Tuần 2 - Tổng hợp kiến thức Buổi học số 3

Time-Series Team

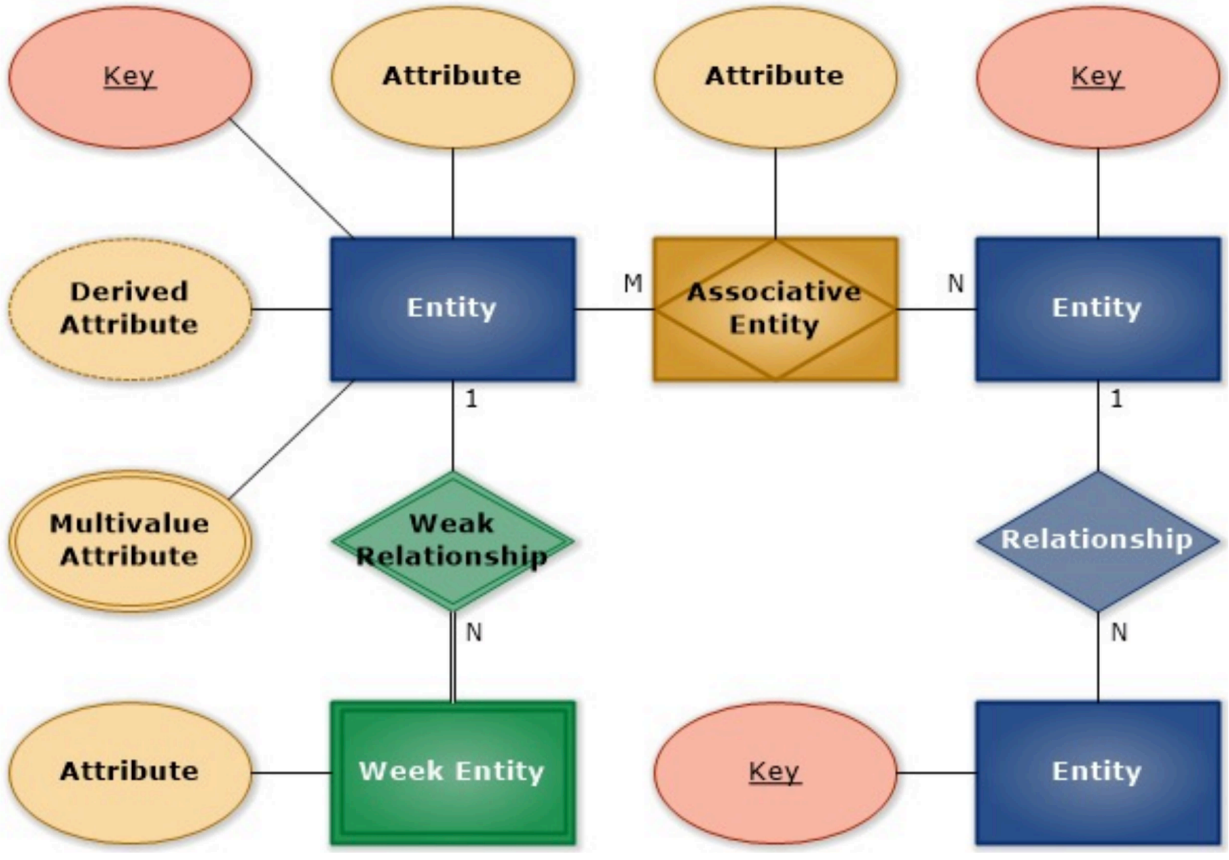
Ngày 16 tháng 6 năm 2025

Buổi học số 3 (Thứ 5, 12/06/2025) bao gồm hai nội dung chính:

