Database Systems
Spring 2013

Conceptual Database Design
SL06

- Entities, Entity Types, Entity Sets, Attributes
- Relationships, Relationship Types, Relationship Sets
- Weak Entities, N-ary Relationships
- Subclasses and Superclasses
- ER-to-Relational Mapping Algorithm

Literature and Acknowledgments

Reading List for SL06:

- Fundamentals of Database Systems, Chapters 7 and 8, Sixth Edition, Ramez Elmasri and Shamkant B. Navathe, Pearson Education, 2010. [optional: EWD696, EWD1036]

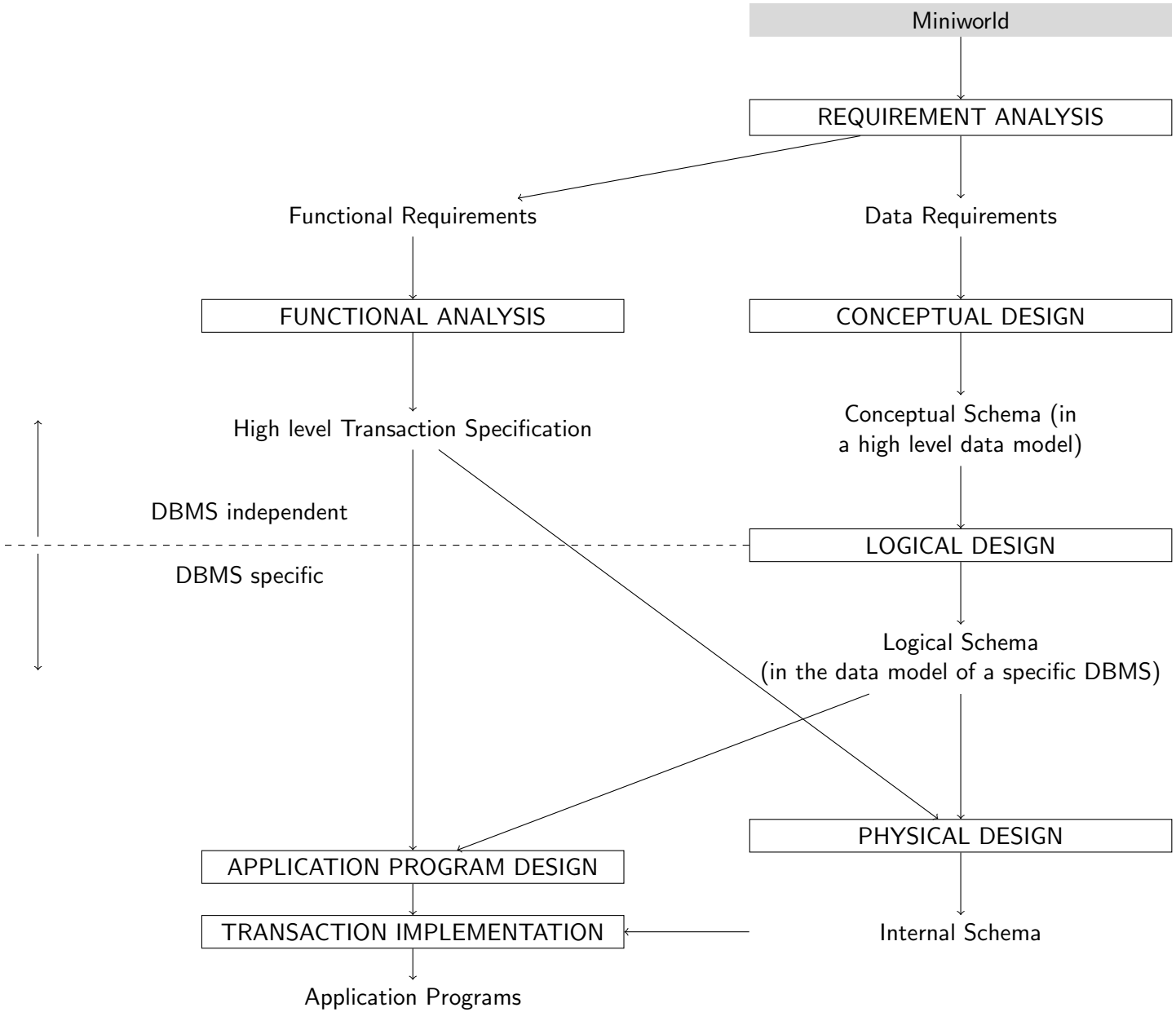
These slides were developed by:

- Michael Böhlen, University of Zürich, Switzerland
- Johann Gamper, Free University of Bozen-Bolzano, Italy

The slides are based on the following text books and associated material:

- Fundamentals of Database Systems, Fourth Edition, Ramez Elmasri and Shamkant B. Navathe, Pearson Addison Wesley, 2004.
- A. Silberschatz, H. Korth, and S. Sudarshan: Database System Concepts, McGraw Hill, 2006.

