

# Chương 2

## Tổng quan về cơ sở dữ liệu

# Nội dung

- ❖ Các khái niệm của CSDL quan hệ.
- ❖ Các ràng buộc toàn vẹn.
- ❖ Chuẩn hóa dữ liệu.
- ❖ Các phép toán đại số quan hệ.
- ❖ Ngôn ngữ SQL.

# Các khái niệm của CSDL quan hệ

## ❖ Cơ sở dữ liệu

- ▶ *database*
- ▶ **Cơ sở dữ liệu** là sự tập hợp có tổ chức các dữ liệu có liên quan luận lý với nhau.
- ▶ **Dữ liệu** (*data*): sự biểu diễn của các đối tượng và sự kiện được ghi nhận và được lưu trữ trên các phương tiện của máy tính.
  - Dữ liệu có cấu trúc: số, ngày, chuỗi ký tự, ...
  - Dữ liệu không có cấu trúc: hình ảnh, âm thanh, đoạn phim, ...
- ▶ **Có tổ chức** (*organized*): người sử dụng có thể dễ dàng lưu trữ, thao tác và truy xuất dữ liệu.

# Các khái niệm của CSDL quan hệ

## ❖ Cơ sở dữ liệu

- ▶ **Có liên quan luận lý** (*logically related*): dữ liệu mô tả một lãnh vực mà nhóm người sử dụng quan tâm và được dùng để trả lời các câu hỏi liên quan đến lãnh vực này.

## ❖ Thông tin

- ▶ *information*
- ▶ **Thông tin** là dữ liệu đã được xử lý để làm tăng sự hiểu biết của người sử dụng.
- ▶ Dữ liệu trong ngữ cảnh.
- ▶ Dữ liệu được tổng hợp / xử lý.



# Các khái niệm của CSDL quan hệ

## ❖ Cơ sở dữ liệu quan hệ

- ▶ *relational database*
- ▶ **CSDL quan hệ** là CSDL mà dữ liệu được lưu trữ trong các bảng.

## ❖ Miền

- ▶ *domain*
- ▶ **Miền** là một tập các giá trị. Thông thường, miền là một tập con của một kiểu dữ liệu và được ký hiệu bắt đầu bằng chữ *D* (ví dụ  $D_1, D_2, \dots$ ).

# Thuộc tính

## ❖ Thuộc tính

- ▶ *attribute*
- ▶ **Thuộc tính** là một đặc tính riêng của đối tượng dữ liệu.
- ▶ Một đối tượng dữ liệu có thể có nhiều thuộc tính.
- ▶ Thông thường, thuộc tính được ký hiệu bắt đầu bằng chữ *A* (ví dụ  $A_1, A_2, \dots$ ).
- ▶ Đối với một đối tượng dữ liệu, mỗi thuộc tính đều được đặt tên phân biệt, được gọi là **tên thuộc tính** (*attribute name*).

# Thuộc tính

## ❖ Thuộc tính

- ▶ Mỗi thuộc tính được nhận giá trị từ một miền, miền này được gọi là **miền trị của thuộc tính**. Miền trị của một thuộc tính  $A$  được ký hiệu là  $dom(A)$ .
- ▶ Mỗi miền trị của thuộc tính đều có chứa một giá trị đặc biệt được gọi là **giá trị rỗng** (*null value*) dùng để đặc trưng cho **một giá trị không thể xác định được** hoặc **một giá trị chưa thể xác định được** tại thời điểm đang xét nhưng có thể được xác định tại một thời điểm sau đó.



# Thuộc tính

Nhà cung cấp có các thuộc tính:

- Mã nhà cung cấp có tên là *Snum*.
- Tên nhà cung cấp có tên là *Name*.
- Thành phố có tên là *City*.

Miền trị của thuộc tính *city* được ký hiệu là *dom(City)*.



# Quan hệ

## ❖ Lược đồ quan hệ

- ▶ *relation schema*

- ▶ **Lược đồ quan hệ  $R$**  là một tập hữu hạn các thuộc tính  $\{A_1, A_2, \dots, A_n\}$  và được ký hiệu là  $R(A_1, A_2, \dots, A_n)$  với  $R$  là tên của lược đồ quan hệ.

- ▶ Trong lược đồ quan hệ  $R$ , các tên thuộc tính  $A_1, A_2, \dots, A_n$  là duy nhất.

## ❖ Vị từ của lược đồ quan hệ

- ▶ *predicate*

- ▶ **Vị từ của lược đồ quan hệ  $R$** , ký hiệu là  $\| R \|$ , là một phát biểu cho biết ngữ nghĩa của lược đồ quan hệ  $R$ .

# Quan hệ

## ❖ **Bậc của lược đồ quan hệ**

- ▶ *degree, arity*
- ▶ ***Bậc của lược đồ quan hệ*** là số lượng thuộc tính của lược đồ quan hệ.

cuu duong than cong . com

Lược đồ quan hệ: *Supplier* (*Snum*, *Name*, *City*)

Vị từ: “Mỗi nhà cung cấp có một mã nhà cung cấp *Snum* phân biệt, tên nhà cung cấp *Name*, thuộc thành phố *City*”.

Bậc: 3

cuu duong than cong . com

# Quan hệ

## ❖ Quan hệ

### ▶ *relation*

- ▶ Gọi  $D_1, D_2, \dots, D_n$  là các miền tương ứng với các thuộc tính  $A_1, A_2, \dots, A_n$  và  $D = D_1 \cup D_2 \cup \dots \cup D_n$ .
- ▶ **Quan hệ  $r$  trên lược đồ quan hệ  $R$** , ký hiệu là  $r(R)$ , là một tập hữu hạn các ánh xạ  $\{t_1, t_2, \dots, t_p\}$  từ  $R$  vào  $D$ , với điều kiện mọi ánh xạ  $t \in r$  thì  $t(A_i) \in D_i, 1 \leq i \leq n$ .
- ▶ Quan hệ  $r$  là một **thể hiện quan hệ** (*relation instance*) của lược đồ quan hệ  $R$  tại một thời điểm.

# Quan hệ

## ❖ Quan hệ

- ▶ Mỗi ánh xạ được gọi là một **bộ** (*tuple*) của quan hệ. Một bộ của một quan hệ bậc  $n$  được gọi là **bộ- $n$**  ( *$n$ -tuple*).
- ▶ Một bộ của quan hệ bao gồm nhiều giá trị, mỗi giá trị được gọi là **thành phần** (*component*) của bộ.
  - Thành phần  $A$  của bộ  $u$  được ký hiệu là  $u[A]$  hoặc  $u.A$ , các thành phần  $X = \{A_1, A_2, \dots, A_k\}$  của bộ  $u$  được ký hiệu là  $u[X]$ .
- ▶ **Bậc của một quan hệ** trên lược đồ quan hệ bằng bậc của lược đồ quan hệ.

# Quan hệ

Lược đồ quan hệ: *Supplier* (*Snum*, *Name*, *City*)

Khóa: *Snum*

Thuộc tính không khóa: *Name*, *City*

Siêu khóa: {*Snum*, *Name*}, ...

Quan hệ: *Supplier*

Snum	Name	City
S1	Nguyễn Trung Tiến	SF
S2	Trần Thị Yến	LA
S3	Nguyễn Văn An	SF

Bộ:  $u = ('S1', 'Nguyễn Trung Tiến', 'SF')$

Thành phần:

$u[Snum, Name] = ('S1', 'Nguyễn Trung Tiến')$

# Quan hệ

## ❖ Khóa

▶ *key*

▶ **Khóa của lược đồ quan hệ  $R$**  có tập thuộc tính  $U = \{A_1, A_2, \dots, A_m\}$  là một tập con  $K = \{A_{j_1}, A_{j_2}, \dots, A_{j_n}\}$ , với  $j_1, j_2, \dots, j_n$  là các số nguyên phân biệt và nằm trong khoảng từ 1 đến  $m$ , phải thỏa mãn đồng thời hai điều kiện sau đây:

- (1)  $\forall r(R), \forall t_1, t_2 \in r$ , nếu  $t_1 \neq t_2$  thì  $t_1[K] \neq t_2[K]$
- (2) Không tồn tại  $K' \subset K$  sao cho  $K'$  thỏa mãn điều kiện (1).

# Quan hệ

## ❖ Khóa

- ▶ Một khóa chỉ có một thuộc tính được gọi là **khóa đơn** (*simple key*).
- ▶ Một khóa có nhiều thuộc tính được gọi là **khóa phức hợp** (*composite key*).
- ▶ Khóa thường được sử dụng làm **chỉ mục** (*index*) của bảng dữ liệu để làm tăng tốc độ xử lý câu truy vấn.
- ▶ Một tập thuộc tính  $K \subseteq U$  thỏa mãn điều kiện (1) được gọi là **siêu khóa** (*superkey*) của  $R$ . Một siêu khóa hiển nhiên của  $R$  là  $U$ .

# Quan hệ

## ❖ Khóa

- ▶ Một lược đồ quan hệ  $R$  phải có ít nhất một khóa và có thể có nhiều khóa.
- ▶ Các thuộc tính thuộc một khóa được gọi là **thuộc tính khóa** (*prime attribute*), các thuộc tính còn lại trong lược đồ quan hệ được gọi là các **thuộc tính không khóa** (*nonprime attribute*).
- ▶ Trong một lược đồ quan hệ, các thuộc tính khóa được gạch dưới.



# Quan hệ

## ❖ Khóa

- ▶ Tất cả các khóa của một lược đồ quan hệ được gọi là **khóa dự tuyển** (*candidate key*).
- ▶ Một trong các khóa dự tuyển được chọn làm khóa tiêu biểu, khóa này được gọi là **khóa chính** (*primary key*).
- ▶ Một lược đồ quan hệ chỉ có một khóa chính và có thể có nhiều khóa dự tuyển.
- ▶ Trong một lược đồ quan hệ, một hoặc nhiều thuộc tính được gọi là **khóa ngoại** (*foreign key*) nếu chúng là khóa chính của một lược đồ quan hệ khác.

# Quan hệ

## ❖ Lược đồ cơ sở dữ liệu

- ▶ *database schema*
- ▶ **Lược đồ cơ sở dữ liệu** là một tập hợp các lược đồ quan hệ.
- ▶ Trong một lược đồ cơ sở dữ liệu, các tên lược đồ quan hệ là duy nhất.

Lược đồ cơ sở dữ liệu:

*Emp (Empnum, Name, Sal, Tax, Mgrnum, Deptnum)*

*Dept (Deptnum, Name, Area, Mgrnum)*

*Supplier (Snum, Name, City)*

*Supply (Snum, Pnum, Deptnum, Quan)*

# Ràng buộc toàn vẹn

## ❖ Ràng buộc toàn vẹn

- ▶ *integrity constraint*
- ▶ **Ràng buộc toàn vẹn** là một qui tắc mà tất cả các dữ liệu trong CSDL phải thỏa mãn qui tắc này.

