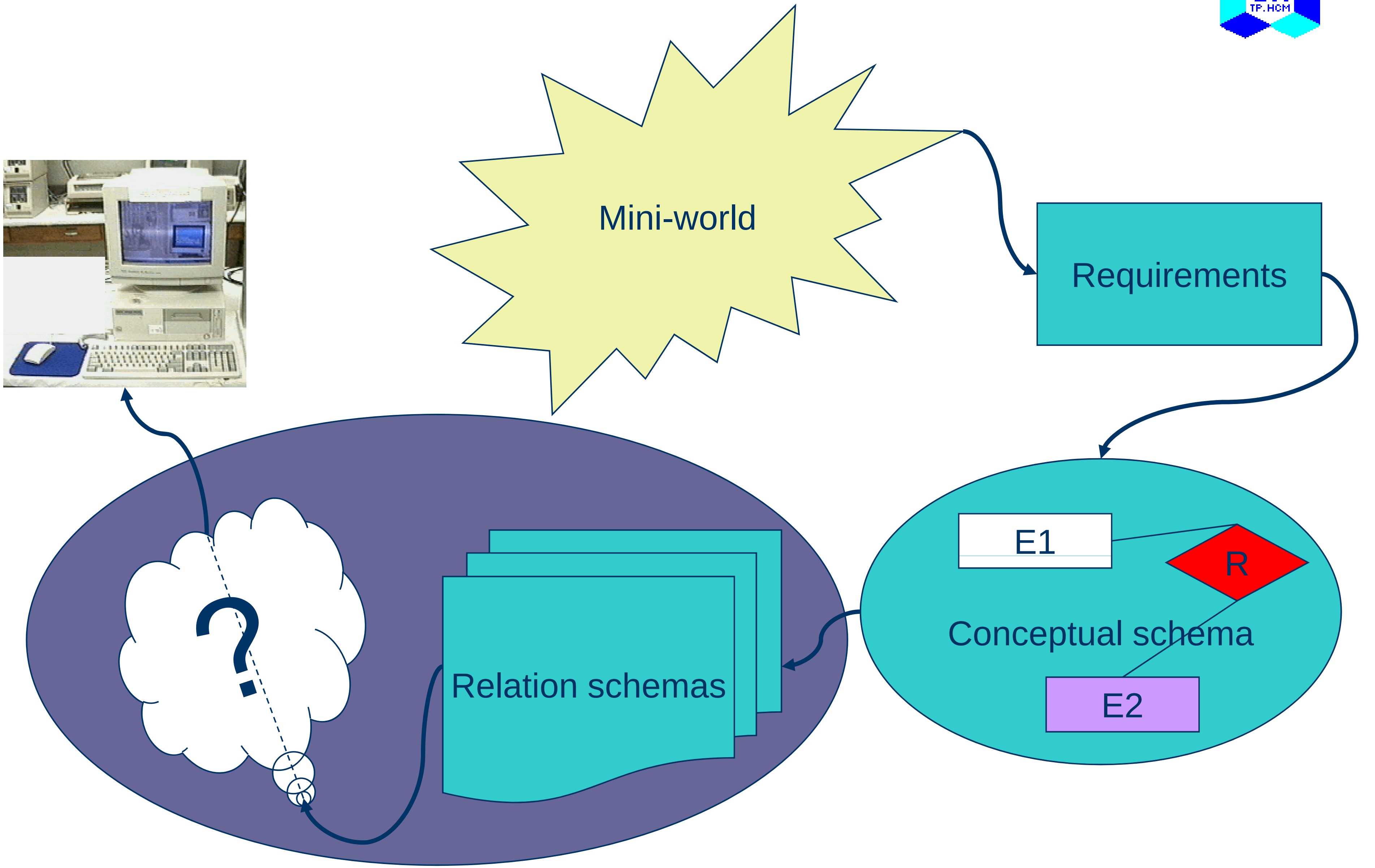
Logical DB Design & Relational Model

Dr. Dang Tran Khanh
Department of Information Systems
Faculty of Computer Science and Engineering
khanh@cse.hcmut.edu.vn

Overview of DB Design Phases



Top-Down DB Design Phases

Outline

- Relational Model
- Relational Database Design by ER/EER-to-Relational Mapping
- Functional Dependencies & Normalization
- Exercise
- Reading Suggestion: [1] chapters 7, 10

Relational Model

- Basic Concepts: relational data model, relation schema, domain, tuple, cardinality & degree, database schema, etc.
- Relational Integrity Constraints
  - key, primary key & foreign key
  - entity integrity constraint
  - referential integrity
- Update Operations on Relations

Basic Concepts

- The relational model of data is based on the concept of a relation.
- A relation is a mathematical concept based on the ideas of sets
- The model was first proposed by E.F. Codd of IBM in 1970 in the following paper: "*A Relational Model for Large Shared Data Banks*," Communications of the ACM, June 1970

Basic Concepts

- **Relational data model**: represents a database in the form of **relations** - 2-dimensional table with rows and columns of data. A database may contain one or more such tables. A relation schema is used to *describe* a relation
- **Relation schema**: R(A1, A2,…, An) is made up of a relation name R and a list of attributes A1, A2, . . ., An. Each **attribute** Ai is the name of a role played by some domain D in the relation schema R. R is called the **name** of this relation
- The **degree** of a relation is the number of attributes n of its relation schema.
- **Domain** D: D is called the **domain** of Ai and is denoted by dom(Ai). It is a set of atomic values and a set of integrity constraints

  STUDENT(Name, SSN, HomePhone, Address, OfficePhone, Age, GPA)
  Degree = ??
  dom(GPA) = ??

Basic Concepts

- **Tuple**: row/record in table
- **Cardinality**: number of tuples
- **Database schema** S = {R1, R2,…, Rm}
- A **relation** (or **relation state**, **relation instance**) r of the relation schema R(A1, A2, . . ., An), also denoted by r(R), is a set of n-tuples r = {t1, t2, . . ., tm}. Each **n-tuple** t is an ordered list of n values t = <v1, v2, . . ., vn>, where each value vi, i=1..n, is an element of dom(Ai) or is a special **null** value. The ith value in tuple t, which corresponds to the attribute Ai, is referred to as t[Ai]

Relational data model
Database schema
Relation schema
Relation
Tuple
Attribute

Basic Concepts

- A relation can be conveniently represented by a table, as the example shows
- The columns of the tabular relation represent attributes
- Each attribute has a distinct name, and is always referenced by that name, never by its position
- Each row of the table represents a tuple. The ordering of the tuples is immaterial and all tuples must be distinct

Relation name

Attributes

Tuples

STUDENT	Name	SSN	HomePhone	Address	OfficePhone	Age	GPA
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	null	19	3.21
	Katherine Ashly	381-62-1245	375-4409	125 Kirby Road	null	18	2.89
	Dick Davidson	422-11-2320	null	3452 Elgin Road	749-1253	25	3.53
	Charles Cooper	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
	Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	null	19	3.25

Basic Concepts

Alternative Terminology for Relational Model

Formal terms	Alternative 1	Alternative 2
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

Basic Concepts

Mathematical Definition of Relation

- Consider two sets, D_1 & D_2, where $D_1 = \{2, 4\}$ and $D_2 = \{1, 3, 5\}$
- Cartesian product, $D_1 \mathrm{x} D_2$, is set of all ordered pairs, where first element is member of D_1 and second element is member of D_2

$$D_1 \mathrm{x} D_2 = \{(2, 1), (2, 3), (2, 5), (4, 1), (4, 3), (4, 5)\}$$

Basic Concepts

Mathematical Definition of Relation

- Any subset of Cartesian product is a relation; e.g.

  $R = \{(2, 1), (4, 1)\}$

- May specify which pairs are in relation using some condition for selection; e.g.
  - second element is 1:

    $R = \{(x, y) \mid x \in D_1, y \in D_2, \text{ and } y = 1\}$
  - first element is always twice the second:

    $S = \{(x, y) \mid x \in D_1, y \in D_2, \text{ and } x = 2y\}$

Basic Concepts

Mathematical Definition of Relation

- Consider three sets D_1, D_2, D_3 with Cartesian Product $D_1 \times D_2 \times D_3$; e.g.

$D_1 = \{1, 3\}$ $D_2 = \{2, 4\}$ $D_3 = \{5, 6\}$

$D_1 \times D_2 \times D_3 = \{(1,2,5), (1,2,6), (1,4,5), (1,4,6), (3,2,5), (3,2,6), (3,4,5), (3,4,6)\}$

- Any subset of these ordered triples is a relation

Basic Concepts

Mathematical Definition of Relation

- Cartesian product of *n* sets ($D_1, D_2, \ldots, D_n$) is:

$$D_1 \times D_2 \times \ldots \times D_n = \{(d_1, d_2, \ldots, d_n) \mid d_1 \in D_1, d_2 \in D_2, \ldots, d_n \in D_n\}$$

- Any set of *n*-tuples from this Cartesian product is a relation on the *n* sets

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
- But …

Relational Integrity Constraints

- Null
  - Represents value for an attribute that is currently unknown or not applicable for tuple
  - Deals with incomplete or exceptional data
  - Represents the absence of a value and is not the same as zero or spaces, which are values

Relational Integrity Constraints
Key Constraints

- **Superkey** of R: A set of attributes SK of R such that no two tuples *in any valid relation instance r(R)* will have the same value for SK. That is, for any distinct tuples t1 and t2 in r(R), t1[SK] ≠ t2[SK]
- **Key** of R: A "minimal" superkey; that is, a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey

  Example: The CAR relation schema:

  CAR(<u>State</u>, <u>Reg#</u>, SerialNo, Make, Model, Year)

  has two keys Key1 = {State, Reg#}, Key2 = {SerialNo}, which are also superkeys. {SerialNo, Make} is a superkey but *not* a key.
- If a relation has *several* **candidate keys**, one is chosen arbitrarily to be the **primary key**. The primary key attributes are *underlined*

Relational Integrity Constraints

Key Constraints

- The CAR relation, with two candidate keys: LicenseNumber and EngineSerialNumber

CAR	LicenseNumber	EngineSerialNumber	Make	Model	Year
	Texas ABC-739	A69352	Ford	Mustang	96
	Florida TVP-347	B43696	Oldsmobile	Cutlass	99
	New York MPO-22	X83554	Oldsmobile	Delta	95
	California 432-TFY	C43742	Mercedes	190-D	93
	California RSK-629	Y82935	Toyota	Camry	98
	Texas RSK-629	U028365	Jaguar	XJS	98

Relational Integrity Constraints
Entity Integrity

- **Relational Database Schema**: A set S of relation schemas that belong to the same database. S is the *name* of the **database**

$$S = \{R_1, R_2, ..., R_n\}$$

- **Entity Integrity**: The *primary key attributes* PK of each relation schema R in S cannot have null values in any tuple of r(R). This is because primary key values are used to *identify* the individual tuples.

$$t[PK] \neq \text{null for any tuple } t \text{ in } r(R)$$

- Note: Other attributes of R may be similarly constrained to disallow null values, even though they are not members of the primary key

Relational Integrity Constraints
Referential Integrity

- A constraint involving *two* relations (the previous constraints involve a *single* relation)
- Used to specify a *relationship* among tuples in two relations: the **referencing relation** and the **referenced relation**
- Tuples in the *referencing relation* R_1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the *referenced relation* R_2. A tuple t_1 in R_1 is said to **reference** a tuple t_2 in R_2 if $t_1[FK] = t_2[PK]$
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from $R_1.FK$ to R_2

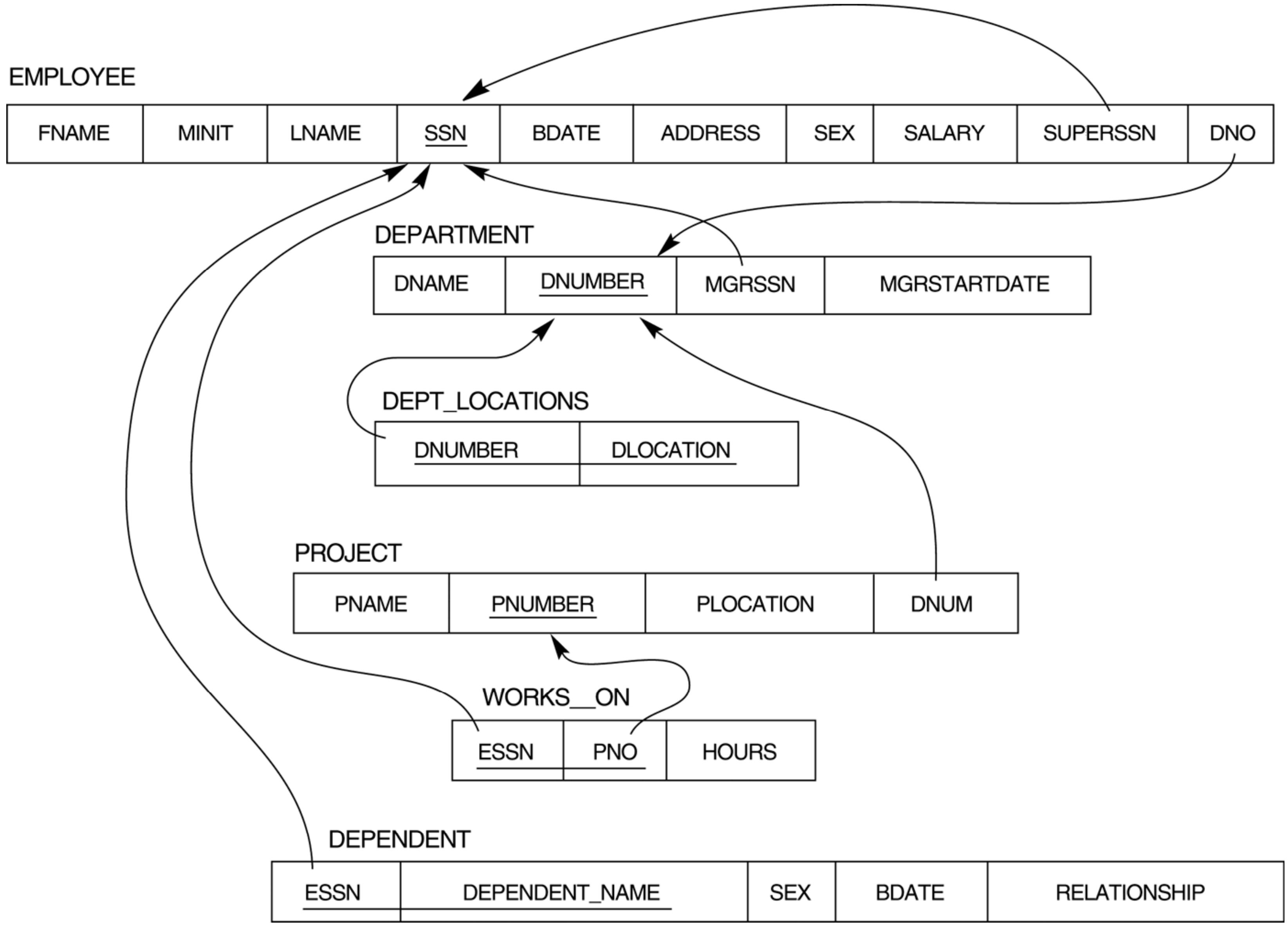
Relational Integrity Constraints

Referential Integrity

Statement of the constraint

- The value in the foreign key column (or columns) FK of the the **referencing relation** R_1 can be either:
  - (1) a value of an existing primary key value of the corresponding primary key PK in the **referenced relation** $R_{2,}$, or
  - (2) a NULL
- In case (2), the FK in R_1 should not be a part of its own primary key

Referential integrity constraints displayed on the COMPANY relational database schema



Relational Integrity Constraints

Other Types of Constraints

- Semantic Integrity Constraints:
  - based on application semantics and cannot be expressed by the model per se
  - E.g., “the max. no. of hours per employee for all projects he or she works on is 56 hrs per week”
  - A *constraint specification language* may have to be used to express these
  - SQL-99 allows triggers and ASSERTIONS to allow for some of these
- State/static constraints (so far)
- Transition/dynamic constraints: e.g., “the salary of an employee can only increase”

Relational Model

-
-
  -
  -
  -
- Update Operations on Relations

Update Operations on Relations

- INSERT a tuple
- DELETE a tuple
- MODIFY a tuple

- Integrity constraints should not be violated by the update operations

Update Operations on Relations

- **Insertion**: to insert a new tuple t into a relation R. When inserting a new tuple, it should make sure that the database constraints are not violated:
  - The value of an attribute should be of the correct data type (i.e. from the appropriate domain).
  - The value of a prime attribute (i.e. the key attribute) must not be null
  - The key value(s) must not be the same as that of an existing tuple in the same relation
  - The value of a foreign key (if any) must refer to an existing tuple in the corresponding relation
- Options if the constraints are violated
  - Homework !!

Update Operations on Relations

- **Deletion**: to remove an existing tuple t from a relation R. When deleting a tuple, the following constraints must not be violated:
  - The tuple must already exist in the database
  - The referential integrity constraint is not violated
- **Modification**: to change values of some attributes of an existing tuple t in a relation R

Update Operations on Relations

- In case of integrity violation, several actions can be taken:
  - Cancel the operation that causes the violation (REJECT option)
  - Perform the operation but inform the user of the violation
  - Trigger additional updates so the violation is corrected (CASCADE option, SET NULL option)
  - Execute a user-specified error-correction routine
- Again, homework !!

