo 'failed '.$username.' '.$password_h;
      $this->connection->close();
      echo '<script type="text/javascript">
```

```
            alert("Wrong Username and/or Password!");
            window.location = "http://' . $_SERVER['SERVER_NAME'] . '/ds/";
        </script>';
        return false;
    } else {
        $this->connection->close();
        return true;
    }
}
```

Main page

After the login successful, the website will render the main page for user, which is the "manage" page.



Human Resource Management

Welcome tri

DS - Employee Management

Add Employee

Show 10 entries

Search:

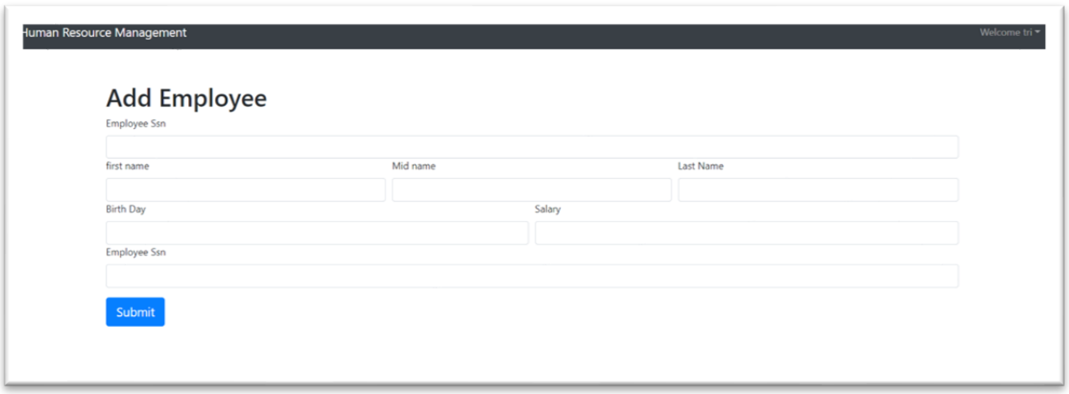
Ssn	L-Name	Salary	Gender	Admin ID	Manipulation
1	Hoo	5000 $	male	1	Edit Delete
2	Hoang	1234 $	female	1	Edit Delete
3	jungle	2352 $	male	2	
4	Tra	23423 $	male	2	
5	dominating	3546346 $	male	2	
6	3g	2435 $	male	2	Edit Delete
7	checc	2352 $	male	1	
8	mare	4567 $	male	1	Edit Delete
9	Mind	42345 $	male	1	
10	Lord	3463 $	male	2	Edit Delete

Showing 1 to 10 of 20 entries

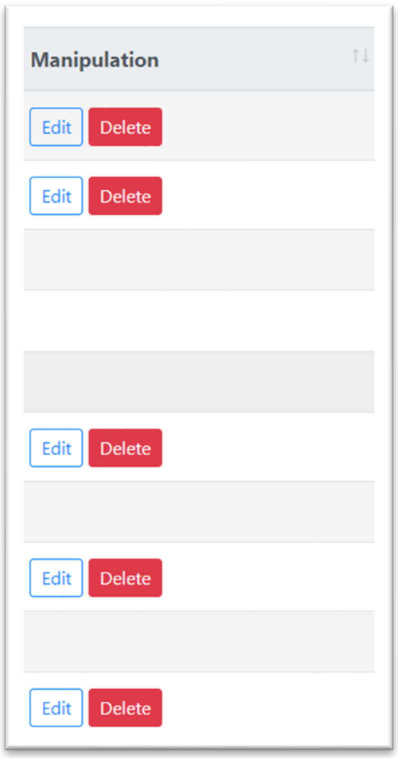
Previous 1 2 Next

Copyright © 2020 DS2 - website. All Rights Reserved

Main point of the application:
User after using the application can logout to prevent leaking data.
User can add more rows to the database, which is the employee information.

