CHƯƠNG 2.
CÁC MÔ HÌNH DỮ LIỆU

(Phần 2)

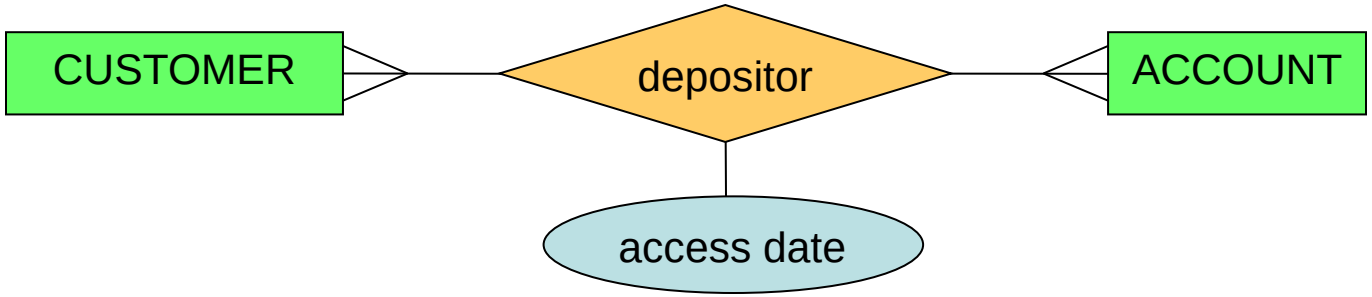
MÔ HÌNH DỮ LIỆU QUAN HỆ

- Giới thiệu
- Quá trình thiết kế một CSDL
- Lược đồ thực thể liên kết E-R
- Một số vấn đề cần quan tâm khi thiết kế lược đồ E-R
- Lược đồ dữ liệu quan hệ
- Ánh xạ lược đồ thực thể liên kết sang lược đồ quan hệ

MỘT SỐ VẤN ĐỀ CẦN QUAN TÂM KHI THIẾT KẾ LƯỢC ĐỒ E-R

ẢNH HƯỞNG CỦA ÁNH XẠ LỰC LƯỢNG LIÊN KẾT LÊN CÁC KHÓA

- Cấu trúc của khóa chính cho tập các quan hệ phụ thuộc vào việc ánh xạ lực lượng liên kết. Xét lược đồ E-R sau:



=> Lược đồ này thể hiện một quan hệ M-M cho quan hệ ***depositor*** với thuộc tính ***access-date*** liên quan tới tập quan hệ giữa hai thực thể **CUSTOMER** và **ACCOUNT**.

Khóa chính của quan hệ này sẽ bao gồm hợp của các khóa chính của hai tập thực thể **CUSTOMER** và **ACCOUNT**.

ẢNH HƯỞNG CỦA ÁNH XẠ LỰC LƯỢNG LIÊN KẾT LÊN CÁC KHÓA (Cont.)

- Để rõ hơn, xét lược đồ dữ liệu của hai tập thực thể:

  CUSTOMER (*customer-id, customer-name, address, city*)

  ACCOUNT (*account-number, balance*)

  - Quan hệ M-M giữa hai tập thực thể **CUSTOMER** và **ACCOUNT** nghĩa là: một khách hàng có thể có nhiều tài khoản, và tương tự một tài khoản có thể được quản lý bởi nhiều khách hàng.
  - Phép hợp các khóa chính của cả hai tập thực thể **CUSTOMER** và **ACCOUNT** sẽ xác định duy nhất một quan hệ giữa hai thực thể trong **CUSTOMER** và **ACCOUNT**.

ẢNH HƯỞNG CỦA ÁNH XẠ LỰC LƯỢNG LIÊN KẾT LÊN CÁC KHÓA (Cont.)

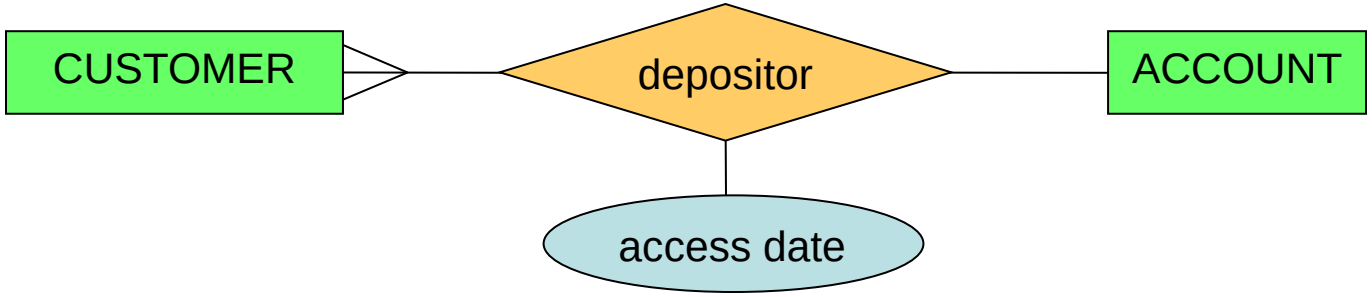
- Để nhìn thấy lần nộp tiền cuối cùng *(last deposit)* vào một tài khoản cụ thể nào đó, cần xác định người nộp tiền vì với mỗi tài khoản có thể có một số người nộp tiền vào.
- Lược đồ cho mối quan hệ ***depositor*** như sau:

  Depositor (*customer-id, account-number, access-date*) .

ẢNH HƯỞNG CỦA ÁNH XẠ LỰC LƯỢNG LIÊN KẾT LÊN CÁC KHÓA (Cont.)

- Xét trường hợp một khách hàng chỉ được phép nộp tiền vào một tài khoản duy nhất.

  => quan hệ ***depositor*** là M:1 từ **CUSTOMER** tới **ACCOUNT**:



  => Khóa chính của quan hệ ***depositor*** sẽ chỉ bao gồm khóa chính của tập thực thể **CUSTOMER**.

ẢNH HƯỞNG CỦA ÁNH XẠ LỰC LƯỢNG LIÊN KẾT LÊN CÁC KHÓA (Cont.)

- Để rõ hơn, xét lược đồ dữ liệu của hai tập thực thể:

  CUSTOMER (*customer-id, customer-name, address, city*)

  ACCOUNT (*account-number, balance*)

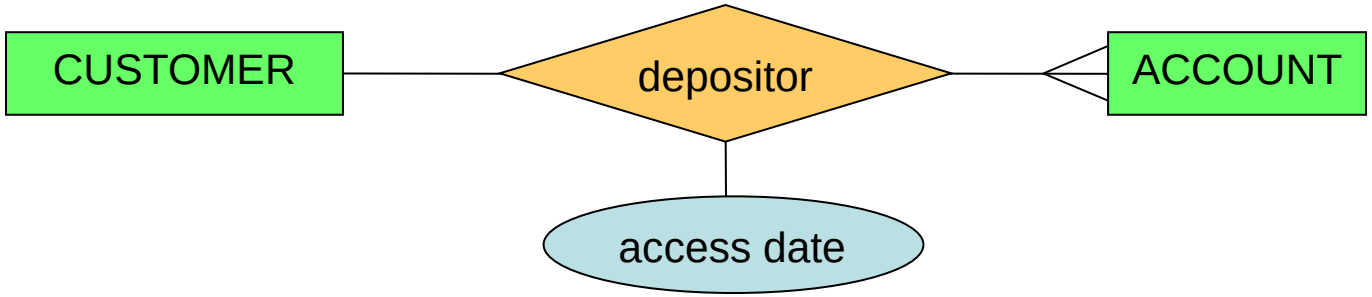
  - Quan hệ M:1 nghĩa là một khách hàng chỉ có duy nhất một tài khoản. Như vậy, việc chỉ ra số tài khoản đó là không cần thiết.
  - Khóa chính của quan hệ ***depositor*** chỉ đơn giản là khóa chính của thực thể **CUSTOMER**. Một khách hàng xác định chỉ có thể thực hiện một lần nộp tiền gần nhất tới tài khoản duy nhất họ có thể truy nhập.

- Lược đồ của tập các quan hệ ***depositor*** như sau:

  Depositor (*customer-id, access-date*)

ẢNH HƯỞNG CỦA ÁNH XẠ LỰC LƯỢNG LIÊN KẾT LÊN CÁC KHÓA (Cont.)

- Xét trường hợp quan hệ ***depositor*** là M:1 từ **ACCOUNT** tới **CUSTOMER**, nghĩa là: mỗi tài khoản được làm chủ bởi nhiều nhất là một khách hàng nhưng mỗi khách hàng có thể có nhiều hơn một tài khoản.



ẢNH HƯỞNG CỦA ÁNH XẠ LỰC LƯỢNG LIÊN KẾT LÊN CÁC KHÓA (Cont.)

=> Trong trường hợp này, khóa chính của quan hệ ***depositor*** chỉ đơn giản bao gồm khóa chính của thực thể **ACCOUNT** vì chỉ có thể có nhiều nhất một lần nộp tiền gần nhất tới một tài khoản nào đó xác định, và chỉ có nhiều nhất một khách hàng có thể thực hiện việc nộp tiền.