Overview of Database Design Process



Example COMPANY Database

- We want to create a database schema design based on the following **requirements** of the COMPANY database:
  - The company is organized into departments. Each department has a name, number and an employee who manages the department. We keep track of the start date of the department manager. A department may have several locations.
  - Each department controls a number of projects. Each project has a unique name, unique number and is located at a single location.
  - We store each employee's social security number, address, salary, sex, and birthdate. Each employee works for one department but may work on several projects. We keep track of the number of hours per week that an employee currently works on each project. We also keep track of the direct supervisor of each employee.
  - Each employee may have a number of dependents. For each dependent, we keep track of their name, sex, birthdate, and relationship to the employee.

Entities and Attributes

- Entities, entity types, entity sets
- Attributes

Entities and Attributes

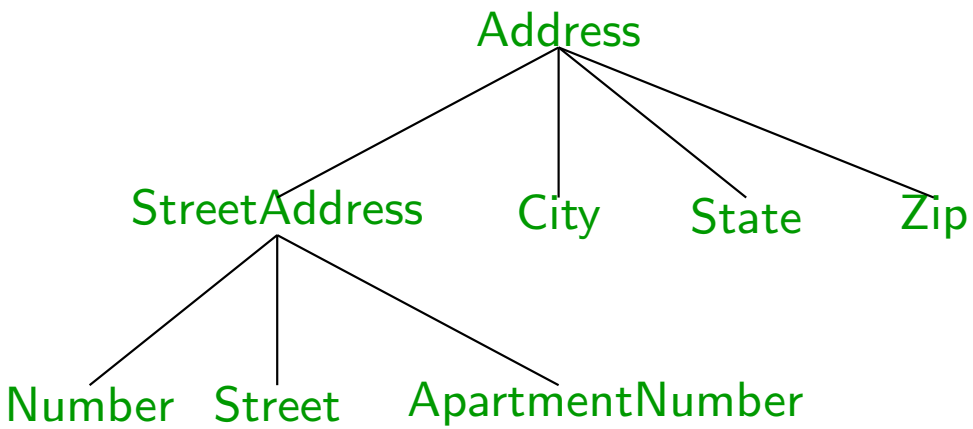
- **Entities** are specific objects or things in the mini-world that are represented in the database.
  - For example employee John Smith, the Research department, project ProductX
- **Attributes** are properties used to describe an entity.
  - For example an employee entity may have the attributes Name, SSN, Address, Sex, BirthDate
- A specific entity will have a value for each of its attributes.
  - For example a specific employee entity may have
    Name = 'John Smith',
    SSN = '123456789',
    Address = '731, Fondren, Houston, TX',
    Sex = 'M',
    BirthDate = '09-JAN-55'
- Each attribute has a *domain* (value set, data type) associated with it, e.g., enumerated type, integer, string, subrange, ...

Types of Attributes/1

- **Simple attribute**
  - Each entity has a single atomic value for the attribute. For example, SSN or Sex.
- **Composite attribute**
  - The attribute may be composed of several components. For example:
    - Address(Apt#, House#, Street, City, State, ZipCode, Country), or
    - Name(FirstName, MiddleName, LastName).
- **Multi-valued attribute**
  - An entity may have multiple values for that attribute. For example, color of a cars or degrees of a student.
    - Denoted as {Color} or {PreviousDegrees}.
- **Derived attribute**
  - Attributes can be derived (computed) rather than stored. Attribute Number_of_employees is a derived attribute.

