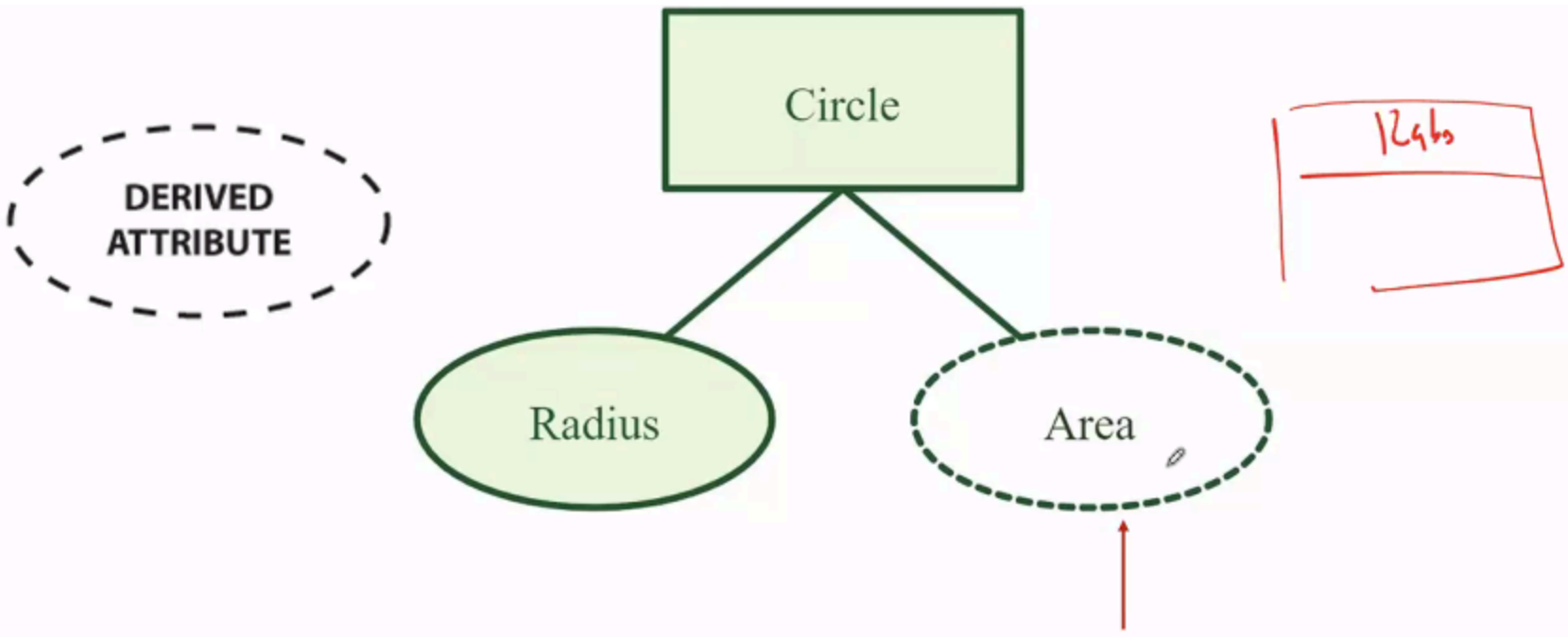
AIO2025 - SQL2_vn

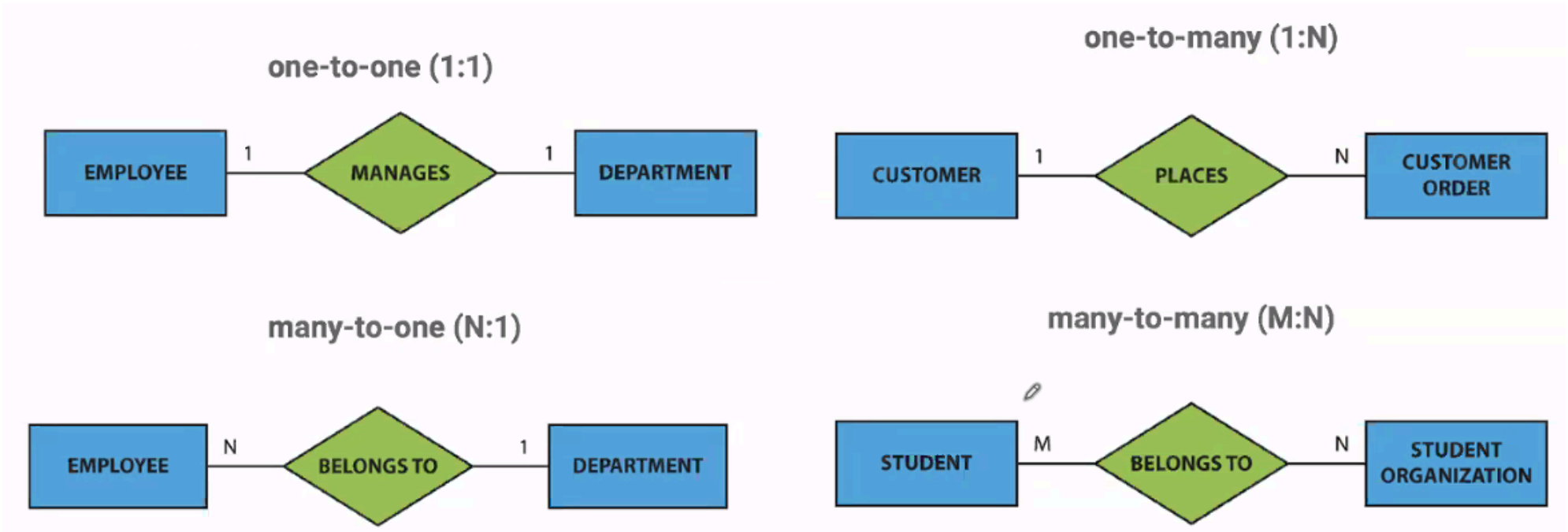
Giới thiệu về Entity Relationship Diagram