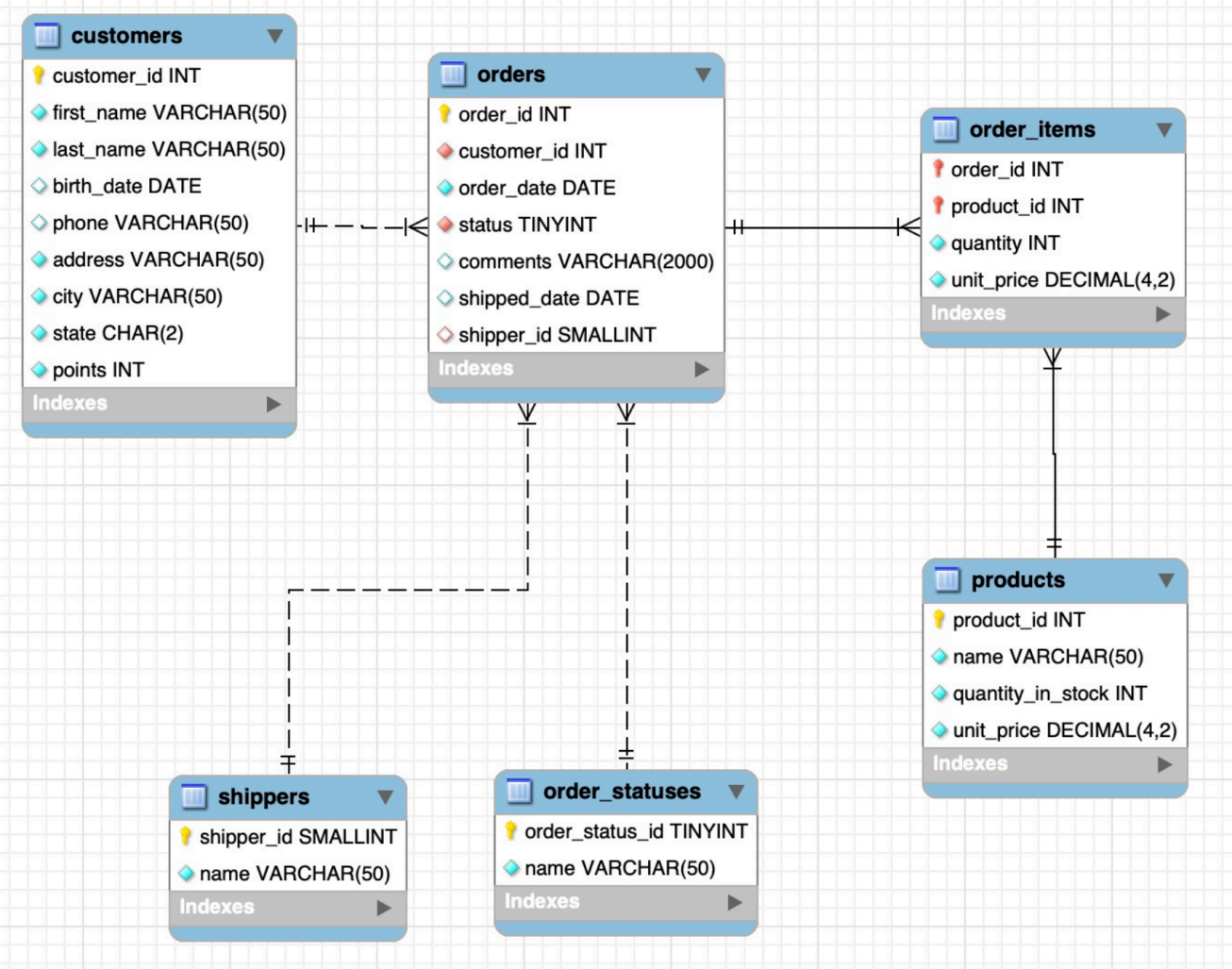
- ***Phần I: Sơ đồ Thực thể–Quan hệ (Entity–Relationship Diagram (ERD))***
- ***Phần II: Chuẩn hoá (Database Normalization)***

Phần I: Sơ đồ Thực thể–Quan hệ (Entity–Relationship Diagram (ERD))

Định nghĩa: ERD là công cụ đồ họa giúp chúng ta hình dung cấu trúc logic của cơ sở dữ liệu trước khi triển khai thực tế.



