|  |  |
| --- | --- |
| **Mối quan hệ 1 - n** | Đối với mối quan hệ 1 - nhiều chúng ta sẽ bổ sung khóa chính của bảng 1 làm khóa ngoại của bảng nhiều.  Ví dụ thực thể PhongBan và thực thể NhanVien có mối quan hệ 1 - nhiều (1 phòng ban có nhiều nhân viên) thì khi đó bảng NhanVien sẽ có thêm một khóa ngoại là MaPHG (đây là khóa chính Bảng phòng ban)    Từ hình trên ta sẽ thu được 2 bảng:  NHANVIEN(**MaNV**, HoNV, TenNV, GT, NS, Luong, DiaChi, MaPHG)  PHONGBAN(**MaPHG**, TenPHG) |
| **Mối quan hệ 1 - 1** | Đối với mối quan hệ 1 - 1 chúng ta sẽ bổ sung khóa chính của bên này làm khóa ngoại của bảng bên kia  Ví dụ thực thể NhanVien có mối quan hệ 1 - 1 với thực thể MayTinh (Mỗi nhân viên chỉ có 1 máy tính) thì chúng ta sẽ bổ sung thêm MaNV làm khóa ngoại của bảng MayTinh    Từ hình trên ta sẽ thu được kết quả là 2 bảng:  NHANVIEN(**MaNV**, HoNV, TenNV, GT, NS, Luong, DiaChi)  MAYTINH(**SoMay**, TenMay, MaNV) |
| **Mối quan hệ nhiều - nhiều** | Đối với mối quan n - m để chuyển đổi từ ERD sang mô hình dữ liệu quan hệ sẽ phải làm các bước như sau:  Bước 1: Tạo ra bảng trung gian được sinh ra từ mối quan hệ n - m với chứa khóa chính của 2 bảng chuyển hóa thành khóa ngoại của bảng trung gian cùng với các thuộc tính của mối quan hệ  Bước 2: Xác định khóa trung gian như sau:  Nếu cặp khóa ngoại đủ điều kiện để đảm bảo thành khóa chính thì sẽ chọn cặp khóa ngoại thành khóa chính  Nếu cặp khóa ngoại không đủ điều kiện để đảm bảo tính duy nhất và not null thì phải tạo thêm một trường mới là mã của bảng trung gian làm khóa chính  Ví dụ thực thể NhanVien có mối quan hệ n - m với thực thể DuAn (Một nhân viên có thể làm nhiều dự án và một dự án có thể do nhiều nhân viên thực hiện) thì chúng ta sẽ làm như sau:    Từ hình trên ta thu được các bảng như sau:  NHANVIEN(**MaNV**, HoNV, TenNV, GT, NS, Luong, DiaChi)  DEAN(**MaDA**, TenDA, DDiem\_DA)  PHANCONG(MaNV, MaDA, ThoiGian) |
| **Thuộc tính đa trị** | Đối với các thuộc tính đa trị của một thực thể chúng ta sẽ chuyển đổi thành một quan hệ mới (bảng mới) có khóa chính bao gồm 2 thuộc tính là thuộc tính đa trị và thuộc tính khóa của thực thể đồng thời loại bỏ thuộc tính đó khỏi thực thể cũ.  Ví dụ: Thực thể NhanVien có thuộc tính đa trị là bằng cấp khi đó chúng ta chuyển đổi sang mô hình dữ liệu quan hệ sẽ sinh thêm một bảng có tên là NV\_BangCap với khóa chính là MaNV và BangCap và đồng thời xóa bỏ trường BangCap khỏi bảng NhanVien    Từ hình trên chúng ta thu được 2 bảng:  NHANVIEN(MaNV, HoNV, TenNV, GT, NS, Luong, DiaChi)  NV\_BANGCAP(MaNV, BangCap) |
| **Khoá chính ( primary key )** | - Khóa chính (Primary Key, hay ràng buộc khóa chính) được sử dụng để định danh duy nhất mỗi bản ghi (dòng) trong bảng của cơ sở dữ liệu.  - Ngoài ra, nó còn dùng để thiết lập quan hệ 1-n (hay ràng buộc tham chiếu) giữa hai bảng trong cơ sở dữ liệu.  - Dữ liệu (value) của field khóa chính phải có tính duy nhất, và không chứa các giá trị Null.  - Mỗi bảng chỉ có một khóa chính  - Khóa chính có thể tạo ra dựa trên một cột hoặc nhiều cột (được gọi là khoá chính tổ hợp) của table.    Hoặc |
