Chapter5 Logical Database Design and the Relational Model

Kun Yue

April, 2009

Outline

- Introduction
- The relational data model
- Transforming EER diagrams into relations
- Normalization, Normal forms, relational schema decomposition
- Merging relations

Introduction

- **Logical database design**

  the conceptual data model →

  relational schemas as the logical data model

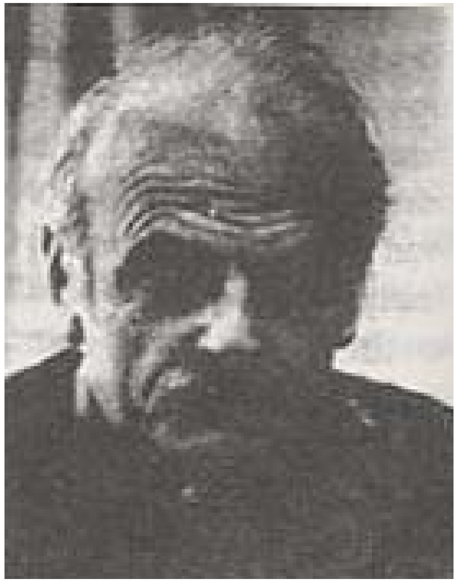
  - relational model

  - E-R and EER model → relational model

The objective of logical database design:

translate the conceptual design into a logical database design that can be implemented on a chosen DBMS.

The relational data model



- **History**

  - Relational data model (E .F. Codd, 1970)

  - System R

  - Ingres

  - Commercial RDBMS (1980)

- **Basic Definitions**
- **Integrity constraints**
- **Creating relational tables and well-structured relations**

Basic definitions (1)

- **3 components of the relational data model**

- Data structure: tables

- Data manipulation: operations (SQL)

- Data integrity: constraints

Relational schema

- **Relational data structure**

- Relation: a named, 2-dimensional table of data

- Relation —— table; Row——tuple, record; Column——attribute

- E.g. Employee1(<u>Emp_id</u>, Name, Dept_Name, Salary)

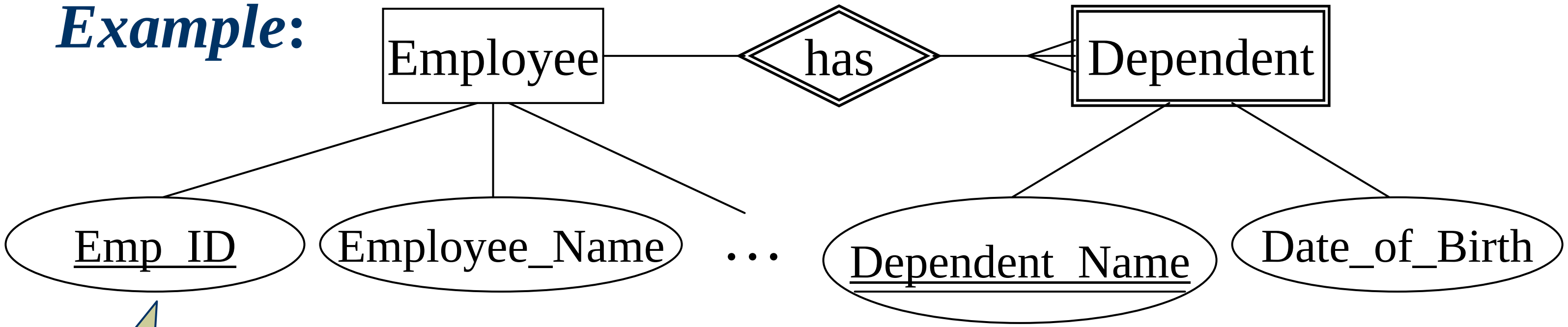
<u>**Emp_id**</u>	**Name**	**Dep_Name**	**Salary**
100	**Margaret**	**Marketing**	**48,000**
140	**Allen**	**Accounting**	**52,000**

Basic definitions (2)

- **Relational keys**

  - Primary key (identifier): an attribute (or composition of attributes) that uniquely identifies each row in a relation —— Entity integrity

***Example*:**



Attribute as primary key

The identifier of "*Dependent*":
Composition of attributes as primary key
Dependent (<u>Emp_id</u>, <u>Dependent_Name</u>, Data_of_Birth)

Basic definitions (3)

- **Relational keys**
  - Foreign key——Referential integrity

    ① An attribute in a relation of a database that serves as **the primary key attribute** of another relation in the same database

    ② Represents the relationship between two tables or relations

    ③ Imply the semantics of **contexts** and **relationships**

E.g., Department(Dept_Name, Location, Fax)

Employee1(Emp_id, Name, Dept_Name, Salary)

Dependent (Emp_id, Dependent_Name, Data_of_Birth)

"Dept_Name" is the foreign key of "Employee1"

"Emp_id" is the foreign key of "Dependent"

Basic definitions (4)

- **Properties of relations**

- Each relation (table) has a unique name
- No multivalued attributes in a relation (Atomic value)
- Each row is unique
- Each attribute has a unique name
- The sequence of columns and rows is insignificant

How to deal with the multivalued attributes?