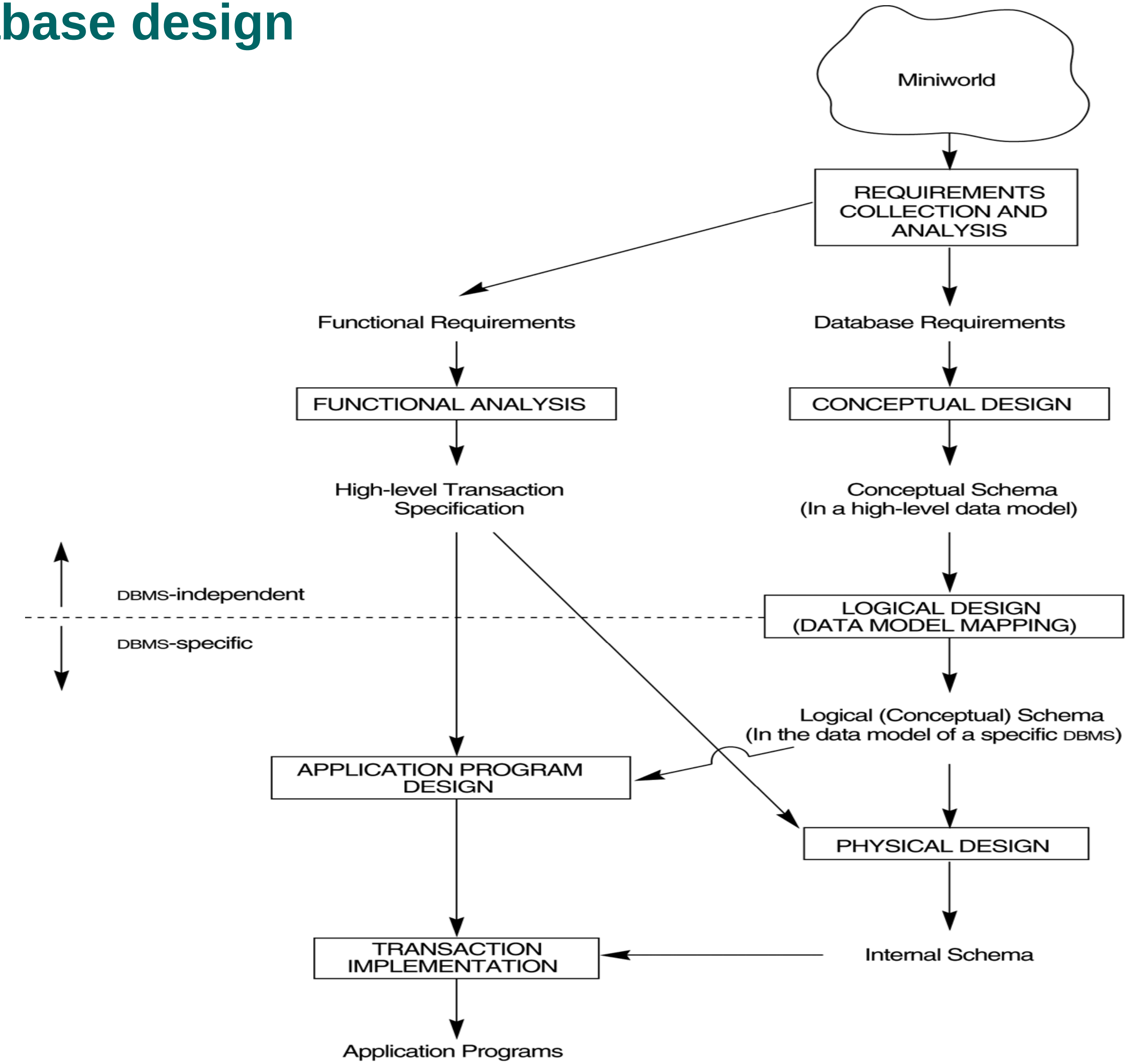
Relational Database Design by ER/EER-to-Relational Mapping

- A Review of Database Design
- Conceptual Database Design
- Logical Database Design
  - ER- & EER-to-Relational Mapping

A Review of Database Design

- Three main phases
  - Conceptual database design
  - Logical database design
  - Physical database design
- Detailed discussions: see [1] (chapter 12)
  - Six phases

A simplified diagram to illustrate the main phases of database design



A Review of Database Design

- Conceptual database design
  - The process of constructing a model of the data used in an enterprise, independent of *all* physical considerations
- Logical database design
  - The process of constructing a model of the data used in an enterprise based on a specific data model (e.g. relational), but independent of a particular DBMS and other physical considerations

A Review of Database Design

- Physical database design
  - The process of producing a description of the implementation of the database on secondary storage; it describes the base relations, file organizations, and indexes design used to achieve efficient access to the data, and any associated integrity constraints and security measures

Conceptual Database Design
Summarization

- Read [1]: chapters 3, 12 for details
- To build a conceptual data model of the data requirements of the enterprise
  - Model comprises entity types, relationship types, attributes and attribute domains, primary and alternate keys, structural and integrity constraints

Conceptual Database Design
Summarization

- Step 1: Identify entity types
- Step 2: Identify relationship types
- Step 3: Identify and associate attributes with entity or relationship types
- Step 4: Determine attribute domains
- Step 5: Determine candidate, primary, and alternate key attributes
- Step 6: Consider use of enhanced modeling concepts (optional step)
- Step 7: Check model for redundancy
- Step 8: Validate conceptual model against user transactions
- Step 9: Review conceptual data model with user

Conceptual Database Design

Summarization

- Step 1: Identify entity types
  - To identify the required entity types
- Step 2: Identify relationship types
  - To identify the important relationships that exist between the entity types
- Step 3: Identify and associate attributes with entity or relationship types
  - To associate attributes with the appropriate entity or relationship types and document the details of each attribute
- Step 4: Determine attribute domains
  - To determine domains for the attributes in the data model and document the details of each domain

Conceptual Database Design

Summarization

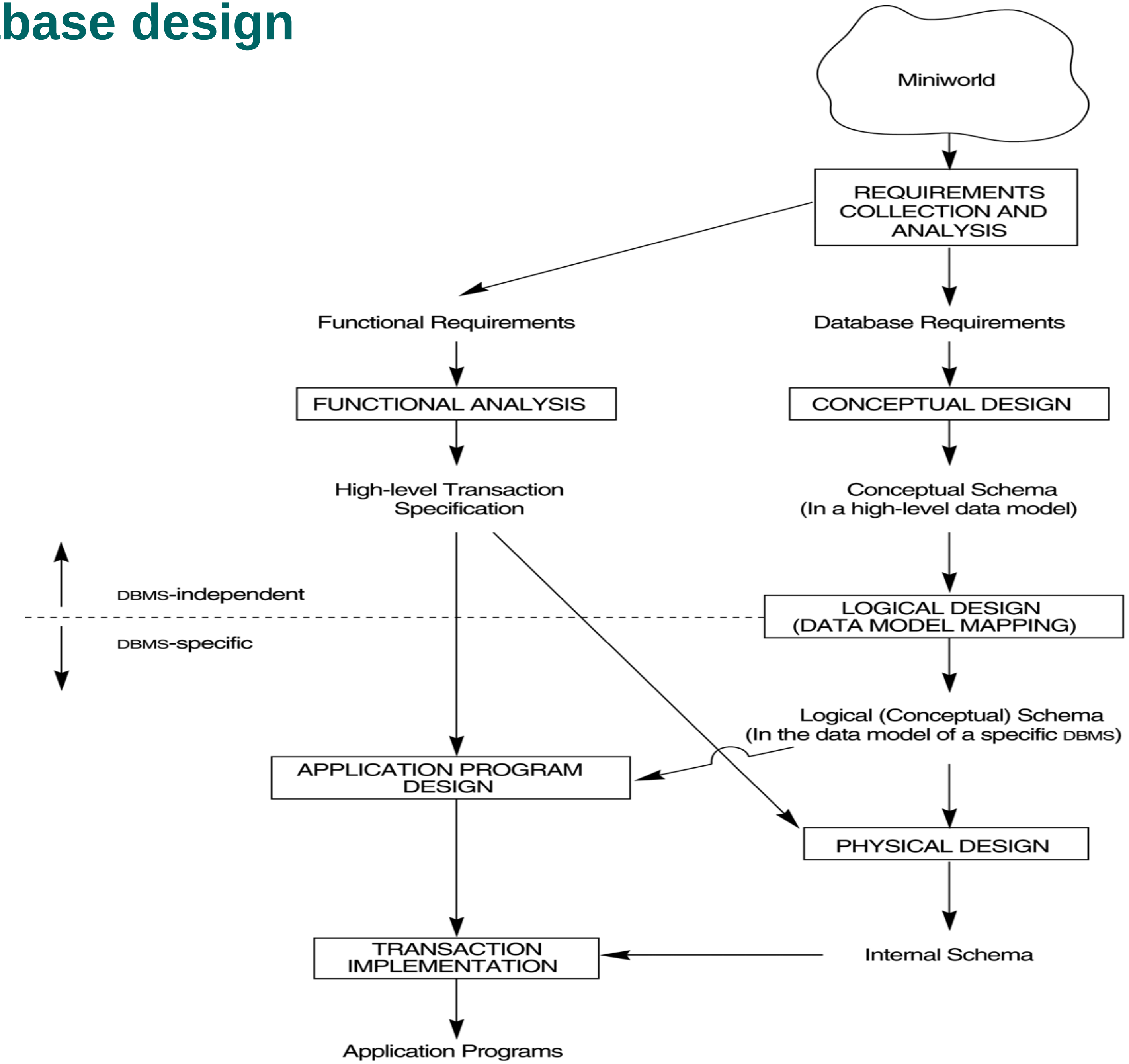
- Step 5: Determine candidate, primary, and alternate key attributes
  - To identify the candidate key(s) for each entity and if there is more than one candidate key, to choose one to be the primary key and the others as alternate keys
- Step 6: Consider use of enhanced modeling concepts (optional step)
  - To consider the use of enhanced modeling concepts, such as specialization/generalization, categories (union types)
- Step 7: Check model for redundancy
  - To check for the presence of any redundancy in the model and to remove any that does exist

Conceptual Database Design

Summarization

- Step 8: Validate conceptual model against user transactions
  - To ensure that the conceptual model supports the required transactions
- Step 9: Review conceptual data model with user
  - To review the conceptual data model with the user to ensure that the model is a 'true' representation of the data requirements of the enterprise

A simplified diagram to illustrate the main phases of database design



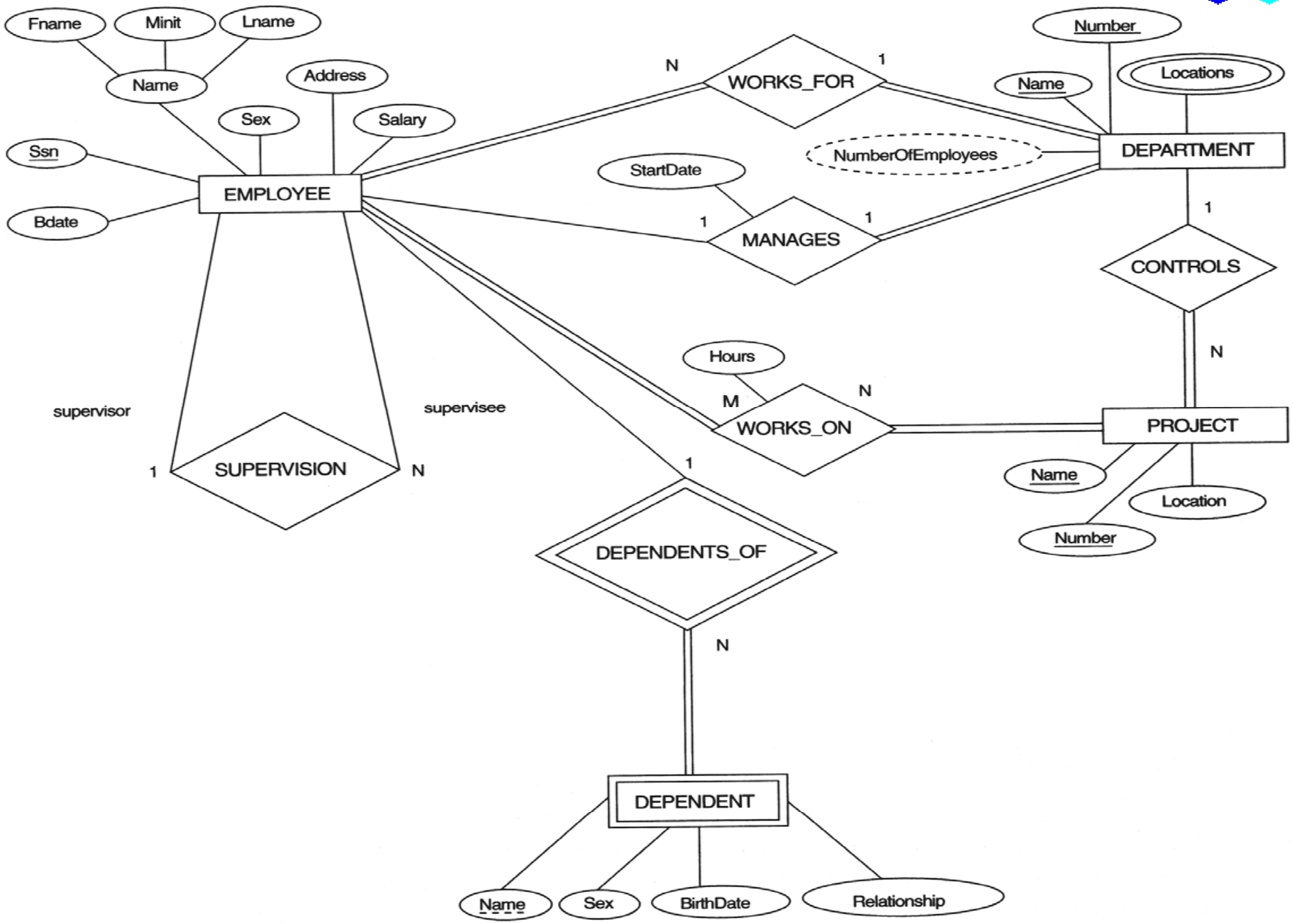
Logical Database Design

- To translate the conceptual data model into a logical data model and then to validate this model to check that it is structurally correct using normalization and supports the required transactions

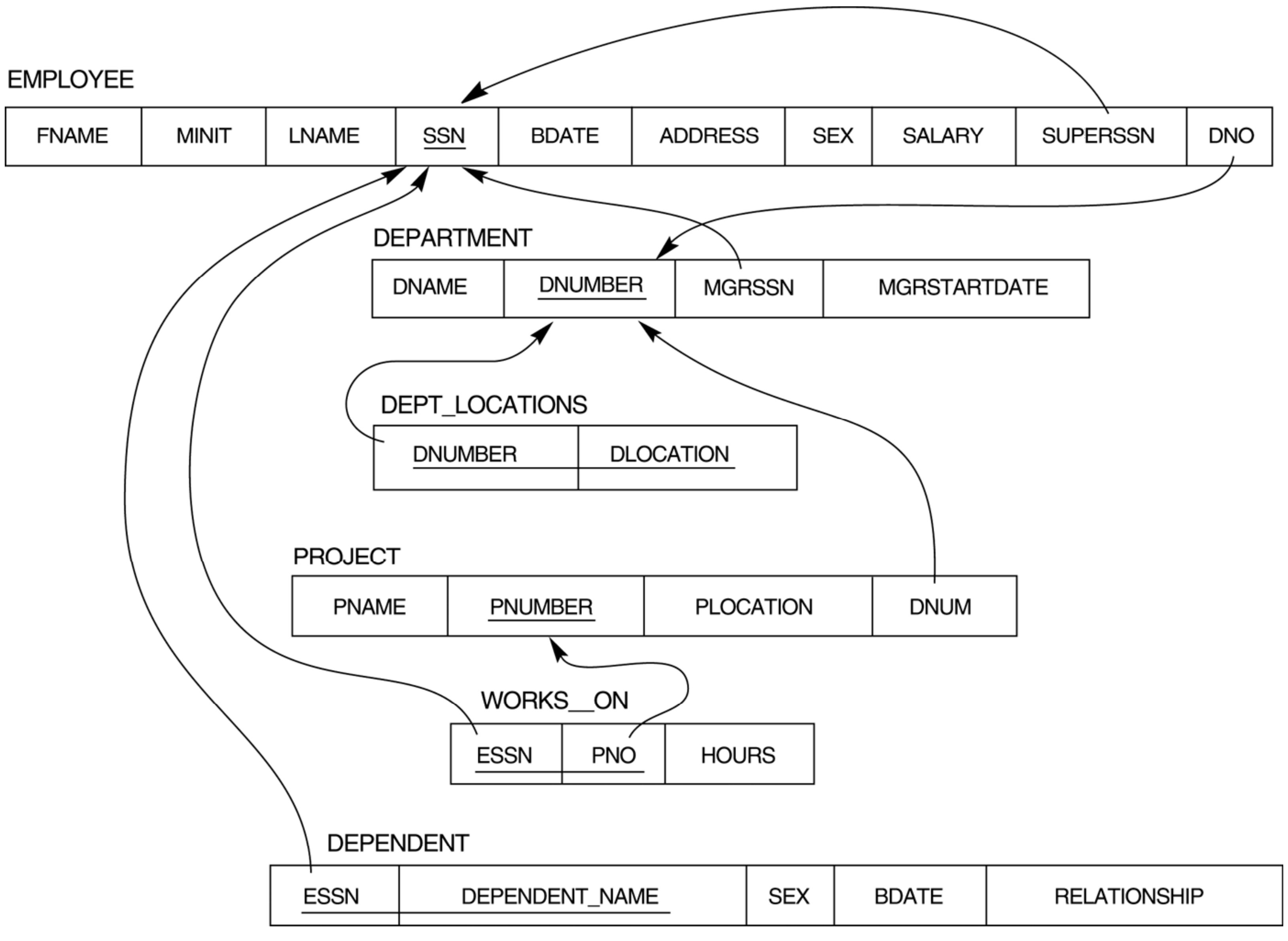
Logical Database Design

- Logical database design for the relational model
  - Step 1: Derive relations for logical data model
  - Step 2: Validate relations using normalization
  - Step 3: Validate relations against user transactions
  - Step 4: Define integrity constraints
  - Step 5: Review logical data model with user
  - Step 6: Merge logical data models into global model (optional step)
  - Step 7: Check for future growth
- ER- & EER-to-Relational Mapping

The ERD for the COMPANY database



Result of mapping the COMPANY ER schema into a relational schema



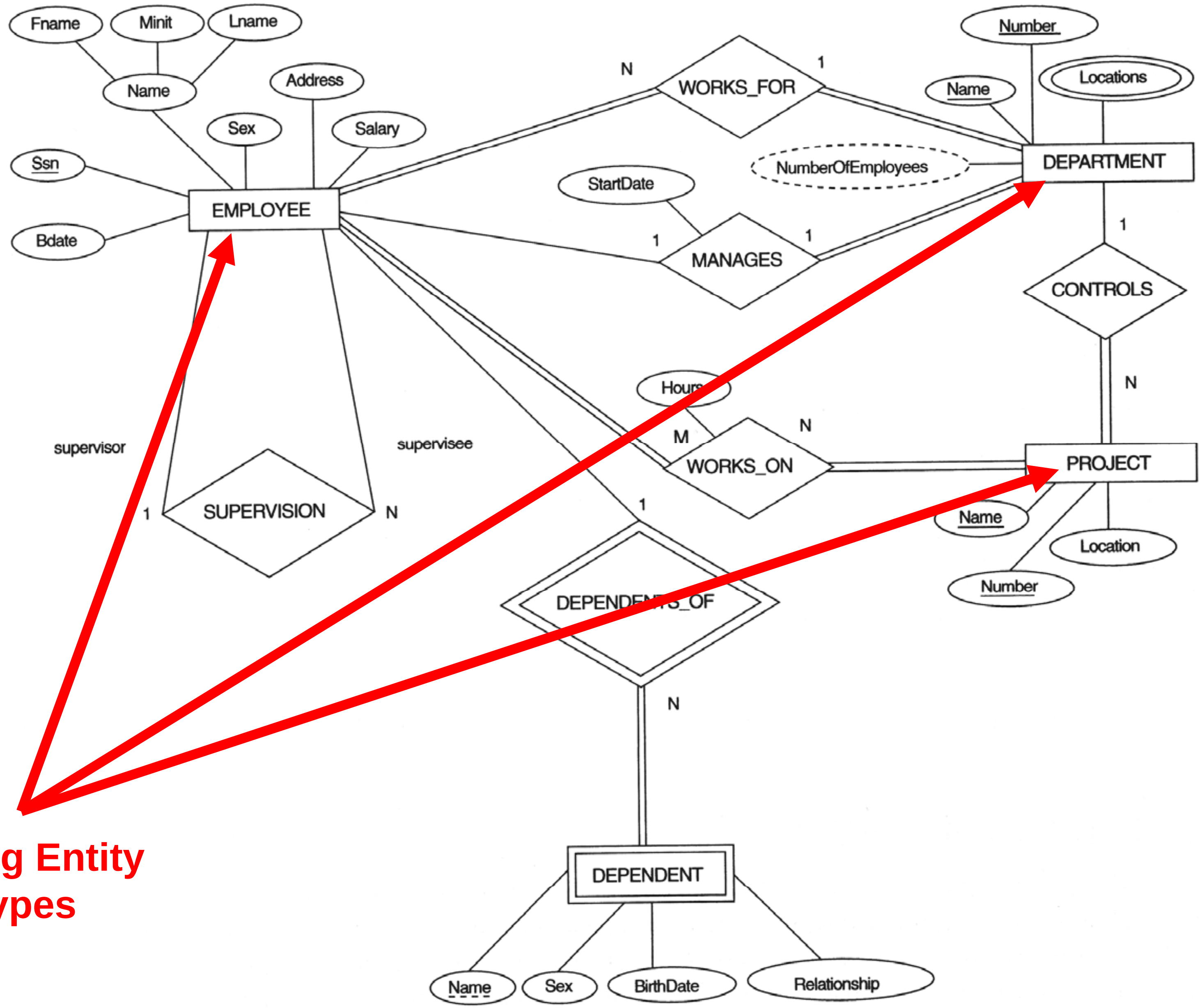
ER- & EER-to-Relational Mapping

- 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)

ER-to-Relational Mapping

- Step 1: Mapping of Regular (strong) Entity Types
  - Entity --> Relation
  - Attribute of entity --> Attribute of relation
  - Primary key of entity --> Primary key of relation
  - **Example:** We create the relations EMPLOYEE, DEPARTMENT, and PROJECT in the relational schema corresponding to the regular entities in the ER diagram. SSN, DNUMBER, and PNUMBER are the primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT as shown

