DỰ ÁN QUẢN LÝ HỆ THỐNG TRUNG TÂM DẠY VÀ HỌC THÊM

NHÓM 2

NỘI DUNG

NHÓM 6

NỘI DUNG
CHÍNH

I. GIỚI THIỆU

II. MÔ TẢ NGHIỆP VỤ

III.MÔ HÌNH ER

IV. TỪ ĐIỂN DỮ LIỆU

V. CÀI ĐẶT

VI. KẾT LUẬN

GIỚI THIỆU

THỰC TRẠNG HIỆN NAY

Thông tin học viên phân tán, dễ sai lệch, thất lạc (điểm số, lịch học, học phí).

Xếp lịch, phân lớp rối rắm khi số lượng học viên tăng, dễ trùng giờ hoặc quá tải giáo viên.

Quản lý giáo viên thiếu đồng bộ, khó theo dõi chất lượng giảng dạy và phản hồi học viên.

Thu học phí, báo cáo tài chính thủ công → tốn thời gian, dễ sai sót, thiếu minh bạch.

Giao tiếp giáo viên - học viên không hiệu quả, dẫn tới thiếu thông tin.

Các phương pháp thủ công không đáp ứng khi trung tâm mở rộng quy mô.

VẬY NÊN ...

GIỚI THIỆU

THỰC TRẠNG HIỆN NAY

VẬY NÊN

Việc xây dựng một hệ thống quản lý tập trung và hiệu quả là vô cùng cần thiết.

GIỚI THIỆU

THỰC TRẠNG HIỆN NAY

Việc xây dựng một hệ thống quản lý tập trung và hiệu quả là vô cùng cần thiết.

Hệ thống EducateDB được ra đời từ đây, mong muốn đơn giản hoá quá trình quản lý của các trung tâm dạy học.

GIỚI THIỆU

GIẢI PHÁP

CẦN ĐẢM BẢO CÁC TIÊU CHÍ SAU

Xây dựng hệ thống cơ sở dữ liệu tích hợp để quản lý toàn diện

Quản lý đầy đủ thông tin học viên, giáo viên

Quản lý khóa học theo cấp độ: chương trình, tài liệu, điều kiện đầu vào.

Tự động hóa xếp lớp, phân công giảng dạy và theo dõi tiến độ học tập.

Ghi nhận điểm số, chuyên cần minh bạch, dễ tra cứu.

Giảm thao tác thủ công, hạn chế sai sót.

Quản lý học phí, báo cáo tài chính chính xác và kịp thời.

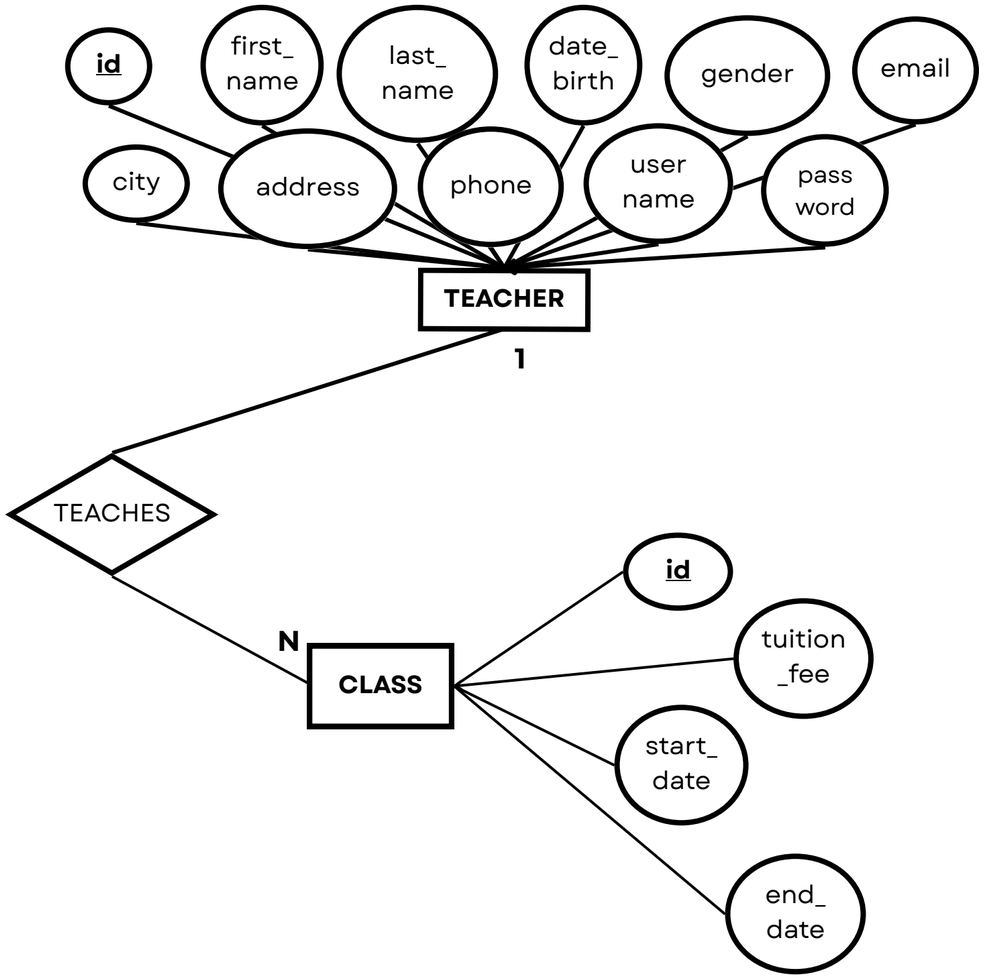
MÔ TẢ NGHIỆP VỤ

NHIỆM VỤ TỪNG BẢNG

STUDENT **TEACHER**	Có các thuộc tính như id, tên, tuổi, cho đến tài khoản cá nhân, để quản lý thông tin cá nhân của học sinh/ giáo viên giảng dạy.
COURSE **COURSE_MATERIAL**	Quản lý các khoá học, tài liệu giảng dạy
CLASS **CLASS_STUDENT**	Quản lý các lớp và học sinh của từng lớp
EXAM **EXAM_RESULT**	Quản lý thông tin về điểm số và các bài tập assignment/ kiểm tra.
PAYMENT	Quản lý về thông tin sổ sách/ học phí của từng học viên.

