

Kịch bản hệ thống quản lý sinh viên

Hệ thống quản lý sinh viên của trường đại học được thiết kế để quản lý thông tin về **khoa**, **ngành**, **lớp học hành chính**, **môn học**, **lớp học phần**, **giảng viên** và **sinh viên**. Dưới đây là mô tả chi tiết các thực thể và mối quan hệ giữa chúng:

1 Khoa (Department)

- Mỗi **khoa** là đơn vị quản lý hành chính cao nhất trong trường.
- Mỗi khoa có nhiều **ngành đào tạo** và nhiều **giảng viên** thuộc về khoa đó.

Quan hệ:

- 1–N giữa Department → Major**: Một khoa có nhiều ngành; mỗi ngành thuộc đúng một khoa.
`Major.dept_id → Department.dept_id`
- 1–N giữa Department → Lecturer**: Một khoa có nhiều giảng viên; mỗi giảng viên chỉ thuộc một khoa.
`Lecturer.dept_id → Department.dept_id`
- 1–N giữa Department → Course**: Một khoa có thể mở nhiều môn học khác nhau.
`Course.dept_id → Department.dept_id`

2 Ngành (Major)

- Mỗi **ngành học** thuộc về một **khoa**, và trong ngành sẽ có nhiều **lớp học hành chính (ClassGroup)**.
- Ngành cũng là đơn vị phân chia sinh viên về chuyên môn đào tạo.

Quan hệ:

- 1–N giữa Major → ClassGroup**: Một ngành có nhiều lớp hành chính.
`ClassGroup.major_id → Major.major_id`

3 Giảng viên (Lecturer)

- Mỗi giảng viên thuộc về một khoa, có thể giảng dạy nhiều lớp học phần khác nhau.

- Ngoài ra, một giảng viên có thể được phân công làm **cố vấn học tập** cho các lớp hành chính.

Quan hệ:

- **1–N** giữa **Lecturer** → **ClassGroup (advisor)**: Một giảng viên có thể làm cố vấn cho nhiều lớp, nhưng mỗi lớp chỉ có tối đa một cố vấn.
`ClassGroup.advisor_lecturer_id` → `Lecturer.lecturer_id` (*FK có thể NULL nếu chưa có cố vấn*)
- **1–N** giữa **Lecturer** → **Section**: Một giảng viên có thể phụ trách nhiều lớp học phần (môn học cụ thể trong học kỳ).
`Section.lecturer_id` → `Lecturer.lecturer_id`

4 Lớp hành chính (ClassGroup)

- Là nhóm sinh viên thuộc cùng một ngành, học cùng niên khóa và có một giảng viên làm cố vấn học tập.
- Mỗi lớp hành chính có nhiều sinh viên.

Quan hệ:

- **1–N** giữa **ClassGroup** → **Student**: Mỗi lớp hành chính có nhiều sinh viên, và mỗi sinh viên chỉ thuộc một lớp.
`Student.class_id` → `ClassGroup.class_id`

5 Sinh viên (Student)

- Mỗi sinh viên thuộc về một lớp hành chính, và có thể đăng ký nhiều lớp học phần khác nhau trong các học kỳ.

Quan hệ:

- **N–N** giữa **Student** ↔ **Section** (qua bảng trung gian Enrollment):
 Một sinh viên có thể đăng ký nhiều lớp học phần, và một lớp học phần có nhiều sinh viên.
`Enrollment(section_id, student_id)`
 Có ràng buộc **UNIQUE**(`section_id`, `student_id`) để tránh đăng ký trùng.

6 Môn học (Course)

- Là các môn đào tạo thuộc về một khoa.
- Mỗi môn có thể có các môn tiên quyết (phải học trước).

Quan hệ:

- **1-N giữa Course → Section:** Một môn có thể mở nhiều lớp học phần ở các học kỳ khác nhau.
Section.course_id → Course.course_id
- **N-N tự tham chiếu trong Course ↔ Course (Prerequisite):**
Một môn có thể yêu cầu nhiều môn tiên quyết, và đồng thời có thể là tiên quyết của nhiều môn khác.
Prerequisite(course_id, required_course_id)
Cả hai cột FK cùng tham chiếu đến Course(course_id).