Emp_id	Name	Dep_Name	Salary
100	Margaret	Marketing	48,000
140	Allen	Accounting	52,000

Emp_id	Name	Dep_Name	Salary
140	Allen	Accounting	52,000
100	Margaret	Marketing	48,000

Emp_id	Dep_Name	Name	Salary
100	Marketing	Margaret	48,000
140	Accounting	Allen	52,000

An example

Customer (Customer_id, Customer_name, Address, …)

Order (Order_id, Customer_id, Order_date)

Product (Product_id, Product_description, Product_finish, …)

Order Line (Order_id, Product_id, Quantity)

Whether "Order" is a strong entity or weak entity?

How to insert data into above tables?

Integrity constraints (1)

- **Domain constraints**

  The set of values that may be assigned to an attribute:

  domain name, meaning, data type, size, allowable values or range

- **Entity constraints**

  - No primary key attribute (or component of a primary key attribute) can be null

  - There do not exist any two tuples with the same primary key values

- **Referential constraints**

  - A rule that states either each foreign key value must match a primary key value in another relation or the foreign key value must be null

Integrity constraints (2)

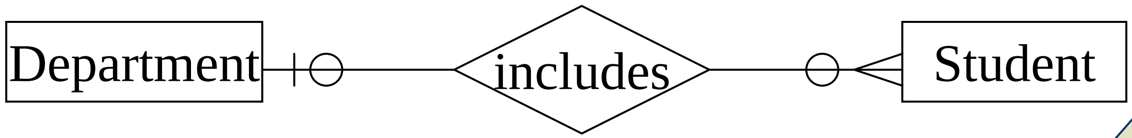
- **Referential constraints (Cont.)**

  - Identifying associations where referential integrity by means of the graphical representation using arrows in the schema
  - Foreign key and cardinality

  How do you know if a foreign key is allowed to be null?

Order (<u>Order_id</u>, Customer_id, Order_date)

Customer_id can not be NULL!



Student (<u>Student_id</u>, Student_name, Department_id, …)

Department_id can be NULL!

Integrity constraints (3)

- **Referential constraints (Cont.)**

 - The foreign key attributes must match the existing primary key attributes
 - The cardinality determines whether the foreign key can be null
 - Whether a foreign key can be null must be specified as a property of the foreign key attribute when the database is defined
 - The 3 choices when deleting the tuple that is referred due to foreign keys:
   ① Cascading deletion
   ② Prohibiting deletion
   ③ Place a null value in the foreign key

How to carry them out in practice?

Integrity constraints (4)

- **Referential constraints (Cont.)**

- Cascading deletion

Customer:

Customer_id	Customer_name	Address
C001	Mary	YNU
C002	John	FDU

How about the ***Order*** data referring to ***Customer*** "C001"?

Order:

Order_id	Customer_id	Order_date
O001	C001	2005-04-02
O002	C001	2005-04-05
O003	C002	2004-03-01

Integrity constraints (5)

- **Referential constraints (Cont.)**

  - Prohibiting deletion

Customer:

Customer_id	Customer_name	Address
C001	Mary	YNU
C002	John	FDU

How about the ***Order*** data referring to ***Customer*** "C001"?

Order:

Order_id	Customer_id	Order_date
O001	C001	2005-04-02
O002	C001	2005-04-05
O003	C002	2004-03-01

Integrity constraints (6)

- **Referential constraints (Cont.)**

  - Place a null value in the foreign key

Customer:

Customer_id	Customer_name	Address
C001	Mary	YNU
C002	John	FDU

Order:

Order_id	Customer_id	Order_date
O001	C001	2005-04-02
O002	C001	2005-04-05
O003	C002	2004-03-01

How about the ***Order*** data referring to ***Customer*** "C001"?

"Customer_id" as the foreign key of ***Order*** can not be NULL!

Well-structured relations (1)

- **Well-structured relations**

  - Contains minimal redundancy

  - Allows users to insert, modify and delete the rows in a table without errors or inconsistencies

- **Anomaly**

  - Why do these anomalies exist?
  - How to get the well-structured relations?

  - Insertion anomaly: Null primary key value, nonexistent referred tuple, etc.

  - Deletion anomaly: The tuple in the parent table is deleted, but the tuples in the child table still exist, etc.

  - Modification anomaly: multiple times of modifications

- To the extreme, all the data can just stored in one relation!

  —— Universal relational schema (Discussion☺)

Well-structured relations (2)

Emp_id	Name	Dept_name	Salary	Course_title	Date_completed
100	Simpson	Marketing	48,000	SPSS	6/19/200X
100	Simpson	Marketing	48,000	Surveys	10/7/200X
150	Martin	Marketing	42,000	Java	8/12/200X
…					

Analysis:

Insertion anomaly, deletion anomaly; modification anomaly?