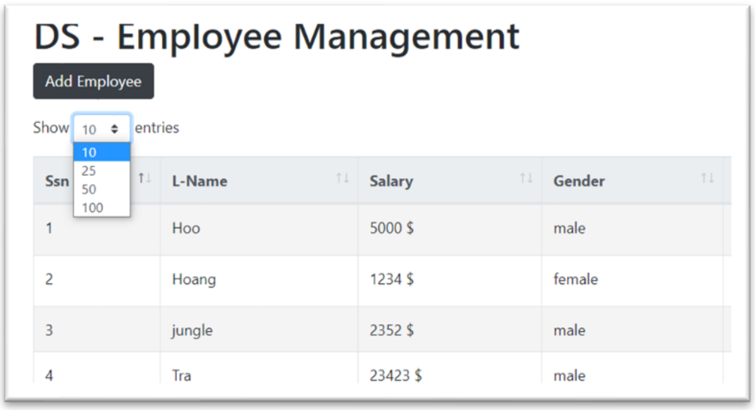


Back to the main page, user can only edit and delete the users whom they add, the object which doesn't have the creator key identical to the user's will not show the edit and delete button. For example, the missing button row actually doesn't belong to the current login account.



Sub function: showing number of rows and searching

Set number of entries:



DS - Employee Management

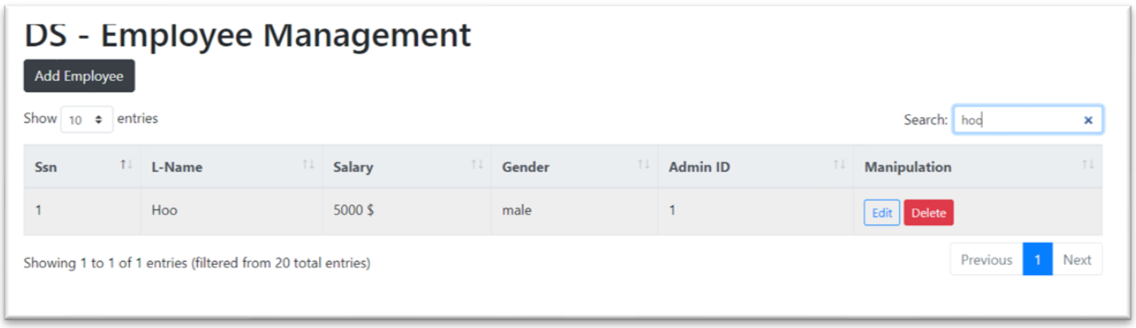
Add Employee

Show 10 entries

Ssn	L-Name	Salary	Gender
1	Hoo	5000 $	male
2	Hoang	1234 $	female
3	jungle	2352 $	male
4	Tra	23423 $	male

The page can show more or less rows depending on the chosen number of rows, this one is actually a bootstrap 4 function, which helps developer to build the website interface incredibly faster.

Search function:



DS - Employee Management

Add Employee

Show 10 entries

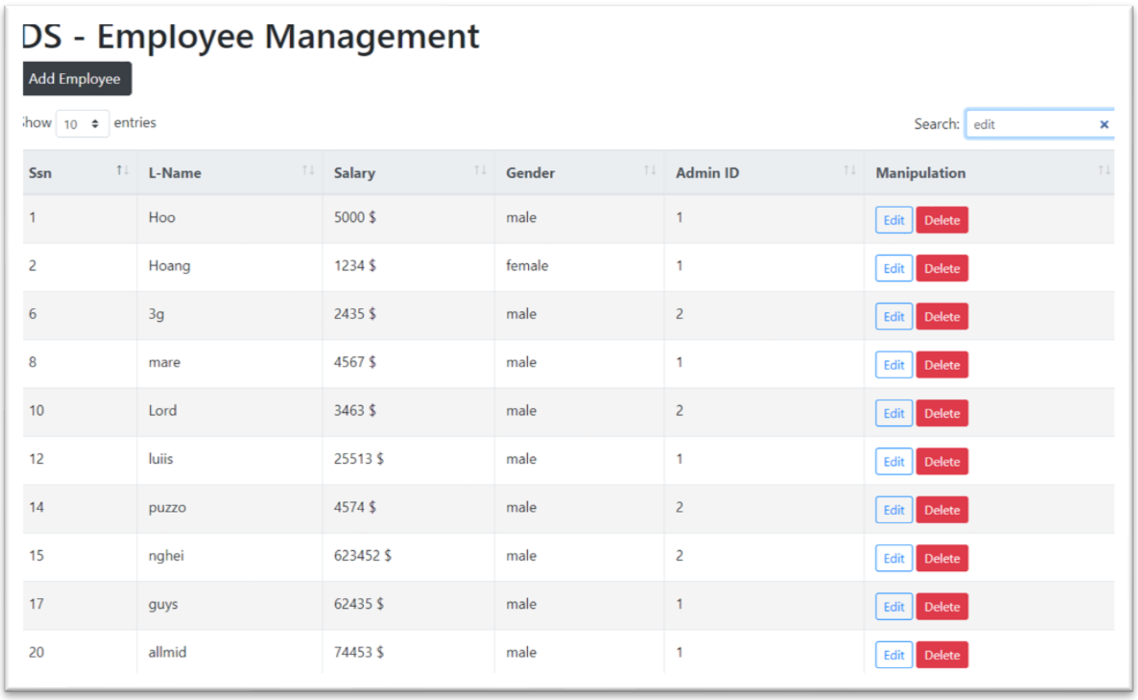
Search: hoo

Ssn	L-Name	Salary	Gender	Admin ID	Manipulation
1	Hoo	5000 $	male	1	Edit Delete