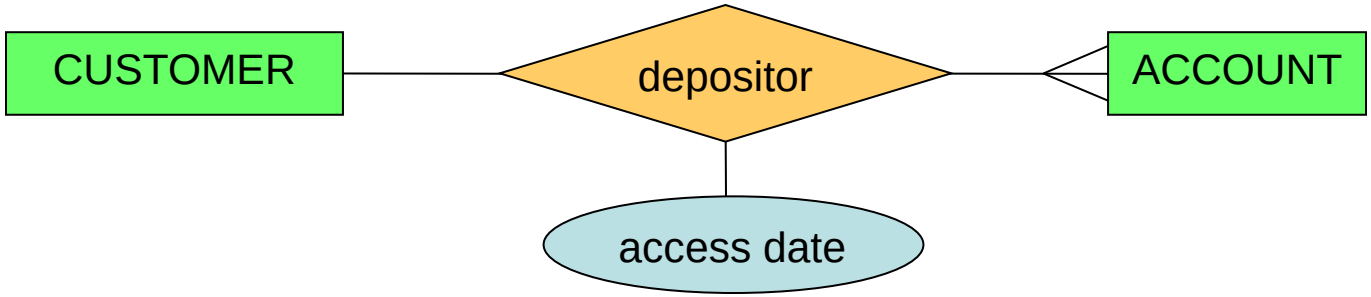
Không cần xác định khách hàng nào đã thực hiện nộp tiền vì chỉ có thể có một và chỉ một khách hàng có khả năng nộp tiền vào một tài khoản xác định.

- Lược đồ cho mối quan hệ ***depositor*** như sau:

Depositor (*account-number, access-date*)

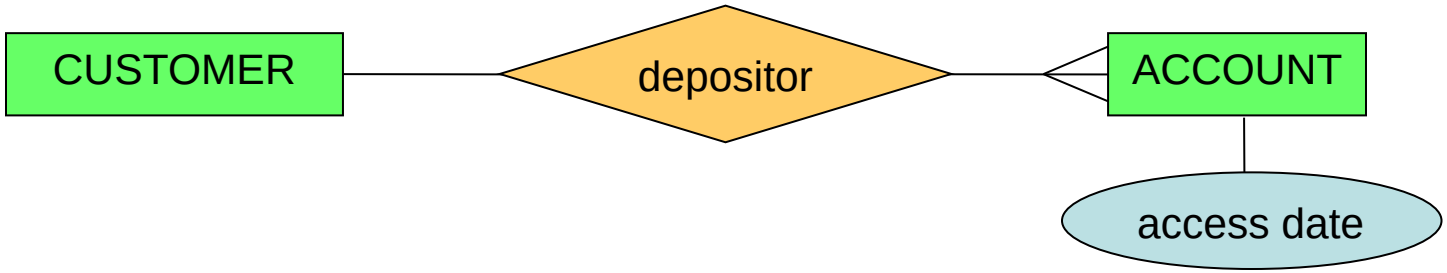
ĐẶT VỊ TRÍ CHO CÁC THUỘC TÍNH CỦA QUAN HỆ

- Các thuộc tính của một tập quan hệ dạng 1:1 hoặc 1:M thường được đặt vào trong các tập thực thể tham gia liên kết, hơn là được đặt vào bản thân tập các mối quan hệ đó.
  - Ví dụ, với quan hệ ***depositor***:



ĐẶT VỊ TRÍ CHO CÁC THUỘC TÍNH CỦA QUAN HỆ (Cont.)

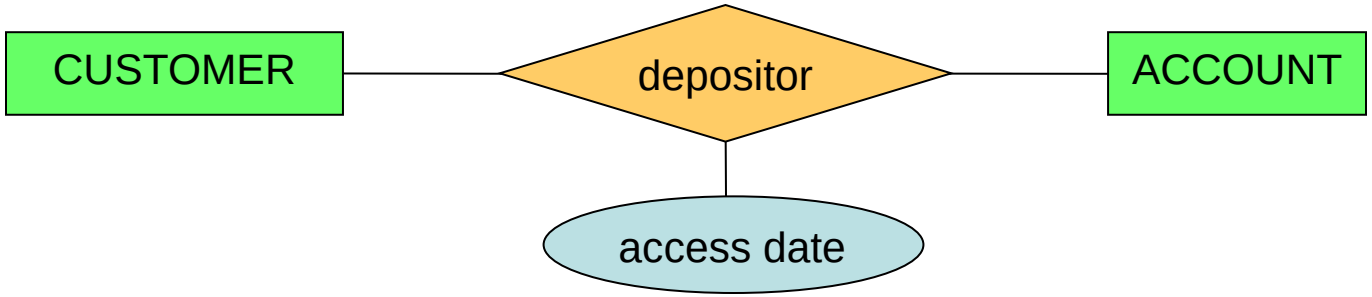
- Thuộc tính ***access-date*** có thể được đặt liên quan tới thực thể **ACCOUNT** mà không làm tổn thất thông tin:



=> Vì một tài khoản cụ thể thuộc sở hữu bởi nhiều nhất là một khách hàng, và tài khoản đó có thể có nhiều nhất một ***access-date***.

ĐẶT VỊ TRÍ CHO CÁC THUỘC TÍNH CỦA QUAN HỆ (Cont.)

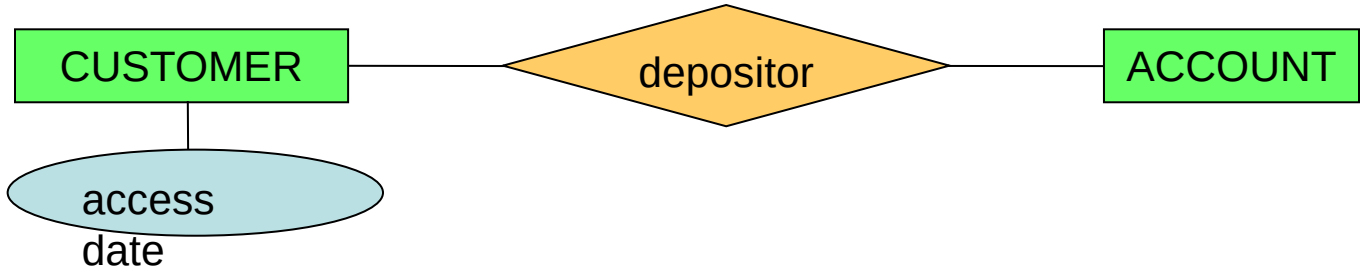
- Xét trường hợp: một tài khoản được làm chủ bởi nhiều nhất một khách hàng và một khách hàng chỉ có thể sở hữu duy nhất một tài khoản.



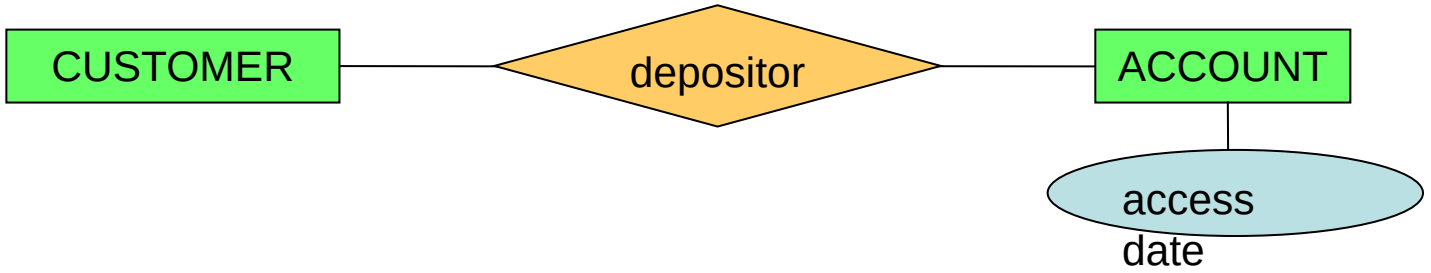
=> Thuộc tính ***access-date*** có thể gắn vào hoặc thực thể **CUSTOMER** hoặc tập thực thể **ACCOUNT** mà không làm tổn thất thông tin.

ĐẶT VỊ TRÍ CHO CÁC THUỘC TÍNH CỦA QUAN HỆ (Cont.)

- Nếu thuộc tính ***access-date*** được lưu trữ với tập **CUSTOMER** thì nó phải tham chiếu tới lần truy nhập cuối cùng của khách hàng tới tài khoản duy nhất mà họ có.

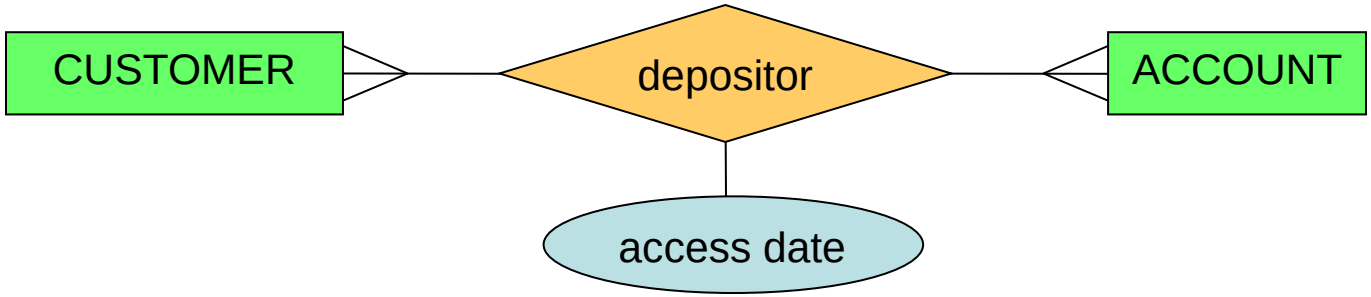


- Nếu thuộc tính ***access-date*** được lưu trữ trong thực thể **ACCOUNT** thì nó sẽ tham chiếu tới lần truy nhập cuối cùng tới tài khoản bởi người khách hàng duy nhất sở hữu nó.



ĐẶT VỊ TRÍ CHO CÁC THUỘC TÍNH CỦA QUAN HỆ (Cont.)

- Xét trường hợp: quan hệ ***depositor*** có ràng buộc N:N



=> Việc gắn thuộc tính ***access-date*** với bất kỳ tập thực thể tham gia nào cũng không mô hình hóa được tình huống này mà không làm tổn thất thông tin.

ĐẶT VỊ TRÍ CHO CÁC THUỘC TÍNH CỦA QUAN HỆ (Cont.)

