**10 bài tập tổng hợp mức Expert** về chuẩn hóa 1NF, 2NF, 3NF trong thiết kế CSDL SQL, yêu cầu phân tích sâu về phụ thuộc hàm, khóa chính và các vấn đề phức tạp trong thực tế.

### **Bài 1: Hệ thống đặt phòng khách sạn**

Bảng HotelBookings:

sql

Copy

(booking\_id, guest\_id, guest\_phone, room\_id, room\_type, room\_price, check\_in, check\_out, hotel\_id, hotel\_city)

* **Khóa chính**: (booking\_id, room\_id, check\_in).
* **Phụ thuộc hàm**:
  + room\_id → room\_type, room\_price, hotel\_id
  + hotel\_id → hotel\_city
  + guest\_id → guest\_phone  
    **Yêu cầu**:

1. Chỉ ra vi phạm 1NF (nếu có).
2. Chuẩn hóa lần lượt đến 3NF, giải thích từng bước.

**Không vi phạm 1NF** → Dữ liệu là nguyên tử.

**Vi phạm 2NF** do các thuộc tính phụ thuộc **một phần khóa chính** (room\_id, guest\_id).

**Chuẩn hoá 2NF:**

**HotelBookings**(booking\_id(PK), guest\_id(PK) , room\_id(PK) , check\_in, check\_out)

**Room**(room\_id(PK), room\_type, room\_price, hotel\_id)

**Hotel**(hotel\_id(PK), hotel\_city)

**Guest**(guest\_id(PK), guest\_phone)

Room: room\_id → hotel\_id, hotel\_id → hotel\_city  
⇒ **Phụ thuộc bắc cầu**: room\_id → hotel\_city

**Chuẩn hoá 3NF:**

**HotelBookings**(booking\_id(PK), guest\_id(FK), room\_id(FK), check\_in(PK), check\_out)

**Guest**(guest\_id(PK) , guest\_phone)

**Room**(room\_id(PK), room\_type, room\_price, hotel\_id(FK))

**Hotel**(hotel\_id(PK), hotel\_city)

### **Bài 2: Quản lý đơn hàng với thuộc tính đa giá trị**

Bảng Orders:

sql

Copy

(order\_id, customer\_id, items, order\_date)

* items là JSON chứa danh sách sản phẩm:
* json
* Copy
* [{"product\_id": 1, "name": "Laptop", "price": 1000, "quantity": 2}, ...]

**Yêu cầu**:

1. Đưa về 1NF bằng cách khử thuộc tính đa giá trị.
2. Phân tích phụ thuộc hàm và áp dụng 2NF, 3NF.

**Vi phạm:**

* items chứa danh sách nhiều sản phẩm ⇒ **thuộc tính đa giá trị** ⇒ vi phạm 1NF.

Tách Order thành 2 bảng:

**Orders**(order\_id(PK), customer\_id, order\_date)

**OrderItems**(order\_id(PK), product\_id(PK), name, price, quantity)

product\_id → name, price ❌ ⇒ thuộc tính **phụ thuộc vào một phần khóa**

order\_id, product\_id → quantity ✅ ⇒ phụ thuộc đầy đủ vào khóa

**OrderItems**(order\_id(PK), product\_id(PK) , quantity)

**Product**(product\_id(PK), name, price)

3NF:

* Product: product\_id → name, price ✅ không có phụ thuộc bắc cầu
* OrderItems: (order\_id, product\_id) → quantity ✅
* Orders: order\_id → customer\_id, order\_date ✅

⏹️ **Tất cả phụ thuộc đều trực tiếp vào khóa chính**  
→ **Đã thỏa mãn 3NF**

### **Bài 3: Hệ thống giáo dục (quan hệ phức tạp)**

Bảng StudentCourses:

sql

Copy

(student\_id, student\_name, department\_id, department\_head, course\_id, course\_name, instructor\_id, instructor\_email, grade)

* **Khóa chính**: (student\_id, course\_id).
* **Phụ thuộc hàm**:
  + student\_id → student\_name, department\_id
  + department\_id → department\_head
  + course\_id → course\_name, instructor\_id
  + instructor\_id → instructor\_email  
    **Yêu cầu**:

1. Chứng minh vi phạm 2NF/3NF.
2. Thiết kế lại đạt 3NF.

**vi phạm 2NF:**

* student\_id → student\_name, department\_id ❌ phụ thuộc **một phần khóa**
* course\_id → course\_name, instructor\_id ❌ phụ thuộc **một phần khóa**
* department\_id → department\_head ❌ không phụ thuộc vào khóa chính, mà vào một non-key → xét ở 3NF
* instructor\_id → instructor\_email ❌ tương tự, xét ở 3NF
* student\_id → department\_id, department\_id → department\_head  
  ⇒ student\_id → department\_head ❌
* course\_id → instructor\_id, instructor\_id → instructor\_email  
  ⇒ course\_id → instructor\_email ❌

⛔ **Kết luận**: Vi phạm **3NF** do có **phụ thuộc bắc cầu**.

**StudentCourses**(student\_id(PK), course\_id(PK), grade)

**Student**(student\_id(PK), student\_name, department\_id(FK))

**Department**(department\_id(PK), department\_head)

Course(course\_id(PK), course\_name, instructor\_id(FK))

Instructor(instructor\_id(PK), instructor\_email)

### **Bài 4: Hệ thống IoT (dữ liệu phi cấu trúc)**

Bảng SensorReadings:

sql

Copy

(sensor\_id, timestamps, values, location\_id, location\_zone)

* timestamps: Chuỗi phân cách bằng dấu phẩy (ví dụ: "2023-01-01 12:00, 2023-01-01 12:05").
* values: Chuỗi giá trị tương ứng (ví dụ: "23.5, 24.1").  
  **Yêu cầu**:

1. Áp dụng 1NF bằng cách tách timestamps và values.
2. Giả sử location\_id → location\_zone, chuẩn hóa tiếp đến 3NF.

Tách **timestamps** và **values** thành các hàng riêng biệt, mỗi hàng tương ứng với một cặp giá trị (timestamp, value) cho cùng một sensor\_id, location\_id, và location\_zone.

SensorReadings\_1NF(sensor\_id, timestamp, value, location\_id, location\_zone)

**để đạt 2NF:**

SensorLocations(sensor\_id, location\_id)

SensorReadings\_2NF(sensor\_id, timestamp, value)

**để đạt 3NF**:

**SensorReadings\_3NF**(sensor\_id, timestamp, value)

Khóa chính: (sensor\_id, timestamp)

**SensorLocations**(sensor\_id, location\_id)

Khóa chính: sensor\_id

Khóa ngoại: location\_id tham chiếu đến Locations(location\_id)

**Locations**(location\_id, location\_zone)

Khóa chính: location\_id

### **Bài 5: Quản lý dự án với phụ thuộc bắc cầu**

Bảng ProjectTasks:

sql

Copy

(project\_id, project\_name, client\_id, client\_industry, task\_id, task\_description, employee\_id, employee\_department)

* **Khóa chính**: (project\_id, task\_id).
* **Phụ thuộc hàm**:
  + project\_id → project\_name, client\_id
  + client\_id → client\_industry
  + task\_id → task\_description
  + employee\_id → employee\_department  
    **Yêu cầu**:

1. Chỉ ra phụ thuộc bắc cầu vi phạm 3NF.
2. Chuẩn hóa triệt để.

* project\_id → client\_id → client\_industry ⛔
* employee\_id → employee\_department (nếu không phải khóa chính, thì đây cũng vi phạm 3NF) ⛔

→ Đây là **phụ thuộc bắc cầu không qua khóa chính đầy đủ**, nên **vi phạm 3NF**.