Showing 1 to 1 of 1 entries (filtered from 20 total entries)

Previous 1 Next

As can be seen, we use this function for filtering some matched wanting conditions. In the below example, we search for Hoo, which is the name of an employee in the database and only his result shows up.



DS - Employee Management

Add Employee

how 10 entries

Search: edit

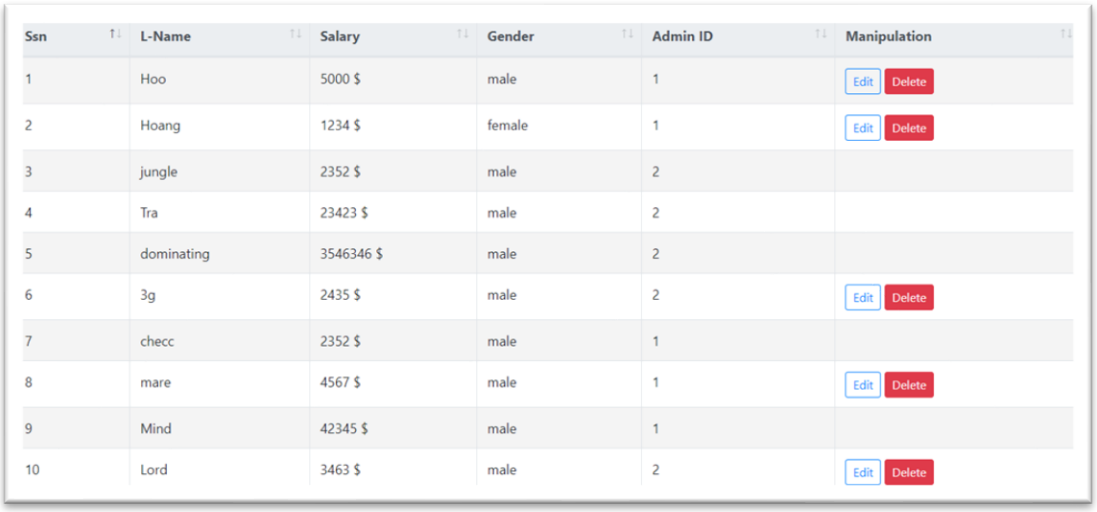
Ssn	L-Name	Salary	Gender	Admin ID	Manipulation
1	Hoo	5000 $	male	1	Edit Delete
2	Hoang	1234 $	female	1	Edit Delete
6	3g	2435 $	male	2	Edit Delete
8	mare	4567 $	male	1	Edit Delete
10	Lord	3463 $	male	2	Edit Delete
12	luiis	25513 $	male	1	Edit Delete
14	puzzo	4574 $	male	2	Edit Delete
15	nghei	623452 $	male	2	Edit Delete
17	guys	62435 $	male	1	Edit Delete
20	allmid	74453 $	male	1	Edit Delete

In the case that we just want to show the row that's created by the current logged in account, just enter edit, the page will automatically search for the text in the row, mean that any condition can be matched to the search bar.

