Trong một Table SQl xét duyệt từng hàng từ trên xuống

+ muốn reset toàn bộ index khóa chính của table

DBCC CHECKIDENT ('tableName', RESEED, 0);

+ muốn tự điền index khóa chính mà không để tự động database điền

SET IDENTITY\_INSERT Courses ON;

+ muốn Database tự động điền index khóa chính

SET IDENTITY\_INSERT Courses off;

+ Create database my\_Database; (tạo )

+ Use my\_Database ; (sử dụng)

+ Drop database my\_Database ; (xóa toàn bộ database đó)

+ DROP TABLE ten\_bang; ; (xóa toàn bộ Table đó)

+ RENAME TABLE information to employeesInfor; (đổi tên bảng lại)

+ ALTER TABLE ten\_bang

RENAME COLUMN oldcolumnname TO newcolumnname;

* Đổi tên column của Table

+ ALTER TABLE employeesInfor

ADD BirthDate DATE NOT NULL;

* Thêm 1 cột vào Table

+ALTER TABLE tableName DROP COLUMN columnName ;

* Xóa 1 cột trong Table (cột đó là cột bình thường còn muốn xóa cột có khóa chính hay khóa ngoại thì phải khóa khóa chính hay khóa ngoại trước rồi mới xóa cột đó )

+ ALTER TABLE TenBang

DROP PRIMARY KEY;

* Xóa khóa chính (PRIMARY KEY);

+ ALTER TABLE tableName DROP FOREIGN KEY <Tên\_constraint>; // tên constraint của cột = > xóa khóa ngoại (FOREIGN KEY)

Muốn xem <tên\_Constraint> : SHOW CREATE TABLE tableName ;

A screenshot of a computer

AI-generated content may be incorrect.

VD : CREATE TABLE `enrollments` (

`EnrollmentID` int NOT NULL,

`StudentID` int DEFAULT NULL,

`CourseID` int DEFAULT NULL,

PRIMARY KEY (`EnrollmentID`), -- Khóa chính

KEY `StudentID` (`StudentID`), -- Index cho StudentID khóa ngoại

KEY `CourseID` (`CourseID`), -- Index cho CourseID khóa ngoại

CONSTRAINT `enrollments\_ibfk\_1` FOREIGN KEY (`StudentID`) REFERENCES `students` (`StudentID`), -- Khóa ngoại 1

CONSTRAINT `enrollments\_ibfk\_2` FOREIGN KEY (`CourseID`) REFERENCES `courses` (`CourseID`) -- Khóa ngoại 2

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

+ INSERT INTO nameTable VALUES (val1, val2 , val3, val4);

* Thêm giá trị vào Table

+ ALTER dùng để thay đổi structure của table or database or **huộc tính / trạng thái** của database .

+ **ALTER DATABASE** my\_Database **READ ONLY = 1** nghĩa là:  
👉 Chuyển database my\_Database sang chế độ **chỉ đọc** → chỉ được **xem dữ liệu (SELECT)**, không được **thêm, sửa, xóa (INSERT, UPDATE, DELETE**

+ **CHECK constraint : lọc dữ liệu của cột khi bỏ vào database**

**Check + cột**

**//Thêm column**

**ALTER TABLE employeesinfor**

**ADD Email NVARCHAR(50),**

**CHECK (CHAR\_LENGTH(Email) > 10); // CHAR\_LENGTH lấy độ dài của nvarchar**

**// thêm table**

**CREATE TABLE TableName (**

**Column1 INT,**

**Column2 VARCHAR(50),**

**Column3 INT,**

**CHECK (Column3 > 0) -- Column3 phải lớn hơn 0**

**);**

**A white board with writing on it

AI-generated content may be incorrect.**

Cấu trúc Table ; A screenshot of a computer

AI-generated content may be incorrect.

**CONSTRAINT** = **điều kiện/qui tắc áp dụng cho dữ liệu khi nhập vào bảng**.

DataType : Int , VARCHAR trong MySQL = kiểu **string (chuỗi ký tự) varchar (100) chứa 100 kí tự ,**

**DECIMAL(5,2)**

**Tổng = 5 chữ số**

**Trong đó 2 chữ số sau dấu phẩy**

**Vậy phần nguyên tối đa = 3 chữ số trước dấu phẩy**

**👉 Giá trị hợp lệ:**

**999.99 ✅**

**0.01 ✅**

**-123.45 ✅**

**DATE là kiểu dữ liệu ngày tháng trong MySQL.**

**INSERT INTO SinhVien VALUES (1, 'Nam', '2003-05-20');**

Khóa chính của bảng A có thể trở thành khóa ngoại trong bảng B → từ đó kết nối 2 bảng lại với nhau và khi truy vấn có thể đi từ bảng A đến bảng B và ngược lại quan hệ hai chiều