7 Học kỳ (Semester)

- Mỗi học kỳ được định danh theo năm học và thứ tự học kỳ (ví dụ: “HK1 - 2025”).
- Mỗi học kỳ có thể mở nhiều lớp học phần.

Quan hệ:

- **1-N giữa Semester → Section:** Một học kỳ có nhiều lớp học phần.
Section.semester_id → Semester.sem_id

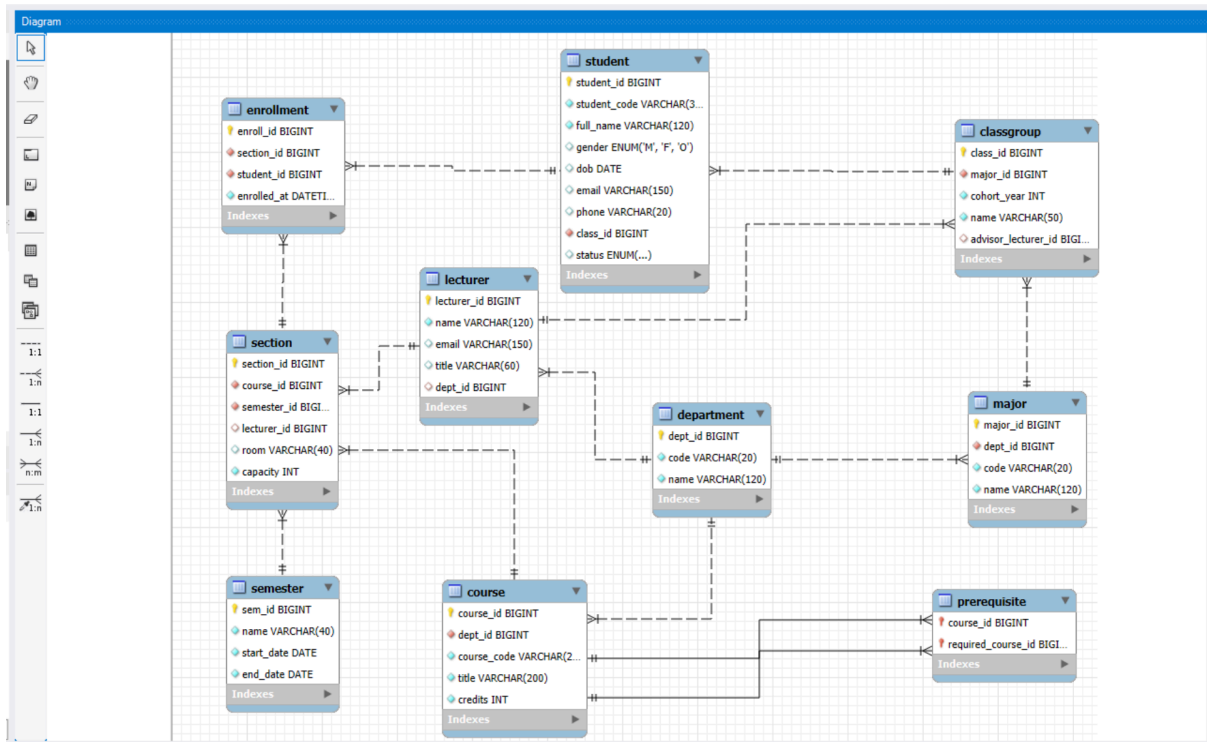
8 Lớp học phần (Section)

- Mỗi lớp học phần là một phiên bản giảng dạy của một môn học trong một học kỳ cụ thể, có giảng viên phụ trách.
- Mỗi lớp học phần sẽ có nhiều sinh viên đăng ký.