| **Khoá ngoại ( Foreign key )** | - Khoá ngoại (foreign key - viết ngắn gọn là FK) là cơ chế để tạo liên kết giữa các bảng trong cùng một cơ sở dữ liệu. Chẳng hạn, trong CSDL quản lý bán hàng, bảng customer và bảng order có thể có khoá ngoại để liên kết với nhau.  - Khoá ngoại được đặt trên một cột của bảng này và tham chiếu đến khoá chính của một bảng khác. Chẳng hạn, khoá ngoại được đặt trên cột customer\_id của bảng order và tham chiếu đến cột id của bảng customer.  - Kiểu dữ liệu của khoá ngoại và khoá chính mà nó tham chiếu phải giống nhau. Thông thường thì các bảng đều sử dụng kiểu INTEGER cho khoá chính. |
| **Constraint ( ràng buộc )** | | Loại ràng buộc | Ý nghĩa | | --- | --- | | NOT NULL | Sử dụng để đảm bảo dữ liệu của cột không được nhận giá trị NULL | | DEFAULT | Gán giá trị mặc định trong trường hợp dữ liệu của cột không được nhập vào hay không được xác định. | | UNIQUE | Sử dụng để đảm bảo dữ liệu của cột là duy nhất, không trùng lặp giá trị trên cùng 1 cột. | | PRIMARY KEY (Khóa chính) | Dùng để thiết lập khóa chính trên bảng, xác định giá trị trên tập các cột làm khóa chính phải là duy nhất, không được trùng lặp. Việc khai báo ràng buộc khóa chính yêu cầu các cột phải NOT NULL. | | FOREIGN KEY (Khóa ngoại) | Dùng để thiết lập khóa ngoại trên bảng, tham chiếu đến bảng khác thông qua giá trị của cột được liên kết. Giá trị của cột được liên kết phải là duy nhất trong bảng kia. | | CHECK | Bảo đảm tất cả giá trị trong cột thỏa mãn điều kiện nào đó. Đây là hình thức sử dụng phổ biến để kiểm tra tính hợp lệ của dữ liệu (validate data) | |
| **Alter ( Thay đổi )** | Câu lệnh ALTER trong SQL được sử dụng để thay đổi cấu trúc của bảng hoặc đối tượng khác trong cơ sở dữ liệu. Với câu lệnh ALTER, bạn có thể thêm, sửa đổi hoặc xóa các cột, ràng buộc hoặc đối tượng khác của bảng. Các tác vụ thường được thực hiện bởi câu lệnh ALTER bao gồm thêm hoặc xóa cột, thêm hoặc xóa ràng buộc, thay đổi kiểu dữ liệu của cột và nhiều tác vụ khác nữa tùy thuộc vào loại đối tượng muốn thay đổi. |
| **Insert into ( chèn vào )** | Câu lệnh INSERT INTO trong SQL được sử dụng để thêm các hàng dữ liệu mới vào một bảng trong cơ sở dữ liệu.  Cú pháp  Có hai cú pháp cơ bản của câu lệnh INSERT INTO thể hiện dưới đây.  INSERT INTO TABLE\_NAME (column1, column2, column3,...columnN)  VALUES (value1, value2, value3,...valueN);  Ở đây, column1, column2, column3, ... columnN là tên của các cột trong bảng muốn chèn dữ liệu.  Chúng ta có thể không cần phải chỉ rõ tên cột trong truy vấn SQL nếu đang thêm các giá trị cho tất cả các cột của bảng. Nhưng hãy chắc chắn thứ tự của các giá trị theo thứ tự như các cột trong bảng. |
| **Select ( chọn )** | Câu lệnh SELECT trong SQL được sử dụng để lấy dữ liệu từ một bảng cơ sở dữ liệu trả về dữ liệu này dưới dạng một bảng kết quả. Các bảng kết quả này được gọi là tập kết quả (result-sets).  Cú pháp  Cú pháp cơ bản của câu lệnh SELECT trong SQL như sau:  SELECT column1, column2, columnN FROM table\_name;  Ở đây, column1, column2... là các trường của một bảng có các giá trị bạn muốn lấy ra.  Nếu muốn lấy tất cả các cột có trong bảng sử dụng câu lệnh sau:  SELECT \* FROM table\_name; |
| **Where ( ở đâu )** | Mệnh đề WHERE trong SQL được sử dụng để chỉ định một điều kiện trong khi lấy dữ liệu từ một bảng hoặc bằng cách join với nhiều bảng. Nếu điều kiện thỏa mãn, thì nó trả về một giá trị cụ thể từ bảng.  Cú pháp  Cú pháp cơ bản của câu lệnh SELECT với mệnh đề WHERE:  SELECT column1, column2, columnN  FROM table\_name  WHERE [condition] |
