VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY

HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY

PROJECT REPORT

DATABASE SYSTEM

INSTRUCTOR: DR PHAN TRỌNG NHÂN
DR NGUYỄN THỊ ÁI THẢO

TEAM MEMBERS

Name	Student ID	Workload	Contribution
Văn Tuấn Anh (Leader)	2152400	Part 2,3,4; Report; Client-side; Server-side; Database; Data design; Mock data…	100%
Phan Quang Đạo	2152500	Part 1;	100%
Lê Sơn Cao	2152442	Draft report (without any explanation or usable content)	10%
Lương Trần Công Danh	2152459	Unreachable	0%
Lưu Quang Hoàng Cương	2152032	Unreachable	0%

1. PHYSICAL DATABASE DESIGN

Table: Employee

Field Name	Data Type	Data Length	Constraints	Description
emp_Code	VARCHAR(4)	4	PRIMARY KEY,NOT NULL, UNIQUE	Unique identifier for each employee
emp_DateOf Birth	DATE	10	NOT NULL	Every employee must have Date of Birth
emp_Address	VARCHAR(45)	45	NOT NULL	Every employee must have Address
emp_Specialit y	VARCHAR(45)	45		Employee's Speciality
emp_Degree's Year	int(4)	4	NOT NULL, UNSIGNED, ZEROFILL	Employee's Speciality Degree's Year
emp_FName	VARCHAR(45)	45	NOT NULL	Every employee must have First name
emp_LName	VARCHAR(45)	45	NOT NULL	Every employee must have Last name
emp_Phone	VARCHAR(10)	10	NOT NULL, UNIQUE	Every employee must have unique Phone number
emp_Gender	enum('Male','Fe male','Unknown')	7	NOT NULL, DEFAULT 'Unknown'	Every employee must have Gender (can only be Male, Female or Unknown)
emp_SDate	DATE	10	NOT NULL	Every employee must have Start day of work
dep_Code	VARCHAR(3)	3	NOT NULL	Foreign key referencing

				department.dep_Code, every employee must work in a department

Table: Department

Field Name	Data Type	Data Length	Constraints	Description
dep_Code	VARCHAR(3)	3	PRIMARY KEY,NOT NULL, UNIQUE	Unique identifier for each department
dep_Title	VARCHAR(45)	45	NOT NULL	Every department must have a title
dean_Code	INT(10)	10	NOT NULL	Foreign key referencing employee.emp_Code

Table: Doctor

Field Name	Data Type	Data Length	Constraints	Description
emp_Code	VARCHAR(4)	4	PRIMARY KEY,NOT NULL, UNIQUE	Foreign key referencing employee.emp_Code

Table: Nurse

Field Name	Data Type	Data Length	Constraints	Description
emp_Code	VARCHAR(4)	4	PRIMARY KEY,NOT NULL, UNIQUE	Foreign key referencing employee.emp_Code

Table: Patient

Field Name	Data Type	Data Length	Constraints	Description
patient_Code	VARCHAR(9)	9	PRIMARY KEY,NOT NULL, UNIQUE	Unique identifier for each patient, code length 9
patient_FName	VARCHAR(45)	45	NOT NULL	Every patient must have First name
patient_LName	VARCHAR(45)	45	NOT NULL	Every patient must have Last name
patient_Address	VARCHAR(45)	45	NOT NULL	Every patient must have Address
patient_DateOf Birth	DATE	10	NOT NULL	Every patient must have Date of Birth
patient_Gender	enum('Male','Female','Unknown')	7	NOT NULL, DEFAULT 'Unknown'	Every patient must have a Gender
patient_Phone	VARCHAR(45)	45	NOT NULL	Every patient must have their unique Phone No

Table: Outpatient

Field Name	Data Type	Data Length	Constraints	Description
character_Code	VARCHAR(11)	11	PRIMARY KEY,NOT NULL, UNIQUE	Unique identifier for each outpatient, start with ‘OP’
num_Code	VARCHAR(9)	9	PRIMARY KEY,NOT NULL, UNIQUE	Foreign key referencing patient.patient_Code