Derived Attribute: thuộc tính có thể được tính toán từ các thuộc tính khác.



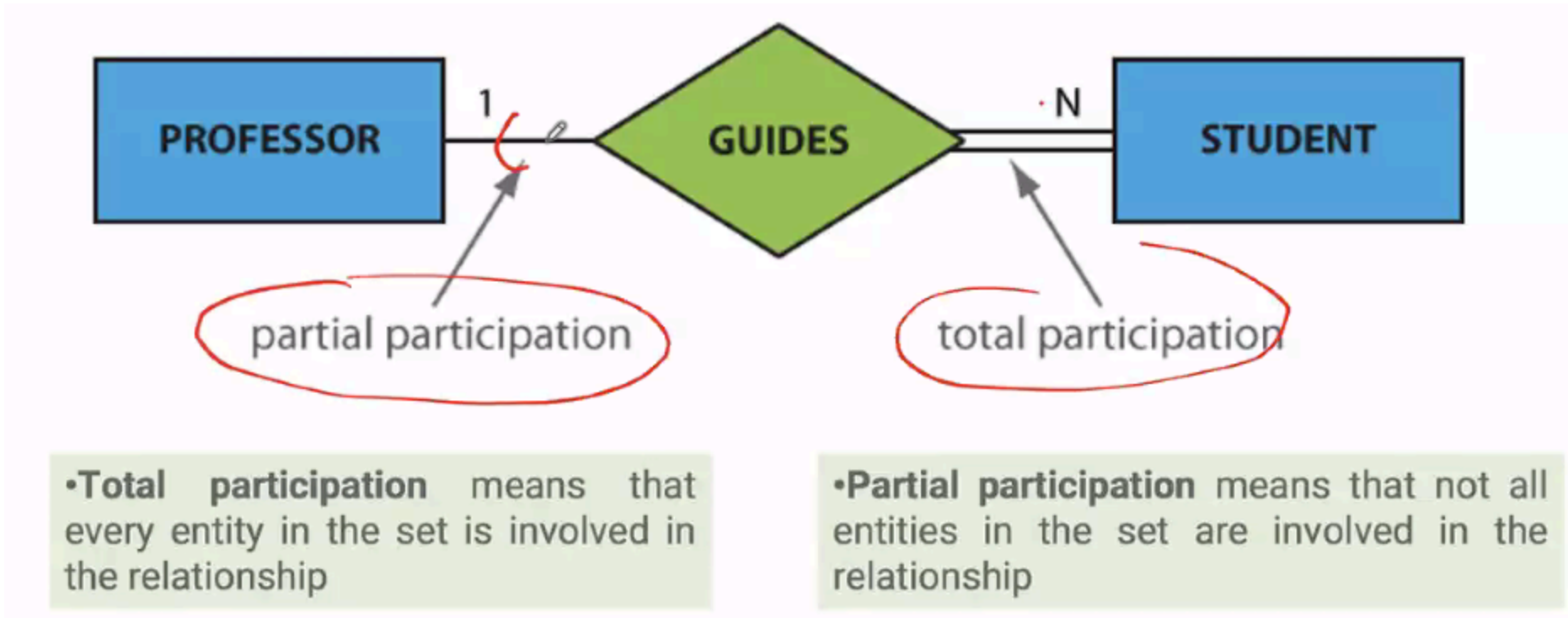
Composite Attributes (thuộc tính chứa nhiều sub-attribute) - một thuộc tính có kiểu giá trị là Object.
Multivalue attribute (thuộc tính chứa/nhận nhiều giá trị) - một list trong lập trình