MÔ TẢ NGHIỆP VỤ

NHIỆM VỤ HỆ THỐNG



NGHIỆP VỤ QUẢN LÝ LỚP HỌC VÀ GIẢNG VIÊN

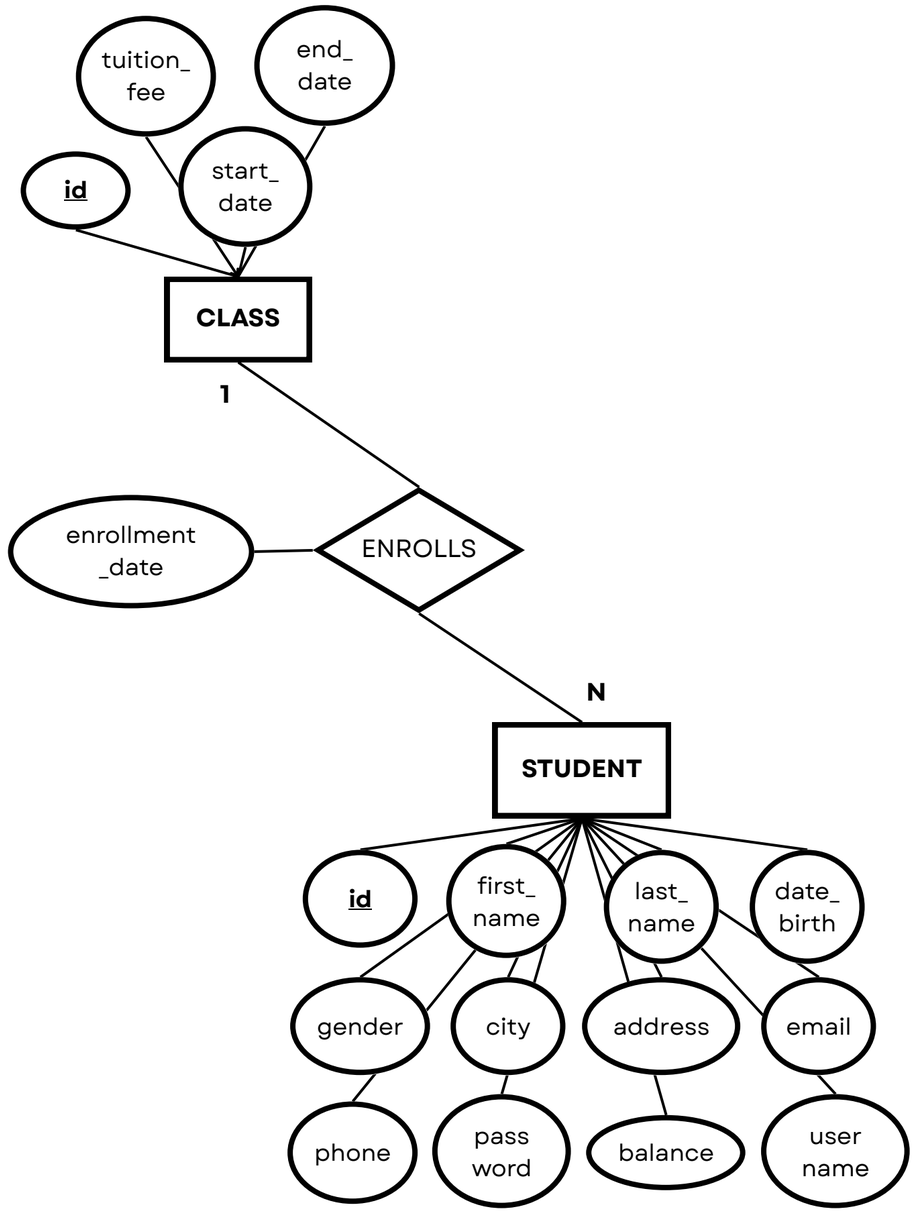
- Mỗi lớp học là một phiên bản của một khóa học, có lịch học, phòng học, ngày bắt đầu/kết thúc riêng và được phụ trách bởi một giáo viên.
- Một giáo viên có thể dạy nhiều lớp khác nhau.
- Bảng Teacher lưu trữ thông tin giáo viên.
- Bảng Class lưu trữ thông tin các lớp học. Cột course_id liên kết lớp học với một Course (quan hệ 1-Nhiều từ Course đến Class), và cột teacher_id liên kết lớp học với một Teacher (quan hệ 1-Nhiều từ Teacher đến Class).

