Review

Course Outline

Lecture	Content
Database System Concepts & Architecture	-Introduction to Data Models, Database Systems -Three-Level Architecture & Data Independence -Modern Database Applications
Entity-Relationship (ER) Model	-ER Model -Introduction to Enhanced ER (EER) Model
Relational Model	-Relational Data Model -ER- & EER-to-Relational Mapping -Relational Algebra
SQL	-Data Definition Language (DDL) -Data Manipulation Language (DML) -Introduction to Triggers & Stored Procedures
Database Design Theory & Methodology	-Functional Dependencies & Normalization -Relational Database Design: Algorithms
Data Storage, Indexing	-Data Storage -Hashing & Indexing Structures
Database Security	-Discretionary & Mandatory Access Control (DAC & MAC) -Flow Control, Inference Problem -Security Issues in Modern Data Management Systems
Emerging Technologies & Applications	- Presentations

Course Outline

Lecture	Content
Database System Concepts & Architecture	-Introduction to Data Models, Database Systems -Three-Level Architecture & Data Independence -Modern Database Applications
Entity-Relationship (ER) Model	-ER Model -Introduction to Enhanced ER (EER) Model
Relational Model	-Relational Data Model -ER- & EER-to-Relational Mapping -Relational Algebra
SQL	-Data Definition Language (DDL) -Data Manipulation Language (DML) -Introduction to Triggers & Stored Procedures
Database Design Theory & Methodology	-Functional Dependencies & Normalization -Relational Database Design: Algorithms
Data Storage, Indexing, Query Processing & Physical Design	-Hashing & Indexing Structures (B-tree & R-tree families) -Physical Database Design
Database Security	-Discretionary & Mandatory Access Control (DAC & MAC) -Flow Control, Inference Problem -Security Issues in Modern Data Management Systems
Emerging Technologies & Applications	-Introduction to XML, Data Mining & Data Warehousing, GIS, M-Commerce & LBS -Emerging Database Technologies & Applications

Outline

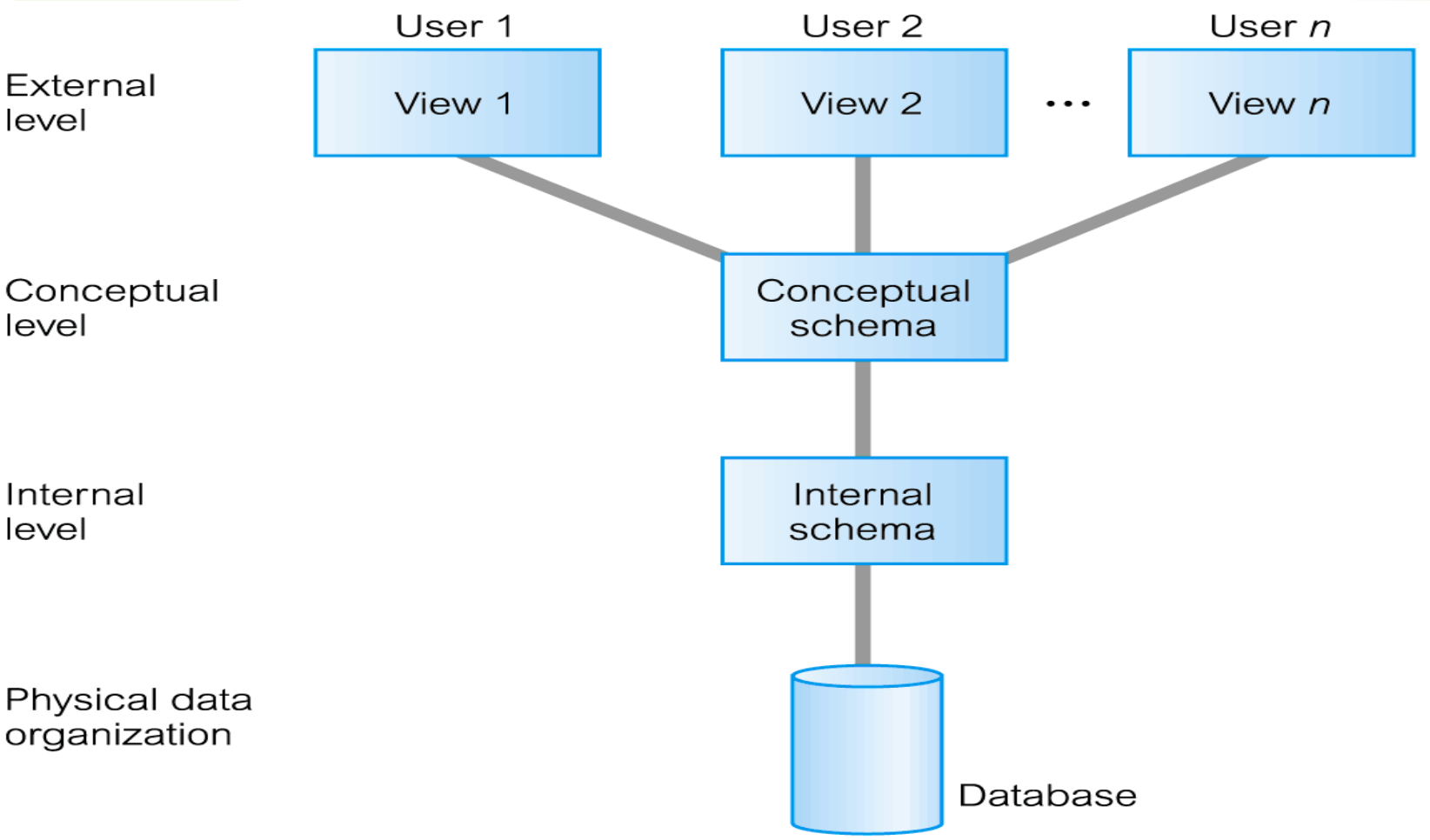
- File-based Approach
- Database Approach
- Three-Schema Architecture and Data Independence
- Database Languages
- Data Models, Database Schema and Database State
- Data Management Systems Framework

File-based Approach and Database Approach

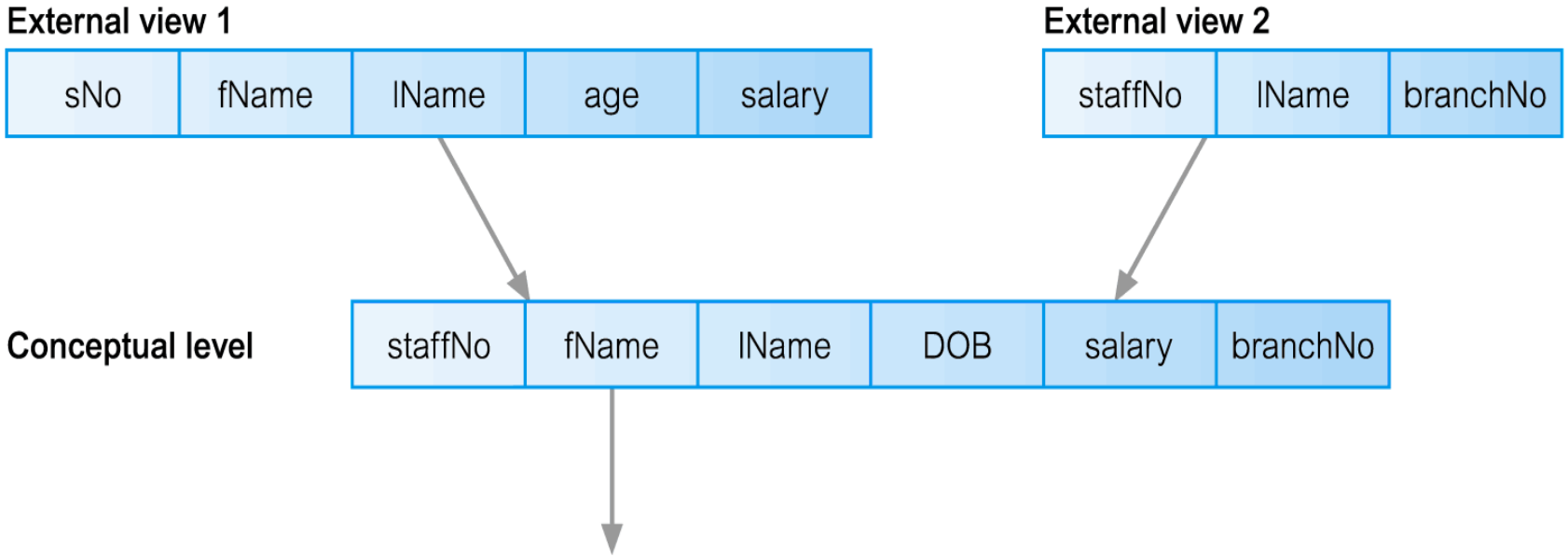
- File-based approach:
  - Problems
  - Shared file approach
- Database approach
  - DBMS
  - Characteristics of the Database Approach

Three-Schema Architecture and Data Independence

- **Three-level architecture and data independence**



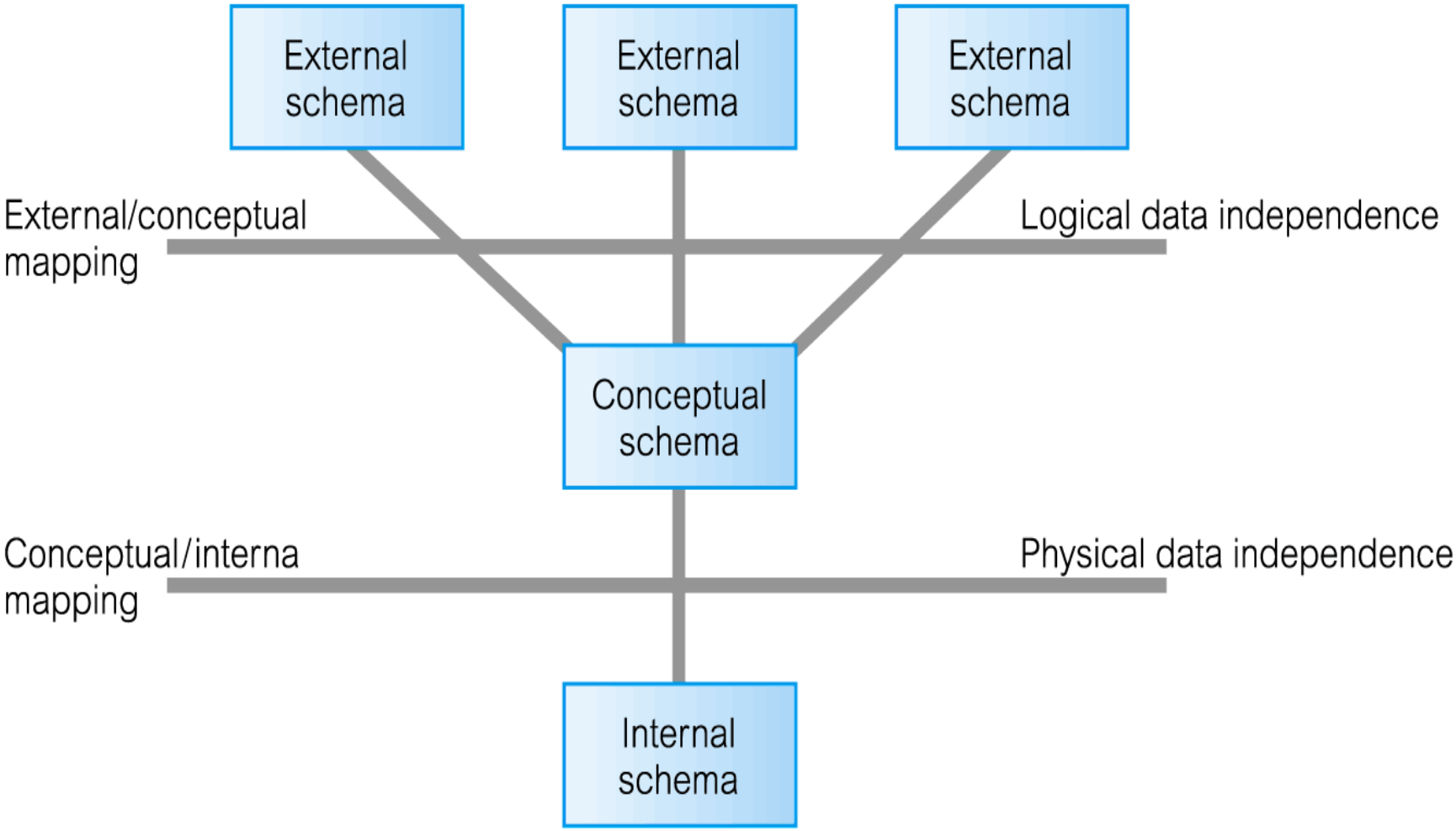
Three-Schema Architecture and Data Independence



Internal level

```
struct STAFF {
      int staffNo;
      int branchNo;
      char fName [15];
      char lName [15];
      struct date dateOfBirth;
      float salary;
      struct STAFF *next;                  /* pointer to next Staff record */
};
index staffNo; index branchNo;            /* define indexes for staff */
```

Three-Schema Architecture and Data Independence



Database Languages

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Data Control Language (DCL)

Data Models, Database Schema and Database State

- Data Model
- Database Schema
- Schema Diagram
- Database State (Snapshot)

Course Outline

Lecture	Content
Database System Concepts & Architecture	-Introduction to Data Models, Database Systems -Three-Level Architecture & Data Independence -Modern Database Applications
Entity-Relationship (ER) Model	-ER Model -Introduction to Enhanced ER (EER) Model
Relational Model	-Relational Data Model -ER- & EER-to-Relational Mapping -Relational Algebra
SQL	-Data Definition Language (DDL) -Data Manipulation Language (DML) -Introduction to Triggers & Stored Procedures
Database Design Theory & Methodology	-Functional Dependencies & Normalization -Relational Database Design: Algorithms
Data Storage, Indexing, Query Processing & Physical Design	-Hashing & Indexing Structures (B-tree & R-tree families) -Physical Database Design
Database Security	-Discretionary & Mandatory Access Control (DAC & MAC) -Flow Control, Inference Problem -Security Issues in Modern Data Management Systems
Emerging Technologies & Applications	-Introduction to XML, Data Mining & Data Warehousing, GIS, M-Commerce & LBS -Emerging Database Technologies & Applications

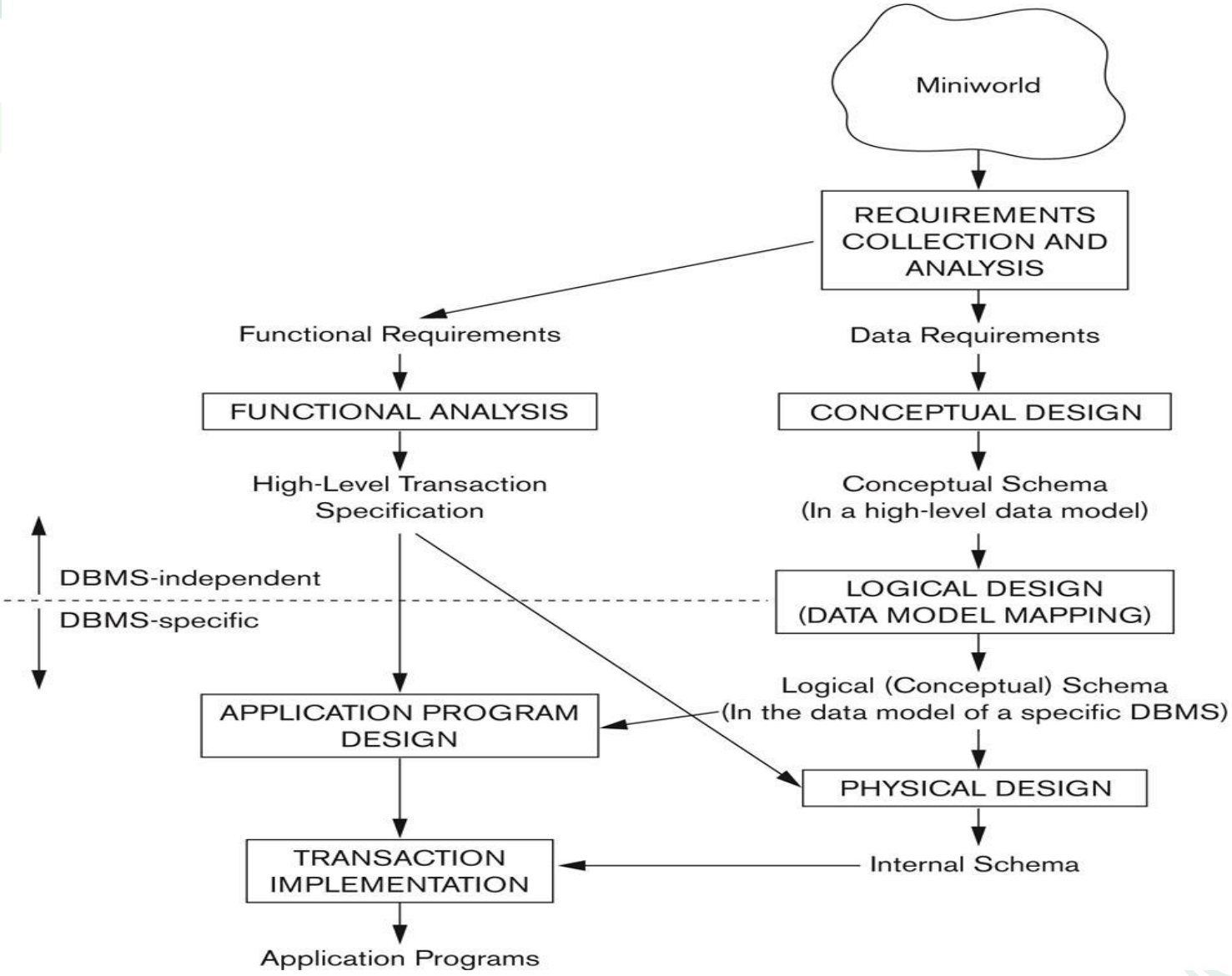
Outline

- What is ER Model? And Why?
- Overview of Database Design Process
- Example COMPANY Database
- ER Model Concepts
- ER Diagram
- Alternative Diagrammatic Notations
- Problems with ER Models

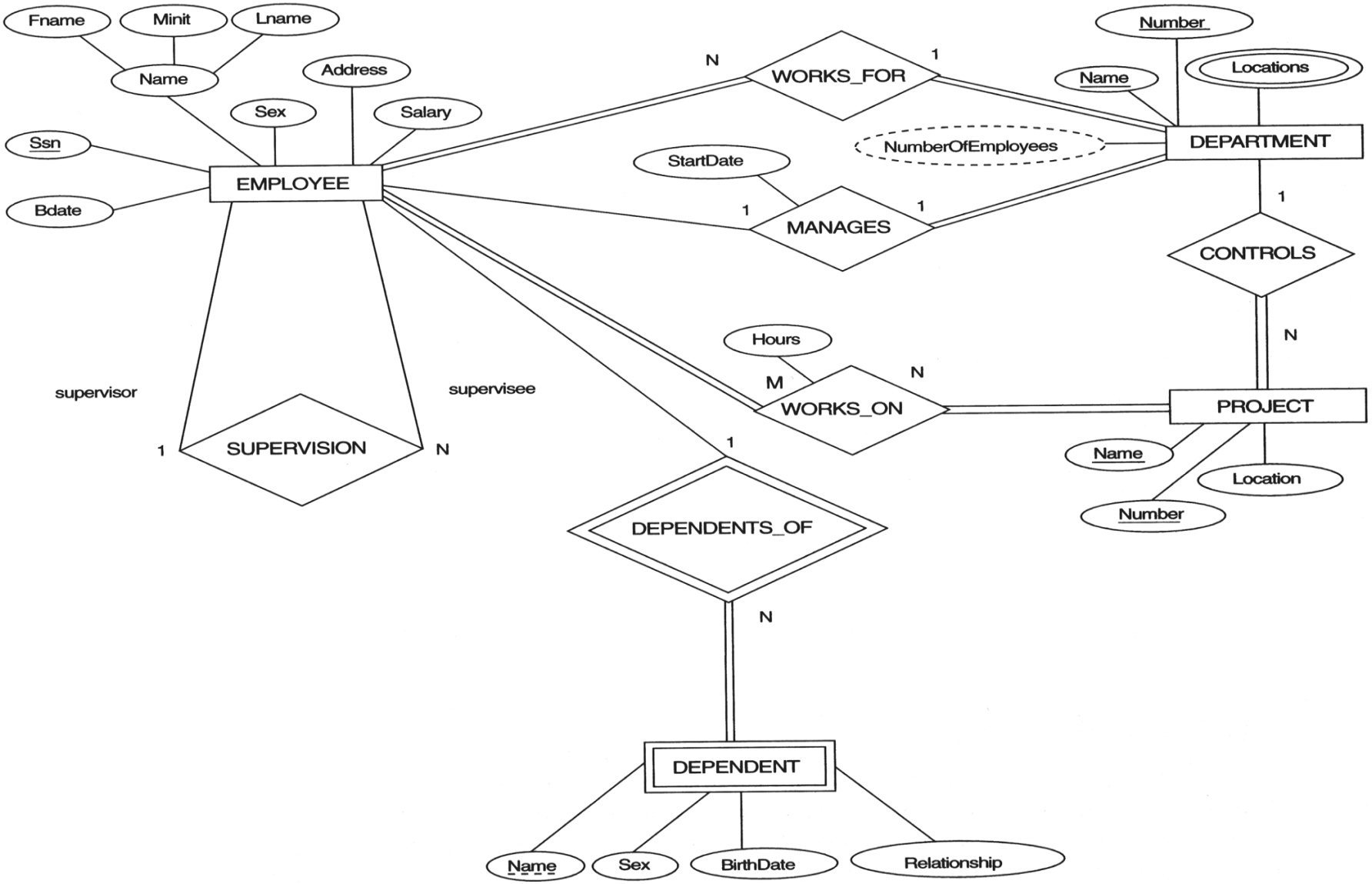
What is ER Model? And Why?