Hình 1: Sơ đồ Quan hệ dưới góc nhìn Chen's Notation



Hình 2: Sơ đồ Quan hệ dưới góc nhìn Crow's Foot Notation

1 Các thành phần chính

1.1 Theo Chen's Notation

Trong Chen's Notation, sơ đồ ERD bao gồm:

- **Thực thể (Entity)**: Các thực thể được biểu diễn trong sơ đồ ER bằng một hình chữ nhật và được đặt tên bằng danh từ số ít, ví dụ `Order`.

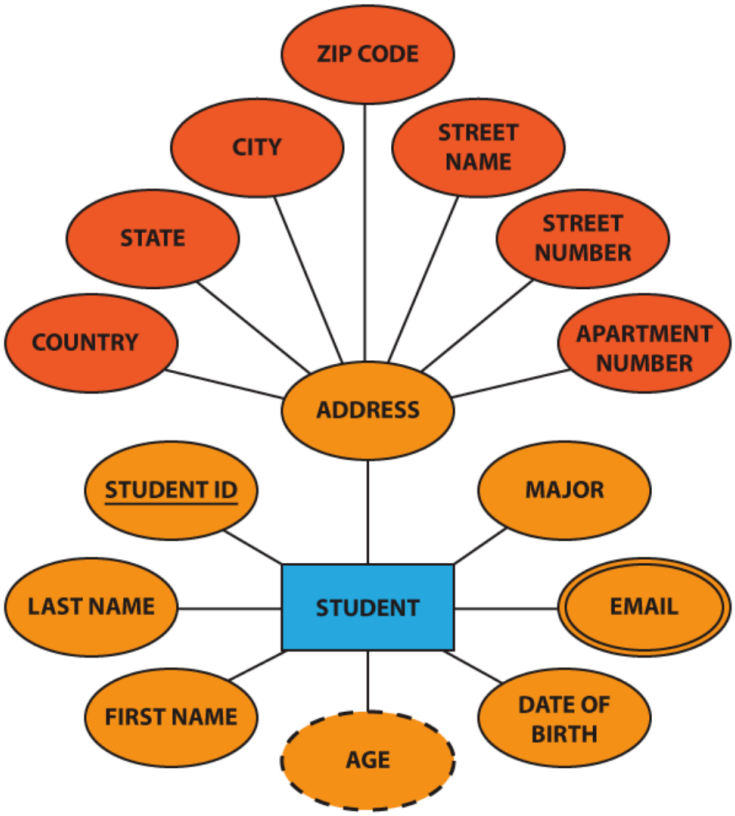
Hình 3: Thực thể dưới góc nhìn Chen's Notation

- **Thực thể yếu (Weak Entity):** Một thực thể yếu là một thực thể phụ thuộc vào sự tồn tại của một thực thể khác.

  Một thực thể không thể được xác định duy nhất chỉ bằng các thuộc tính của nó.

Hình 4: Thực thể yếu dưới góc nhìn Chen's Notation

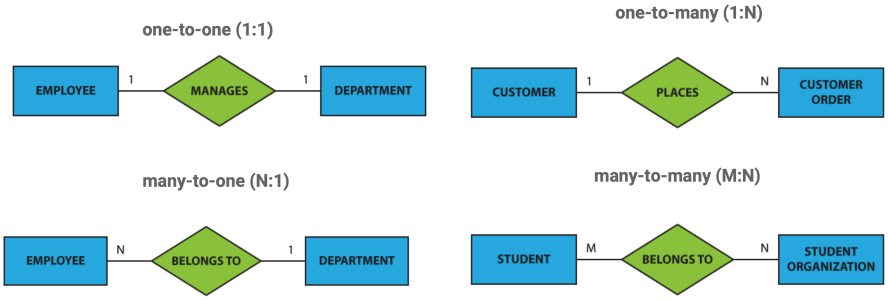
- **Thuộc tính (Attribute)**: hình oval nối với thực thể.