MÔ TẢ NGHIỆP VỤ

NHIỆM VỤ HỆ THỐNG

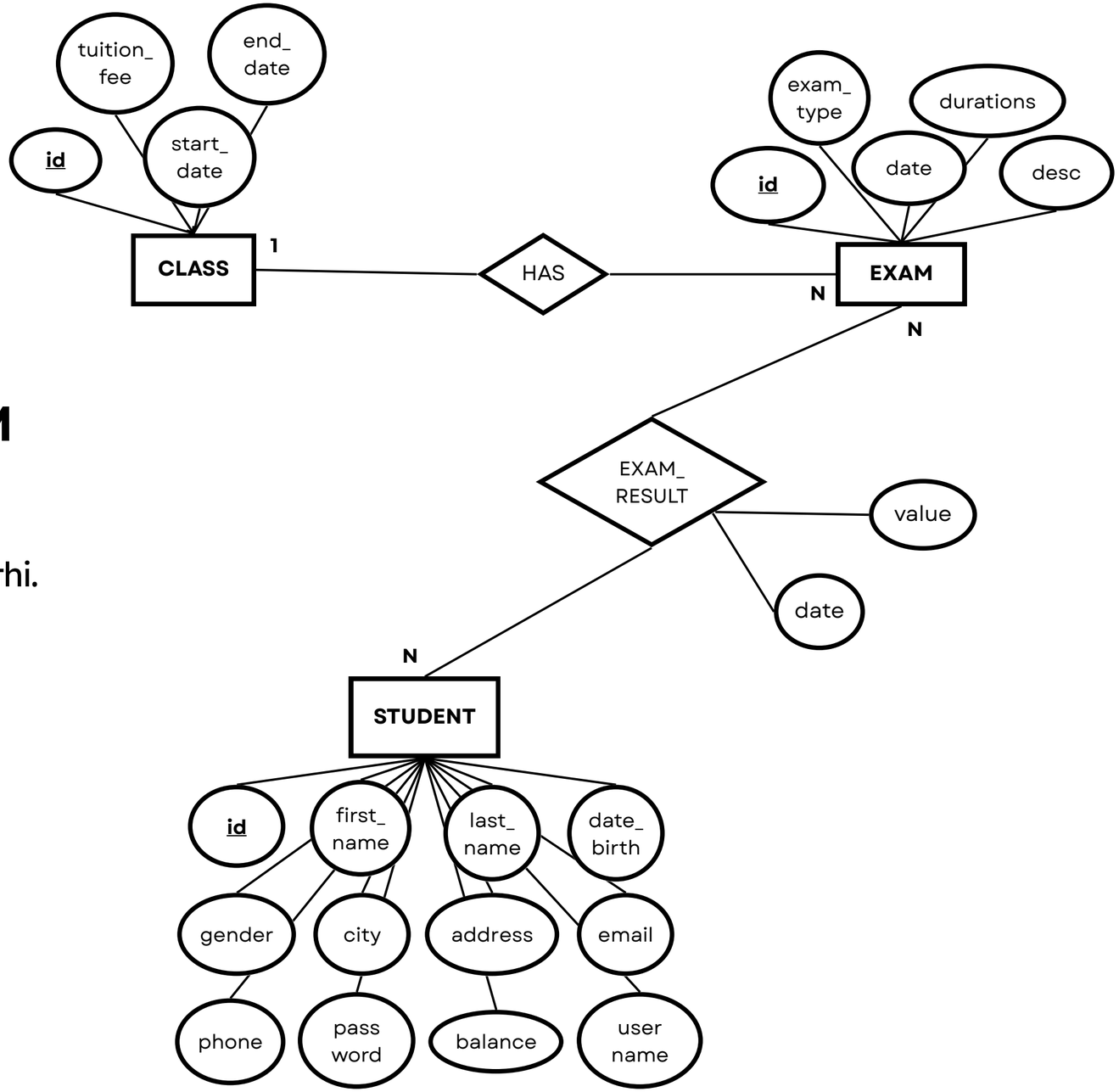
NGHIỆP VỤ QUẢN LÝ HỌC VIÊN VÀ GHI DANH

- Trung tâm quản lý thông tin của học viên.
- Học viên có thể đăng ký (ghi danh) vào nhiều lớp học khác nhau, và một lớp học cũng có nhiều học viên tham gia.
- Bảng Student lưu trữ thông tin học viên.
- Mối quan hệ Nhiều-Nhiều giữa Student và Class được giải quyết bằng bảng trung gian Class_Student. Mỗi dòng trong bảng này là một bản ghi ghi danh, cho biết học viên nào đã đăng ký vào lớp học nào.



MÔ TẢ NGHIỆP VỤ

NHIỆM VỤ HỆ THỐNG



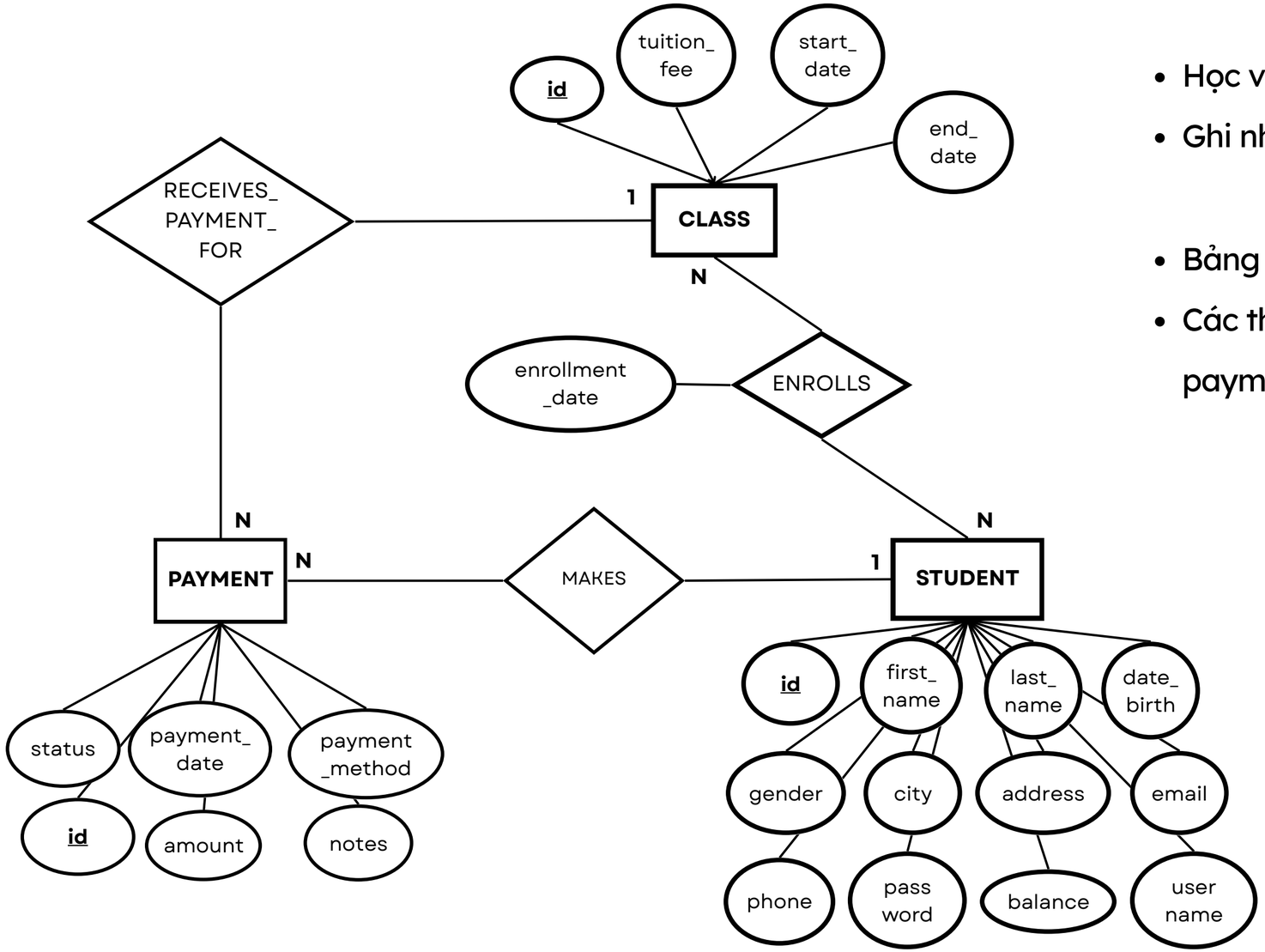
NGHIỆP VỤ THI CỬ VÀ CHẤM ĐIỂM

- Mỗi lớp học có nhiều kỳ thi: quiz, giữa kỳ, cuối kỳ.
- Học viên tham gia và được chấm điểm cho từng kỳ thi.

- Class (1-N) Exam
- Bảng Exam_Result lưu trữ điểm số, có mối quan hệ Nhiều - Nhiều với cả Student và Exam.

MÔ TẢ NGHIỆP VỤ

NHIỆM VỤ HỆ THỐNG



NGHIỆP VỤ THANH TOÁN HỌC PHÍ

- Học viên thanh toán học phí theo khóa học đã đăng ký.
- Ghi nhận số tiền, trạng thái, phương thức thanh toán.

- Bảng Payment liên kết Student và Course.
- Các thuộc tính: amount, status, payment_method, payment_date.