## ❖ Ràng buộc miền trị

- ▶ *domain constraint*
- ▶ Các giá trị cho phép của một thuộc tính.

## ❖ Toàn vẹn thực thể

- ▶ *entity integrity*
- ▶ Thuộc tính khóa chính không có giá trị rỗng (*null value*).

# Ràng buộc toàn vẹn

## ❖ Qui tắc hoạt động

- ▶ *action assertion*
- ▶ Các qui tắc nghiệp vụ (*business rule*).

## ❖ Ràng buộc toàn vẹn tham chiếu

- ▶ *referential integrity constraint*
- ▶ **Ràng buộc toàn vẹn tham chiếu** là một qui tắc mà tất cả các giá trị của khóa ngoại (nếu khác *null*) trong quan hệ bên phía *nhiều* phải có trong các giá trị của khóa chính trong quan hệ bên phía *một*.

# Ràng buộc toàn vẹn

## ❖ Ràng buộc toàn vẹn tham chiếu

### ▶ Qui tắc xóa các hàng dữ liệu

- **Hạn chế** (*restrict*): không cho phép xóa các hàng bên phía cha (*parent*) nếu tồn tại các hàng liên quan bên phía phụ thuộc (*dependent*).
- **Tầng** (*cascade*): tự động xóa các hàng bên phía phụ thuộc tương ứng với các hàng bên phía cha.
- **Gán null** (*set-to-null*): gán *null* cho khóa ngoại của các hàng bên phía phụ thuộc tương ứng với các hàng bên phía cha. Không áp dụng cho các thực thể yếu.

# Chuẩn hóa dữ liệu

## ❖ Chuẩn hóa dữ liệu

- ▶ *data normalization*
- ▶ **Chuẩn hóa dữ liệu** là một quá trình thuận nghịch từng bước để thay thế tập hợp các quan hệ cho trước thành các quan hệ có **cấu trúc đơn giản hơn** và **chuẩn hơn**.
- ▶ Chuẩn hóa dữ liệu nhằm để cải tiến một thiết kế CSDL luận lý **thỏa mãn các ràng buộc toàn vẹn** và **tránh dữ liệu bị lặp lại không cần thiết**.

# Chuẩn hóa dữ liệu

## ❖ Mục đích của chuẩn hóa dữ liệu

- ▶ Loại bỏ các bất thường (*anomaly*) của một quan hệ để có được các quan hệ ***có cấu trúc tốt hơn, nhỏ hơn.***
- ▶ ***Quan hệ có cấu trúc tốt*** (*well-structured relation*)
  - Có sự dư thừa dữ liệu là tối thiểu.
  - Cho phép người sử dụng thêm vào, cập nhật và xóa bỏ dữ liệu mà không gây ra sự mâu thuẫn dữ liệu.

# Chuẩn hóa dữ liệu

- ❖ Các vấn đề sau đây có thể tồn tại trong một lược đồ quan hệ:
  - ▶ Bất thường do **sự lặp lại** (*repetition anomaly*)
  - ▶ Bất thường khi **cập nhật** (*update anomaly*)
  - ▶ Bất thường khi **thêm vào** (*insertion anomaly*)
  - ▶ Bất thường khi **xóa bỏ** (*deletion anomaly*)
- ❖ Quy tắc: Một quan hệ không được chứa thông tin của nhiều hơn một kiểu thực thể.



# Chuẩn hóa dữ liệu

## ❖ Bất thường do sự lặp lại

- ▶ *repetition anomaly*
- ▶ Thông tin có thể bị lặp lại **không cần thiết**. Điều này làm lãng phí vùng nhớ lưu trữ.
- ▶ **Ví dụ:** xét lược đồ quan hệ  
Supply (Snum, Pnum, Sname, Deptnum, Quan)
  - Tên nhà cung cấp *Sname* bị lặp lại với mỗi mặt hàng *Pnum* của nhà cung cấp có mã *Snum*.

# Chuẩn hóa dữ liệu

## Supply

Snum	Sname	Pnum	Deptnum	Quan
S1	Nguyễn Trung Tiến	P1	D1	10
S1	Nguyễn Trung Tiến	P2	D1	20
S2	Trần Thị Yến	P1	D1	15
S2	Trần Thị Yến	P2	D2	30
S3	Nguyễn Văn An	P3	D1	25

Hình 2.1. Bất thường do sự lặp lại.

# Chuẩn hóa dữ liệu

## ❖ Bất thường khi cập nhật

- ▶ *update anomaly*
- ▶ Đây là hệ quả của sự lặp lại dữ liệu, thực hiện việc cập nhật có thể gặp nhiều rắc rối:  
***tốn thời gian cập nhật, mâu thuẫn dữ liệu.***
- ▶ **Ví dụ:** nếu thay đổi tên của một nhà cung cấp (sửa *Sname*) trong quan hệ *Supply* thì cần phải cập nhật nhiều bộ của nhà cung cấp này.

# Chuẩn hóa dữ liệu

## Supply

Snum	Sname	Pnum	Deptnum	Quan
S1	Nguyễn Trung Thành	P1	D1	10
S1	Nguyễn Trung Thành	P2	D1	20
S2	Trần Thị Yến	P1	D1	15
S2	Trần Thị Yến	P2	D2	30
S3	Nguyễn Văn An	P3	D1	25

Hình 2.2. Bất thường khi cập nhật.

# Chuẩn hóa dữ liệu

## ❖ Bất thường khi thêm vào

- ▶ *insertion anomaly*
- ▶ Trong một số trường hợp không thể thêm thông tin mới vào CSDL.
- ▶ **Ví dụ:** khi có thông tin của một nhà cung cấp mới mà chưa cung cấp mặt hàng nào cả thì thông tin này không thể thêm vào quan hệ *Supply*, bởi vì nếu thêm vào quan hệ này thì *Pnum* phải có giá trị *null*, nhưng *Pnum* là thuộc tính khóa nên nó không thể có giá trị *null* (ràng buộc toàn vẹn về khóa).

# Chuẩn hóa dữ liệu

## Supply

Snum	Sname	Pnum	Deptnum	Quan
S1	Nguyễn Trung Tiến	P1	D1	10
S1	Nguyễn Trung Tiến	P2	D1	20
S2	Trần Thị Yến	P1	D1	15
S2	Trần Thị Yến	P2	D2	30
S3	Nguyễn Văn An	P3	D1	25
S4	Nguyễn Ngọc Thanh Trúc	null	null	null

Hình 2.3. Bất thường khi thêm vào.

# Chuẩn hóa dữ liệu

## ❖ Bất thường khi xóa bỏ

- ▶ *deletion anomaly*
- ▶ Việc xóa bỏ có thể làm mất thông tin trong CSDL.
- ▶ **Ví dụ:** nếu nhà cung cấp có mã 'S3' không còn cung cấp mặt hàng có mã 'P3', việc xóa bỏ này trong quan hệ *Supply* sẽ làm mất thông tin của nhà cung cấp có mã 'S3', bởi vì nhà cung cấp này chỉ xuất hiện một lần trong quan hệ này.

# Chuẩn hóa dữ liệu

## Supply

Snum	Sname	Pnum	Deptnum	Quan
S1	Nguyễn Trung Tiến	P1	D1	10
S1	Nguyễn Trung Tiến	P2	D1	20
S2	Trần Thị Yến	P1	D1	15
S2	Trần Thị Yến	P2	D2	30
<del>S3</del>	<del>Nguyễn Văn An</del>	<del>P3</del>	<del>D1</del>	<del>25</del>

Hình 2.4. Bất thường khi xóa bỏ.



# Phụ thuộc hàm

## ❖ Phụ thuộc hàm

- ▶ FD – *Functional Dependency*
- ▶ Cho  $r$  là một quan hệ,  $X$  và  $Y$  là hai tập thuộc tính của  $r$ .
- ▶ Chúng ta nói “ $X$  xác định hàm  $Y$ ” hoặc “ $Y$  phụ thuộc hàm vào  $X$ ”, ký hiệu là  $X \rightarrow Y$  và được gọi là **phụ thuộc hàm** nếu:  
$$\forall u, v \in r: u[X] = v[X] \Rightarrow u[Y] = v[Y]$$
tức là, với mỗi giá trị của  $X$  trong  $r$  chỉ tương ứng với một giá trị của  $Y$ .
- ▶ Khóa của một quan hệ xác định hàm các thuộc tính không khóa của quan hệ này.

# Phụ thuộc hàm

## ❖ Định thuộc

- ▶ *determinant*
- ▶ Trong phụ thuộc hàm  $X \rightarrow Y$ ,  $X$  được gọi là **định thuộc**.

## ❖ Phụ thuộc hàm riêng phần

- ▶ *partial functional dependency*
- ▶  $X \rightarrow A$  được gọi là **phụ thuộc hàm riêng phần** nếu tồn tại  $Y \subset X$  để cho  $Y \rightarrow A$ .

## ❖ Phụ thuộc hàm đầy đủ

- ▶ *full functional dependency*
- ▶  $X \rightarrow A$  được gọi là **phụ thuộc hàm đầy đủ** nếu không tồn tại  $Y \subset X$  để cho  $Y \rightarrow A$ .

# Phụ thuộc hàm

## ❖ Phụ thuộc bắc cầu

- ▶ *transitive dependency*
- ▶  $X \rightarrow A$  được gọi là **phụ thuộc bắc cầu** nếu tồn tại  $Y$  để cho  $X \rightarrow Y$ ,  $Y \rightarrow A$ ,  $Y \not\rightarrow X$  và  $A \notin XY$ .

cuu duong than cong . com

cuu duong than cong . com

# Hệ luật suy diễn Armstrong

## ❖ Hệ luật suy diễn Armstrong

- ▶ *Armstrong's axioms*

- ▶ Gồm các luật suy diễn (*inference axiom*):

**F1. Phản xạ** (*reflexivity*):  $Y \subseteq X \Rightarrow X \rightarrow Y$

**F2. Gia tăng** (*augmentation*):  $X \rightarrow Y \Rightarrow XZ \rightarrow YZ$

**F3. Bắc cầu** (*transitivity*):

$$X \rightarrow Y \text{ và } Y \rightarrow Z \Rightarrow X \rightarrow Z$$

## ❖ Các luật suy diễn khác

**F4. Hợp** (*additivity*):  $X \rightarrow Y \text{ và } X \rightarrow Z \Rightarrow X \rightarrow YZ$

**F5. Chiếu** (*projectivity*):  $X \rightarrow YZ \Rightarrow X \rightarrow Y$

**F6. Bắc cầu giả** (*pseudotransitivity*):

$$X \rightarrow Y \text{ và } YZ \rightarrow W \Rightarrow XZ \rightarrow W$$

# Dạng chuẩn 1

## ❖ Định nghĩa

Lược đồ quan hệ  $R$  ở **dạng chuẩn 1** (1NF - *First Normal Form*) nếu mọi thuộc tính của  $R$  đều chứa các **giá trị nguyên tố** (*atomic value*), giá trị này không là *một danh sách các giá trị* hoặc *các giá trị phức hợp* (*composite value*).