Projects(

project\_id,

project\_name,

client\_id,

PRIMARY KEY (project\_id),

FOREIGN KEY (client\_id) REFERENCES Clients(client\_id)

)

Clients(

client\_id,

client\_industry,

PRIMARY KEY (client\_id)

)

Employees(

employee\_id,

employee\_department,

PRIMARY KEY (employee\_id)

)

ProjectTasks(

project\_id,

task\_id,

employee\_id,

PRIMARY KEY (project\_id, task\_id),

FOREIGN KEY (project\_id) REFERENCES Projects(project\_id),

FOREIGN KEY (task\_id) REFERENCES Tasks(task\_id),

FOREIGN KEY (employee\_id) REFERENCES Employees(employee\_id)

)

### **Bài 6: Cửa hàng bán lẻ (khóa phức hợp)**

Bảng RetailSales:

sql

Copy

(store\_id, product\_id, product\_category, supplier\_id, supplier\_region, sale\_date, quantity, price)

* **Khóa chính**: (store\_id, product\_id, sale\_date).
* **Phụ thuộc hàm**:
  + product\_id → product\_category, supplier\_id
  + supplier\_id → supplier\_region  
    **Yêu cầu**:

1. Tại sao bảng này chưa đạt 2NF?
2. Đưa về 3NF và vẽ sơ đồ quan hệ.

* product\_category và supplier\_id **chỉ phụ thuộc vào product\_id** (một phần của khóa chính)
* supplier\_region phụ thuộc bắc cầu qua supplier\_id

→ Các thuộc tính **không phụ thuộc vào toàn bộ khóa chính** → **vi phạm 2NF**

RetailSales(

store\_id,

product\_id,

sale\_date,

quantity,

price,

PRIMARY KEY (store\_id, product\_id, sale\_date),

FOREIGN KEY (product\_id) REFERENCES Products(product\_id)

)

Products(

product\_id,

product\_category,

supplier\_id,

PRIMARY KEY (product\_id),

FOREIGN KEY (supplier\_id) REFERENCES Suppliers(supplier\_id)

)

Suppliers(

supplier\_id,

supplier\_region,

PRIMARY KEY (supplier\_id)

)

### **Bài 7: Hệ thống y tế (dữ liệu lồng nhau)**

Bảng PatientRecords:

sql

Copy

(patient\_id, patient\_name, visits)

* visits là XML:
* xml
* Copy

<visits>

<visit date="2023-01-01" doctor\_id="D1" diagnosis="Fever"/>

<visit date="2023-02-01" doctor\_id="D2" diagnosis="Flu"/>

* </visits>
* Run HTML

**Yêu cầu**:

1. Khử 1NF bằng cách "làm phẳng" XML.
2. Giả sử doctor\_id → doctor\_name, áp dụng 3NF.

Cột visits chứa **danh sách lồng nhau** (XML), vi phạm nguyên tắc **"mỗi ô chỉ chứa 1 giá trị nguyên tử"**.

Patients(

patient\_id,

patient\_name,

PRIMARY KEY (patient\_id)

)

Visits(

patient\_id,

visit\_date,

doctor\_id,

diagnosis,

PRIMARY KEY (patient\_id, visit\_date),

FOREIGN KEY (patient\_id) REFERENCES Patients(patient\_id),

FOREIGN KEY (doctor\_id) REFERENCES Doctors(doctor\_id)

)

Doctors(

doctor\_id,

doctor\_name,

PRIMARY KEY (doctor\_id)

)

### **Bài 8: Hệ thống đấu giá (phụ thuộc đa trị)**

Bảng AuctionBids:

sql

Copy

(auction\_id, item\_name, bidder\_ids, bid\_amounts, current\_winner\_id)

* bidder\_ids và bid\_amounts là mảng (ví dụ: "B001,B002", "100,150").  
  **Yêu cầu**:

1. Thiết kế lại để đạt 1NF.
2. Phân tích phụ thuộc hàm để đạt 3NF.

Auctions(

auction\_id,

item\_name,

current\_winner\_id,

PRIMARY KEY (auction\_id),

FOREIGN KEY (current\_winner\_id) REFERENCES Bidders(bidder\_id)

)

Bidders(

bidder\_id,

bidder\_name, -- tùy chọn

...

PRIMARY KEY (bidder\_id)

)

Bids(

auction\_id,

bidder\_id,

bid\_amount,

PRIMARY KEY (auction\_id, bidder\_id),

FOREIGN KEY (auction\_id) REFERENCES Auctions(auction\_id),

FOREIGN KEY (bidder\_id) REFERENCES Bidders(bidder\_id)

)

 auction\_id → item\_name, current\_winner\_id

 bidder\_id không phụ thuộc vào auction\_id hay item\_name, nên tách ra bảng riêng

 Nếu sau này có: bidder\_id → bidder\_name → tách Bidders là hợp lý.