The ERD for the COMPANY database



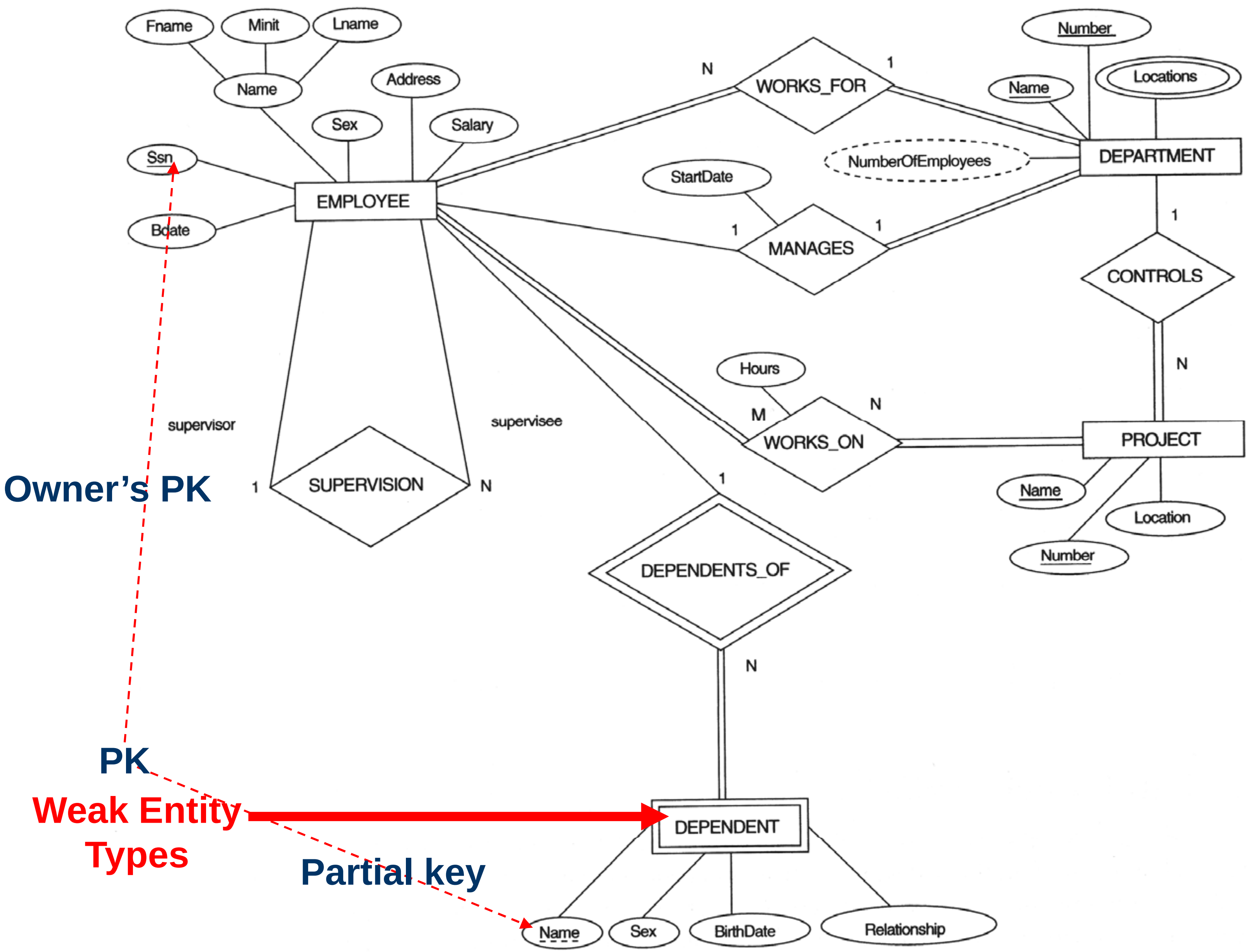
Strong Entity Types

ER-to-Relational Mapping

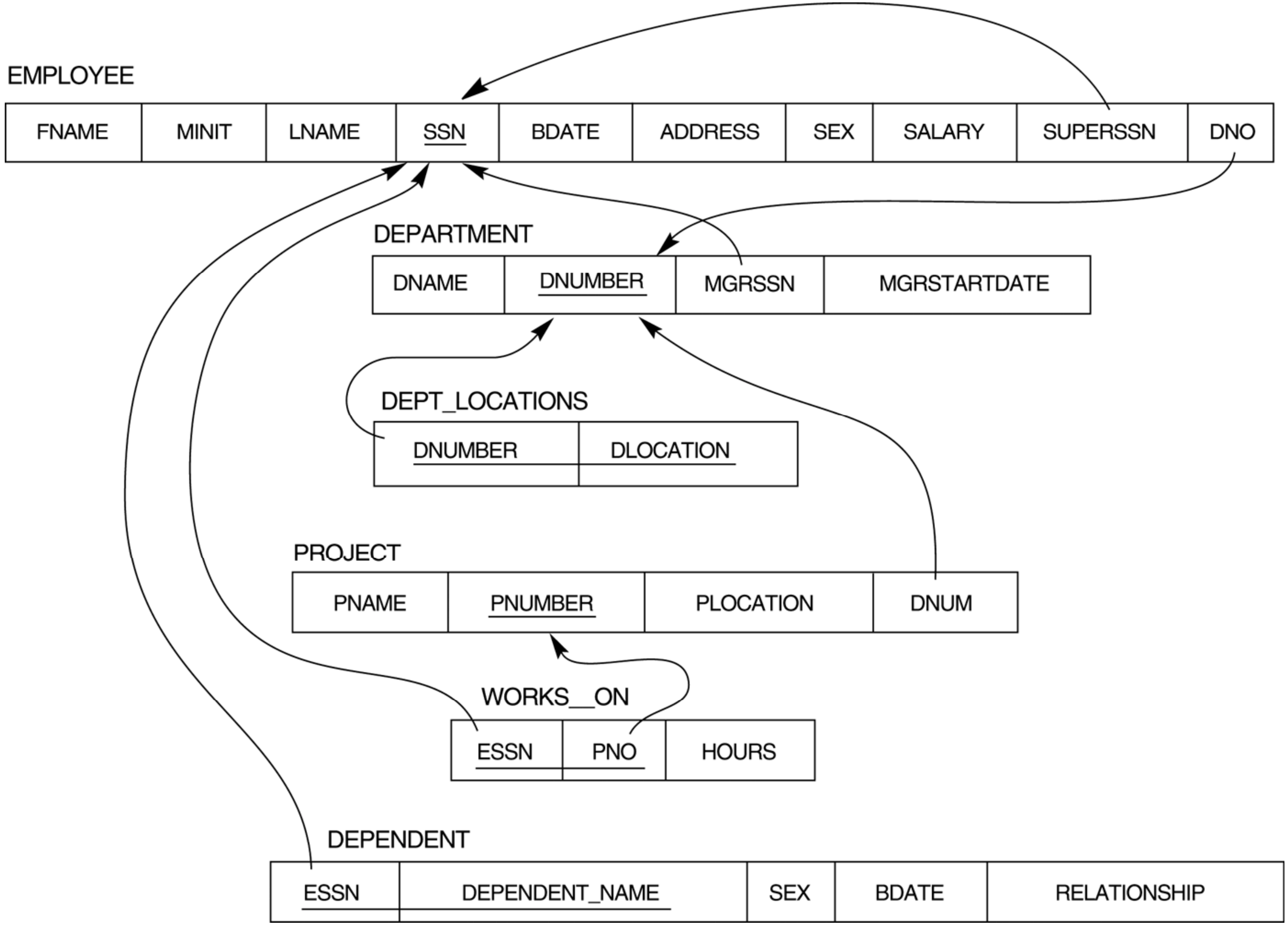
- Step 2: Mapping of Weak Entity Types
  - For each weak entity type W in the ER schema with owner entity type E, create a relation R and include all simple attributes (or simple components of composite attributes) of W as attributes of R
  - In addition, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s)
  - The primary key of R is the *combination of* the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any
  - **Example:** Create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT. Include the primary key SSN of the EMPLOYEE relation as a foreign key attribute of DEPENDENT (renamed to ESSN)

    The primary key of the DEPENDENT relation is the combination {ESSN, DEPENDENT_NAME} because DEPENDENT_NAME is the partial key of DEPENDENT
  - Note: CASCADE option as implemented

The ERD for the COMPANY database



Result of mapping the COMPANY ER schema into a relational schema



ER-to-Relational Mapping

- ER-
  -
  -
  - 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
- Transformation of binary relationships - depends on *functionality* of relationship and *membership class* of participating entity types

ER-to-Relational Mapping

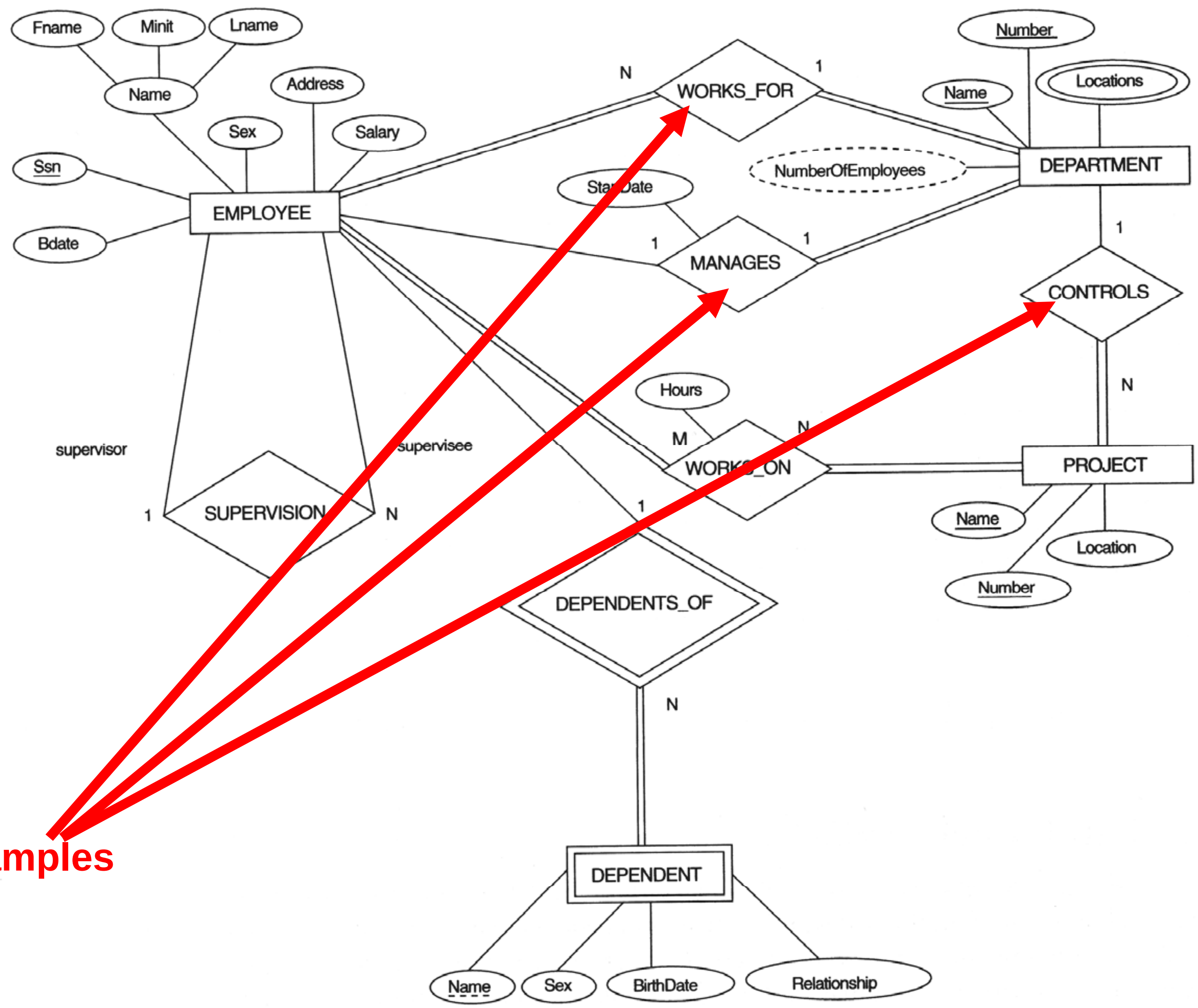
- **Mandatory** membership class
  - For two entity types E1 and E2: If E2 is a mandatory member of an N:1 (or 1:1) relationship with E1, then the relation for E2 will include the prime attributes of E1 as a foreign key to represent the relationship
  - For a 1:1 relationship: If the membership class for E1 and E2 are both mandatory, a foreign key can be used in either relation
  - For an N:1 relationship: If the membership class of E2, which is at the N-side of the relationship, is *optional* (i.e. partial), then the above guideline is not applicable

ER-to-Relational Mapping

- Assume every module must be offered by a department, then the entity type MODULE is a **mandatory** member of the relationship OFFER. The relation for MODULE is:

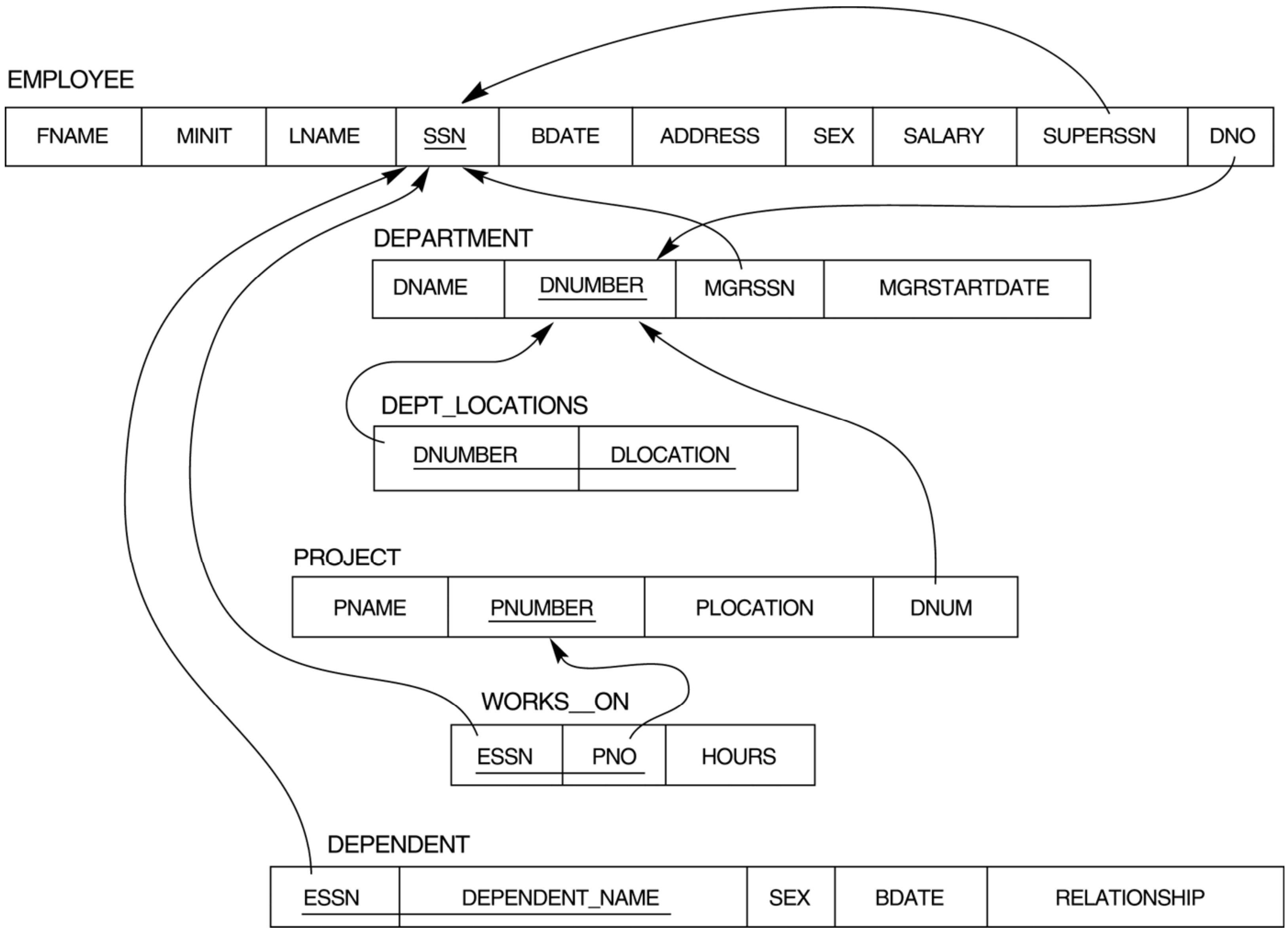
MODULE(<u>MDL-NUMBER</u>, TITLE, TERM, ..., **DNAME**)

The ERD for the COMPANY database



Examples

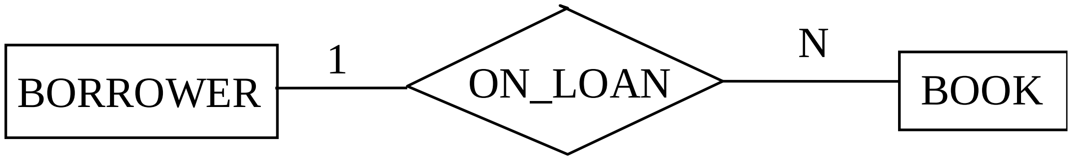
Result of mapping the COMPANY ER schema into a relational schema



ER-to-Relational Mapping

- **Optional** membership classes
  - If entity type E2 is an optional member of the N:1 relationship with entity type E1 (i.e. E2 is at the N-side of the relationship), then the relationship is **usually** represented by a new relation containing the prime attributes of E1 and E2, together with any attributes of the relationship. The key of the entity type at the N-side (i.e. E2) will become the key of the new relation
  - If both entity types in a 1:1 relationship have the optional membership, a new relation is created which contains the prime attributes of both entity types, together with any attributes of the relationship. The prime attribute(s) of either entity type will be the key of the new relation

ER-to-Relational Mapping



- One possible representation of the relationship:

  BORROWER(<u>BNUMBER</u>, NAME, ADDRESS, ...)

  BOOK(<u>ISBN</u>, TITLE, ..., **BNUMBER**)

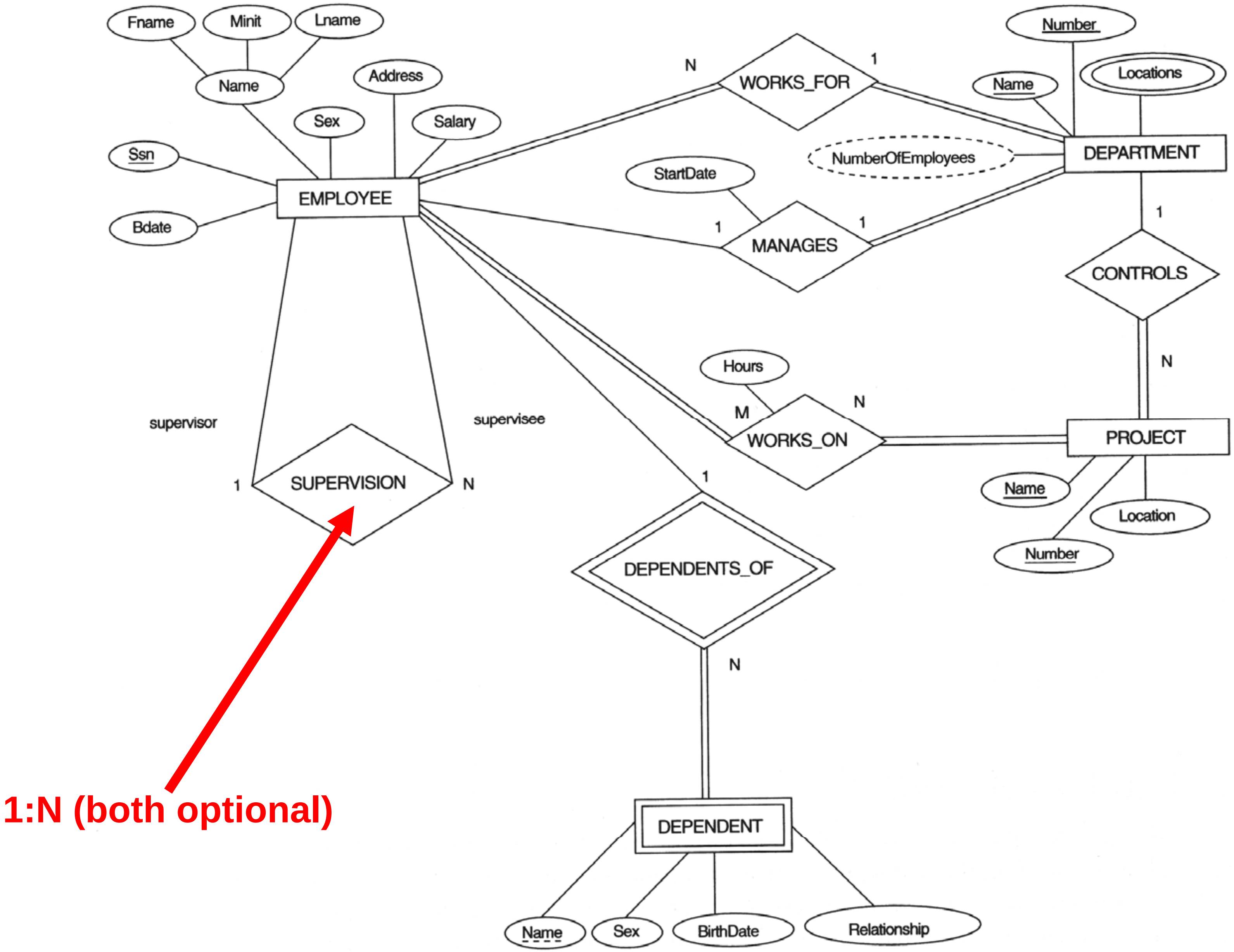
- A better alternative:

  BORROWER(<u>BNUMBER</u>, NAME, ADDRESS, ...)