Table: Inpatient

Field Name	Data Type	Data Length	Constraints	Description
character_Code	VARCHAR(11)	11	PRIMARY KEY,NOT NULL, UNIQUE	Unique identifier for each outpatient, start with 'IP'
num_Code	VARCHAR(9)	9	PRIMARY KEY,NOT NULL, UNIQUE	Foreign key referencing patient.patient_Code
Date_Of_Admission	DATE	10	NOT NULL	Every inpatient must have Date of Admission
Date_Of_Discharge	DATE	10	DEFAULT NULL	Inpatient's Date of Discharge
Fee	DECIMAL(10,2)	5	DEFAULT NULL	Inpatient's hospital Fee, decimal can accommodate a wide range of values and stores two decimal places for precision.
Diagnosis	VARCHAR(200)	200	NOT NULL	Every inpatient must have Diagnosis
Sickroom	VARCHAR(45)	45	NOTNULL	Every inpatient must have Sickroom
nurse_Code	VARCHAR(4)	4	NOTNULL, UNSIGNED, ZEROFILL	Foreign key referencing nurse.emp_Code

Table: Provider

Field Name	Data Type	Data Length	Constraints	Description
prov_Code	INT(4)	4	PRIMARY KEY,NOT NULL, UNIQUE, UNSIGNED,	Unique identifier for each provider, zerofill to ensure code length 4 and

			ZEROFILL	unsigned to ensure there's no negative No
prov_Name	VARCHAR(45)	45	NOTNULL	Every provider must have their Name
prov_Phone	VARCHAR(10)	10	NOTNULL, UNIQUE	Every provider must have their unique Phone number
prov_Address	VARCHAR(45)	45	NOTNULL	Every provider must have Address

Table: Medication

Field Name	Data Type	Data Length	Constraints	Description
med_Code	VARCHAR(5)	5	PRIMARY KEY,NOT NULL, UNIQUE	Unique identifier for each medication
med_Name	VARCHAR(45)	45	NOT NULL	Every medication must have their name
med_Effects	VARCHAR(200)	200	NOT NULL	Every medication must have their effects
med_ExpDate	DATE	10	NOT NULL	Every medication must have expiration date
med_Fee	DECIMAL(10,2)	5	NOT NULL	Every medication must have their price, decimal can accommodate a wide range of values and stores two decimal places for precision.

Table: Examination

Field Name	Data Type	Data Length	Constraints	Description
character_Code	VARCHAR(11)	11	PRIMARY KEY,NOT NULL, UNIQUE	Foreign key referencing outpatient.character_ Code
num_Code	VARCHAR(9)	9	PRIMARY KEY,NOT NULL, UNIQUE	Foreign key referencing outpatient.num_Code
exam_Code	VARCHAR(4)	4	PRIMARY KEY,NOT NULL, UNIQUE	Unique identifier for each examination
doctor_CodeE	VARCHAR(4)	4	PRIMARY KEY,NOT NULL, UNIQUE	Foreign key referencing doctor.emp_Code
exam_Diagnosis	VARCHAR(200)	200	NOT NULL	Every examination must have diagnosis
exam_Fee	DECIMAL(10,2)	5	DEFAULT NULL	Examination's Fee
exam_Date	DATE	10	NOT NULL	Every examination must have conduct Date
exam_NDate	DATE	10	DEFAULT NULL, exam_NDate > exam_Date	Examination's next Date

Table: Import

Field Name	Data Type	Data Length	Constraints	Description
prov_Code	INT(4)	4	PRIMARY KEY,NOT NULL, UNIQUE, UNSIGNED, ZEROFILL	Foreign key referencing provider.prov_Code

med_Code	VARCHAR(5)	5	NOT NULL	Foreign key referencing medical.med_Code
imp_Date	DATE	10	NOT NULL	Every imports must have their date
Quantity	INT(4)	4	NOT NULL	Every imports must have their quantity
Price	DECIMAL(10,2)	5	DEFAULT NULL	Import's Price, decimal can accommodate a wide range of values and stores two decimal places for precision.