### **Bài 9: Quản lý kho (thuộc tính tính toán)**

Bảng Inventory:

sql

Copy

(product\_id, product\_name, warehouse\_id, warehouse\_location, current\_stock, reorder\_level, last\_restock\_date)

* **Khóa chính**: (product\_id, warehouse\_id).
* **Phụ thuộc hàm**:
  + product\_id → product\_name
  + warehouse\_id → warehouse\_location
  + reorder\_level phụ thuộc vào product\_id và warehouse\_id.  
    **Yêu cầu**:

1. Giải thích vi phạm 2NF/3NF.
2. Chuẩn hóa và đưa reorder\_level vào bảng phù hợp.

**Vi phạm 2NF:**

* Bảng có **khóa chính phức hợp** (product\_id, warehouse\_id), nên:
  + Nếu thuộc tính **chỉ phụ thuộc vào 1 phần** của khóa → **vi phạm 2NF**.

**📌 Vi phạm 3NF (nếu có):**

* Không có phụ thuộc **bắc cầu** hoặc phụ thuộc vào **thuộc tính không khóa** → **không vi phạm 3NF sau khi xử lý 2NF**.

Products(

product\_id PRIMARY KEY,

product\_name

)

Warehouses(

warehouse\_id PRIMARY KEY,

warehouse\_location

)

InventoryStock(

product\_id,

warehouse\_id,

current\_stock,

reorder\_level,

last\_restock\_date,

PRIMARY KEY (product\_id, warehouse\_id),

FOREIGN KEY (product\_id) REFERENCES Products(product\_id),

FOREIGN KEY (warehouse\_id) REFERENCES Warehouses(warehouse\_id)

)

### **Bài 10: Mạng xã hội (quan hệ đệ quy)**

Bảng UserPosts:

sql

Copy

(post\_id, user\_id, user\_name, post\_content, parent\_post\_id, parent\_post\_user)

* **Khóa chính**: post\_id.
* **Phụ thuộc hàm**:
  + user\_id → user\_name
  + parent\_post\_id → parent\_post\_user  
    **Yêu cầu**:

1. Phân tích phụ thuộc hàm và chuẩn hóa đến 3NF.
2. Xử lý quan hệ đệ quy (self-referencing) thế nào?

**Phụ thuộc hàm:**

* user\_id → user\_name (Tên người dùng phụ thuộc vào user\_id)
* parent\_post\_id → parent\_post\_user (Người dùng của bài viết gốc phụ thuộc vào parent\_post\_id)

**Vi phạm 2NF:**

* Có **phụ thuộc không đầy đủ** (chỉ phụ thuộc vào một phần khóa chính):
  + user\_name phụ thuộc vào user\_id, không phụ thuộc vào toàn bộ khóa chính.
  + parent\_post\_user phụ thuộc vào parent\_post\_id, không phải phụ thuộc vào khóa chính post\_id.

**Vi phạm 3NF:**

* Không có phụ thuộc bắc cầu, vì các phụ thuộc đều dựa vào các khóa chính hoặc khóa ngoại trực tiếp.

Users(

user\_id PRIMARY KEY,

user\_name

)

UserPosts(

post\_id PRIMARY KEY,

user\_id,

post\_content,

parent\_post\_id, -- Làm mối quan hệ đệ quy

FOREIGN KEY (user\_id) REFERENCES Users(user\_id),

FOREIGN KEY (parent\_post\_id) REFERENCES UserPosts(post\_id) -- Quan hệ đệ quy (self-referencing)

)

 **Quan hệ đệ quy** xảy ra khi một bảng có **khóa ngoại tham chiếu lại chính nó**.

 Trong trường hợp này, parent\_post\_id là khóa ngoại tham chiếu tới post\_id trong cùng bảng UserPosts. Điều này cho phép bài viết có thể là một **bài viết gốc** hoặc một **phản hồi** tới bài viết khác.