- ER model is a logical organisation of data within a database system
- ER model technique is based on relational data model
- Why use ER data modelling

Overview of Database Design Process



Example COMPANY Database



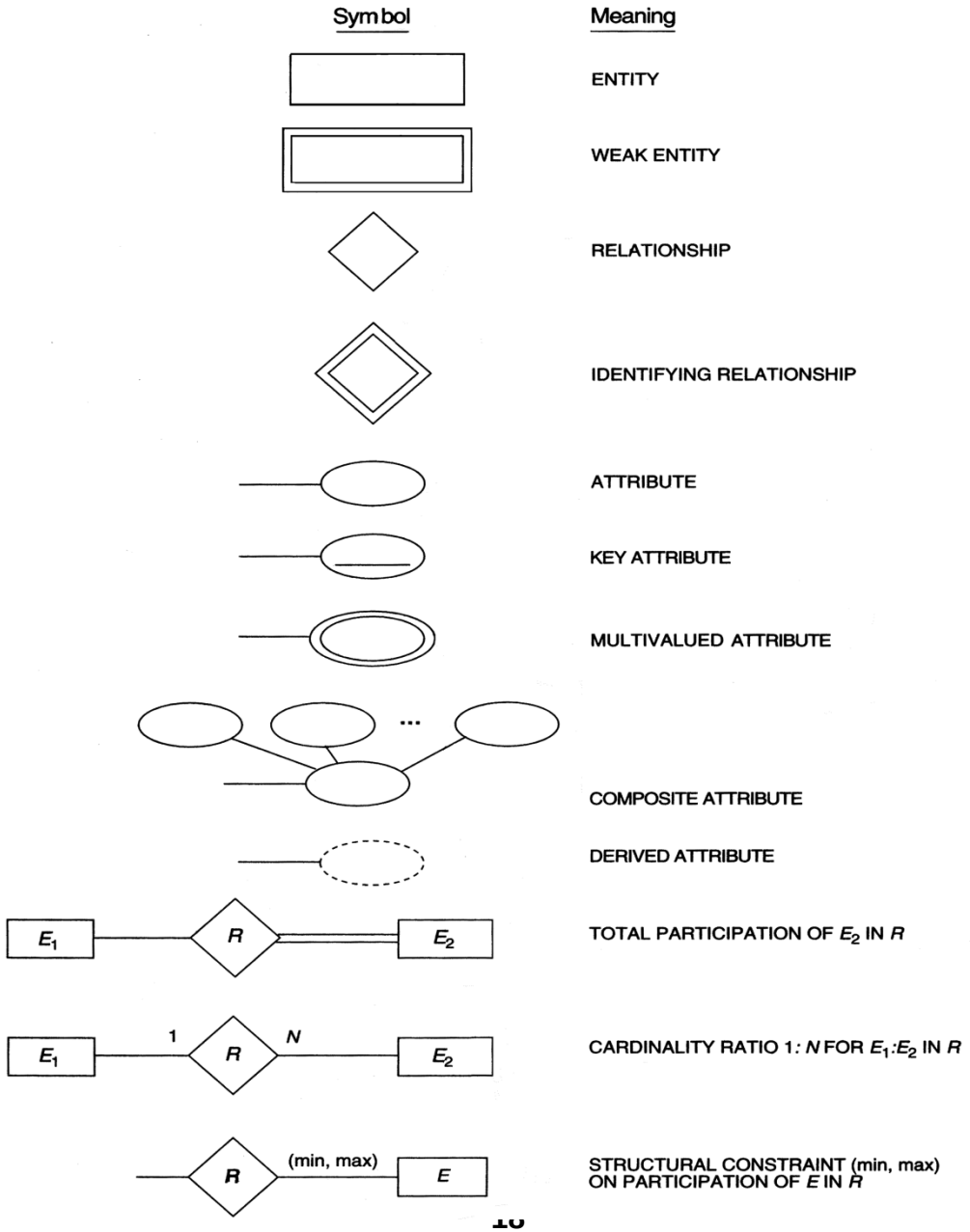
ER Model Concepts

- Entities and Attributes
- Types of Attributes:
  - Simple
  - Composite
  - Multi-valued
  - Derived Attribute
- Entity Types and Key Attributes
- Relationships and Relationship Types
- Weak Entity Types
- Recursive relationships

ER Model Concepts

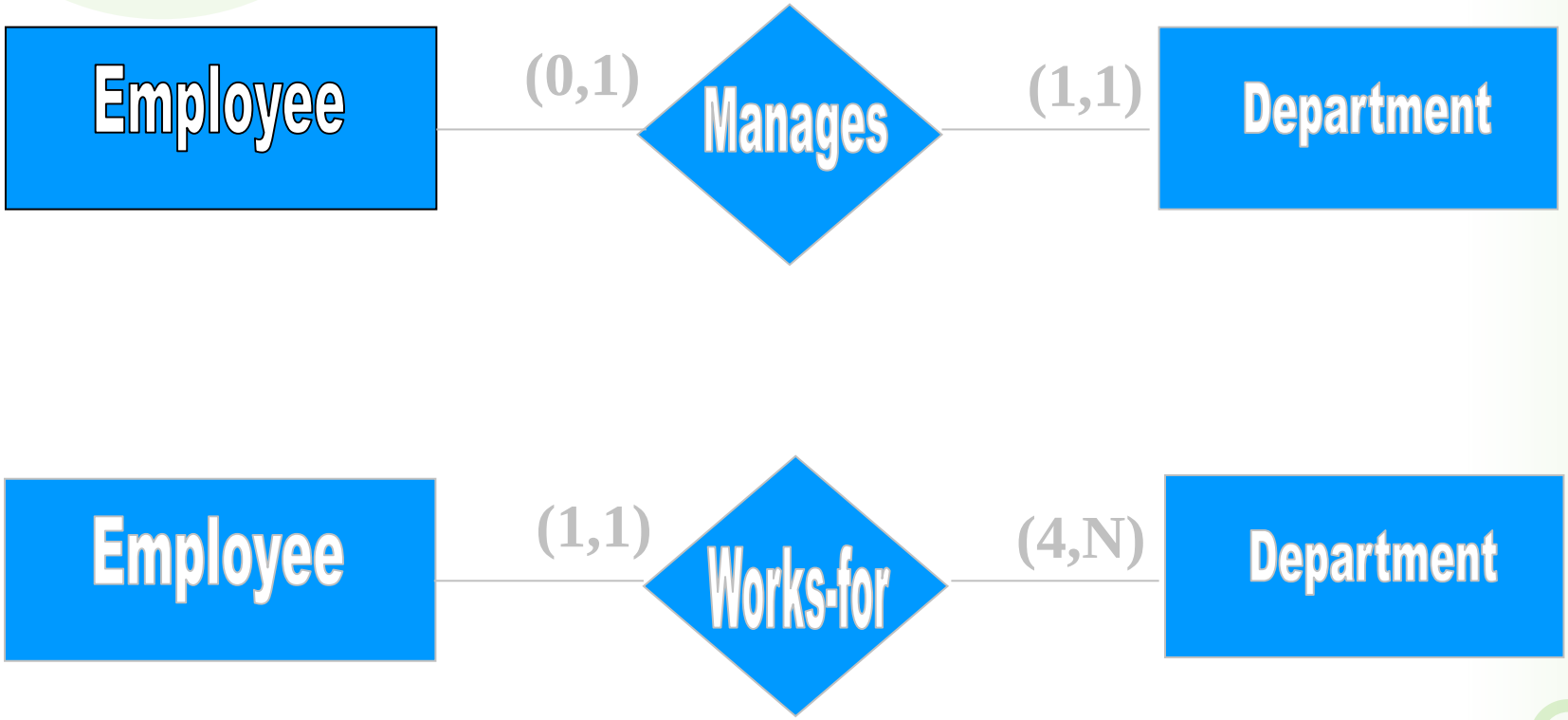
- **Structural constraints:** one way to express semantics of relationship: cardinality ratio and membership class
- **Cardinality ratio (functionality):** It specifies the number of relationship instances that an entity can participate in a binary relationship
  - one-to-one (1:1)
  - one-to-many (1:M) or many-to-one (M:1)
  - many-to-many (M:N)
- **Membership class (participation constraint)**:
  - Mandatory (total participation)
  - Optional (partial participation)

Summary of the Notation for ER Diagrams

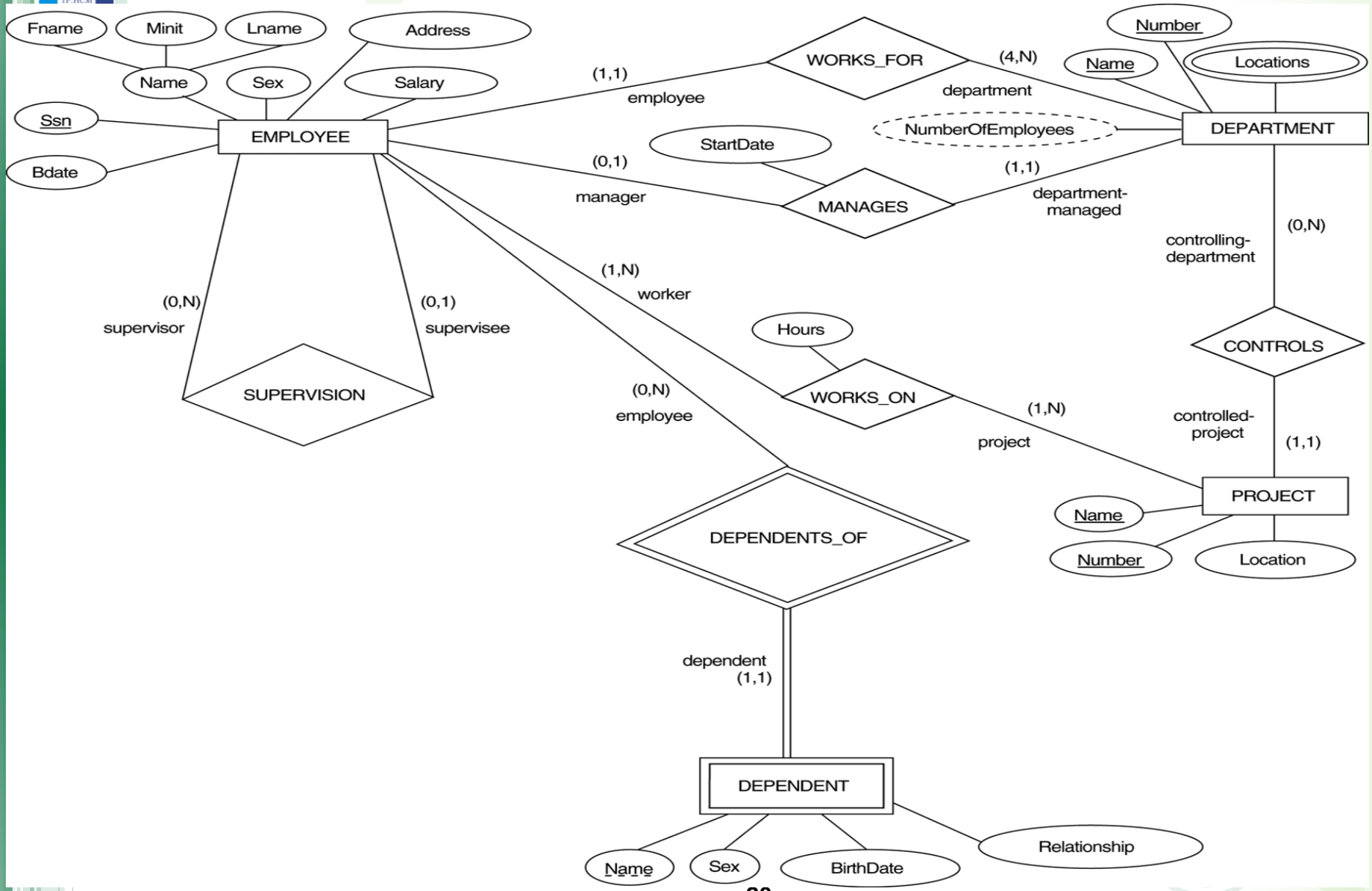


ER Diagram

(min, max) notation for relationship structural constraints

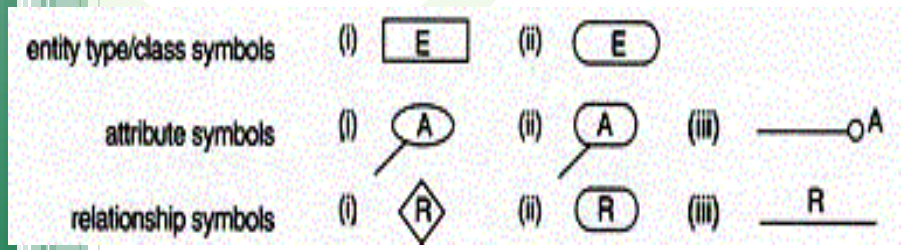


ER diagrams for the COMPANY schema, with structural constraints specified using (min, max) notation

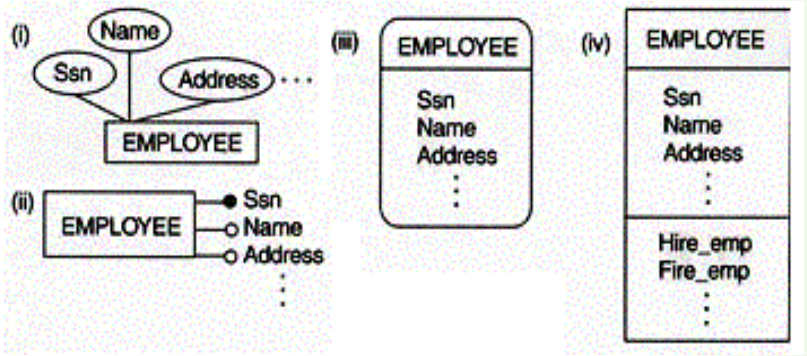


Alternative Diagrammatic Notations

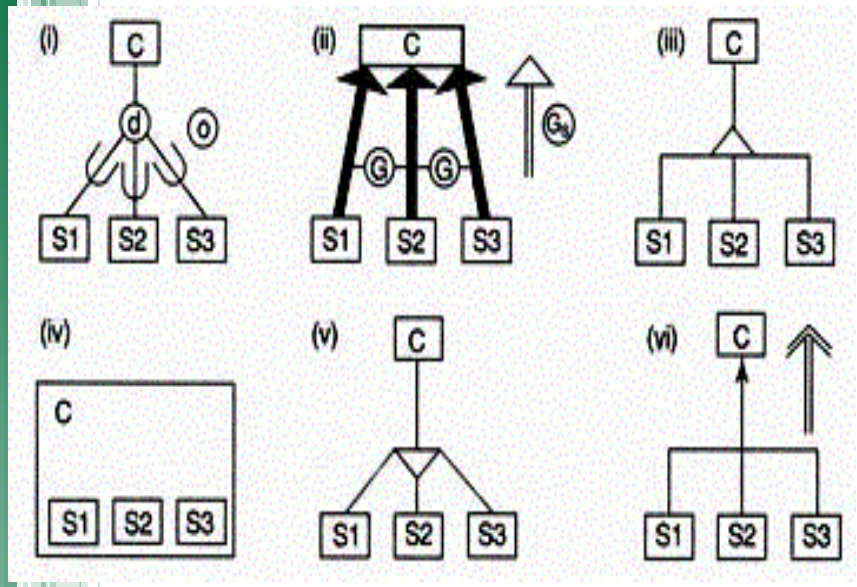
Symbols for entity type / class, attribute and relationship



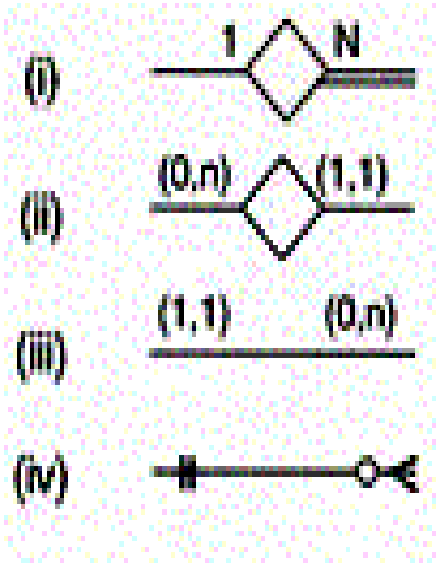
Displaying attributes



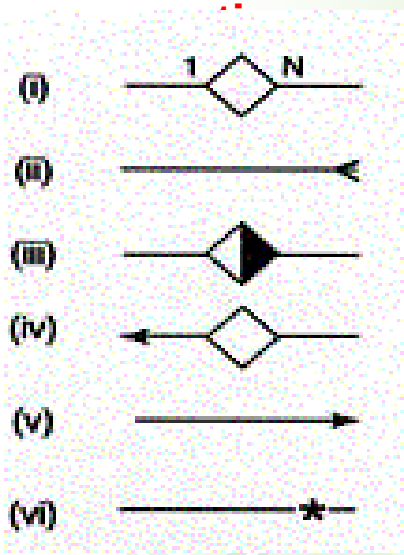
Notations for displaying specialization / generalization



Various (min, max) notations



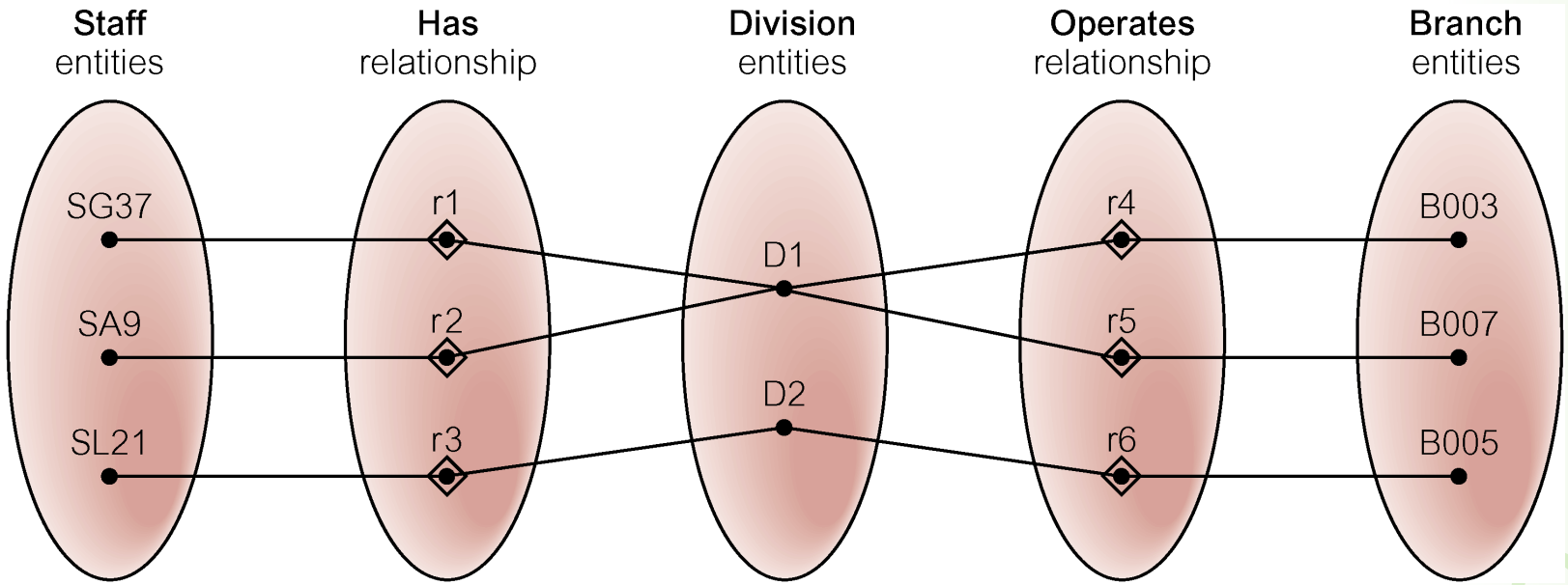
Displaying cardinality



Problems with ER Models

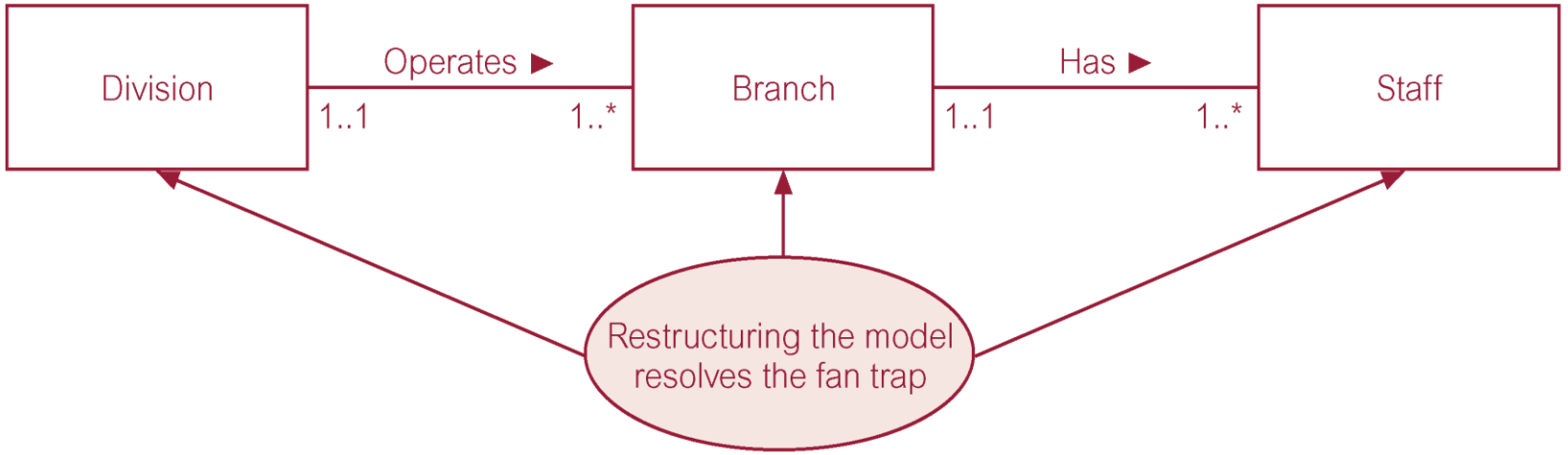
- Fan Trap
- Chasm Trap

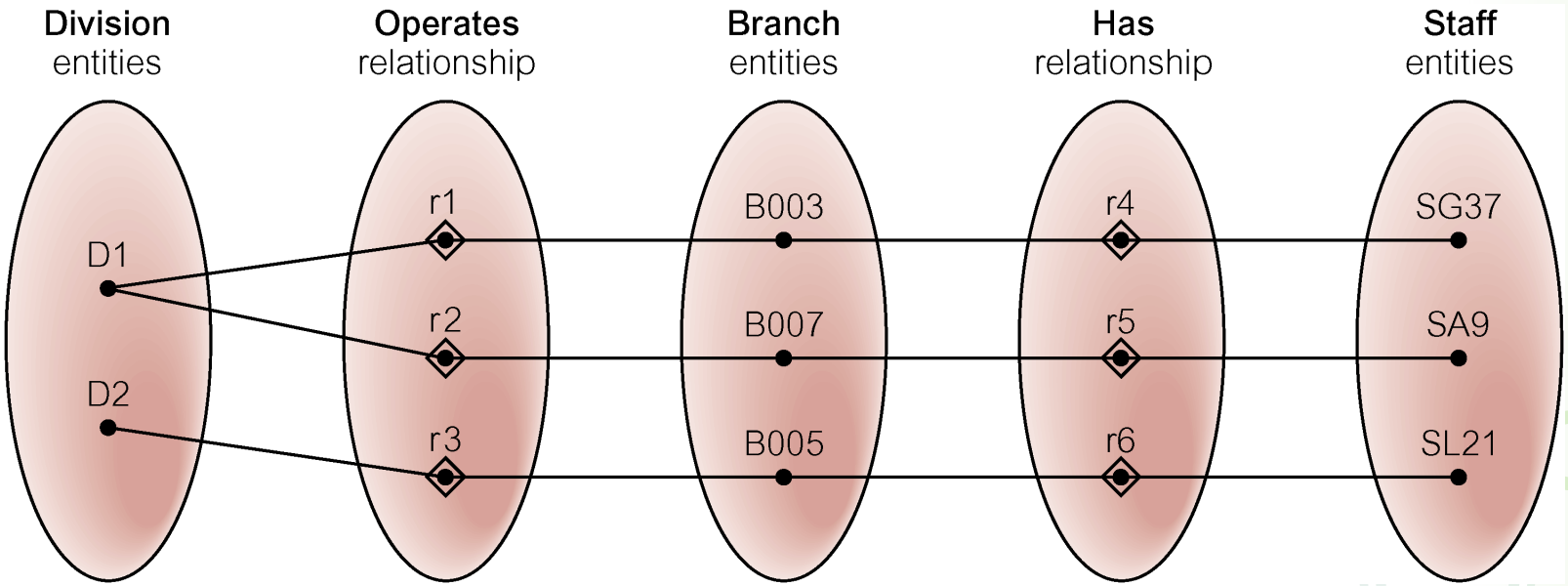
An Example of a Fan Trap



At which branch office does staff number SG37 work?