⇒ Nếu cần lưu trữ ngày truy nhập cuối cùng của một khách hàng cụ thể tới một tài khoản cụ thể thì thuộc tính ***access-date*** nhất thiết phải là một thuộc tính của tập quan hệ ***depositor***, chứ không thể là thuộc tính của bất kỳ tập thực thể tham gia nào.

- Nếu ***access-date*** là một thuộc tính của **ACCOUNT** thì không thể xác định được khách hàng nào đã thực hiện việc chuyển tiền vào tài khoản đó.
- Còn nếu ***access-date*** là một thuộc tính của **CUSTOMER**, thì cũng không thể xác định được tài khoản nào khách hàng đã truy nhập vào lần cuối.

TẬP THỰC THỂ HAY CÁC THUỘC TÍNH

- Xét một tập thực thể:

  EMPLOYEE*(emp-name, telephone-number, age)*

- Nếu máy điện thoại telephone có thể được coi là một thực thể (với các thuộc tính *telephone-number, location, manufacturer, serial-num*, ...).

1. Thực thể EMPLOYEE phải được định nghĩa lại như sau: EMPLOYEE *(emp-name, age).*
2. Sau đó phải tạo ra một tập thực thể mới:

   TELEPHONE *(telephone-number, location, manufacturer, serial-num, ...)*
3. Và phải tạo ra một tập các mối quan hệ để xác định mối liên hệ giữa các nhân viên và các máy điện thoại mà họ sở hữu:

   EMP-PHONE*(emp-name, telephone-number, age, location, manufacturer, serial-num, ...).*

TẬP THỰC THỂ HAY CÁC THUỘC TÍNH (Cont.)

- Trong trường hợp mỗi nhân viên có một số điện thoại => coi máy điện thoại như một thuộc tính *telephone-number*.
- Trong trường hợp các nhân viên có thể sở hữu nhiều điện thoại => coi máy điện thoại như một thực thể (không xét đến trường hợp thuộc tính là đa trị).

=> Sự phân biệt 2 vai trò thực thể và thuộc tính phụ thuộc vào cấu trúc ngữ cảnh trong thế giới thực cần mô hình hóa dữ liệu và dựa vào ngữ nghĩa liên quan đến các thuộc tính trong bài toán cụ thể.

TẬP THỰC THỂ HAY TẬP QUAN HỆ

- *Xét ví dụ ngân hàng:* đã mô hình hóa khoản vay như một thực thể (**LOAN**).

  => *Cách khác:* có thể mô hình hóa khoản vay như một mối quan hệ giữa khách khàng (**CUSTOMER**) và các chi nhánh (**BRANCHES**) của ngân hàng với ***loan-number*** và ***amount*** là các thuộc tính mô tả.

  (Trong trường hợp: một khoản vay chỉ được sở hữu bởi duy nhất một khách hàng và có liên hệ với duy nhất một chi nhánh ngân hàng).

TẬP THỰC THỂ HAY TẬP QUAN HỆ (Cont.)

- Việc mô hình hóa theo cách trên không thuận lợi trong tình huống nhiều khách hàng cùng sở hữu chung một khoản vay.

  => Cần định nghĩa các mối quan hệ riêng cho từng người sở hữu khoản vay chung. Sau đó, dùng lại tất cả các giá trị cho các thuộc tính ***loan-number*** và ***amount*** trong mỗi quan hệ này (hiển nhiên, các quan hệ phải có cùng giá trị cho các thuộc tính mô tả).

- Có 2 vấn đề phát sinh khi dùng lặp lại các giá trị:

  1. Dữ liệu được lưu trữ ở nhiều nơi.
  2. Việc cập nhật dữ liệu làm tăng khả năng không nhất quán dữ liệu.

TẬP THỰC THỂ MẠNH HAY TẬP THỰC THỂ YẾU

- Tập thực thể không đủ các thuộc tính để hình thành một khóa chính gọi là tập thực thể yếu. Tập thực thể có khóa chính được gọi là tập thực thể mạnh.
- Ví dụ: Xét tập thực thể trả tiền:

  PAYMENT(*payment-number, payment date, payment amount*).

  => Mã số trả tiền (***payment-number***) thường là các số liên tiếp, bắt đầu từ 1 và được sinh ra riêng rẽ cho mỗi khoản nợ. Do đó, mặc dù mỗi thực thể **PAYMENT** là khác nhau, việc trả tiền cho các khoản nợ khác nhau có thể có cùng mã số ***payment-number***.

  => tập **PAYMENT** không có khóa chính và chỉ là một tập thực thể yếu.

TẬP THỰC THỂ MẠNH HAY TẬP THỰC THỂ YẾU

(Cont.)

- Để một tập thực thể yếu có ý nghĩa, nó phải liên hệ với một tập thực thể khác, được gọi là tập thực thể xác định hay tập thực thể sở hữu.
- Mối quan hệ giữa tập thực thể yếu với tập thực thể xác định được gọi là mối quan hệ xác định.
- Mối quan hệ xác định là quan hệ M:1 từ tập thực thể yếu tới tập xác định và sự tham gia của tập thực thể yếu trong quan hệ là đầy đủ.

TẬP THỰC THỂ MẠNH HAY TẬP THỰC THỂ YẾU
(Cont.)

- Mặc dù tập thực thể yếu không có khóa chính, nhưng có một phương thức để phân biệt tất cả các thực thể trong tập thực thể yếu phụ thuộc vào một thực thể mạnh cụ thể.

  => Tập các thuộc tính của một thực thể yếu cho phép phân biệt các thực thể được gọi là thuộc tính phân biệt (hay khóa bán phần).

  Ví dụ: Thuộc tính phân biệt của tập thực thể yếu **PAYMENT** là ***payment-number***, vì với mỗi khoản nợ, một mã số trả tiền sẽ xác định duy nhất một lần trả tiền riêng biệt cho khoản nợ này.

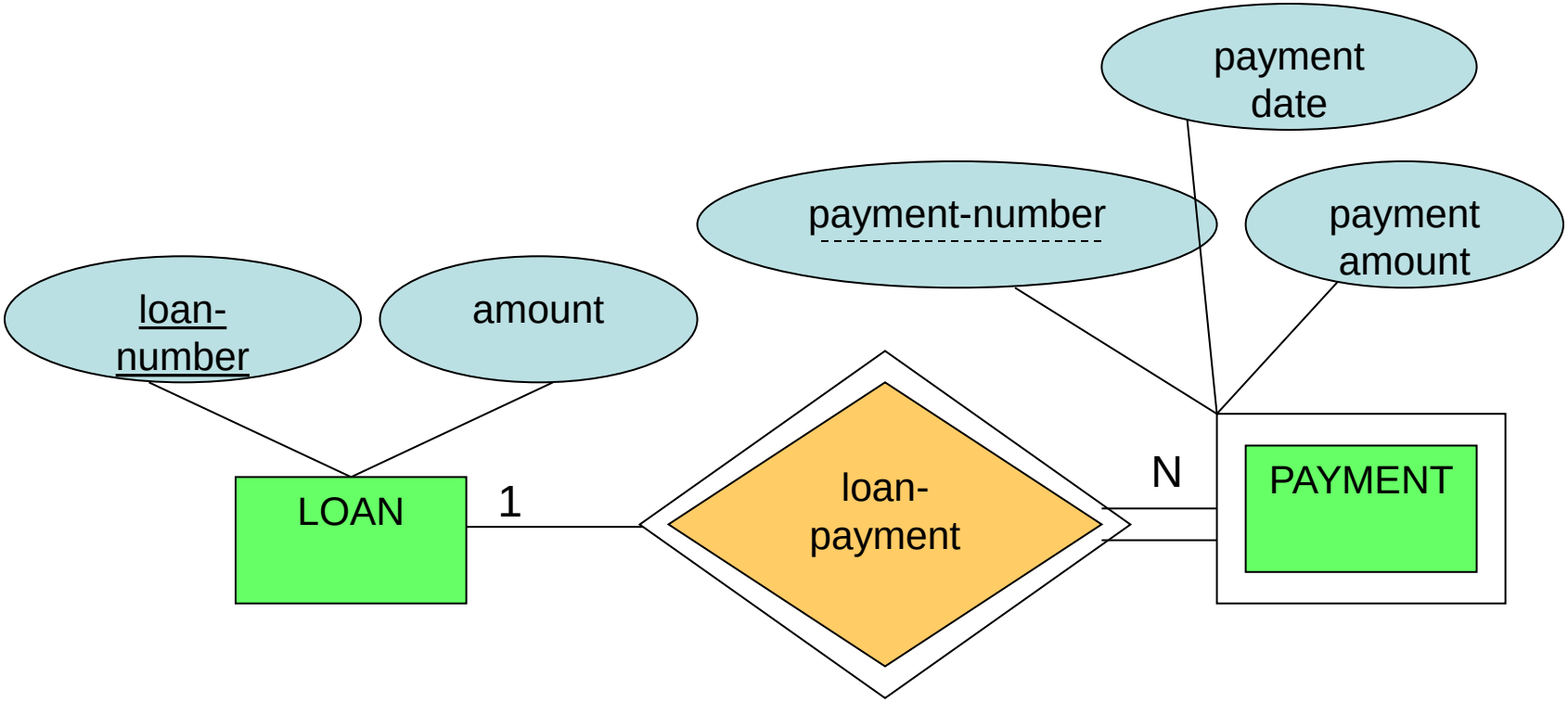
TẬP THỰC THỂ MẠNH HAY TẬP THỰC THỂ YẾU
(Cont.)

- Khóa chính của một tập thực thể yếu được cấu thành bởi khóa chính của tập thực thể xác định và thuộc tính phân biệt của tập thực thể yếu.