Types of Attributes/2

- Composite and multi-valued attributes may be nested arbitrarily to any number of levels.
- For example, PreviousDegrees of a student is a composite multi-valued attribute denoted by {PreviousDegrees (College, Year, Degree, Field)}
  - Multiple PreviousDegrees values can exist
  - Each has four subcomponent attributes:
    - College, Year, Degree, Field
- A hierarchy of composite attributes:

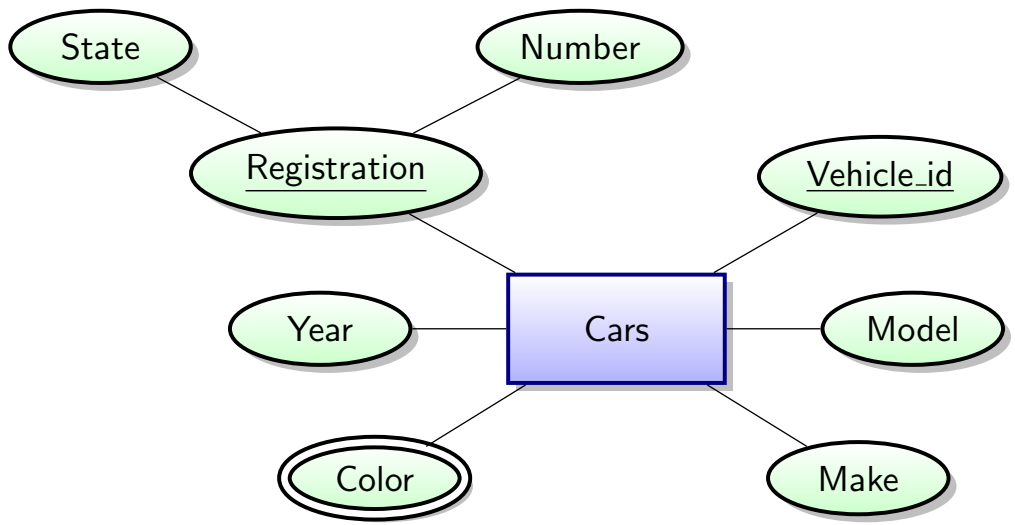


Entity Types and Key Attributes

- Entities with the same basic attributes are grouped or typed into an **entity type**.
  - For example, the entity types Employees and Projects.
- An attribute of an entity type for which each entity must have a unique value is called a **key attribute** of the entity type.
  - For example, SSN of Employees.
- A key attribute may be composite.
  - VehicleTagNumber is a key of the Cars entity type with components (Number, State).
- An entity type may have more than one key.
  - The Cars entity type may have two keys:
    - VehicleIdentificationNumber (popularly called VIN)
    - VehicleTagNumber(Number, State), aka license plate number.
- Each key is underlined.

Displaying an Entity Type

- In ER diagrams, an **entity type** is displayed in a rectangular box
- Attributes are displayed in ovals
  - Each attribute is connected to its entity type
  - Components of a composite attribute are connected to the oval representing the composite attribute
  - Each key attribute is underlined
  - Multivalued attributes are displayed in double ovals
  - Derived attributes are displayed in dashed ovals
- Entity type Cars with attributes Registration(Number, State), Vehicle_id, Make, Model, Year, {Color}



Review 6.1

Consider schema $R(A, B, C, D, E)$ with candidate keys $\{A, BE, C\}$.
Represent R as an entity type.

Entity Set

- The collection of all entities of a particular entity type in the database is called the **entity set**.
- The same name (e.g., Cars) is used to refer to both the entity type and the entity set.
- The entity set is the current *state* of the entities of that type that are stored in the database.
- Entity set Cars:

Car_1
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004, {red, black})