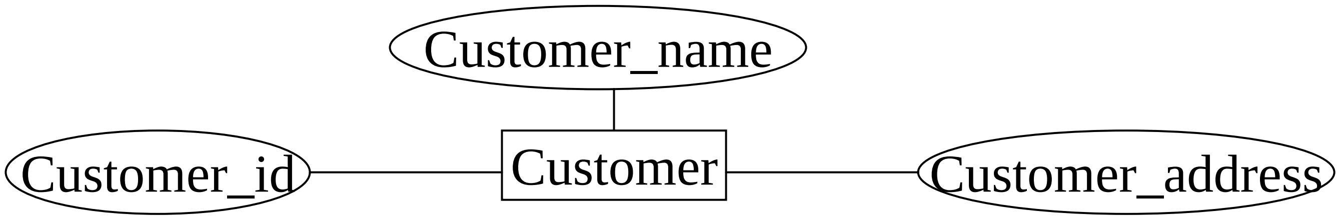
Transforming EER diagrams into relations

- Step1: Map regular entities
- Step2: Map weak entities
- Step3: Map binary relationships
- Step4: Map associative entities
- Step5: Map unary relationships
- Step6: Map ternary (and n-ary) relationships
- Step7: Map supertype/subtype relationships

Step1: Map regular entities (1)

① - name of entity type → name of the relation

- name of attribute of the entity → attribute of the relation

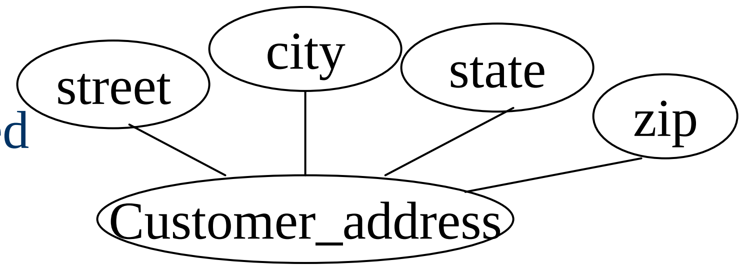


Customer

<u>Customer_id</u>	Customer_name	Customer_address

② Composite attributes

- only the simple components are included



<u>Customer_id</u>	Customer_name	street	city	state	zip

Step1: Map regular entities (2)

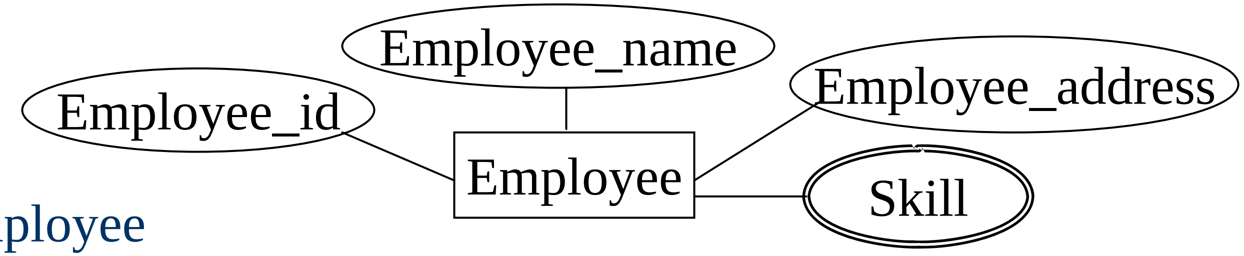
③ Multivalued attributes

2 relations:

1st relation: all the attributes except the multivalued attribute

2nd relation:

the primary key of the 1st relation (foreign key)+ multivalued attribute



Employee

<u>**Employee_id**</u>	**Employee_name**	**Employee_address**

Employee_skill

<u>**Employee_id**</u>	<u>**skill**</u>

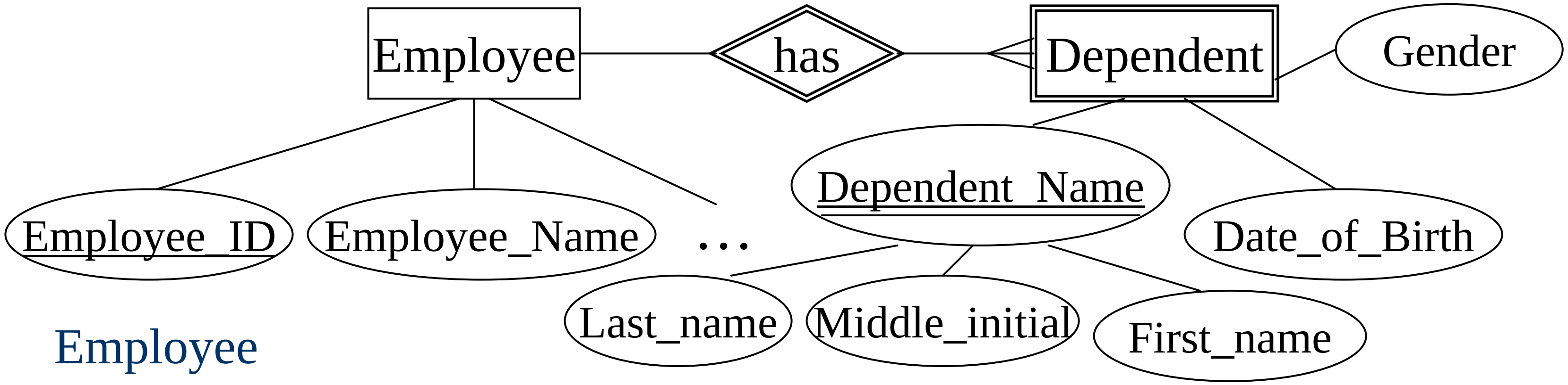
“skill” attribute can be extended

Step2: Map weak entities

Suppose the corresponding relation (strong entity) called identifying relation

① include all simple attributes of the weak entity

② primary key = primary key of the identifying relation + partial identifier



Employee

Employee_id	Employee_name	Employee_address

Dependent

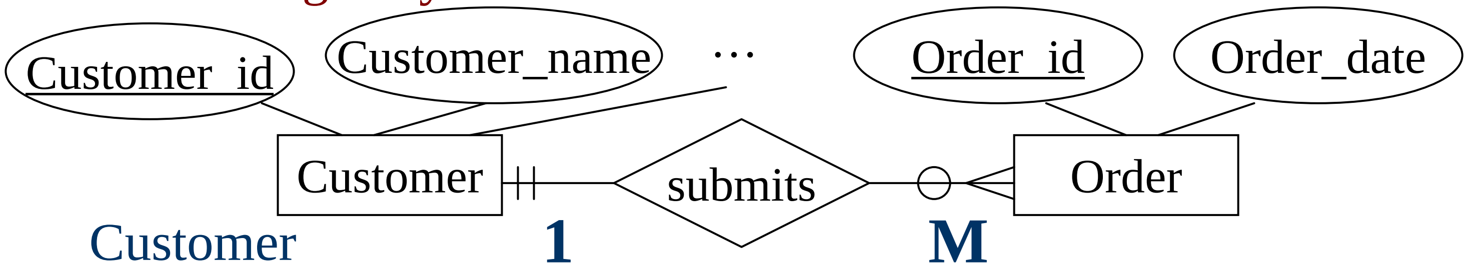
Employee_id	First_name	Middle_initial	Last_name	Date_of_Birth	Gender

Step3: Map binary relationships (1)

- **Map binary one-to-many relationships (1:M)**

① create a relation for each of the two participating entity types

② add the primary key of **One** side relation to the **Many** side relation as the foreign key



Customer

Customer_id	**Customer_name**	**street**	**city**	**state**	**zip**

Order

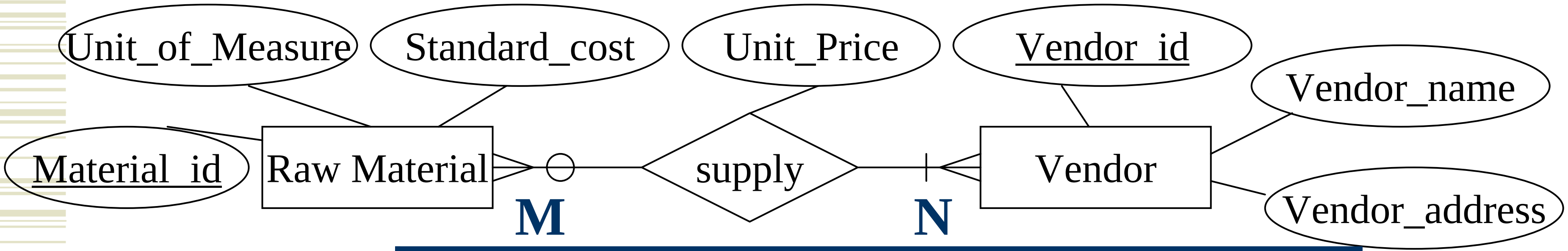
Order_id	**Customer_id**	**Order_date**

Step3: Map binary relationships (2)

- **Map binary many-to-many relationships (M:N)**

① create a relation for each of the two participating entity types: A, B

② create a new relation C, whose primary key is the combination of the primary keys of A and B, which are the foreign keys of C



Raw Material

Material_id	**Standard_cost**	**Unit_of_Measure**

Supply

Material_id	**Vendor_id**	**Unit_Price**

Vendor

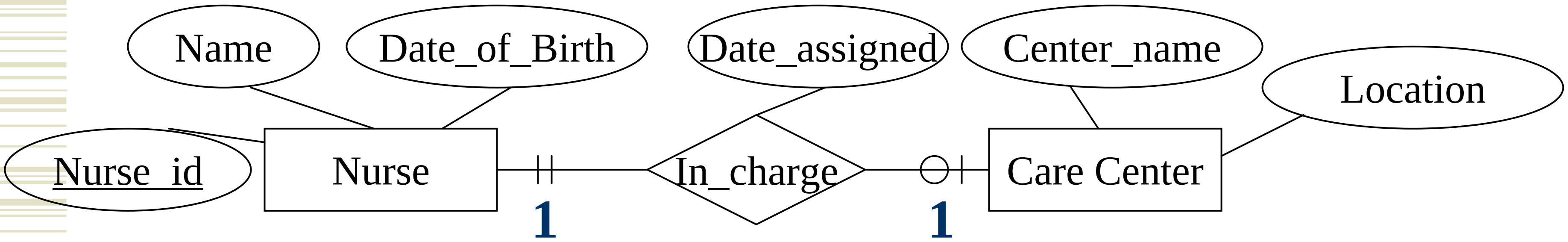
Vendor_id	**Vendor_name**	**Vendor_address**

Step3: Map binary relationships (3)

- **Map binary one-to-one relationships (1:1)**

① create a relation for each of the two participating entity types

② the primary key of one of the relations is included as a foreign key in the another relation



Nurse

Nurse_id	Name	Date_of_Birth

Care Center

Center_name	Location	Nurse_in_Charge	Date_assigned

Note: Include the primary key of mandatory side to the optional side

Step4: Map associative entities (1)

Similar to step3

- one relation for each of the two participating entity types
- the third for the associative entity

Whether an identifier is assigned to the associative entity?