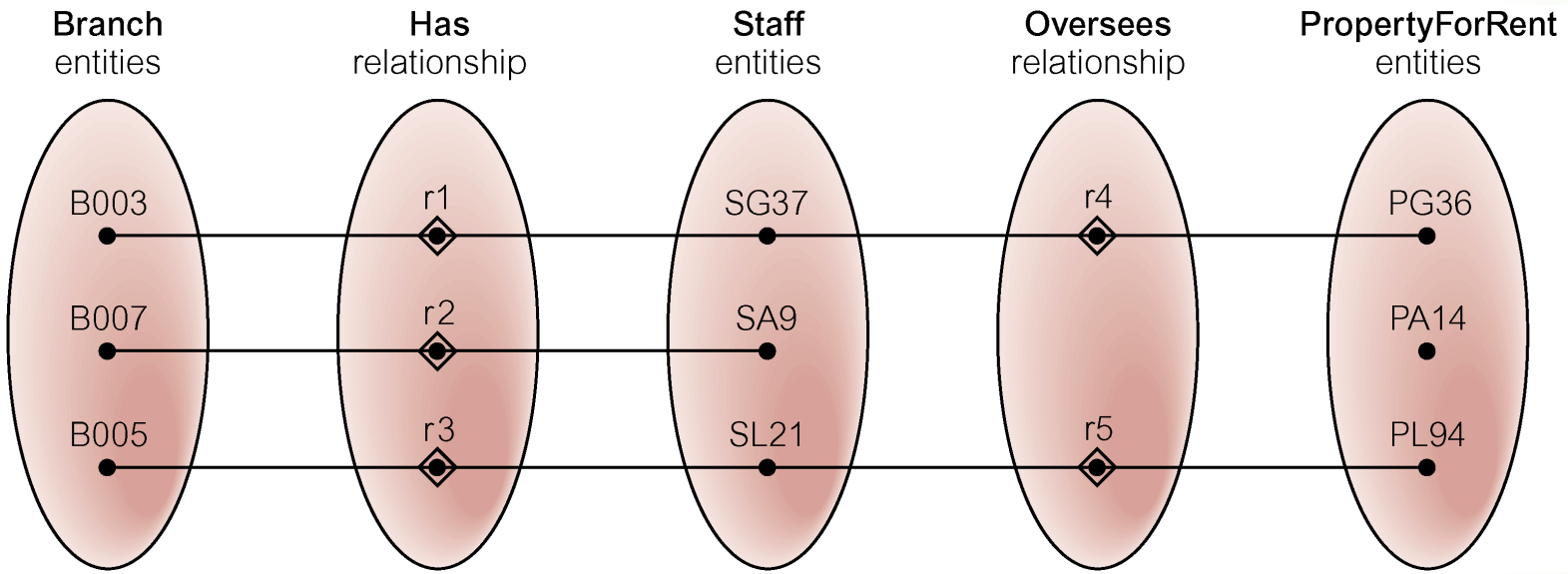
Restructuring ER model to remove Fan Trap





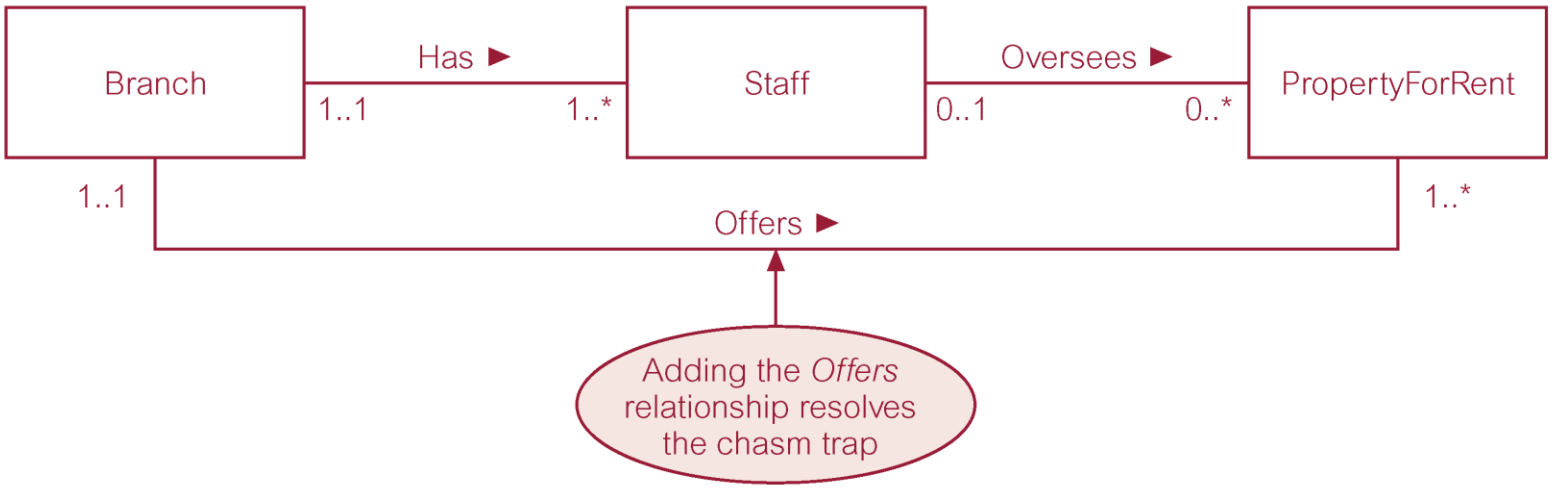
SG37 works at branch B003

An Example of a Chasm Trap

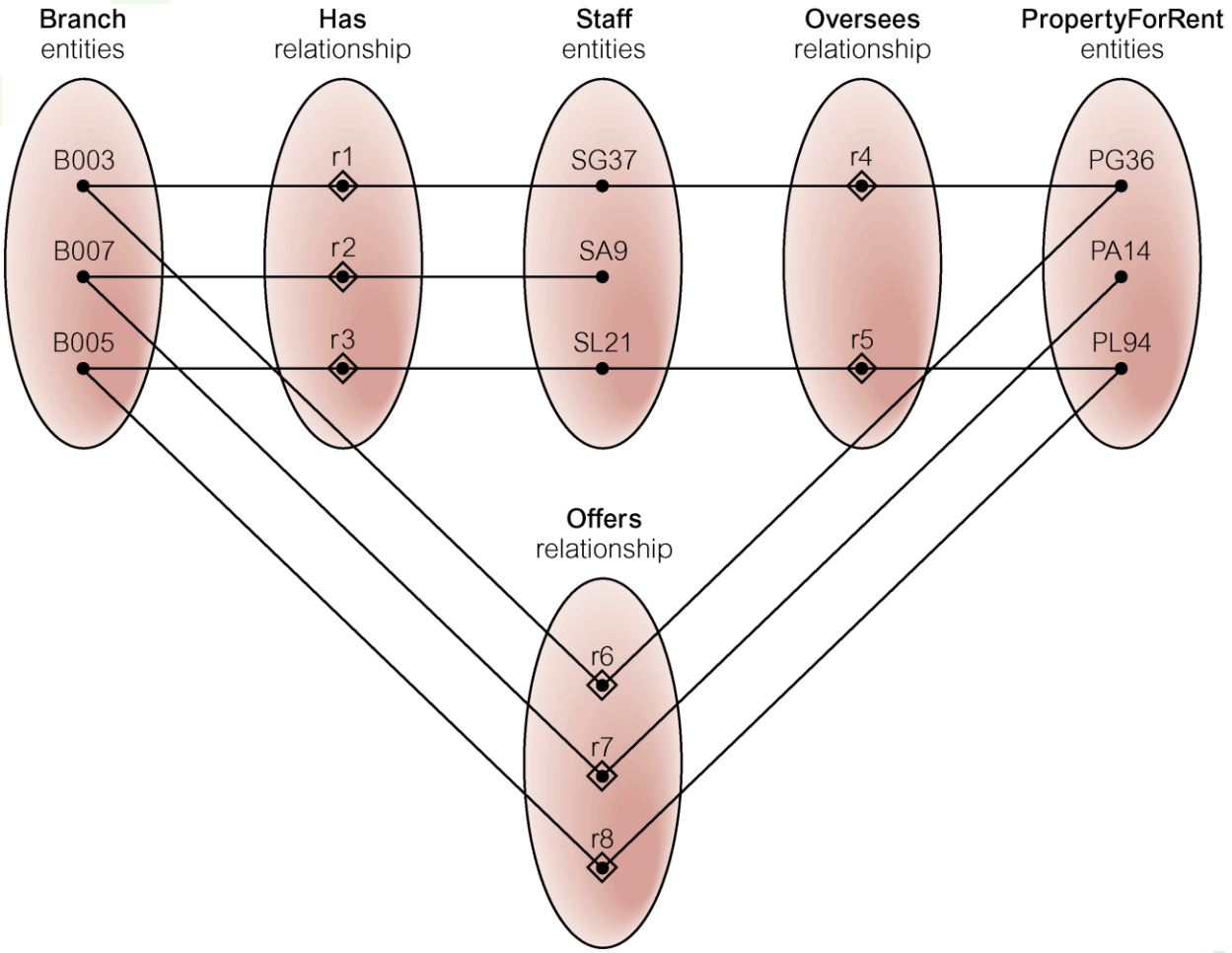


At which branch office is property PA14 available?

ER Model restructured to remove Chasm Trap



ER Model restructured to remove Chasm Trap



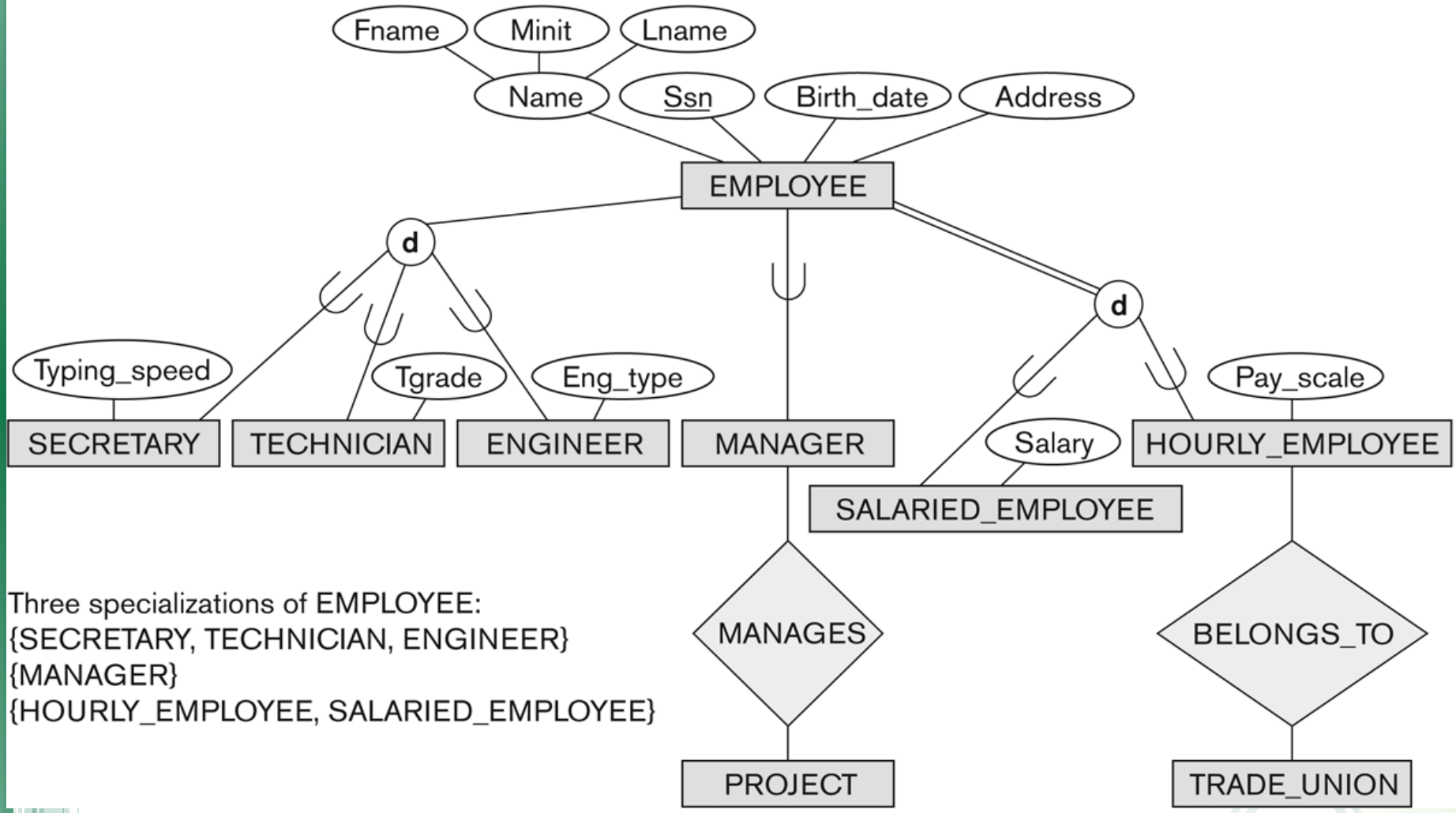
Course Outline

Lecture	Content
Database System Concepts & Architecture	-Introduction to Data Models, Database Systems -Three-Level Architecture & Data Independence -Modern Database Applications
Entity-Relationship (ER) Model	-ER Model -Introduction to Enhanced ER (EER) Model
Relational Model	-Relational Data Model -ER- & EER-to-Relational Mapping -Relational Algebra
SQL	-Data Definition Language (DDL) -Data Manipulation Language (DML) -Introduction to Triggers & Stored Procedures
Database Design Theory & Methodology	-Functional Dependencies & Normalization -Relational Database Design: Algorithms
Data Storage, Indexing, Query Processing & Physical Design	-Hashing & Indexing Structures (B-tree & R-tree families) -Physical Database Design
Database Security	-Discretionary & Mandatory Access Control (DAC & MAC) -Flow Control, Inference Problem -Security Issues in Modern Data Management Systems
Emerging Technologies & Applications	-Introduction to XML, Data Mining & Data Warehousing, GIS, M-Commerce & LBS -Emerging Database Technologies & Applications

Outline

- Limitations of Basic Concepts of the ER Model
- Enhanced-ER (EER) Model Concepts
- Subclasses and Superclasses
- Specialization and Generalization
- Specialization / Generalization Hierarchies, Lattices and Shared Subclasses
- Categories
- Formal Definitions of EER Model
- Database Design Modeling Tools

Subclasses and Superclasses



Three specializations of EMPLOYEE:
{SECRETARY, TECHNICIAN, ENGINEER}
{MANAGER}
{HOURLY_EMPLOYEE, SALARIED_EMPLOYEE}

Subclasses and Superclasses

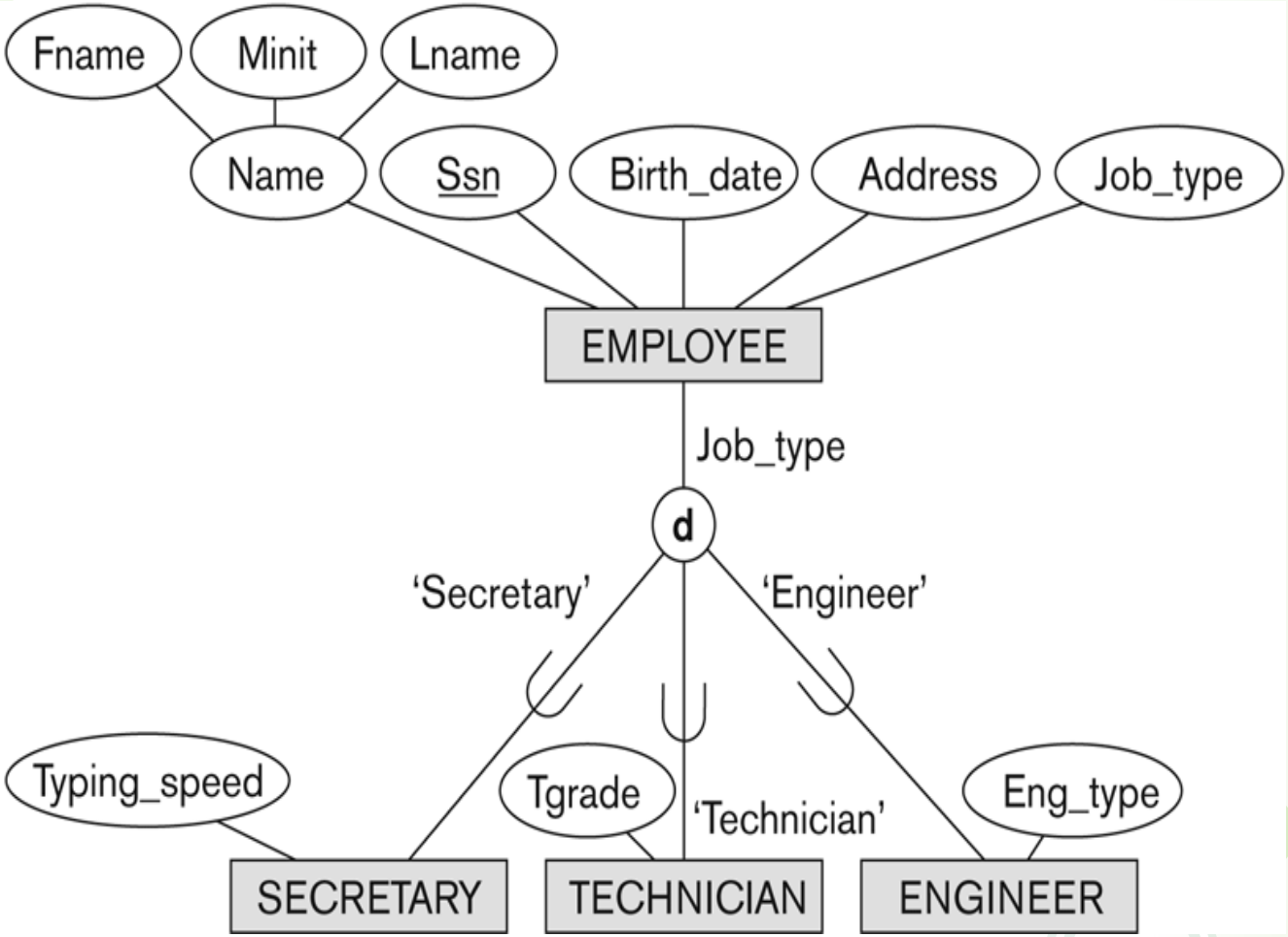
- These are also called IS-A (IS-AN) relationships
- Superclass/subclass relationship is one-to-one (1:1)
- Inheritance in Superclass/Subclass Relationships: attributes and relationships
- Generalization and Specialization

Constraints on Specialization and Generalization

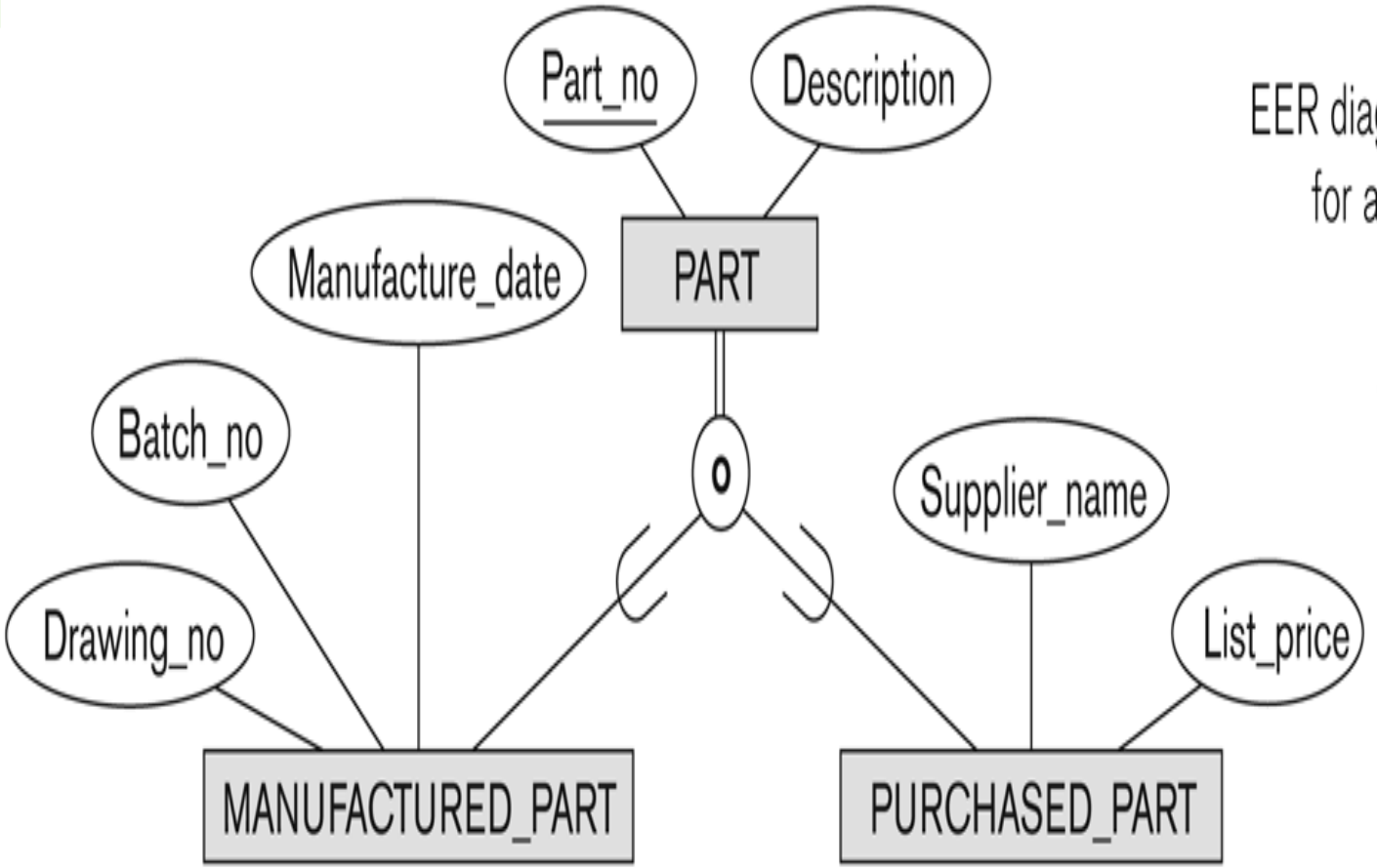
- Two basic conditions: disjointness and completeness constraints
- **Disjointness Constraint**:
  - Disjoint and overlap
- **Completeness Constraint**:
  - Total and partial

Example of Disjoint Partial Specialization

EER diagram notation for an attribute-defined specialization on Job_type.