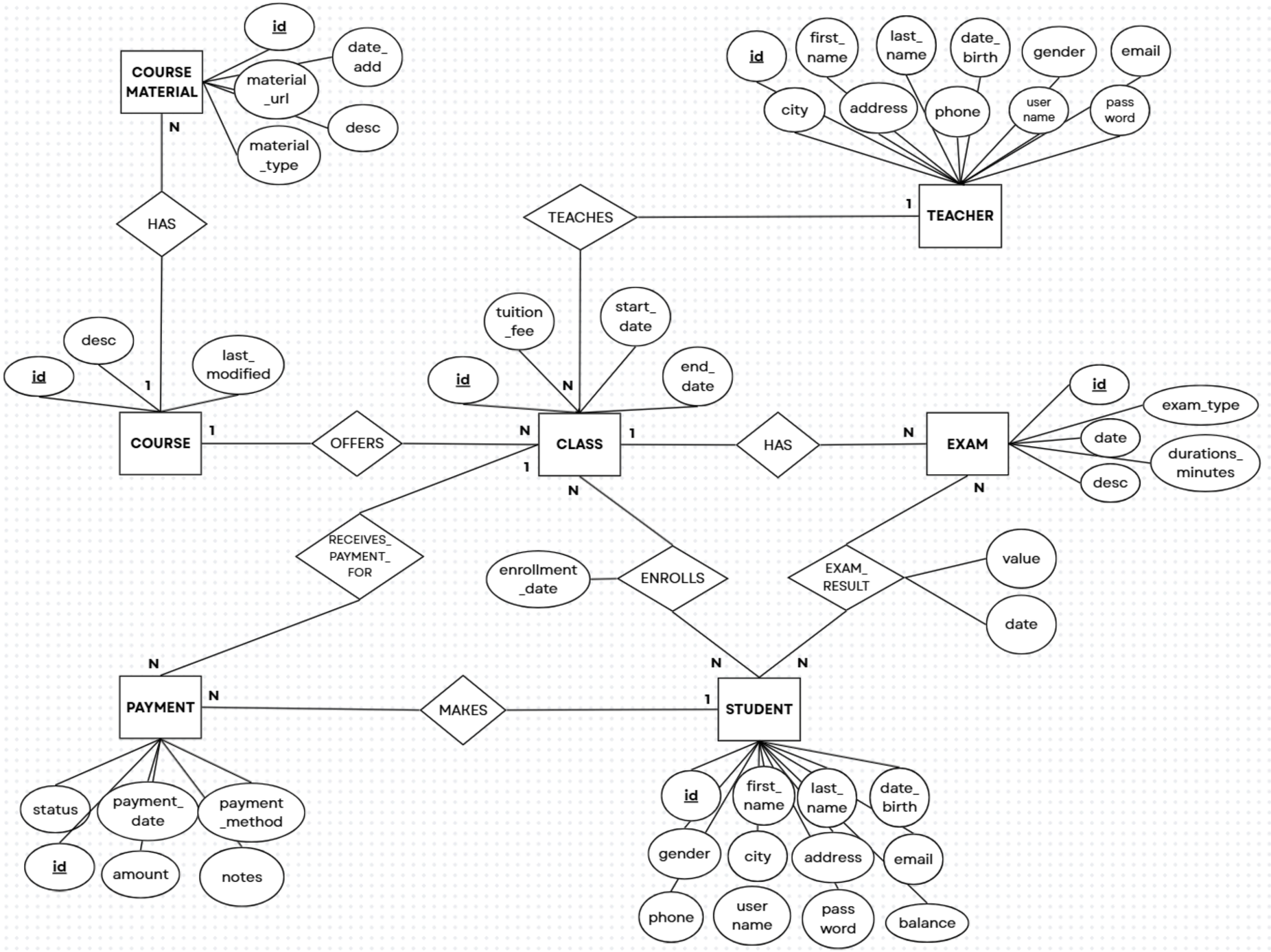
MÔ TẢ NGHIỆP VỤ

THUỘC TÍNH CÁC BẢNG

course_material	**id (PK), course_id (FK),** description, material_type, material_url, date_add
course	**id (PK)**, description, last_modified
class	**id (PK)**, start_date, end_date, **teacher_id (FK), course_id (FK)**, schedule_info, room_number, tuition_fee
student	**id (PK)**, first_name , last_name, date_birth, gender, email, phone, address, city, user_name, password,balance, created_date
class_student	**class_id (PK, FK), student_id (PK, FK),** enrollment_date
teacher	**id (PK)**, first_name, last_name, date_birth, gender, email, phone, address, city, description, user_name, password
payment	**id (PK),** payment_date, amount , status, **student_id (FK), class_id (FK)**, payment_method, notes
exam_result	value , **student_id (PK, FK), exam_id (PK, FK)** date
exam	**id (PK),** date, description, **class_id (FK),** exam_type, duration_minutes

MÔ TẢ NGHIỆP VỤ

MÔ HÌNH ER



TỪ ĐIỂN DỮ LIỆU
BẢNG TEACHER

TEACHER			
Column Name	Data Type	Check	Key/Index/Constraint
id	Varchar(5)	LIKE 'TE[0-9][0-9][0-9]'	PK, Not null
first_name	Nvarchar(50)		Not null
last_name	Nvarchar(50)		Not null
date_birth	Date		
gender	Nvarchar(3)	IN (N'Nam', N'Nữ')	
email	Varchar(100)	LIKE '%_@__%.__%'	Unique, Not null
phone	Varchar(20)		
address	Nvarchar(255)		
city	Nvarchar(50)		
description	Nvarchar(255)		
user_name	Varchar(50)		Unique, Not null
password	Varchar(255)		Not null

TỪ ĐIỂN DỮ LIỆU

BẢNG TEACHER

```sql
CREATE TABLE Teacher (
    id VARCHAR(5) PRIMARY KEY,
    first_name NVARCHAR(50) NOT NULL,
    last_name NVARCHAR(50) NOT NULL,
    date_birth DATE,
    gender NVARCHAR(3),
    email VARCHAR(100) UNIQUE NOT NULL,
    phone VARCHAR(20),
    address NVARCHAR(255),
    city NVARCHAR(50),
    description NVARCHAR(255),
    user_name VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    CONSTRAINT CK_Teacher_ID CHECK (id LIKE 'TE[0-9][0-9][0-9]'),
    CONSTRAINT CK_Teacher_Gender CHECK (gender IN (N'Nam', N'Nữ')),
    CONSTRAINT CK_Teacher_Email CHECK (email LIKE '%_@__%.__%')
);
```