Security

In this application we apply two methods for security: DAC on application level and hashing for password account.

Application level: each account can only modify user who was created by them



Ssn	L-Name	Salary	Gender	Admin ID	Manipulation
1	Hoo	5000 $	male	1	Edit Delete
2	Hoang	1234 $	female	1	Edit Delete
3	jungle	2352 $	male	2	
4	Tra	23423 $	male	2	
5	dominating	3546346 $	male	2	
6	3g	2435 $	male	2	Edit Delete
7	checc	2352 $	male	1	
8	mare	4567 $	male	1	Edit Delete
9	Mind	42345 $	male	1	
10	Lord	3463 $	male	2	Edit Delete

In the previous introduction, the data can be modified by signed in account, in this part, after logged in to the system, the back-end side will call the id of the user and compare to each row of data. If matched, the modify button will appear for modifying and deleting.

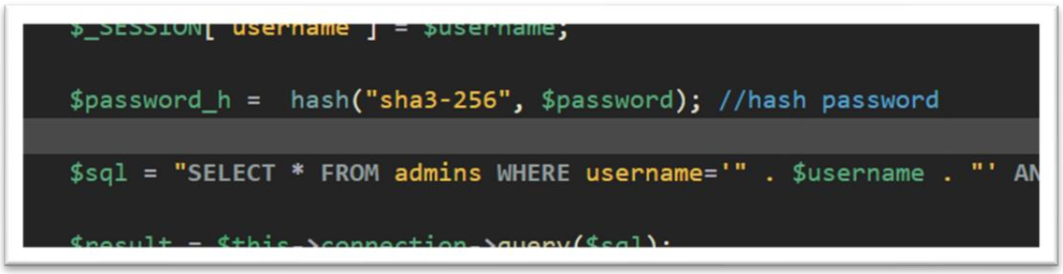
```
if ($res == true) {
   echo "Success";
   $_SESSION['username'] = $_POST['username'];
   $_SESSION['ad_id'] = $users_model->getID($_POST);
   $url = 'http://'.$_SERVER['SERVER_NAME'].'/ds/manage;
   header("Location: $url");
}
```

The given code says that if you successfully sign in to the system, there will be a session storing your account ID. After that, for each row of employee shown in the main page, it has to check whether the session and the Admin ID of the employee row

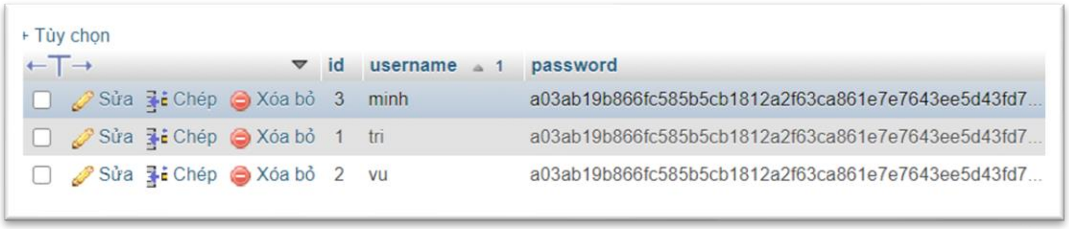
Extend: Super admin can make all changes even for employees that are not created by them, the super admin can take the power of the other lower admins.