① Identifier not assigned:

The default primary key: the two primary key attributes from the other 2 relations, also as the foreign key respectively

② Identifier assigned:

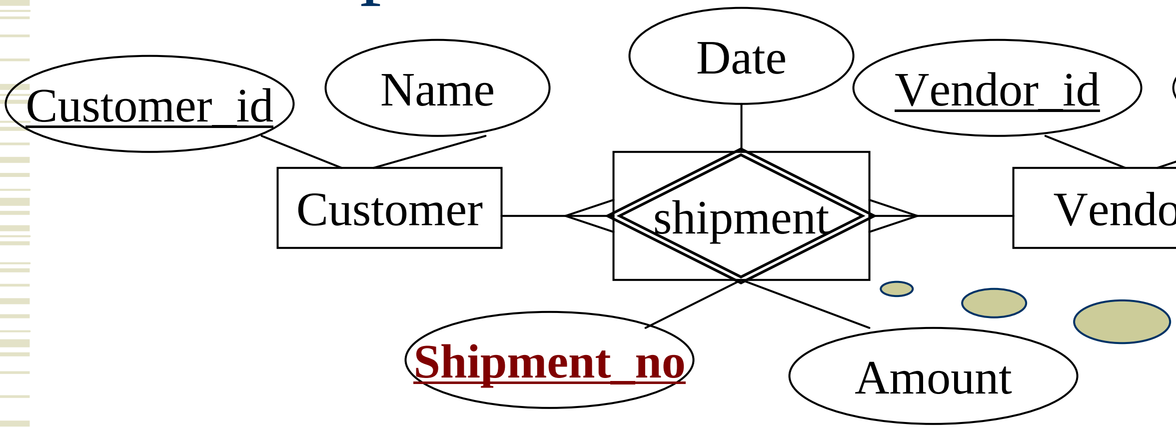
The default identifier may not uniquely identify instances of the associative entity

a) The identifier is the primary key of the associative entity

b) The primary keys for the 2 participating relations are the foreign keys in the associative relation

Step4: Map associative entities (2)

- Example



What's the case without Shipment_no?

Customer

Customer_id	Name	…

Shipment

Shipment_no	Customer_id	Vendor_id	Date	Amount

Vendor

Vendor_id	Address	…

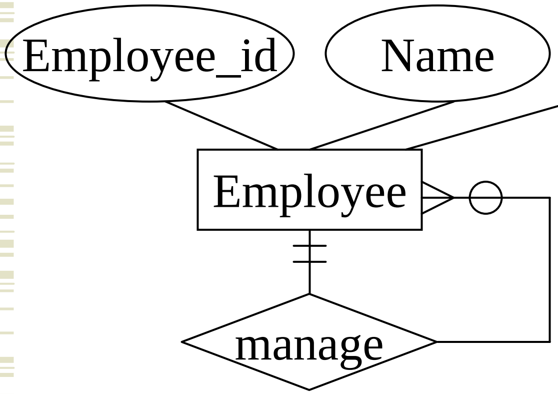
Step5: Map unary relationship (1)

- **Map unary one-to-many relationships:**

①map the entity type into relation

②a foreign key attribute is added within the **same** relation that references the primary key values

- the same domain as the primary key
- recursive foreign key



Can be null

Employee

<u>**Employee_id**</u>	**Name**	**Birthdate**	**Manager_id**

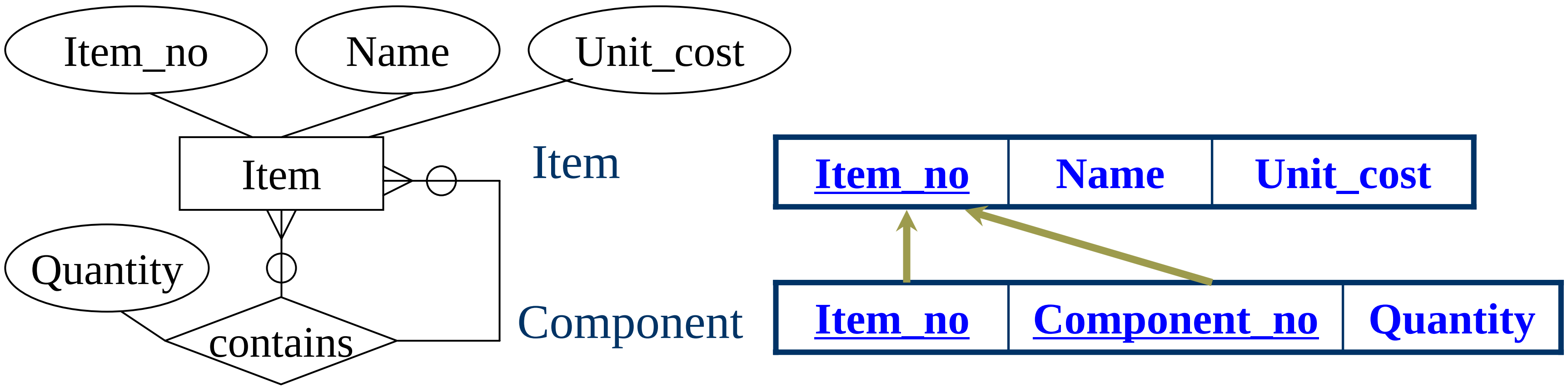
Step5: Map unary relationship (2)