## ❖ Các thuộc tính của lược đồ quan hệ $R$

- ▶ Không là thuộc tính đa trị (*multivalued attribute*).
- ▶ Không là thuộc tính phức hợp (*composite attribute*).

# Dạng chuẩn 1

- ❖ Các quan hệ ở dạng chuẩn 1 vẫn có các bất thường đã được đề cập ở trên.
- ❖ Để loại bỏ một số bất thường này, các quan hệ ở dạng chuẩn 1 cần phải được phân rã thành các quan hệ ở dạng chuẩn cao hơn.

cuu duong than cong . com

# Dạng chuẩn 1

**R**

Mãsv	Họtên	Mãlớp	Tênlớp	Điểmthi	
S1	Tiến	L1	MT01	M1	9
S1	Tiến	L1	MT01	M2	7
S1	Tiến	L1	MT01	M3	8
S2	Trúc	L1	MT01	M1	9
S2	Trúc	L1	MT01	M2	8
S3	Hiền	L2	MT02	M1	5

Hình 2.5. Lược đồ quan hệ *R* không ở dạng chuẩn 1 vì thuộc tính *Điểmthi* là thuộc tính phức hợp.

# Dạng chuẩn 1

**R**

Mãsv	Họ tên	Mã lớp	Tên lớp	Môn học	Điểm
S1	Tiến	L1	MT01	M1	9
S1	Tiến	L1	MT01	M2	7
S1	Tiến	L1	MT01	M3	8
S2	Trúc	L1	MT01	M1	9
S2	Trúc	L1	MT01	M2	8
S3	Hiền	L2	MT02	M1	5

Hình 2.6. Lược đồ quan hệ  $R$  ở 1NF vì các thuộc tính của  $R$  không là thuộc tính đa trị, không là thuộc tính phức hợp.



# Dạng chuẩn 1

## ❖ Các bất thường của quan hệ ở 1NF

### ▶ Thêm vào

- Không thể thêm thông tin của sinh viên mới có mã là *S4*, tên là *Thành*, thuộc lớp có mã là *L1* nếu sinh viên này chưa đăng ký học môn học nào cả.

### ▶ Cập nhật duong than cong . com

- Sửa tên của sinh viên có tên là *Tiến* với tên mới là *Thành* sẽ phải sửa tất cả các hàng của sinh viên này.

### ▶ Xóa bỏ

- Xóa thông tin sinh viên *S3* đăng ký môn học *M1* sẽ làm mất thông tin của sinh viên này.

### ▶ Nguyên nhân

- Tồn tại thuộc tính không khóa phụ thuộc hàm riêng phần vào khóa.

# Dạng chuẩn 2

## ❖ Định nghĩa

Lược đồ quan hệ  $R$  ở dạng chuẩn 2 (2NF - *Second Normal Form*) đối với tập phụ thuộc hàm  $\mathcal{F}$  nếu  $R$  ở dạng chuẩn 1 và mọi thuộc tính không khóa đều phụ thuộc hàm đầy đủ vào mọi khóa của  $R$ .

# Dạng chuẩn 2

**R**

Mãsv	Họ tên	Mã lớp	Tên lớp	Môn học	Điểm
S1	Tiến	L1	MT01	M1	9
S1	Tiến	L1	MT01	M2	7
S1	Tiến	L1	MT01	M3	8
S2	Trúc	L1	MT01	M1	9
S2	Trúc	L1	MT01	M2	8
S3	Hiền	L2	MT02	M1	5

**Các phụ thuộc hàm:**

**Mãsv** → {**Họ tên**, **Mã lớp**}

**Mã lớp** → **Tên lớp**

**{Mãsv, Môn học}** → **Điểm**

**Khóa của R: {Mãsv, Môn học}**

**Hình 2.7. Lược đồ quan hệ R không ở 2NF vì thuộc tính không khóa *Họ tên* phụ thuộc hàm riêng phần vào khóa {*Mãsv*, *Môn học*}.**

# Dạng chuẩn 2

$R_1$

Mãsv	Họ tên	Mã lớp	Tên lớp
S1	Tiến	L1	MT01
S2	Trúc	L1	MT01
S3	Hiền	L2	MT02

Khóa của  $R_1$ : Mãsv

$R_2$

Mãsv	Môn học	Điểm
S1	M1	9
S1	M2	7
S1	M3	8
S2	M1	9
S2	M2	8
S3	M1	5

Khóa của  $R_2$ : {Mãsv, Môn học}

Hình 2.8. Lược đồ quan hệ  $R_1$  và  $R_2$  đều ở 2NF vì các thuộc tính không khóa đều phụ thuộc hàm đầy đủ vào khóa.

## Dạng chuẩn 2

### ❖ Các bất thường của quan hệ ở 2NF

#### ▶ Thêm vào

- Không thể thêm thông tin của lớp *L3* có tên là *MT03* nếu chưa có sinh viên nào học lớp này.

#### ▶ Cập nhật

- Sửa tên của lớp có mã *L1* với tên mới là *MT\_1* sẽ phải sửa tất cả các hàng của lớp này.

#### ▶ Xóa bỏ

- Xóa thông tin của sinh viên có mã *S3* sẽ làm mất thông tin của lớp *L2*.

#### ▶ Nguyên nhân

- Tồn tại thuộc tính không khóa phụ thuộc bắc cầu vào khóa.

# Dạng chuẩn 3

## ❖ Định nghĩa 1

Lược đồ quan hệ  $R$  ở **dạng chuẩn 3** (3NF-*Third Normal Form*) đối với tập phụ thuộc hàm  $\mathcal{F}$  nếu  $R$  ở dạng chuẩn 1 và mọi thuộc tính không khóa đều không phụ thuộc bắc cầu vào một khóa của  $R$ .

## ❖ Định nghĩa 2

Lược đồ quan hệ  $R$  ở **dạng chuẩn 3** đối với tập phụ thuộc hàm  $\mathcal{F}$  nếu  $R$  ở dạng chuẩn 1 và mọi phụ thuộc hàm  $X \rightarrow A$  với  $A \notin X$  thì  $X$  là một siêu khóa của  $R$  hoặc  $A$  là một thuộc tính khóa.

# Dạng chuẩn 3

$R_1$

Mãsv	Họ tên	Mãlớp	Tênlớp
S1	Tiến	L1	MT01
S2	Trúc	L1	MT01
S3	Hiền	L2	MT02

$Mãsv \rightarrow Mãlớp$

$Mãlớp \rightarrow Tênlớp$

$Mãlớp \not\rightarrow Mãsv$

$Tênlớp \notin \{Mãsv, Mãlớp\}$

Hình 2.9. Lược đồ quan hệ  $R_1$  không ở 3NF vì thuộc tính không khóa  $Tênlớp$  phụ thuộc bắc cầu vào khóa  $Mãsv$ .

# Dạng chuẩn 3

$R_{11}$

Mã lớp	Tên lớp
L1	MT01
L2	MT02

Khóa của  $R_{11}$ : Mã lớp

$R_{12}$

Mã sv	Họ tên	Mã lớp
S1	Tiến	L1
S2	Trúc	L1
S3	Hiền	L2

Khóa của  $R_{12}$ : Mã sv

Hình 2.10. Lược đồ quan hệ  $R_{11}$  và  $R_{12}$  đều ở 3NF vì các thuộc tính không khóa đều không phụ thuộc bắc cầu vào khóa.



# Dạng chuẩn 3

**R**

Sinhvien	Môn học	Giảngviên
Tiến	CSDL	G1
Tiến	CNPM	G2
Trúc	CSDL	G1

Các phụ thuộc hàm:      Khóa của R: {*Sinhvien*, *Môn học*}  
*Giảngviên* → *Môn học*      {*Sinhvien*, *Giảngviên*}  
 {*Sinhvien*, *Môn học*} → *Giảngviên*

Hình 2.11. Lược đồ quan hệ R ở 3NF nhưng không ở BCNF vì định thuộc *Giảngviên* không là siêu khóa của R.

# Dạng chuẩn 3

## ❖ Các bất thường của quan hệ ở 3NF

### ▶ Thêm vào

- Không thể thêm thông tin giảng viên *G3* dạy môn học *KTLT* nếu chưa có sinh viên nào học môn học này.

### ▶ Cập nhật

- Sửa tên của môn học của giảng viên *G1* với môn học mới là *KTLT* sẽ phải sửa tất cả các hàng của giảng viên này.

### ▶ Xóa bỏ

- Xóa thông tin của sinh viên *Tiến* học môn học *CNPM* sẽ làm mất thông tin của giảng viên *G2*.

### ▶ Nguyên nhân

- Tồn tại định thuộc không là siêu khóa của quan hệ.

# Dạng chuẩn *Boyce-Codd*

## ❖ Định nghĩa

Lược đồ quan hệ  $R$  ở dạng chuẩn *Boyce-Codd* (BCNF - *Boyce Codd Normal Form*) đối với tập phụ thuộc hàm  $\mathcal{F}$  nếu  $R$  ở dạng chuẩn 1 và mọi phụ thuộc hàm  $X \rightarrow A$  với  $A \notin X$  thì  $X$  là một siêu khóa của  $R$ .

❖ Nếu lược đồ quan hệ  $R$  ở dạng chuẩn *Boyce-Codd* thì  $R$  cũng ở dạng chuẩn 3.

cuu duong than cong . com

# Dạng chuẩn *Boyce-Codd*

$R_1$

Giảngviên	Môn học
G1	CSDL
G2	CNPM

Khóa của  $R_1$ : Giảngviên

$R_2$

Sinhviên	Giảngviên
Tiến	G1
Tiến	G2
Trúc	G1

Khóa của  $R_2$ : {Sinhviên, Giảngviên}

Hình 2.12. Lược đồ quan hệ  $R_1$  và  $R_2$  đều ở BCNF vì mọi định thuộc đều là siêu khóa.

# Phân rã

## ❖ Phân rã

▶ *decomposition*

▶ **Phân rã**  $\rho$  của  $R(U)$  là một tập  $\{R_1(U_1), R_2(U_2), \dots, R_k(U_k)\}$  sao cho:

$$(1) U = U_1 \cup U_2 \cup \dots \cup U_k$$

$$(2) \forall r(R), r_i(R_i): r_i = \Pi_{U_i}(r) \text{ với } i = 1, 2, \dots, k$$

## ❖ Phân rã bảo toàn thông tin

▶ *lossless decomposition*

▶ Phân rã  $\rho$  của  $R$  là **phân rã bảo toàn thông tin** nếu:

$$\forall r(R): r = r_1 \bowtie r_2 \bowtie \dots \bowtie r_k$$

# Phân rã

## ❖ Phân rã bảo toàn phụ thuộc hàm

- ▶ *dependency-preserving decomposition*
- ▶ Phân rã  $\rho$  của  $R$  là **phân rã bảo toàn phụ thuộc hàm** nếu:

$$\mathcal{F}^+ = (\mathcal{F}_1 \cup \mathcal{F}_2 \cup \dots \cup \mathcal{F}_k)^+$$

với  $\mathcal{F}$  là tập phụ thuộc hàm trong  $R$  và

$\mathcal{F}_i$  là tập phụ thuộc hàm trong  $R_i$  ( $i = 1, \dots, k$ ).