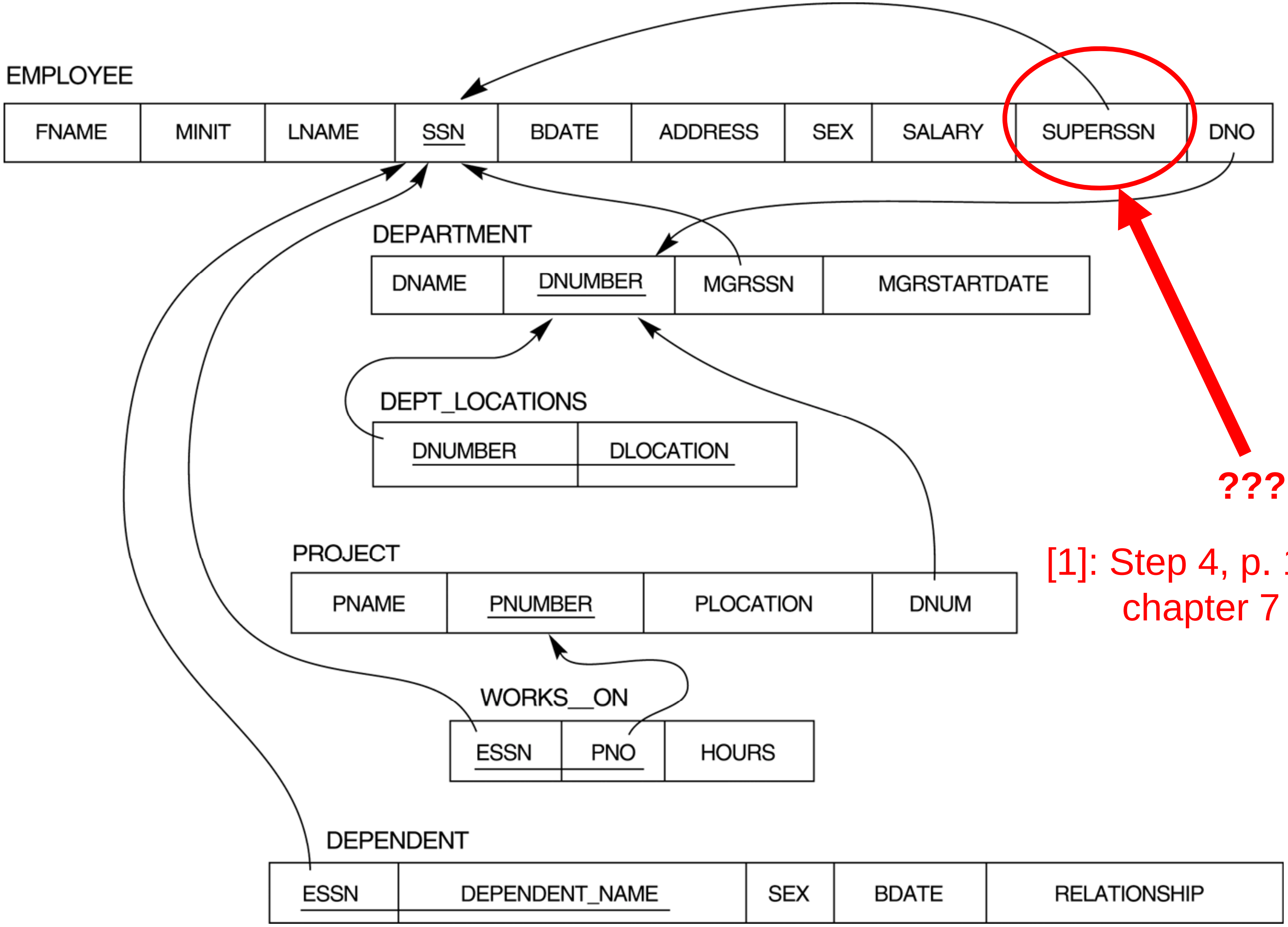
  BOOK(<u>ISBN</u>, TITLE, ...)

  ON_LOAN(<u>ISBN</u>, BNUMBER)

The ERD for the COMPANY database



Result of mapping the COMPANY ER schema into a relational schema



[1]: Step 4, p. 195, chapter 7

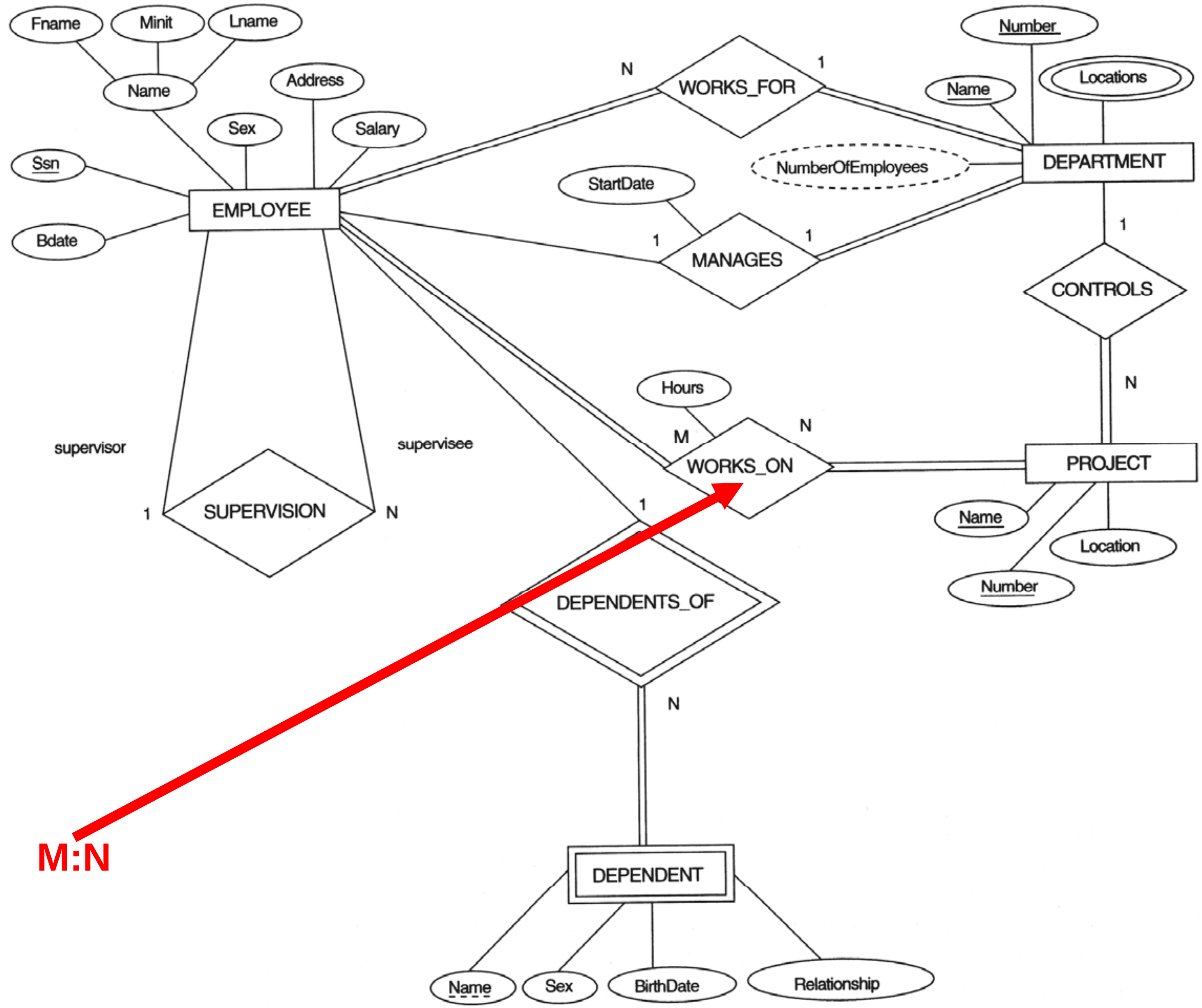
ER-to-Relational Mapping

- N:M binary relationships:
  - An N:M relationship is always represented by a new relation which consists of the prime attributes of both participating entity types together with any attributes of the relationship
  - The combination of the prime attributes will form the primary key of the new relation

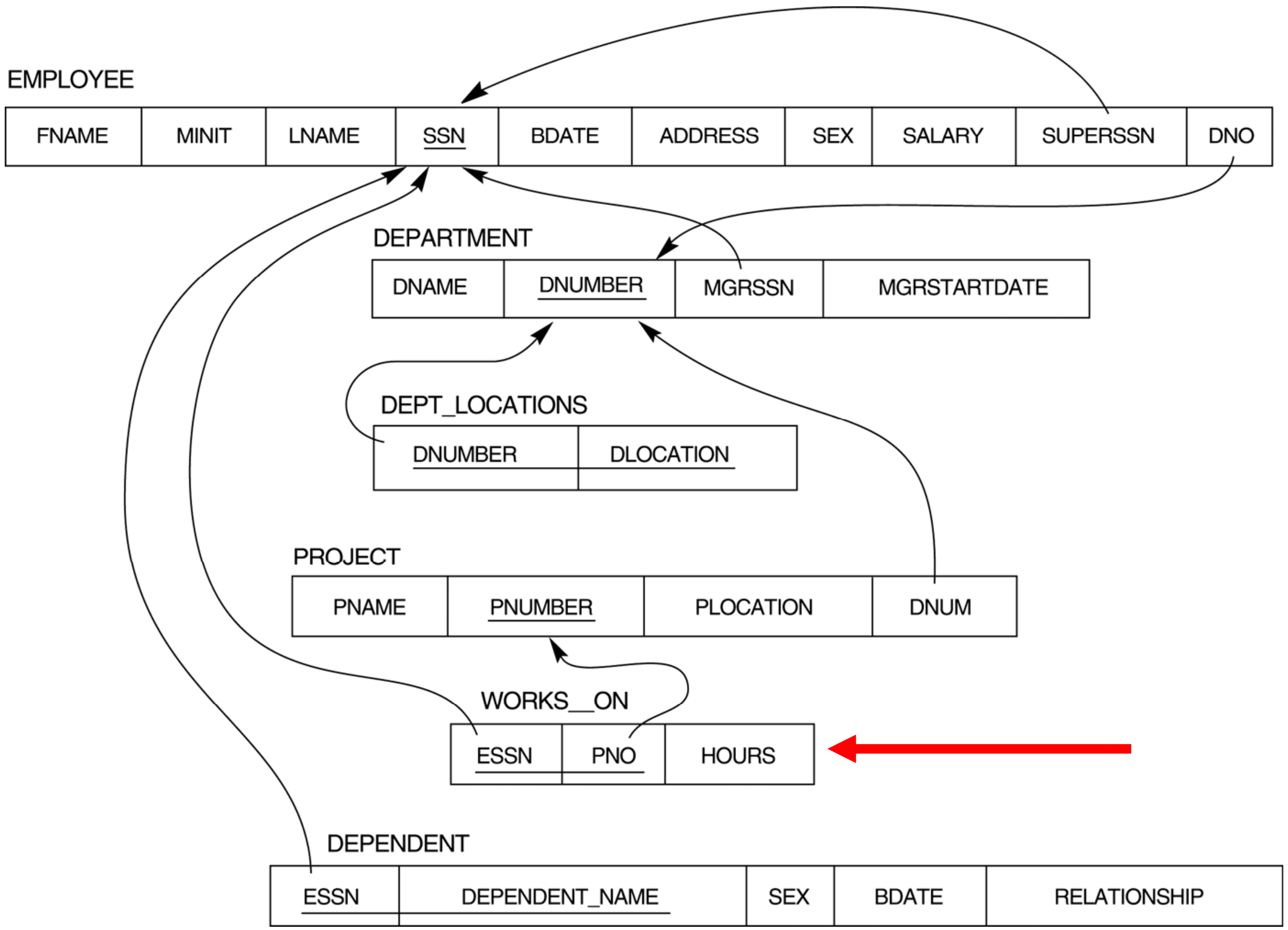
- **Example:** ENROL is an M:N relationship between STUDENT and MODULE. To represent the relationship, we have a new relation:

  ENROL(<u>SNUMBER, MDL-NUMBER</u>, DATE)

The ERD for the COMPANY database



Result of mapping the COMPANY ER schema into a relational schema

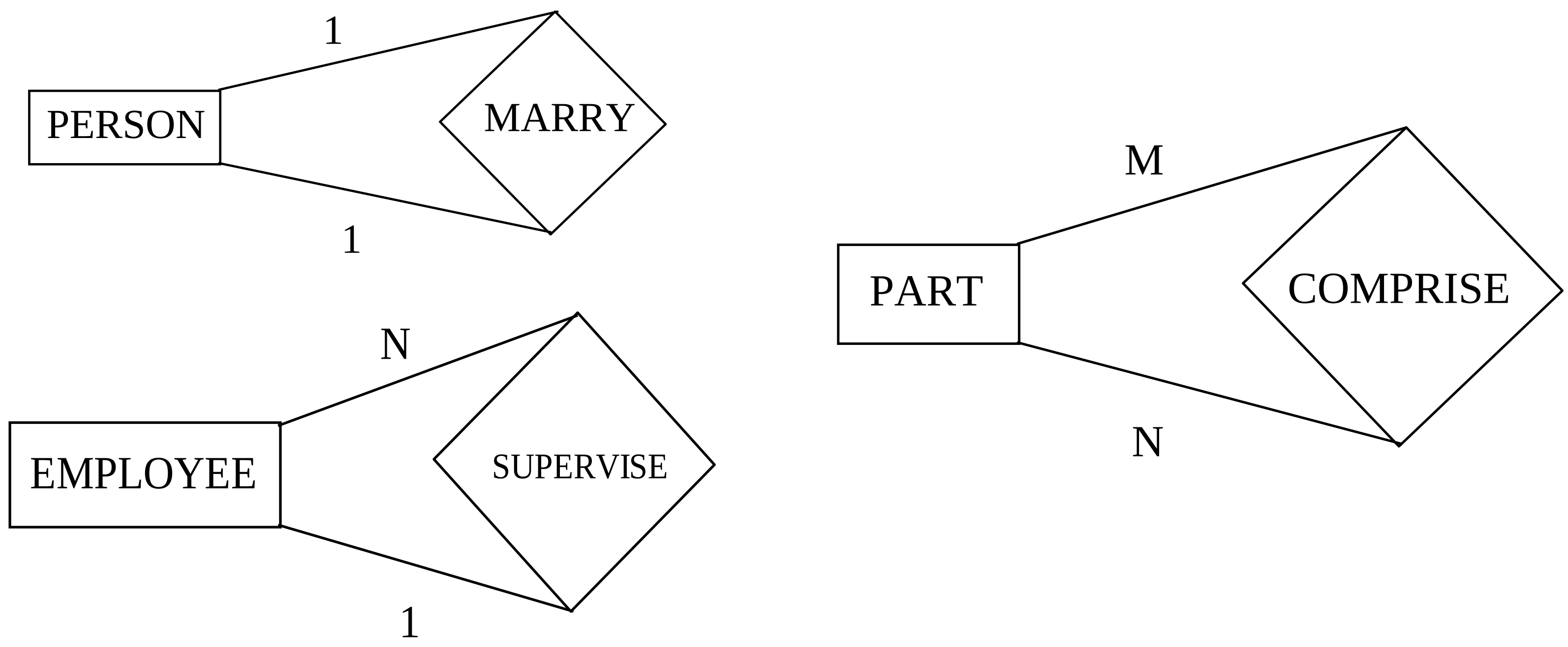


ER-to-Relational Mapping

- ER-
  -
  -
  -
  -
  -
  - Step 6: Mapping of Multivalued attributes
  - Step 7: Mapping of N-ary Relationship Types

ER-to-Relational Mapping

- Transformation of recursive/involuted relationships
  - Relationship among different instances of the same entity
  - The name(s) of the prime attribute(s) needs to be changed to reflect the role each entity plays in the relationship



ER-to-Relational Mapping

- **Example 1:** 1:1 involuted relationship, in which the memberships for both entities are optional

PERSON(<u>ID</u>, NAME, ADDRESS, ...)