Hashing password

It is popular to úe the hashing method for all the system that required the use of account. In this case, each account will be hashed with SHA 3, the hashing method and hashed password will have the look as in the following picture:



```
$_SESSION['username'] = $username;

$password_h =  hash("sha3-256", $password); //hash password

$sql = "SELECT * FROM admins WHERE username='" . $username . "' AN

$result = $this->connection->query($sql);
```



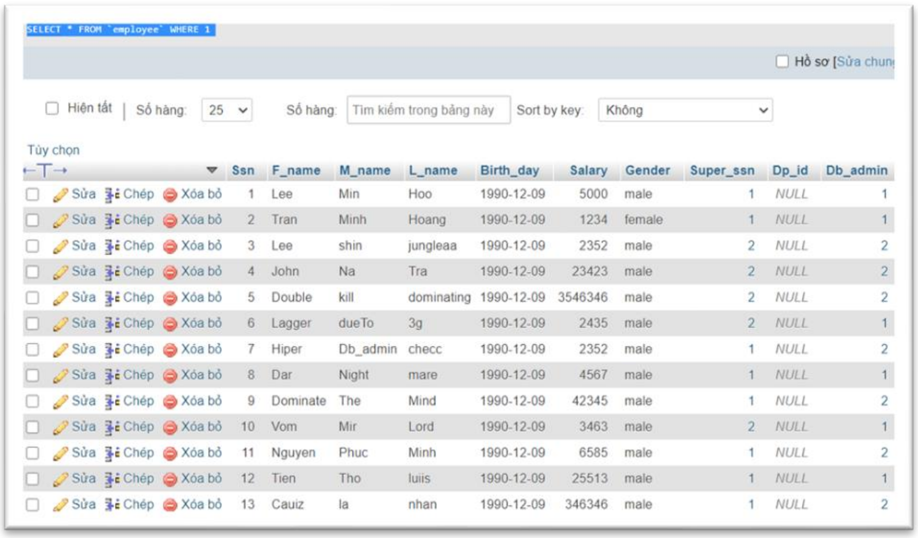
id	username	password
3	minh	a03ab19b866fc585b5cb1812a2f63ca861e7e7643ee5d43fd7...
1	tri	a03ab19b866fc585b5cb1812a2f63ca861e7e7643ee5d43fd7...
2	vu	a03ab19b866fc585b5cb1812a2f63ca861e7e7643ee5d43fd7...

Data Manipulation

Query

This part of the implementation, there will be two versions of querying, one is the simple query and the other is another more complex one.

Simple query: select all employee from table employee



```
SELECT * FROM `employee` WHERE 1
```

☐ Hồ sơ [Sửa chun

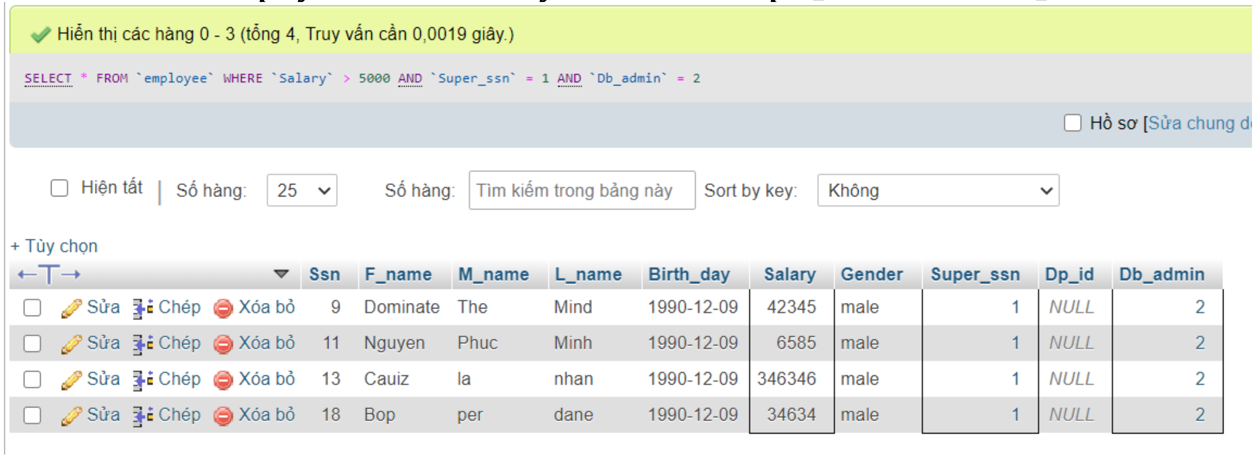
☐ Hiện tất | Số hàng: 25 Số hàng: Tìm kiếm trong bảng này Sort by key: Không

Tùy chọn

	Ssn	F_name	M_name	L_name	Birth_day	Salary	Gender	Super_ssn	Dp_id	Db_admin
☐ Sửa Chép Xóa bỏ	1	Lee	Min	Hoo	1990-12-09	5000	male	1	NULL	1
☐ Sửa Chép Xóa bỏ	2	Tran	Minh	Hoang	1990-12-09	1234	female	1	NULL	1
☐ Sửa Chép Xóa bỏ	3	Lee	shin	jungleaa	1990-12-09	2352	male	2	NULL	2
☐ Sửa Chép Xóa bỏ	4	John	Na	Tra	1990-12-09	23423	male	2	NULL	2
☐ Sửa Chép Xóa bỏ	5	Double	kill	dominating	1990-12-09	3546346	male	2	NULL	2