Relationships

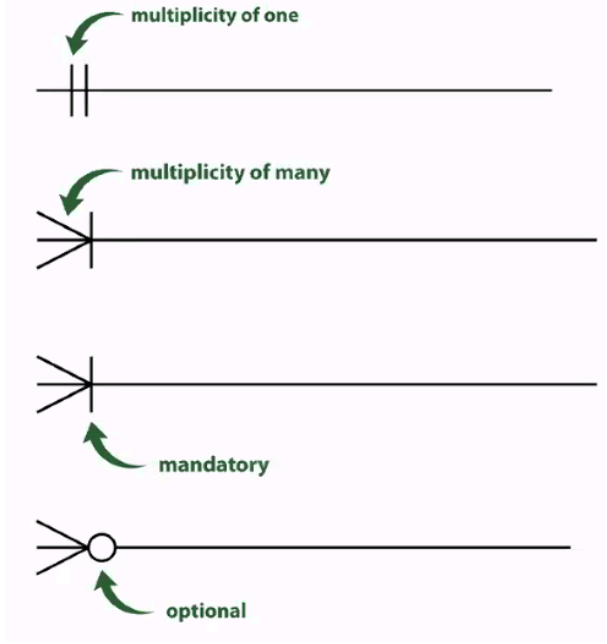


Participation Constraints

Partial participation: 1 phần (có hay không, không có cũng được)
Total participation: toàn phần (bắt buộc phải có)
-> Các sinh viên có thể không có giảng viên hướng dẫn.
-> Nhưng 1 giảng viên khi hướng dẫn chắc chắn sẽ hướng dẫn ít nhất 1 sinh viên.

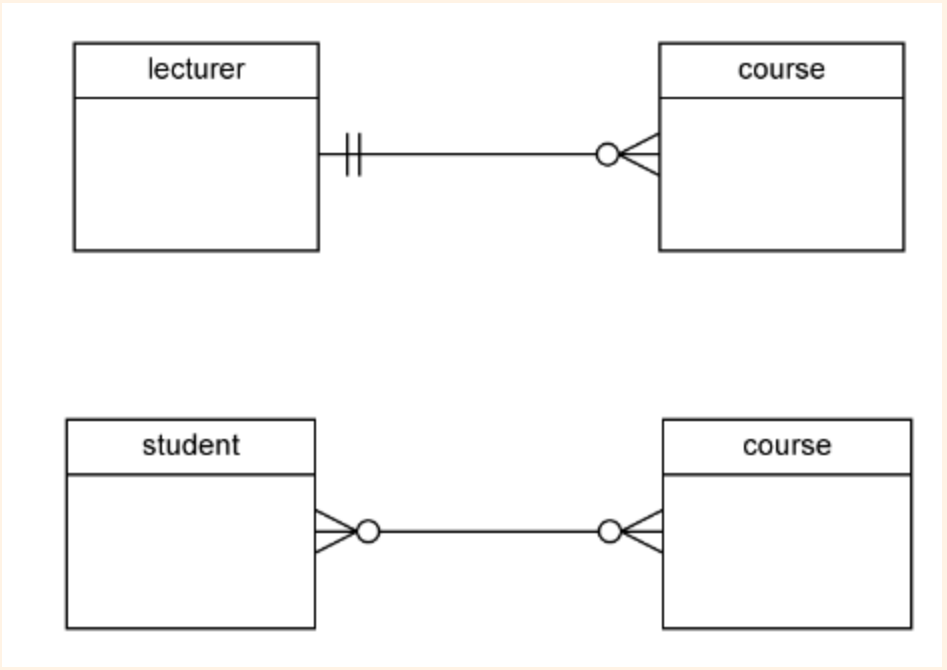


Crow's Foot Notation

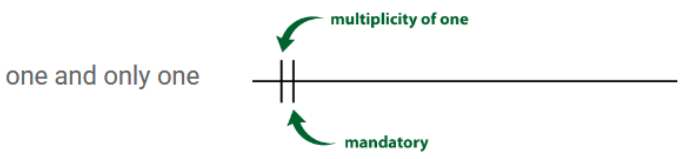


crow foot nghĩa là nhiều
single line thay cho crow's foot nghĩa là 1
circle before the foot nghĩa là optional (giống partial participation)
line before the foot nghĩa là mandatory (total participation)

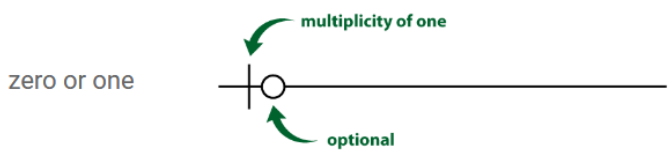
- ? 1 - một course phải có duy nhất 1 lecturer, và 1 lecturer có thể có 0 hoặc nhiều course.
- ? 2 - một student có thể đăng ký từ 0 đến nhiều course, và một course có thể có từ 0 đến nhiều student đã đăng ký.



Line thay cho foot nghĩa là one. One + Mandatory nghĩa là One and only One.



Line thay cho foot nghĩa là One. One + Optional nghĩa là Zero or One (Một hoặc Không có vì optional nghĩa là có hoặc không, hoặc có (1) hoặc không (1))



- ? ví dụ: Student có duy nhất 1 seat. 1 seat có duy nhất 1 student

Giới thiệu về Database Normalization

Database Normalization giúp giảm dư thừa dữ liệu và lỗi bất thường dữ liệu (ví dụ: thiếu dữ liệu, dữ liệu không đồng bộ giữa 2 bảng)

Data Normalization là hành động chia nhỏ các bảng lớn thành các bảng nhỏ hơn và liên kết chúng để tạo mối quan hệ giữa các bảng.