Quan hệ:

- Liên kết với **Course**, **Semester**, **Lecturer**, và **Enrollment** như mô tả ở trên.

Dữ liệu các bảng trong cơ sở dữ liệu



1. Bảng department

	dept_id	code	name
▶	1	CNTT	Công nghệ Thông tin
	2	DTVT	Điện tử Viễn thông
✱	NULL	NULL	NULL

2. Bảng major

	major_id	dept_id	code	name
▶	1	1	KTPM	Kỹ thuật phần mềm
	2	1	HTTT	Hệ thống thông tin
	3	2	DT	Điện tử
	4	2	VT	Viễn thông
✱	NULL	NULL	NULL	NULL

3. Bảng lecturer

	lecturer_id	name	email	title	dept_id
▶	1	Nguyễn Văn A	a@ptit.edu.vn	ThS	1
	2	Trần Thị B	b@ptit.edu.vn	TS	1
	3	Phạm Văn C	c@ptit.edu.vn	ThS	2
	4	Lê Thị D	d@ptit.edu.vn	TS	2
	5	Hoàng Minh E	e@ptit.edu.vn	ThS	1
•	NULL	NULL	NULL	NULL	NULL

4. Bảng classgroup

	class_id	major_id	cohort_year	name	advisor_lecturer_id
▶	1	1	2023	D23-085	1
	2	2	2023	D23-101	2
	3	1	2023	D23-102	5
	4	2	2023	D23-201	2
	5	3	2023	VT23-01	3
	6	4	2023	DT23-01	4
•	NULL	NULL	NULL	NULL	NULL

5. Bảng section

	section_id	course_id	semester_id	lecturer_id	room	capacity
▶	1	1	1	1	A101	80
	2	2	1	2	A202	60
	3	3	1	5	B201	70
	4	4	1	1	C105	60
	5	5	1	3	A303	60
	6	6	2	4	A204	60
	7	1	2	1	A101	80
	8	2	2	2	A202	60
•	NULL	NULL	NULL	NULL	NULL	NULL

6. Bảng semester

	sem_id	name	start_date	end_date
	1	HK1 2025-2026	2025-09-01	2026-01-10
	2	HK2 2025-2026	2026-02-15	2026-06-30
▶*	NULL	NULL		NULL

7. Bảng student

	student_id	student_code	full_name	gender	dob	email	phone	class_id	status
▶	1	B23DCCN894	Vũ Anh Tuấn	M	2006-05-20	tuan@ptit.edu.vn	0901234567	1	ACTIVE
	2	B23DCCN777	Trung	M	2006-07-10	trung@ptit.edu.vn	0902345678	1	ACTIVE
	3	B23DCCN895	Nguyễn Minh Anh	F	2006-03-12	anh@ptit.edu.vn	0903456789	1	ACTIVE
	4	B23DCCN678	Phạm Gia Huy	M	2006-08-21	huy@ptit.edu.vn	0904567890	2	ACTIVE
	5	B23DVT001	Trần Bảo Long	M	2006-11-02	long@ptit.edu.vn	0911111111	3	ACTIVE
	6	B23DVT002	Lê Khánh Linh	F	2006-01-28	linh@ptit.edu.vn	0912222222	4	ACTIVE
	7	B23DCCN679	Đỗ Hải Yến	F	2006-06-05	yen@ptit.edu.vn	0905678901	2	ACTIVE
	8	B23DVT003	Vũ Đức Mạnh	M	2006-09-17	manh@ptit.edu.vn	0913333333	3	ACTIVE
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

8. Bảng course

	course_id	dept_id	course_code	title	credits
▶	1	1	CT101	Nhập môn Lập trình	3
	2	1	CT201	Cấu trúc dữ liệu	3
	3	1	CT301	Cơ sở dữ liệu	3
	4	1	CT302	Lập trình Web	3
	5	2	VT101	Mạch điện	3
	6	2	VT201	Thông tin số	3
*	NULL	NULL	NULL	NULL	NULL