Hình 5: Thuộc tính dưới góc nhìn Chen's Notation

  - *Primary Key*: gạch chân, ví dụ CustomerID.
  - *Foreign Key*: in nghiêng hoặc kèm chú thích (FK).
  - *Multivalued Attribute*: hai vòng oval chồng nhau cho thuộc tính có nhiều giá trị.
  - *Derived Attribute*: thuộc tính suy ra, vẽ oval nét đứt.

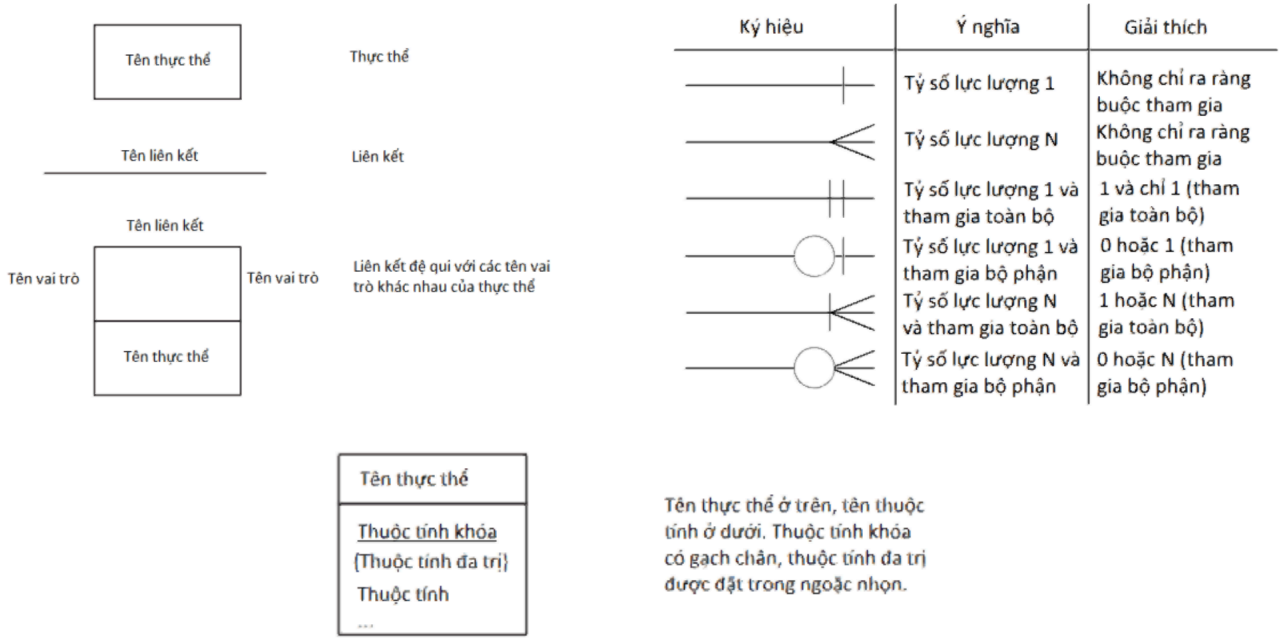
- **Quan hệ (Relationship)**: hình thoi, nối các thực thể, tên quan hệ để cạnh thoi.



Hình 6: Quan hệ dưới góc nhìn Chen's Notation

- **Cardinality**: ghi rõ 1:1, 1:N, M:N ngay bên đường nối.
- **Associative Entity**: với quan hệ M:N, sử dụng bảng trung gian (hình chữ nhật) chứa hai FK và thuộc tính bổ sung.

1.2 Theo Crow's Foot Notation



Crow's Foot Notation tối ưu cho sơ đồ lớn:

- **Thực thể (Entity)**: hộp chữ nhật liệt kê tên thực thể và các thuộc tính, khóa chính in đậm hoặc gạch chân, khóa ngoại in nghiêng.
- **Thuộc tính**: liệt kê trong hộp, không dùng oval.
- **Cardinality**: ký hiệu ngay đầu nối:

Ký hiệu	Ý nghĩa
$--\|$	Một (1)
$--\|\|$	Chính xác một (bắt buộc)
$--\circ$	Không hoặc một (0..1)
$--<$	Nhiều (N)
$--\circ<$	Không hoặc nhiều (0..N)
$--\|\|<$	Ít nhất một nhiều (1..N)