Trong 1 Table chỉ có duy nhất 1 khóa chính (PRIMARY KEY) và có thể có nhiều khóa ngoại (FOREIGN KEY)

 **Giá trị của cột khóa chính trong bẳng không được trùng nhau**.

 **Giá trị của cột khóa chính trong bảng không được để NULL**.

 **Giá trị của cột khóa ngoại trong bảng có thể được trùng nhau**.

 **Giá trị của cột khóa ngoại trong bảng có thể được để NULL**.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Khi nào sử dụng dấu : Khi bạn liệt kê nhiều thứ trong SQL **cùng loại** thì **dùng dấu phẩy giữa các phần tử**A screenshot of a computer

AI-generated content may be incorrect.

Giải thích **UNIQUE là gì?**

* Là quy tắc trong MySQL dùng để **ngăn trùng lặp dữ liệu** trong một cột hoặc nhiều cột.
* INSERT INTO Users (ID, Email) VALUES (1, 'a@gmail.com');
* INSERT INTO Users (ID, Email) VALUES (2, 'b@gmail.com');
* ✅ Thành công (Email khác nhau).
* Thử thêm email trùng
* INSERT INTO Users (ID, Email) VALUES (3, 'a@gmail.com');
* ❌ Lỗi! Vì a@gmail.com đã tồn tại, mà cột Email có UNIQUE.

Thứ tự viết (cú pháp, phải tuân theo SQL trong Database)

SELECT [cột1, cột2, ...] -- Chọn các cột muốn hiển thị

FROM [tên bảng] -- Bảng dữ liệu

[JOIN ...] -- (Nếu có JOIN với bảng khác)

[WHERE điều kiện] -- Lọc bản ghi

[GROUP BY cột] -- Gom nhóm (nếu cần)

[HAVING điều kiện nhóm] -- Lọc nhóm (nếu có GROUP BY)

[ORDER BY cột ASC|DESC] -- Sắp xếp kết quả

[LIMIT số\_lượng] -- Giới hạn số bản ghi trả về

Thứ tự viết (cú pháp, phải tuân theo trong LINQ trong IDE )

from item in Collection // FROM

join x in OtherCollection on ... // JOIN

where item.Price > 1000 // WHERE

group item by item.Category into g // GROUP BY

where g.Count() > 2 // HAVING

orderby g.Key descending // ORDER BY

select g // SELECT

------------------------------------------------------------

Câu lệnh truy vấn cơ bản

SET SQL\_SAFE\_UPDATES = 0; dùng để tắt chế độ safe Update Mode

Select + cột (lấy dữ liệu của cột trong table , chưa lọc dữ liệu)

FROM = bảng dữ liệu bạn muốn truy vấn , Không có FROM, SQL không biết lấy dữ liệu từ đâu.

WHERE : **cột + điều kiện** (dùng để lọc dữ liệu )

UPDATE NameTable [lấy colum của Table để Update giá trị ]

SET nameColumn1 = value , nameColumn2 = value

Kết hợp vs where

DELETE FROM nameTable [xóa dữ liệu trong bảng.]

Kết hợp vs where

SET autocommit = 0; -- tắt tự động commit

|  |  |
| --- | --- |
| AUTOCOMMIT 1 | Khi **AUTOCOMMIT = ON (1)** → mỗi câu lệnh **INSERT, UPDATE, DELETE,.....** sẽ **tự động lưu vào cơ sở dữ liệu** ngay lập tức. |
| AUTOCOMMIT 0 | Các thay đổi chưa lưu, phải dùng COMMIT hoặc ROLLBACK |
| COMMIT | Xác nhận những thay đổi và lưu những thay đổi và được ghi vĩnh viễn vào DB. |
| ROLLBACK | Hủy bỏ các thay đổi chưa commit |

A screenshot of a computer

AI-generated content may be incorrect.

+ AUTO\_INCREMENT = để MySQL **tự động đánh số thứ tự tăng dần** cho cột, thường dùng cho PRIMARY KEY , có thể dùng cho những cột khác

TH1 : dùng cho cột khóa chính

A screenshot of a computer

AI-generated content may be incorrect.

TH1 : dùng cho cột khác

A screenshot of a computer

AI-generated content may be incorrect.

JOIN trong MySQL

Bài mẫu 1

**1️⃣ INNER JOIN**

Lấy danh sách **nhân viên cùng thông tin điểm danh**, chỉ những nhân viên có điểm danh mới xuất hiện.