9. Bảng enrollment

Result Grid					Filter Rows:	Edit:
	enroll_id	section_id	student_id	enrolled_at		
▶	1	1	1	2025-10-23 01:10:41		
	2	1	2	2025-10-23 01:10:41		
	3	3	1	2025-10-24 10:14:50		
	4	3	2	2025-10-24 10:14:50		
	5	3	3	2025-10-24 10:14:50		
	6	4	1	2025-10-24 10:14:50		
	7	4	5	2025-10-24 10:14:50		
	8	4	6	2025-10-24 10:14:50		
	9	5	3	2025-10-24 10:14:50		
	10	5	4	2025-10-24 10:14:50		
	11	5	8	2025-10-24 10:14:50		
	12	6	3	2025-10-24 10:14:50		
	13	6	4	2025-10-24 10:14:50		
	14	6	8	2025-10-24 10:14:50		
	15	7	1	2025-10-24 10:14:50		
	16	7	2	2025-10-24 10:14:50		
	17	7	5	2025-10-24 10:14:50		
	18	8	1	2025-10-24 10:14:50		
	19	8	2	2025-10-24 10:14:50		
	20	8	5	2025-10-24 10:14:50		
★	NULL	NULL	NULL	NULL		





10. Bảng prerequisite

Result Grid			Filter Rows:
	course_id	required_course_id	
▶	2	1	
	4	1	
	3	2	
	6	5	
★	NULL	NULL	

Một số câu lệnh SQL cơ bản

1.1 Danh sách ngành theo khoa

```
1 • use quanlysv;  
2 • SELECT m.major_id, m.code AS major_code, m.name AS major_name,  
3         d.code AS dept_code  
4 FROM Major m  
5 JOIN Department d ON d.dept_id = m.dept_id  
6 WHERE d.code = 'CNTT'  
7 ORDER BY m.code;
```

Result Grid   Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 				
	major_id	major_code	major_name	dept_code
▶	2	HTTT	Hệ thống thông tin	CNTT
	1	KTPM	Kỹ thuật phần mềm	CNTT

1.2 SV kèm lớp/ ngành/ khoa, sắp xếp theo mã SV


```

1 • SELECT s.student_code, s.full_name,
2         cg.name AS class_name, m.code AS major_code, d.code AS dept_code
3 FROM Student s
4 JOIN ClassGroup cg ON s.class_id = cg.class_id
5 JOIN Major m      ON cg.major_id = m.major_id
6 JOIN Department d ON m.dept_id   = d.dept_id
7 WHERE s.status = 'ACTIVE'          -- WHERE
8 ORDER BY d.code, m.code, cg.name, s.student_code; -- ORDER BY

```

student_code	full_name	class_name	major_code	dept_code
B23DCCN678	Phạm Gia Huy	D23-101	HTTT	CNTT
B23DCCN679	Đỗ Hải Yến	D23-101	HTTT	CNTT
B23DTV002	Lê Khánh Linh	D23-201	HTTT	CNTT
B23DCCN777	Trung	D23-085	KTPM	CNTT
B23DCCN894	Vũ Anh Tuấn	D23-085	KTPM	CNTT
B23DCCN895	Nguyễn Minh Anh	D23-085	KTPM	CNTT
B23DTV001	Trần Bảo Long	D23-102	KTPM	CNTT
B23DTV003	Vũ Đức Mạnh	D23-102	KTPM	CNTT

1.3 Tìm môn có tên chứa “Cấu trúc” hoặc “Cơ sở”

```

1 • SELECT course_code, title
2 FROM Course
3 WHERE title LIKE '%Cấu trúc%' OR title LIKE '%Cơ sở'; -- LIKE + OR

```





course_code	title
CT201	Cấu trúc dữ liệu
CT301	Cơ sở dữ liệu

1.4 Thống kê số lượng theo lớp, chỉ hiển thị lớp có >= 2 SV

```

1 • SELECT cg.name AS class_name, COUNT(*) AS student_count
2 FROM Student s
3 JOIN ClassGroup cg ON cg.class_id = s.class_id
4 GROUP BY cg.class_id, cg.name
5 HAVING COUNT(*) >= 2 -- HAVING
6 ORDER BY student_count DESC;