Table: Treatment

Field Name	Data Type	Data Length	Constraints	Description
character_Code	VARCHAR(11)	11	PRIMARY KEY,NOT NULL, UNIQUE	Foreign key referencing outpatient.character_ Code
num_Code	VARCHAR(9)	9	PRIMARY KEY,NOT NULL, UNIQUE	Foreign key referencing outpatient.num_Code
treat_Code	VARCHAR(4)	4	PRIMARY KEY,NOT NULL, UNIQUE	Unique identifier for each examination
doctor_CodeT	VARCHAR(4)	4	PRIMARY KEY,NOT NULL, UNIQUE	Foreign key referencing doctor.emp_Code
treat_SDate	DATE	10	NOT NULL	Every treatment must have a start date
treat_EDate	DATE	10	NOT NULL	Every treatment must have a end date
treat_Result	VARCHAR(200)	200	NOT NULL	Every treatment must

				have a result
treat_RecoverStatus	VARCHAR(45)	45	DEFAULT NULL	Treatment's Recover Status

Table: Applies_exam

Field Name	Data Type	Data Length	Constraints	Description
E_Pcharacter_Code	VARCHAR(11)	11	PRIMARY KEY,NOT NULL, UNIQUE	Foreign key referencing examination.character_Code
E_Pnum_Code	VARCHAR(9)	9	PRIMARY KEY,NOT NULL, UNIQUE	Foreign key referencing examination.num_Code
Exam_Code	VARCHAR(4)	4	NOT NULL	Foreign key referencing examination.exam_Code
Med_Code	VARCHAR(5)	5	NOT NULL	Foreign key referencing medication.med_Code

Table: Applies_Treat

Field Name	Data Type	Data Length	Constraints	Description
Tcharacter_Code	VARCHAR(11)	11	PRIMARY KEY,NOT NULL, UNIQUE	Foreign key referencing treatment.character_Code
E_Pnum_Code	VARCHAR(9)	9	PRIMARY KEY,NOT NULL, UNIQUE	Foreign key referencing treatment.num_Code
Treat_Code	VARCHAR(4)	4	NOT NULL	Foreign key referencing treatment.exam_Code
TMed_Code	VARCHAR(5)	5	NOT NULL	Foreign key referencing medication.med_Code

2. STORE PROCEDURE / FUNCTION / SQL

1. Increase Inpatient Fee to 10% for all the current inpatients who are admitted to hospital from 01/09/2020

```
UPDATE inpatient
SET inpatient.Fee = inpatient.Fee * 1.1
WHERE inpatient.Date_of_admission >= '2020-09-01' AND
inpatient.Date_of_discharge IS NULL;
```

character_Code	num_Code	Date_Of_Admission	Date_Of_Discharge	Fee	Diagnosis	Sickroom	nurse_Code
IP000000006	000000006	2021-10-20	*NULL*	76.71	External constriction of unspecified shoulder, seq...	09	E009
IP000000007	000000007	2021-10-25	2023-12-09	56.51	Poisoning by digestants, intentional self-harm, in...	09	E008
IP000000008	000000008	2021-12-10	2023-08-31	68.60	Other congenital malformations of pulmonary valve	07	E006
IP000000009	000000009	2021-12-10	*NULL*	75.05	Unspecified Zone II fracture of sacrum, sequela	03	E010
IP000000010	000000010	2021-10-28	2023-07-14	95.63	Unsp physeal fracture of lower end of unspecified ...	06	E006

`inpatient` table before executing the code

character_Code	num_Code	Date_Of_Admission	Date_Of_Discharge	Fee	Diagnosis	Sickroom	nurse_Code
IP000000006	000000006	2021-10-20	*NULL*	84.38	External constriction of unspecified shoulder, seq...	09	E009
IP000000007	000000007	2021-10-25	2023-12-09	62.16	Poisoning by digestants, intentional self-harm, in...	09	E008
IP000000008	000000008	2021-12-10	2023-08-31	68.60	Other congenital malformations of pulmonary valve	07	E006
IP000000009	000000009	2021-12-10	*NULL*	82.56	Unspecified Zone II fracture of sacrum, sequela	03	E010
IP000000010	000000010	2021-10-28	2023-07-14	95.63	Unsp physeal fracture of lower end of unspecified ...	06	E006