Vấn đề với bảng chưa được chuẩn hóa

StudentID	StudentName	CourseID	CourseName	Professor
1	Alice	101	Math	Dr. Smith
2	Bob	102	Science	Dr. Jones

Redundant Data

If a new student, Carol, needs to enroll in the Math course, you would have to insert the course information again, even though it already exists.

StudentID	StudentName	CourseID	CourseName	Professor
3	Carol	101	Math	Dr. Smith

Partial Data Insertion

If a new course needs to be added to the database but no students are enrolled yet, it's difficult to insert this data. For example, if a History course taught by Dr. Brown is introduced:

StudentID	StudentName	CourseID	CourseName	Professor
		103	History	Dr. Brown

Null Value Issues

If a new student, Dave, is admitted but hasn't yet enrolled in any course, inserting his data might require null values for CourseID, CourseName, and Professor.

StudentID	StudentName	CourseID	CourseName	Professor
4	Dave	NULL	NULL	NULL

Redundant Data Update: cập nhật của 1 người dẫn đến việc cập nhật nhiều dòng.
Inconsistent Data: nếu chỉ cập nhật tên 1 professor trong khi các dòng khác với cùng professor đó thì không.
Delete Anomalies: xóa student đồng nghĩa với việc xóa tất cả các cột của dòng đó, nếu course 101 nằm trong dòng bị xóa thì dữ liệu course cũng bị xóa theo dù ta chỉ muốn xóa student.

Các loại lỗi dữ liệu (trong bảng chưa được chuẩn hóa)

Students Table

StudentID	StudentName	CourseID	CourseName	Professor
1	Alice	101	Math	Dr. Smith
2	Bob	102	Science	Dr. Jones

Insertion Anomaly

- ? Xảy ra khi thuộc tính bị thiếu hoặc không đồng bộ trong quá trình chèn dữ liệu. Việc chèn dữ liệu trong bảng chưa chuẩn hóa có thể tạo ra các vấn đề sau:

Redundant Data: Chèn một student dẫn đến việc lặp lại giá trị 'CourseName' và 'Professor'. Bạn phải chèn lại các giá trị này dù chúng đã tồn tại trước đó.

Partial Data Insertion: Chèn một Course mới mà chưa có student đăng ký sẽ tạo ra các giá trị bị thiếu cho Student ID và Name (thường là giá trị NULL).

StudentID	StudentName	CourseID	CourseName	Professor
		103	History	Dr. Brown

Null Value Issues: Tương tự như Partial Data Insertion, chèn student chưa đăng ký course nào sẽ tạo ra giá trị null cho các cột Courses và Professor.

StudentID	StudentName	CourseID	CourseName	Professor
4	Dave	NULL	NULL	NULL

Updation anomalies

- ? Điều này xảy ra khi có nhiều dữ liệu bị lặp lại trong cùng một bảng dẫn đến việc xử lý không hiệu quả.

StudentID	StudentName	CourseID	CourseName	Professor
1	Alice	101	Math	Dr. Smith
2	Bob	102	Science	Dr. Jones
3	Carol	101	Math	Dr. Smith

Redundant Data Update: cập nhật 1 dòng đồng nghĩa với việc phải cập nhật nhiều dòng khác. Ví dụ: khi một Professor đổi tên thành Dr.Brown, tất cả các dòng có Dr.Smith cũng cần được cập nhật để đảm bảo tính nhất quán dữ liệu.

StudentID	StudentName	CourseID	CourseName	Professor
1	Alice	101	Math	Dr. Brown
2	Bob	102	Science	Dr. Jones
3	Carol	101	Math	Dr. Brown

Inconsistant Data: khi bạn cập nhật 1 dòng nhưng quên cập nhật các dòng khác. Ví dụ: khi cập nhật tên 1 Professor nhưng không đồng bộ phần còn lại.

StudentID	StudentName	CourseID	CourseName	Professor
1	Alice	101	Math	Dr. Brown
2	Bob	102	Science	Dr. Jones
3	Carol	101	Math	Dr. Smith

3. **Delete Anomalies:** Xóa một dữ liệu dẫn đến việc xóa ngoài ý muốn các dữ liệu khác.

StudentID	StudentName	CourseID	CourseName	Professor
1	Alice	101	Math	Dr. Smith
2	Bob	102	Science	Dr. Jones
3	Carol	101	Math	Dr. Smith

Unwanted Data Loss: Xóa dữ liệu Student cũng xóa dữ liệu Course & Professor.
Điều này có thể tạo ra *Orphan Records* nếu Course Name phụ thuộc vào CourseID.