```

Result Grid |   Filter Rows: | Export:  | Wrap Cell Content: 

class_name	student_count
D23-085	3
D23-102	2
D23-101	2

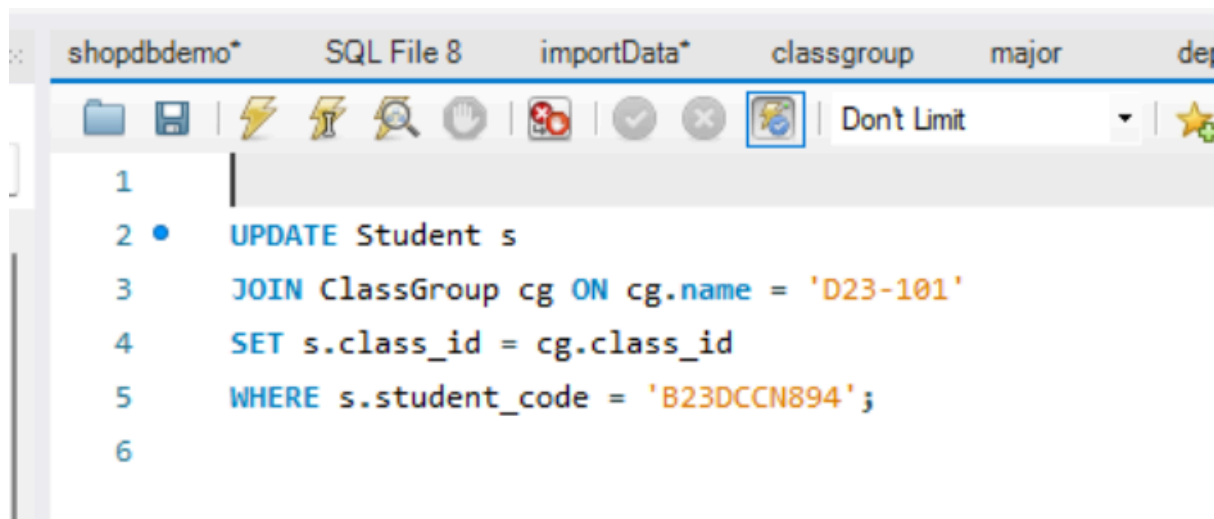
1.5 Thêm một giảng viên vào khoa CNTT

```

1 • INSERT INTO Lecturer(name, email, title, dept_id)
2 SELECT 'Ngô Thị H', 'h@ptit.edu.vn', 'ThS', dept_id
3 FROM Department WHERE code='CNTT';

```

1.6 Chuyển lớp hành chính cho một SV

A screenshot of a SQL IDE window. The title bar shows several tabs: 'shopdbdemo*', 'SQL File 8', 'importData*', 'classgroup', 'major', and 'de'. Below the title bar is a toolbar with various icons for file operations, execution, and settings. The main text area contains an SQL query with line numbers 1 through 6 on the left. The query is:

```
1
2 • UPDATE Student s
3   JOIN ClassGroup cg ON cg.name = 'D23-101'
4   SET s.class_id = cg.class_id
5   WHERE s.student_code = 'B23DCCN894';
6
```