TỪ ĐIỂN DỮ LIỆU
BẢNG STUDENT

STUDENT			
Column Name	Data Type	Check	Key/Index/Constraint
id	Varchar(5)	LIKE 'ST[0-9][0-9][0-9]'	PK, Not null
first_name	Nvarchar(50)		Not null
last_name	Nvarchar(50)		Not null
date_birth	Date		
gender	Nvarchar(3)	IN (N'Nam', N'Nữ')	
email	Varchar(100)	LIKE '%_@__%.__%'	Unique, Not null
phone	Varchar(20)		
address	Nvarchar(255)		
city	Nvarchar(50)		
user_name	Varchar(50)		Unique, Not null
password	Varchar(255)		Not null
balance	Decimal(12,2)		>= 0
created_date	Date		

TỪ ĐIỂN DỮ LIỆU

BẢNG STUDENT

```sql
CREATE TABLE Student (
    id VARCHAR(5) PRIMARY KEY,
    first_name NVARCHAR(50) NOT NULL,
    last_name NVARCHAR(50) NOT NULL,
    date_birth DATE,
    gender NVARCHAR(3),
    email VARCHAR(100) UNIQUE NOT NULL,
    phone VARCHAR(20),
    address NVARCHAR(255),
    city NVARCHAR(50),
    user_name VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    balance DECIMAL(12,2),
    created_date DATE,
    CONSTRAINT CK_Student_ID CHECK (id LIKE 'ST[0-9][0-9][0-9]'),
    CONSTRAINT CK_Student_Gender CHECK (gender IN (N'Nam', N'Nữ')),
    CONSTRAINT CK_Student_Email CHECK (email LIKE '%_@__%.__%'),
    CONSTRAINT CK_Student_Balance CHECK (balance >= 0)
);
```

TỪ ĐIỂN DỮ LIỆU
BẢNG COURSE

COURSE			
Column Name	Data Type	Check	Key/Index/Constraint
id	Nvarchar(50)		PK, Not null
description	Nvarchar(MAX)		
last_modified	Datetime2		

TỪ ĐIỂN DỮ LIỆU

BẢNG COURSE

```
CREATE TABLE Course (
    id NVARCHAR(50) PRIMARY KEY,
    description NVARCHAR(MAX),
    last_modified DATETIME2
);
```

TỪ ĐIỂN DỮ LIỆU
BẢNG COURSE_MATERIAL

COURSE_MATERIAL			
Column Name	Data Type	Check	Key/Index/Constraint
id	Varchar(5)	LIKE 'CM[0-9][0-9][0-9]'	PK, Not null
course_id	Nvarchar(50)		FK references Course(id), Not null
description	Nvarchar(MAX)		
material_type	Nvarchar(50)	IN (N'Sách giáo trình', N'Sách bài tập', N'Tệp âm thanh', N'Video Links', N'Học liệu', N'Sách luyện đề', N'Tài liệu tham khảo')	
material_url	Varchar(255)		
date_add	Date		

TỪ ĐIỂN DỮ LIỆU

BẢNG COURSE_MATERIAL

```
CREATE TABLE Course_Material (
    id VARCHAR(5) PRIMARY KEY,
    course_id NVARCHAR(50) NOT NULL,
    description NVARCHAR(MAX),
    material_type NVARCHAR(50),
    material_url VARCHAR(255),
    date_add DATE,
    FOREIGN KEY (course_id) REFERENCES Course(id),
    CONSTRAINT CK_Course_Material_ID CHECK (id LIKE 'CM[0-9][0-9][0-9]'),
    CONSTRAINT CK_Course_Material_Type CHECK (material_type IN (N'Sách giáo trình', N'Sách bài tập',
    N'Tệp âm thanh', N'Video Links', N'Học liệu', N'Sách luyện đề', N'Tài liệu tham khảo'))
);
```

TỪ ĐIỂN DỮ LIỆU
BẢNG CLASS

CLASS			
Column Name	Data Type	Check	Key/Index/Constraint
id	Nvarchar(20)		PK, Not null
start_date	Date		
end_date	Date	>= start_date	
teacher_id	Varchar(5)		FK references Teacher(id)
course_id	Nvarchar(50)		FK references Course(id), Not null
schedule_info	Nvarchar(100)		
room_number	Nvarchar(20)	LIKE 'P[1-3][0-9][0-9]'	
tuition_fee	Decimal(12, 2)	>=0	

TỪ ĐIỂN DỮ LIỆU

BẢNG CLASS

```sql
CREATE TABLE Class (
    id NVARCHAR(20) PRIMARY KEY,
    start_date DATE,
    end_date DATE,
    teacher_id VARCHAR(5),
    course_id NVARCHAR(50) NOT NULL,
    schedule_info NVARCHAR(100),
    room_number NVARCHAR(20),
    tuition_fee DECIMAL(12,2) NOT NULL,
    FOREIGN KEY (teacher_id) REFERENCES Teacher(id),
    FOREIGN KEY (course_id) REFERENCES Course(id),
    CONSTRAINT CK_Course_TuitionFee CHECK (tuition_fee >= 0),
    CONSTRAINT CK_Class_Dates CHECK (end_date >= start_date),
    CONSTRAINT CK_Class_RoomNumber CHECK (room_number LIKE 'P[1-3][0-9][0-9]')

);
```

TỪ ĐIỂN DỮ LIỆU

BẢNG CLASS_STUDENT

CLASS_STUDENT			
Column Name	Data Type	Check	Key/Index/Constraint
class_id	Nvarchar(20)		PK, FK references Class(id), Not null
student_id	Varchar(5)		PK, FK references Student(id), Not null
enrollment_date	Date		

TỪ ĐIỂN DỮ LIỆU

BẢNG CLASS_STUDENT

```sql
CREATE TABLE Class_Student (
    class_id NVARCHAR(20) NOT NULL,
    student_id VARCHAR(5) NOT NULL,
    enrollment_date DATE,
    PRIMARY KEY (class_id, student_id),
    FOREIGN KEY (class_id) REFERENCES Class(id),
    FOREIGN KEY (student_id) REFERENCES Student(id)
);
```

TỪ ĐIỂN DỮ LIỆU

BẢNG EXAM

EXAM			
Column Name	Data Type	Check	Key/Index/Constraint
id	Varchar(5)	LIKE 'EX[0-9][0-9][0-9]'	PK, Not null
date	Date		
description	Nvarchar(MAX)		
class_id	Nvarchar(20)		FK references Class(id), Not null
exam_type	Nvarchar(50)	IN (N'Midterm', N'Final', N'Quiz', N'Mock Test', N'Speaking Test')	
duration_minutes	Int	> 0	