MARRY(<u>HUSBAND-ID</u>, WIFE_ID, DATE_OF_MARRIAGE)

ER-to-Relational Mapping

- **Example 2:** 1:M involuted relationship.
  - If the relationship is mandatory or almost mandatory:
    EMPLOYEE(<u>ID</u>, ENAME, ..., **SUPERVISOR_ID**)
  - If the relationship is optional:
    EMPLOYEE(<u>ID</u>, ENAME, ...)
    SUPERVISE(<u>ID</u>, START_DATE, ..., **SUPERVISOR_ID**)

- **Example 3:** N:M involuted relationship

  PART(<u>PNUMBER</u>, DESCRIPTION, ...)
  COMPRISE(<u>MAJOR-PNUMBER, MINOR-PNUMBER</u>, QUANTITY)

ER- & EER-to-Relational Mapping

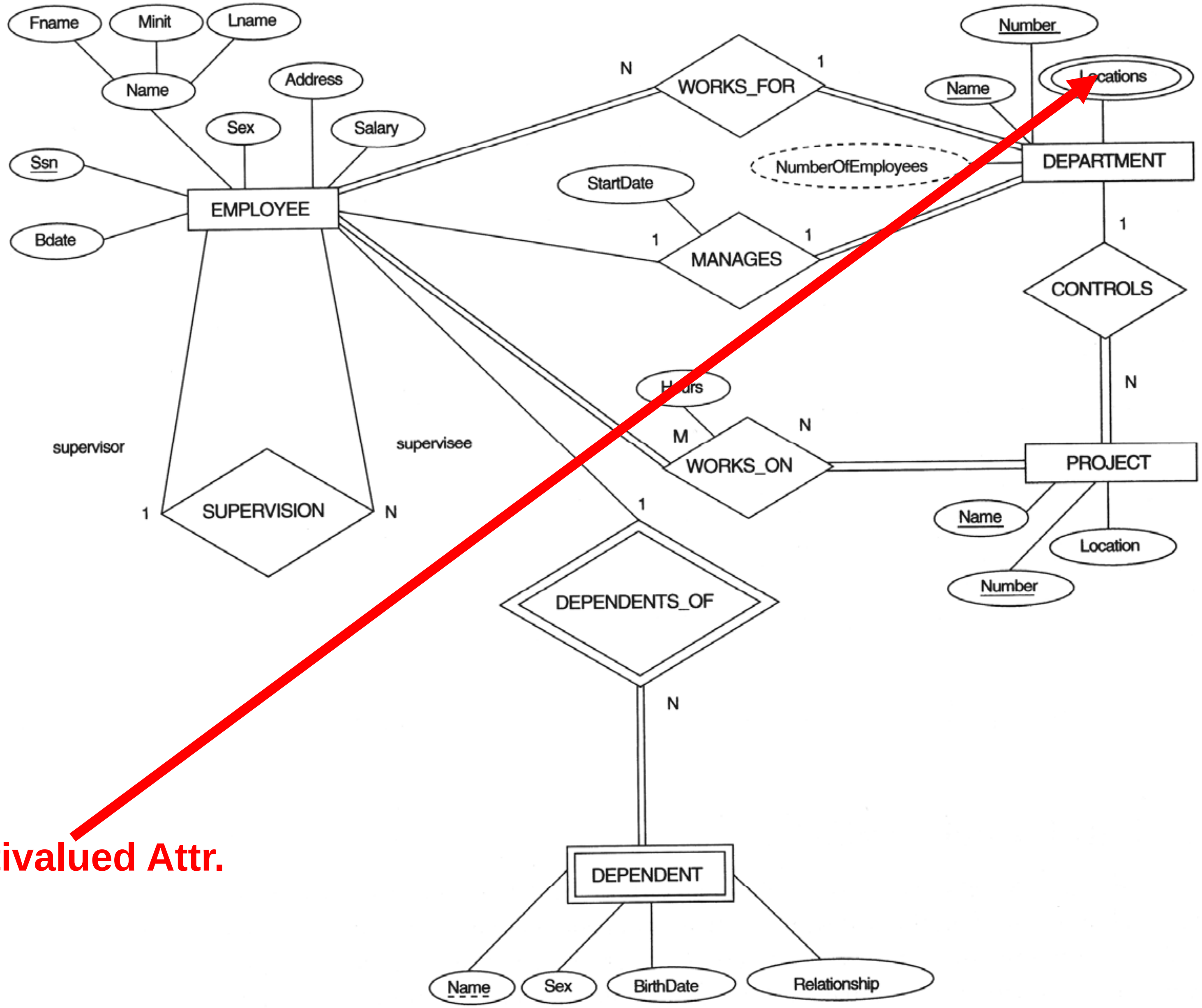
- ER-
  -
  -
  -
  -
  -
  - Step 6: Mapping of Multivalued attributes
  - Step 7: Mapping of N-ary Relationship Types
- EER-
  - Step 8: Options for Mapping Specialization or Generalization.
  - Step 9: Mapping of Union Types (Categories)

ER-to-Relational Mapping

- Step 6: Mapping of Multivalued attributes
  - For each multivalued attribute A, create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type or relationship type that has A as an attribute
  - The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components

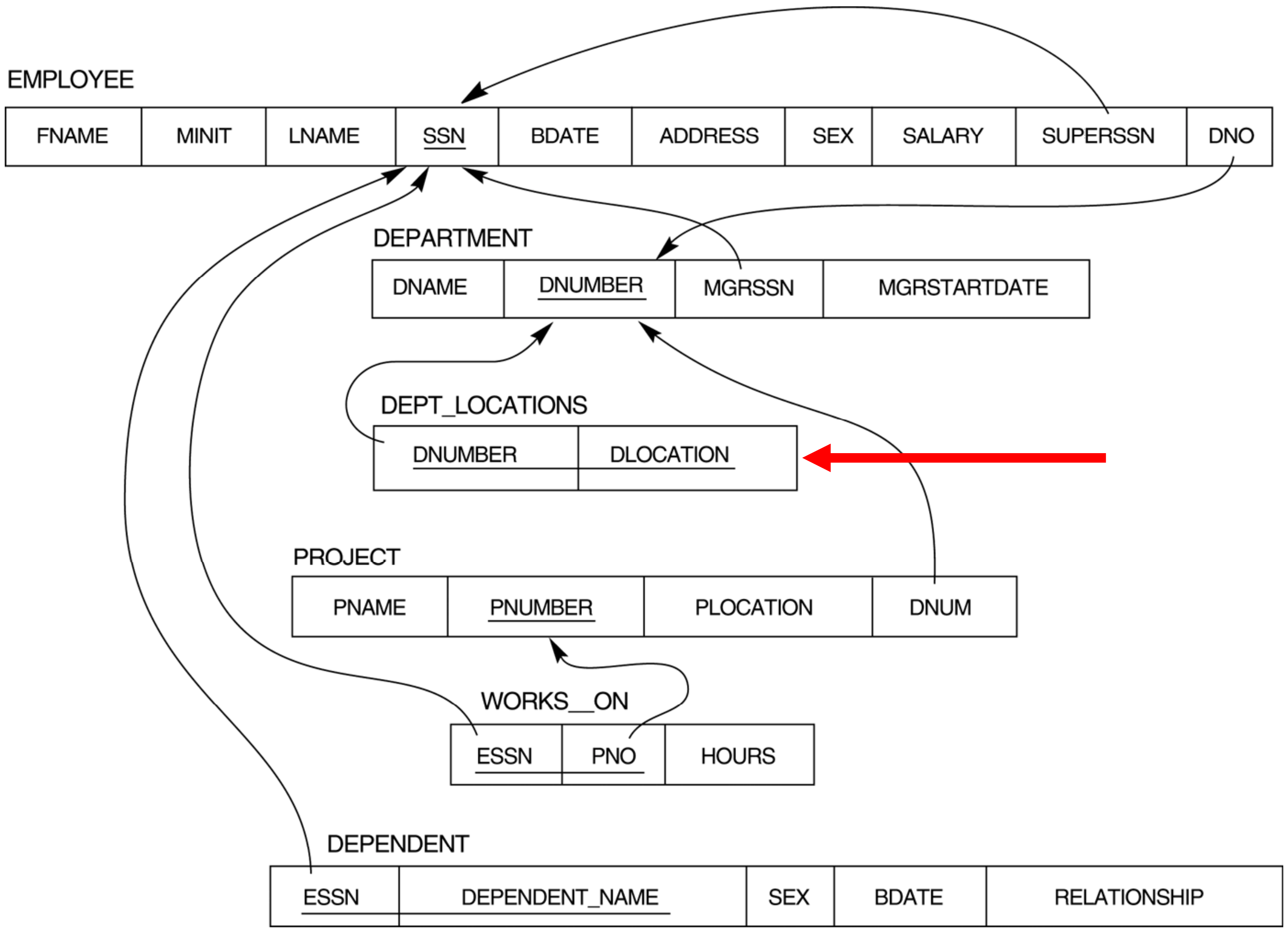
    Example: The relation DEPT_LOCATIONS is created. The attribute DLOCATION represents the multivalued attribute LOCATIONS of DEPARTMENT, while DNUMBER-as foreign key-represents the primary key of the DEPARTMENT relation. The primary key of R is the combination of {DNUMBER, DLOCATION}

The ERD for the COMPANY database



Multivalued Attr.

Result of mapping the COMPANY ER schema into a relational schema



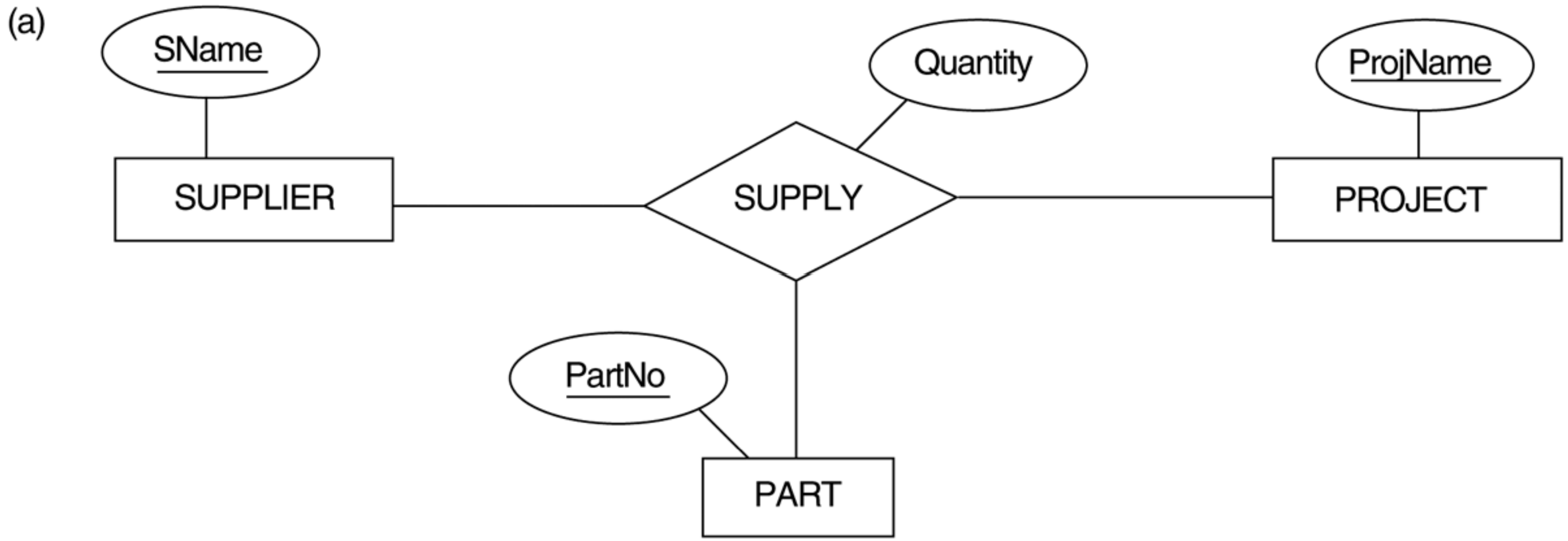
ER-to-Relational Mapping

- Step 7: Mapping of N-ary Relationship Types
  - For each n-ary relationship type R, where n>2, create a new relationship S to represent R
  - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types
  - Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S

    Example: The relationship type SUPPY in the ER below. This can be mapped to the relation SUPPLY shown in the relational schema, whose primary key is the combination of the three foreign keys {SNAME, PARTNO, PROJNAME}

ER-to-Relational Mapping

FIGURE 4.11 Ternary relationship types
(a) The SUPPLY relationship



SUPPLIER

SNAME	. . .

PROJECT

PROJNAME	. . .

PART

PARTNO	. . .

SUPPLY

SNAME	PROJNAME	PARTNO	QUANTITY

Note: if the cardinality constraint on any of the entity types E participating in the relationship is 1, the PK should not include the FK attributes that reference the relation E' corresponding to E (see section 4.7 [1])

ER-to-Relational Mapping
Summary of Mapping Constructs & Constraints

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

ER- & EER-to-Relational Mapping

- ER-
  -
  -
  -
  -
  -
  -
  -
- EER-
  - Step 8: Options for Mapping Specialization or Generalization.
  - Step 9: Mapping of Union Types (Categories)

EER-to-Relational Mapping

- **Step8: Options for Mapping Specialization or Generalization.**

Convert each specialization with m subclasses $\{S_1, S_2, \ldots, S_m\}$ and generalized superclass C, where the attributes of C are $\{k, a_1, \ldots a_n\}$ and k is the (primary) key, into relational schemas using one of the four following options:

Option 8A: Multiple relations-Superclass and subclasses.

Create a relation L for C with attributes Attrs(L) = $\{k, a_1, \ldots a_n\}$ and PK(L) = k. Create a relation L_i for each subclass S_i, 1 <= i <= m, with the attributesAttrs(L_i) = {k} U {attributes of S_i} and PK(L_i)=k. This option works **for any specialization** (total or partial, disjoint of over-lapping).

Option 8B: Multiple relations-Subclass relations only

Create a relation L_i for each subclass S_i, 1 <= i <= m, with the attributes Attr(L_i) = {attributes of S_i} U $\{k, a_1, \ldots, a_n\}$ and PK(L_i) = k. This option only works for a specialization whose subclasses are **total** (every entity in the superclass must belong to (at least) one of the subclasses)

EER-to-Relational Mapping

Option 8C: Single relation with one type attribute

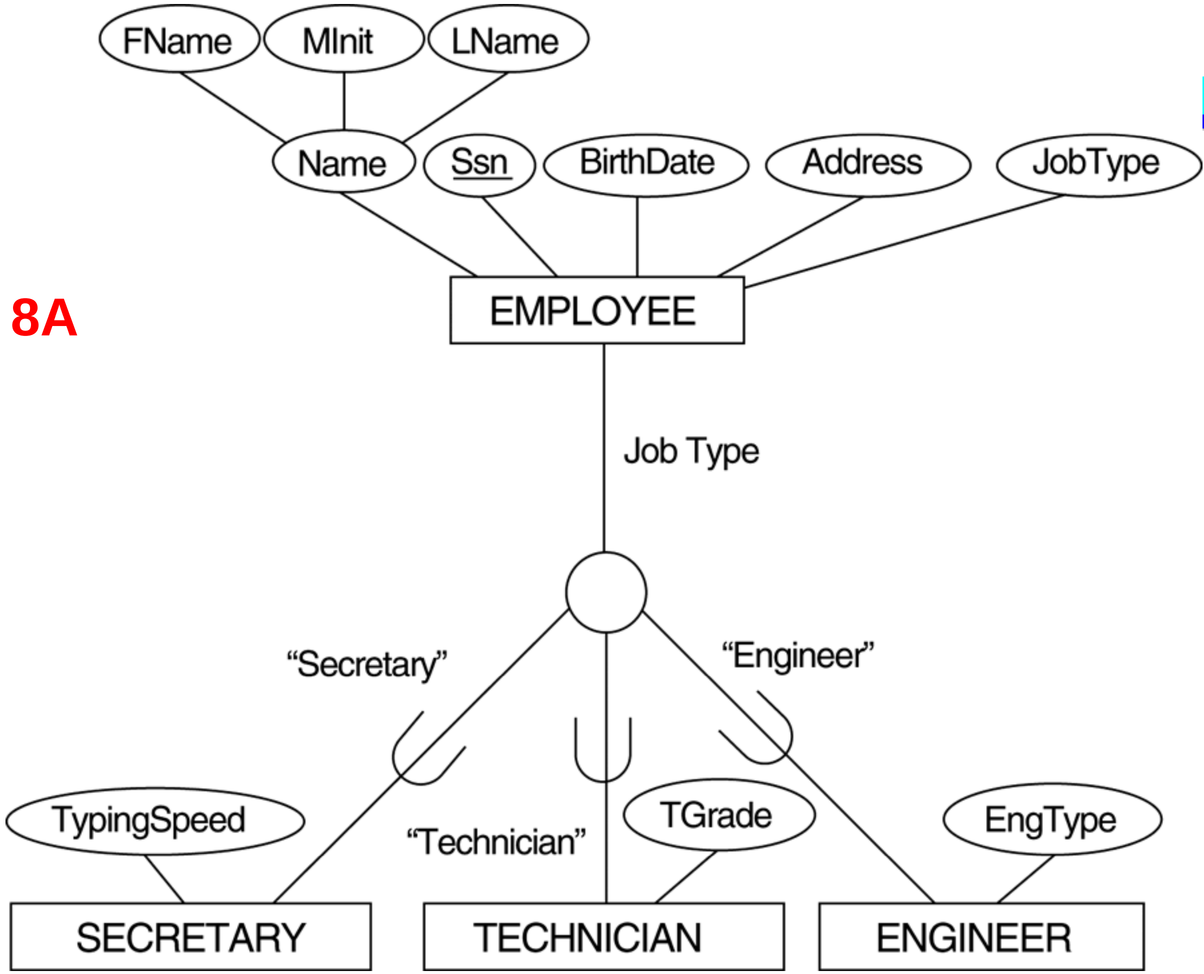
Create a single relation L with attributes Attrs(L) = {k,a_1,…a_n} U {attributes of S_1} U…U {attributes of S_m} U {t} and PK(L) = k. The attribute t is called a type (or **discriminating**) attribute that indicates the subclass to which each tuple belongs

Option 8D: Single relation with multiple type attributes

Create a single relation schema L with attributes Attrs(L) = {k,a_1,…a_n} U {attributes of S_1} U…U {attributes of S_m} U {t_1, t_2,…,t_m} and PK(L) = k. Each t_i, 1 <= i <= m, is a Boolean type attribute indicating whether a tuple belongs to the subclass S_i

Option 8A is preferred !!

Example: Option 8A



(a) EMPLOYEE

SSN	FName	MInit	LName	BirthDate	Address	JobType

SECRETARY

SSN	TypingSpeed

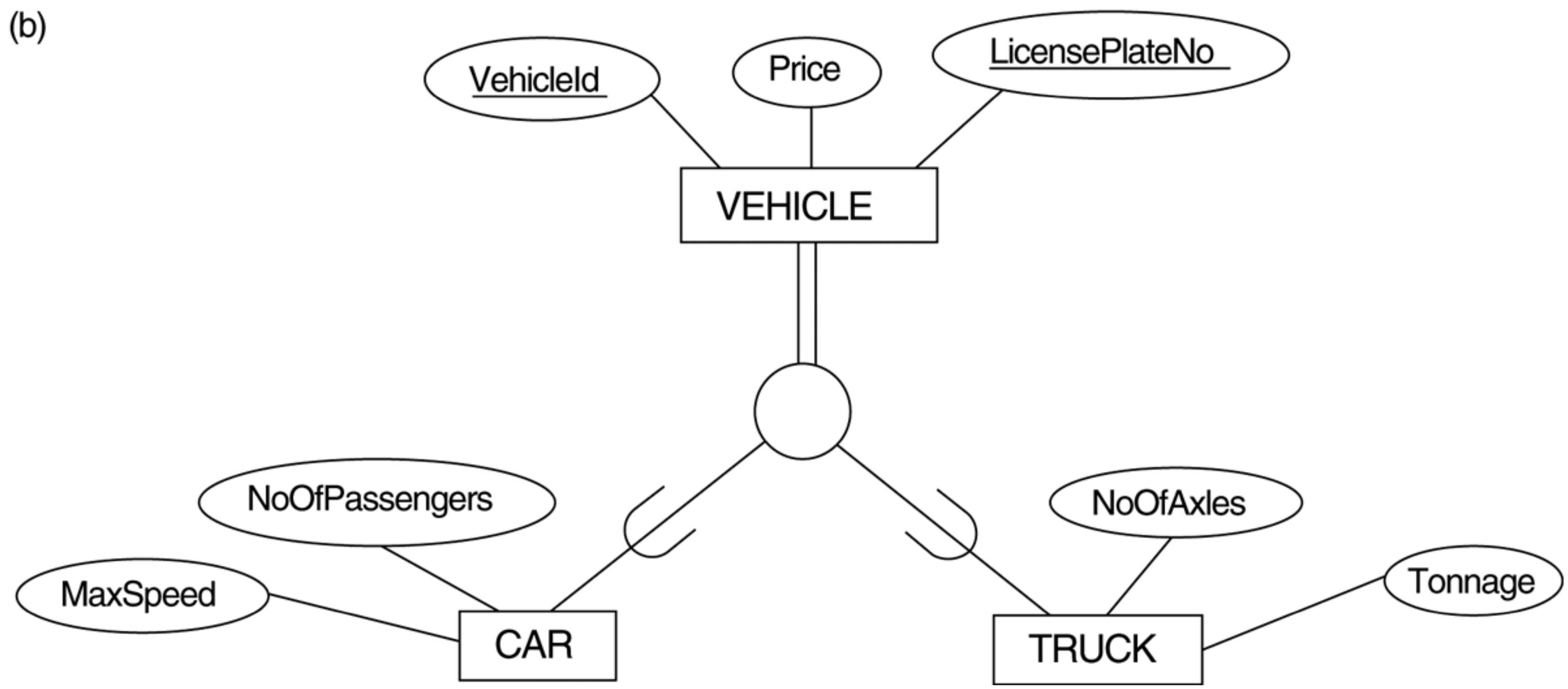
TECHNICIAN

SSN	TGrade

ENGINEER

SSN	EngType

Example: Option 8B



(b)