Transaction trong SQL

- TRANSACTION trong SQL là tiến trình thực hiện một nhóm các câu lệnh SQL. Các câu lệnh này được thực thi một cách tuần tự và độc lập. Một Transaction được thực hiện thành công khi tất cả câu lệnh đều thành công, khi đó tất cả các thay đổi dữ liệu được thực hiện trong Transaction được lưu vào cơ sở dữ liệu. Tuy nhiên, nếu chỉ một trong số đó thất bại thì toàn bộ tiến trình sẽ thất bại, đồng nghĩa với việc dữ liệu phải rollback về trạng thái ban đầu (dữ liệu được khôi phục về trạng thái trước khi thực hiện Transaction).
- Điều này làm ta liên tưởng đến Git vậy. Khi ta thực hiện đến câu lệnh nào mà ta muốn lưu lại thay đổi ở đó ta sẽ tạo ra các điểm lưu trữ tại đó, nó giống với những commit trong Git cũng là lưu những bản sao mà lưu lại những thay đổi tại thời điểm đó mà ta muốn, mà không hề ảnh hưởng đến bản gốc. Nhưng ở Transaction thì lại khác là nếu bất kỳ một lệnh nào trong nhóm lệnh bị sai hay lỗi thì tất cả sẽ quay về điểm lưu trữ ban đầu trước khi mà ta thực hiện Transaction. Một điểm nữa giống nhau giữa 2 thằng này là nếu muốn quay trở lại điểm lưu trữ nào ta chỉ việc chạy câu lệnh quay lại.
- Trong SQL, có các lệnh sau được sử dụng để điều khiển Transaction:
 - + COMMIT: để lưu các thay đổi.
 - + ROLLBACK: để quay trở lại trạng thái trước khi có thay đổi.
 - + SAVEPOINT: tạo các điểm (point) bên trong các nhóm Transaction để ROLLBACK, tức là để quay trở lại điểm trạng thái đó.
 - + SET TRANSACTION: đặt một tên cho một Transaction.

Các lệnh điều khiển Transaction chỉ được sử dụng với các lệnh thao tác dữ liệu như INSERT, UPDATE và DELETE. Tuy nhiên chúng không thể được sử dụng trong lệnh CREATE TABLE hoặc DROP TABLE vì các hoạt động này được tự động xác định trong cơ sở dữ liệu.

Transaction để đăng ký lớp học phần cho sinh viên

```
START TRANSACTION;

-- Bước 1: Kiểm tra xem sinh viên có đủ điều kiện đăng ký các lớp học phần không
-- (Ví dụ: kiểm tra môn học đã đủ sinh viên chưa, sinh viên đã học môn tiên quyết chưa, v.v.)
-- (Bước này có thể thực hiện nhiều truy vấn kiểm tra dữ liệu)

-- Bước 2: Thêm sinh viên vào bảng Enrollment cho các lớp học phần
INSERT INTO Enrollment (section_id, student_id)
VALUES
    (101, 1001), -- Sinh viên 1001 đăng ký lớp học phần 101
    (102, 1001); -- Sinh viên 1001 đăng ký lớp học phần 102

-- Bước 3: Cập nhật thông tin sinh viên (nếu cần)
-- Ví dụ, cập nhật trạng thái đăng ký của sinh viên

COMMIT;
```

Transaction để thêm giảng viên vào khoa và phân công giảng dạy

```
START TRANSACTION;

-- Bước 1: Thêm giảng viên mới vào bảng Lecturer
INSERT INTO Lecturer (lecturer_id, name, dept_id)
VALUES (101, 'Nguyễn Văn A', 1); -- Thêm giảng viên thuộc khoa 1

-- Bước 2: Phân công giảng viên vào lớp học phần (Section)
INSERT INTO Section (section_id, course_id, semester_id, lecturer_id)
VALUES (201, 101, 1, 101); -- Giảng viên 101 giảng dạy lớp học phần 201 trong học kỳ 1

-- Bước 3: Kiểm tra xem lớp học phần đã được phân công đầy đủ chưa
SELECT * FROM Section WHERE section_id = 201;

-- Nếu không có lỗi, commit thay đổi
COMMIT;
```