TỪ ĐIỂN DỮ LIỆU

BẢNG EXAM

```
CREATE TABLE Exam (
    id VARCHAR(5) PRIMARY KEY,
    date DATE,
    description NVARCHAR(MAX),
    class_id NVARCHAR(20) NOT NULL,
    exam_type NVARCHAR(50),
    duration_minutes INT,
    FOREIGN KEY (class_id) REFERENCES Class(id),
    CONSTRAINT CK_Exam_ID CHECK (id LIKE 'EX[0-9][0-9][0-9]'),
    CONSTRAINT CK_Exam_Type CHECK (exam_type IN (N'Midterm', N'Final',
    N'Quiz', N'Mock Test', N'Speaking Test')),
    CONSTRAINT CK_Exam_Duration CHECK (duration_minutes > 0)
);
```

TỪ ĐIỂN DỮ LIỆU

BẢNG GRADE

EXAM_RESULT			
Column Name	Data Type	Check	Key/Index/Constraint
value	Decimal(4,2)		>= 0.0 AND <= 10.0, PK, Not null
student_id	Varchar(5)		PK, FK references Student(id), Not null
exam_id	Varchar(5)		PK, FK references Exam(id), Not null
date	Date		

TỪ ĐIỂN DỮ LIỆU

BẢNG GRADE

```sql
CREATE TABLE Exam_Result (
    student_id VARCHAR(5) NOT NULL,
    exam_id VARCHAR(5) NOT NULL,
    value DECIMAL(4,2) NOT NULL,
    date DATE,
    PRIMARY KEY (student_id, exam_id),
    FOREIGN KEY (student_id) REFERENCES Student(id),
    FOREIGN KEY (exam_id) REFERENCES Exam(id),
    CONSTRAINT CK_Grade_Value CHECK (value >= 0.00 AND value <= 10.00)
);
```

TỪ ĐIỂN DỮ LIỆU
BẢNG PAYMENT

PAYMENT			
Column Name	Data Type	Check	Key/Index/Constraint
id	Varchar(5)	LIKE 'PA[0-9][0-9][0-9]'	PK, Not null
payment_date	Date		
amount	Decimal(12,2)		> 0, Not null
status	Nvarchar(20)	IN (N'Success', N'Failed')	Not null
student_id	Varchar(5)		FK references Student(id), Not null
class_id	Nvarchar(50)		FK references Class(id), Not null
payment_method	Nvarchar(50)	IN (N'Tiền mặt', N'Chuyển khoản', N'Thẻ tín dụng', N'Momo')	
notes	Nvarchar(255)		

TỪ ĐIỂN DỮ LIỆU

BẢNG PAYMENT

```
CREATE TABLE Payment (
    id VARCHAR(5) PRIMARY KEY,
    payment_date DATE,
    amount DECIMAL(12,2) NOT NULL,
    status NVARCHAR(20) NOT NULL,
    student_id VARCHAR(5) NOT NULL,
    class_id NVARCHAR(20) NOT NULL,
    payment_method NVARCHAR(50),
    notes NVARCHAR(255),
    FOREIGN KEY (student_id) REFERENCES Student(id),
    FOREIGN KEY (class_id) REFERENCES Class(id),
    CONSTRAINT CK_Payment_ID CHECK (id LIKE 'PA[0-9][0-9][0-9]'),
    CONSTRAINT CK_Payment_Amount CHECK (amount > 0),
    CONSTRAINT CK_Payment_Status CHECK (status IN (N'Success', N'Failed')),
    CONSTRAINT CK_Payment_Method CHECK (payment_method IN
    (N'Tiền mặt', N'Chuyển khoản', N'Thẻ tín dụng', N'Momo'))
);
```

CÀI ĐẶT VẬT LÝ
VIEWS

V_CLASS_DETAILS

- Mục đích: Cung cấp một cái nhìn tổng quan, chi tiết và dễ đọc về thông tin của từng lớp học.
- Nghiệp vụ thực tế: Giúp nhân viên tư vấn, quản lý học vụ hoặc giáo viên nhanh chóng tra cứu thông tin của một lớp học mà không cần phải viết các câu lệnh JOIN phức tạp giữa các bảng Class, Course, và Teacher.

```
CREATE VIEW V_Class_Details AS
SELECT
    cl.id AS ClassID,
    cl.schedule_info AS Schedule,
    cl.room_number AS Room,
    co.id AS CourseID,
    co.description AS CourseDescription,
    t.last_name + N' ' + t.first_name AS TeacherFullName
FROM Class cl
JOIN Course co ON cl.course_id = co.id
LEFT JOIN Teacher t ON cl.teacher_id = t.id;
```

CÀI ĐẶT VẬT LÝ
VIEWS

```sql
CREATE VIEW V_Student_Grades AS
SELECT
    s.id AS StudentID,
    s.last_name + N' ' + s.first_name AS StudentFullName,
    e.description AS ExamDescription,
    e.exam_type AS ExamType,
    e.date AS ExamDate,
    g.value AS GradeValue
FROM Exam_Result g
JOIN Student s ON g.student_id = s.id
JOIN Exam e ON g.exam_id = e.id;
```

V_STUDENT_GRADES

- Mục đích: Tạo một báo cáo điểm số chi tiết của học viên, kết hợp thông tin từ học viên, kỳ thi và điểm số.

- Nghiệp vụ thực tế: Ta truy vấn View này để có ngay một danh sách rõ ràng bao gồm tên học viên, tên kỳ thi, loại kỳ thi, ngày thi và điểm số thay vì phải JOIN 3 bảng Grade, Student, Exam.

CÀI ĐẶT VẬT LÝ
VIEWS

```sql
CREATE VIEW V_Course_Summary AS
SELECT
    c.id AS CourseID,
    c.description AS CourseDescription,
    MIN(cl.tuition_fee) AS MinTuitionFee,
    MAX(cl.tuition_fee) AS MaxTuitionFee,
    AVG(cl.tuition_fee) AS AvgTuitionFee,
    COUNT(DISTINCT cl.id) AS NumberOfClasses,
    COUNT(DISTINCT cs.student_id) AS TotalEnrollments
FROM Course c
LEFT JOIN Class cl ON c.id = cl.course_id
LEFT JOIN Class_Student cs ON cl.id = cs.class_id
GROUP BY c.id, c.description;
```

V_COURSE_SUMMARY

- Mục đích: Cung cấp một báo cáo tóm tắt về tình hình của từng khóa học.

- Nghiệp vụ thực tế: Giúp ta quản lý trung tâm đánh giá hiệu quả và mức độ phổ biến của các khóa học.

CÀI ĐẶT VẬT LÝ
VIEWS

V_TEACHER_WORKLOAD

- Mục đích: Tổng hợp và hiển thị khối lượng công việc của từng giáo viên.