☐ Sửa Chép Xóa bỏ	6	Lagger	dueTo	3g	1990-12-09	2435	male	2	NULL	1
☐ Sửa Chép Xóa bỏ	7	Hiper	Db_admin	checc	1990-12-09	2352	male	1	NULL	2
☐ Sửa Chép Xóa bỏ	8	Dar	Night	mare	1990-12-09	4567	male	1	NULL	1
☐ Sửa Chép Xóa bỏ	9	Dominate	The	Mind	1990-12-09	42345	male	1	NULL	2
☐ Sửa Chép Xóa bỏ	10	Vom	Mir	Lord	1990-12-09	3463	male	2	NULL	1
☐ Sửa Chép Xóa bỏ	11	Nguyen	Phuc	Minh	1990-12-09	6585	male	1	NULL	2
☐ Sửa Chép Xóa bỏ	12	Tien	Tho	luiis	1990-12-09	25513	male	1	NULL	1
☐ Sửa Chép Xóa bỏ	13	Cauiz	la	nhan	1990-12-09	346346	male	1	NULL	2

More complex query:

SQL command:
SELECT * FROM `employee` WHERE `Salary` > 5000 AND `Super_ssn` = 1 AND `Db_admin` = 2



✓ Hiển thị các hàng 0 - 3 (tổng 4, Truy vấn cần 0,0019 giây.)

```
SELECT * FROM `employee` WHERE `Salary` > 5000 AND `Super_ssn` = 1 AND `Db_admin` = 2
```

☐ Hồ sơ [Sửa chung d

☐ Hiện tất | Số hàng: 25 Số hàng: Tìm kiếm trong bảng này Sort by key: Không

+ Tùy chọn

	Ssn	F_name	M_name	L_name	Birth_day	Salary	Gender	Super_ssn	Dp_id	Db_admin
☐ Sửa Chép Xóa bỏ	9	Dominate	The	Mind	1990-12-09	42345	male	1	NULL	2
☐ Sửa Chép Xóa bỏ	11	Nguyen	Phuc	Minh	1990-12-09	6585	male	1	NULL	2
☐ Sửa Chép Xóa bỏ	13	Cauiz	la	nhan	1990-12-09	346346	male	1	NULL	2
☐ Sửa Chép Xóa bỏ	18	Bop	per	dane	1990-12-09	34634	male	1	NULL	2

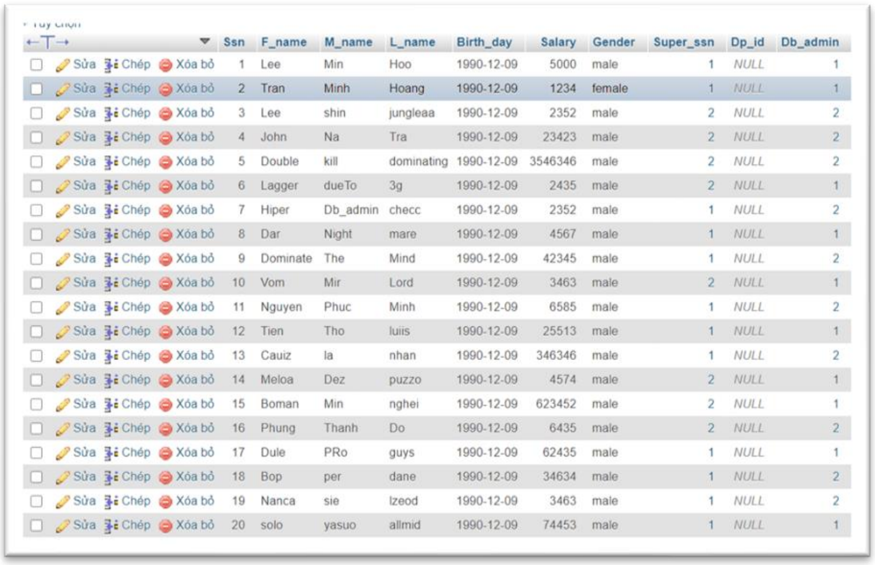
Trigger

For the implementation of trigger in this assignment, we use a trigger to store the changes of employee to track and back up for the case that there will be some misunderstanding actions.