`inpatient` table after executing the code

2. **Select all the patients (outpatient & inpatient) of the doctor named 'Nguyen Van A'**

```
SELECT
   p.num_Code,
   p.patient_FName,
   p.patient_LName,
   e.emp_Code
FROM
   employee e
JOIN doctor d ON
   e.emp_Code = d.emp_Code
JOIN examination ex ON
   e.emp_Code = ex.doctor_CodeE
JOIN patient p ON
   ex.num_Code = p.num_Code
WHERE
   emp_fname = 'Nguyen' AND emp_lname = 'Van A'
UNION
SELECT
   p.num_Code,
   p.patient_FName,
   p.patient_LName,
   e.emp_Code
FROM
   employee e
JOIN treatment t ON
   e.emp_Code = t.doctor_CodeT
JOIN patient p ON
   t.num_Code = p.num_Code
WHERE
   emp_fname = 'Nguyen' AND emp_lname = 'Van A';
```

a doctor's name to test is 'Jacky McCoid'

num_Code	patient_FName	patient_LName	emp_Code

The result is empty since there is no doctor named 'Nguyen Van A'

3. **Write a function to calculate the total medication price a patient has to pay for each treatment or examination**

```
CREATE VIEW applied_med AS
SELECT applies_exam.E_Pnum_Code,
   applies_exam.Exam_Code AS Treat_code,
   applies_exam.Med_Code,
   medication.med_Name,
   medication.med_Fee
FROM applies_exam
   JOIN medication ON applies_exam.Med_Code =
medication.med_Code
UNION
SELECT applies_treat.TNum_Code,
   applies_treat.Treat_Code,
   applies_treat.TMed_Code,
   medication.med_Name,
   medication.med_Fee
FROM applies_treat
   JOIN medication ON applies_treat.TMed_Code =
medication.med_Code;

DELIMITER //
CREATE PROCEDURE med_payment (IN PatientID varchar(9))
BEGIN
SELECT * FROM applied_med py WHERE py.E_Pnum_Code =
PatientID;
END //
DELIMITER ;
```

```
CALL med_payment('000000001');
```

Our call to the function

E_Pnum_Code	Treat_code	Med_Code	med_Name	Med_fee
000000001	EX01	MED09	White Mulberry	20.13

The result of the call

4. **Write a procedure to sort the doctor in increasing number of patients he/she takes care in a period of time**

```
DELIMITER //
 CREATE PROCEDURE DOCTOR_NOP (IN STARTD VARCHAR(10), IN
ENDD VARCHAR(10)) BEGIN
SELECT
   doctor_Code,
   emp_FName,
   emp_LName,
   COUNT(`doctor_Code`) as Numo_patient
FROM
   (
      SELECT
         ex.doctor_CodeE as doctor_Code,
         e.emp_FName,
         e.emp_LName,
         ex.num_code
      FROM
         employee e
         JOIN examination ex ON e.emp_Code = ex.doctor_CodeE
      WHERE
         (
            ex.exam_Date > startD
            AND ex.exam_Date < endD
         )
         OR (
            ex.exam_NDate > startD
            AND ex.exam_NDate < endD
         )
      UNION
      SELECT
         t.doctor_CodeT,
         e.emp_FName,
         e.emp_LName,
         t.num_code
      FROM
```

```
        employee e
        JOIN treatment t ON e.emp_Code = t.doctor_CodeT
      WHERE
        (t.treat_SDate > startD)
        AND (
          t.treat_EDate IS NULL
          OR t.treat_EDate < endD
        )
  ) AS doctor_workload
GROUP BY
  `doctor_Code`
ORDER BY
  Numo_patient;

END // DELIMITER ;
```

```
call DOCTOR_NOP('2019-01-01', '2023-01-01');
```