# Các phép toán đại số quan hệ

## ❖ Năm phép toán cơ bản

- ▶ Phép chọn
- ▶ Phép chiếu
- ▶ Phép hợp
- ▶ Phép hiệu
- ▶ Phép tích *Descartes*

# Các phép toán đại số quan hệ

## ❖ Các phép toán khác

- ▶ Phép giao
- ▶ Phép kết- $\theta$
- ▶ Phép kết tự nhiên
- ▶ Phép kết ngoài
- ▶ Phép nửa kết- $\theta$
- ▶ Phép nửa kết tự nhiên
- ▶ Phép chia



# Phép chọn

- ❖ Cho  $r(R)$  và điều kiện  $F$  (là một biểu thức luận lý có giá trị là *true* hoặc *false*) bao gồm:
- ▶ Các toán hạng là các hằng hoặc các tên thuộc tính của lược đồ quan hệ  $R$ .
  - ▶ Các phép toán so sánh:  $=, \neq, <, \leq, >, \geq$ .
  - ▶ Các phép toán luận lý: *not* ( $\neg$ ), *and* ( $\wedge$ ), *or* ( $\vee$ ).

cuu duong than cong . com

# Phép chọn

❖ **Phép chọn** (*selection*) trên  $r$  theo điều kiện  $F$ , ký hiệu là  $\sigma_F(r)$ , cho kết quả là một quan hệ gồm các bộ của  $r$  thỏa mãn điều kiện  $F$  là *true*.

$$\sigma_F(r) = \{u \mid u \in r \text{ và } F(u) = \text{true}\}$$

với  $F(u)$  là điều kiện có được bằng cách thay thế các tên thuộc tính trong điều kiện  $F$  bởi các giá trị tương ứng trong bộ  $u$ .

# Phép chọn

**R**

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

$\sigma_{A \leq 4}$  **R**

A	B	C
1	2	3
4	5	6
1	2	7

Hình 2.13. Ví dụ về phép chọn.

# Phép chiếu

- ❖ Cho lược đồ quan hệ  $R (A_1, A_2, \dots, A_m)$  và một tập con các thuộc tính  $X = \{A_{j_1}, A_{j_2}, \dots, A_{j_n}\}$  với  $j_1, j_2, \dots, j_n$  là các số nguyên phân biệt và nằm trong khoảng từ 1 đến  $m$ .
- ❖ **Phép chiếu** (*projection*)  $r(R)$  trên một tập thuộc tính  $X$ , ký hiệu là  $\Pi_X(r)$ , cho kết quả là một quan hệ gồm các bộ  $u(u_1, u_2, \dots, u_n)$  sao cho tồn tại một bộ  $v(v_1, v_2, \dots, v_m)$  trong  $r$  để  $u_i = v_{j_i}$  với  $i = 1, 2, \dots, n$ .

$$\Pi_X(r) = \{u \mid \exists v \in r. u = v[X]\}$$

# Phép chiếu

**R**

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

$\pi_{AB}$  **R**

A	B
1	2
4	5
8	4

Hình 2.14. Ví dụ về phép chiếu.

# Phép hợp

- ❖ Cho hai quan hệ  $r$  và  $s$  trên cùng một lược đồ quan hệ  $R$ .
- ❖ **Phép hợp** (*union*) của hai quan hệ  $r$  và  $s$ , ký hiệu là  $r \cup s$ , là một quan hệ gồm các bộ của  $r$  hay của  $s$ .

$$r \cup s = \{u \mid u \in r \text{ hay } u \in s\}$$

- ❖ Phép hợp có tính giao hoán.

$$r \cup s = s \cup r$$

# Phép hợp

**R**

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

**S**

A	B	C
1	2	3
3	5	7
6	2	9

**$R \cup S$**

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5
3	5	7
6	2	9

Hình 2.15. Ví dụ về phép hợp.

# Phép hiệu

- ❖ Cho hai quan hệ  $r$  và  $s$  trên cùng một lược đồ quan hệ  $R$ .
- ❖ **Phép hiệu** (*difference*) của quan hệ  $r$  cho  $s$ , ký hiệu là  $r - s$ , là một quan hệ gồm các bộ của  $r$  không có trong  $s$ .

$$r - s = \{u \mid u \in r \text{ và } u \notin s\}$$

- ❖ Phép hiệu không có tính giao hoán.

$$r - s \neq s - r$$



# Phép hiệu

**R**

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

**R – S**

A	B	C
4	5	6
1	2	7
8	4	5

**S**

A	B	C
1	2	3
3	5	7
6	2	9

Hình 2.16. Ví dụ về phép hiệu.

# Phép tích *Descartes*

- ❖ Cho quan hệ  $r$  trên lược đồ quan hệ  $R (A_1, A_2, \dots, A_m)$  và  $s$  trên lược đồ quan hệ  $S (B_1, B_2, \dots, B_n)$ .
- ❖ **Phép tích *Descartes* (Cartesian product)** của hai quan hệ  $r$  và  $s$ , ký hiệu là  $r \times s$ , là một quan hệ trên lược đồ  $T(A_1, A_2, \dots, A_m, B_1, B_2, \dots, B_n)$  gồm các bộ  $u$  sao cho  $m$  thành phần đầu tiên là một bộ của  $r$  và  $n$  thành phần cuối cùng là một bộ của  $s$ .

$$r \times s = \{ (u_1, \dots, u_m, u_{m+1}, \dots, u_{m+n}) \mid (u_1, \dots, u_m) \in r \text{ và } (u_{m+1}, \dots, u_{m+n}) \in s \}$$

- ❖ **Phép tích *Descartes* có tính giao hoán.**

$$r \times s = s \times r$$

# Phép tích *Descartes*

**R**

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

**T**

A	D
1	5
3	7

**$R \times T$**

R.A	B	C	T.A	D
1	2	3	1	5
1	2	3	3	7
4	5	6	1	5
4	5	6	3	7
1	2	7	1	5
1	2	7	3	7
8	4	5	1	5
8	4	5	3	7

Hình 2.17. Ví dụ về phép tích.

# Phép giao

- ❖ Cho hai quan hệ  $r$  và  $s$  trên cùng một lược đồ quan hệ  $R$ .
- ❖ **Phép giao** (*intersection*) của hai quan hệ  $r$  và  $s$ , ký hiệu là  $r \cap s$ , là một quan hệ gồm các bộ có trong cả hai  $r$  và  $s$ .

$$r \cap s = \{u \mid u \in r \text{ và } u \in s\}$$

- ❖ Phép giao của hai quan hệ  $r$  và  $s$  có thể được tính từ phép hiệu:

$$r \cap s = r - (r - s) = s - (s - r)$$

- ❖ Phép giao có tính giao hoán.

$$r \cap s = s \cap r$$

# Phép giao

**R**

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

**$R \cap S$**

A	B	C
1	2	3

**S**

A	B	C
1	2	3
3	5	7
6	2	9

Hình 2.18. Ví dụ về phép giao.

# Phép kết - $\theta$

❖ Cho  $r(R)$ ,  $s(S)$  và  $T = R \cup S$  (các thuộc tính của  $R$  được xem là khác với các thuộc tính của  $S$ ),  $\theta$  là một phép so sánh ( $=, \neq, <, \leq, >, \geq$ ),  $A \in R$  và  $B \in S$  là hai thuộc tính có thể so sánh với nhau bởi phép  $\theta$ .

❖ **Phép kết- $\theta$**  ( $\theta$ -join) của  $r$  và  $s$  trên hai thuộc tính  $A$  và  $B$ , ký hiệu là  $r \bowtie_{A \theta B} s$ , cho kết quả là một quan hệ trên lược đồ quan hệ  $T$  gồm các bộ  $t$  như sau:

$$q(T) = \{t \mid \exists t_r \in r, \exists t_s \in s:$$

$$t[R] = t_r, t[S] = t_s, t[A] \theta t[B]\}$$

# Phép kết - $\theta$

- ❖ Phép kết- $\theta$  của hai quan hệ  $r$  và  $s$  có thể được tính từ phép tích và phép chọn:

$$r \bowtie_{A \theta B} s = \sigma_{A \theta B} (r \times s)$$

- ❖ Phép kết- $\theta$  có tính giao hoán.

$$r \bowtie_{A \theta B} s = s \bowtie_{A \theta B} r$$

cuu duong than cong . com

# Phép kết - $\theta$

**R**

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

**R**  $\bowtie$  **T**  
 $R.A > T.A$

R.A	B	C	T.A	D
4	5	6	1	5
4	5	6	3	7
8	4	5	1	5
8	4	5	3	7

**T**

A	D
1	5
3	7

Hình 2.19. Ví dụ về phép kết -  $\theta$ .



# Phép kết tự nhiên

- ❖ Cho  $r(R)$ ,  $s(S)$  và  $T = R \cup S$  ( $R$  và  $S$  có thể có các thuộc tính chung).
- ❖ **Phép kết tự nhiên** (*natural join*) của  $r$  và  $s$ , ký hiệu là  $r \bowtie s$ , cho kết quả là một quan hệ  $q(T)$  gồm các bộ  $t$  sao cho tồn tại các bộ  $t_r \in r$  và  $t_s \in s$  với  $t_r = t[R]$  và  $t_s = t[S]$ .  

$$r \bowtie s = \{t \mid \exists t_r \in r, \exists t_s \in s: t[R] = t_r, t[S] = t_s\}$$

cuu duong than cong . com

# Phép kết tự nhiên

- ❖ Phép kết tự nhiên không đòi hỏi hai tập thuộc tính  $R$  và  $S$  giao nhau khác rỗng. Nếu  $R \cap S = \emptyset$  thì  $r \bowtie s$  là phép tích Descartes của  $r$  và  $s$ .
- ❖ Phép kết tự nhiên của hai quan hệ  $r$  và  $s$  có thể được tính từ phép tích, phép chọn và phép chiếu:

$$r \bowtie s = \Pi_T \sigma_F (r \times s)$$

với  $F$  là biểu thức  $R.A_1 = S.A_1$  and  $R.A_2 = S.A_2$  and ... and  $R.A_m = S.A_m$

trong đó  $R \cap S = \{A_1, A_2, \dots, A_m\}$ .