Giả sử có hai thực thể `A` và `B` với quan hệ sau:

$$\text{A} \quad --| \quad \longleftrightarrow \quad < -- \quad \text{B}$$

- Ký hiệu | bên A nghĩa mỗi bản ghi bên B chỉ liên kết với đúng 1 bản ghi A.
- Ký hiệu '<' bên B nghĩa mỗi bản ghi bên A có thể liên kết với nhiều bản ghi B.
- Thêm ∘ nếu tuỳ chọn (có thể không có liên kết), thêm ‖ nếu bắt buộc ít nhất một.

1–N bắt buộc–bắt buộc (‖–<) Mỗi bản ghi bên "1" phải có ít nhất một bản ghi bên "N", và mỗi bản ghi bên "N" chỉ thuộc một bên "1". *Ví dụ:* Hoá đơn phải có ít nhất một mục hàng; mỗi mục hàng chỉ gắn với một hoá đơn.

1–N tuỳ chọn–bắt buộc (∘–<) Bên "1" có thể không có liên kết, bên "N" vẫn phải liên kết với đúng một "1". *Ví dụ:* Khách hàng có thể chưa đặt đơn nào (0..N), nhưng mỗi đơn luôn phải gắn với một khách hàng.

1–1 bắt buộc–tuỳ chọn (‖–∘) Mỗi bản ghi bên bắt buộc phải có đúng một đối tác, còn bên tuỳ chọn có thể không có. *Ví dụ:* Nhân viên luôn thuộc một phòng ban; nhưng một phòng ban có thể chưa có nhân viên.

1–1 bắt buộc–bắt buộc (‖–‖) Mối quan hệ một–một chặt chẽ hai chiều. *Ví dụ:* Mỗi hồ sơ lương chỉ gắn với một nhân viên và mỗi nhân viên có đúng một hồ sơ lương.

N–M (<–<) Mỗi bên có thể liên kết nhiều với bên kia; thường hiện thực qua bảng trung gian. *Ví dụ:* Sinh viên – Khóa học: một sinh viên có thể học nhiều khóa, và một khóa có nhiều sinh viên.

- **Bảng trung gian (Associative Entity)**: quan hệ M:N được giải bằng bảng phụ với hai chân quạ nối tới hai bảng cha.

Phần II: Chuẩn hoá (Database Normalization)

1 Mục tiêu của chuẩn hóa

Chuẩn hóa nhằm:

- Loại bỏ dư thừa dữ liệu.
- Ngăn ngừa các dị thường (anomalies) khi chèn, cập nhật, xóa.
- Đảm bảo tính nhất quán, toàn vẹn dữ liệu.

2 Kiến thức bổ trợ

Một số khái niệm cơ bản trước khi vào các dạng chuẩn:

Phụ thuộc hàm đầy đủ (Full Functional Dependency)

Gọi $X \to Y$ là phụ thuộc hàm đầy đủ nếu Y phụ thuộc vào toàn bộ tập X, không phụ thuộc chỉ một phần của X.

Phụ thuộc hàm bộ phận (Partial Dependency)

Khi một thuộc tính phụ thuộc chỉ vào một phần của khóa chính (trong trường hợp khóa chính ghép), tức là tồn tại phụ thuộc hàm bộ phận.

Phụ thuộc hàm bắc cầu (Transitive Dependency)

Nếu $A \to B$ và $B \to C$, nhưng A không trực tiếp xác định C, ta nói có phụ thuộc bắc cầu.

Phụ thuộc hàm không tầm thường (Non-trivial Functional Dependency)

Cho một quan hệ R với các tập thuộc tính X và $Y \subseteq R$. Một phụ thuộc hàm $X \to Y$ được gọi là *không tầm thường* (non-trivial) nếu và chỉ nếu

$$Y \not\subseteq X.$$

Nghĩa là, có ít nhất một thuộc tính trong Y không nằm trong X.