- Nghiệp vụ thực tế: Ta có thể nhanh chóng biết được giáo viên nào đang dạy nhiều lớp, giáo viên nào còn trống lịch, hoặc giáo viên dạy nhiều khóa học khác nhau để lên kế hoạch phân công cho các kỳ học tiếp theo.

```sql
CREATE VIEW V_Teacher_Workload AS
SELECT
    t.id AS TeacherID,
    t.last_name + N' ' + t.first_name AS TeacherFullName,
    ISNULL(COUNT(DISTINCT cl.id), 0) AS AssignedClasses,
    ISNULL(COUNT(DISTINCT cl.course_id), 0) AS DistinctCoursesTaught
FROM Teacher t
LEFT JOIN Class cl ON t.id = cl.teacher_id
GROUP BY t.id, t.first_name, t.last_name;
```

CÀI ĐẶT VẬT LÝ
FUNCTIONS

FN_GETSTUDENTFULLNAME

-Mục đích: Nhận vào StudentID và trả về tên đầy đủ của sinh viên theo định dạng "Họ Tên" (ví dụ: "Nguyễn Văn An").

-Nghiệp vụ thực tế: Thay vì phải viết lại logic last_name + N' ' + first_name ở mọi nơi, chúng ta có thể gọi hàm này.

```
CREATE FUNCTION fn_GetStudentFullName (@StudentID VARCHAR(5))
RETURNS NVARCHAR(101)
AS
BEGIN
    DECLARE @FullName NVARCHAR(101);
    SELECT @FullName = last_name + N' ' + first_name
    FROM Student
    WHERE id = @StudentID;
    RETURN @FullName;
END
```

CÀI ĐẶT VẬT LÝ
FUNCTIONS

FN_CALCULATESTUDENTAGE

-Mục đích: Tính tuổi hiện tại của một người dựa vào ngày sinh của họ.

-Nghiệp vụ thực tế: Hữu ích khi cần tạo các báo cáo thống kê nhân khẩu học về học viên hoặc giáo viên khi ta tính tuổi

```
CREATE FUNCTION fn_CalculateStudentAge (@DateOfBirth DATE)
RETURNS INT
AS
BEGIN
    RETURN DATEDIFF(YEAR, @DateOfBirth, GETDATE()) -
           CASE WHEN (MONTH(@DateOfBirth) > MONTH(GETDATE()))
           OR (MONTH(@DateOfBirth) = MONTH(GETDATE()) AND
           DAY(@DateOfBirth) > DAY(GETDATE())) THEN 1 ELSE 0 END;
END
GO
```

CÀI ĐẶT VẬT LÝ
FUNCTIONS

FN_GETCLASSESBYTEACHER

-Mục đích: Lấy danh sách chi tiết các lớp học mà một giáo viên cụ thể đang dạy.

-Nghiệp vụ thực tế: Giáo viên có thể muốn xem danh sách lớp của mình và JOIN thêm với bảng Exam để xem các kỳ thi sắp tới của những lớp đó trong cùng một câu truy vấn.

```
CREATE FUNCTION dbo.fn_GetClassesByTeacher (@TeacherID VARCHAR(5))
RETURNS TABLE
AS
RETURN
(
    SELECT
        cl.id AS ClassID,
        cl.schedule_info AS Schedule,
        co.description AS CourseDescription
    FROM Class cl
    JOIN Course co ON cl.course_id = co.id
    WHERE cl.teacher_id = @TeacherID
);
```

CÀI ĐẶT VẬT LÝ
PROCEDURES

```
CREATE PROCEDURE usp_GetStudentEnrollments @StudentID VARCHAR(5)
AS
BEGIN
    SET NOCOUNT ON;
    SELECT ClassID, Schedule, CourseDescription, TeacherFullName
    FROM V_Class_Details
    WHERE ClassID IN (
        SELECT class_id FROM Class_Student
        WHERE student_id = @StudentID);
END
GO
```

USP_GETSTUDENTENROLLMENTS

-Mục đích: Lấy danh sách tất cả các lớp học mà một sinh viên cụ thể đã ghi danh.

-Nghiệp vụ thực tế: Thay vì phải viết lại một câu lệnh JOIN phức tạp mỗi lần, ứng dụng chỉ cần gọi thủ tục này với StudentID sẽ trả về một bảng ghi chứa thông tin chi tiết về các lớp học của sinh viên đó.

CÀI ĐẶT VẬT LÝ
PROCEDURES

USP_UPDATESTUDENTBALANCE

-Mục đích: Cung cấp một cách an toàn để nạp tiền vào tài khoản của sinh viên.

-Nghiệp vụ thực tế: Khi sinh viên nộp tiền mặt hoặc chuyển khoản để nạp vào tài khoản (không phải thanh toán học phí trực tiếp), nhân viên sẽ sử dụng chức năng này.

```
CREATE PROCEDURE usp_UpdateStudentBalance
    @StudentID VARCHAR(5),
    @AmountToAdd DECIMAL(12,2)
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @NewBalance DECIMAL(12,2);
    IF NOT EXISTS (SELECT 1 FROM Student WHERE id = @StudentID)
    BEGIN
        PRINT N'Lỗi: Không tìm thấy sinh viên với ID ' + @StudentID;
        RETURN;
    END
    UPDATE Student SET balance = ISNULL(balance, 0) + @AmountToAdd WHERE id = @StudentID;
    SELECT @NewBalance = balance FROM Student WHERE id = @StudentID;
    PRINT N'Đã cập nhật số dư cho sinh viên ' + @StudentID + N'. Số dư mới: ' + CAST(@NewBalance AS VARCHAR);
END
GO
```

CÀI ĐẶT VẬT LÝ
PROCEDURES

USP_PROCESSCOURSEPAYMENT

-Mục đích: Đóng gói toàn bộ logic nghiệp vụ phức tạp của việc thanh toán học phí cho một khóa học.

-Nghiệp vụ thực tế: Kiểm tra đủ tiền, học phí có khớp không, sau đó mới trừ tiền và ghi nhận giao dịch thành công. Nếu không, nó sẽ ghi nhận là giao dịch thất bại.

```
CREATE PROCEDURE usp_ProcessCoursePayment
    @PaymentID VARCHAR(5), @StudentID VARCHAR(5), @CourseID NVARCHAR(50),
    @PaymentAmount DECIMAL(12,2), @PaymentMethod NVARCHAR(50) = NULL, @TransactionNotes NVARCHAR(255) = NULL
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @StudentBalance DECIMAL(12,2), @CourseTuition DECIMAL(12,2), @PaymentStatus NVARCHAR(20);
    DECLARE @CurrentPaymentDate DATE = GETDATE();
    SELECT @StudentBalance = ISNULL(balance, 0) FROM Student WHERE id = @StudentID;
    SELECT @CourseTuition = tuition_fee FROM Course WHERE id = @CourseID;
    IF @StudentBalance IS NULL OR @CourseTuition IS NULL
    BEGIN
        SET @PaymentStatus = N'Failed';
        SET @TransactionNotes = ISNULL(@TransactionNotes + N'; ', N'') + N'Lỗi: Sinh viên hoặc Khóa học không hợp lệ.';
        INSERT INTO Payment (id, payment_date, amount, status, student_id, course_id, payment_method, notes)
        VALUES (@PaymentID, @CurrentPaymentDate, @PaymentAmount,
        @PaymentStatus, @StudentID, @CourseID, @PaymentMethod, @TransactionNotes);
        PRINT N'Thanh toán thất bại: Sinh viên hoặc Khóa học không hợp lệ.'; RETURN;
    END
```

TRG_UPDATECOURSELASTMODIFIED

-Bảng kích hoạt: Course_Material

-Mục đích: Tự động cập nhật trường last_modified của một khóa học mỗi khi có bất kỳ thông tin bị thay đổi.