Car_2
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

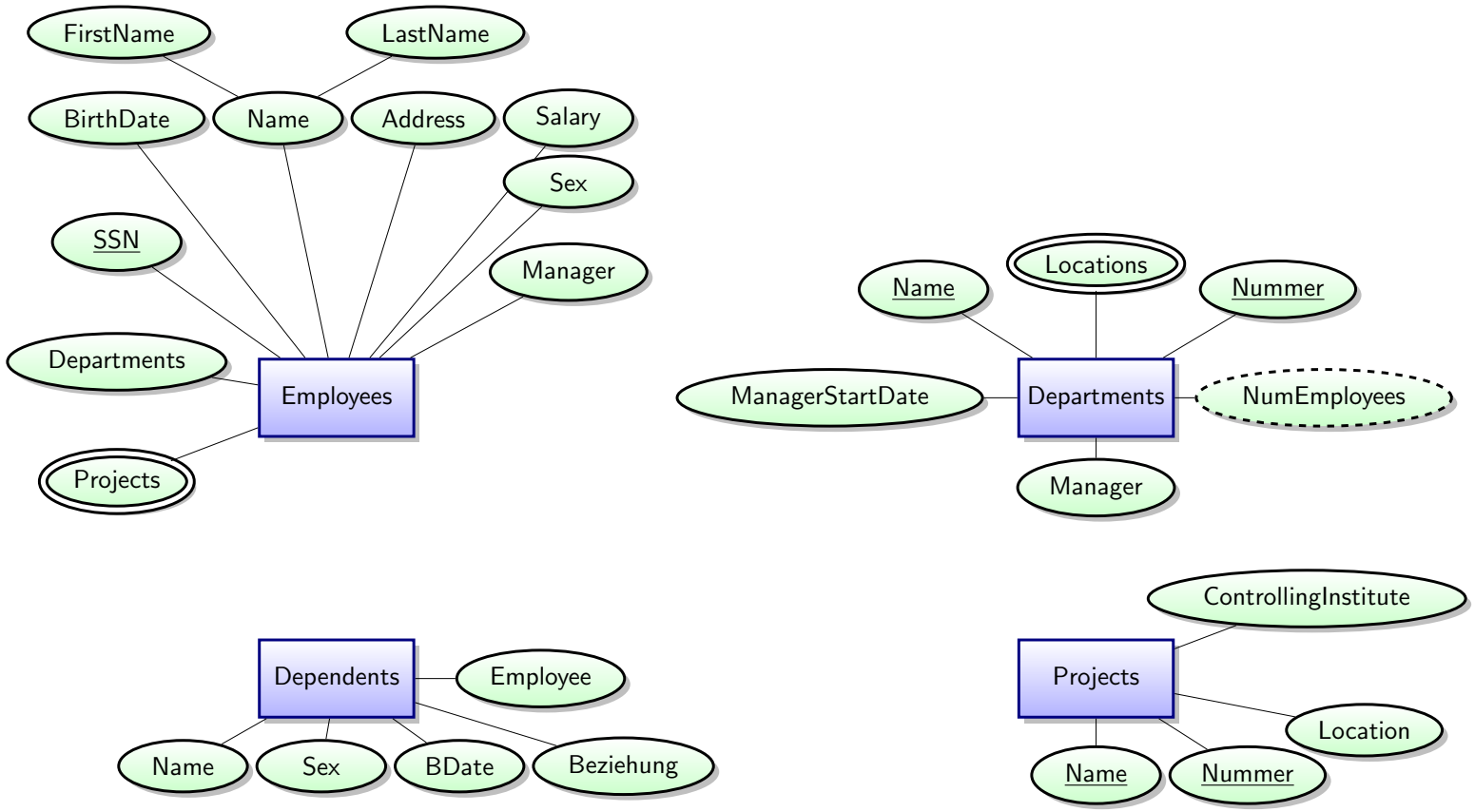
Car_3
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

.
.
.

Entity Types for the COMPANY Database/1

- Based on the requirements, we can identify four initial entity types in the COMPANY database:
  - Departments
  - Projects
  - Employees
  - Dependents
- The initial attributes are derived from the requirements description
- Guidelines for determining entity types and attributes:
  - The *nouns* in a descriptions translate to entity types
    - "organized into departments"
  - *Nouns that describe nouns* of entity types translate to attributes.
    - "department has a name, number"

Entity Types for the COMPANY Database/2



Relationships

- Relationships, Relationship Types, Relationship Sets
- Structural Constraints on Relationships
- Recursive Relationships
- Attributes of Relationships

Relationships and Relationship Types/1

- ER model has three main concepts:
  - Entities (and their entity types and entity sets)
  - Attributes (simple, composite, multivalued)
  - Relationships (and their relationship types and relationship sets)
- The initial design is typically not complete.
- The schema design process is an *iterative refinement process*. The initial design is created and iteratively refined.
- Some aspects in the requirements will be represented as **relationships**.