Trigger trong SQL

- Trigger là một đoạn mã SQL được thiết lập để tự động chạy khi một sự kiện cụ thể xảy ra trong một bảng hoặc view trong cơ sở dữ liệu. Sự kiện này có thể là một hành động như INSERT, UPDATE, hoặc DELETE. Khi hành động đó xảy ra, trigger sẽ tự động được kích hoạt để thực hiện một chuỗi các hành động đã được định sẵn.
- Trigger được sử dụng khi nào?
 - Trigger thường được sử dụng để kiểm tra ràng buộc (check constraints) trên nhiều quan hệ (nhiều bảng/table) hoặc trên nhiều dòng (nhiều record) của bảng.
 - Ngoài ra việc sử dụng Trigger để chương trình có những hàm chạy ngầm nhằm phục vụ nhưng trường hợp hữu hạn và thường không sử dụng cho mục đích kinh doanh hoặc giao dịch.
 - Ngăn chặn việc xóa những dữ liệu quan trọng. (có thể dùng back up các dữ liệu quan trọng sang table khác phòng khi ...bị xóa ngoài ý muốn).

Mỗi table thường sẽ có 3 thao tác làm thay đổi dữ liệu đó là: UPDATE, INSERT, DELETE. Và đôi khi mỗi hành động như vậy ta sẽ có những ràng buộc trên bảng để giúp bảo toàn dữ liệu, lúc này sử dụng trigger là một giải pháp tốt.

Ví dụ bạn thiết kế cho bảng **product** và **category**, trong đó **product** sẽ có một column tên là **total_product** dùng để lưu trữ tổng số sản phẩm của category đó. Khi thêm một product thì ta phải tăng column đó lên một đơn vị. Khi update phải kiểm tra có thay đổi category không để tăng hoặc giảm cho hợp lý, khi delete thì bớt đi một. Việc này hoàn toàn có thể code bằng các ngôn ngữ đang sử dụng SQL Server, tuy nhiên bạn có thể sử dụng trigger để giúp hệ thống dữ liệu hoạt động tốt hơn.

Trigger khi thêm sinh viên mới vào lớp hành chính

```

1  DELIMITER $$
2
3  ● CREATE TRIGGER update_class_group_count
4  AFTER INSERT ON Student
5  FOR EACH ROW
6  BEGIN
7      -- Cập nhật số lượng sinh viên trong lớp hành chính
8      UPDATE ClassGroup
9      SET student_count = student_count + 1
10     WHERE class_id = NEW.class_id;
11 END$$
12
13 DELIMITER ;
14

```

GRANT, REVOKE trong SQL

- Lệnh GRANT trong SQL

- + Lệnh GRANT trong SQL được sử dụng để cấp quyền truy cập hoặc các đặc quyền với đối tượng cơ sở dữ liệu cho người dùng
- + Cú pháp lệnh GRANT trong SQL

Cú pháp lệnh GRANT trong SQL có dạng như sau:

GRANT privilege_name

ON object_name

TO {user_name |PUBLIC |role_name}

[WITH GRANT OPTION];

Trong đó:

- **privilege_name** là quyền truy cập hoặc đặc quyền được cấp cho người dùng. Một số quyền truy cập bao gồm ALL, EXECUTE, và SELECT.
- **object_name** là tên của đối tượng cơ sở dữ liệu như TABLE, VIEW, STORED PROC và

SEQUENCE.

- **user_name** là tên của người dùng được cấp quyền truy cập.
- **PUBLIC** được sử dụng để cấp quyền truy cập cho tất cả người dùng.
- **ROLES** là tập hợp các đặc quyền được nhóm lại với nhau.
- **WITH GRANT OPTION** cho phép người dùng cấp quyền truy cập cho người dùng khác

- **Lệnh REVOKE trong SQL**

- Lệnh REVOKE trong SQL được sử dụng để thu hồi quyền truy cập của người dùng hoặc các đặc quyền với các đối tượng cơ sở dữ liệu.

- Cú pháp lệnh REVOKE trong SQL

Cú pháp lệnh REVOKE trong SQL có dạng như sau:

REVOKE privilege_name

ON object_name

FROM {user_name |PUBLIC |role_name}

Ví dụ lệnh REVOKE trong SQL

Lệnh: REVOKE SELECT ON employee FROM user1.