The trigger creation below says that whenever an admin changes some information of table employee, the Ssn, the name and the supervisor ID will be stored in table employee temp for back track data.

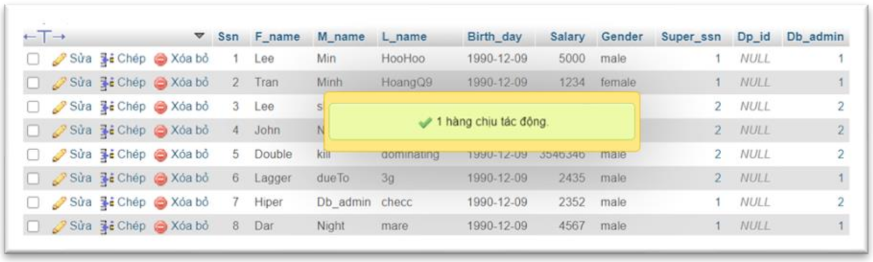
```
DELIMITER $$
 CREATE TRIGGER before_update_employee
 BEFORE UPDATE ON employee
 FOR EACH ROW
BEGIN
 INSERT INTO emp_tmp
SET
 ssb = OLD.Ssn,
 name = OLD.L_name,
 super_ssn = OLD.Super_ssn,
 date = NOW();
END$$
 DELIMITER ;
```

Here is an example that we processed:
Original data:



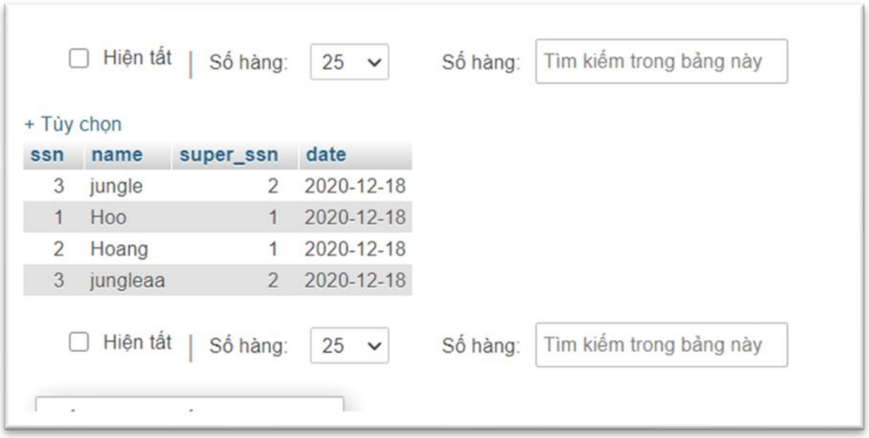
←T→	Ssn	F_name	M_name	L_name	Birth_day	Salary	Gender	Super_ssn	Dp_id	Db_admin
Sửa Chép Xóa bỏ	1	Lee	Min	Hoo	1990-12-09	5000	male	1	NULL	1
Sửa Chép Xóa bỏ	2	Tran	Minh	Hoang	1990-12-09	1234	female	1	NULL	1
Sửa Chép Xóa bỏ	3	Lee	shin	jungleaa	1990-12-09	2352	male	2	NULL	2
Sửa Chép Xóa bỏ	4	John	Na	Tra	1990-12-09	23423	male	2	NULL	2
Sửa Chép Xóa bỏ	5	Double	kill	dominating	1990-12-09	3546346	male	2	NULL	2
Sửa Chép Xóa bỏ	6	Lagger	dueTo	3g	1990-12-09	2435	male	2	NULL	1
Sửa Chép Xóa bỏ	7	Hiper	Db_admin	checc	1990-12-09	2352	male	1	NULL	2
Sửa Chép Xóa bỏ	8	Dar	Night	mare	1990-12-09	4567	male	1	NULL	1
Sửa Chép Xóa bỏ	9	Dominate	The	Mind	1990-12-09	42345	male	1	NULL	2
Sửa Chép Xóa bỏ	10	Vom	Mir	Lord	1990-12-09	3463	male	2	NULL	1
Sửa Chép Xóa bỏ	11	Nguyen	Phuc	Minh	1990-12-09	6585	male	1	NULL	2
Sửa Chép Xóa bỏ	12	Tien	Tho	luiis	1990-12-09	25513	male	1	NULL	1
Sửa Chép Xóa bỏ	13	Cauiz	la	nhan	1990-12-09	346346	male	1	NULL	2
Sửa Chép Xóa bỏ	14	Meloa	Dez	puzzo	1990-12-09	4574	male	2	NULL	1
Sửa Chép Xóa bỏ	15	Boman	Min	nghei	1990-12-09	623452	male	2	NULL	1
Sửa Chép Xóa bỏ	16	Phung	Thanh	Do	1990-12-09	6435	male	2	NULL	2
Sửa Chép Xóa bỏ	17	Dule	PRo	guys	1990-12-09	62435	male	1	NULL	1
Sửa Chép Xóa bỏ	18	Bop	per	dane	1990-12-09	34634	male	1	NULL	2
Sửa Chép Xóa bỏ	19	Nanca	sie	Izeod	1990-12-09	3463	male	1	NULL	2
Sửa Chép Xóa bỏ	20	solo	yasuo	allmid	1990-12-09	74453	male	1	NULL	1