For example, if Alice (StudentID 1) is deleted:

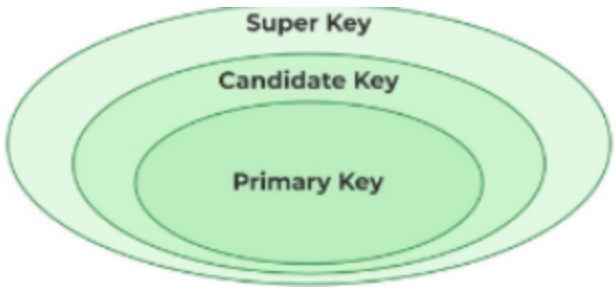
StudentID	StudentName	CourseID	CourseName	Professor
2	Bob	102	Science	Dr. Jones
3	Carol	101	Math	Dr. Smith

Thuật ngữ

Note: Thuộc tính là Tên của mỗi cột trong bảng. e..g StudentID, StudentName, CourseID, CourseName, Professor.

For example, if Alice (StudentID 1) is deleted:

StudentID	StudentName	CourseID	CourseName	Professor
2	Bob	102	Science	Dr. Jones
3	Carol	101	Math	Dr. Smith



Qua sơ đồ Venn, ta thấy có 3 lớp khóa SuperKey -> CandidateKey -> PrimaryKey, với mỗi tầng đại diện cho sự mở rộng của khóa.

- *Cách Giải thích 1:* Nghĩa là tầng 1 bao gồm 1 khóa, tầng hai bao gồm mọi khóa tầng 1 có thể chọn ra, tầng 3 bao gồm mọi khóa ở tầng 2 + các thuộc tính khác.
- *Cách Giải thích 2:* Tầng trên là 1 tổ hợp khóa mở rộng ra từ tổ hợp khóa ở tầng dưới)
- *SuperKey = Candidate Key + Các thuộc tính khác* có thể là khóa hoặc không phải là khóa. Có thể hiểu SuperKey là 1 list chứa mọi tổ hợp (i.e. khóa + khóa và thuộc tính) có thể dùng để lấy ra làm Khóa Chính sau này.

StudentID	Name	Age
SV01	An	20
SV02	Bình	21

  Ví dụ:
  - Nếu `StudentID` là duy nhất thì StudentID có thể là SuperKey.
  - `{StudentID, Name}, {StudentID, Age}, {StudentID, Name, Age}` cũng là super key (vì nó vẫn đảm bảo duy nhất dù có thừa).
    - `SuperKey = { {StudentID, Name}, {StudentID, Age}, {StudentID, Name, Age}, {StudentID} }`
- *Candidate Key = Chứa mọi thuộc tính có thể là khóa*.
  e.g. Candidate Key={StudentID, Email}
- *Primary Key* = Khóa Chính của 1 bảng, được chọn ra từ SuperKey hoặc Candidate Key. Với điều kiện khóa được chọn ra là tối giản, nghĩa là *luôn chọn số khóa ít nhất*, chỉ chọn nhiều hơn 1 khóa là nó cần thiết để thể hiện sự Duy Nhất.
  - ? Ví dụ, trong bảng trên ta có `SuperKey = { {StudentID, Name}, {StudentID, Age}, {StudentID, Name, Age}, {StudentID} }` thì chỉ chọn `StudentID` là được.
- Chính vì thế, mọi **Primary Key** đều là **Candidate Key**, bởi vì nó được chọn ra trực tiếp từ một candidate key. Tương tự, mọi **Candidate Key** đều là **Super Key**, vì Super Key là bao gồm mọi Candidate Key + các thuộc tính khác.

Chúng ta biết rằng tất cả Super, Candidate đều xây dựng dựa trên Primary Key. Vậy hãy bắt đầu từ cơ bản, bằng cách hiểu Primary Key là gì, và nó được tạo nên từ gì.

Trong một bảng có các cột và hàng, mỗi cột đại diện cho một attribute và mỗi hàng là một ví dụ cụ thể với các attribute đó. Primary Key là một attribute hoặc tổ hợp attribute được sử dụng để xác định duy nhất các attribute khác.

- ? Ví dụ trong bảng này, StudentCode và ProjectID có thể được kết hợp làm Primary Key vì các thuộc tính này đảm bảo mỗi hàng là duy nhất. (chỉ dùng Student thì dòng 1 và dòng 2 sẽ trùng nhau)

Student Code	Project ID	Project Leader
101	P03	Andy
101	P01	Peter
102	P04	Marry
103	P02	Smith

Từ ví dụ trên, ta biết rằng **có những attribute có thể và không thể trở thành Primary Key**.

- ? Ví dụ, thuộc tính ProjectLeader không phải là prime-attribute vì tên là lặp lại, nếu có 2 người tên Andy, ta sẽ có 2 dòng `{101, (P03, Andy)}` và `{102, (P03, Andy)}`, nếu 2 thuộc tính `(ProjectID, Project_Leader)` không thể xác định duy nhất các attribute khác (ví dụ `StudentCode`) thì nó không phải là Primary Key.

Vậy nên **Primary Key** là **một cột hoặc tổ hợp cột dùng để xác định duy nhất mỗi hàng trong bảng**. (chỉ có 1 trong mỗi bảng)
Hơn nữa, Primary Key là **tối thiểu**, nên nếu **không cần thiết kết hợp** `ProjectID` và `Project_leader` làm Primary Key thì chỉ cần chọn 1 attribute (`ProjectID` hoặc `Project_Leader`) làm Primary Key.