3 Các dị thường (Anomalies)

- **Insertion Anomaly**: Không thể chèn bản ghi nếu thiếu thông tin phụ thuộc (ví dụ không thể thêm một đơn hàng khi chưa có khách hàng).
- **Update Anomaly**: Phải cập nhật cùng một thông tin ở nhiều bản ghi khác nhau, dẫn đến nguy cơ không đồng nhất.
- **Delete Anomaly**: Xóa một bản ghi có thể vô tình xóa luôn thông tin cần giữ ở các bản ghi khác.

4 Các dạng chuẩn hóa

4.1 First Normal Form (1NF)

Điều kiện để đạt 1NF:

- Mỗi ô trong bảng chỉ chứa một giá trị nguyên tử (atomic), không có danh sách hay nhóm lặp.
- Mỗi hàng là duy nhất, thường thông qua một khóa chính.
- Không có các cột lặp hoặc nhóm cột lặp cho cùng một loại dữ liệu.

Giải thích:
Nếu một ô chứa nhiều giá trị (ví dụ "Math, Physics") hoặc các cột lặp như `Phone1, Phone2`, thì không thể truy vấn, lọc hay đếm riêng từng phần tử. 1NF yêu cầu tách mỗi giá trị thành một ô riêng và bỏ các nhóm cột lặp.

Ví dụ: (Chưa 1NF)

StudentID	Name	Subjects
S01	Alice	Math, Physics, History
S02	Bob	Physics, Chemistry

Anomalies: Không thể đếm số sinh viên học "Physics" hoặc tìm sinh viên chỉ học một môn.

Chuyển sang 1NF:

- Tách mỗi môn thành một hàng mới, để mỗi ô chỉ chứa một môn.

Sau khi đạt 1NF:

StudentID	Name	Subject
S01	Alice	Math
S01	Alice	Physics
S01	Alice	History
S02	Bob	Physics
S02	Bob	Chemistry

—

4.2 Second Normal Form (2NF)

Điều kiện để đạt 2NF:

- Quan hệ đã ở 1NF.
- Mọi thuộc tính không khóa phải phụ thuộc *đầy đủ* vào toàn bộ khóa chính (không có phụ thuộc bộ phận).

Giải thích:
Nếu khóa chính là composite (ví dụ (A, B)) nhưng một cột chỉ phụ thuộc vào một phần (A hoặc B), sẽ lặp lại thông tin đó nhiều lần. 2NF tách các phụ thuộc bộ phận ra thành bảng riêng.

Ví dụ: (1NF nhưng chưa 2NF)

OrderID	ProductID	OrderDate	Quantity
1001	A123	2025-06-14	2
1001	B456	2025-06-14	1

Anomalies: `OrderDate` lặp cho mỗi sản phẩm trong cùng đơn hàng; khó cập nhật ngày duy nhất.

Chuyển sang 2NF:

1. Tách `Orders`(`OrderID`, `OrderDate`).
2. Giữ `OrderItems`(`OrderID`, `ProductID`, `Quantity`).

Sau khi đạt 2NF:

Orders

OrderID	OrderDate
1001	2025-06-14

OrderItems

OrderID	ProductID	Quantity
1001	A123	2
1001	B456	1

—

4.3 Third Normal Form (3NF)

Điều kiện để đạt 3NF:

- Quan hệ đã ở 2NF.
- Không có phụ thuộc bắc cầu: mọi thuộc tính không khóa chỉ phụ thuộc trực tiếp vào khóa chính, không phụ thuộc qua thuộc tính không khóa khác.

Giải thích:
Nếu $A \to B$ và $B \to C$, nhưng C không trực tiếp phụ thuộc vào A, thì C bị lặp khi B bị lặp. 3NF tách phụ thuộc bắc cầu ra thành bảng riêng.

Ví dụ: (2NF nhưng chưa 3NF)

CustomerID	**Name**	**ZipCode**	**City**
C01	Alice	10001	New York
C02	Bob	10002	New York

Anomalies: `City` lặp cho mỗi `ZipCode`; cập nhật thành phố phải sửa nhiều hàng.