- **Map unary many-to-many relationships (BOM):**

2 relations:

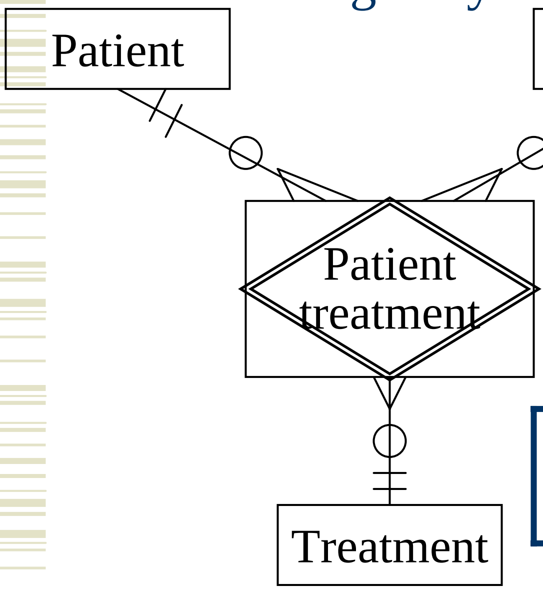
① one to represent the entity type in the relationship

② the other is an associative relation to represent the M:N relationship itself;
Primary key includes 2 attributes



Step6: Map ternary relationships

- Create one relation for each participating entity type
- Create a new associative relation:
  - The default primary key is the 3 primary keys of the 3 relations, also as the foreign keys respectively (maybe there are some additional ones)



Patient

Patient_id	**Patient_name**

Physician

Physician_id	**Physician_name**

Patient Physician Treatment

Patient _id	**Physician _id**	**Treatment_ id**	**Date**	**Time**	**Results**

Treatment

Treatment_id	**Description**

Step7: Map supertype/subtype relationships (1)

- Create a separate relation for the supertype and each of its subtypes, including the corresponding attributes
- Assign to the relation for each subtype the primary key of the supertype (also as the foreign keys respectively)
- Assign one or more attributes of the supertype as the subtype discriminator

Employee

Employee_ number	Employee_ name	Address	Employee_ type	Date_hired

Hourly_Employee

H_Employee_number	Hourly_rate

Salaried_Employee

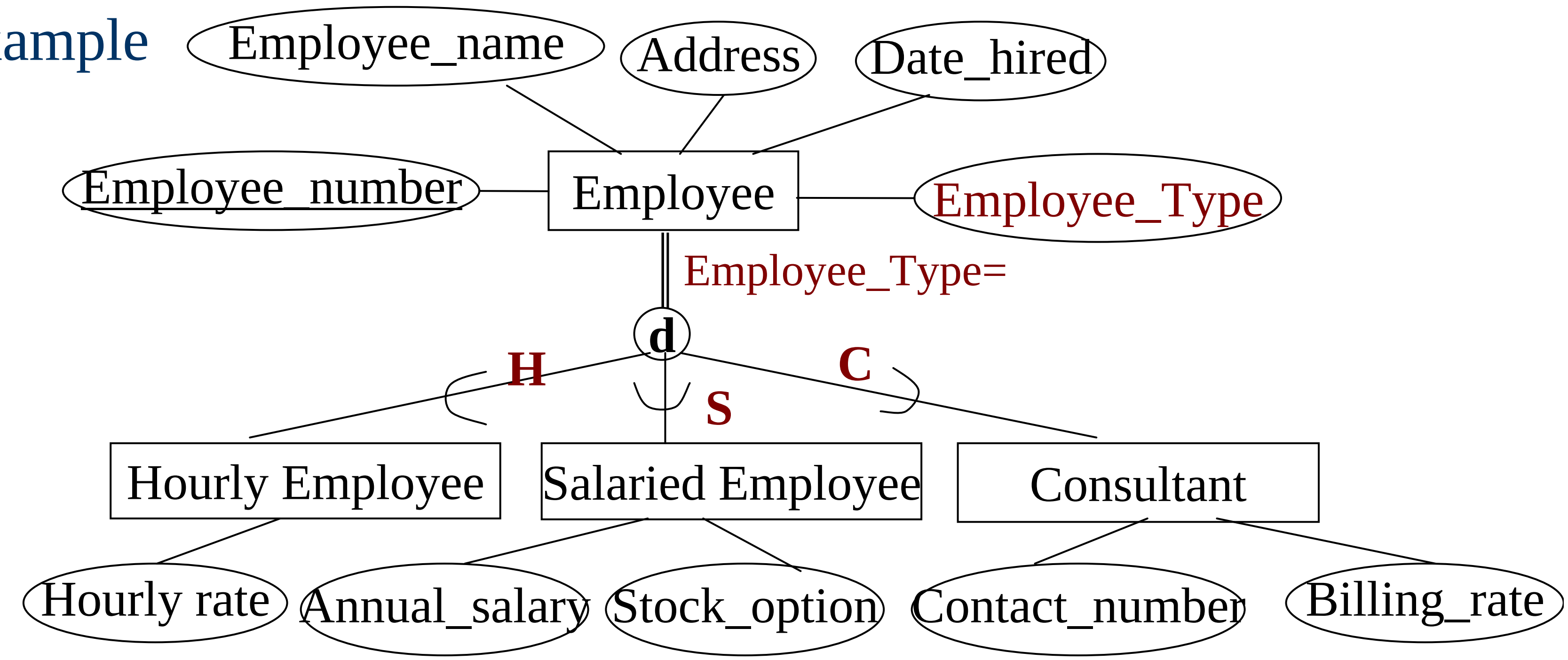
S_Employee_number	Annual_Salary	Stock_Options