- ? Ví dụ cho PrimaryKey: StudentID, Email (vs điều kiện là Unique và không NULL)

- **Prime Attribute (thuộc tính có thể là Primary Key):** thuộc tính là duy nhất, có thể dùng để xác định duy nhất các attribute khác. Ví dụ: StudentID hoặc Email.
  Prime attribute cũng là **một phần của Candidate Key** (key có thể được chọn làm primary key)
- **Non-Prime Attribute (thuộc tính không thể là Primary Key):** thuộc tính không duy nhất và lặp lại như Name hoặc SchoolName.
  Attribute này cũng **không là một phần của Candidate Key** (không thể được chọn làm primary key vì tính lặp lại). Ví dụ: Name, Age, SchoolName

Candidate Key - là **TẬP HỢP các thuộc tính có thể được chọn làm Primary Key**. Do đó, prime attribute là một phần của Candidate Key (vì prime-attribute là thuộc tính có thể được chọn làm Primary Key).
Ví dụ:
`{StudentID}, {Email}` là prime-attribute trong `Candidate Key = {{StudentID}, {Email}}`. Và chỉ 1 khóa (`{StudentID}` hoặc `{Email}`) trong Candidate Key được chọn làm Primary Key.

StudentID	Email	CCCD	HoTen
SV01	sv01@gmail.com	123456789012	An
SV02	sv02@gmail.com	123456789013	Bình
SV03	sv03@gmail.com	123456789014	Cường

note: *không nên viết* `CandidateKey={StudentID, Email, CCCD}` vì nó sẽ giống như CompositeKey mà ta học sau này.

Alternate Key - là **Candidate Key không được chọn làm Primary Key**.
Ví dụ: Nếu StudentID được chọn làm Primary Key, thì Email trở thành Alternate Key.

Super Key - **bất kỳ tổ hợp nào của Candidate Key** và bất kỳ tập con nào của Candidate Key (tức prime attribute) miễn là **xác định duy nhất bản ghi trong bảng**. Bao gồm cả tổ hợp giữa Candidate Key và Non-Prime attribute.
Lưu ý: thuộc tính trong {} biểu diễn một Super Key.
Ví dụ: {StudentID}, {Email}, {StudentID, Email}, {Email, StudentName}, {StudentID, Email, StudentName}

- $ Tóm lại, Super Key bao gồm tất cả tổ hợp khóa có thể tổ hợp được.

> **Note**
>
> Sự khác biệt giữa Primary Key và Super Key là *Super Key bao gồm mọi tổ hợp khóa có thể được chọn là khóa chính*, còn *PrimaryKey là khóa chính thức* được chọn làm Khóa Chính trong 1 bảng.
>
> **Candidate Key** trong **SuperKey**

Composite Key 1 cách gọi khác của Primary Key (khóa chính) khi **Primary Key nhưng có nhiều hơn 1 thuộc tính**. Cụ thể là **Composite Key cũng là khóa tối thiểu giống Primary Key, vì có > 1 khóa nên gọi Composite Key có là tổ hợp khóa tối thiểu**.
Điều này có nghĩa nếu 2 khóa là đủ để xác định mỗi hàng trong bảng, thì chỉ 2 khóa đó được chọn làm Composite Key.
Trong khi Super Key có thể dư thừa, ví dụ {StudentID, Email, StudentName}.

Concept	Minimal?	Uniquely identifies row?	Example
Super Key	❌ Can be redundant	✅ Yes	{StudentID, CourseID, Grade}
Composite Key	✅ Yes	✅ Yes	{StudentID, CourseID}

Lưu ý: Prime và Non-Prime không giống với Candidate và Alternate Key. Nó mô tả xem một thuộc tính **có thể trở thành key** hay không, trong khi Candidate và Alternate Key mô tả xem một Prime Attribute **đã được chọn làm Primary Key hay chưa**.

Lưu ý: $X \to Y$ biểu thị sự phụ thuộc hàm (functional dependency - FD), nghĩa là nếu 2 hàng có cùng giá trị X, thì chúng phải có cùng giá trị Y.
Ví dụ: nếu $StudentID \to Name$, thì $StudentID$ xác định duy nhất Name.

Transitive Dependency (Tính chất bắc cầu thông qua non-prime attribute): non-prime attribute phụ thuộc vào một non-prime attribute khác thay vì phụ thuộc vào Primary Key.
Giả sử Z và Y là non-prime. X là primary key.
Vì $X \to Y$ và $Z \to Y$ nên ta có thể xác định $X \to Y \to Z$ (xác định X thông qua Y và Y thông qua Z) $\to$ dẫn đến dư thừa dữ liệu.

Tóm tắt: X $\to$ Y là trivial (tầm thường) nếu $Y \in X$, còn $X \to Y$ là non-trivial (không tầm thường) ngược lại.