Chuyển sang 3NF:

1. Tách `ZipCodes`(`ZipCode`, `City`).
2. Giữ `Customers`(`CustomerID`, `Name`, `ZipCode`).

Sau khi đạt 3NF:

ZipCodes

ZipCode	City
10001	New York
10002	New York

Customers

CustomerID	Name	ZipCode
C01	Alice	10001
C02	Bob	10002

4.4 Boyce–Codd Normal Form (BCNF)

Điều kiện để đạt BCNF:

- Quan hệ đã ở dạng 3NF.
- Với mọi phụ thuộc hàm non-trivial $X \to Y$, tập X phải là siêu khóa của quan hệ.

Giải thích:
Nếu có một phụ thuộc $X \to Y$ mà X không đủ duy nhất để xác định hàng (không phải siêu khóa), thì khi giá trị Y thay đổi bạn sẽ phải cập nhật nhiều hàng, dẫn đến bất thường cập nhật.

Ví dụ: (3NF nhưng chưa BCNF)

StudentID	CourseID	Instructor
S1	C1	Prof. A
S2	C1	Prof. A

$\{\texttt{Instructor}\} \rightarrow \{\texttt{CourseID}\}$ nhưng $\{\texttt{Instructor}\}$ không phải siêu khóa.

Phân rã sang BCNF:

1. `InstructorCourses`(`Instructor`, `CourseID`)
2. `StudentInstructors`(`StudentID`, `Instructor`)

Sau khi đạt BCNF:

InstructorCourses

Instructor	CourseID
Prof. A	C1

StudentInstructors

StudentID	Instructor
S1	Prof. A
S2	Prof. A

Lợi ích: Mọi FD non-trivial giờ đều có vế trái là siêu khóa, loại bỏ anomalies cập nhật tên giảng viên.

4.5 Fourth Normal Form (4NF)

Điều kiện để đạt 4NF:

- Quan hệ đã ở dạng BCNF.
- Không tồn tại bất kỳ phụ thuộc đa trị non-trivial $X \twoheadrightarrow Y$ nào với X không phải siêu khóa.

Giải thích:
Khi X đa xác định Y và Z độc lập, bạn sẽ phải lưu Cartesian product $Y \times Z$, gây dư thừa. Nếu X không phải siêu khóa, phải tách MVD này ra thành bảng riêng.

Ví dụ: (BCNF nhưng chưa 4NF)

EmpID	Skill	ProjectID
E1	Java	P1
E1	Java	P2
E1	Python	P1
E1	Python	P2

Có các MVD:

$$\texttt{EmpID} \twoheadrightarrow \{\texttt{Java,Python}\}, \quad \texttt{EmpID} \twoheadrightarrow \{\texttt{P1,P2}\},$$

và $\{\texttt{EmpID}\}$ không phải siêu khóa.

Phân rã sang 4NF:

1. `EmployeeSkills`(`EmpID`, `Skill`)
2. `EmployeeProjects`(`EmpID`, `ProjectID`)

Sau khi đạt 4NF:

EmployeeSkills

EmpID	Skill
E1	Java
E1	Python

EmployeeProjects

EmpID	ProjectID
E1	P1
E1	P2

Lợi ích: Số hàng giảm từ $M \times N$ xuống $M + N$, và anomalies thêm/xóa kỹ năng hay dự án được giải quyết riêng biệt.

4.6 Fifth Normal Form (5NF / PJNF)

Điều kiện để đạt 5NF:

- Quan hệ đã ở dạng 4NF.
- Mọi join-dependency non-trivial trên quan hệ đều phải là hệ quả logic của các khóa ứng viên. Nếu tồn tại một join-dependency không được suy ra từ khóa ứng viên, quan hệ phải được phân rã thêm.