SELECT e.EmployeeID, e.Name, e.Position, a.WorkDate, a.CheckIn, a.CheckOut

FROM employeesinfor e // [bảng có cột khóa chính ]

INNER JOIN attendance a ON e.EmployeeID = a.EmployeeID; // bảng có cột khóa ngoại

**Ví dụ dữ liệu giả lập:**

**employeesinfor**

| **EmployeeID** | **Name** | **Position** | **Salary** |
| --- | --- | --- | --- |
| 1 | Alice | Manager | 1500.00 |
| 2 | Bob | Staff | 1200.00 |
| 3 | Charlie | Staff | 1200.00 |

**attendance**

| **AttendanceID** | **EmployeeID** | **WorkDate** | **CheckIn** | **CheckOut** |
| --- | --- | --- | --- | --- |
| 1 | 1 | 2025-09-12 | 08:00:00 | 17:00:00 |
| 2 | 2 | 2025-09-12 | 08:30:00 | 17:30:00 |

**Kết quả INNER JOIN:**

| **EmployeeID** | **Name** | **Position** | **WorkDate** | **CheckIn** | **CheckOut** |
| --- | --- | --- | --- | --- | --- |
| 1 | Alice | Manager | 2025-09-12 | 08:00:00 | 17:00:00 |
| 2 | Bob | Staff | 2025-09-12 | 08:30:00 | 17:30:00 |

Charlie không xuất hiện vì chưa có điểm danh.

**2️⃣ LEFT JOIN**

Lấy **tất cả nhân viên**, nếu chưa có điểm danh thì thông tin điểm danh là NULL.

SELECT e.EmployeeID, e.Name, e.Position, a.WorkDate, a.CheckIn, a.CheckOut

FROM employeesinfor e

LEFT JOIN attendance a ON e.EmployeeID = a.EmployeeID;

**Kết quả LEFT JOIN:**

| **EmployeeID** | **Name** | **Position** | **WorkDate** | **CheckIn** | **CheckOut** |
| --- | --- | --- | --- | --- | --- |
| 1 | Alice | Manager | 2025-09-12 | 08:00:00 | 17:00:00 |
| 2 | Bob | Staff | 2025-09-12 | 08:30:00 | 17:30:00 |
| 3 | Charlie | Staff | NULL | NULL | NULL |

Charlie xuất hiện nhưng chưa có điểm danh, nên các cột WorkDate, CheckIn, CheckOut là NULL.

**3️⃣ RIGHT JOIN**

Lấy **tất cả các bản ghi điểm danh**, nếu có EmployeeID không khớp trong bảng employeesinfor thì thông tin nhân viên là NULL.

SELECT e.EmployeeID, e.Name, e.Position, a.WorkDate, a.CheckIn, a.CheckOut

FROM employeesinfor e

RIGHT JOIN attendance a ON e.EmployeeID = a.EmployeeID;

**Kết quả RIGHT JOIN (với dữ liệu trên):**

| **EmployeeID** | **Name** | **Position** | **WorkDate** | **CheckIn** | **CheckOut** |
| --- | --- | --- | --- | --- | --- |
| 1 | Alice | Manager | 2025-09-12 | 08:00:00 | 17:00:00 |
| 2 | Bob | Staff | 2025-09-12 | 08:30:00 | 17:30:00 |

Nếu trong attendance có EmployeeID lạ mà không có trong employeesinfor, tên và chức vụ sẽ là NULL.

Bài mẫu 2 :

**employeesinfor**

| **EmployeeID** | **Name** |
| --- | --- |
| 1 | Alice |
| 2 | Bob |
| 3 | Charlie |

**attendance**

| **AttendanceID** | **EmployeeID** | **WorkDate** |
| --- | --- | --- |
| 1 | 1 | 2025-09-12 |
| 2 | 1 | 2025-09-13 |
| 3 | 2 | 2025-09-12 |
| 4 | 3 | 2025-09-14 |

**department**

| **DeptID** | **AttendanceID** | **DeptName** |
| --- | --- | --- |
| 1 | 1 | Sales |
| 2 | 2 | Marketing |
| 3 | 3 | IT |
| **INNER JOIN Result Grid**  Đây là **INNER JOIN**, nghĩa là chỉ lấy những dòng **có dữ liệu khớp ở cả 3 bảng** |  |  |

A screenshot of a computer

AI-generated content may be incorrect.