| **Order by ( đặt bởi )** | Mệnh đề ORDER BY trong SQL được sử dụng để sắp xếp dữ liệu theo thứ tự tăng dần hoặc giảm dần, dựa trên một hoặc nhiều cột. ASC được sử dụng để sắp xếp tăng dần và DESC được sử dụng để sắp xếp giảm dần.  Cú pháp  Cú pháp cơ bản của mệnh đề ORDER BY như sau:  SELECT column-list  FROM table\_name  [WHERE condition]  [ORDER BY column1, column2, .. columnN] [ASC | DESC];    \* nếu dùng order by sắp xếp theo tên theo thứ tự giảm dần mà nếu có tên trùng nhau thì nó sẽ sắp xếp theo salary. |
| **Group by** | Mệnh đề GROUP BY trong SQL được sử dụng hợp tác với câu lệnh SELECT để sắp xếp dữ liệu giống nhau thành các nhóm. Mệnh đề GROUP BY này tuân theo mệnh đề WHERE trong câu lệnh SELECT và đứng trước mệnh đề ORDER BY.  Cú pháp  Cú pháp cơ bản của mệnh đề GROUP BY được hiển thị trong khối mã sau. Mệnh đề GROUP BY trong SQL phải tuân theo các điều kiện trong mệnh đề WHERE và phải đứng trước mệnh đề ORDER BY nếu mệnh đề được sử dụng.  SELECT column1, column2  FROM table\_name  WHERE [ conditions ]  GROUP BY column1, column2  ORDER BY column1, column2 |
| **Having** | HAVING là một mệnh đề trong ngôn ngữ truy vấn SQL được sử dụng để lọc kết quả của một truy vấn có chứa hàm nhóm (GROUP BY). HAVING được sử dụng để áp dụng điều kiện cho kết quả được nhóm lại.  Khi sử dụng HAVING, các bản ghi được nhóm theo giá trị của một hoặc nhiều cột và sau đó, kết quả của từng nhóm được đưa ra để so sánh với điều kiện được chỉ định trong mệnh đề HAVING. Nếu điều kiện được chỉ định trong HAVING đúng, thì kết quả của nhóm đó sẽ được đưa vào kết quả cuối cùng của truy vấn. |
| **Toán tử AND** | Toán tử AND là một toán tử logic kết hợp hai hay nhiều biểu thức Boolean và trả về true khi và chỉ khi cả hai biểu thức được đánh giá là true. Toán tử AND trả về false nếu một trong hai biểu thức kết hợp được đánh giá là sai.    Trong cú pháp trên:  WHERE, AND là các từ khóa  boolean\_expression\_1, boolean\_expression\_2 là các biểu thức trả về kiểu boolean. |
| **Toán tử OR** | Toán tử OR kết hợp hai hay nhiều biểu thức dạng boolean, chỉ cần một trong các điều kiện kết hợp là đúng thì kết quả là true. Vậy để sự kết hợp giữa các biểu thức trả về là sai thì tất cả các biểu thức kết hợp đều phải trả về sai.    Trong cú pháp trên:  WHERE, OR là các từ khóa  boolean\_expression\_1 và boolean\_expression\_2 là các biểu thức boolean có thể trả về true, false, hoặc NULL. |
| **Câu lệnh Join** | Mệnh đề INNER JOIN khớp các hàng trong một bảng với các hàng trong các bảng khác và cho phép bạn truy vấn các hàng có chứa các cột từ cả hai bảng. Nó cũng là một phần tùy chọn của câu lệnh SELECT, xuất hiện ngay sau mệnh đề FROM.  Trước khi sử dụng mệnh đề INNER JOIN, bạn phải xác định các tiêu chí sau:  Xác định bảng chính xuất hiện trong mệnh đề FROM.  Chỉ định bảng kết hợp với bản chính sẽ xuất hiện trong mệnh đề INNER JOIN.  Chỉ định điều kiện nối. Điều kiện nối xuất hiện sau từ khóa ON của mệnh đề INNER JOIN và là quy tắc để kết hợp các hàng giữa bảng chính và các bảng khác. |
| **Dấu ( \* ) trong select** | Dấu sao (\*) trong SELECT được sử dụng để lấy tất cả các trường của bảng "Class" trong kết quả. Lấy tất cả các cột trong cái table. |
|  |  |