Giải thích:
Join-dependency tam phân nghĩa là bạn có thể phân rã quan hệ R thành ba quan hệ con R_1, R_2, R_3 sao cho

$$R = R_1 \bowtie R_2 \bowtie R_3$$

mà phép phân rã này không mất mát. Tuy nhiên, nếu bạn chỉ join hai trong ba quan hệ, có thể sinh ra các *tuple giả* (spurious tuples). 5NF yêu cầu mọi ràng buộc đa chiều như vậy phải được biểu diễn qua các phép chiếu từ khóa ứng viên, hoặc phải phân rã thêm để tránh tuple giả.

Ví dụ: Quan hệ chưa 5NF (đã ở 4NF)

Agent	**Company**	**Product**
A	X	P
A	Y	Q

Quy tắc nghiệp vụ: Nếu A làm việc cho X và X sản xuất P, thì A phải bán P. Đây là một join-dependency tam phân giữa (Agent, Company), (Company, Product), (Agent, Product). Nếu chỉ join hai trong ba, ví dụ join `AgentCompany` với `CompanyProduct`, ta sẽ thu được bộ (A, Y, P) – một tuple giả.

Phân rã để đạt 5NF:

1. `AgentCompany`(`Agent`, `Company`)
2. `CompanyProduct`(`Company`, `Product`)
3. `AgentProduct`(`Agent`, `Product`)

Sau khi đạt 5NF:

AgentCompany

Agent	Company
A	X
A	Y

CompanyProduct

Company	Product
X	P
Y	Q

AgentProduct

Agent	Product
A	P
A	Q

Kết quả: Chỉ khi join đầy đủ ba bảng con mới khôi phục lại quan hệ gốc R mà không sinh tuple giả, đảm bảo tính toàn vẹn của ràng buộc tam phân ngay trong cấu trúc cơ sở dữ liệu.

4.7 Ảnh hưởng của chuẩn hóa đến phân tích dữ liệu

Mặc dù chuẩn hóa mang lại cấu trúc chặt chẽ và nhất quán, nó cũng tạo ra một số thách thức khi phân tích dữ liệu:

- **Phân mảnh dữ liệu (Fragmentation):** Dữ liệu ban đầu bị tách thành nhiều bảng nhỏ, làm cho phép phân tích phải tổng hợp qua nhiều phép `JOIN`. Điều này không chỉ làm phức tạp logic truy vấn mà còn dễ gây sai lệch nếu sót bảng con nào.
- **Giảm khả năng truy vấn ad-hoc nhanh:** Các nhà phân tích thường cần hỏi nhanh bằng các truy vấn đơn giản. Khi dữ liệu đã highly-normalized, họ phải biết chính xác sơ đồ nhiều bảng, gây cản trở việc khám phá dữ liệu (data exploration).
- **Hiệu năng kém trên khối lượng lớn:** Các phép `JOIN` trên bảng lớn, đặc biệt trong các môi trường OLAP hay BI, có thể rất tốn tài nguyên và thời gian. Ta có thể thấy chậm trễ trong việc tính toán các chỉ số tổng hợp, báo cáo hoặc dashboard.
- **Khó tổng hợp và tính toán nhiều chiều:** Để tính các chỉ số như doanh thu theo khu vực, hay phân tích chuỗi thời gian, dữ liệu tách rời buộc phải xây dựng lại (reconstruct) từ nhiều bảng, dẫn đến mã SQL phức tạp và dễ lỗi.
- **Tính linh hoạt kém trong ETL / ELT:** Đối với pipelines dữ liệu, việc join nhiều bảng tạo độ trễ, khó xử lý incremental loads, và dễ gặp lỗi khi bảng con thay đổi schema.
- **Mất một số ngữ cảnh (Context):** Khi tách bảng, một số thông tin ngữ cảnh (ví dụ, thông tin liên kết tạm thời giữa các thuộc tính) có thể bị "đánh mất" nếu không lưu trữ lại, gây khó khăn khi phân tích các tình huống phức tạp hoặc truy vết nguyên nhân.
- **Đòi hỏi kiến thức về sơ đồ quan hệ:** Người phân tích phải hiểu rõ mô hình dữ liệu đã được chuẩn hóa để viết đúng câu lệnh truy vấn, dẫn đến chi phí đào tạo và onboarding cao hơn.