- ❖ Phép kết tự nhiên có tính giao hoán.

$$r \bowtie s = s \bowtie r$$

# Phép kết tự nhiên

**R**

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

**R  $\triangleright \triangleleft$  T**

A	B	C	D
1	2	3	5
1	2	7	5

**T**

A	D
1	5
3	7

**Hình 2.20. Ví dụ về phép kết.**

# Phép kết ngoài

- ❖ Cho quan hệ  $r$  trên lược đồ quan hệ  $R$ , quan hệ  $s$  trên lược đồ quan hệ  $S$ ,  $R$  và  $S$  có thể có các thuộc tính chung. Gọi  $T = R \cup S$ .
- ❖ **Phép kết ngoài** (*outer join*) của  $r$  và  $s$ , ký hiệu là  $r \bowtie^o s$ , cho kết quả là một quan hệ  $q$  trên lược đồ quan hệ  $T$  bao gồm các bộ của  $r \bowtie s$  và các bộ được tạo từ các bộ của  $r$  không kết với các bộ của  $s$  và các bộ được tạo từ các bộ của  $s$  không kết với các bộ của  $r$ ; các thuộc tính bị thiếu của các bộ được tạo thêm sẽ lấy các giá trị rỗng (*null*).

# Phép kết ngoài

**R**

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

**R  $\bowtie$  T**

A	B	C	D
1	2	3	5
1	2	7	5
4	5	6	null
8	4	5	null
3	null	null	7

**T**

A	D
1	5
3	7

Hình 2.21. Ví dụ về phép kết ngoài.

# Phép kết ngoài

- ❖ **Phép kết ngoài trái** (*left outer join*) của  $r$  và  $s$ , ký hiệu là  $r \bowtie^{\circ}_L s$ , cho kết quả là một quan hệ  $q$  trên lược đồ quan hệ  $T$  bao gồm các bộ của  $r \bowtie s$  và các bộ được tạo từ các bộ của  $r$  không kết với các bộ của  $s$ .
- ❖ **Phép kết ngoài phải** (*right outer join*) của  $r$  và  $s$ , ký hiệu là  $r \bowtie^{\circ}_R s$ , cho kết quả là một quan hệ  $q$  trên lược đồ quan hệ  $T$  bao gồm các bộ của  $r \bowtie s$  và các bộ được tạo từ các bộ của  $s$  không kết với các bộ của  $r$ .

# Phép kết ngoài

**R**

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

**$R \bowtie T$**

A	B	C	D
1	2	3	5
1	2	7	5
4	5	6	null
8	4	5	null

**T**

A	D
1	5
3	7

Hình 2.22. Ví dụ về phép kết ngoài trái.

# Phép kết ngoài

**R**

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

**$R \bowtie_R^o T$**

A	B	C	D
1	2	3	5
1	2	7	5
3	null	null	7

**T**

A	D
1	5
3	7

Hình 2.23. Ví dụ về phép kết ngoài phải.



# Phép nửa kết - $\theta$

- ❖ Cho  $r(R)$ ,  $s(S)$ ,  $\theta$  là một phép so sánh ( $=$ ,  $\neq$ ,  $<$ ,  $\leq$ ,  $>$ ,  $\geq$ ),  $A \in R$  và  $B \in S$  là hai thuộc tính có thể so sánh với nhau bởi phép  $\theta$ .
- ❖ **Phép nửa kết- $\theta$**  ( $\theta$ -semijoin) của  $r$  và  $s$  trên hai thuộc tính  $A$  và  $B$ , ký hiệu là  $r \bowtie_{A \theta B} s$ , cho kết quả là một quan hệ trên lược đồ quan hệ  $R$  như sau:

$$r \bowtie_{A \theta B} s = \{u \mid u \in r, \exists v \in s: u[A] \theta v[B]\}$$

# Phép nửa kết - $\theta$

- ❖ Phép nửa kết- $\theta$  của hai quan hệ  $r$  và  $s$  có thể được tính từ phép kết- $\theta$  và phép chiếu:

$$r \bowtie_{A \theta B} s = \Pi_R (r \bowtie_{A \theta B} s)$$

- ❖ Phép nửa kết- $\theta$  không có tính giao hoán.

$$r \bowtie_{A \theta B} s \neq s \bowtie_{A \theta B} r$$

cuu duong than cong . com

# Phép nửa kết - $\theta$

**R**

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

**R**  $\triangleright <_{R.A > T.A}$  **T**

A	B	C
4	5	6
8	4	5

**T**

A	D
1	5
3	7

Hình 2.24. Ví dụ về phép nửa kết -  $\theta$ .

# Phép nửa kết tự nhiên

- ❖ Cho  $r(R)$ ,  $s(S)$  và  $X = R \cap S \neq \emptyset$ .
- ❖ **Phép nửa kết tự nhiên** (*natural semijoin*) của  $r$  và  $s$ , ký hiệu là  $r \bowtie s$ , cho kết quả là một quan hệ trên lược đồ quan hệ  $R$  như sau:

$$r \bowtie s = \{u \mid u \in r, \exists v \in s: u[X] = v[X]\}$$

- ❖ Phép nửa kết tự nhiên của hai quan hệ  $r$  và  $s$  có thể được tính từ phép kết tự nhiên và phép chiếu:

$$r \bowtie s = \Pi_R (r \Join s)$$

- ❖ Phép nửa kết tự nhiên không có tính giao hoán.

$$r \bowtie s \neq s \bowtie r$$

# Phép nửa kết tự nhiên

**R**

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

**$R \triangleright < T$**

A	B	C
1	2	3
1	2	7

**T**

A	D
1	5
3	7

**Hình 2.25. Ví dụ về phép kết tự nhiên.**

# Phép chia

- ❖ Cho quan hệ  $r$  trên lược đồ quan hệ  $R (A_1, A_2, \dots, A_k, A_{k+1}, \dots, A_m)$  và quan hệ  $s$  trên lược đồ quan hệ  $S (A_1, A_2, \dots, A_k)$ .
- ❖ **Phép chia** (*division / quotient*) quan hệ  $r$  cho  $s$  với  $s \neq \emptyset$ , ký hiệu là  $r \div s$ , cho kết quả là một quan hệ trên lược đồ quan hệ  $T (A_{k+1}, \dots, A_m)$  gồm các bộ  $(u_{k+1}, \dots, u_m)$  sao cho đối với tất cả các bộ  $(u_1, \dots, u_k)$  thuộc  $s$  thì bộ  $(u_1, \dots, u_k, u_{k+1}, \dots, u_m)$  thuộc  $r$ .

# Phép chia

- ❖ Phép chia quan hệ  $r$  cho  $s$  có thể được tính từ các phép chiếu, phép tích, phép hiệu:

$$r \div s = \prod_{A_{k+1}, \dots, A_m} (r) - \prod_{A_{k+1}, \dots, A_m} ((\prod_{A_{k+1}, \dots, A_m} (r) \times s) - r)$$

- ❖ Phép chia không có tính giao hoán.

$$r \div s \neq s \div r$$

cuu duong than cong . com

# Phép chia

P		Q	P ÷ Q
A	B	A	B
1	6	1	6
1	8	2	
2	6	3	
2	9	4	
3	6		
3	5		
4	6		
4	7		

Hình 2.26. Ví dụ về phép chia.



# Giới thiệu SQL

- ❖ ***Ngôn ngữ truy vấn có cấu trúc (SQL - Structured Query Language)*** là một ngôn ngữ chuẩn được dùng để tạo lập và truy vấn các cơ sở dữ liệu quan hệ.
- ❖ ***SQL là một ngôn ngữ chuẩn cho các hệ quản trị CSDL quan hệ (RDBMS - Relational DBMS).***

cuu duong than cong . com

# Các đặc điểm của ngôn ngữ SQL

- ❖ Ngôn ngữ SQL là một **ngôn ngữ tựa tiếng Anh** (*English-like language*), sử dụng các từ như *select, insert, delete* trong tập lệnh.
- ❖ Ngôn ngữ SQL là một **ngôn ngữ phi thủ tục** (*nonprocedural language*).
  - ▶ Chỉ ra các thông tin **gì** cần thiết (*what*).
  - ▶ **Không** cần phải chỉ ra cách thực hiện **như thế nào** (*how*) để có được các thông tin này.
- ❖ SQL xử lý các tập hợp mẫu tin (bảng) hơn là mỗi lần một mẫu tin đơn lẻ.

# Các đặc điểm của ngôn ngữ SQL

- ❖ Nhiều loại người có thể sử dụng SQL: người quản trị CSDL (DBA), người lập trình ứng dụng, người quản lý, người sử dụng cuối cùng (*end user*).
- ❖ SQL cung cấp nhiều lệnh cho nhiều công việc khác nhau:
  - ▶ Truy vấn dữ liệu.
  - ▶ Thêm vào, cập nhật và xóa bỏ các hàng của bảng.
  - ▶ Tạo lập, thay đổi và xóa bỏ các đối tượng CSDL.
  - ▶ Điều khiển truy xuất cơ sở dữ liệu và các đối tượng CSDL.
  - ▶ Bảo đảm tính nhất quán của CSDL.

# Môi trường SQL

## ❖ Danh mục

- ▶ *catalog*
- ▶ Tập các lược đồ dùng để mô tả CSDL.

## ❖ Lược đồ

- ▶ *schema*
- ▶ Cấu trúc dùng để chứa mô tả của các đối tượng được người sử dụng tạo ra (bảng cơ sở, khung nhìn, ràng buộc).

## ❖ Ngôn ngữ định nghĩa dữ liệu

- ▶ DDL - *Data Definition Language*
- ▶ Các lệnh dùng để định nghĩa CSDL: tạo lập (*create*), thay đổi (*alter*) và hủy bỏ (*drop*) các đối tượng dữ liệu, thiết lập các ràng buộc.



# Môi trường SQL

## ❖ Ngôn ngữ thao tác dữ liệu

- ▶ DML - *Data Manipulation Language*
- ▶ Các lệnh dùng để bảo trì và truy vấn CSDL: thêm (*insert*), sửa (*update*), xóa (*delete*) dữ liệu của bảng, truy vấn (*select*).

## ❖ Ngôn ngữ điều khiển dữ liệu

- ▶ DCL - *Data Control Language*
- ▶ Các lệnh dùng để điều khiển CSDL: quản trị các quyền (*grant*, *revoke*) và ghi nhận (*committing*) dữ liệu.

# Định nghĩa bảng

## ❖ Các bước tạo một bảng

- ▶ **Bước 1.** Xác định kiểu dữ liệu của các cột.
- ▶ **Bước 2.** Xác định các cột có thể hoặc không thể có giá trị rỗng (*null value*).
- ▶ **Bước 3.** Xác định các cột phải có các giá trị duy nhất (các khóa dự tuyển).
- ▶ **Bước 4.** Xác định khóa chính – khóa ngoại.
- ▶ **Bước 5.** Xác định các giá trị mặc nhiên.
- ▶ **Bước 6.** Xác định các ràng buộc trên các cột (mô tả miền trị).
- ▶ **Bước 7.** Tạo bảng và các chỉ mục của bảng.