- Xét ví dụ trên, khóa chính của tập thực thể **PAYMENT** sẽ là (***loan-number***, ***payment-number***), trong đó ***loan-number*** là khóa chính của tập thực thể xác định **LOAN** và ***payment-number*** là thuộc tính phân biệt của tập thực thể yếu **PAYMENT**.

TẬP THỰC THỂ MẠNH HAY TẬP THỰC THỂ YẾU

(Cont.)



CỤ THỂ HÓA (Specialization)

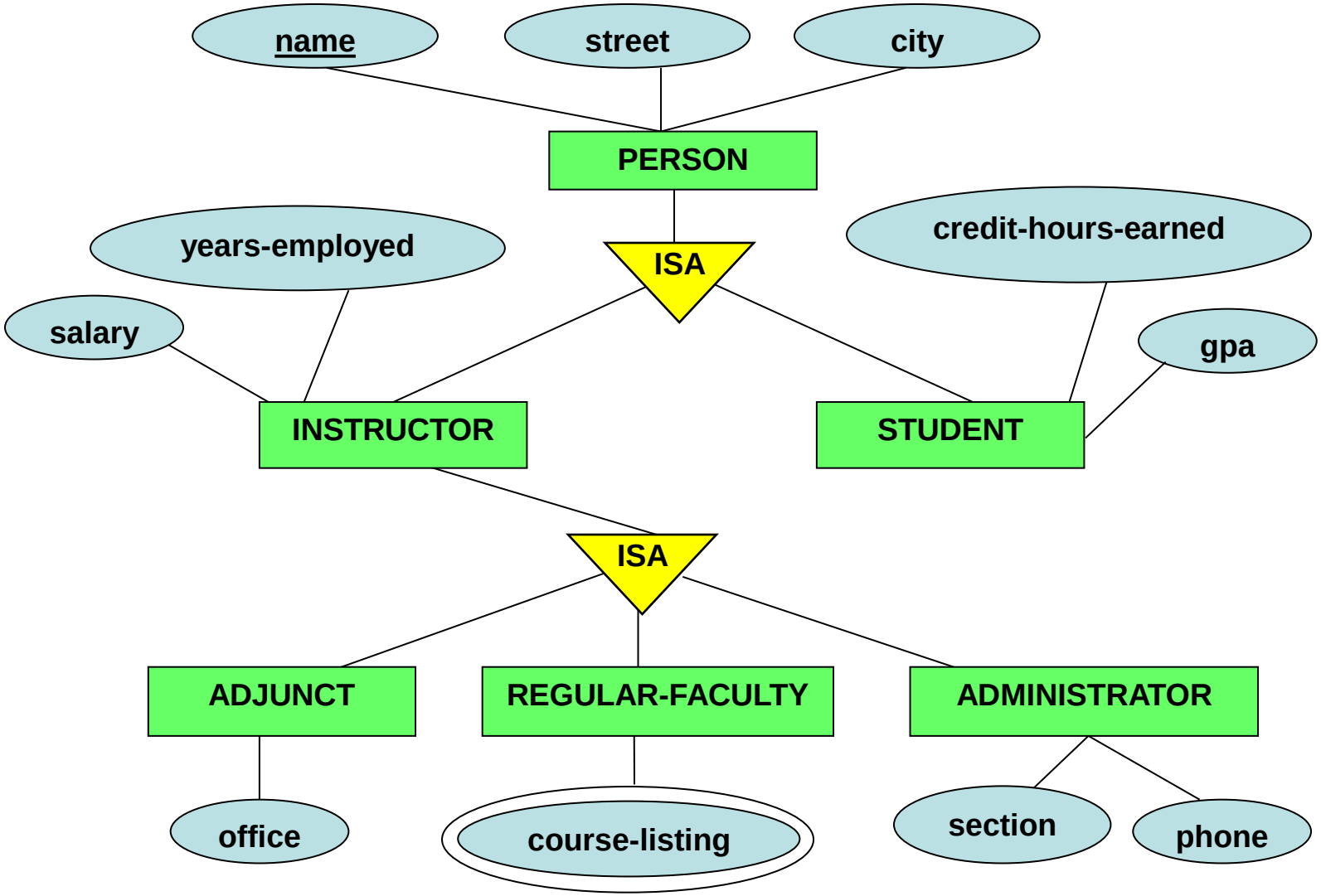
- Cụ thể hóa là quá trình thiết kế các phân nhóm trong một tập thực thể. Nghĩa là, phân biệt các tập thực thể con trong một tập thực thể khi chúng có các thuộc tính không giống nhau.

- Ví dụ: Xét tập thực thể người: PERSON(*name, street, city*).

  Một **PERSON** có thể được phân chia nhỏ hơn thành **STUDENT** (sinh viên) hoặc **INSTRUCTOR** (giảng viên). Mỗi loại này được mô tả bởi một tập các thuộc tính bao gồm tất cả các thuộc tính của tập thực thể **PERSON**, và một số thuộc tính bổ sung riêng.

  - Thực thể **STUDENT** có thể bổ sung thêm các thuộc tính ***gpa*** và ***credit-hours-earned***;
  - Thực thể **INSTRUCTOR** bổ sung thêm các thuộc tính ***salary*** và ***years-employed.***

CỤ THỂ HÓA (Cont.)



TỔNG QUÁT HÓA (Generalization)

- Cụ thể hóa thể hiện cách tiếp cận thiết kế từ trên-xuống. Quá trình ngược lại, được tiến hành theo cách tiếp cận từ dưới-lên gọi là tổng quát hóa.

  => Tổng quát hóa là nhiều tập thực thể được đồng bộ vào một tập thực thể ở mức cao hơn trên cơ sở các thuộc tính chung.

TỔNG QUÁT HÓA (Generalization)

❖ Xét ví dụ trước:

PERSON là tập thực thể ở mức cao (gọi là *cha*), và INSTRUCTOR và STUDENT là các tập thực thể ở mức thấp (gọi là *con*).

- Đầu tiên có thể xác định 2 tập thực thể:

  STUDENT*(name, address, city, gpa, credit-hours-earned)*,

  INSTRUCTOR*(name, address, city, salary, years-employed)*.

- Sau đó, tìm điểm chung giữa các thuộc tính để xây dựng tập thực thể:

  PERSON *(name, address, city).*

CỤ THỂ HÓA VÀ TỔNG QUÁT HÓA

- Cụ thể hóa và tổng quát hóa là 2 khái niệm ngược và có thể hoán đổi cho nhau trong việc thiết kế các lược đồ CSDL.
- Trong sơ đồ E-R, không có sự khác biệt giữa tổng quát hóa và cụ thể hóa mà chỉ khác nhau ở cách xem sơ đồ từ dưới lên hay từ trên xuống.

CỤ THỂ HÓA VÀ TỔNG QUÁT HÓA (Cont.)

- Sự khác nhau của hai cách tiếp cận thường được thể hiện bởi điểm bắt đầu và mục đích tổng thể của chúng:
  - Cụ thể hóa xuất phát từ tập thực thể riêng biệt; nhấn mạnh sự khác nhau giữa các thực thể trong cùng tập bằng cách tạo ra các tập thực thể phân biệt nhau ở mức thấp hơn.
  - Tổng quát hóa xuất phát từ việc nhận ra một số tập thực thể có chung một số các đặc tính. Trên cơ sở đó, tổng quát hóa tổng hợp các tập thực thể này thành một tập thực thể ở mức cao hơn. Quá trình tổng quát hóa nhấn mạnh tính tương đồng giữa các tập thực thể ở mức thấp hơn và giấu đi những khác biệt.

KẾ THỪA THUỘC TÍNH

- Các tập thực thể ở mức thấp hơn kế thừa các thuộc tính từ tập thực thể ở mức cao hơn.

  Ví dụ: **INSTRUCTOR** và **STUDENT** đều kế thừa tất cả các thuộc tính của **PERSON**.

- Tập thực thể mức thấp hơn cũng kế thừa các mối quan hệ thuộc về tập thực thể mức cao hơn định nghĩa nó.

- Cây phân cấp các tập thực thể:
  - Thực thể ở mức cao nhất nằm trên đỉnh của cây phân cấp.
  - Kế thừa đơn: Tập thực thể ở mức thấp hơn có một mối quan hệ ISA.
  - Đa kế thừa: Tập thực thể mức thấp có nhiều hơn một mối quan hệ ISA.

CÁC RÀNG BUỘC
TỔNG QUÁT HÓA (CỤ THỂ HÓA)

- ***Ràng buộc thứ nhất:*** Xác định thực thể nào có thể là thành viên của tập thực thể mức thấp hơn.

  Thành viên được định nghĩa theo một trong 2 cách:

  - Mệnh đề xác định: Thành viên được đánh giá trên cơ sở xác định xem thực thể có thỏa mãn một mệnh đề (điều kiện) tường minh nào đó không.
  - Người dùng xác định: Tập các thực thể mức thấp do người dùng xác định không bị ràng buộc bởi các điều kiện thành viên mà người dùng cơ sở dữ liệu gán các thực thể tới tập thực thể nào đó.

CÁC RÀNG BUỘC
TỔNG QUÁT HÓA (CỤ THỂ HÓA) (Cont.)

- ***Ràng buộc thứ hai:*** Trong quá trình tổng quát hóa, liệu các thực thể có thuộc vào nhiều hơn một tập thực thể ở mức thấp hơn không.

  Các tập thực thể ở mức thấp hơn có thể:

  - Không giao nhau: Một thực thể không thể thuộc vào nhiều tập thực thể ở mức thấp hơn. (Trong mô hình E-R, ràng buộc không giao nhau được thể hiện bằng từ "disjoint" đặt cạnh biểu tượng hình tam giác).
  - Giao nhau: Một thực thể có thể thuộc vào nhiều tập thực thể ở mức thấp hơn trong cùng một quá trình tổng quát hóa.