Our call to the function

	doctor_Code varchar	emp_FNar varchar	emp_LNar varchar	Numo_patient bigint
1	E005	Rodrique	Seals	1
2	E002	Wadsworth	Prazer	1
3	E003	Gerri	Farries	2
4	E004	Early	Phoebe	2
5	E001	Jacky	McCoid	4

Our result from the function

3. DATABASE MANAGEMENT

A. Use-case of indexing efficiency

In a database, indexing is a technique used to improve the speed of data retrieval operations on a database table. An index is a data structure that provides a fast and efficient way to look up records based on the values of specific columns. Without an index, the database management system would need to scan through every row in a table to find the requested data, which can become inefficient as the size of the table grows.

Indexing in MySQL Database management system can either be used with the 'INDEX' or 'KEY' keywords. Both of them have the same meanings as well as the same effect upon using (though 'KEY' might be the more user friendly keyword). In our scenarios, one use-case of indexing that is almost impossible to be replaced is in GROUP BY for the fee summing function.

```
      CREATE VIEW pay AS
      SELECT patient.num_Code,
          COALESCE(SUM(examination.exam_Fee), 0) AS exam_fee,
          COALESCE(inpatient.Fee, 0) AS treatm_fee,
          (
              COALESCE(SUM(examination.exam_Fee), 0) +
COALESCE(inpatient.Fee, 0) + COALESCE(SUM(applied_med.Med_fee), 0)
          ) AS Total_fee,
          COALESCE(SUM(applied_med.Med_fee), 0) AS medication_fee
      FROM patient
          LEFT JOIN examination ON patient.num_Code =
examination.num_Code
          LEFT JOIN inpatient ON patient.num_Code = inpatient.num_Code
          LEFT JOIN applied_med ON patient.num_Code =
applied_med.E_Pnum_Code
      GROUP BY patient.num_Code;
```

In this view named **pay**, the SUM function makes a very good use of GROUP BY by the indexed column **num_Code** of the **patient** table. Without the GROUP BY function on the **num_Code** column, we will have to add a lot more codes to take the sum of fee for each patient while the engine will have to do a lot of extra work to select the right patient then take the sum of all of their fee.

B. Solving one case of database security

Database Injection Attacks

To prevent Database Injection attacks that may occur by taking use of the input box to search for patients on screen **Search** or screen **Insert**, we used the Input Validation method directly in our PHP script. The method is implemented through a function called **validateInput**

```php
function validateInput($input)
{
    $input = trim($input); // Exterminate whitespace
    $input = stripslashes($input); // Remove backslashes
    $input = htmlspecialchars($input); // Convert special characters
    // Check if the input is not empty
    if (empty($input)) {
        return false;
    }
    return true;
}
```

1. On the first layer, we use the **trim** function to take out any whitespace at the beginning and the end of the input. This is not a type of injection attack, though trimming might help the other layers work better and so as the other lines in the rest of our code.
2. The second layer of validation is **stripslashes** function to remove any existing backslashes that may have been added to the input. This is particularly relevant in scenarios where the PHP directive "magic_quotes_gpc" is enabled. Magic quotes automatically add backslashes to data that comes from external sources (e.g., form submissions, GET, POST, and COOKIE data). The stripslashes function is employed here to ensure that the input does not contain unintended backslashes.
3. In the third layer, we use the **htmlspecialchars** function. This step is crucial for preventing cross-site scripting (XSS) attacks. By converting special characters to their HTML entities, we ensure that any potentially harmful code entered by the user is treated as plain text and not interpreted by the browser. This is particularly important when displaying user-generated content on a web page to avoid the execution of scripts or HTML tags that could compromise the security of the website and its users. To be specific, below is some examples of how it interacts with the particular input:
    a. & (ampersand): Converted to &
    b. " (double quote): Converted to "
    c. ' (single quote): Converted to '
    d. < (less than): Converted to <

    e. > (greater than): Converted to >
4. The last layer will be a layer to check the emptiness of the input. This layer must be added because an empty string may malfunction other functions later in the code.

By implementing these layers, we establish a robust input validation mechanism that enhances the resilience of the application against various types of input-related vulnerabilities.