# Định nghĩa bảng

## Cú pháp của lệnh CREATE TABLE

```
CREATE TABLE tablename  
( { column definition [ table constraint ] } . . .  
[ON COMMIT {DELETE | PRESERVE} ROWS] );
```

where *column definition* ::=

*column\_name*

{ *domain name* | *datatype* [(*size*)] }

[ *column\_constraint\_clause* . . . ]

[ *default value* ]

[ *collate clause* ]

and *table constraint* ::=

[ CONSTRAINT *constraint\_name* ]

*Constraint\_type* [ *constraint\_attributes* ]

```
CREATE TABLE <table name> [<list of columns>]  
AS SELECT statement;
```

# Thay đổi định nghĩa bảng

- ❖ Lệnh ***ALTER TABLE*** dùng để thay đổi định nghĩa của một bảng.

Cú pháp của lệnh **ALTER TABLE**.

**ALTER TABLE** *<table name>*

**[ADD | MODIFY | DROP options]**

**(*<column definition>* [*<column constraint>*])**

**[ENABLE clause | DISABLE clause];**

cuu duong than cong . com



# Thay đổi định nghĩa bảng

## ❖ Các tùy chọn của lệnh *ALTER TABLE*

- ▶ **ADD:** Thêm một cột và/hoặc các ràng buộc vào một bảng.
- ▶ **MODIFY:** Thay đổi định nghĩa của một cột.
- ▶ **DROP:** Hủy bỏ một ràng buộc của một bảng.

cuu duong than cong . com

cuu duong than cong . com

# Thay đổi định nghĩa bảng

## ❖ Hạn chế của lệnh ***ALTER TABLE***

- ▶ Không thể thay đổi một cột đang chứa các giá trị ***NULL*** thành ***NOT NULL***.
- ▶ Không thể thêm một cột mới với ràng buộc ***NOT NULL***. Phải cho cột này chứa các giá trị rỗng, điền đầy đủ vào cột này và sau đó thay đổi cột này thành ***NOT NULL***.
- ▶ Không thể giảm kích thước hoặc thay đổi kiểu dữ liệu của một cột, trừ khi cột này không có chứa dữ liệu.
- ▶ Không thể sử dụng tùy chọn ***MODIFY*** để định nghĩa các ràng buộc trên một cột ngoại trừ ***NULL/NOT NULL***.



# Thay đổi định nghĩa bảng

Thêm một cột và/hoặc các ràng buộc vào một bảng:

```
ALTER TABLE <table name>
```

```
ADD (<column definition> [<column constraint>]);
```

Thêm cột Type vào bảng Customer\_T

```
ALTER TABLE Customer_T
```

```
ADD (Type VARCHAR(2));
```

Thêm ràng buộc của cột Standard\_Price của bảng Product\_T

```
ALTER TABLE Product_T
```

```
ADD (CHECK(Standard_Price > 0));
```



# Thay đổi định nghĩa bảng

Thay đổi định nghĩa của một cột:

```
ALTER TABLE <table name>
```

```
MODIFY (<column name> <type> [NULL]);
```

Thay đổi chiều dài của cột Customer\_Name của bảng Customer\_T

```
ALTER TABLE Customer_T
```

```
MODIFY (Customer_Name VARCHAR2(30));
```

cuu duong than cong . com



# Thay đổi định nghĩa bảng

Hủy bỏ ràng buộc của một bảng:

```
ALTER TABLE <table name>
```

```
DROP
```

```
[CONSTRAINT <constraint name>
```

```
| PRIMARY KEY
```

```
| UNIQUE (<list of columns>)]
```

```
[CASCADE];
```

Hủy bỏ ràng buộc khóa chính của bảng Order\_line\_T

```
ALTER TABLE Order_Line_T
```

```
DROP CONSTRAINT Order_Line_PK;
```

Tùy chọn CASCADE hủy bỏ tất cả ràng buộc khóa ngoại tham chiếu đến bảng Customer\_T

```
ALTER TABLE Customer_T
```

```
DROP CONSTRAINT Customer_PK CASCADE;
```



# Thay đổi định nghĩa bảng

Cho phép một ràng buộc có hiệu lực / không có hiệu lực:

```
ALTER TABLE <table name>
```

```
[ENABLE | DISABLE]
```

```
[CONSTRAINT <constraint name>
```

```
| PRIMARY KEY
```

```
| UNIQUE (<list of columns>)]
```

```
[CASCADE];
```

Làm mất hiệu lực của ràng buộc Customer\_PK của bảng Customer\_T

```
ALTER TABLE Customer_T
```

```
DISABLE CONSTRAINT Customer_PK CASCADE;
```



# Hủy bỏ bảng

- ❖ Lệnh ***DROP TABLE*** dùng để hủy bỏ một bảng trong một lược đồ.

Cú pháp của lệnh **DROP TABLE**:

**DROP TABLE** <*table name*> [CASCADE CONSTRAINTS];

cuu duong than cong . com

**Hủy bỏ bảng Order\_Line\_T**

**DROP TABLE** Order\_Line\_T;

cuu duong than cong . com

# Lệnh *INSERT*

## ❖ Thêm dữ liệu vào một bảng

Cú pháp của lệnh *INSERT* - Thêm một hàng:

```
INSERT INTO <table name> [(<list of columns>)]  
VALUES (<list of expressions>);
```

[cuuduongthancong.com](http://cuuduongthancong.com)

Cú pháp của lệnh *INSERT* - Thêm nhiều hàng:

```
INSERT INTO <table name> [(<list of columns>)]  
SELECT statement;
```

[cuuduongthancong.com](http://cuuduongthancong.com)





# Lệnh *INSERT*

```
INSERT INTO Customer_T  
VALUES (001, 'Contemporary Casuals',  
'1355 S. Himes Blvd.', 'Gainesville', 'FL', 32601);
```

```
INSERT INTO Product_T (Product_ID,  
    Product_Description, Product_Finish,  
    Standard_Price, Product_On_Hand)  
VALUES (1, 'End Table', 'Cherry', 175, 8);
```

```
INSERT INTO CA_Customer_T  
SELECT *  
FROM Customer_T  
WHERE State = 'CA';
```



# Lệnh *DELETE*

## ❖ Xóa bỏ các hàng của một bảng

Cú pháp của lệnh **DELETE**:

```
DELETE [FROM] <table name>  
[WHERE <row conditions>];
```

*cuu duong than cong . com*

**Xóa một số hàng của bảng Customer\_T**

```
DELETE FROM Customer_T  
WHERE State = 'HI';
```

*cuu duong than cong . com*

**Xóa tất cả các hàng của bảng Customer\_T**

```
DELETE FROM Customer_T ;
```



# Lệnh *UPDATE*

- ❖ Cập nhật dữ liệu của các hàng của một bảng

## Cú pháp của lệnh *UPDATE*:

*UPDATE* *<table name>* [*<alias>*]

*SET* *<column1>* = {*<expression>*, *<subquery>*}

[, *<column2>* = {*<expression>*, *<subquery>*} ...]

[*WHERE* *<row conditions>*];

## Cập nhật một số hàng của bảng *Product\_T*

*UPDATE* *Product\_T*

*SET* *Unit\_Price* = 775

*WHERE* *Product\_ID* = 7;

# Lệnh ***SELECT***

- ❖ Dùng để truy vấn dữ liệu của một bảng hoặc nhiều bảng.
- ❖ Lệnh ***SELECT*** thực hiện các phép toán của đại số quan hệ.
  - ▶ Phép tích
  - ▶ Phép kết
  - ▶ Phép chọn
  - ▶ Phép chiếu



# Lệnh *SELECT*

Cú pháp của lệnh **SELECT**:

```
SELECT [DISTINCT] <list of expressions>  
          [INTO <list of variables>]  
FROM <list of tables>  
[WHERE <row conditions>]  
[GROUP BY <list of expressions> com]  
[HAVING <group conditions>]]  
[ORDER BY <list of expressions>];
```

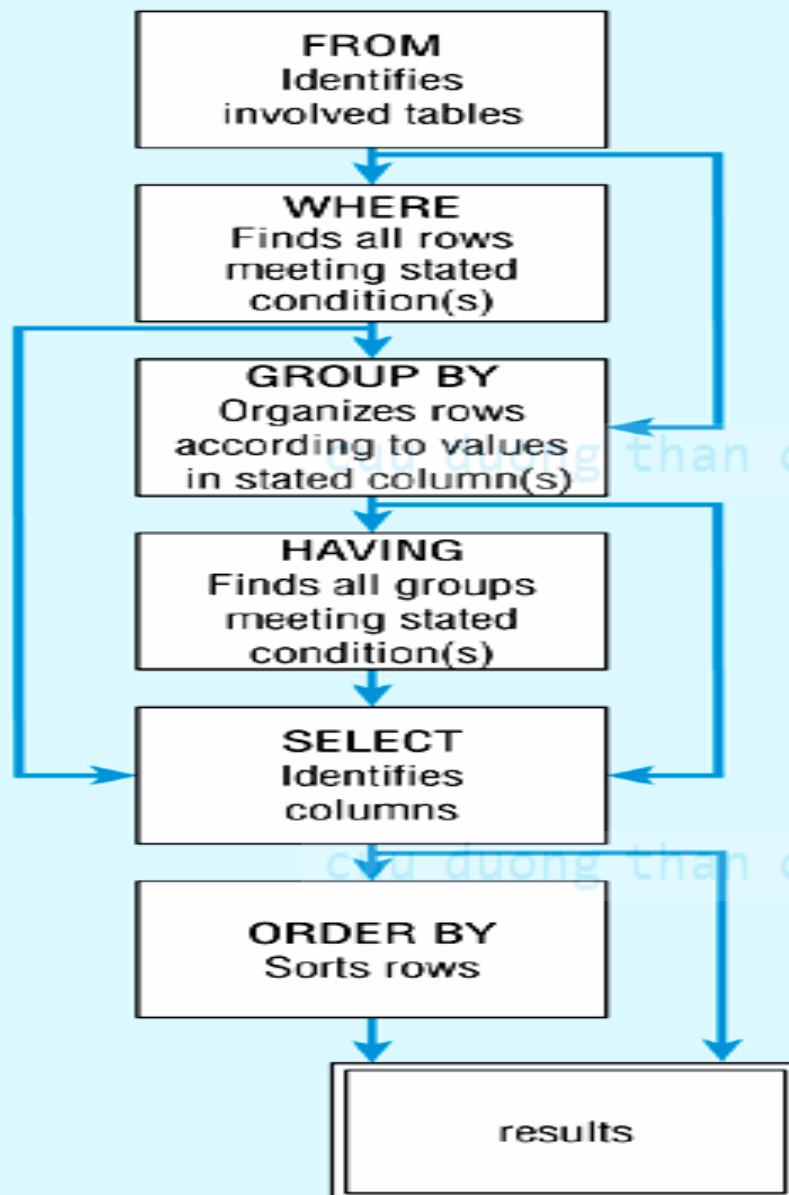
**cuu duong than cong . com**

# Lệnh *SELECT*

## ❖ Các mệnh đề của lệnh *SELECT*

- ▶ ***SELECT***: liệt kê các cột (các biểu thức) của kết quả.
- ▶ ***FROM***: các bảng hoặc các khung nhìn chứa dữ liệu cần thiết cho truy vấn.
- ▶ ***WHERE***: điều kiện xử lý các hàng để tạo ra kết quả.
- ▶ ***GROUP BY***: gom nhóm các hàng.
- ▶ ***HAVING***: điều kiện xử lý các nhóm.
- ▶ ***ORDER BY***: sắp thứ tự kết quả.