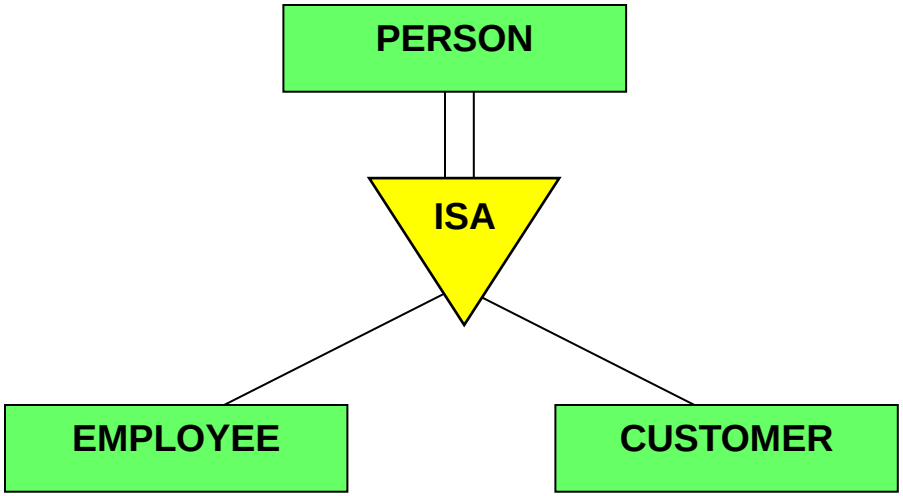
CÁC RÀNG BUỘC
TỔNG QUÁT HÓA (CỤ THỂ HÓA) (Cont.)

- ***Ràng buộc thứ ba:*** Là ràng buộc dựa trên tính toàn bộ, xác định xem một thực thể trong tập thực thể ở mức cao có thuộc vào một trong các tập thực thể ở mức thấp hơn hay không.

  Có 2 loại ràng buộc này:

  - Tổng quát hóa/cụ thể hóa toàn bộ: mỗi thực thể ở mức cao phải thuộc vào một thực thể ở mức thấp hơn.
  - Tổng quát hóa/cụ thể hóa một phần: Tồn tại thực thể ở mức cao không thuộc vào tập thực thể nào ở mức thấp hơn. (Đây là trường hợp mặc định).

VÍ DỤ ERD VỚI CÁC RÀNG BUỘC

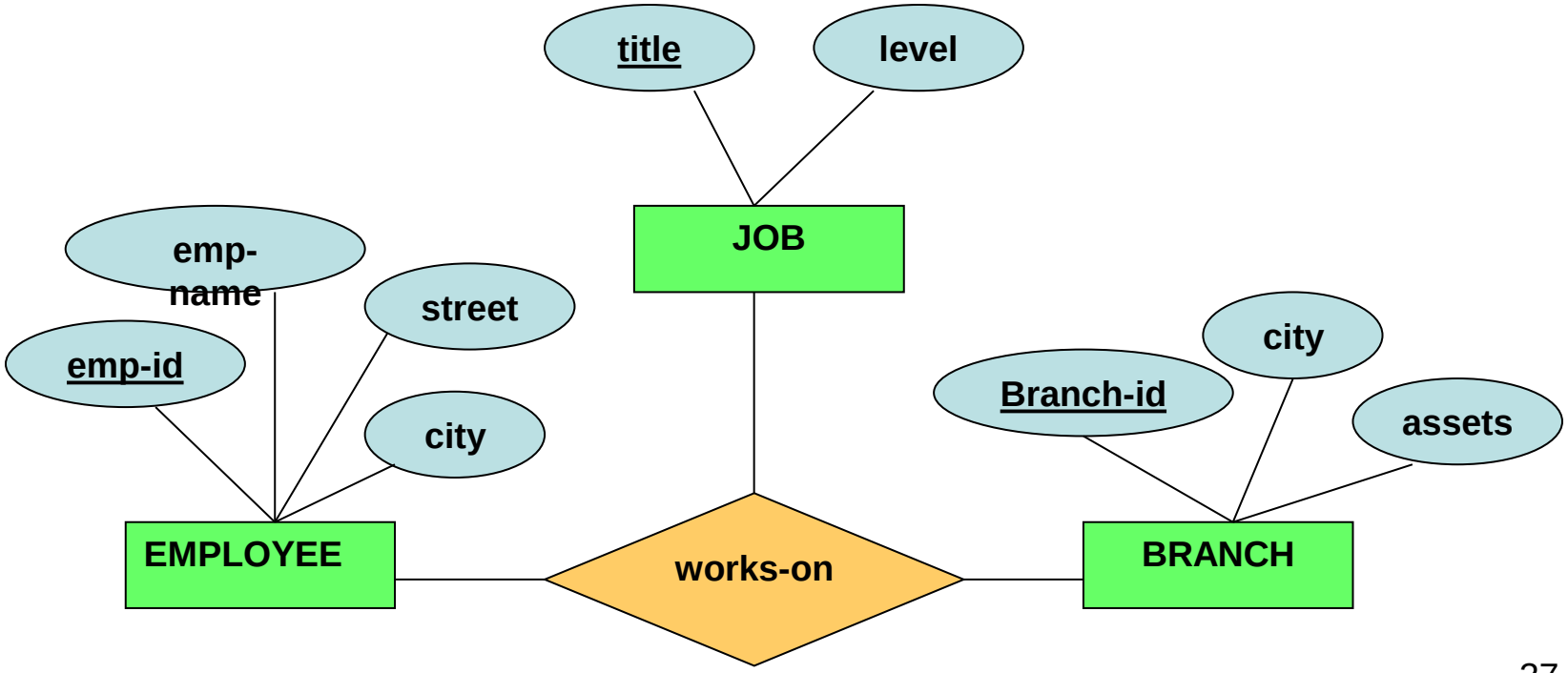


Tổng quát hóa toàn bộ và có giao nhau. Nghĩa là, mỗi người trong công ty xuất hiện như là một nhân viên hoặc là một khách hàng hoặc cũng có thể là cả hai.

KẾT HỢP

- Hạn chế của lược đồ E-R: không thể biểu diễn các mối quan hệ trong các mối quan hệ.

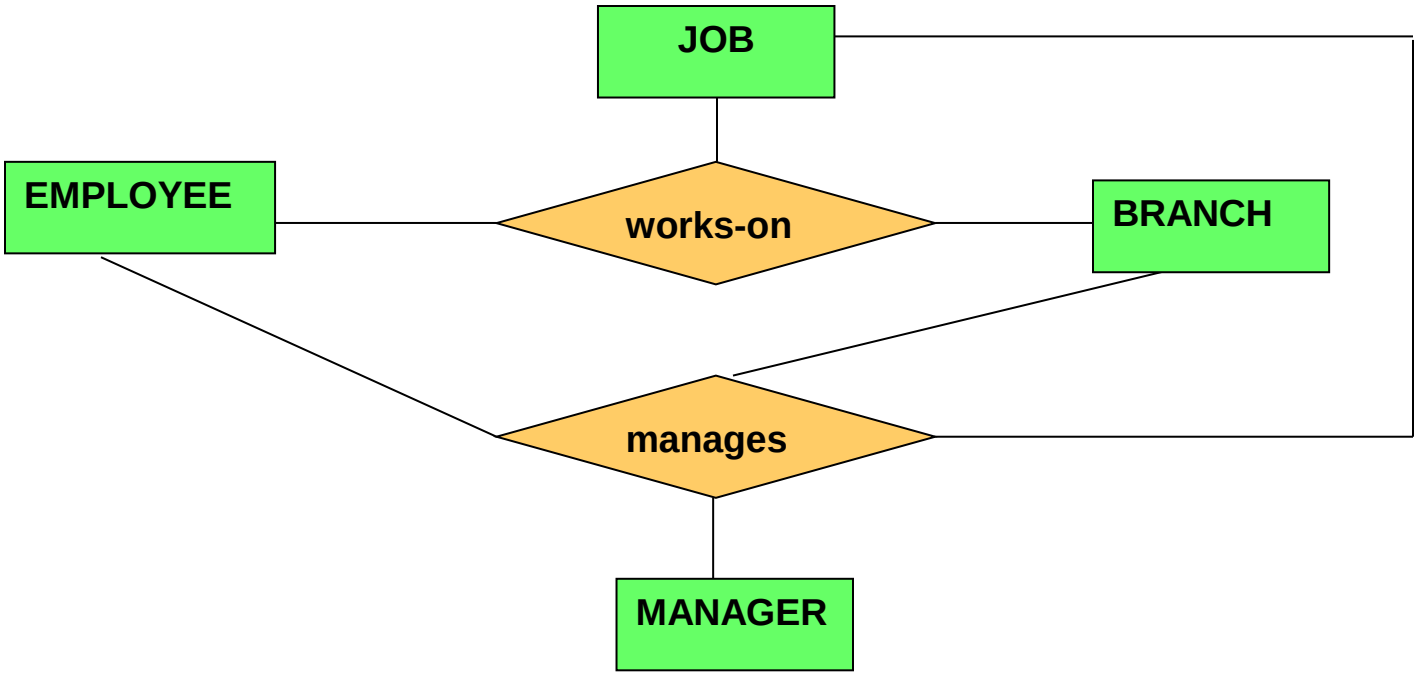
  Xét quan hệ ba ngôi ***works-on*** giữa **EMPLOYEE** *(nhân viên)*, **BRANCH** *(chi nhánh ngân hàng)* và **JOB** *(công việc)*:



KẾT HỢP (Cont.)

Giả sử muốn ghi nhận việc quản lý từng công việc được thực hiện bởi một nhân viên tại một chi nhánh; nghĩa là muốn lưu trữ người quản lý cho quan hệ ba ngôi (**EMPLOYEE**, **BRANCH**, **JOB**).

=> Một cách để thực hiện: tạo ra một quan hệ bốn ngôi với một tập thực thể **MANAGER**.