Example of Overlapping Total Specialization

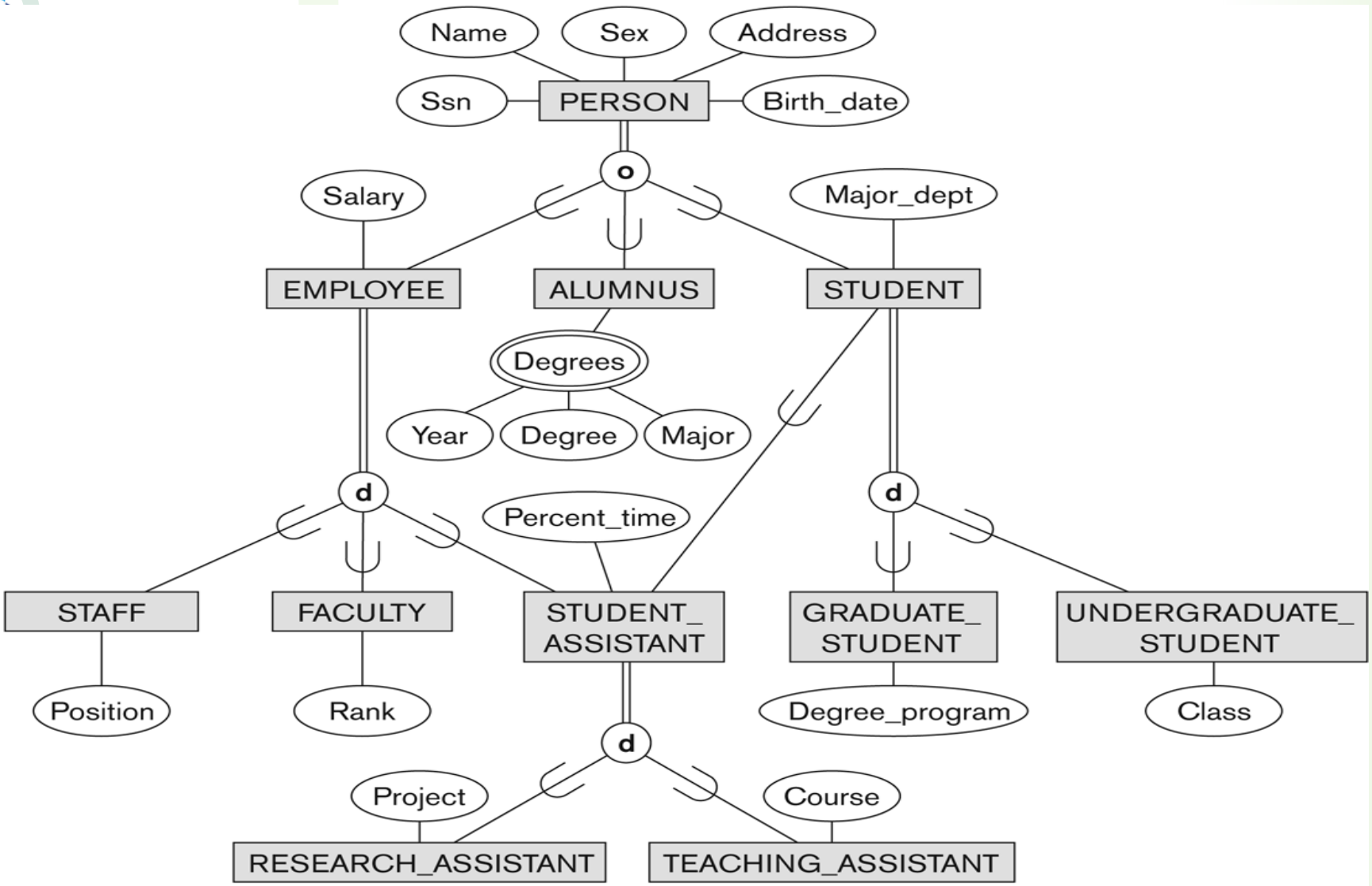


EER diagram notation for an overlapping (nondisjoint) specialization.

Specialization / Generalization Hierarchies, Lattices and Shared Subclasses

- A subclass may itself have further subclasses specified on it, forming a hierarchy or a lattice
- **Hierarchy** has a constraint that every subclass has only one superclass (called *single inheritance*)
- In a **lattice**, a subclass can be subclass of more than one superclass (called *multiple inheritance*)
- A subclass with more than one superclass is called a shared subclass

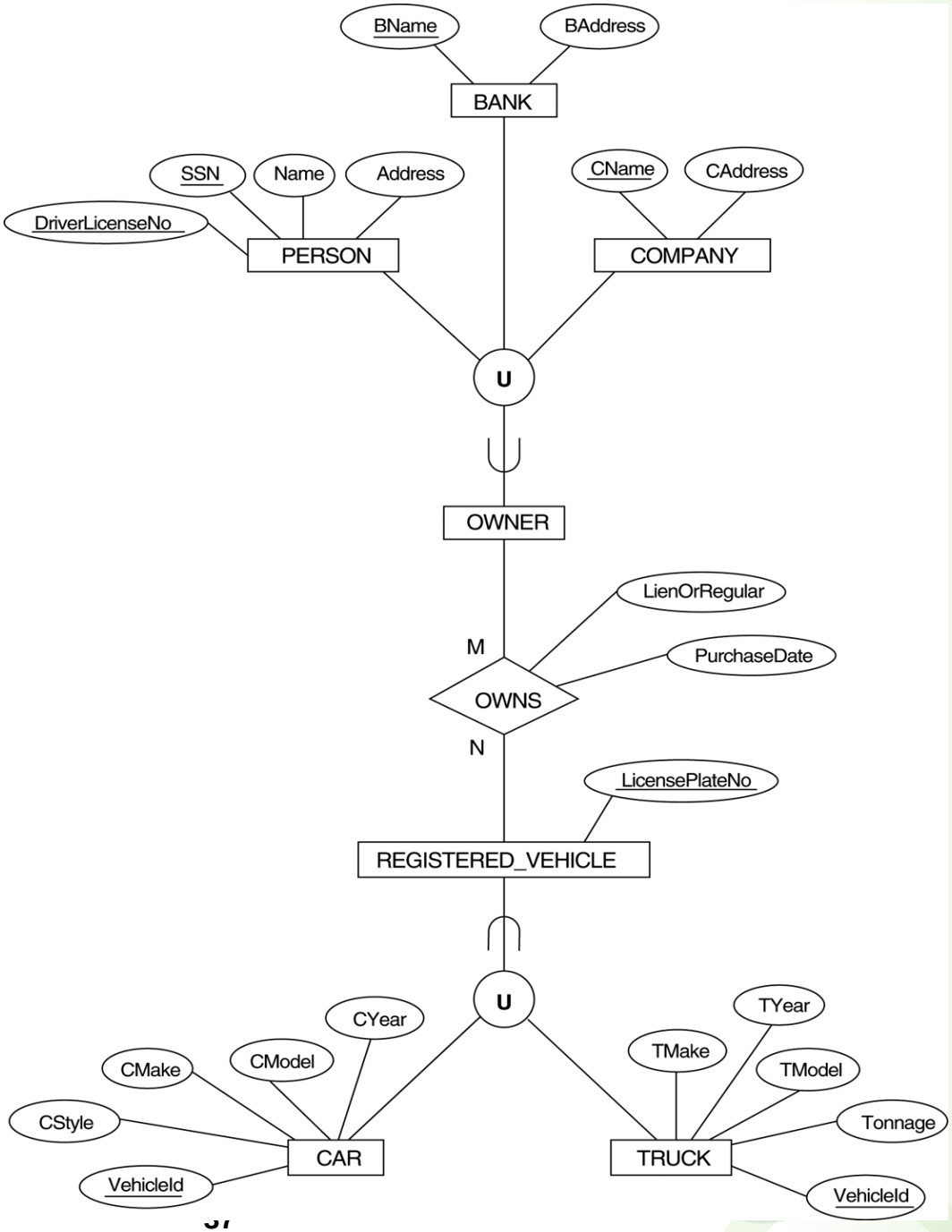
Specialization / Generalization Lattice Example (UNIVERSITY)



A specialization lattice with multiple inheritance for a UNIVERSITY database.

Categories

Two categories (union types):
OWNER and REGISTERED_VEHICLE



Database Design Modeling Tools

COMPANY	TOOL	FUNCTIONALITY
Embarcadero Technologies	ER Studio	Database Modeling in ER and IDEF1X
	DB Artisan	Database administration and space and security management
Oracle	Developer 2000 and Designer 2000	Database modeling, application development
Popkin Software	System Architect 2001	Data modeling, object modeling, process modeling, structured analysis/design
Platinum Technology (Computer Associates)	Platinum Enterprice Modeling Suite: Erwin, BPWin, Paradigm Plus	Data, process, and business component modeling
Persistence Inc.	Pwertier	Mapping from O-O to relational model
Rational (IBM)	Rational Rose	Modeling in UML and application generation in C++ and JAVA
Rogue Ware	RW Metro	Mapping from O-O to relational model
Resolution Ltd.	Xcase	Conceptual modeling up to code maintenance
Sybase	Enterprise Application Suite	Data modeling, business logic modeling
Visio (Microsoft)	Visio Enterprise	Data modeling, design and reengineering Visual Basic and Visual C++

Course Outline

Lecture	Content
Database System Concepts & Architecture	-Introduction to Data Models, Database Systems -Three-Level Architecture & Data Independence -Modern Database Applications
Entity-Relationship (ER) Model	-ER Model -Introduction to Enhanced ER (EER) Model
Relational Model	-Relational Data Model -ER- & EER-to-Relational Mapping -Relational Algebra
SQL	-Data Definition Language (DDL) -Data Manipulation Language (DML) -Introduction to Triggers & Stored Procedures
Database Design Theory & Methodology	-Functional Dependencies & Normalization -Relational Database Design: Algorithms
Data Storage, Indexing, Query Processing & Physical Design	-Hashing & Indexing Structures (B-tree & R-tree families) -Physical Database Design
Database Security	-Discretionary & Mandatory Access Control (DAC & MAC) -Flow Control, Inference Problem -Security Issues in Modern Data Management Systems
Emerging Technologies & Applications	-Introduction to XML, Data Mining & Data Warehousing, GIS, M-Commerce & LBS -Emerging Database Technologies & Applications

Relational Data Model and ER-/EER-to-Relational Mapping

Basic Concepts

- **Relational data model**
- **Relation schema**: R(A1, A2,…, An)
- The **degree** of a relation
- **Domain** D
- **Tuple**
- **Cardinality**
- **Database schema** S = {R1, R2,…, Rm}

Relational data model
Database schema
Relation schema
Relation
Tuple
Attribute

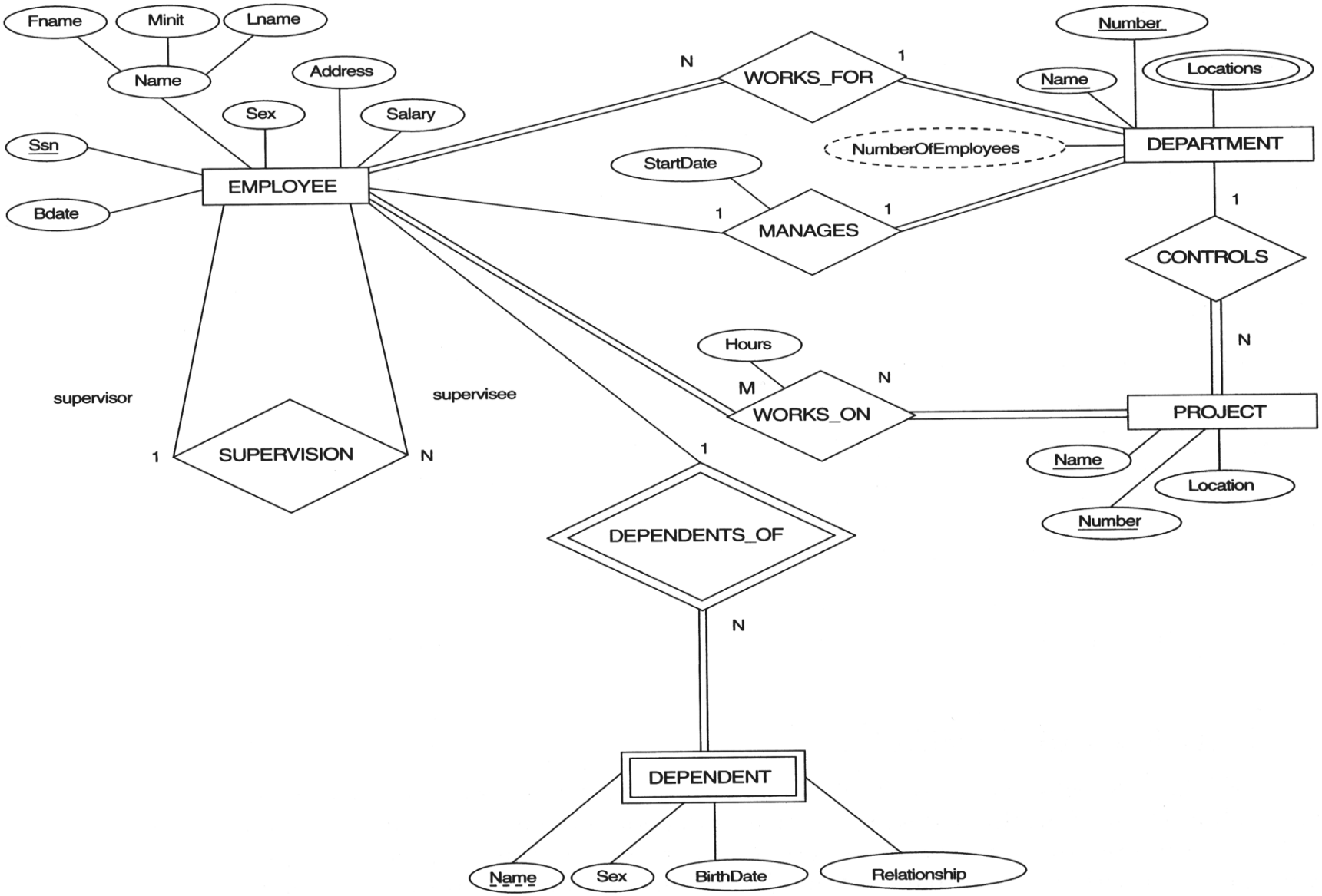
Relational Integrity Constraints

- Constraints are *conditions* that must hold on *all* valid relation instances. There are three main types of constraints:
  1. **Key** constraints
  2. **Entity integrity** constraints
  3. **Referential integrity** constraints
- Other types:
  - Semantic Integrity Constraints
  - State/static constraints (so far)
  - Transition/dynamic constraints

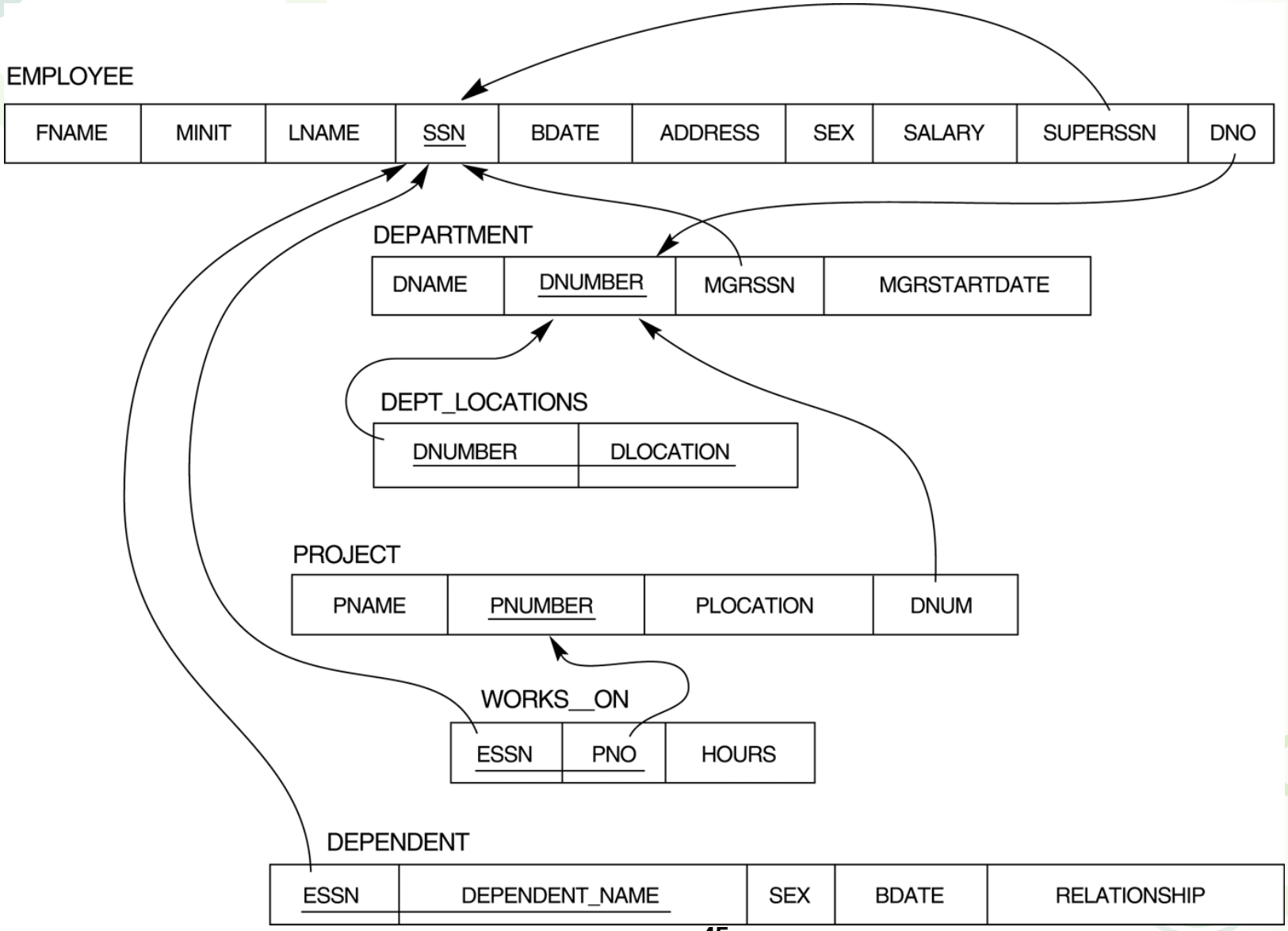
ER- & EER-to-Relational Mapping

- ER-
  - Step 1: Mapping of Regular Entity Types
  - Step 2: Mapping of Weak Entity Types
  - Step 3: Mapping of Binary 1:1 Relationship Types
  - Step 4: Mapping of Binary 1:N Relationship Types
  - Step 5: Mapping of Binary M:N Relationship Types
  - Step 6: Mapping of Multivalued attributes
  - Step 7: Mapping of N-ary Relationship Types
- *EER-*
  - *Step 8: Options for Mapping Specialization or Generalization.*
  - *Step 9: Mapping of Union Types (Categories)*

The ERD for the COMPANY database



Result of mapping the COMPANY ER schema into a relational schema



ER-to-Relational Mapping

Correspondence between ER and Relational Models

ER Model	Relational Model
Entity type	"Entity" relation
1:1 or 1:N relationship type	Foreign key (or "relationship" relation)
M:N relationship type	"Relationship" relation and two foreign keys
n-ary relationship type	"Relationship" relation and n foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Value set	Domain
Key attribute	Primary (or secondary) key

Q&A

Review

Những phát biểu nào sau đây là SAI

A. Khóa chính của lược đồ quan hệ sinh ra từ mối liên kết 2 ngôi 1-n sẽ là khóa chính của lược đồ quan hệ tương ứng bên phía 1 của mối liên kết trên

B. Khóa chính của lược đồ quan hệ sinh ra từ thực thể yếu là khóa riêng phần của kiểu thực thể yếu đó

C. Lược đồ quan hệ được sinh ra từ thuộc tính đa trị sẽ không có khóa chính

D. Mọi kiểu mối liên kết trong ERD/EERD đều được biến đổi thành một lược đồ quan hệ tương ứng

Review

Trong sơ đồ ERD/EERD, những phát biểu nào sau đây SAI

A. Bậc của mối liên kết là số lượng các kiểu thực thể (entity type) tham gia vào mối liên kết
B. Một kiểu thực thể có thể có nhiều thuộc tính khóa
C. Kiểu thực thể yếu phải luôn luôn có khóa riêng phần (partial key)
D. Kiểu thực thể yếu phải luôn luôn có mối liên kết với một kiểu thực thể mạnh nào đó

Review

Gọi R là kiểu mối liên kết 1-n giữa hai kiểu thực thể A và B (n bên phía B theo tập ký hiệu của Chen). Những phát biểu nào sau đây là SAI

A. Một thực thể a của A có thể liên kết với n thực thể b của B
B. Một thực thể a của A có thể tham gia vào n thể hiện (r_i) của R
C. Một thực thể b của B có thể tham gia vào n thể hiện (r_i) của R
D. Mọi thực thể b của B đều phải tham gia vào ít nhất 1 thể hiện (r_i) của R

Review

Chuyển đổi từ mô hình ERD/EERD sang lược đồ cơ sở dữ liệu quan hệ nằm ở giai đoạn nào trong quá trình thiết kế một cơ sở dữ liệu

A. Thiết kế ý niệm (conceptual design)
B. Thiết kế luận lý (logical design)
C. Thiết kế vật lý (physical design)
D. Phân tích chức năng của hệ thống (functional analysis)

Review

Khi thiết kế ERD/EERD cho một thư viện sách, những nhóm đối tượng nào sau đây có thể được xem là các kiểu thực thể.