Consultant

C_Employee_number	Contract_ Number	Billing_Rate

Step7: Map supertype/subtype relationships (2)

Example



How to get all the information for " Salaried Employee"?

Introduction to normalization (The theory of database design)

- Motivation
- Preliminaries
- The basic normal forms

Motivation:

- To get the well-structured relations
- To avoid unnecessary duplication and redundancy of data
- Decompose the relations and give the formal process
- …

Normalization: the process of decomposing relations ***with anomalies*** to produce smaller, well-structured relations

Preliminaries and basic concepts (1)

- **Normal forms**

First normal form, second normal form, third normal form, BC normal form, …

- **Functional dependencies(FDs)**

- A constraint between two attributes or two sets of attributes

- A → B

① The value of A uniquely determines the value of B.

② Attribute B is functionally dependent on attribute A.

③ If the values on A are equal, then the values on B must be equal

E.g, Sno → Sname, Cno → Cname, Sno,Cno → Grade,

ISBM → Title, First_ Author_Name

③ The attribute may be functionally dependent on more than one attributes

④ Determinants: the attribute on the left-hand size of the arrow in a FD

Preliminaries and basic concepts (2)

- **Candidate keys and primary keys**
  - An attribute, or combination of attributes that **uniquely identifies** a row in a relation

    ① Unique identification

    ② Nonredundancy

    E.g, Enroll(Sno, Cno, Grade)

    ◇ Sno → Grade? Cno → Grade? Sno, Cno → Grade?

    ◇ Either Sno or Cno can not respectively identify an instance in ***Enroll*** uniquely——The minimal attribute set as identifier

    E.g, Shipment(Shipment_no, Customer_id, Vendor_id, Date, Amount)

    ◇ Shipment_no → Date, Amount Customer_id, Vendor_id → Date, Amount

    ◇ Both Shipment_no and (Customer_id, Vendor_id) can be the unique identifier——Candidate keys☺!

Preliminaries and basic concepts (3)

- **Primary key (PK)**

  - The chosen candidate key (only one)

  - Multiple candidate keys, unique primary key

  E.g, Either Shipment_no or (Customer_id, Vendor_id) be chosen.

- **Nonkey attributes**

  Attributes except the key attributes

  E.g, Shipment(<u>Shipment_no</u>, Customer_id, Vendor_id, Date, Amount)

  ◇ nonkey attributes? Date, Amount

- **Candidate key revisited**

  uniquely identifies the nonkey attributes (**all?**) in a relation

Preliminaries and basic concepts (4)

- **Some concepts of FDs**

  E.g, Employee2(<u>Emp_id</u>, Name, Dept_name, Salary, <u>Course_title</u>, Date_completed)

  ◇ Primary key: Emp_id, Course_title

  ◇ Emp_id → Name, Dept_name, Salary

  ◇ Emp_id, Course_title → Date_completed

Part of the PK as determinant

- **Partial functional dependencies**

  One or more nonkey attributes are functionally dependent on part of the PK

- **Full functional dependencies**

  All nonkey attributes are functional dependent on the PK

  E.g, Shipment_no → Date, Amount

Preliminaries and basic concepts (5)

- **Some concepts of FDs (Cont.)**

 - **Trivial functional dependencies**: X→Y, and Y⊆X

 E.g, Shipment_no → Shipment_no,

 Customer_id, Vendor_id → Customer_id

 - **Nontrivial functional dependencies** (Generally)

 - **Transitive functional dependencies**

 E.g, Sales(Cust_id, Name, Salesperson, Region)

 ◇ Cust_id → Name, Salesperson, Region

 ◇ Salesperson → Region

 ◇ Cust_id → Salesperson, Salesperson → Region

 Region transitively dependent on Cust_id

 Between two or more nonkey attributes, such as A → B, B → C

Preliminaries and basic concepts (6)

- What results will be produced due to the partial and transitive FDs?