Trivial Dependency: Phụ thuộc **cung cấp thông tin mới**. Trong thuật ngữ, ta nói đây là 1 phụ tuộc "Không hiển nhiên" (i.e. nghĩa là cung cấp thông tin đã biết rồi)
Ví dụ:

- $StudentID, Course \to Instructor$
- $Course \to Department$

Non-Trivial Dependency: Phụ thuộc **không cung cấp thông tin mới** -> Trong thuật ngữ, có thể nói đây là 1 phụ thuộc "Hiển nhiên".
Ví dụ:

- $StudentID, Course \to StudentID$
- $StudentID \to StudentID$

Normalization Form

Giải thích 1NF → 3NF, tại BCNF giải thích sự khác biệt giữa BCNF và 3NF → 4NF đến 5NF.

1NF: Loại bỏ các thuộc tính đa giá trị trong 1 cột và lặp dòng điều này nghĩa là ô trong bảng chỉ chứa 1 giá trị và mỗi bản ghi (row) trong bảng được xác định là duy nhất. Không có hàng nào trùng lặp.

StudentID	Subjects
SV01	Math, Physics

Đúng (chuẩn 1NF):

StudentID	Subject
SV01	Math
SV01	Physics

2NF: loại bỏ partial dependency tức là không còn phụ thuộc một phần trong bảng, nghĩa là mọi attribute không phải là PrimaryKey đều phụ thuộc hoàn toàn vào 1 primary key (i.e. ko nhiều 1 khóa).

- **!** Hạn chế phụ thuộc của attributes vào nhiều primary key nhưng không hạn chế phụ thuộc giữa Attribute và Attribute.

> Ví dụ: nếu {Player_ID, Item_Type} → Item_Quantity, Player_Rating - thì là 2NF
>
> Primary key: { Player_ID, Item_Type }
>
> Player_Inventory
>
> Non-key attributes: Item_Quantity, Player_Rating
>
Player_ID	Item_Type	Item_Quantity	Player_Rating
jdog21	amulets	2	Intermediate
jdog21	rings	4	Intermediate
gila19	copper coins	18	Beginner
trev73	shields	3	Advanced
trev73	arrows	5	Advanced
trev73	copper coins	30	Advanced
trev73	rings	7	Advanced

3NF loại bỏ transitive dependency nghĩa là **không có sự phụ thuộc bắc cầu giữa các attributes không phải là primary key**. Nghĩa là mọi attribute trong bảng chỉ phụ thuộc hoàn toàn vào primary key và không phụ thuộc vào bất kỳ prime-attribute hay non-prime attribute nào khác.
Ví dụ:

- *StudentCity phụ thuộc vào StudentZipcode* vì một City có thể xác định thông qua ZipCode (lưu ý rằng StudentZipcode là một phần của Candidate key)
- *StudentZipcode phụ thuộc vào StudentCode*. Vậy ta có thể dùng StudentCode để xác định StudentCity thông qua StudentZipCode → transitive dependency.

- ! Transitive Dependency có thể **gây ra sự dư thừa khi cập nhật** vì nếu có 2 dòng cùng StudentCity, bạn sẽ phải cập nhật StudentCity 2 lần, đúng không?

❖ **3NF:** The normalization of 2NF relations to 3NF involves the elimination of transitive dependencies.

A functional dependency X -> Z is said to be transitive if the following three functional dependencies hold:
•X -> Y & Y does not -> X & Y -> Z

Student Code	Student Name	Student Zipcode	Student City
1	Vinh	94000	CanTho
2	Vinh	84000	Tay Ninh
3	Andy	84000	Tây Ninh

- $ Thay vào đó, nếu ta tách StudentZipcode và StudentCity ra riêng và dùng StudentZipcode để tham chiếu tới bảng Student từ bảng Student_Location thì ta chỉ cần cập nhật StudentCity một lần, vì mỗi bản ghi trong Student_Location được xác định duy nhất, không lặp lại như ở ví dụ đầu tiên.

Student

Student Code	Student Name	Student Zipcode
1	Vinh	94000
2	Vinh	84000
3	Andy	84000

Student_Location

Student Zipcode	Student City
94000	CanTho
84000	Tay Ninh
84000	Tây Ninh

Khác biệt chính giữa 3NF và BCNF: Trong **3NF**, một **non-prime attribute có thể phụ thuộc vào 1 PrimaryKey hoặc 1 prime attribute bên trong PrimaryKey. Chứ không bắt buộc phải phụ thuộc hoàn toàn vào PrimaryKey**.
Ví dụ:
`Instructor` phụ thuộc vào `PrimaryKey(StudentCode, Course)`
`Instructor` cũng phụ thuộc vào `Course` → **không phụ thuộc hoàn toàn vào PrimaryKey.**

StudentCode	Course	Instructor
101	CS101	Dr. Smith
101	CS102	Dr. Jones
102	CS101	Dr. Smith
103	CS102	Dr. Jones

- Trong **BCNF**, là *3NF nhưng không attribute nào (prime hay non-prime) được phép phụ thuộc vào bất kỳ thứ gì khác ngoài PrimaryKey*
  Ví dụ:
  `Instructor` phụ thuộc vào `PrimaryKey(StudentCode, Course)`
  `Instructor` cũng phụ thuộc vào `Course` → **vi phạm BCNF Tóm lại:** Mọi attribute chỉ được phép phụ thuộc vào toàn bộ super key (ví dụ: phụ thuộc cả Student_Code và ProjectID), chứ không được phụ thuộc vào một prime attribute đơn lẻ (tức là chỉ phụ thuộc vào ProjectID hoặc StudentCode).