-Nghiệp vụ thực tế: Giúp hệ thống luôn có dấu vết về lần cập nhật gần nhất liên quan đến một khóa học.

```
CREATE TRIGGER trg_UpdateCourseLastModified
ON Course_Material
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;
    UPDATE Course SET last_modified = GETDATE()
    WHERE id IN (SELECT course_id FROM inserted);
END
```

CÀI ĐẶT VẬT LÝ
TRIGGERS

TRG_LOGSTUDENTCREATION,
TRG_LOGSTUDENTUPDATE,
TRG_LOGSTUDENTDELETION

-Bảng kích hoạt: Student

-Mục đích: Tự động ghi lại các hành động quan trọng (tạo mới, cập nhật, xóa) đối với dữ liệu sinh viên vào một bảng AuditLog riêng biệt.

-Nghiệp vụ thực tế: Đây là một cơ chế kiểm toán (auditing) cơ bản. Nó giúp quản trị viên hệ thống theo dõi đầu vào/ đầu ra thông tin của học sinh.

```
CREATE TRIGGER trg_LogStudentCreation ON Student
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;
    INSERT INTO AuditLog (TableName, RecordID, ActionType, ChangeDetails)
    SELECT
        'Student',
        i.id,
        'INSERT',
        'A new student was created: ' + i.last_name + N' ' + i.first_name
    FROM inserted i;
END
```

CÀI ĐẶT VẬT LÝ
MỘT VÀI BÀI TOÁN

Tìm các lớp học có cùng giáo viên

```sql
SELECT
    t.last_name + N' ' + t.first_name AS TeacherFullName,
    c1.id AS ClassID_1,
    c2.id AS ClassID_2
FROM Class c1
JOIN Class c2 ON c1.teacher_id = c2.teacher_id AND c1.id < c2.id
JOIN Teacher t ON c1.teacher_id = t.id
WHERE c1.teacher_id IS NOT NULL
```

	TeacherFullName	ClassID_1	ClassID_2
1	Trần Quốc Tuấn	AWPRO-2401	AWPRO-2402
2	Trần Quốc Tuấn	AWPRO-2401	AWPRO-2403
3	Trần Quốc Tuấn	AWPRO-2402	AWPRO-2403
4	Miller Jessica	BEADV-2401	BEADV-2402
5	Baker John	BECOM-2401	BECOM-2403
6	Vũ Thị Kim Ngân	GEA1-2401	GEA1-2401B
7	Vũ Thị Kim Ngân	GEA1-2401	GEA1-2402
8	Vũ Thị Kim Ngân	GEA1-2401B	GEA1-2402

CÀI ĐẶT VẬT LÝ
MỘT VÀI BÀI TOÁN

Báo cáo điểm trung bình, điểm cao nhất, thấp nhất của học viên trong một khóa học

```
WITH StudentGradesInCourse AS (
    SELECT
        cs.student_id,
        er.value
    FROM Class_Student cs
    JOIN Class cl ON cs.class_id = cl.id
    JOIN Exam e ON cl.id = e.class_id
    JOIN Exam_Result er ON e.id = er.exam_id AND cs.student_id = er.student_id
    WHERE cl.course_id = N'IELTS_70'
)
SELECT
    s.last_name + N' ' + s.first_name AS StudentFullName,
    AVG(sg.value) AS AverageScore,
    MAX(sg.value) AS HighestScore,
    MIN(sg.value) AS LowestScore
FROM StudentGradesInCourse sg
JOIN Student s ON sg.student_id = s.id
GROUP BY s.id, s.first_name, s.last_name
ORDER BY AverageScore DESC
```

	StudentFullName	AverageScore	HighestScore	LowestScore
1	Trần Diệu Linh	7.250000	7.50	7.00
2	Nguyễn Văn An	6.750000	7.00	6.50
3	Đặng Bá Sơn	6.750000	7.00	6.50
4	Vũ Huyền Trang	6.250000	6.50	6.00

CÀI ĐẶT VẬT LÝ
MỘT VÀI BÀI TOÁN

Top 2 lớp học có điểm trung bình của học viên cao nhất.
Top 2 lớp học có điểm trung bình của học viên thấp nhất.

```
WITH ClassAverages AS (
    SELECT
        cl.id AS ClassID,
        co.description AS CourseDescription,
        t.last_name + N' ' + t.first_name AS TeacherFullName,
        AVG(er.value) AS AverageGrade
    FROM Exam_Result er
    JOIN Exam e ON er.exam_id = e.id
    JOIN Class cl ON e.class_id = cl.id
    JOIN Course co ON cl.course_id = co.id
    LEFT JOIN Teacher t ON cl.teacher_id = t.id
    GROUP BY cl.id, co.description, t.last_name, t.first_name),
Highest AS (SELECT TOP 2 WITH TIES * FROM ClassAverages ORDER BY AverageGrade DESC),
Lowest AS (SELECT TOP 2 WITH TIES * FROM ClassAverages ORDER BY AverageGrade ASC)
SELECT * FROM Highest
UNION ALL
SELECT * FROM Lowest
ORDER BY AverageGrade DESC
```

	ClassID	CourseDescription	TeacherFullName	AverageGrade
1	IELTSPRO-2401	IELTS Chuyên sâu (Speaking & Writing)	Nguyễn Bình An	9.250000
2	KIDSADV-2401	Tiếng Anh Trẻ Em Nâng cao (11-14 tuổi)	White Emily	9.250000
3	TOEICB-2401	Luyện thi TOEIC Cơ bản Mục tiêu 500+	Lê Thu Phương	6.000000
4	IELTS55-2403	Luyện thi IELTS Mục tiêu 5.0-5.5	Hoàng Thủy Linh	5.500000

ANY QUESTIONS?

THANK YOU FOR LISTENING