SELECT e.EmployeeID, e.Name, a.AttendanceID, a.WorkDate, d.DeptName

FROM employeesinfor e

INNER JOIN attendance a ON e.EmployeeID = a.EmployeeID

INNER JOIN department d ON a.AttendanceID = d.AttendanceID;

| **EmployeeID** | **Name** | **AttendanceID** | **WorkDate** | **DeptName** |
| --- | --- | --- | --- | --- |
| 1 | Alice | 1 | 2025-09-12 | Sales |
| 1 | Alice | 2 | 2025-09-13 | Marketing |
| 2 | Bob | 3 | 2025-09-12 | IT |

**LEFT JOIN Result Grid**

SELECT e.EmployeeID, e.Name, a.AttendanceID, a.WorkDate, d.DeptName

FROM employeesinfor e

LEFT JOIN attendance a ON e.EmployeeID = a.EmployeeID

LEFT JOIN department d ON a.AttendanceID = d.AttendanceID;

| **EmployeeID** | **Name** | **AttendanceID** | **WorkDate** | **DeptName** |
| --- | --- | --- | --- | --- |
| 1 | Alice | 1 | 2025-09-12 | Sales |
| 1 | Alice | 2 | 2025-09-13 | Marketing |
| 2 | Bob | 3 | 2025-09-12 | IT |
| 3 | Charlie | 4 | 2025-09-14 | NULL |

**RIGHT JOIN Result Grid**

SELECT e.EmployeeID, e.Name, a.AttendanceID, a.WorkDate, d.DeptName

FROM employeesinfor e

RIGHT JOIN attendance a ON e.EmployeeID = a.EmployeeID

RIGHT JOIN department d ON a.AttendanceID = d.AttendanceID;

| **EmployeeID** | **Name** | **AttendanceID** | **WorkDate** | **DeptName** |
| --- | --- | --- | --- | --- |
| 1 | Alice | 1 | 2025-09-12 | Sales |
| 1 | Alice | 2 | 2025-09-13 | Marketing |
| 2 | Bob | 3 | 2025-09-12 | IT |

.

**1. Với câu lệnh LEFT JOIN**

FROM employeesinfor e

LEFT JOIN attendance a ON e.EmployeeID = a.EmployeeID

LEFT JOIN department d ON a.AttendanceID = d.AttendanceID;

* **Bước 1:**
  + employeesinfor = bảng trái
  + attendance = bảng phải
  + Vì LEFT JOIN → giữ tất cả nhân viên (employeesinfor).
* **Bước 2:**
  + Kết quả của bước 1 = bảng trái
  + department = bảng phải
  + Vì LEFT JOIN → giữ toàn bộ kết quả bước 1, thêm department nếu có, nếu không thì NULL.

➡️ **Kết quả cuối cùng giữ tất cả nhân viên.**

**2. Nếu đổi thành RIGHT JOIN**

Ví dụ:

FROM employeesinfor e

RIGHT JOIN attendance a ON e.EmployeeID = a.EmployeeID

RIGHT JOIN department d ON a.AttendanceID = d.AttendanceID;

* **Bước 1:**
  + employeesinfor = bảng trái
  + attendance = bảng phải
  + Vì RIGHT JOIN → giữ tất cả attendance, chỉ thêm thông tin nhân viên nào khớp.
* **Bước 2:**
  + Kết quả bước 1 = bảng trái
  + department = bảng phải
  + Vì RIGHT JOIN → giữ tất cả department, thêm dữ liệu từ kết quả bước 1 nếu khớp, còn không thì NULL.

➡️ **Kết quả cuối cùng giữ tất cả department.**

Note :

Trong SQL xét theo hàng từ trên xuống nhưng xử lý là lấy cột trong hàng đó ra xử lý

**Nối nhiều bảng (n bảng ) lại lới nhau cấu trúc :**

SELECT ...

FROM bảng1 // ví trí bảng1 : chọn Table có thể phủ hết tất cả các bảng

LEFT JOIN bảng2 ON bảng1.id = bảng2.id; // chọn bảng đi hết nhánh rồi đi qua nhánh khác

LEFT JOIN bảng3 ON bảng1.id = bảng3.id;

......

Bảng sau from

Bảng trái (LEFT table) → bảng viết ngay sau FROM → bảng1

Bảng phải (RIGHT table) → bảng viết sau JOIN → bảng2,

**LEFT JOIN → lấy hết cột bảng trái còn bảng phải không có thì null**

**RIGHT JOIN → lấy hết cột bảng phải còn bảng trái không có thì null**

+ thực hiện truy vần ở một Table giống như 1 vòng lập for vậy quét từ trên xuống dưới