# Lệnh *SELECT*



**Hình 2.27. Thứ tự xử lý các mệnh đề của lệnh *SELECT*.**

# Lệnh *SELECT*

## ❖ Mệnh đề *SELECT*

- ▶ *SELECT [DISTINCT] <list of expressions>*
- ▶ Thực hiện **phép chiếu** của đại số quan hệ.
- ▶ ***list of expressions*** là danh sách các biểu thức dùng để tạo ra kết quả, các biểu thức này được phân cách nhau bởi dấu phẩy.
- ▶ Biểu thức có thể là một hằng, một biến (hoặc cột), hoặc sự kết hợp giữa các hằng, các biến với các phép toán (hàm).
- ▶ Mỗi biểu thức có thể có một bí danh (*alias*) đứng ngay phía sau, được gọi là ***bí danh cột*** (*column alias*).



# Lệnh *SELECT*

## ❖ Mệnh đề *SELECT*

- ▶ Nếu *list of expressions* là dấu \* thì tất cả các cột được đưa vào kết quả.

- ▶ Cột có thể có dạng:

*table\_name.column\_name*

- ▶ Từ khóa ***DISTINCT*** dùng để bảo đảm các hàng trong kết quả của truy vấn là khác nhau.
  - Nếu có nhiều cột được chọn thì *DISTINCT* ảnh hưởng đến toàn bộ các cột này.
  - Từ khóa *DISTINCT* phải được đặt ngay sau từ khóa *SELECT*.



# Lệnh *SELECT*

## ❖ Mệnh đề *INTO*

- ▶ *INTO* <list of variables>
- ▶ Được sử dụng trong *Oracle PL/SQL*, dùng để gán giá trị của các biểu thức cho các biến theo thứ tự tương ứng.
- ▶ *List of variables* là danh sách các biến được phân cách nhau bởi dấu phẩy.

# Lệnh *SELECT*

## ❖ Mệnh đề *FROM*

- ▶ *FROM* <list of tables>
- ▶ Thực hiện **phép tích** của đại số quan hệ, dùng để chỉ ra các bảng chứa dữ liệu cần lấy ra.
- ▶ **List of tables** là danh sách các bảng được phân cách nhau bởi dấu phẩy.
- ▶ Mỗi bảng có thể có bí danh bảng đứng ngay phía sau tên bảng được phân cách bởi ký tự trống; khi đó, chỉ sử dụng bí danh bảng và không được sử dụng tên bảng.



# Lệnh *SELECT*

```
SELECT *  
FROM Order_T;
```

```
SELECT Order_ID, Order_Date, Customer_ID  
FROM Order_T;
```

cuu duong than cong . com

```
SELECT DISTINCT Order_Date “Date of Order”  
FROM Order_T;
```

```
SELECT Order_ID AS Identifier, Order_Date Date  
FROM Order_T;
```

cuu duong than cong . com

# Lệnh *SELECT*

## ❖ Mệnh đề *WHERE*

- ▶ *WHERE* <row conditions>
- ▶ Thực hiện các **phép chọn, phép kết** của đại số quan hệ.
- ▶ **Row conditions** là các điều kiện được xét trên mỗi hàng, các hàng nào thỏa mãn các điều kiện này thì được đưa vào kết quả của truy vấn.
- ▶ Để lệnh *SELECT* có ngữ nghĩa, nếu mệnh đề *FROM* có  $n$  bảng thì mệnh đề *WHERE* phải có  $n-1$  điều kiện kết.

# Lệnh *SELECT*

## Các phép toán so sánh

=	Bằng
<	Nhỏ hơn
<=	Nhỏ hơn hoặc bằng
>	Lớn hơn
>=	Lớn hơn hoặc bằng
<>	Không bằng
!=	Không bằng
BETWEEN ... AND ...	Giữa hai giá trị
IN (list)	Một trong các giá trị của danh sách
LIKE string	Giống khuôn mẫu
IS NULL	Là giá trị rỗng

## Các phép toán phủ định

NOT BETWEEN ... AND ...  
NOT IN (list)  
NOT LIKE string  
IS NOT NULL

# Lệnh *SELECT*

<b>NOT</b>	True	False	Null
	False	True	Null

<b>AND</b>	True	False	Null
True	True	False	Null
False	False	False	False
Null	Null	False	Null

<b>OR</b>	True	False	Null
True	True	True	True
False	True	False	Null
Null	True	Null	Null



# Lệnh **SELECT**

```
SELECT Product_ID, Standard_Price  
FROM Product_T  
WHERE Standard_Price < 275;
```

```
SELECT Cust.Customer_Name AS Name,  
       Customer_Address  
FROM Customer_T Cust  
WHERE Customer_Name = 'Home Furnishings';
```

```
SELECT Product_ID, Standard_Price  
FROM Product_  
WHERE Standard_Price BETWEEN 100 AND 200;
```

```
SELECT Customer_Name, City, State  
FROM Customer_T  
WHERE State IN ('FL', 'TX', 'CA', 'HI');
```





# Lệnh ***SELECT***

```
SELECT Product_Description, Product_Finish,  
       Standard_Price  
FROM Product_T  
WHERE (Product_Description LIKE '%Desk'  
       OR Product_Description LIKE '_A%')  
       AND Standard_Price > 300;
```

```
SELECT Product_ID, Product_Finish, Standard_Price  
FROM Product_T  
WHERE Product_Description IS NULL;
```

```
SELECT COUNT(*)  
FROM Order_Line_T  
WHERE Order_ID = 1004;
```



# Lệnh *SELECT*

## ❖ Mệnh đề *GROUP BY*

- ▶ *GROUP BY* <list of expressions>
- ▶ Dùng để phân chia các hàng của một bảng thành các nhóm nhỏ hơn.
- ▶ Các hàm nhóm có thể được sử dụng để trả về thông tin chung cho mỗi nhóm.
- ▶ Mỗi nhóm chỉ xuất hiện ở một hàng trong kết quả của truy vấn.
- ▶ Trong trường hợp lệnh *SELECT* có cả hai mệnh đề *WHERE* và *GROUP BY* thì các hàng sẽ được chọn bởi điều kiện của mệnh đề *WHERE*, rồi sau đó phân chia các hàng được chọn thành các nhóm.

# Lệnh ***SELECT***

## ❖ Mệnh đề ***GROUP BY***

- ▶ Các cột trong mệnh đề ***GROUP BY*** không bắt buộc phải có trong mệnh đề ***SELECT***. Để cho kết quả của truy vấn có ngữ nghĩa thì các cột của mệnh đề ***GROUP BY*** nên có trong mệnh đề ***SELECT***.
- ▶ Khi có mệnh đề ***GROUP BY***, tất cả các cột có trong mệnh đề ***SELECT*** phải có trong mệnh đề ***GROUP BY***, ngoại trừ chúng ở trong hàm nhóm.
- ▶ Nếu lệnh ***SELECT*** không có mệnh đề ***GROUP BY*** thì toàn bộ bảng được xem là một nhóm. Nếu mệnh đề ***SELECT*** có chứa hàm nhóm thì không thể lấy được chi tiết của mỗi hàng của nhóm.



# Lệnh ***SELECT***

```
SELECT State, COUNT(State)
FROM Customer_T
GROUP BY State;
```

```
SELECT State, COUNT(State)
FROM Customer_T
WHERE State IN ('FL', 'TX', 'CA', 'HI')
GROUP BY State;
```

# Lệnh *SELECT*

## ❖ Mệnh đề *HAVING*

- ▶ *HAVING* <group conditions>
- ▶ Dùng để xác định các nhóm được đưa vào kết quả của truy vấn.
- ▶ *Group conditions* là các điều kiện được xét cho mỗi nhóm.
- ▶ Mệnh đề *HAVING* có thể đứng trước mệnh đề *GROUP BY*, nhưng mệnh đề *GROUP BY* nên đứng trước để cho dễ hiểu.
- ▶ Tất cả các cột có trong mệnh đề *HAVING* phải có trong mệnh đề *GROUP BY*, ngoại trừ chúng ở trong hàm nhóm.

# Lệnh *SELECT*

## ❖ Mệnh đề *HAVING*

- ▶ Các nhóm được tạo ra và các hàm nhóm được tính toán trước khi thực hiện mệnh đề *HAVING* để chọn ra các nhóm.
- ▶ Mệnh đề *WHERE* không được dùng để chọn các nhóm.
- ▶ Nếu lệnh *SELECT* có các mệnh đề *WHERE*, *GROUP BY* và *HAVING* thì thứ tự thực hiện của các mệnh đề này là *WHERE* (để chọn các hàng), kế tiếp *GROUP BY* (để phân chia nhóm) và sau cùng là *HAVING* (để chọn các nhóm).



# Lệnh ***SELECT***

```
SELECT State, COUNT(State)
FROM Customer_T
GROUP BY State
HAVING COUNT(State) > 1;
```

```
SELECT State, COUNT(State)
FROM Customer_T
WHERE State IN ('FL', 'TX', 'CA', 'HI')
GROUP BY State
HAVING COUNT(State) > 1;
```

# Lệnh *SELECT*

## ❖ Mệnh đề *ORDER BY*

- ▶ *ORDER BY* <list of expressions>
- ▶ Thông thường, thứ tự của các hàng được trả về trong kết quả của truy vấn là không xác định. Mệnh đề *ORDER BY* có thể được dùng để sắp thứ tự các hàng này.
- ▶ Mệnh đề *ORDER BY* luôn luôn là mệnh đề cuối cùng của lệnh *SELECT*.
- ▶ Thứ tự ngầm định là tăng dần (**ASC** – *ASCending*), từ khóa **DESC** (*descending*) đứng ngay sau tên cột dùng để chỉ định thứ tự giảm dần.
- ▶ Các cột trong mệnh đề *ORDER BY* không bắt buộc phải có trong mệnh đề *SELECT*.