After some changes:



	Ssn	F_name	M_name	L_name	Birth_day	Salary	Gender	Super_ssn	Dp_id	Db_admin
Sửa Chép Xóa bỏ	1	Lee	Min	HooHoo	1990-12-09	5000	male	1	NULL	1
Sửa Chép Xóa bỏ	2	Tran	Minh	HoangQ9	1990-12-09	1234	female	1	NULL	1
Sửa Chép Xóa bỏ	3	Lee	s	[illegible]	[illegible]	[illegible]	[illegible]	2	NULL	2
Sửa Chép Xóa bỏ	4	John	N	[illegible]	[illegible]	[illegible]	[illegible]	2	NULL	2
Sửa Chép Xóa bỏ	5	Double	kill	dominating	1990-12-09	3546346	male	2	NULL	2
Sửa Chép Xóa bỏ	6	Lagger	dueTo	3g	1990-12-09	2435	male	2	NULL	1
Sửa Chép Xóa bỏ	7	Hiper	Db_admin	checc	1990-12-09	2352	male	1	NULL	2
Sửa Chép Xóa bỏ	8	Dar	Night	mare	1990-12-09	4567	male	1	NULL	1

Data appear in table em_tpm:



Hiện tất | Số hàng: 25
Số hàng: Tìm kiếm trong bảng này
+ Tùy chọn

ssn	name	super_ssn	date
3	jungle	2	2020-12-18
1	Hoo	1	2020-12-18
2	Hoang	1	2020-12-18
3	jungleaa	2	2020-12-18

Hiện tất | Số hàng: 25
Số hàng: Tìm kiếm trong bảng này

Indexing

phpMyAdmin is a part of Xampp server for hosting local in this case. MySQL in Xampp has the ability to make index and due to that improves the total performance of the system. In our assignment, we tried to implement the Index, and make each primary key to become the index one.

But lately, in the implementation part, we recognized that MySQL in phpMyAdmin was already built in the ability to use Btree as an Indexing method, that explains why the system itself is so popular due to the performance it brings for developing php web server. In our point of view, the server is incredibly fast compared to the NON indexing one. Testing on other SQL servers which don't already use the primary key as the index like phpMyAdmin, the performance was the same with the small amount of data. However, when we generated a lot of samples to test it, and returned that after using Indexing, the performance for querying and searching was dramatically faster.