A. Sách, tác giả, nhà xuất bản, nhà cung cấp, độc giả, phiếu mượn

B. Sách, thư viện, tác giả, độc giả, nhà cung cấp, phiếu mượn

C. Sách, tác giả, nhà xuất bản, nhà cung cấp, độc giả, quốc tịch

D. Thư viện, sách, độc giả, tác giả, phiếu mượn, phiếu trả

Review

Những phát biểu nào sau đây là ĐÚNG trong mô hình dữ liệu quan hệ

A. *Tuple, row, record* là thuật ngữ tương đương trong mô hình CDSL quan hệ

B. Một quan hệ có thể có nhiều khóa chính (primary key)

C. Một quan hệ phải có ít nhất 1 khóa dự tuyển (candidate key)

D. Một quan hệ có thể có nhiều khóa ngoại (foreign key)

Review

Gọi R là mối liên kết 1-n của A và B (n bên phía B và option ở cả 2 phía). Những chọn lựa sau đây là có thể chấp nhận được

A. Biến đổi R thành một quan hệ

B. Không biến đổi R thành một quan hệ mà đặt khóa chính của B vào quan hệ tương ứng của A để làm khóa ngọai

C. Không biến đổi R thành một quan hệ mà đặt khóa chính của A vào quan hệ tương ứng của B để làm khóa ngọai

D. Tất cả các câu trên đều sai

Những đối tượng nào sau đây trong ERD/EERD luôn được chuyển thành một quan hệ

A. Thực thể mạnh
B. Thực thể yếu
C. Mối liên kết 2 ngôi 1-1
D. Mối liên kết 2 ngôi 1-n
E. Mối liên kết 2 ngôi n-n
F. Mối liên kết 3 ngôi
G. Thuộc tính đa trị
H. Thuộc tính kết hợp
I. Thuộc tính dẫn xuất

Review

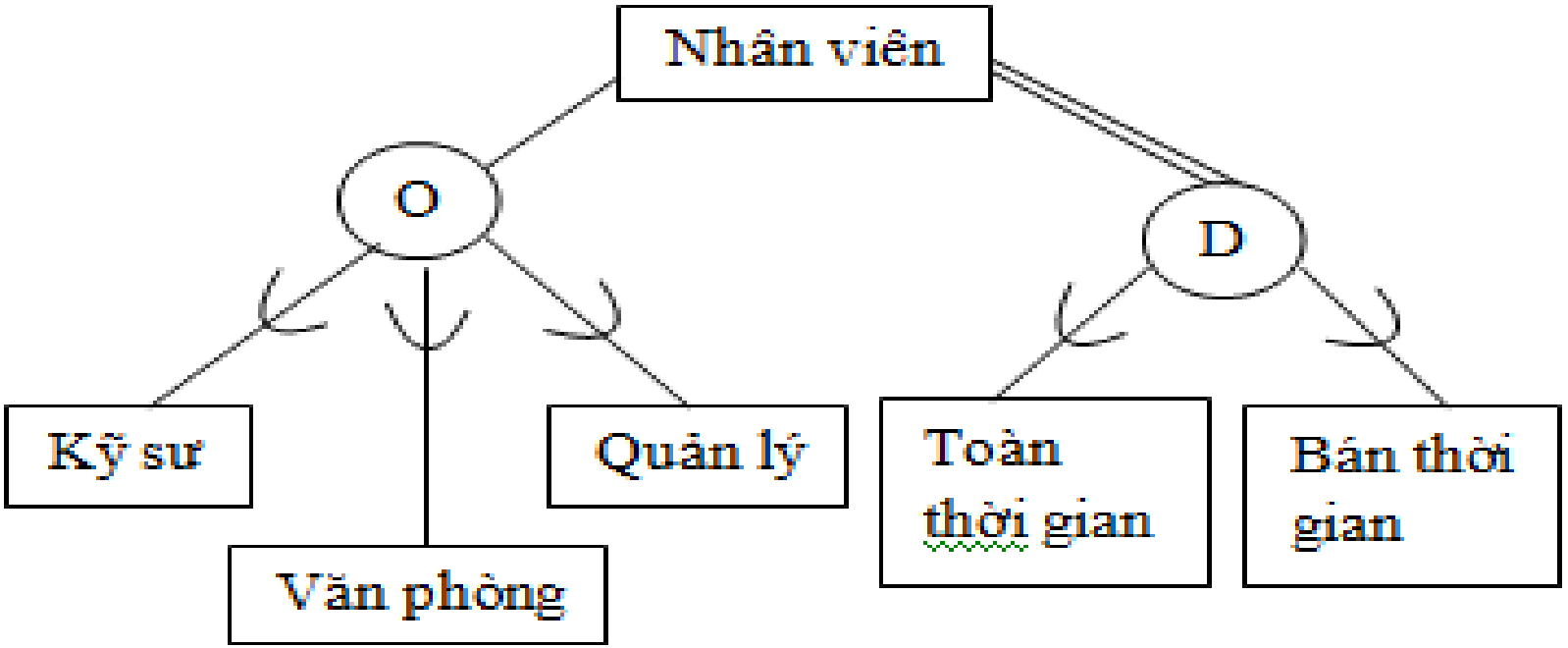
Một ERD gồm 2 kiểu thực thể A và B liên kết n-n với nhau và mối liên kết này có thuộc tính đa trị. Theo cách biến đổi thông thường thì lược đồ CSDL tương ứng của ERD này có bao nhiêu lược đồ quan hệ:

A. 2

B. 1

C. 4

D. 5

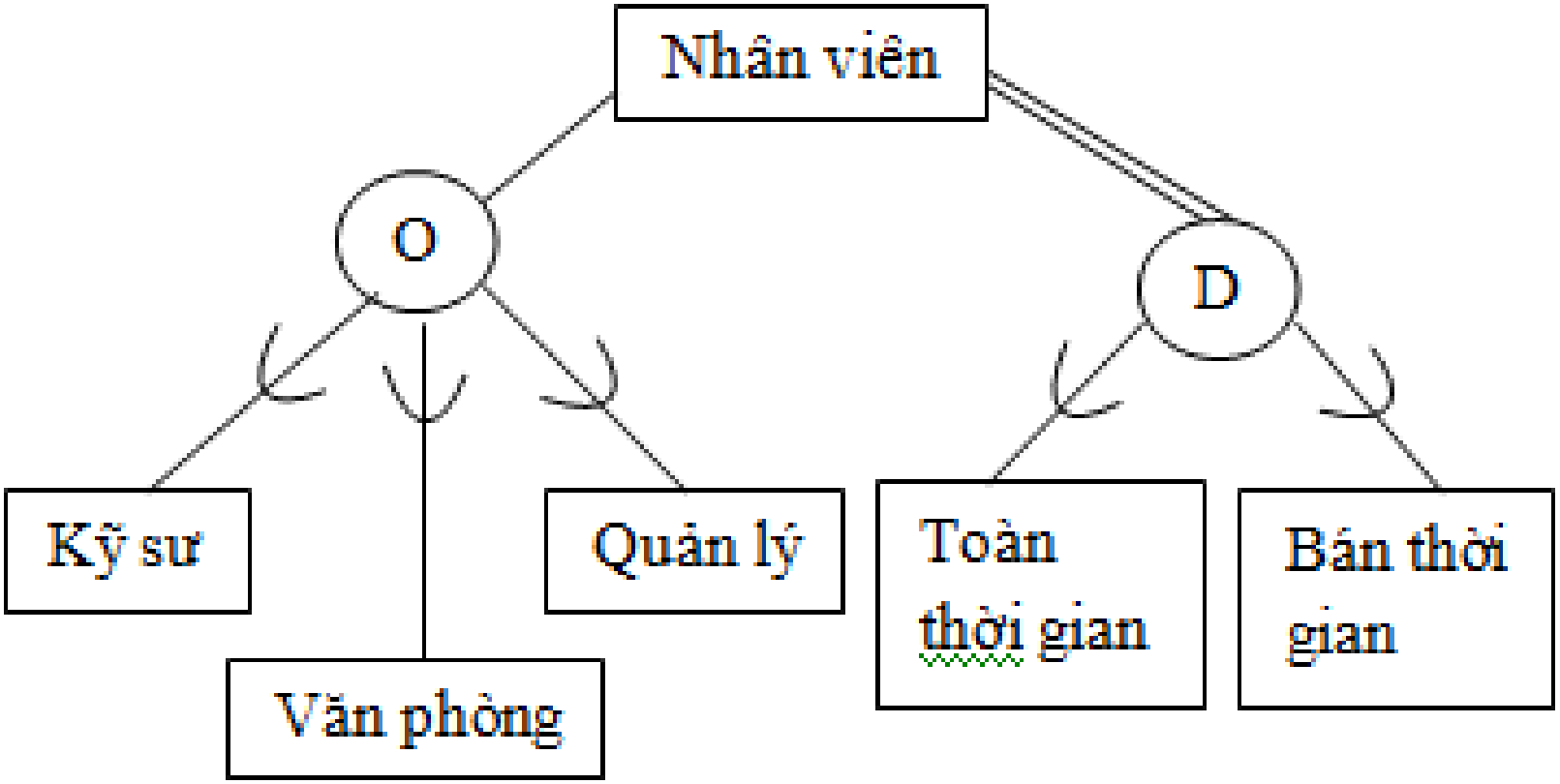


A. Có thể có nhân viên vừa làm toàn thời gian vừa làm bán thời gian

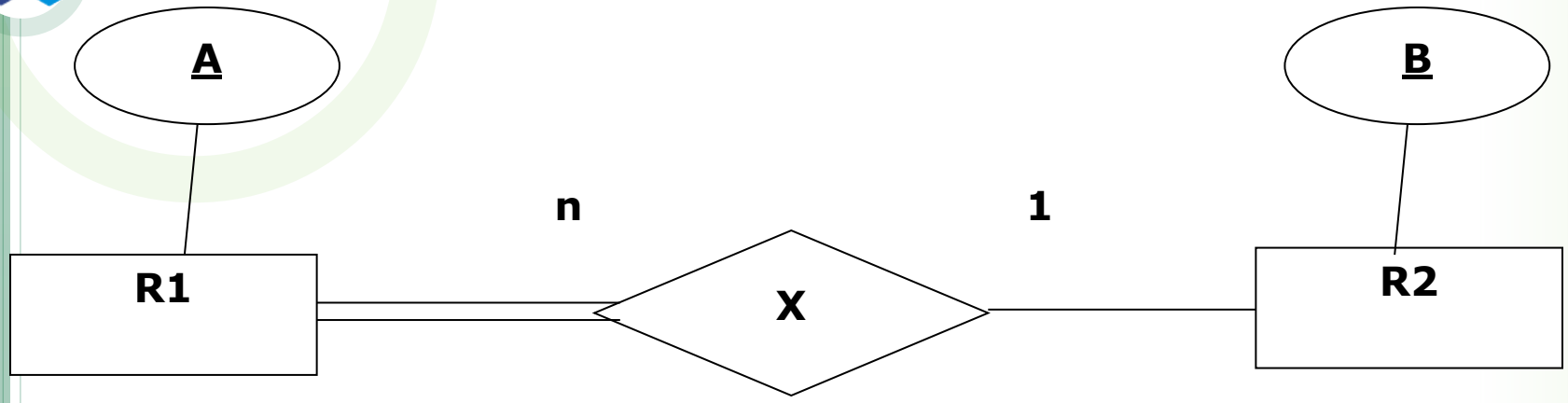
B. Một nhân viên nào đó có thể vừa làm kỹ sư, vừa làm văn phòng, vừa làm quản lý

C. Nếu một nhân viên không làm văn phòng và không làm quản lý thì phải làm kỹ sư

D. Tổng số nhân viên toàn thời gian và bán thời gian có thể nhỏ hơn số nhân viên trong công ty


Nhân viên
O
D
Kỹ sư
Văn phòng
Quản lý
Toàn thời gian
Bán thời gian

Review



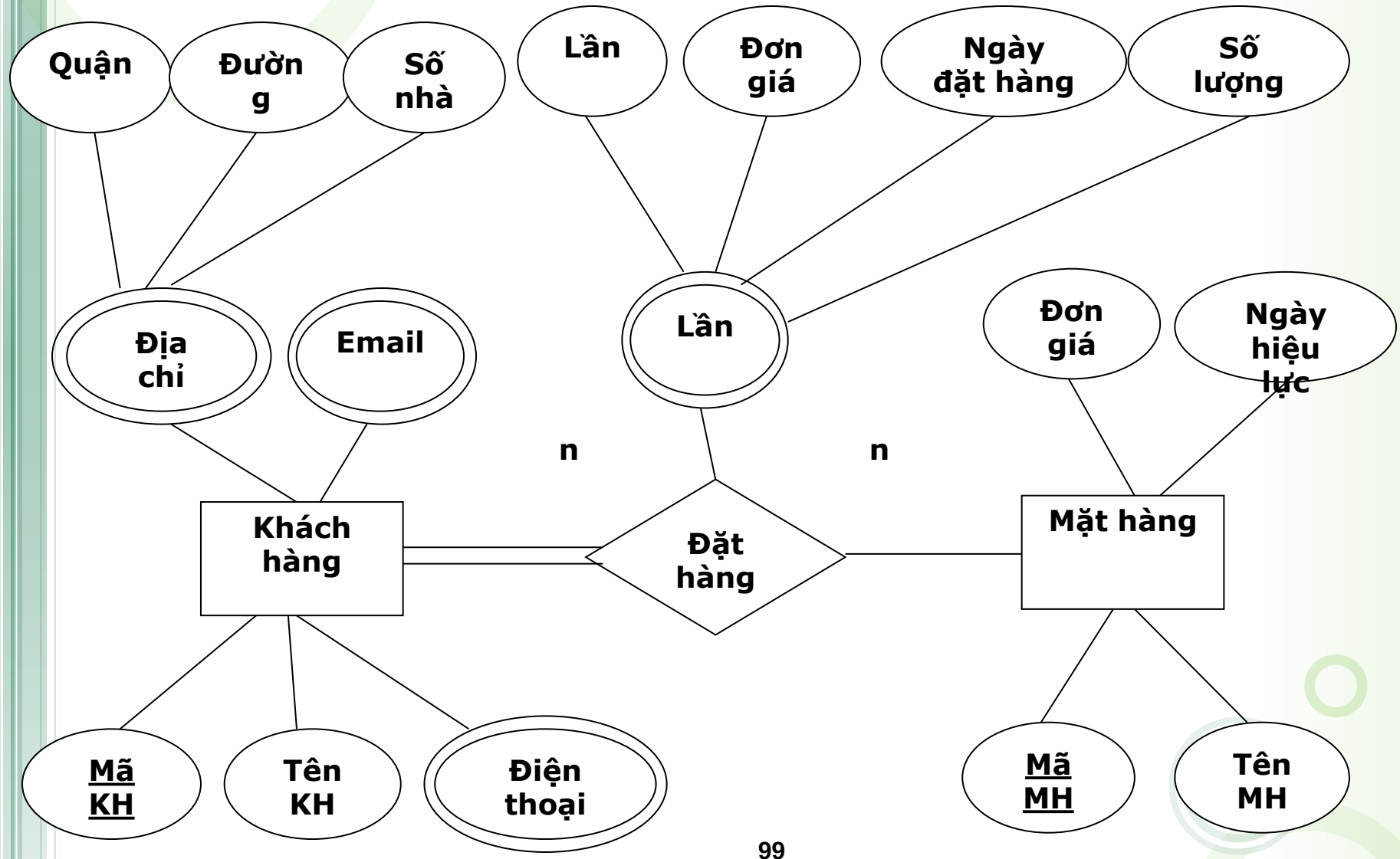
A. R1 (A, B)
   R2 (B)

B. R1 (A)
   R2 (B, A)

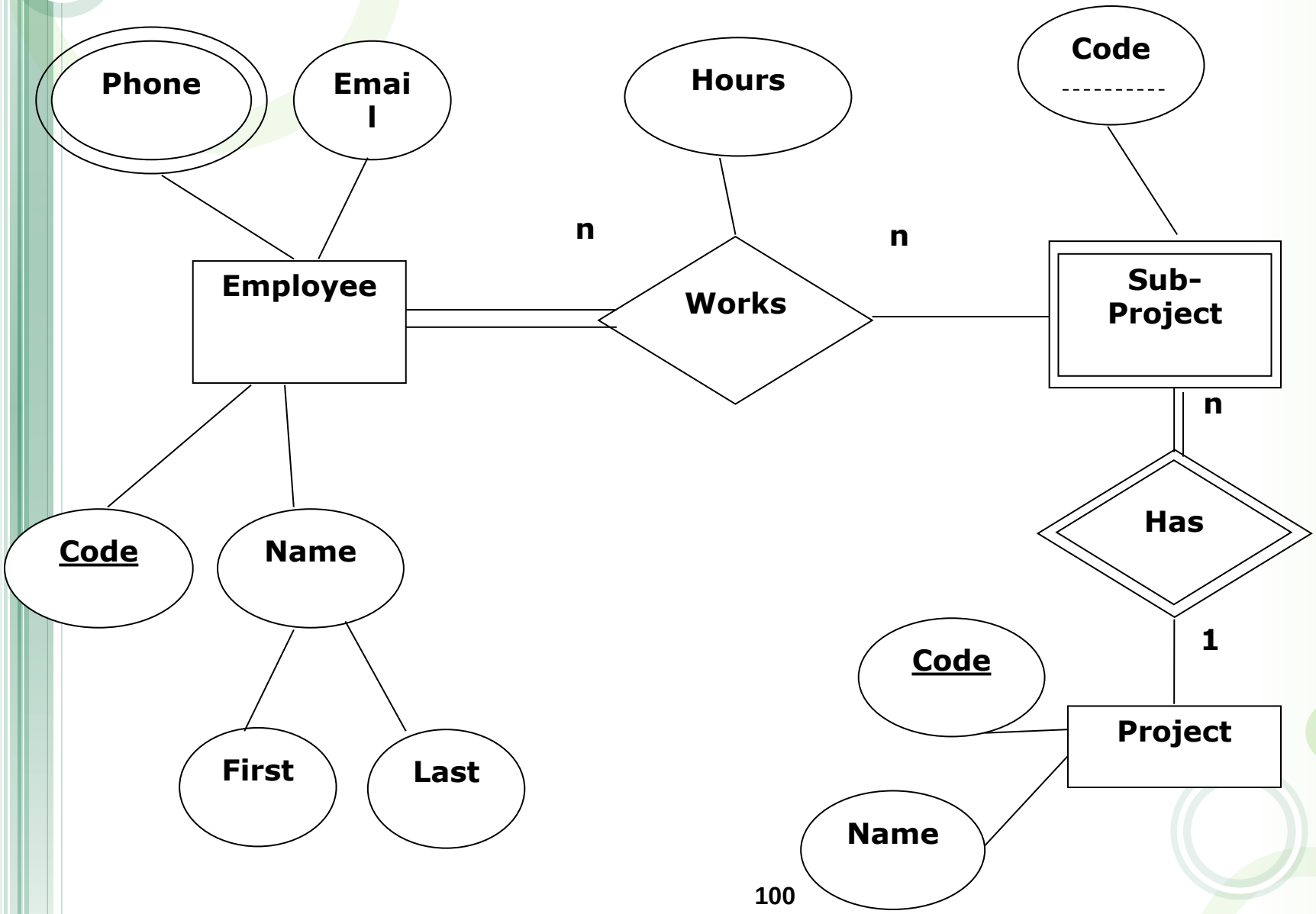
C. R1 (A)
   R2 (B)
   X (A, B)

D. R1 (A)
   R2 (B)
   X (A, B)

Review


Quận
Đường
Số nhà
Địa chỉ
Email
Khách hàng
Mã KH
Tên KH
Điện thoại
Lần
Đơn giá
Ngày đặt hàng
Số lượng
Lần
n
Đặt hàng
n
Đơn giá
Ngày hiệu lực
Mặt hàng
Mã MH
Tên MH

Review


Phone
Emai
l
Hours
Code
Employee
n
Works
n
Sub-
Project
n
Has
1
Code
Name
First
Last
Code
Project
Name

Review

Choose the **WRONG** answers

A. Primary key of a relation generated from the 1-n binary relationship is the primary key of a relation that is 1 side of that relationship

B. Primary key of a relation generated from weak entity is the partial key of that weak entity

C. A relation generated from multivalued attributes has no primary key

D. Every relationship in ERD/EERD must be mapped to a relation

Review

In ERD/EERD, choose the **WRONG** answers

A. The degree of a relationship is the number of entity types joining that relationship

B. An entity type may have many key attributes

C. A weak entity type must always have a partial key

D. A weak entity type must always have a relationship with a strong entity type

Review

Let R be 1-n relationship type between the two entity types A and B (n is on the side of B with regards to Chen's notation). Choose the **WRONG** answers:

A. An entity of A can participate in a relationship with many entities of B

B. An entity of A can participate in n instances (r_i) of R

C. An entity of B can participate in n instances (r_i) of R

D. All entities of B must participate in at least one instance (r_i) of R

Review

Mapping ERD/EERD to relational database schema belongs to which phase of database design process

A. Conceptual design
B. Logical design
C. Physical design
D. Functional analysis

Review

When designing ERD/EERD for a library, which group of objects are considered as entity types?

A. Book, author, publisher, provider, reader, library ticket
B. Book, library, author, reader, provider, library ticket
C. Book, author, publisher, provider, author, nationality
D. Library, book, reader, author, library ticket

Review

Choose the **CORRECT** answers in relational data model

A. *Tuple*, *row*, *record* are equivalent terms
B. One relation may have many primary keys
C. One relation must have at least one candidate key
D. One relation may have many foreign keys

Review

Let R be 1-n relationship of A and B (n is on B' side and optional participation constraints are on both sides). Which design is acceptable?

A. Mapping R into a relation
B. Do not map R into a relation. Instead, putting the primary key of B into the relation of A as a foreign key
C. Do not map R into a relation. Instead, putting the primary key of A into the relation of B as a foreign key
D. All are incorrect

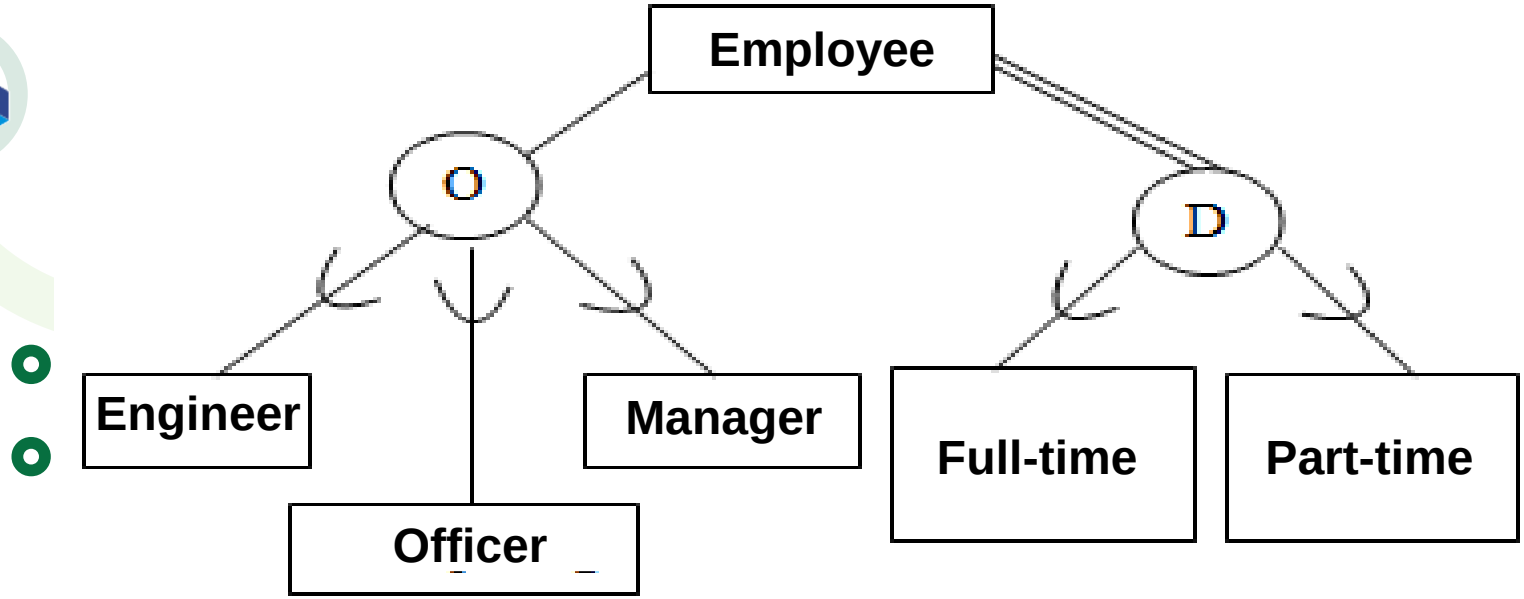
Which objects in ERD/EERD are always mapped into relations

A. Strong entity type
B. Weak entity type
C. 1-1 relationship
D. 1-n relationship
E. n-n relationship
F. Ternary relationship
G. Multivalued attributes
H. Composite attributes
I. Derrived attributes

Review

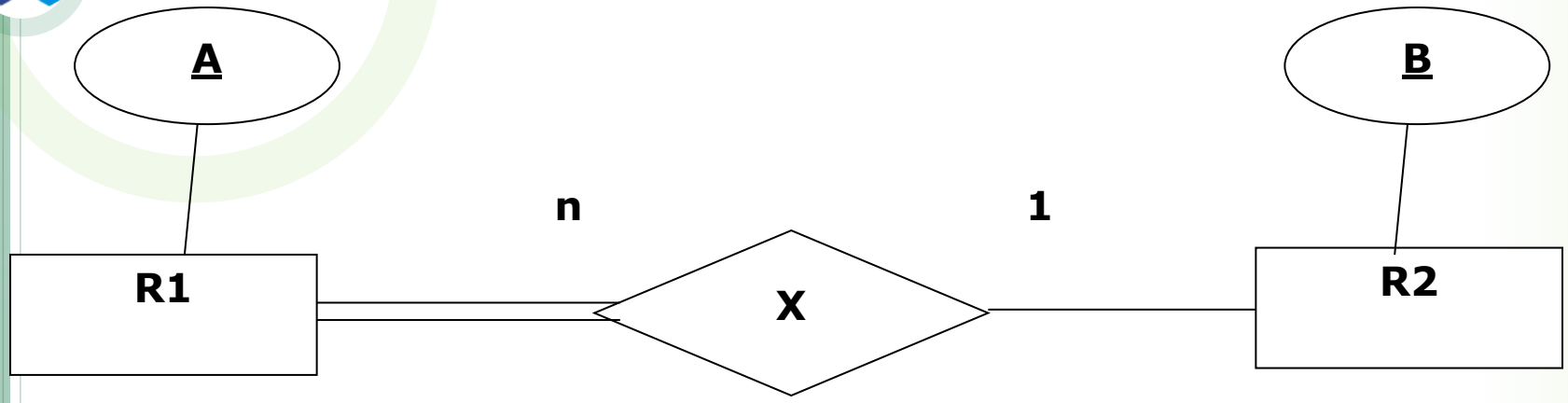
An ERD has 2 entity types A and B with n-n relationship that has a multivalued attribute. According to the mapping rules, how many relations are there?