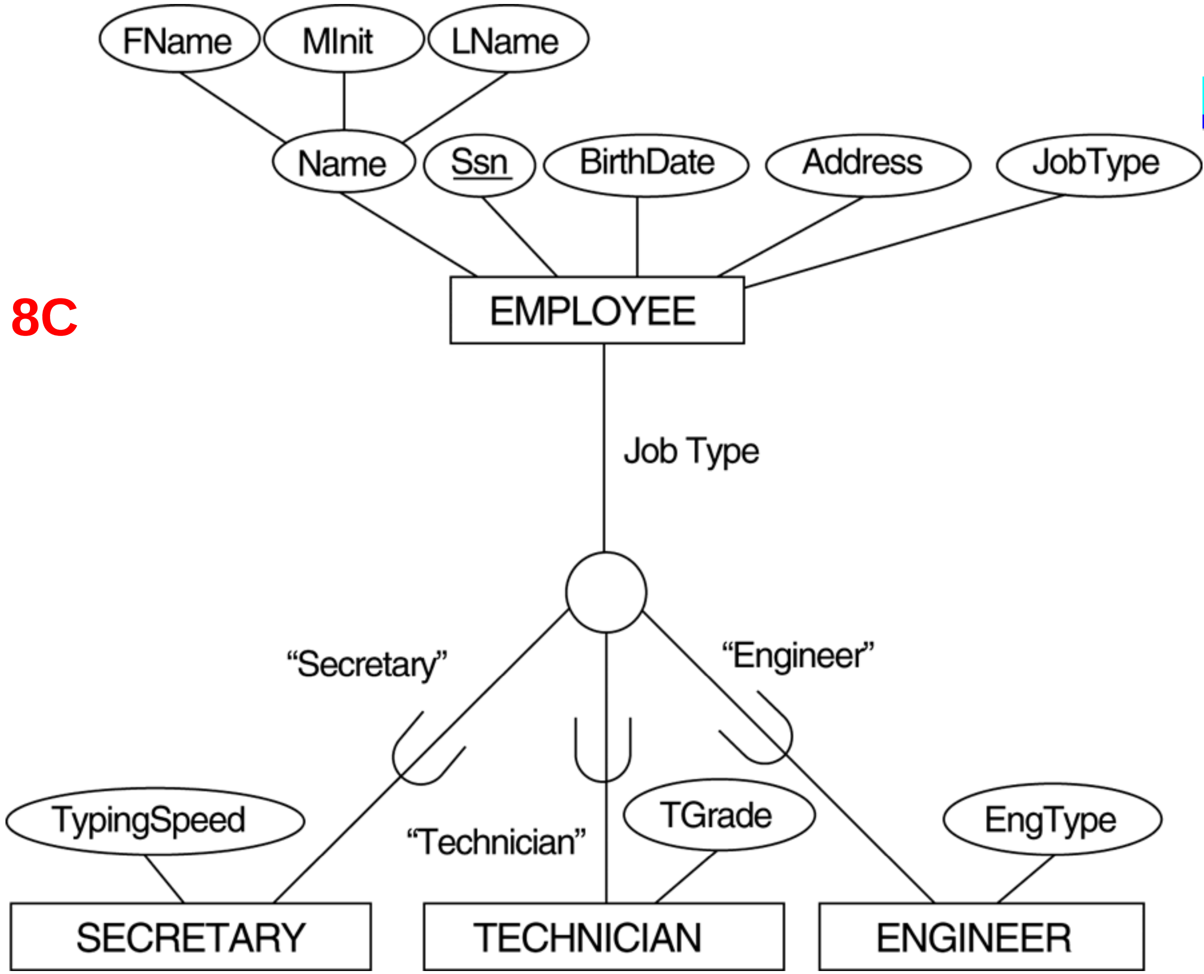
CAR

<u>VehicleId</u>	LicensePlateNo	Price	MaxSpeed	NoOfPassengers

TRUCK

<u>VehicleId</u>	LicensePlateNo	Price	NoOfAxles	**Tonnage**

Example: Option 8C

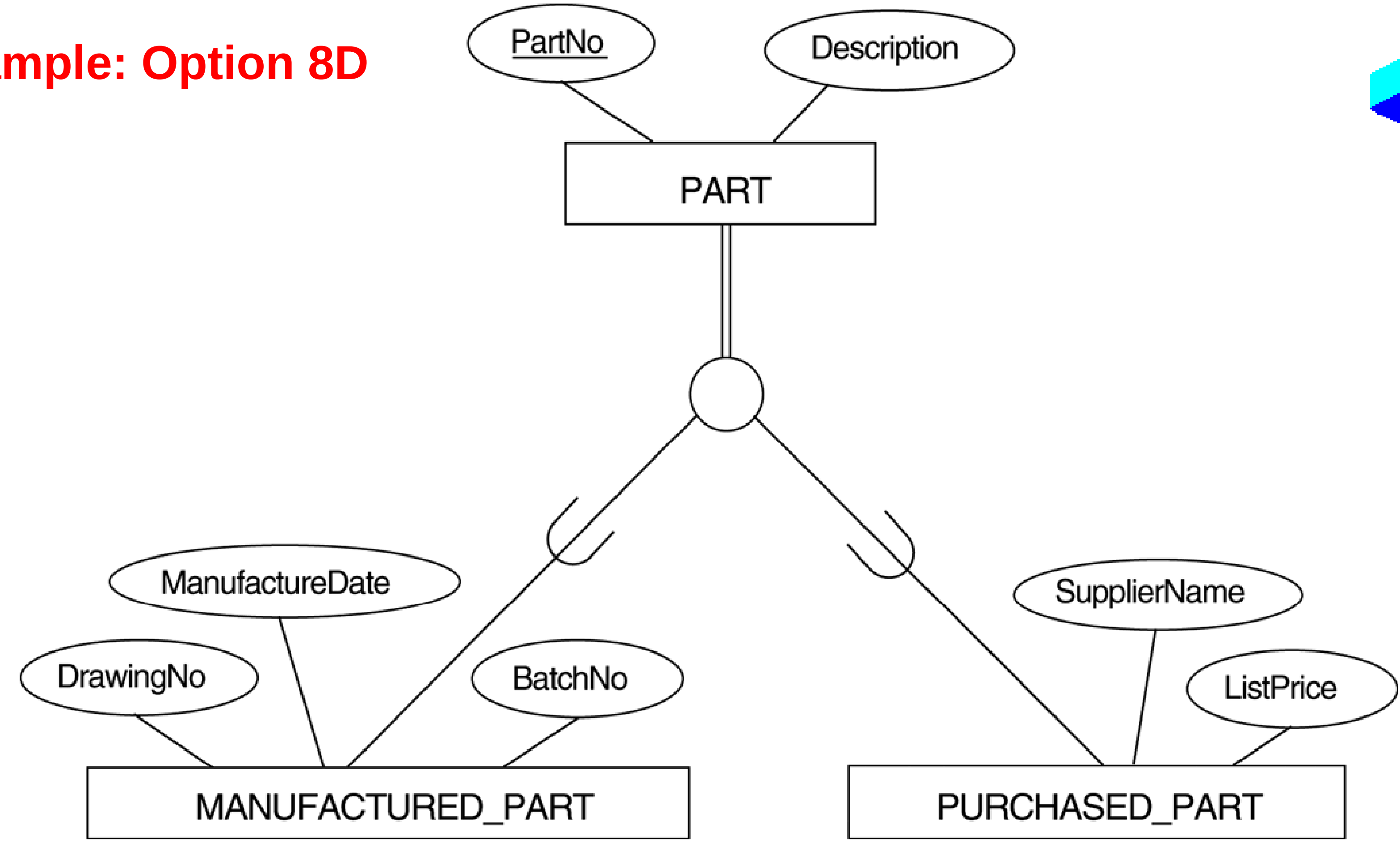


(c) EMPLOYEE

SSN	FName	MInit	LName	BirthDate	Address	JobType	TypingSpeed	TGrade	**EngType**

Serving as the type attribute

Example: Option 8D



(d) PART

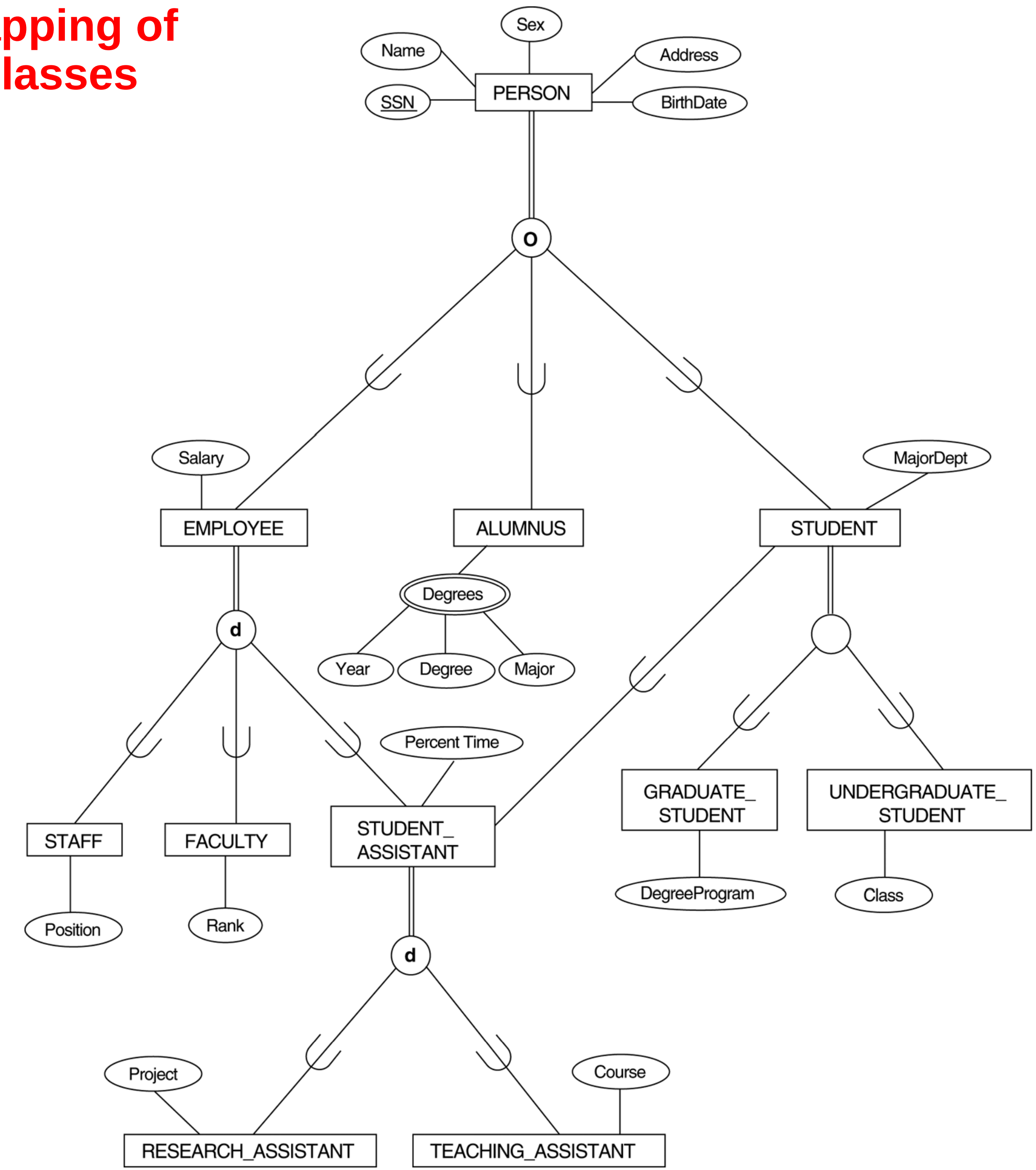
PartNo	Description	MFlag	DrawingNo	ManufactureDate	BatchNo	PFlag	SupplierName	ListPrice

Boolean type attributes

EER-to-Relational Mapping

- **Mapping of Shared Subclasses (Multiple Inheritance)**
  - A shared subclass, such as STUDENT_ASSISTANT, is a subclass of several classes, indicating multiple inheritance. These classes must all have the same key attribute; otherwise, the shared subclass would be modeled as a category.
  - We can apply any of the options discussed in Step 8 to a shared subclass, subject to the restriction discussed in Step 8 of the mapping algorithm. Below both 8C and 8D are used for the shared class STUDENT_ASSISTANT

Example: Mapping of Shared Subclasses



Example: Mapping of Shared Subclasses

PERSON

<u>SSN</u>	Name	BirthDate	Sex	Address

EMPLOYEE

<u>SSN</u>	Salary	EmployeeType	Position	Rank	PercentTime	RAFlag	TAFlag	Project	**Course**

ALUMNUS

<u>SSN</u>

ALUMNUS_DEGREES

<u>SSN</u>	<u>Year</u>	<u>Degree</u>	<u>**Major**</u>

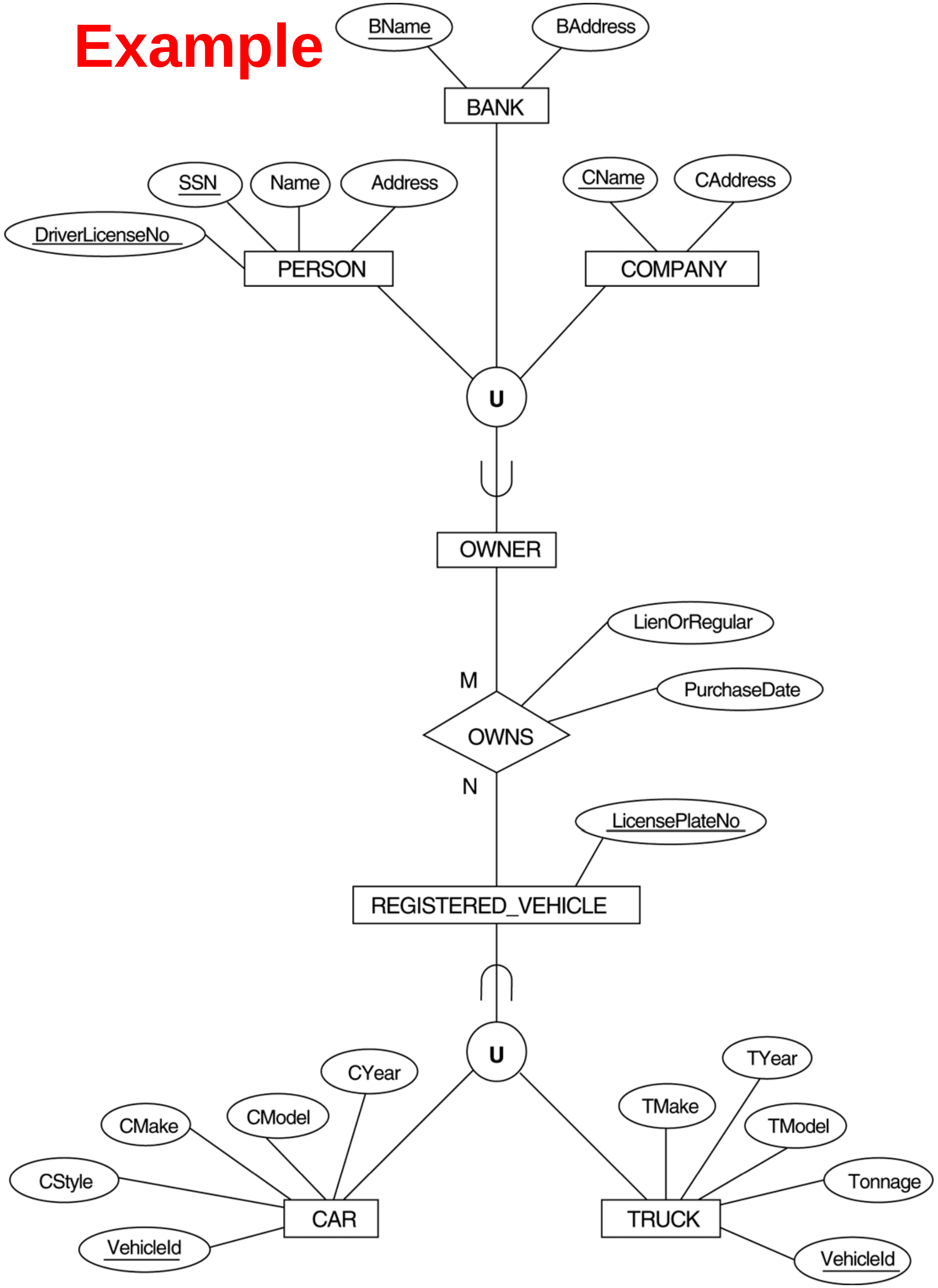
STUDENT

<u>SSN</u>	MajorDept	GradFlag	UndergradFlag	DegreeProgram	Class	StudAssistFlag

EER-to-Relational Mapping

- **Step 9: Mapping of Union Types (Categories).**
  - For mapping a category whose defining superclasses have different keys, it is customary to specify a new key attribute, called a **surrogate key**, when creating a relation to correspond to the category.
  - In the example below we can create a relation OWNER to correspond to the OWNER category and include any attributes of the category in this relation. The primary key of the OWNER relation is the surrogate key, which we called OwnerId
  - We also include the surrogate key attribute OwnerId as FK in each relation corresponding to a superclass of the category in order to specify the correspondence in values between the surrogate key and the PK of each superclass

Example



PERSON

SSN	DriverLicenseNo	Name	Address	**OwnerId**

BANK

BName	BAddress	OwnerId

COMPANY

CName	CAddress	OwnerId

OWNER

OwnerId

REGISTERED_VEHICLE

VehicleId	LicensePlateNumber

CAR

VehicleId	CStyle	CMake	CModel	**CYear**

TRUCK

VehicleId	TMake	TModel	Tonnage	TYear

OWNS

OwnerId	VehicleId	PurchaseDate	LienOrRegular

Functional Dependencies & Normalization

- 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
- Notes and suggestions

Introduction

- Each *relation schema* consists of a number of attributes and the *relational database schema* consists of a number of relation schemas
- Attributes are grouped to form a relation schema
- Need some formal measure of why one grouping of attributes into a relation schema may be better than another

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

Introduction

- Redundant information in tuples: the attribute values pertaining to a particular department (DNUMBER, DNAME, DMGRSSN) are repeated for *every employee who works for that department*

EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
Smith,John B.	123456789	1965-01-09	731 Fondren,Houston,TX	5	Research	333445555
Wong,Franklin T.	333445555	1955-12-08	638 Voss,Houston,TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle,Spring,TX	4	Administration	987654321
Wallace,Jennifer S.	987654321	1941-06-20	291 Berry,Bellaire,TX	4	Administration	987654321
Narayan,Ramesh K.	666884444	1962-09-15	975 FireOak,Humble,TX	5	Research	333445555
English,Joyce A.	453453453	1972-07-31	5631 Rice,Houston,TX	5	Research	333445555
Jabbar,Ahmad V.	987987987	1969-03-29	980 Dallas,Houston,TX	4	Administration	987654321
Borg,James E.	888665555	1937-11-10	450 Stone,Houston,TX	1	Headquarters	888665555

Introduction

- Update anomalies: modification, deletion, insertion
  - Modification
    - As the manager of a dept. changes we have to update many values according to employees working for that dept.
    - Easy to make the DB **inconsistent**

EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
Smith,John B.	123456789	1965-01-09	731 Fondren,Houston,TX	5	Research	333445555
Wong,Franklin T.	333445555	1955-12-08	638 Voss,Houston,TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle,Spring,TX	4	Administration	987654321
Wallace,Jennifer S.	987654321	1941-06-20	291 Berry,Bellaire,TX	4	Administration	987654321
Narayan,Ramesh K.	666884444	1962-09-15	975 FireOak,Humble,TX	5	Research	333445555
English,Joyce A.	453453453	1972-07-31	5631 Rice,Houston,TX	5	Research	333445555
Jabbar,Ahmad V.	987987987	1969-03-29	980 Dallas,Houston,TX	4	Administration	987654321
Borg,James E.	888665555	1937-11-10	450 Stone,Houston,TX	1	Headquarters	888665555

Introduction

- Deletion: if Borg James E. leaves, we delete his tuple and lose the existing of dept. 1, the name of dept. 1, and who is the manager of dept. 1

EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
Smith,John B.	123456789	1965-01-09	731 Fondren,Houston,TX	5	Research	333445555
Wong,Franklin T.	333445555	1955-12-08	638 Voss,Houston,TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle,Spring,TX	4	Administration	987654321
Wallace,Jennifer S.	987654321	1941-06-20	291 Berry,Bellaire,TX	4	Administration	987654321
Narayan,Ramesh K.	666884444	1962-09-15	975 FireOak,Humble,TX	5	Research	333445555
English,Joyce A.	453453453	1972-07-31	5631 Rice,Houston,TX	5	Research	333445555
Jabbar,Ahmad V.	987987987	1969-03-29	980 Dallas,Houston,TX	4	Administration	987654321
Borg,James E.	888665555	1937-11-10	450 Stone,Houston,TX	1	Headquarters	888665555

Introduction

- Insertion:
  - How can we create a department before any employees are assigned to it ??

EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
Smith,John B.	123456789	1965-01-09	731 Fondren,Houston,TX	5	Research	333445555
Wong,Franklin T.	333445555	1955-12-08	638 Voss,Houston,TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle,Spring,TX	4	Administration	987654321
Wallace,Jennifer S.	987654321	1941-06-20	291 Berry,Bellaire,TX	4	Administration	987654321
Narayan,Ramesh K.	666884444	1962-09-15	975 FireOak,Humble,TX	5	Research	333445555
English,Joyce A.	453453453	1972-07-31	5631 Rice,Houston,TX	5	Research	333445555
Jabbar,Ahmad V.	987987987	1969-03-29	980 Dallas,Houston,TX	4	Administration	987654321
Borg,James E.	888665555	1937-11-10	450 Stone,Houston,TX	1	Headquarters	888665555

Introduction

- Reducing the NULL values in tuples
  - Employees not assigned to any dept.: waste the storage space
  - Other difficulties: aggregation operations (e.g., COUNT) and joins

Introduction

- Disallowing the possibility of generating spurious tuples

  EMP_PROJ(<u>SSN, PNUMBER</u>, HOURS, ENAME, PNAME, PLOCATION)

  EMP_LOCS(<u>ENAME, PLOCATION</u>)
  EMP_PROJ1(<u>SSN, PNUMBER</u>, HOURS, PNAME, PLOCATION)

- Generation of invalid and spurious data during JOINS: PLOCATION is the attribute that relates EMP_LOCS and EMP_PROJ1, and PLOCATION is neither a primary key nor a foreign key in either EMP_LOCS or EMP_PROJ1 (*<u>cf. chapter 10 [1] for more details</u>*)

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

☞ Normalization

- It helps DB designers determine the best relation schemas
  - A formal framework for analyzing relation schemas based on their keys and on the functional dependencies among their attributes
  - A series of normal form tests that can be carried out on individual relation schemas so that the relational database can be normalized to any desired degree
- It is based on the concept of normal form 1NF, 2NF, 3NF, BCNF, 4NF, 5 NF
- It is a process which ensures that the data is structured in such a way that attributes are grouped with the PK. Attributes that do not directly depend on PK may be extracted to form a new relation

Introduction

- There are two important properties of decompositions:
  (a) non-additive or losslessness of the corresponding join
  (b) preservation of the functional dependencies
- Note that property (a) is extremely important and *cannot* be sacrificed. Property (b) is less stringent and may be sacrificed (see chapter 11)

Functional Dependencies (FDs)

- Definition of FD
- Direct, indirect, partial dependencies
- Inference Rules for FDs
- Equivalence of Sets of FDs
- Minimal Sets of FDs

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
- FDs are **constraints** that are derived from the *meaning* and *interrelationships* of the data attributes
- A set of attributes X *functionally determines* a set of attributes Y if the value of X determines a unique value for Y

Functional Dependencies (FDs)

- X -> Y holds if whenever two tuples have the same value for X, they *must have* the same value for Y
- For any two tuples t1 and t2 in any relation instance r(R): *If* t1[X]=t2[X], *then* t1[Y]=t2[Y]
- X -> Y in R specifies a *constraint* on all relation instances r(R)
- 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)

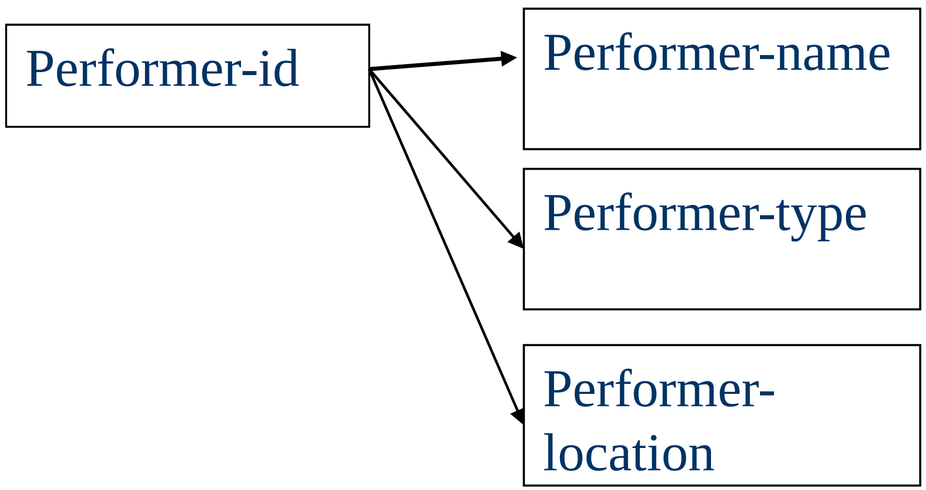
- If K is a key of R, then K functionally determines all attributes in R (since we never have two distinct tuples with t1[K]=t2[K])

Functional Dependencies (FDs)

-
- Direct, indirect, partial dependencies
- Inference Rules for FDs
- Equivalence of Sets of FDs
- Minimal Sets of FDs

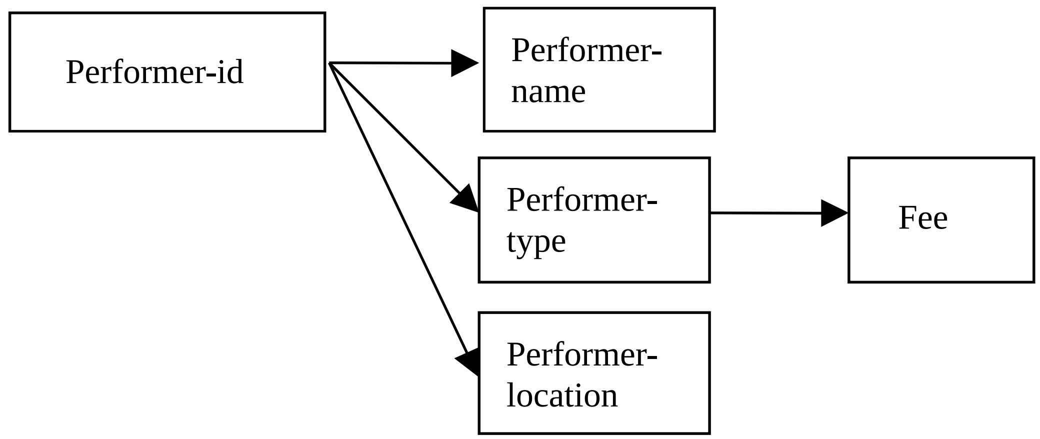
Functional Dependencies (FDs)

- 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(<u>SSN, PNUMBER</u>, HOURS, ENAME, PNAME, PLOCATION)

    {SSN, PNUMBER} -> HOURS

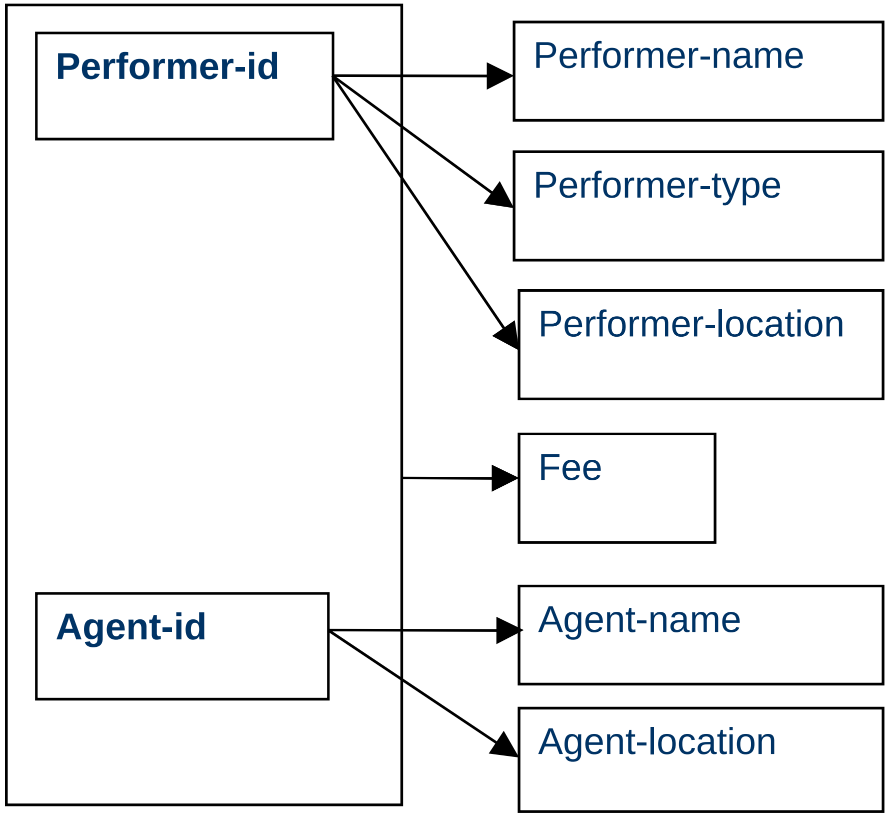
  - **Partial dependency** - if the value of an attribute does not depend on an entire composite determinant, but only part of it, the relationship is known as the partial dependency

    SSN -> ENAME

    PNUMBER -> {PNAME, PLOCATION}

Functional Dependencies (FDs)

- Partial dependency



Functional Dependencies (FDs)

-
-
- Inference Rules for FDs
- Equivalence of Sets of FDs
- Minimal Sets of FDs

Functional Dependencies (FDs)

- Given a set of FDs F, we can *infer* additional FDs that hold whenever the FDs in F hold

<u>Armstrong's inference rules:</u>

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

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

(Notation: XZ stands for X U Z)

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

Functional Dependencies (FDs)

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

- The last three inference rules, as well as any other inference rules, can be deduced from IR1, IR2, and IR3 (completeness property)

Functional Dependencies (FDs)

- **Closure** of a set F of FDs is the set F^+ of all FDs that can be inferred from F
- **Closure** of a set of attributes X with respect to F is the set X^+ of all attributes that are functionally determined by X
- X^+ can be calculated by repeatedly applying IR1, IR2, IR3 using the FDs in F

Functional Dependencies (FDs)

-
-
-
- Equivalence of Sets of FDs
- Minimal Sets of FDs

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

Functional Dependencies (FDs)

- Every set of FDs has an equivalent minimal set
- There can be several equivalent minimal sets
- There is no simple algorithm for computing a minimal set of FDs that is equivalent to a set F of FDs
- To synthesize a set of relations, we assume that we start with a set of dependencies that is a minimal set (e.g., see algorithms 11.2 and 11.4)

Outline

- Normalization
  - 1NF and dependency problems
  - 2NF – solves partial dependency
  - 3NF – solves indirect dependency
  - BCNF – well-normalized relations
- Notes and suggestions
- Summary
- Reading suggestion:
  - [1]: Chapters 10, 11
  - [2]: Chapters 13, 14

Normalization

- **Normalization**: The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations
- **Normal form**: Using keys and FDs of a relation to certify whether a relation schema is in a particular normal form
- **Normalization** is carried out in practice so that the resulting designs are of high quality and meet the desirable properties
- The database designers ***need not*** normalize to the highest possible normal form (3NF, BCNF or 4NF)

Normalization

- 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

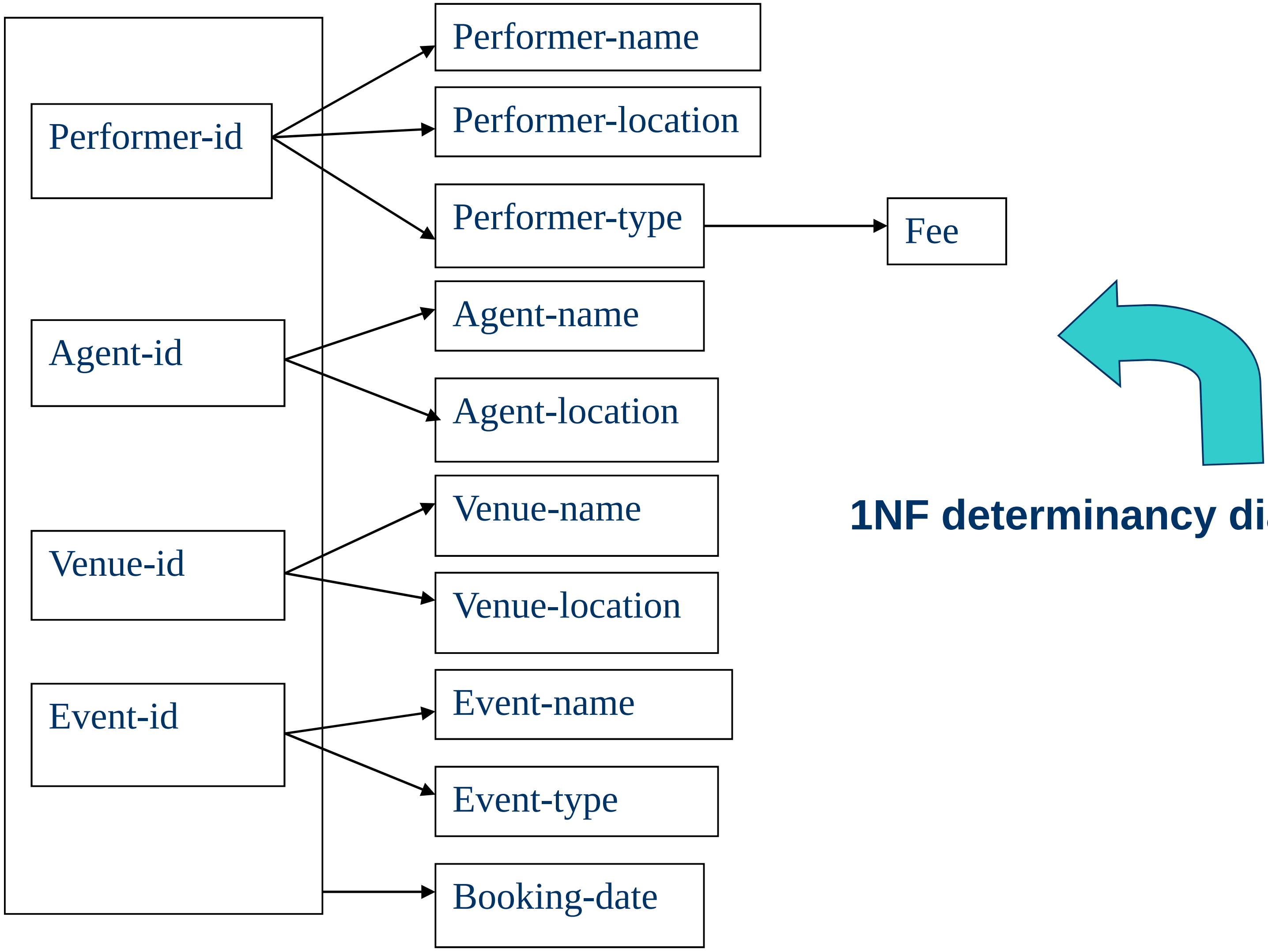
Normalization

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

Normalization

- First normal form (1NF): there is only one value at the intersection of each row and column of a relation - no set valued attributes in 1 NF → Disallows composite attributes, multivalued attributes, and **nested relations**
- To be part of the formal definition of a relation in the basic (flat) relational model

1NF



1NF determinancy diagram

Normalization

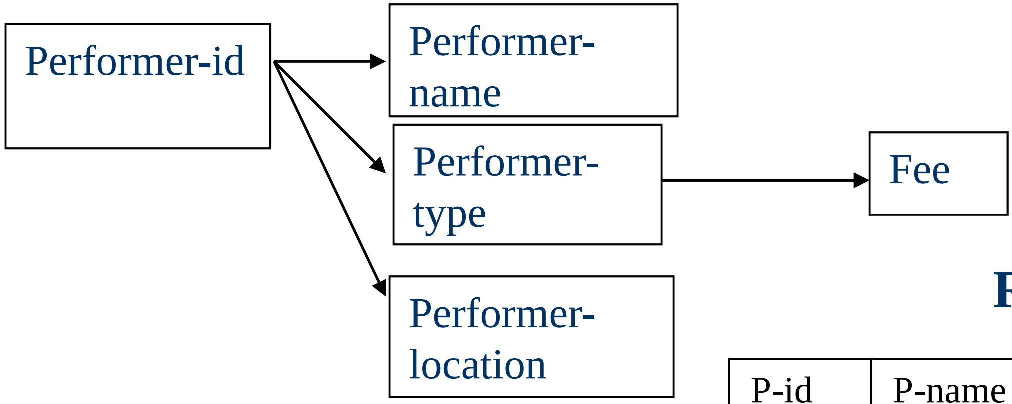
-
- 2NF – solves partial dependency
- 3NF – solves indirect dependency
- BCNF – well-normalized relations

Normalization

- Second normal form (2NF) - all attributes must be fully functionally dependent on the primary key
- 2NF solves partial dependency problem in 1NF
- Method: identify primary keys and group attributes that relate to the key together to form separate new relations

2NF

2NF determinancy diagram

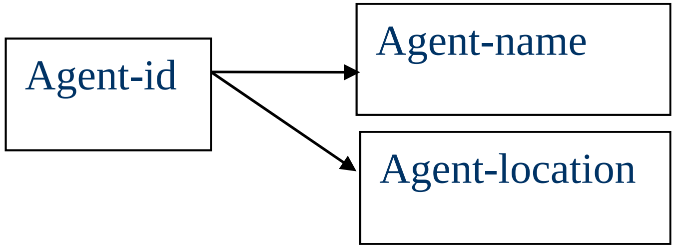


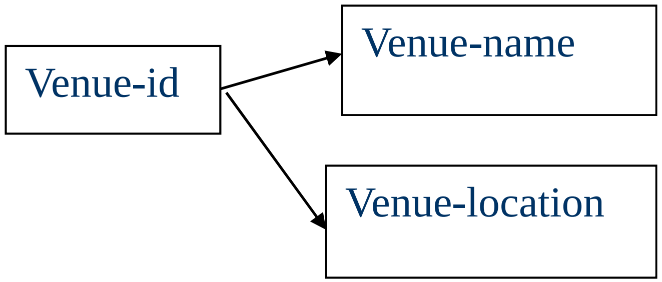
Relation in 2NF

P-id	P-name	P- type	Fee	P-loc’n
101	Baron	Singer	75	York
105	Steed	Dancer	60	Berlin
108	Jones	Actor	85	Bombay
112	Eagles	Actor	85	Leeds
118	Markov	Dancer	60	Moscow
126	Stokes	Comedian	90	Athens
129	Chong	Actor	85	Beijing
134	Brass	Singer	75	London
138	Ng	Singer	75	Penang
140	Strong	Magician	72	Rome
141	Gomez	Musician	92	Lisbon
143	Tan	Singer	75	Chicago
147	Qureshi	Actor	85	London
149	Tan	Actor	85	Taipei
150	Pointer	Magician	72	Paris
152	Peel	Dancer	60	London

2NF

2NF determinancy diagram





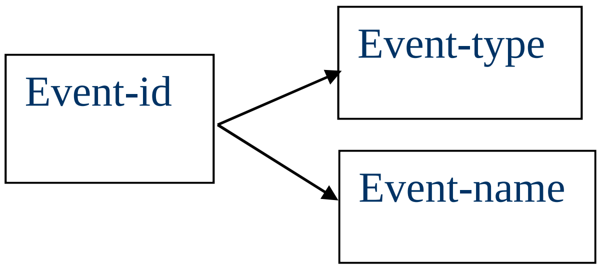
Relation in 2NF

A-id	A-name	A-loc'n
1295	Burton	Lonton
1435	Nunn	Boston
1504	Lee	Taipei
1682	Tsang	Beijing
1460	Stritch	Rome
1522	Ellis	Madrid
1509	Patel	York
1478	Burns	Leeds
1377	Webb	Sydney
1478	Burns	Leeds
1190	Patel	Hue
1802	Chapel	Bristol
1076	Eccles	Oxford
1409	Arkley	York
1428	Vemon	Cairo

V-id	V-name	V-loc'n
59	Atlas	Tokyo
35	Polis	Athens
54	Nation	Lisbon
17	Silbury	Tunis
46	Royale	Cairo
75	Vostok	Kiev
79	Festive	Rome
28	Gratton	Boston
84	State	Kiev
82	Tower	Lima
92	Palace	Milan
62	Shaw	Oxford