KẾT HỢP (Cont.)

Câu hỏi đặt ra:

Tại sao không thể hiện mối quan hệ hai ngôi giữa MANAGER và EMPLOYEE ?

=> Câu trả lời:

Một quan hệ hai ngôi sẽ không cho phép thể hiện liên kết (BRANCH, JOB) của một EMPLOYEE (nhân viên) được quản lý bởi người quản lý.

KẾT HỢP (Cont.)

Nhận xét:

- Trong lược đồ E-R mô hình hóa tình huống trên, tập các mối quan hệ ***works-on*** và ***manages*** có thể kết hợp thành một tập quan hệ duy nhất.

  => Tuy nhiên, không thể thực hiện việc kết hợp vì một bộ ba (**EMPLOYEE**, **BRANCH**, **JOB**) có thể không có người quản lý.

- Có thông tin dư thừa trong lược đồ E-R trên vì từng bộ ba (**EMPLOYEE**, **BRANCH**, **JOB**) của quan hệ ***manages*** cũng giống của ***works-on***.

KẾT HỢP (Cont.)

Nhận xét (cont.)**:**

- Nếu người quản lý là một giá trị chứ không phải là một thực thể, thì có thể biến **MANAGER** thành một thuộc tính đa trị của quan hệ ***works-on***.

  => Biến đổi này sẽ làm việc tìm kiếm khó khăn hơn (cả về logic lẫn về chi phí thực hiện). (Ví dụ, bộ ba **employee-branch-job** sẽ do người quản lý nào chịu trách nhiệm?).

KẾT HỢP (Cont.)

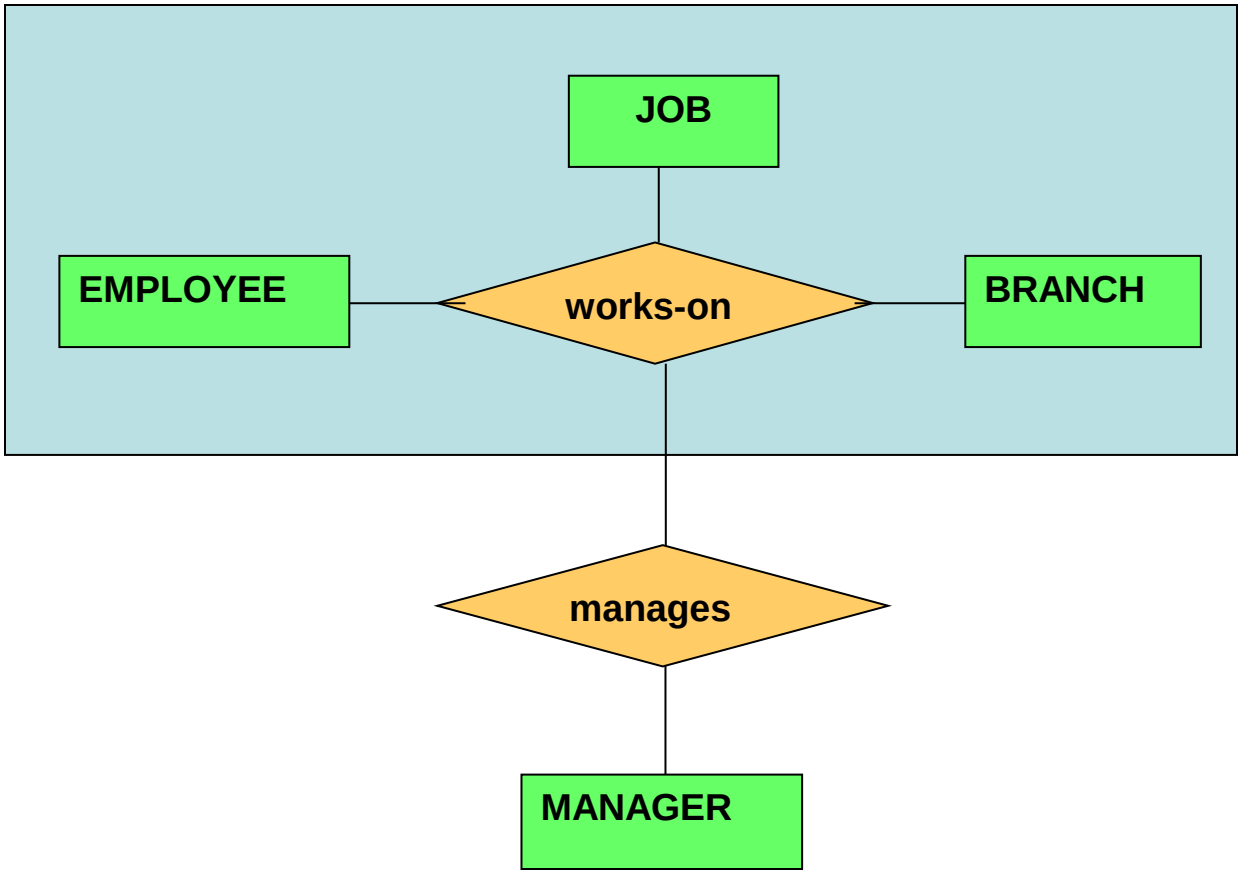
=> Cách tốt nhất để mô hình hóa là sử dụng **kết hợp** (hay **tích hợp**). *Kết hợp là một sự trừu tượng thông qua việc coi các mối quan hệ như là các thực thể ở mức cao.*

=> Xét ví dụ trên:

Có thể coi tập quan hệ **works-on** (liên quan tới các tập thực thể **EMPLOYEE**, **BRANCH** và **JOB**) như một tập thực thể ở mức cao, được đặt tên là **WORKS-ON**.

Tạo một quan hệ hai ngôi **manages** giữa **WORKS-ON** và **MANAGER** để thể hiện ai quản lý các công việc này.

KẾT HỢP (Cont.)



Sự kết hợp trong mô hình E-R

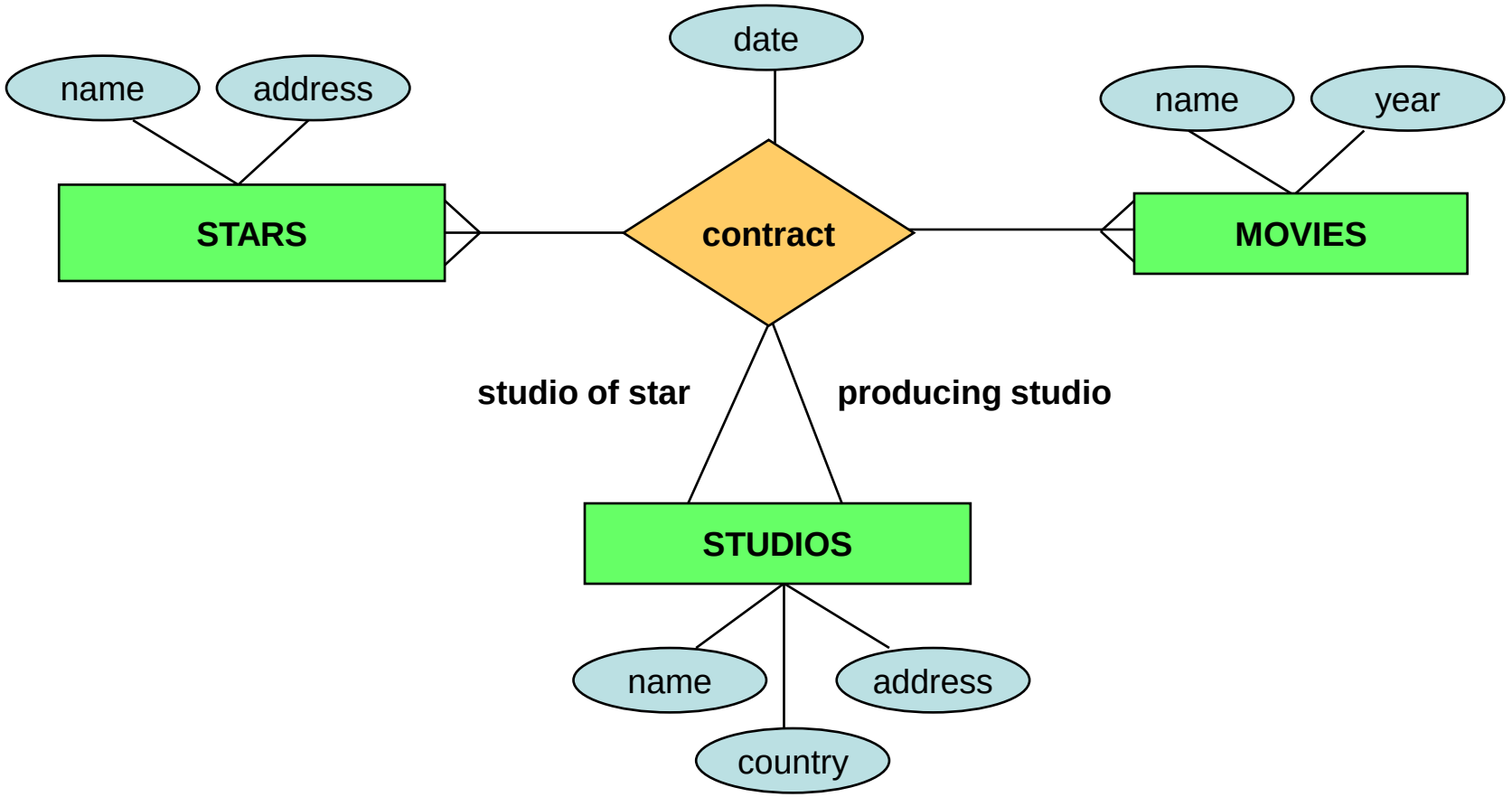
QUAN HỆ NHIỀU NGÔI

- Hầu hết các quan hệ đã xét đều là quan hệ hai ngôi, liên quan tới hai tập thực thể.
- Các quan hệ liên quan tới nhiều hơn hai tập thực thể có thể được chuyển đổi thành một tập các quan hệ hai ngôi, dạng nhiều-một.