A. 2
B. 1
C. 4
D. 5



A. There are some employees who are full-time and part-time
B. There are some employees who are engineers, officers, and managers at the same time
C. If an employee who is neither officer nor manager must be engineer
D. The number of full-time and part-time employees may be smaller than the total number of employees

Review



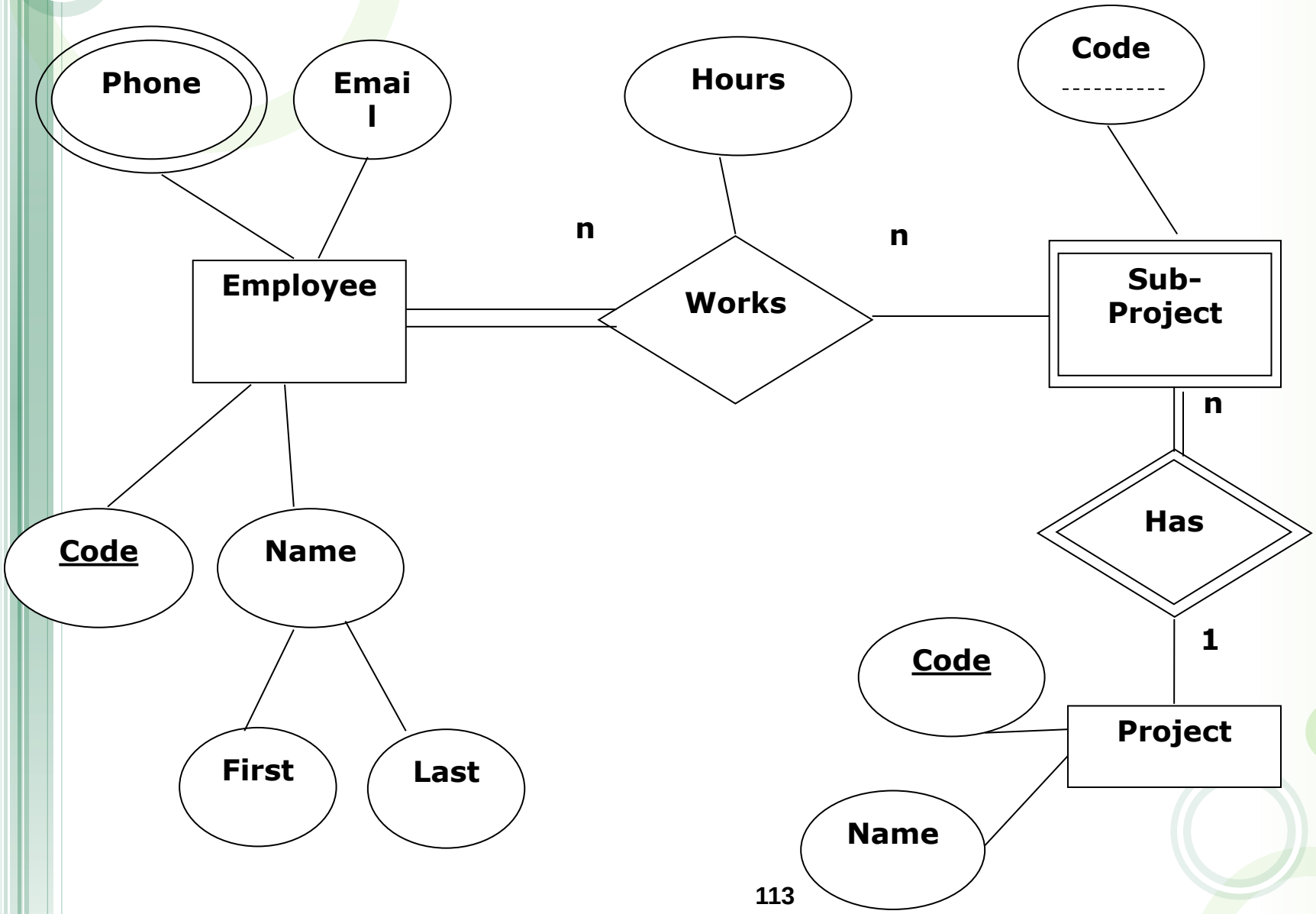
A. R1 (A, B)
   R2 (B)

B. R1 (A)
   R2 (B, A)

C. R1 (A)
   R2 (B)
   X (A, B)

D. R1 (A)
   R2 (B)
   X (A, B)

Review


Phone
Emai
l
Hours
Code
Employee
n
Works
n
Sub-
Project
n
Has
1
Code
Name
First
Last
Code
Project
Name

Course Outline

Lecture	Content
Database System Concepts & Architecture	-Introduction to Data Models, Database Systems -Three-Level Architecture & Data Independence -Modern Database Applications
Entity-Relationship (ER) Model	-ER Model -Introduction to Enhanced ER (EER) Model
Relational Model	-Relational Data Model -ER- & EER-to-Relational Mapping -Relational Algebra
SQL	-Data Definition Language (DDL) -Data Manipulation Language (DML) -Introduction to Triggers & Stored Procedures
Database Design Theory & Methodology	-Functional Dependencies & Normalization -Relational Database Design: Algorithms
Data Storage, Indexing, Query Processing & Physical Design	-Hashing & Indexing Structures (B-tree & R-tree families) -Physical Database Design
Database Security	-Discretionary & Mandatory Access Control (DAC & MAC) -Flow Control, Inference Problem -Security Issues in Modern Data Management Systems
Emerging Technologies & Applications	-Introduction to XML, Data Mining & Data Warehousing, GIS, M-Commerce & LBS -Emerging Database Technologies & Applications

Outline

- Relational Algebra
  - Unary Relational Operations
  - Relational Algebra Operations from Set Theory
  - Binary Relational Operations
  - Additional Relational Operations
- Brief Introduction to Relational Calculus

Relational Algebra Overview

- Relational algebra is the basic set of operations for the relational model
- The result of an operation is a *new relation*
- A sequence of relational algebra operations forms a **relational algebra expression**

Relational Algebra Overview

- Relational Algebra consists of several groups of operations
  - Unary Relational Operations
    - SELECT (symbol: σ (sigma))
    - PROJECT (symbol: π (pi))
    - RENAME (symbol: ρ (rho))
  - Relational Algebra Operations from Set Theory
    - UNION (∪), INTERSECTION (∩), DIFFERENCE (or MINUS, –)
    - CARTESIAN PRODUCT (**x**)
  - Binary Relational Operations
    - JOIN (several variations of JOIN exist)
    - DIVISION
  - Additional Relational Operations
    - OUTER JOINS, OUTER UNION
    - AGGREGATE FUNCTIONS (SUM, COUNT, AVG, MIN, MAX)

Unary Relational Operations: SELECT

- Select operation is denoted by

$$\sigma_{<\text{selection condition}>}(R)$$

- Examples:
  - Select the EMPLOYEE tuples whose department number is 4:

$$\sigma_{DNO = 4}(EMPLOYEE)$$

  - Select the employee tuples whose salary is greater than $30,000:

$$\sigma_{SALARY > 30,000}(EMPLOYEE)$$

Unary Relational Operations: PROJECT

- PROJECT Operation is denoted by π (pi)

$$\pi_{<\text{attribute list}>}(R)$$

- Example: To list each employee's first and last name and salary, the following is used:

$$\pi_{\text{LNAME, FNAME,SALARY}}(\text{EMPLOYEE})$$

Unary Relational Operations: RENAME

- RENAME operation ρ (rho) can be expressed by any of the following forms:
  - $\rho_{S\ (B1,\ B2,\ \ldots,\ Bn\ )}(R)$ changes both:
    - the relation name to S, *and*
    - the column (attribute) names to B1, B1, …..Bn
  - $\rho_S(R)$ changes:
    - the *relation name* only to S
  - $\rho_{(B1,\ B2,\ \ldots,\ Bn\ )}(R)$ changes:
    - the *column (attribute) names* only to B1, B1, …..Bn

Relational Algebra Operations from Set Theory: UNION

- UNION Operation
  - Binary operation, denoted by ∪
  - The result of R ∪ S, is a relation that includes all tuples that are either in R or in S or in both R and S
  - Duplicate tuples are eliminated
  - The two operand relations R and S must be “type compatible” (or UNION compatible)

Relational Algebra Operations from Set Theory: INTERSECTION

- INTERSECTION is denoted by ∩
- The result of the operation R ∩ S, is a relation that includes all tuples that are in both R and S
- The two operand relations R and S must be "type compatible"

Relational Algebra Operations from Set Theory: SET DIFFERENCE (cont.)

- SET DIFFERENCE (also called MINUS or EXCEPT) is denoted by –
- The result of R – S, is a relation that includes all tuples that are in R but not in S
- The two operand relations R and S must be “type compatible”

Relational Algebra Operations from Set Theory: CARTESIAN PRODUCT

- CARTESIAN (or CROSS) PRODUCT Operation
  - Denoted by R(A1, A2, . . ., An) x S(B1, B2, . . ., Bm)
  - Result is a relation Q with degree n + m attributes:
    - Q(A1, A2, . . ., An, B1, B2, . . ., Bm), in that order.
  - Hence, if R has n_R tuples (denoted as |R| = n_R), and S has n_S tuples, then R x S will have n_R * n_S tuples
  - The two operands do NOT have to be "type compatible"

Binary Relational Operations: JOIN

- JOIN Operation (denoted by ⋈)

$$R \bowtie_{\text{<join condition>}} S$$

- The general case of JOIN operation is called a Theta-join: $R \bowtie_{theta} S$
- A join, where the only comparison operator used is =, is called an EQUIJOIN
- NATURAL JOIN Operation - denoted by *
  - NATURAL JOIN was created to get rid of the second (superfluous) attribute in an EQUIJOIN condition
- The OUTER JOIN Operation

Binary Relational Operations: DIVISION

- **DIVISION Operation**
  - The division operation is applied to two relations R(Z)÷S(X), where Z = X ∪ Y (Y is the set of attributes of R that are not attributes of S
  - The result of DIVISION is a relation T(Y) that includes a tuple t if tuples t_R appear in R with t_R [Y] = t, and with

    t_R [X] = t_s *for every tuple* t_s in S, i.e., for a tuple t to appear in the result T of the DIVISION, the values in t must appear in R in combination with *every* tuple in S

Additional Relational Operations

- Aggregate Functions and Grouping
  - Common functions applied to collections of numeric values include SUM, AVERAGE, MAXIMUM, and MINIMUM. The COUNT function is used for counting tuples or values

Course Outline

Lecture	Content
Database System Concepts & Architecture	-Introduction to Data Models, Database Systems -Three-Level Architecture & Data Independence -Modern Database Applications
Entity-Relationship (ER) Model	-ER Model -Introduction to Enhanced ER (EER) Model
Relational Model	-Relational Data Model -ER- & EER-to-Relational Mapping -Relational Algebra
SQL	-Data Definition Language (DDL) -Data Manipulation Language (DML) -Introduction to Triggers & Stored Procedures
Database Design Theory & Methodology	-Functional Dependencies & Normalization -Relational Database Design: Algorithms
Data Storage, Indexing, Query Processing & Physical Design	-Hashing & Indexing Structures (B-tree & R-tree families) -Physical Database Design
Database Security	-Discretionary & Mandatory Access Control (DAC & MAC) -Flow Control, Inference Problem -Security Issues in Modern Data Management Systems
Emerging Technologies & Applications	-Introduction to XML, Data Mining & Data Warehousing, GIS, M-Commerce & LBS -Emerging Database Technologies & Applications

Outline

- DDL: Data Definition Language
  - Create
  - Alter
  - Drop
- DML: Data Manipulation Language
  - Select
  - Insert
  - Update
  - Delete
- DCL: Data Control Language
  - Commit
  - Rollback
  - Grant,
  - Revoke

DDL: Create, Alter, Drop

CREATE SCHEMA

- CREATE SCHEMA SchemaName AUTHORIZATION AuthorizationIdentifier;

CREATE SCHEMA Company AUTHORIZATION JSmith;

DDL: Create, Alter, Drop

CREATE TABLE

```
CREATE TABLE TableName
{(colName dataType [NOT NULL] [UNIQUE]
[DEFAULT defaultOption]
[CHECK searchCondition] [,...]}
[PRIMARY KEY (listOfColumns),]
{[UNIQUE (listOfColumns),] […,]}
{[FOREIGN KEY (listOfFKColumns)
  REFERENCES ParentTableName [(listOfCKColumns)],
  [ON UPDATE referentialAction]
  [ON DELETE referentialAction ]] [,…]}
{[CHECK (searchCondition)] [,…] })
```

DDL: Create, Alter, Drop

CREATE TABLE

- Default values
- Primary key and referential integrity constraints
  - **referential triggered action** clause of FK constraint:

    ON DELETE <action>

    ON UPDATE <action>

    <action>: SET NULL, CASCADE, SET DEFAULT
- Giving names to constraints
- Specifying constraints on tuples using CHECK

DDL: Create, Alter, Drop

DROP Command

- Used to drop *named* schema elements: tables, domains, constraints, and the schema itself

  DROP SCHEMA Company CASCADE; (RESTRICT)

  DROP TABLE Dependent CASCADE; (RESTRICT)

  - Similarly, we can drop constraints & domains

DDL: Create, Alter, Drop

ALTER Command

- Base tables: adding or dropping a column or constraints, changing a column definition. Example:

```
ALTER TABLE Company.Employee ADD Job VARCHAR(15)
    NOT NULL;
```

DML: Select, Insert, Update, Delete

SELECT

```
SELECT [DISTINCT | ALL]
   {* | [columnExpression [AS newName]] [,...] }
FROM TableName [alias] [, ...]
[WHERE condition]
[GROUP BY columnList]   [HAVING condition]
[ORDER BY columnList]
```

DML: Select, Insert, Update, Delete

Insert

- In its simplest form, it is used to add one or more tuples to a relation

```
INSERT INTO TABLE_NAME
VALUES (LIST_OF_VALUES)
```

DML: Select, Insert, Update, Delete

Delete

- Removes tuples from a relation
- Includes a WHERE-clause to select the tuples to be deleted
- Tuples are deleted from only *one table* at a time (unless CASCADE is specified on a referential integrity constraint)
- A missing WHERE-clause specifies that *all tuples* in the relation are to be deleted; the table then becomes an empty table
- The number of tuples deleted depends on the number of tuples in the relation that satisfy the WHERE-clause

DML: Select, Insert, Update, Delete

Update

- Used to modify attribute values of one or more selected tuples
- A WHERE-clause selects the tuples to be modified
- An additional SET-clause specifies the attributes to be modified and their new values
- Each command modifies tuples *in the same relation*
- Referential integrity should be enforced

Advanced DDL: Assertions & Triggers

- ASSERTIONs to express constraints that do not fit in the basic SQL categories
- Mechanism: `CREATE ASSERTION`
  - components include: a constraint name, followed by `CHECK`, followed by a condition
- TRIGGERs: to specify the type of action to be taken as certain events occur & as certain conditions are satisfied

VIEWs

- SQL command: **CREATE VIEW**
  - a view (table) name
  - a possible list of attribute names
  - a query to specify the view contents
- Specify a different WORKS_ON table (view)

```
CREATE VIEW     WORKS_ON_NEW AS
    SELECT      FNAME, LNAME, PNAME, HOURS
    FROM        EMPLOYEE, PROJECT, WORKS_ON
    WHERE       SSN=ESSN AND PNO=PNUMBER
```

DCL: Commit, Rollback, Grant, Revoke

- Chapter 17: Transaction Processing
- Chapter 23: DB security

Course Outline

Lecture	Content
Database System Concepts & Architecture	-Introduction to Data Models, Database Systems -Three-Level Architecture & Data Independence -Modern Database Applications
Entity-Relationship (ER) Model	-ER Model -Introduction to Enhanced ER (EER) Model
Relational Model	-Relational Data Model -ER- & EER-to-Relational Mapping -Relational Algebra
SQL	-Data Definition Language (DDL) -Data Manipulation Language (DML) -Introduction to Triggers & Stored Procedures
Database Design Theory & Methodology	-Functional Dependencies & Normalization -Relational Database Design: Algorithms
Data Storage, Indexing, Query Processing & Physical Design	-Hashing & Indexing Structures (B-tree & R-tree families) -Physical Database Design
Database Security	-Discretionary & Mandatory Access Control (DAC & MAC) -Flow Control, Inference Problem -Security Issues in Modern Data Management Systems
Emerging Technologies & Applications	-Introduction to XML, Data Mining & Data Warehousing, GIS, M-Commerce & LBS -Emerging Database Technologies & Applications

Outline

- Introduction
- Functional dependencies (FDs)
  - Definition of FD
  - Direct, indirect, partial dependencies
  - Inference Rules for FDs
  - Equivalence of Sets of FDs
  - Minimal Sets of FDs
- Normalization
  - 1NF and dependency problems
  - 2NF – solves partial dependency
  - 3NF – solves indirect dependency
  - BCNF – well-normalized relations

Introduction

- "Goodness" measures:
  - Redundant information in tuples
  - Update anomalies: modification, deletion, insertion
  - Reducing the NULL values in tuples
  - Disallowing the possibility of generating spurious tuples

Functional Dependencies (FDs)

- Functional dependencies (FDs) are used to specify *formal measures* of the "goodness" of relational designs
- FDs and keys are used to define **normal forms** for relations
- X -> Y holds if whenever two tuples have the same value for X, they *must have* the same value for Y
- Examples:
  - social security number determines employee name:

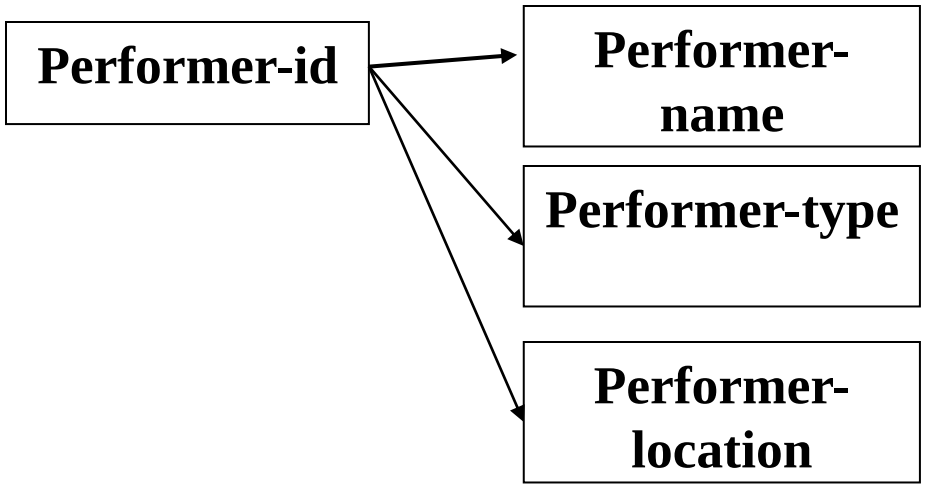
    SSN -> ENAME
  - project number determines project name and location:

    PNUMBER -> {PNAME, PLOCATION}
  - employee ssn and project number determines the hours per week that the employee works on the project:

    {SSN, PNUMBER} -> HOURS

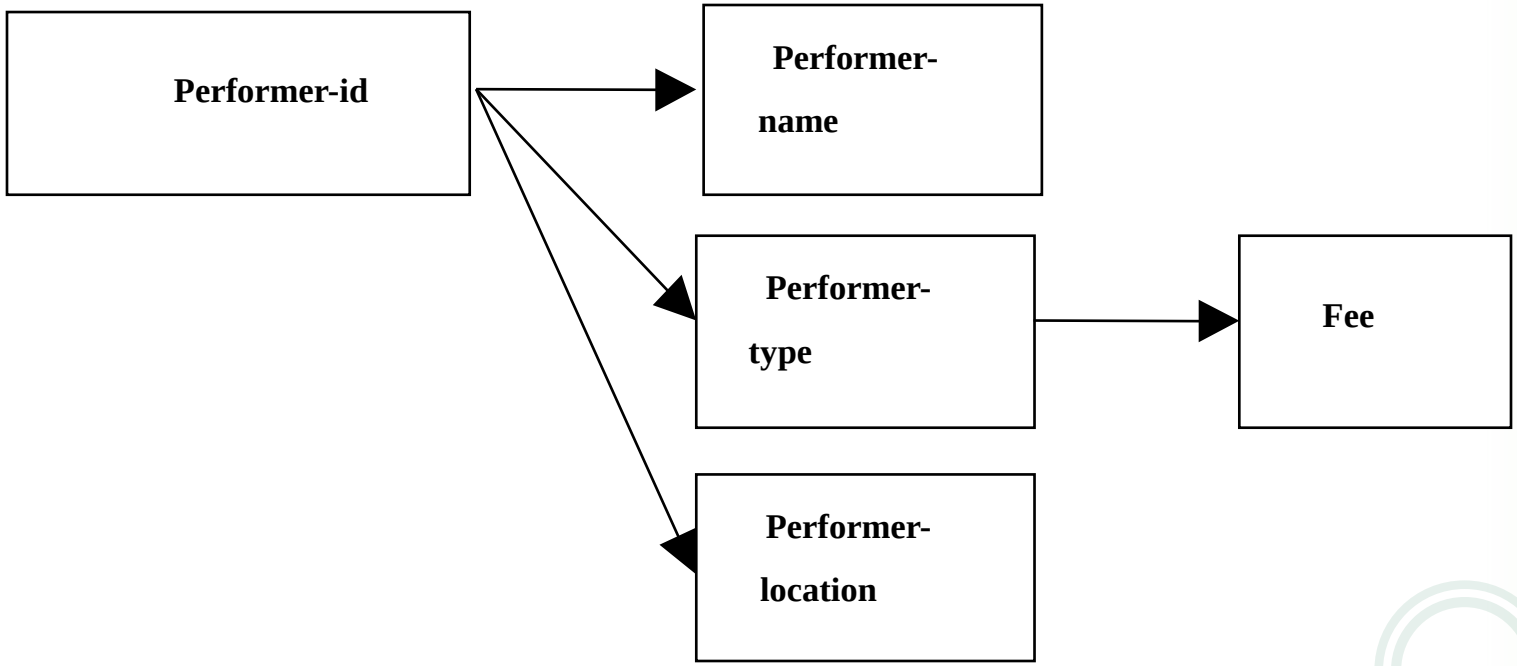
Functional Dependencies (FDs)