# Lệnh ***SELECT***

```
SELECT State, COUNT(State)
FROM Customer_T
WHERE State IN ('FL', 'TX', 'CA', 'HI')
GROUP BY State
HAVING COUNT(State) > 1
ORDER BY State DESC;
```

# Hàm kết hợp

❖ **Hàm kết hợp** (*aggregate function*) còn được gọi là hàm nhóm (*group function*).

Hàm	Giá trị trả về
AVG ([DISTINCT   <u>ALL</u> ] n)	Giá trị trung bình của <i>n</i> , bỏ qua các giá trị rỗng.
COUNT ([DISTINCT   <u>ALL</u> ] expr *)	Số hàng mà <i>expr</i> có giá trị khác rỗng. * làm cho COUNT đếm tất cả các hàng được chọn, bao gồm các hàng trùng nhau và các hàng có giá trị rỗng.
MAX ([DISTINCT   <u>ALL</u> ] expr)	Giá trị lớn nhất của <i>expr</i> .
MIN ([DISTINCT   <u>ALL</u> ] expr)	Giá trị nhỏ nhất của <i>expr</i> .
SUM ([DISTINCT   <u>ALL</u> ] n)	Tổng giá trị của <i>n</i> , bỏ qua các giá trị rỗng.
STDDEV ([DISTINCT   <u>ALL</u> ] n)	Độ lệch chuẩn ( <i>STanDard DEVIation</i> ) của <i>n</i> , bỏ qua các giá trị rỗng.
VARIANCE ([DISTINCT   <u>ALL</u> ] n)	Phương sai của <i>n</i> , bỏ qua các giá trị rỗng.

# Hàm kết hợp

## ❖ Sử dụng các hàm kết hợp

- ▶ **Kết hợp đơn** (*scalar aggregate*): truy vấn trả về một giá trị của hàm kết hợp.
- ▶ **Kết hợp vector** (*vector aggregate*): truy vấn trả về nhiều giá trị của hàm kết hợp (dùng **GROUP BY**).

```
SELECT MAX(Standard_Price), MIN(Standard_Price)  
FROM Product_T;
```

```
SELECT State, COUNT(State)  
FROM Customer_T  
GROUP BY State;
```



# Thứ tự thực hiện ưu tiên của phép toán

❖ Các phép toán có cùng độ ưu tiên sẽ được thực hiện từ trái qua phải:

- ▶ Biểu thức con trong dấu ngoặc
- ▶ Các phép toán số học  $*$ ,  $/$
- ▶ Các phép toán số học  $+$ ,  $-$
- ▶ Các phép toán so sánh và các phép toán SQL:  $=$ ,  $\neq$ ,  $<$ ,  $\leq$ ,  $>$ ,  $\geq$ , ***BETWEEN ... AND, IN, LIKE, IS NULL***
- ▶ ***NOT***
- ▶ ***AND***
- ▶ ***OR***

# Các phép toán tập hợp

- ❖ Hai lệnh ***SELECT*** có thể được kết nối với nhau bằng các phép toán tập hợp bao gồm phép hợp (*union*), phép giao (*intersection*) và phép hiệu (*minus*).

cuu duong than cong . com

```
SELECT statement_1  
UNION [ALL] | INTERSECT | MINUS  
SELECT statement_2;
```

cuu duong than cong . com

# Các phép toán tập hợp

- ❖ **UNION** thực hiện phép hợp kết quả của hai truy vấn.
- ❖ **INTERSECT** thực hiện phép giao kết quả của hai truy vấn.
- ❖ **MINUS** thực hiện phép hiệu kết quả của hai truy vấn.
- ❖ Từ khóa **ALL** cho phép các hàng trong kết quả có thể trùng nhau.

cuu duong than cong . com



# Các phép toán tập hợp

```
SELECT State  
FROM Customer_T  
WHERE State NOT IN ('FL', 'TX', 'CA', 'HI');
```

```
SELECT State  
FROM Customer_T  
MINUS
```

```
SELECT State  
FROM Customer_T  
WHERE State IN ('FL', 'TX', 'CA', 'HI');
```

# Xử lý giá trị rỗng

- ❖ Hàm **NVL** (*Null VaLue*) dùng để đổi giá trị rỗng thành một giá trị khác rỗng.
- ❖ Hàm **NVL** có hai tham số: một biểu thức và một giá trị khác rỗng.
- ❖ Nếu giá trị của các cột trong một biểu thức là *null* thì giá trị của biểu thức này là *null*.

```
SELECT Product_ID, NVL(Standard_Price, 0)  
FROM Product_T  
WHERE Standard_Price IS NULL;
```



# Truy vấn con

## ❖ Truy vấn con

- ▶ *subquery*
- ▶ Là một truy vấn (lệnh **SELECT**) nằm trong một truy vấn khác.
- ▶ Truy vấn ngoài (*outer query*)
- ▶ Truy vấn trong (*inner query*)
- ▶ Truy vấn chính (*main query*)

## ❖ Xuất hiện

- ▶ Trong điều kiện của mệnh đề **WHERE**.
- ▶ Như là một bảng trong mệnh đề **FROM**.
- ▶ Trong điều kiện của mệnh đề **HAVING**.

# Truy vấn con

## ❖ Các loại truy vấn con

### ▶ Truy vấn con lồng nhau (*nested subquery*)

- Không phụ thuộc vào dữ liệu của truy vấn ngoài.
- Được thực hiện duy nhất một lần trước khi thực hiện truy vấn ngoài.
- Kết quả của truy vấn con được dùng để thực hiện truy vấn ngoài.

### ▶ Truy vấn con tương quan (*correlated subquery*)

- Sử dụng dữ liệu của truy vấn ngoài.
- Đối với mỗi hàng của truy vấn ngoài, dữ liệu của hàng này được dùng để thực hiện truy vấn con, kết quả của truy vấn con được dùng để thực hiện truy vấn ngoài.
- Có thể sử dụng phép toán **EXISTS**.

# Truy vấn con

```
SELECT Customer_Name
FROM Customer_T
WHERE Customer_ID IN
```

The IN operator will test to see if the Customer\_ID value of a row is included in the list returned from the subquery

```
(SELECT DISTINCT Customer_ID
FROM Order_T);
```

Subquery is embedded in parentheses. In this case it returns a list that will be used in the WHERE clause of the outer query

Hình 2.28. Ví dụ về truy vấn con lồng nhau.

```
SELECT CUSTOMER_NAME
      FROM CUSTOMER_T
     WHERE CUSTOMER_ID IN
```

```
(SELECT DISTINCT CUSTOMER_ID
   FROM ORDER_T);
```

1. The subquery (shown in the box) is processed first and an intermediate results table created:

CUSTOMER\_ID

1  
6  
15  
5  
3  
2  
11  
12  
4

9 rows selected.

**No reference to data in  
outer query, so subquery  
executes once only**

2. The outer query returns the requested customer information for each customer included in the intermediate results table:

CUSTOMER\_NAME

Contemporary Casuals  
Value Furniture  
Home Furnishings  
Eastern Furniture  
Impressions  
California Classics  
American Euro Lifestyles  
Battle Creek Furniture  
Mountain Scenes  
9 rows selected.

**These are the only  
customers that have  
IDs in the Order\_T table**

# Truy vấn con

```
SELECT DISTINCT Order_ID
```

```
FROM Order_Line_T
```

```
WHERE EXISTS
```

```
(SELECT *
```

```
FROM Product_T
```

```
WHERE Product_ID = Order_Line_T.Product_ID
```

```
AND Product_Finish = 'Natural ash');
```

The EXISTS operator will return a TRUE value if the subquery resulted in a non-empty set, otherwise it returns a FALSE

The subquery is testing for a value that comes from the outer query

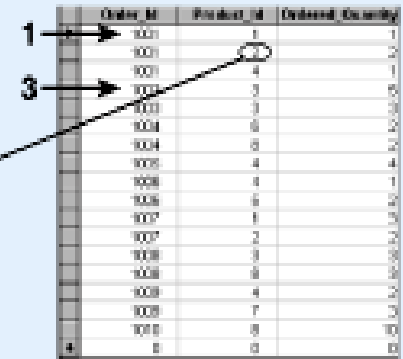
Hình 2.29. Ví dụ về truy vấn con tương quan.

```

SELECT DISTINCT ORDER_ID FROM ORDER_LINE_T
WHERE EXISTS
  (SELECT *
   FROM PRODUCT_T
    WHERE PRODUCT_ID = ORDER_LINE_T.PRODUCT_ID
      AND PRODUCT_FINISH = 'Natural Ash');

```

**Subquery refers to outer-query data, so executes once for each row of outer query**



Order_id	Product_id	Ordered_Quantity
1001	1	1
1001	2	1
1001	4	1
1002	3	1
1003	3	1
1004	6	1
1004	8	1
1005	4	1
1006	4	1
1006	6	1
1007	1	1
1007	2	1
1008	3	1
1008	6	1
1008	4	1
1009	7	1
1010	8	1
1010	3	1

	Product_ID	Product_Description	Product_Finish	Standard_Price	Product_Line_Id
1	1	End Table	Cherry	\$175.00	10001
2	2	Coffee Table	Natural Ash	\$200.00	20001
4	3	Computer Desk	Natural Ash	\$375.00	20001
	4	Entertainment Center	Natural Maple	\$650.00	30001
	5	Writer's Desk	Cherry	\$325.00	10001
	6	8-Drawer Dresser	White Ash	\$750.00	20001
	7	Dining Table	Natural Ash	\$800.00	20001
	8	Computer Desk	Walnut	\$250.00	30001
*	(AutoNumber)			\$0.00	

1. The first order ID is selected from ORDER\_LINE\_T: ORDER\_ID =1001.
2. The subquery is evaluated to see if any product in that order has a natural ash finish. Product 2 does, and is part of the order. EXISTS is valued as true and the order ID is added to the result table.
3. The next order ID is selected from ORDER\_LINE\_T: ORDER\_ID =1002.
4. The subquery is evaluated to see if the product ordered has a natural ash finish. It does. EXISTS is valued as true and the order ID is added to the result table.
5. Processing continues through each order ID. Orders 1004, 1005, and 1010 are not included in the result table because they do not include any furniture with a natural ash finish. The final result table is shown in the text on page 303.

# Truy vấn con

Subquery forms the derived table used in the FROM clause of the outer query

One column of the subquery is an aggregate function that has an alias name. That alias can then be referred to in the outer query

```
SELECT Product_Description, Standard_Price, Avg_Price
FROM (SELECT AVG(Standard_Price) Avg_Price
      FROM Product_T), Product_T
WHERE Standard_Price > Avg_Price;
```

The WHERE clause normally cannot include aggregate functions, but because the aggregate is performed in the subquery its result can be used in the outer query's WHERE clause

Hình 2.30. Ví dụ về truy vấn con xuất hiện trong mệnh đề FROM.