Insertion, deletion, modification anomalies☹

E.g, The insertion and deletion anomaly with partial FD

Emp_id	Name	Dept_name	Salary	Course_title	Date_completed
100	Simpson	Marketing	48,000	SPSS	6/19/200X
100	Simpson	Marketing	48,000	Surveys	10/7/200X
150	Martin	Marketing	42,000	Java	8/12/200X
…					

- How to insert a new course "C++"?
- What will be resulted when a course is deleted?

The basic normal forms (1)

- **First normal form**

  The relation contains no multivalued attributes

- **Second normal form**

  - First normal form
  - Every nonkey attribute is fully functionally dependent on the primary key

  E.g, Employee2(Emp_id, Name, Dept_name, Salary, Course_title, Date_completed)

  ◇ Emp_id → Name, Dept_name, Salary
  Emp_id, Course_title → Date_completed

  ◇ Decomposition:
  Employee1(Emp_id, Name, Dept_name, Salary)
  Emp_Course(Emp_id, Course_title, Date_completed)

The basic normal forms (2)

- **Third normal form**

  - Second normal form 3NF ⊆ 2NF ⊆ 1NF

  - No transitive FDs present

  E.g, Sales(Cust_id, Name, Salesperson, Region)

  ◇ Cust_id → Name, Salesperson, Region

  ◇ Salesperson → Region

  ◇ Decomposition:

  Sales1(Cust_id, Name, Salesperson)

  Sperson(Salesperson, Region)

E.g, Shipment(Snum, Origin, Destination, Distance)

◇ Snum → Origin, Destination, Distance

◇ Origin, Destination → Distance

Shipto(Snum, Origin, Destination)

Distances(Origin, Destination, Distance)

Discussion

- The seven steps of transforming EER diagram into relations are the experiential rules of logical database design
- The normalization and normal forms are the backbones in database design theory
- The seven steps of mapping are consistent with the normalization theory
- We can decompose a universal relational schema into well-structured ones based on the normal form decomposition algorithms
- How to perform the formal decomposition process?

***FDs are the most important thing in normalization*!**

How to get the enough FDs on the given relations?

Merging relations (1)

- **Prerequisite**

  - Not the realm of database design theory

  - From the viewpoint of database system development and view, information integration

- **Motivation**

  - The work of several subteams comes together
  - integrating existing databases, merging tables
  - new data requirements arise

- **Example**

  - Combine(merge) the relations that have the same PK

  Employee1(<u>Employee_id</u>, Name, Address, Phone)
  Employee2(<u>Employee_id</u>, Name, Address, Jobcode, No_Years

  - Result of merging:

  Employee2(<u>Employee_id</u>, Name, Address, Phone, Jobcode, No_Years

Merging relations (2)

- **View integration problems**

- Synonyms

Two or more attributes describing the same characteristics of an entity:

Different names but the same meaning

E.g, Student1(Student_id, Name), Student2(Student_no, Name, Address)

Merging result: Student(SSN, Name, Address)

- Homonyms

An attribute that may have more than one meaning in various contexts

E.g, Student1(Student_id, Name, Address),

Student2(Student_id, Name, Phone_no, Address)

Merging result with specific prefixes:

Student(Student_id, Name, Campus_address, Permanent_address)

Merging relations (3)

- **Transitive FDs after the merging**

E.g, Student1(Student_id, Major),
Student2(Student_id, Advisor) ⇒
Student (Student_id, Major, Advisor)

- ◇ Student_id→Major, Major → Advisor
- ◇ Transitive FD is resulted
- ◇ Decomposition after the merging

Student(Student_id, Major)
Major Advisor(Major, Advisor)

Some other principles in the logical database design

- **Key definition**

  - A primary key whose value is unique across all relations

  - Similar to the object identifier

  - PK maybe has no business meanings

- **Attribute definition**

  - Some extra attributes are necessary in case of database evolution

  - …

- **The normalization is effectively used to check the transformation results from EER diagrams, and some complementary revision is doomed**

Summary

- The relational data model
  - Definitions
  - Constraints: domain, entity, referential integrity (PK and FK)
  - Well-structured relation, anomalies
- Major steps in the logical database design based on transforming EER diagrams to normalized relations
  - Transform EER diagrams to relations (7 steps)
  - Normalize the relations (FDs, normal forms, schema decomposition)
  - Merge the relations (Synonyms, Homonyms)

Spend more time on this chapter☺!

Assignments (1)

- **Define and contrast the following terms:**

(1) Candidate key; Primary key; Foreign key

(2) Entity integrity constraint; Referential integrity constraint

(3) Partial functional dependency; Transitive functional dependency

- **Describe the following concepts, as well as related practical examples (motivation, concept, and application):**

(1) Anomalies that arise in a table

(2) Foreign key, and referential integrity constraint

(3) Normalization and relational schema decomposition

Assignments (2)

- Page 203, Review questions 17
- Page 203, Problems and exercises 3: (c), (d)
- Page 204, Problems and exercise 5: (a)
- Page 204, Problems and exercises 6: (a), (b), (c)
- Page 204, Problems and exercises 7: (a), (b), (c), (d)

The end

Thanks!