=> Điều này rất có ích vì trong mô hình E-R không hạn chế các quan hệ tới quan hệ hai ngôi nhưng các mô hình dữ liệu khác có hạn chế này.

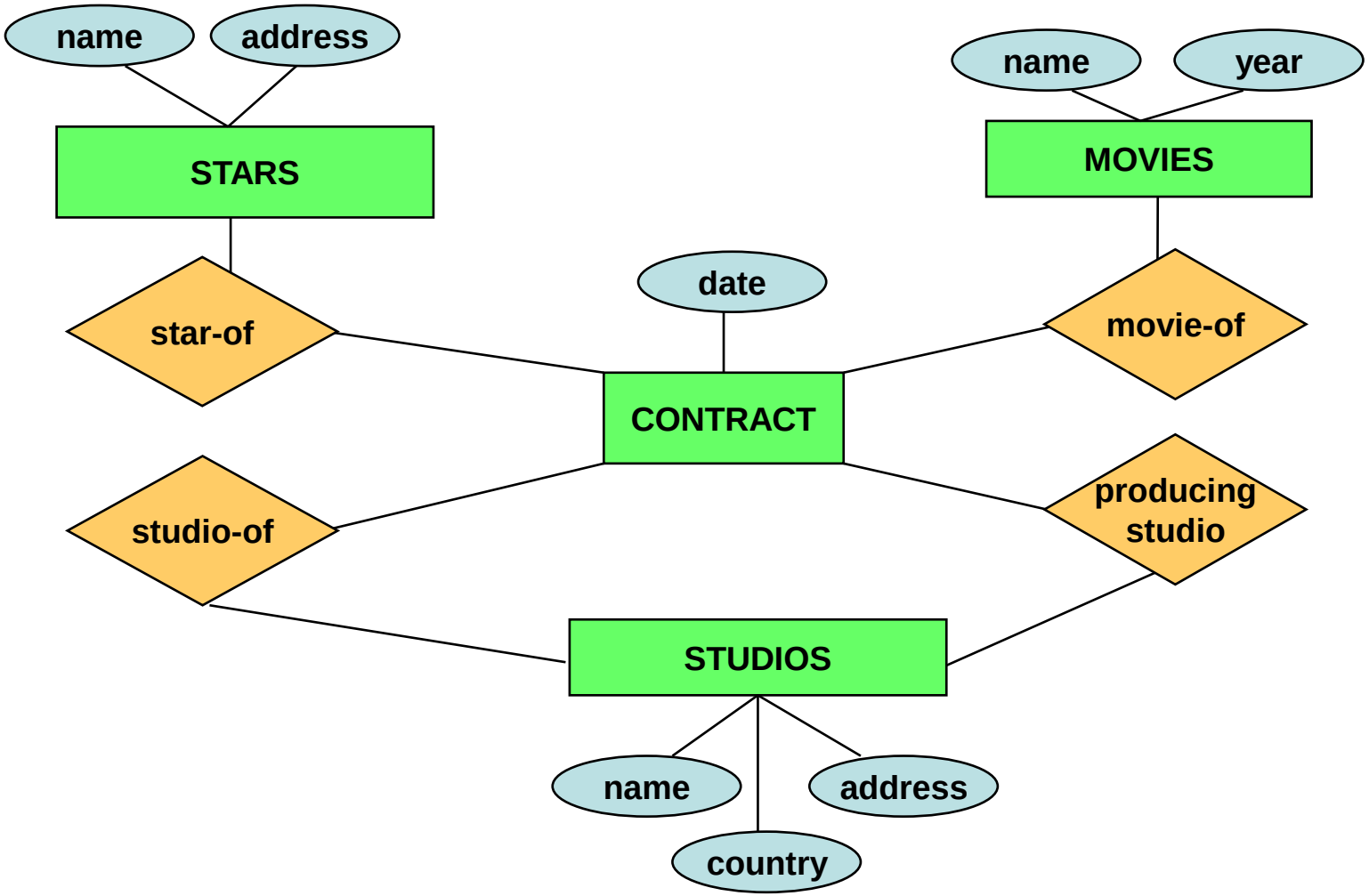
QUAN HỆ NHIỀU NGÔI (Cont.)

Xét ví dụ mô hình E-R với quan hệ nhiều ngôi:



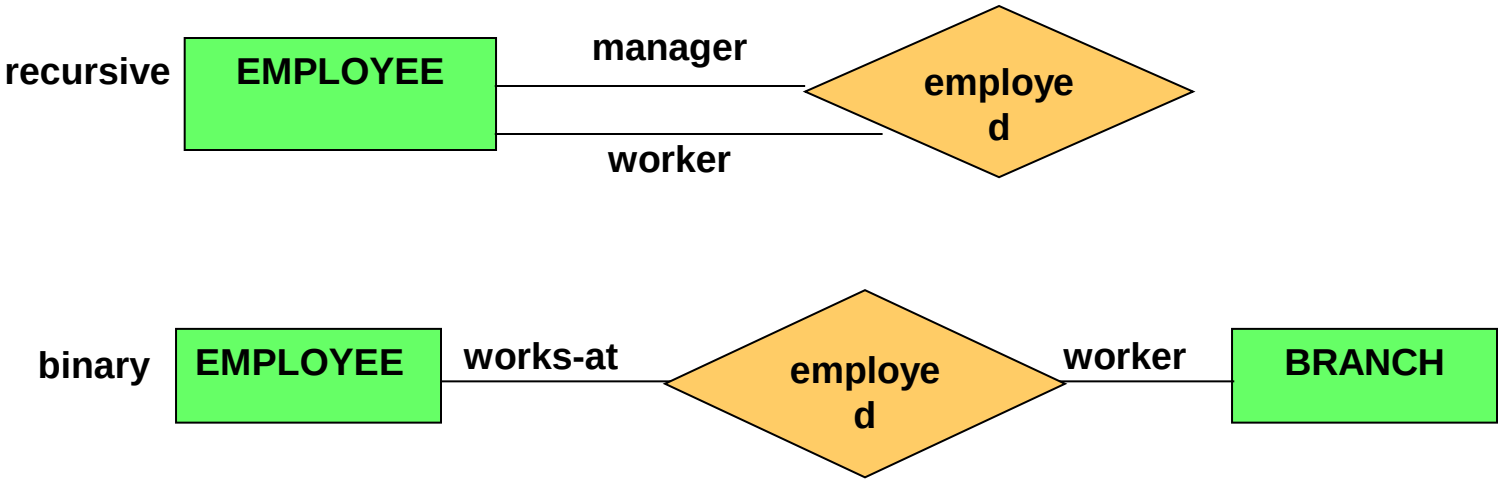
QUAN HỆ NHIỀU NGÔI (Cont.)

Chuyển quan hệ nhiều ngôi thành một tập các quan hệ 2 ngôi:



SƠ ĐỒ E-R VỚI CÁC CHỈ THỊ VAI TRÒ

- Các vai trò trong lược đồ E-R được biểu diễn bằng việc gán nhãn lên các đường kết nối giữa các tập thực thể và các tập quan hệ.
- Các vai trò này có thể được xác định cho các mối quan hệ đệ quy, nhị phân, và không nhị phân.



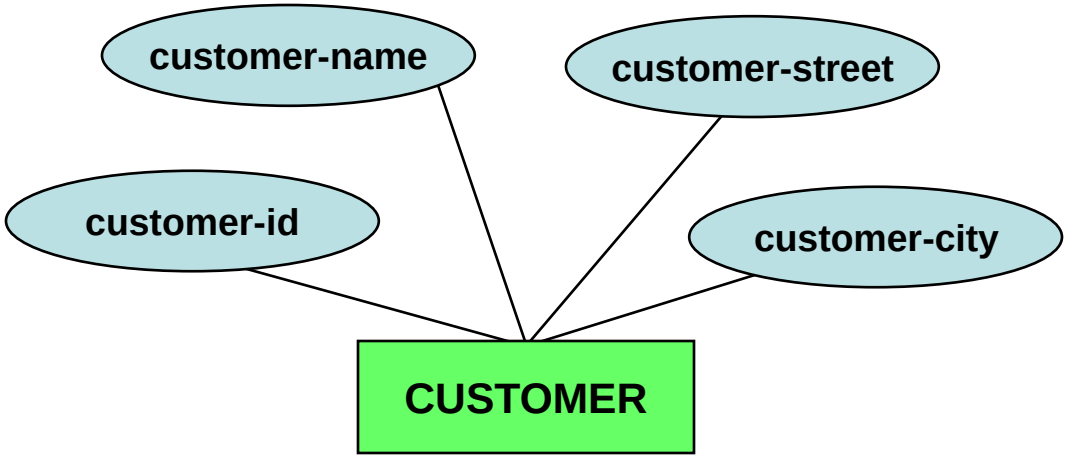
NGÔN NGỮ UML (Unified Modeling Language)

Một số các thành phần của UML:

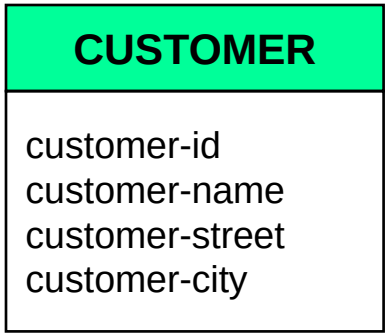
1. **Sơ đồ lớp**: Sơ đồ lớp tương tự như sơ đồ E-R.
2. **Sơ đồ use case**: được dùng để thể hiện sự tương tác giữa người dùng và hệ thống, các bước của công việc mà người dùng cần thực hiện (ví dụ, việc rút tiền từ một tài khoản ngân hàng hoặc việc đăng ký một khóa học).
3. **Sơ đồ hoạt động**: thể hiện luồng công việc giữa các thành phần trong hệ thống.
4. **Sơ đồ cài đặt**: thể hiện các thành phần của hệ thống và sự kết nối giữa chúng ở cả hai khía cạnh phần mềm và phần cứng.