- Direct dependency (fully functional dependency): All attributes in a R must be fully functionally dependent on the primary key (or the PK is a determinant of all attributes in R)



Functional Dependencies (FDs)

- Indirect dependency (transitive dependency): Value of an attribute is not determined directly by the primary key



Functional Dependencies (FDs)

- Partial dependency
  - **Composite determinant** - more than one value is required to determine the value of another attribute, the combination of values is called a composite determinant

    EMP_PROJ(SSN, PNUMBER, HOURS, ENAME, PNAME, PLOCATION)

    {SSN, PNUMBER} -> HOURS

  - **Partial dependency** - if the value of an attribute does not depend on an entire composite determinant, but only part of it, the relationship is known as the partial dependency

    SSN -> ENAME

    PNUMBER -> {PNAME, PLOCATION}

Functional Dependencies (FDs)

- Inference Rules for FDs

Armstrong's inference rules:

IR1. (**Reflexive**) If $Y \subseteq X$, then X -> Y

IR2. (**Augmentation**) If X -> Y, then XZ -> YZ

(Notation: XZ stands for $X \cup Z$)

IR3. (**Transitive**) If X -> Y and Y -> Z, then X -> Z

Some **additional inference rules** that are useful:

(**Decomposition**) If X -> YZ, then X -> Y and X -> Z

(**Union**) If X -> Y and X -> Z, then X -> YZ

(**Psuedotransitivity**) If X -> Y and WY -> Z, then WX -> Z

Functional Dependencies (FDs)

- Two sets of FDs F and G are **equivalent** if $F^+ = G^+$
- Definition: F **covers** G if $G^+ \subseteq F^+$. F and G are equivalent if F covers G and G covers F
- There is an algorithm for checking equivalence of sets of FDs (see chapter 10 [1])

Functional Dependencies (FDs)

- A set of FDs is **minimal** if it satisfies the following conditions:
  - (1) Every dependency in F has a single attribute for its RHS.
  - (2) We cannot remove any dependency from F and have a set of dependencies that is equivalent to F.
  - (3) We cannot replace any dependency X -> A in F with a dependency Y -> A, where Y proper-subset-of X (Y subset-of X) and still have a set of dependencies that is equivalent to F

Normalization

- **Normalization**: The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations
- Two new concepts:
  - A **Prime attribute** must be a member of *some candidate key*
  - A **Nonprime attribute** is not a prime attribute: it is not a member of any candidate key

Normalization

- 1NF and dependency problems
- 2NF – solves partial dependency
- 3NF – solves indirect dependency
- BCNF – well-normalized relations

Normalization

- First normal form (1NF): Disallows composite attributes, multivalued attributes, and **nested relations**

Normalization

- Second normal form (2NF) - all attributes must be **fully functionally dependent** on the primary key

Normalization

- A relation schema R is in **third normal form** (**3NF**) if it is in 2NF *and* no non-prime attribute A in R is **transitively dependent** on the primary key

NOTE:

In X -> Y and Y -> Z, with X as the primary key, we consider this a problem only if Y is not a candidate key. When Y is a candidate key, there is no problem with the transitive dependency .

E.g., Consider EMP (SSN, Emp#, Salary).

Here, SSN -> Emp# -> Salary and Emp# is a candidate key

General Normal Form Definitions

- A relation schema R is in **second normal form** (**2NF**) if every non-prime attribute A in R is fully functionally dependent on *every key* of R
- A relation schema R is in **third normal form** (**3NF**) if whenever a FD X -> A holds in R, then either:

  (a) X is a superkey of R, or

  (b) A is a prime attribute of R

Normalization

- A relation schema R is in **Boyce-Codd Normal Form** (**BCNF**) if whenever an FD X -> A holds in R, then X is a superkey of R

Course Outline

Lecture	Content
Database System Concepts & Architecture	-Introduction to Data Models, Database Systems -Three-Level Architecture & Data Independence -Modern Database Applications
Entity-Relationship (ER) Model	-ER Model -Introduction to Enhanced ER (EER) Model
Relational Model	-Relational Data Model -ER- & EER-to-Relational Mapping -Relational Algebra
SQL	-Data Definition Language (DDL) -Data Manipulation Language (DML) -Introduction to Triggers & Stored Procedures
Database Design Theory & Methodology	-Functional Dependencies & Normalization -Relational Database Design: Algorithms
Data Storage, Indexing, Query Processing & Physical Design	-Hashing & Indexing Structures (B-tree & R-tree families) -Physical Database Design
Database Security	-Discretionary & Mandatory Access Control (DAC & MAC) -Flow Control, Inference Problem -Security Issues in Modern Data Management Systems
Emerging Technologies & Applications	-Introduction to XML, Data Mining & Data Warehousing, GIS, M-Commerce & LBS -Emerging Database Technologies & Applications

Outline

- Data Storage
  - Disk Storage Devices
  - Files of Records
  - Operations on Files
  - Unordered Files
  - Ordered Files
  - Hashed Files
  - RAID Technology
- Indexing Structures for Files
  - Types of Single-level Ordered Indexes
  - Multilevel Indexes
  - Dynamic Multilevel Indexes Using B-Trees and B+-Trees
  - Indexes on Multiple Keys

Disk Storage Devices

- Disks are divided into concentric circular **tracks** on each disk **surface**.
- A track is divided into smaller **blocks** or **sectors**
- A **read-write head** moves to the track that contains the block to be transferred
- A physical disk block address consists of:
  - a cylinder number
  - the track number or surface number (within the cylinder)
  - and block number (within track).

Disk Storage Devices (contd.)

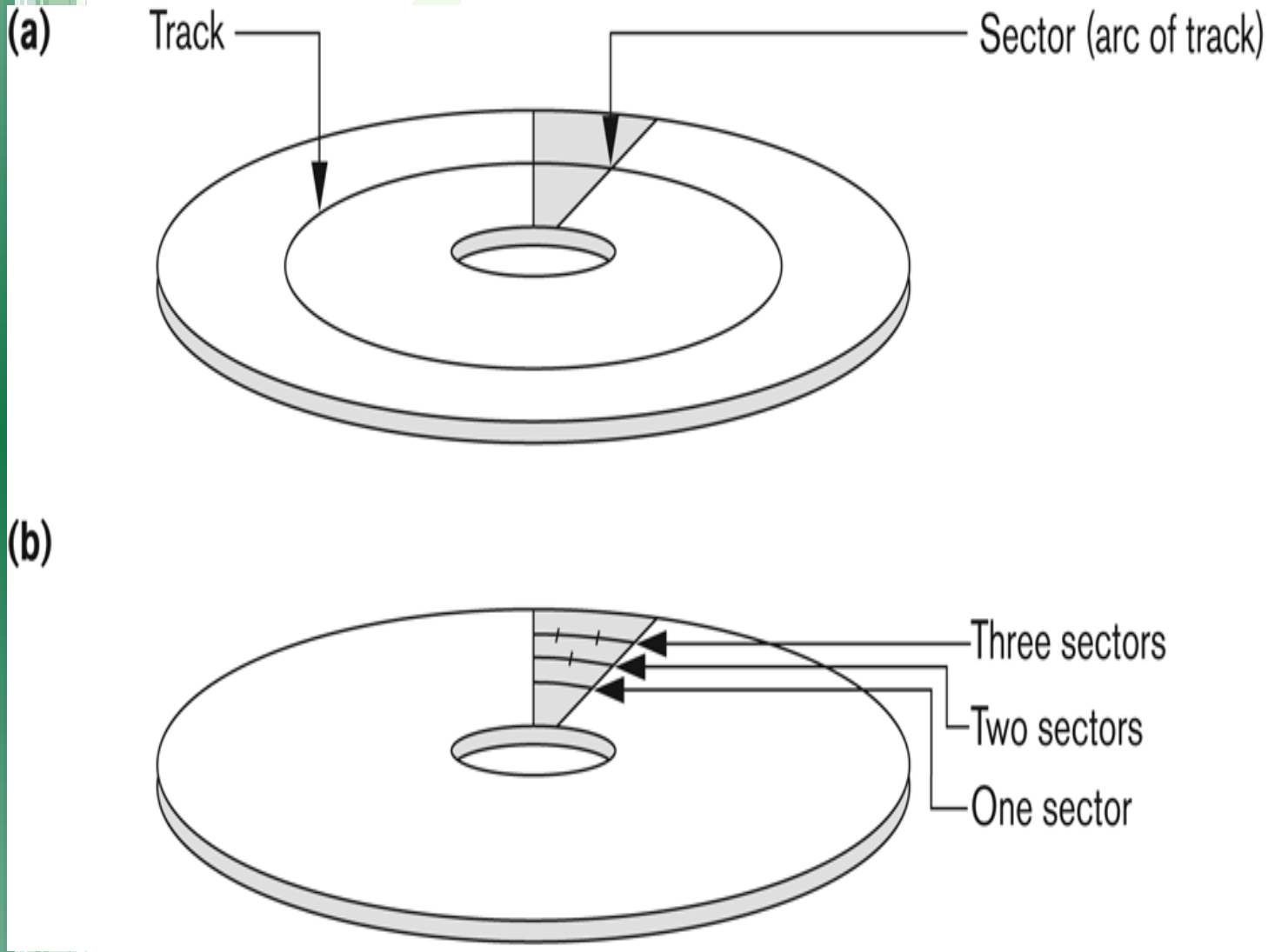
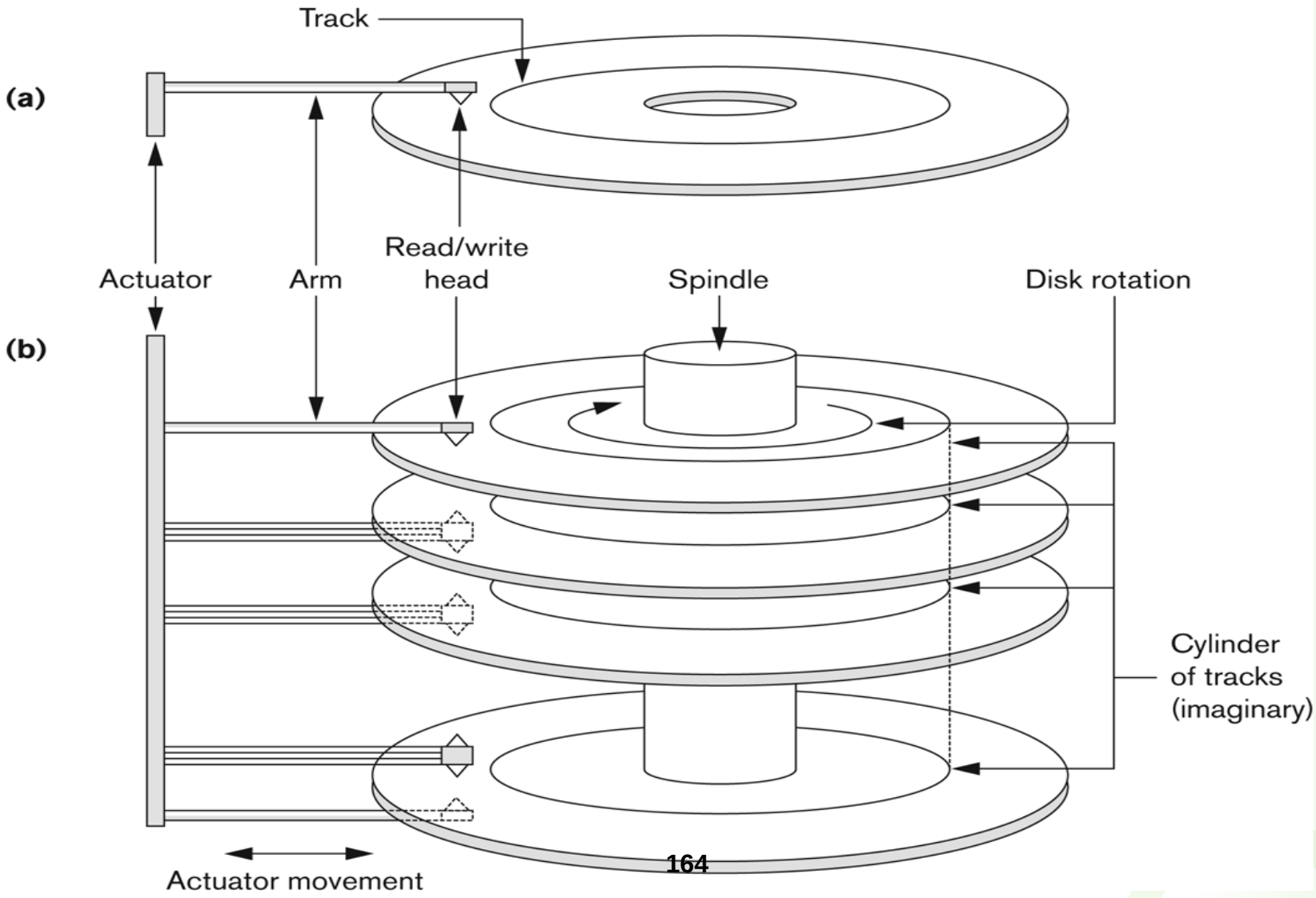


Figure 13.2
Different sector organizations on disk. (a) Sectors subtending a fixed angle. (b) Sectors maintaining a uniform recording density.

Disk Storage Devices (contd.)

Figure 13.1
(a) A single-sided disk with read/write hardware. (b) A disk pack with read/write hardware.



Disk Storage Devices

- **Records**: Fixed and variable length records
  - Records contain fields which have values of a particular type
- **Blocking**: Refers to storing a number of records in one block on the disk.
  - Blocking factor (**bfr**) refers to the number of records per block.
- **Spanned Records**:
  - Refers to records that exceed the size of one or more blocks and hence span a number of blocks.

Files of Records

- A **file** is a *sequence* of records, where each record is a collection of data values (or data items).
- A **file descriptor** (or **file header**) includes information that describes the file
- A file can have **fixed-length** records or **variable-length** records.
- File records can be **unspanned** or **spanned**

Unordered Files

- Also called a **heap** or a **pile** file.
- New records are inserted at the end of the file.
- A **linear search** through the file records is necessary to search for a record.
  - This requires reading and searching half the file blocks on the average, and is hence quite expensive.
- Record insertion is quite efficient.
- Reading the records in order of a particular field requires sorting the file records.

Ordered Files

- Also called a **sequential** file.
- File records are kept sorted by the values of an *ordering field*.
- Insertion is expensive: records must be inserted in the correct order.
  - It is common to keep a separate unordered *overflow* (or *transaction*) file for new records to improve insertion efficiency; this is periodically merged with the main ordered file.
- A **binary search** can be used to search for a record on its *ordering field* value.
  - This requires reading and searching $\log_2$ of the file blocks on the average, an improvement over linear search.
- Reading the records in order of the ordering field is quite efficient.

Indexes as Access Paths

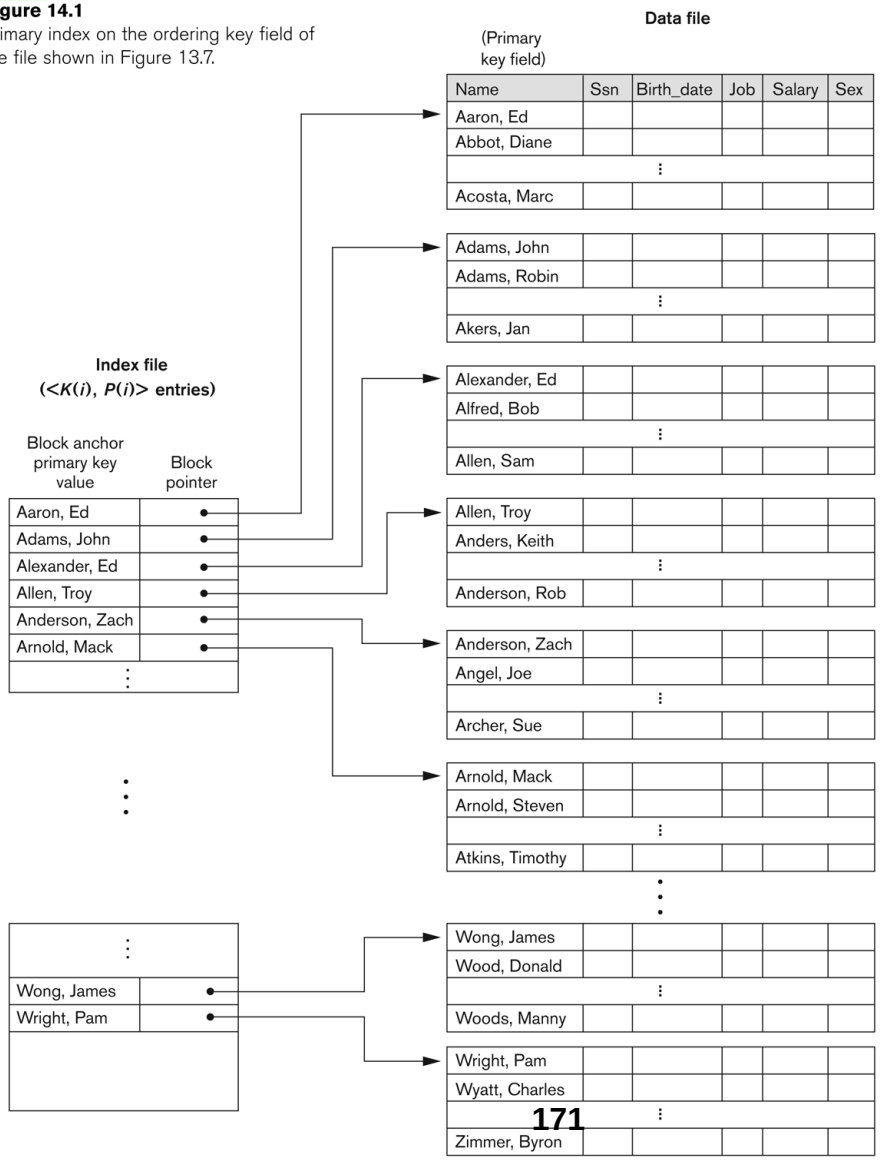
- A single-level index is an auxiliary file that makes it more efficient to search for a record in the data file.
- The index is usually specified on one field of the file (although it could be specified on several fields)
- One form of an index is a file of entries <**field value, pointer to record>**, which is ordered by field value
- The index is called an access path on the field.

Types of Single-Level Indexes

- Primary Index
  - Defined on an ordered data file
  - The data file is ordered on a **key field**
  - Includes one index entry *for each block* in the data file; the index entry has the key field value for the *first record* in the block, which is called the *block anchor*
  - A similar scheme can use the *last record* in a block.
  - A primary index is a nondense (sparse) index, since it includes an entry for each disk block of the data file and the keys of its anchor record rather than for every search value.

Primary index on the ordering key field

Figure 14.1
Primary index on the ordering key field of the file shown in Figure 13.7.

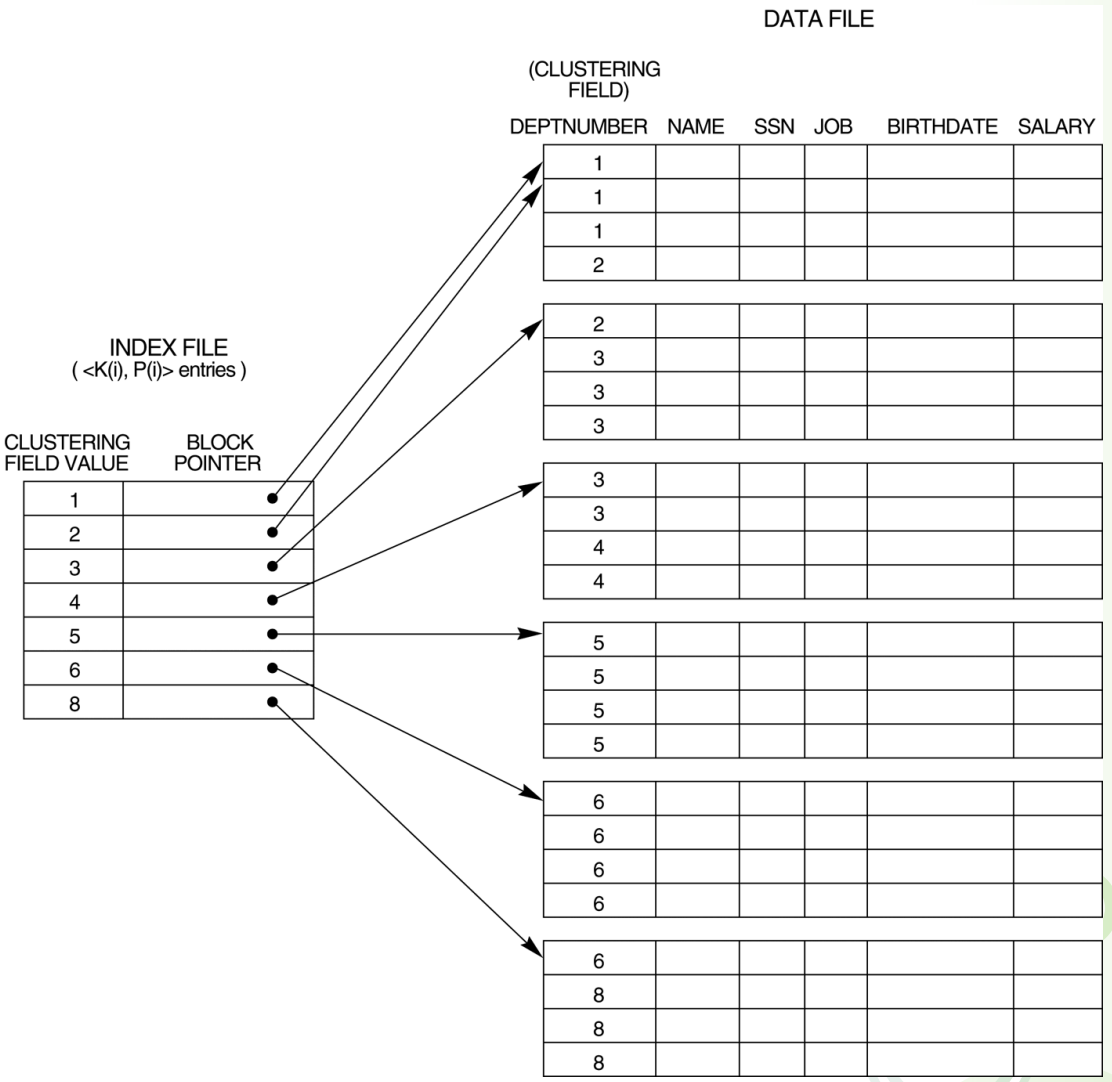


Types of Single-Level Indexes

- Clustering Index
  - Defined on an ordered data file
  - The data file is ordered on a *non-key field* unlike primary index, which requires that the ordering field of the data file have a distinct value for each record.
  - Includes one index entry *for each distinct value* of the field; the index entry points to the first data block that contains records with that field value.
  - It is another example of *nondense* index where Insertion and Deletion is relatively straightforward with a clustering index.