2NF

2NF determinancy diagram

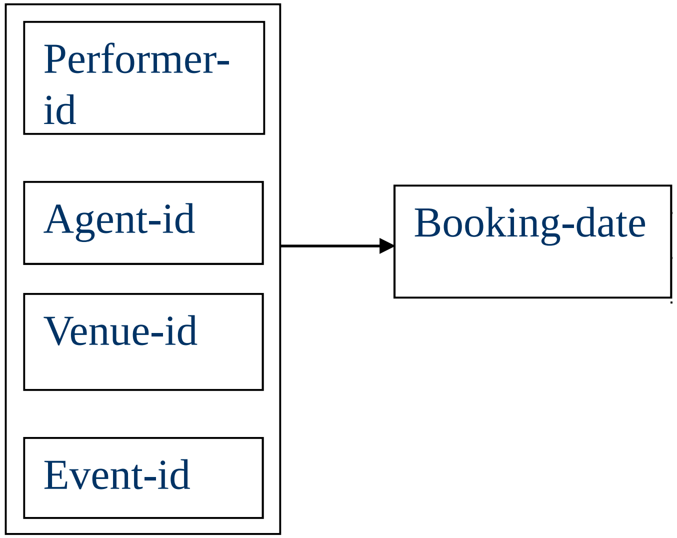


Relation in 2NF

E-id	E-name	E-type
959	Show Time	Musical
907	Elgar 1	Concert
921	Silver Shoe	Ballet
942	White Lace	Ballet
901	The Dark	Drama
913	What Now	Drama
926	Next Year	Drama
952	Gold Days	Drama
934	Angels	Opera
945	Trick-Treat	Variety show
938	New Dawn	Drama
981	Birdsong	Musical
957	Quicktime	Musical
963	Vanish	Magic show
941	Mahler 1	Concert
964	The Friends	Drama
927	Chanson	Opera
971	Card Trick	Magic show
988	Secret Tape	Drama
978	Swift Step	Dance

2NF

2NF determinancy diagram

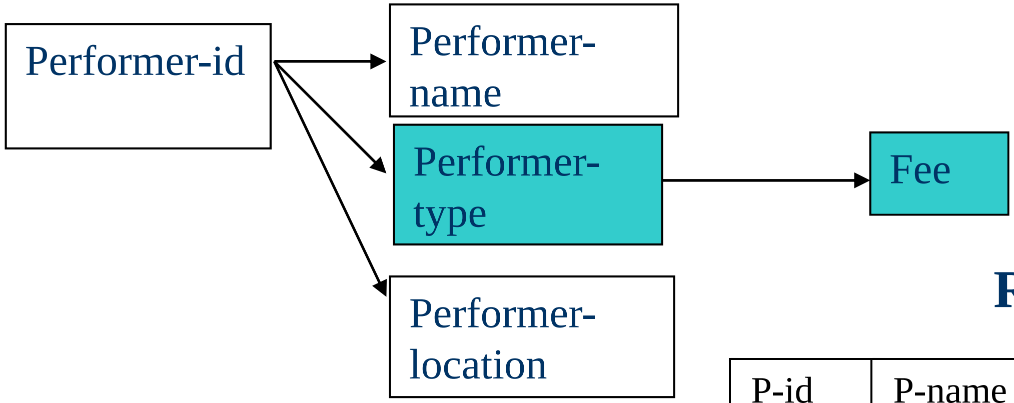


Relation in 2NF

P-id	A-id	V-id	E-id	Booking-date
101	1295	59	959	25-Nov-99
105	1435	35	921	07-Jan-02
105	1504	54	942	10-Feb-02
108	1682	79	901	29-Jul-03
112	1460	17	926	13-Aug-00
112	1522	46	952	05-May-99
112	1504	75	952	16-Mar-99
126	1509	59	945	02-Sept-01
129	1478	79	926	22-Jun-00
134	1504	28	981	18-Sept-01
138	1509	84	957	18-Aug-99
140	1478	17	963	18-Aug-99
141	1478	84	941	21-Jul-00
143	1504	79	927	21-Nov-02
147	1076	17	952	30-Apr-00
147	1409	79	988	17-Apr-00
152	1428	59	978	01-Oct-01

2NF

2NF determinancy diagram



Relation in 2NF

P-id	P-name	P- type	Fee	P-loc'n
101	Baron	Singer	75	York
105	Steed	Dancer	60	Berlin
108	Jones	Actor	85	Bombay
112	Eagles	Actor	85	Leeds
118	Markov	Dancer	60	Moscow
126	Stokes	Comedian	90	Athens
129	Chong	Actor	85	Beijing
134	Brass	Singer	75	London
138	Ng	Singer	75	Penang
140	Strong	Magician	72	Rome
141	Gomez	Musician	92	Lisbon
143	Tan	Singer	75	Chicago
147	Qureshi	Actor	85	London
149	Tan	Actor	85	Taipei
150	Pointer	Magician	72	Paris
152	Peel	Dancer	60	London

- **Problem with 2NF:**
  - Insertion
  - Modification
  - Deletion

Normalization

-
-
- 3NF – solves indirect dependency
- BCNF – well-normalized relations

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 <u>not</u> a candidate key. When Y is a candidate key, there is no problem with the transitive dependency .

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

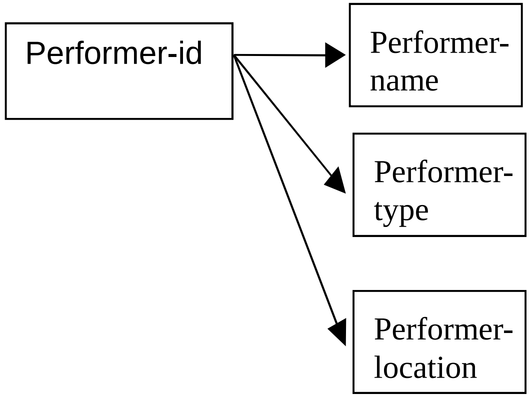
Here, SSN -> Emp# -> Salary and Emp# is a candidate key

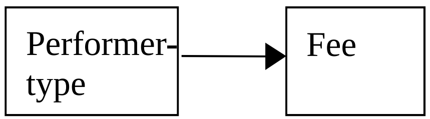
Normalization

- 3NF solves indirect (transitive) dependencies problem in 1NF and 2NF
- Method: identify all transitive dependencies and each transitive dependency will form a new relation, with non-prime attributes participating in the transitive dependency and the attribute which determines others as the attributes for the new relation

3NF

3NF determinancy diagram





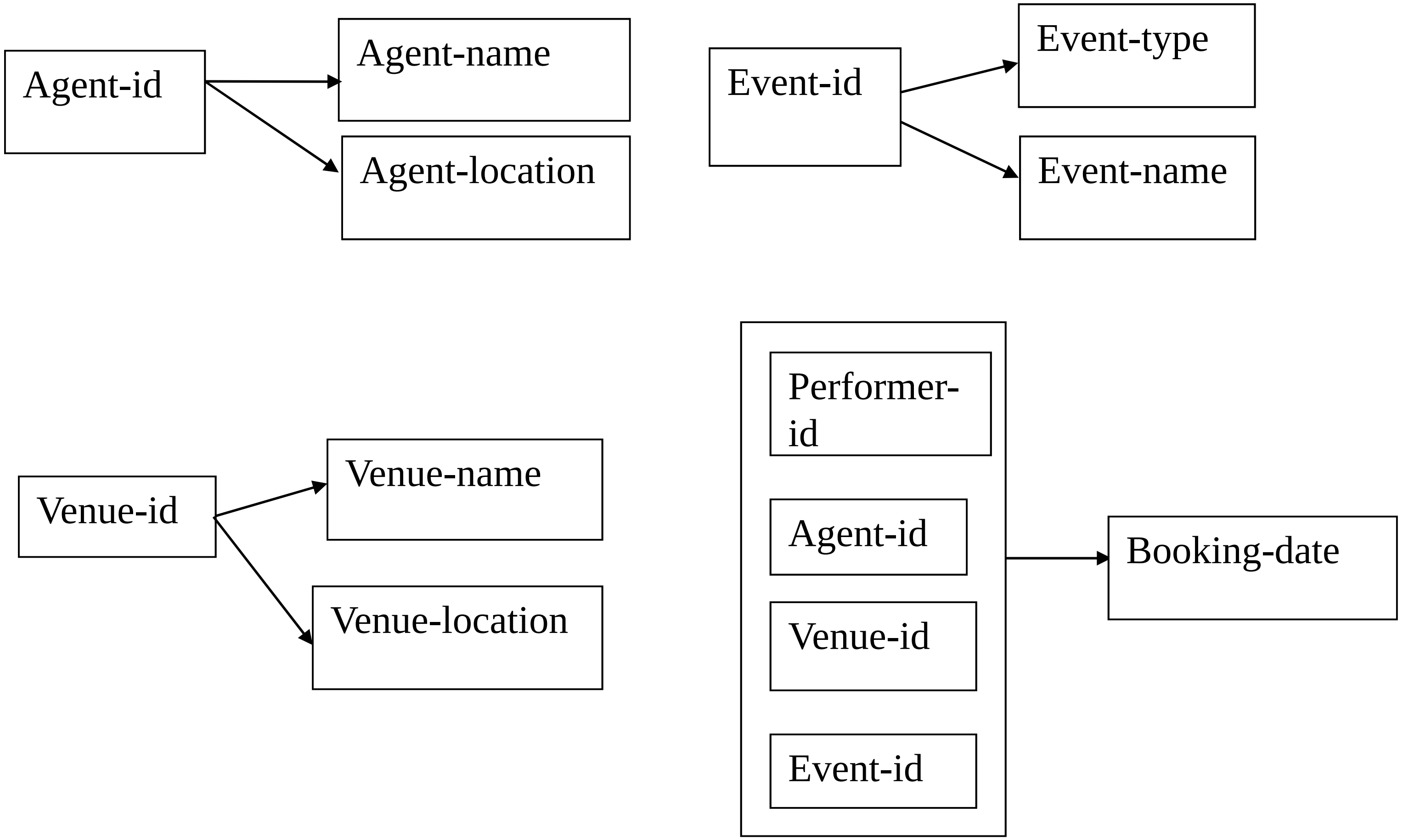
Relation in 3NF

P-id	P-name	P- type	P-loc'n
101	Baron	Singer	York
105	Steed	Dancer	Berlin
108	Jones	Actor	Bombay
112	Eagles	Actor	Leeds
118	Markov	Dancer	Moscow
126	Stokes	Comedian	Athens
129	Chong	Actor	Beijing
134	Brass	Singer	London
138	Ng	Singer	Penang
140	Strong	Magician	Rome
141	Gomez	Musician	Lisbon
143	Tan	Singer	Chicago
147	Qureshi	Actor	London
149	Tan	Actor	Taipei
150	Pointer	Magician	Paris
152	Peel	Dancer	London

P- type	Fee
Singer	75
Dancer	60
Actor	85
Comedian	90
Magician	72
Musician	92

3NF

3NF determinancy diagram



Normalization

-
-
-
- BCNF – well-normalized relations

General Normal Form Definitions

- The above definitions consider the primary key only
- The following more general definitions take into account relations with multiple candidate keys

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

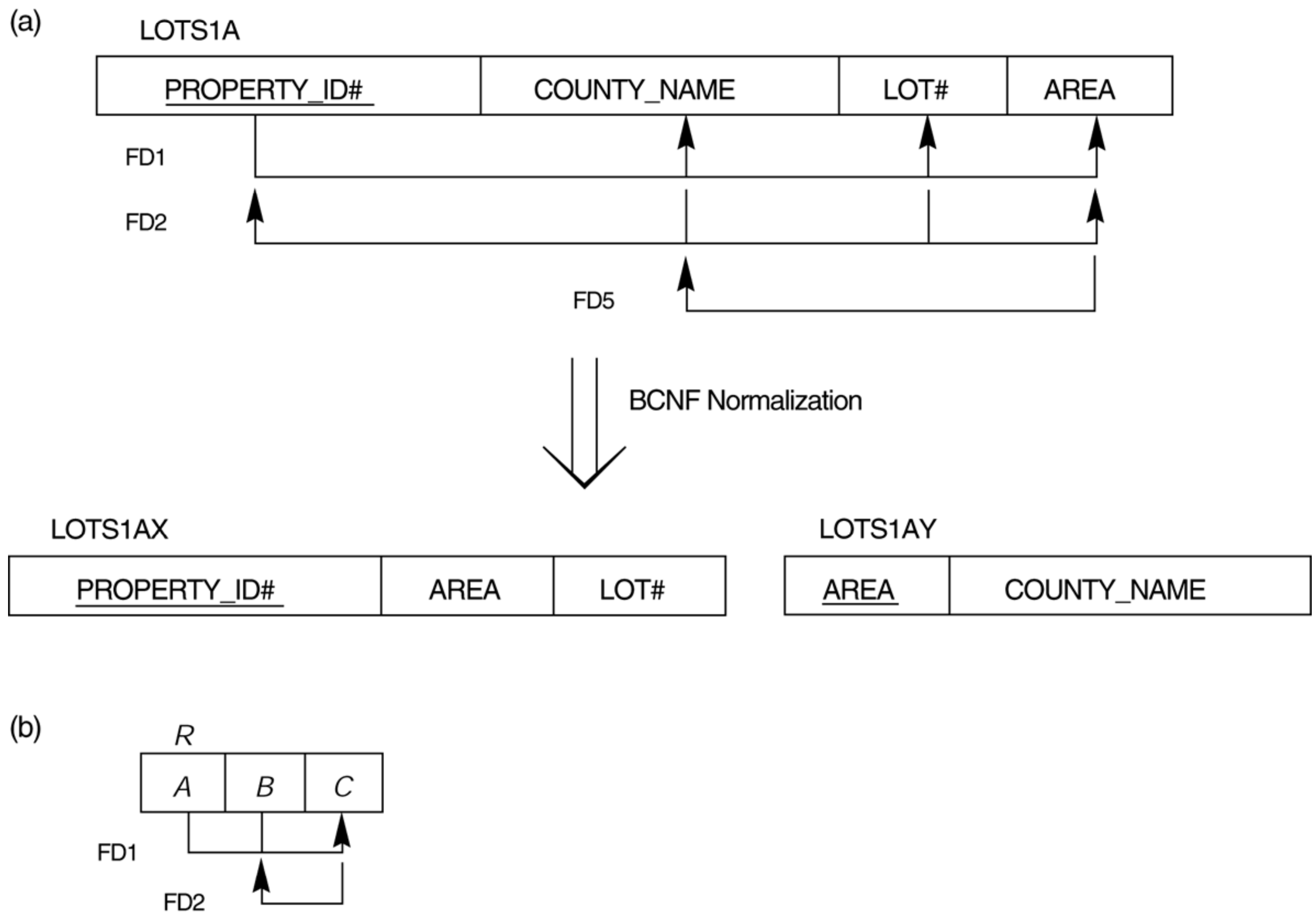
Normalization

-
-
-
- BCNF – well-normalized relations

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
- More details: [1] -> chapters 10, 11
- The goal is to have each relation in BCNF (or 3NF)

BCNF



Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF.

Notes & Suggestions

- [1]: chapter 11
  - 4NF: based on *multivalued dependency* (MVD)
  - 5NF: based on join dependency
    - Such a dependency is very difficult to detect in practice and therefore, normalization into 5NF is considered very rarely in practice
  - Other normal forms & algorithms
  - ER modeling: top-down database design
    - Bottom-up database design ??

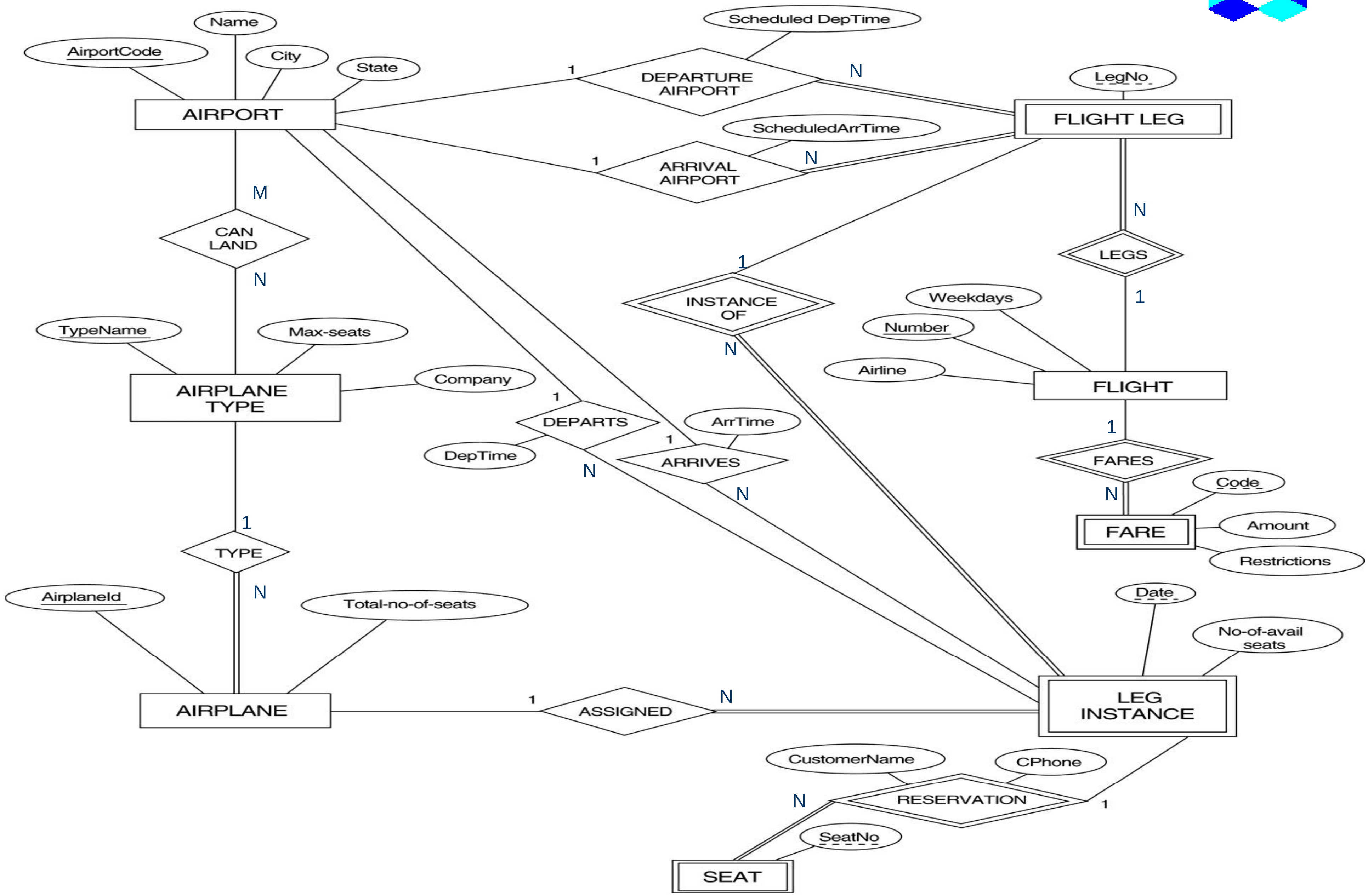
Outline

-
-
-
- Exercise
- Reading Suggestion
  - [1] chapters *(5)*, 7, 10, 11, *(12)*
  - *[4] chapters 13, 14, 15, 16*

Exercise 1
(ERD & Mapping)

- A database needs to be designed for the university student accommodation services. Each hall of residence has a number (unique), name (unique), address, phone, and the total number of rooms. Every hall contains single rooms only. Each room has a number (unique within a hall), and a weekly rent. For each student renting a room, the database should store his/her ID number, name (first and last), home address, date of birth, and the category of the student (for example, 1UG for the first uear undergraduate student). Whenever possible, information about a single next-of-kin related to a student is stored, including the name, relationship, address and phone number
- You are required to design this DB by showing a fully labelled ERD & the corresponding DB schema

Exercise 2 (Mapping)



Summary

- Relational Model
- Relational Database Design by ER/EER-to-Relational Mapping
- Functional Dependencies & Normalization
- Exercise
- Reading Suggestion: do not forget !!
- Next week:
  - Advanced Indexing Techniques & Flexible Query Answering in DBs
    - students' talks
    - [1] chapters 13, 14
    - Other papers (see my homepage)

Q&A

Questions ??