SỰ TƯƠNG ỨNG GIỮA SƠ ĐỒ E-R VÀ SƠ ĐỒ LỚP UML

Các tập thực thể và thuộc tính



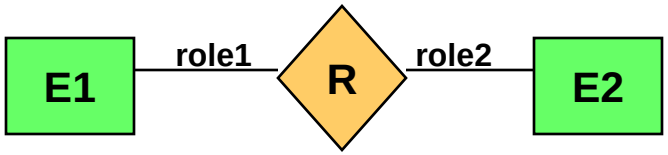
Sơ đồ E-R

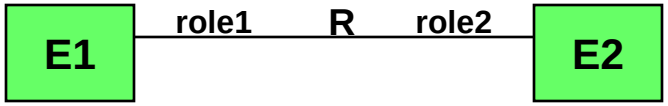


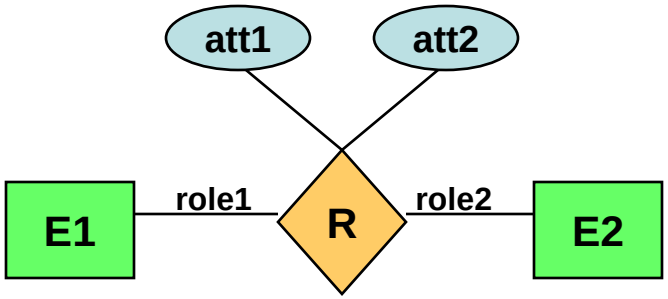
Sơ đồ lớp UML

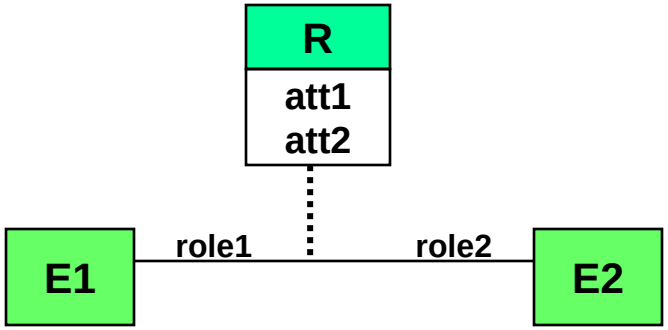
SỰ TƯƠNG ỨNG GIỮA SƠ ĐỒ E-R VÀ SƠ ĐỒ LỚP UML (Cont.)

Các quan hệ







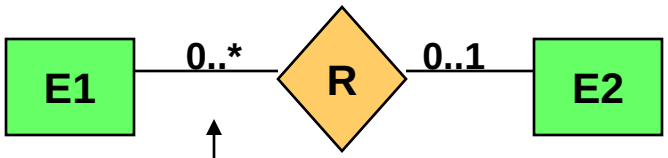


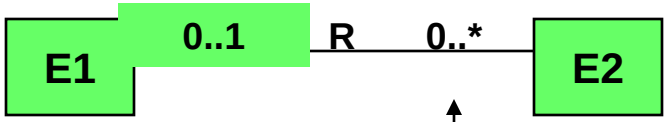
Sơ đồ E-R

Sơ đồ lớp UML

SỰ TƯƠNG ỨNG GIỮA SƠ ĐỒ E-R VÀ SƠ ĐỒ LỚP UML (Cont.)

Các ràng buộc về lực lượng liên kết





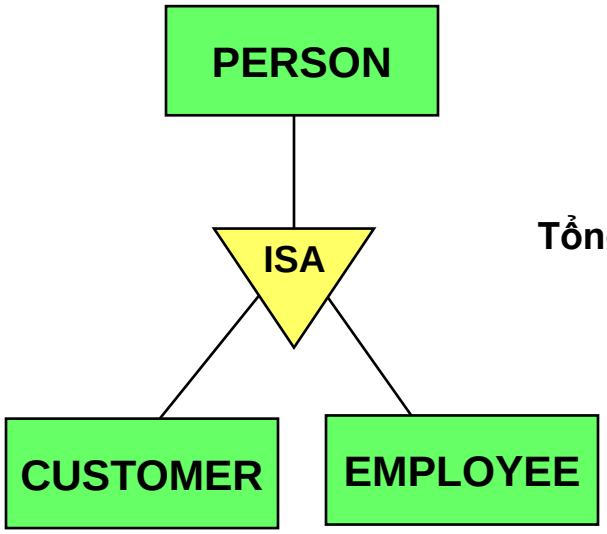
Chú ý: Vị trí các ràng buộc lực lượng liên kết là ngược nhau giữa 2 mô hình. Trong mô hình UML, ràng buộc 0..1 ở bên trái nghĩa là thực thể E2 có thể tham gia vào nhiều nhất một mối quan hệ, trong khi mỗi thực thể E1 có thể tham gia vào nhiều quan hệ; nói cách khác, quan hệ này là nhiều-một từ E2 tới E1.

Sơ đồ E-R

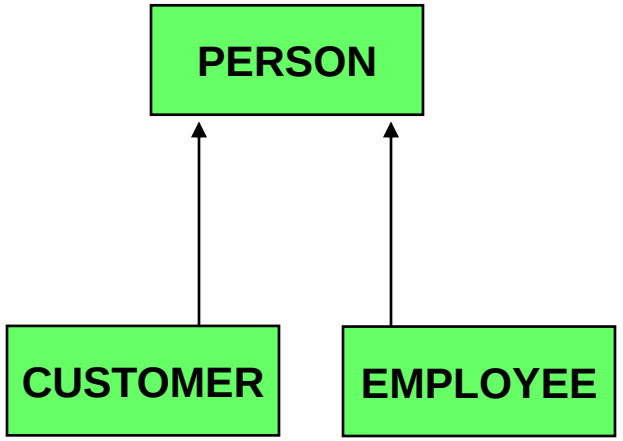
Sơ đồ lớp UML

SỰ TƯƠNG ỨNG GIỮA SƠ ĐỒ E-R VÀ SƠ ĐỒ LỚP UML (Cont.)

Tổng quát hóa và cụ thể hóa



Tổng quát hóa giao nhau

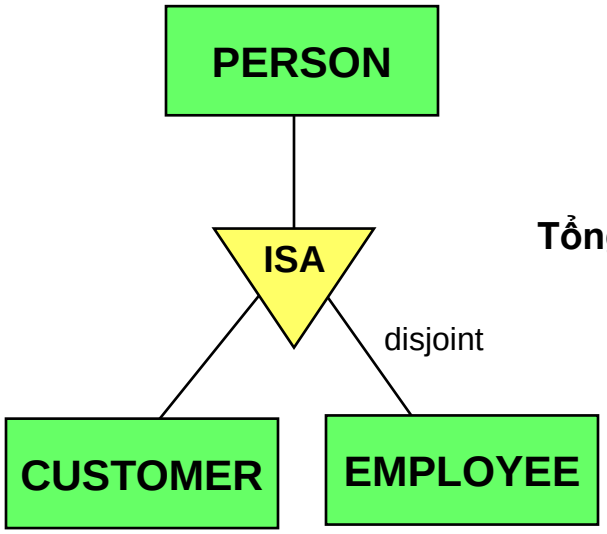


Sơ đồ E-R

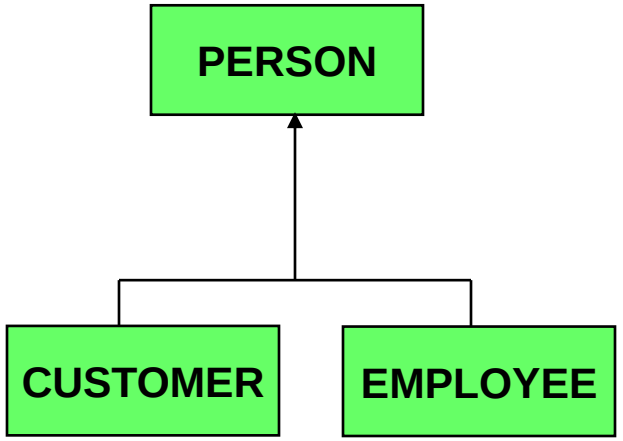
Sơ đồ lớp UML

SỰ TƯƠNG ỨNG GIỮA SƠ ĐỒ E-R VÀ SƠ ĐỒ LỚP UML (Cont.)

Tổng quát hóa và cụ thể hóa



Tổng quát hóa phân tách



Sơ đồ E-R

Sơ đồ lớp UML

RÀNG BUỘC TOÀN VẸN THAM CHIẾU

- Ràng buộc toàn vẹn tham chiếu có thể được hiểu đơn giản là việc đảm bảo một thuộc tính nào đó có một giá trị khác rỗng. Tuy nhiên, các ràng buộc toàn vẹn tham chiếu thường liên quan tới các quan hệ giữa các tập thực thể.

- Ràng buộc toàn vẹn tham chiếu yêu cầu mỗi thực thể "được tham chiếu tới" bởi một quan hệ phải tồn tại trong cơ sở dữ liệu.

RÀNG BUỘC TOÀN VẸN THAM CHIẾU (Cont.)

- Các phương pháp được sử dụng để đảm bảo tính ràng buộc toàn vẹn tham chiếu:
  - Không được phép xóa bỏ một thực thể được tham chiếu đến.
  - Nếu một thực thể được tham chiếu bị xóa bỏ thì tất cả các bản ghi tham chiếu tới thực thể đó cũng bị xóa.