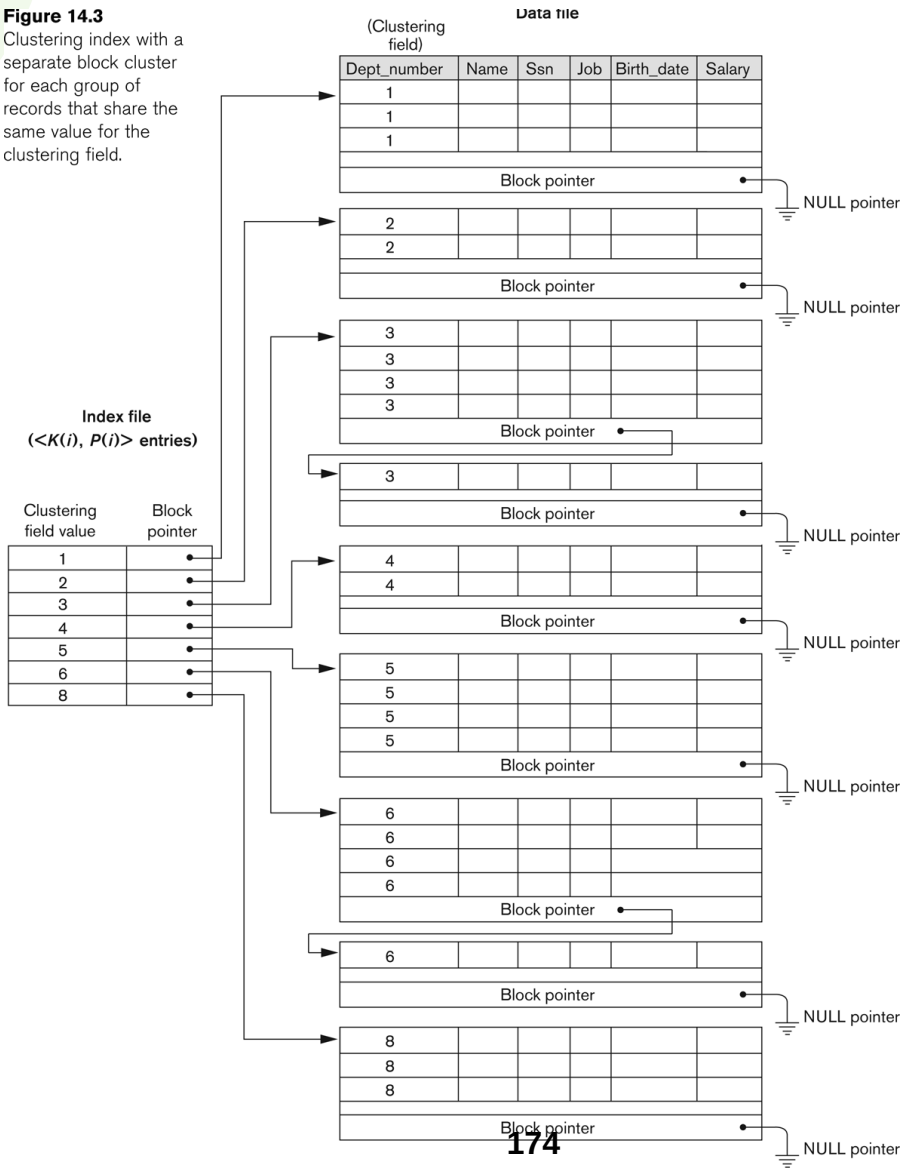
A Clustering Index Example

- FIGURE 14.2
A clustering index on the DEPTNUMBER ordering non-key field of an EMPLOYEE file.



Another Clustering Index Example

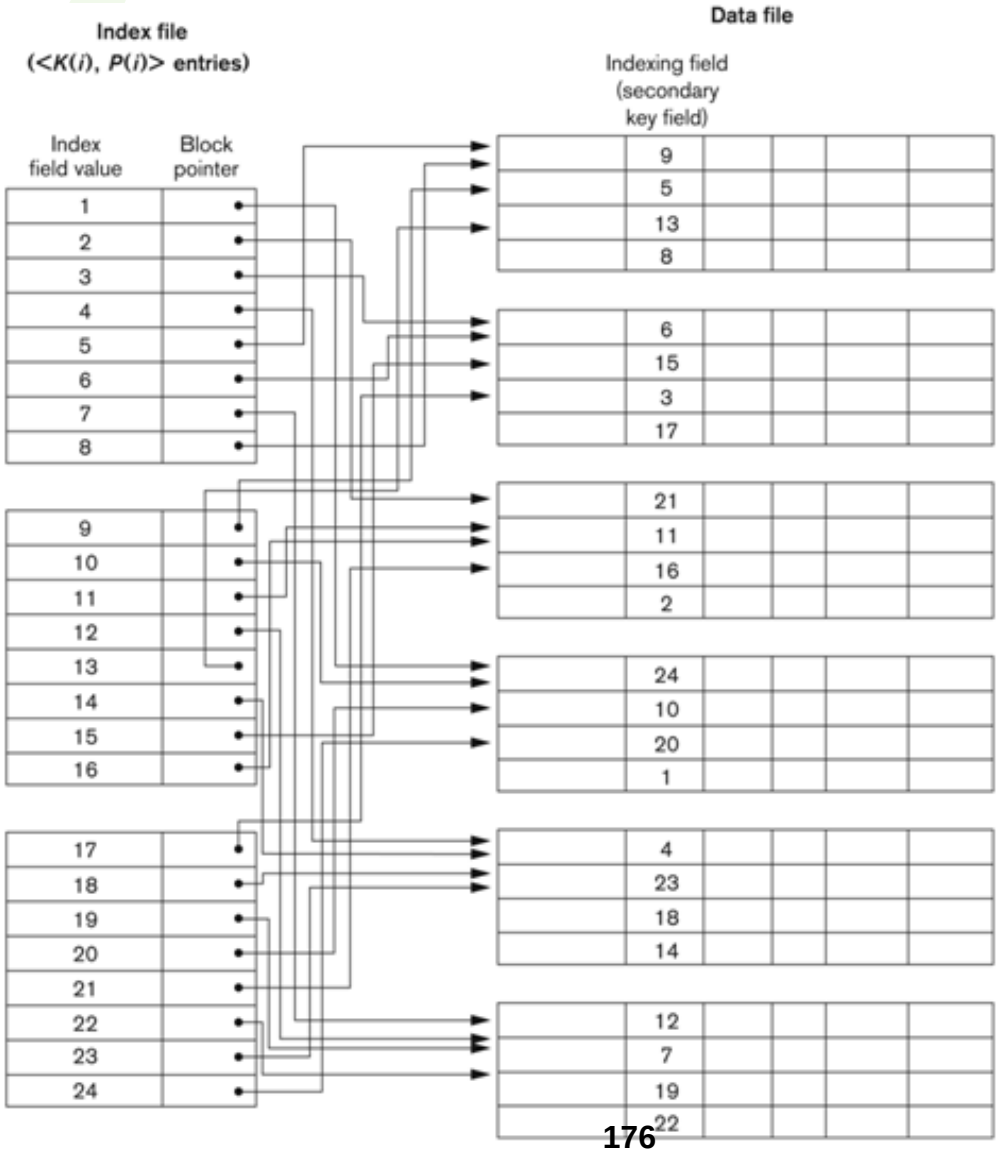
Figure 14.3
Clustering index with a separate block cluster for each group of records that share the same value for the clustering field.



Types of Single-Level Indexes

- Secondary Index
  - A secondary index provides a secondary means of accessing a file for which some primary access already exists.
  - The secondary index may be on a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.
  - The index is an ordered file with two fields.
    - The first field is of the same data type as some **non-ordering field** of the data file that is an indexing field.
    - The second field is either a **block** pointer or a record pointer.
    - There can be *many* secondary indexes (and hence, indexing fields) for the same file.
  - Includes one entry *for each record* in the data file; hence, it is a *dense index*

Example of a Dense Secondary Index



An Example of a Secondary Index

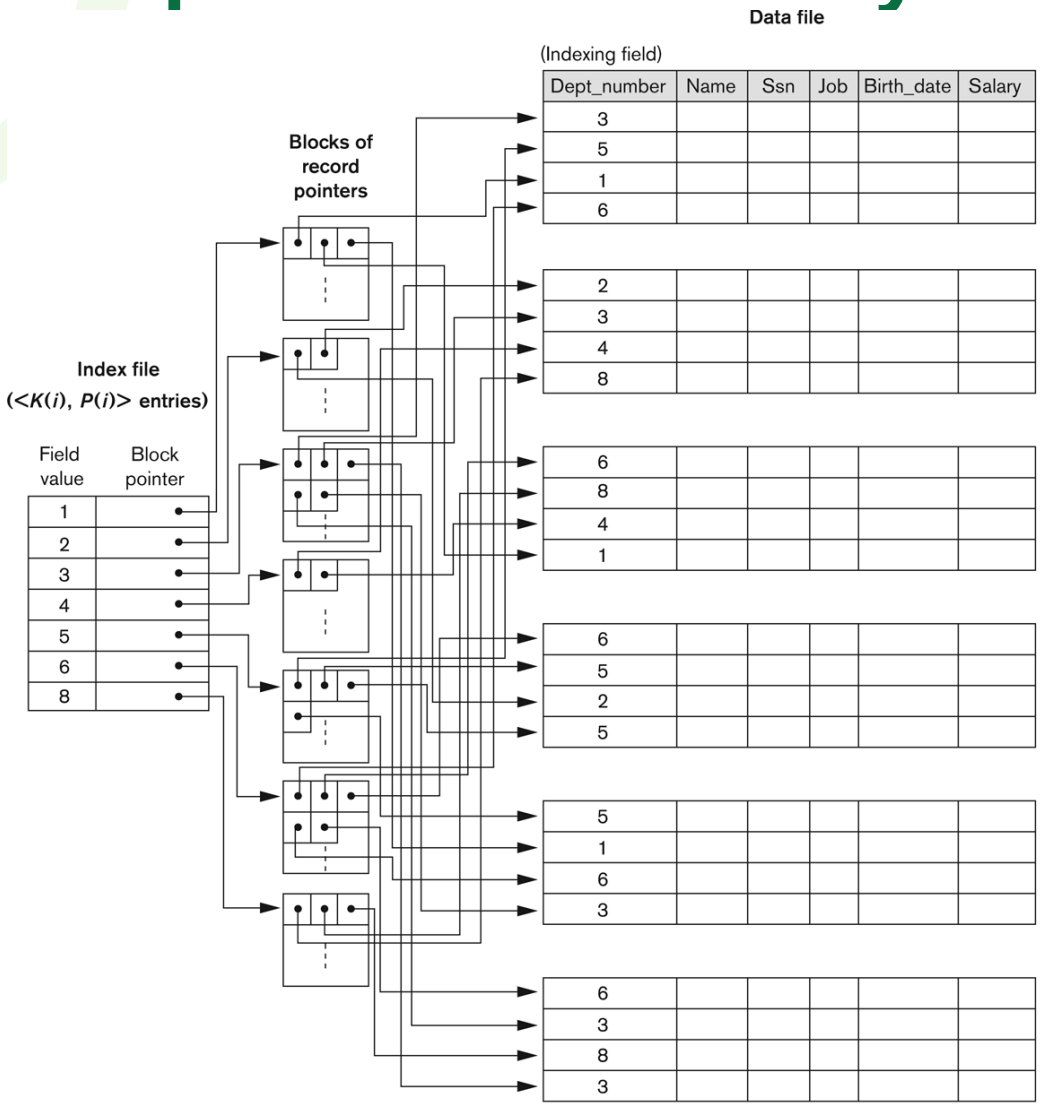


Figure 14.5
A secondary index (with record pointers) on a nonkey field implemented using one level of indirection so that index entries are of fixed length and have unique field values.

Properties of Index Types

TABLE 14.2 PROPERTIES OF INDEX TYPES

TYPE OF INDEX	NUMBER OF (FIRST-LEVEL) INDEX ENTRIES	DENSE OR NONDENSE	BLOCK ANCHORING ON THE DATA FILE
Primary	Number of blocks in data file	Nondense	Yes
Clustering	Number of distinct index field values	Nondense	Yes/no[a]
Secondary (key)	Number of records in data file	Dense	No
Secondary (nonkey)	Number of records[b] or Number of distinct index field values[c]	Dense or Nondense	No

[a]Yes if every distinct value of the ordering field starts a new block; no otherwise.

[b]For option 1.

[c]For options 2 and 3.

Multi-Level Indexes

- Because a single-level index is an ordered file, we can create a primary index *to the index itself*;
  - In this case, the original index file is called the *first-level index* and the index to the index is called the *second-level index*.
- We can repeat the process, creating a third, fourth, ..., top level until all entries of the *top level* fit in one disk block
- A multi-level index can be created for any type of first-level index (primary, secondary, clustering) as long as the first-level index consists of *more than one* disk block

A Two-level Primary Index

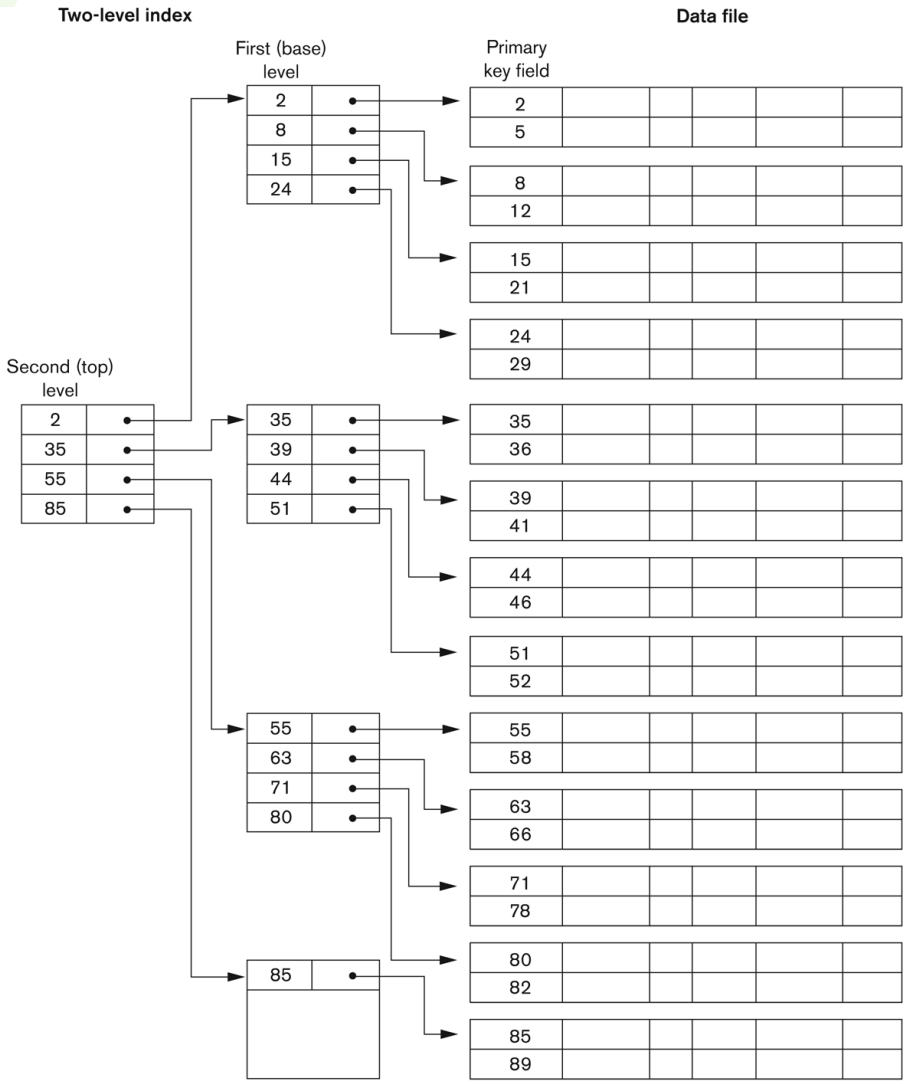


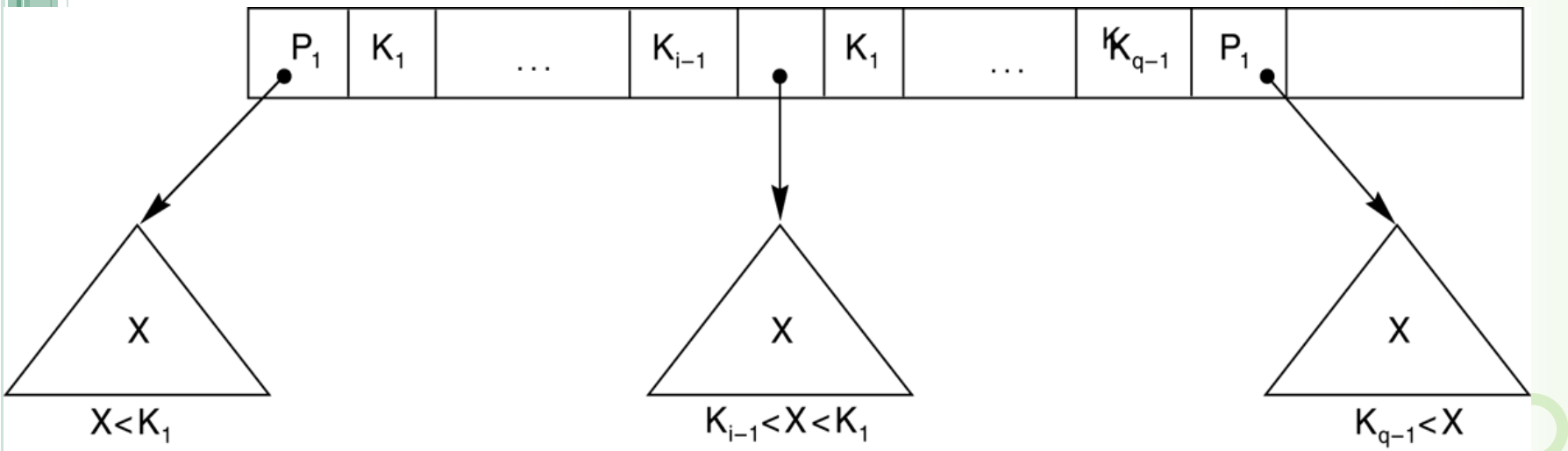
Figure 14.6
A two-level primary index resembling ISAM (Index Sequential Access Method) organization.

Multi-Level Indexes

- Such a multi-level index is a form of *search tree*
  - However, insertion and deletion of new index entries is a severe problem because every level of the index is an *ordered file*.

A Node in a Search Tree with Pointers to Subtrees below It

- FIGURE 14.8



Difference between B-tree and B+-tree

- In a B-tree, pointers to data records exist at all levels of the tree
- In a B+-tree, all pointers to data records exists at the leaf-level nodes
- A B+-tree can have less levels (or higher capacity of search values) than the corresponding B-tree

Q&A

Review

Trong đại số quan hệ, những toán tử nào sau đây có tính giao hoán (commutative)

A. Hội (UNION)
B. Giao (INTERSECTION)
C. Hiệu (DIFFERENCE)
D. Tích đề-các (CARTESIAN PRODUCT)
E. Chọn (SELECT)
F. Chiếu (PROJECT)

Review

Chọn phát biểu ĐÚNG. Hai quan hệ R(A1, A2, ...Am) và S(B1, B2, ..., Bn) là tương thích kiểu (type compatible) nếu:

A. m = n
B. R và S có cùng số lượng record
C. m = n và dom(Ai) = dom(Bi) với i = 1..m
D. Tất cả câu trên đều đúng

Review

Phép kết (join) nào sau đây đòi hỏi mỗi cặp thuộc tính kết (join attributes) phải có cùng tên trên hai quan hệ tham gia vào phép kết:

A. Theta join

B. Equijoin

C. Natural join

D. B và C đúng

Review

Trong SQL, khi áp dụng hàm AVG (tính giá trị trung bình), giá trị NULL sẽ được xử lý:

A. Thay giá trị null bằng 0 trước rồi mới thực hiện phép tính

B. Bỏ qua các hàng có giá trị null

C. Báo lỗi là không tính được do có giá trị Null

D. Tất cả đều sai

Review

Những đặc điểm (tiêu chí) nào sao đây dùng để đánh giá một cơ sở dữ liệu được thiết kế tốt (goodness):

A. Loại bỏ hoàn toàn giá trị NULL trong các bảng

B. Giảm thiểu giá trị NULL trong các bảng

C. Không xảy ra những bất thường khi cập nhật dữ liệu (update anomalies)

D. Giảm thiểu những bất thường khi cập nhật dữ liệu

Review

SQL là ngôn ngữ:

A. Thủ tục

B. Phi thủ tục

Review

Primary *index* là chỉ mục được chỉ định trên:

A. Trường khóa *(key field*) với các mẩu tin (record) được sắp thứ tự vật lý trên trường này, và dữ liệu trên trường này được phép trùng

B. Trường khóa (*key field*) với các mẩu tin (record) được sắp thứ tự vật lý trên trường này, và dữ liệu trên trường này không được trùng

C. Trường khóa (key field) với các mẩu tin (record) không được sắp thứ tự vật lý trên trường này, và dữ liệu trên trường này được phép trùng

D. Trường khóa (key field) với các mẩu tin (record) không được sắp thứ tự vật lý trên trường này, và dữ liệu trên trường này không được phép trùng

Review

Blocking factor (**bfr**) là:

A. Số field trong từng record

B. Số record trong từng block

C. Số block trong từng track

D. Số field trong từng block

Review

Chọn các phát biểu ĐÚNG:

A. Clustering index chỉ được áp dụng đối với file có sắp xếp thứ tự theo thuộc tính cần đánh index

B. Chỉ có thể đánh tối đa 3 index trên cùng 1 file (ứng với 3 loại index)

C. Clustering index là loại dense index

D. Với clustering index, số hàng trong index file bằng với số block trong data file

Review

- Cho quan hệ R(A, B, C, D, E, F, G, H) và tập phụ thuộc hàm sau:

1) A, B, C → F
2) F → G
3) G → H
4) B, C → D, E

- Tìm khoá, chuẩn hoá: 2NF, 3NF

Course Outline

Lecture	Content
Database System Concepts & Architecture	-Introduction to Data Models, Database Systems -Three-Level Architecture & Data Independence -Modern Database Applications
Entity-Relationship (ER) Model	-ER Model -Introduction to Enhanced ER (EER) Model
Relational Model	-Relational Data Model -ER- & EER-to-Relational Mapping -Relational Algebra
SQL	-Data Definition Language (DDL) -Data Manipulation Language (DML) -Introduction to Triggers & Stored Procedures
Database Design Theory & Methodology	-Functional Dependencies & Normalization -Relational Database Design: Algorithms
Data Storage, Indexing, Query Processing & Physical Design	-Hashing & Indexing Structures (B-tree & R-tree families) -Physical Database Design
Database Security	-Discretionary & Mandatory Access Control (DAC & MAC) -Flow Control, Inference Problem -Security Issues in Modern Data Management Systems
Emerging Technologies & Applications	-Introduction to XML, Data Mining & Data Warehousing, GIS, M-Commerce & LBS -Emerging Database Technologies & Applications


Q&A