Tách bảng thành R_1 và R_2 để thỏa mãn BCNF. $R_1(Course, Instructor)$

Course	Instructor
CS101	Dr. Smith
CS102	Dr. Jones

$R_2(StudentCode, Course)$

StudentCode	Course
101	CS101
101	CS102
102	CS101
103	CS102

4NF: loại bỏ multi-value dependency (trong bảng có ít nhất 3 cột)
Bảng thỏa mãn **4NF cũng sẽ thỏa mãn BCNF và không được có bất kỳ Multi-valued Dependency nào.**
Lưu ý: A và B là các attribute, do đó chúng có thể chứa 1 hoặc nhiều giá trị.

Multi-valued Dependency ($\rightarrow\rightarrow$) nghĩa là

Course	Teacher	Book
CS101	Smith	Book1
CS101	Smith	Book2
CS101	Jones	Book1
CS101	Jones	Book2

1. Với phụ thuộc $A \rightarrow B$, nếu với một giá trị A duy nhất `ví dụ: Course=CS101`, tồn tại nhiều giá trị B `ví dụ: Teacher={Smith, Jones}`. (tức là A xác định một tập giá trị của B)
2. Một bảng cần có ít nhất 3 cột để xảy ra Multi-valued Dependency `ví dụ: R(Course, Teacher, Book)` (*tạo điều kiện để chứng minh điều kiện 3.*)
3. Với quan hệ `R(Course, Teacher, Book)`, nếu tồn tại Multi-valued Dependency giữa Course và Teacher, thì Book phải độc lập.

Bảng có Multi-valued Dependency (tức là vi phạm 4NF) có thể được tách thành $R_1(Course, Teacher)$ và $R_2(Course, Book)$, yêu cầu bảng gốc phải có ít nhất 3 attribute.

$R_1(Course, Teacher)$:

Course	Teacher
CS101	Smith
CS101	Jones

$R_2(Course, Book)$:

Course	Book
CS101	Book1
CS101	Book2

Trong đó:

- B `Teacher` không phụ thuộc vào C `Book`
- Primary key của $R_1(A, B)$ nhưng `Course` là attribute liên kết.
- Primary key của $R_2(A, C)$ nhưng `Course` là attribute liên kết. Course chỉ đơn giản là thuộc tính chung.
- $A \rightarrow\rightarrow B$ và $A \rightarrow\rightarrow C$ ($\rightarrow\rightarrow$ nghĩa là multi-valued dependency)
- Lưu ý: A có thể là primary key trong bảng riêng của nó (ví dụ bảng `Course`) và cả $R1$ và R_2 có thể tham chiếu tới nó thông qua foreign key.

Thực Hành: từ 0NF tới 4NF

- **$ Mục Tiêu:** hiểu được hướng tiếp cận và áp dụng chuẩn hóa.

StudentID	StudentName	Courses
1	Alice	{CS101, MA101}
2	Bob	{CS101, PH101, MA101}

StudentID	StudentName	Courses	Instructor	Hobbies
S01	An	Math, Physics	Dr. Nam, Dr. Phuc	Reading, Playing Guitar
S02	Binh	Chemistry	Dr. Hanh	Swimming
S01	An	Literature, History	Dr. Thao, Dr. Tuan	Reading, Playing Guitar
S03	Cuong	Math, Literature, Physics	Dr. Nam, Dr. Thao, Dr. Phuc	Drawing

Các vấn đề tồn tại (vi phạm từ 1NF trở xuống):

- Ô Courses chứa nhiều giá trị (vi phạm nguyên tử).
- Ô Instructor cũng chứa danh sách (vi phạm nguyên tử).
- Hobbies cũng chứa nhiều sở thích trong một ô.
- Lặp dữ liệu (S01 xuất hiện hai lần).
- Instructor không rõ là dạy môn nào.

1NF: Loại bỏ thuộc tính đa giá trị, đảm bảo tính độc nhất cho mỗi dòng

StudentID	StudentName	CoursesID	Instructor	Hobbies
S01	An		Dr. Nam, Dr. Phuc	Reading, Playing Guitar
S02	Binh	Chemistry	Dr. Hanh	Swimming
S01	An	Literature, History	Dr. Thao, Dr. Tuan	Reading, Playing Guitar
S03	Cuong	Math, Literature, Physics	Dr. Nam, Dr. Thao, Dr. Phuc	Drawing

StudentID	StudentName	Courses	Instructor	Hobbies
S01	An	Math, Physics	Dr. Nam, Dr. Phuc	Reading, Playing Guitar
S02	Binh	Chemistry	Dr. Hanh	Swimming
S01	An	Literature, History	Dr. Thao, Dr. Tuan	Reading, Playing Guitar
S03	Cuong	Math, Literature, Physics	Dr. Nam, Dr. Thao, Dr. Phuc	Drawing