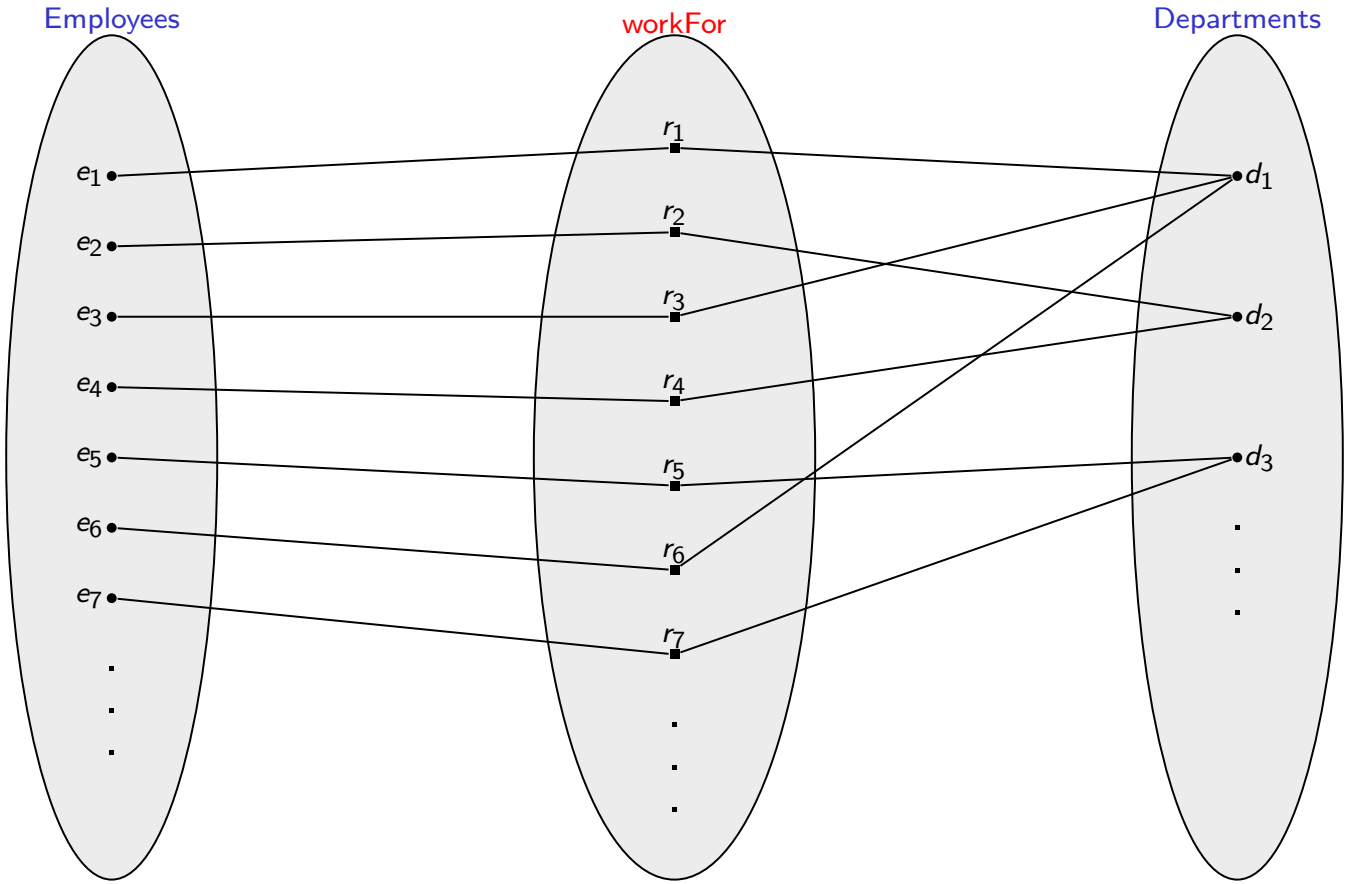
Relationships and Relationship Types/2

- A **relationship** relates two or more distinct entities with a specific meaning. For example,
  - employee John Smith *works on* project ProductX, or
  - employee Franklin Wong *manages* the research department.
- Relationships of the same type are grouped or typed into a **relationship type**.
  - For example, the workOn relationship type in which employees and projects participate, or the manage relationship type in which employees and departments participate.
- The degree of a relationship type is the number of participating entity types.
  - Both manage and workOn are *binary* relationships.
- In ER diagrams, we represent a *relationship type* as follows:
  - Diamond-shaped box is used to display a relationship type
  - Connected to the participating entity types via straight lines

Relationships and Relationship Types/3

- Relationship type:
  - Schema description of a relationship
  - Identifies the relationship name and the participating entity types
  - Identifies relationship constraints
- **Relationship set**:
  - The current set of relationship instances represented in the database
  - The current *state* of a relationship type
- Each instance in the set relates individual participating entities – one from each participating entity type

The workFor Relationship

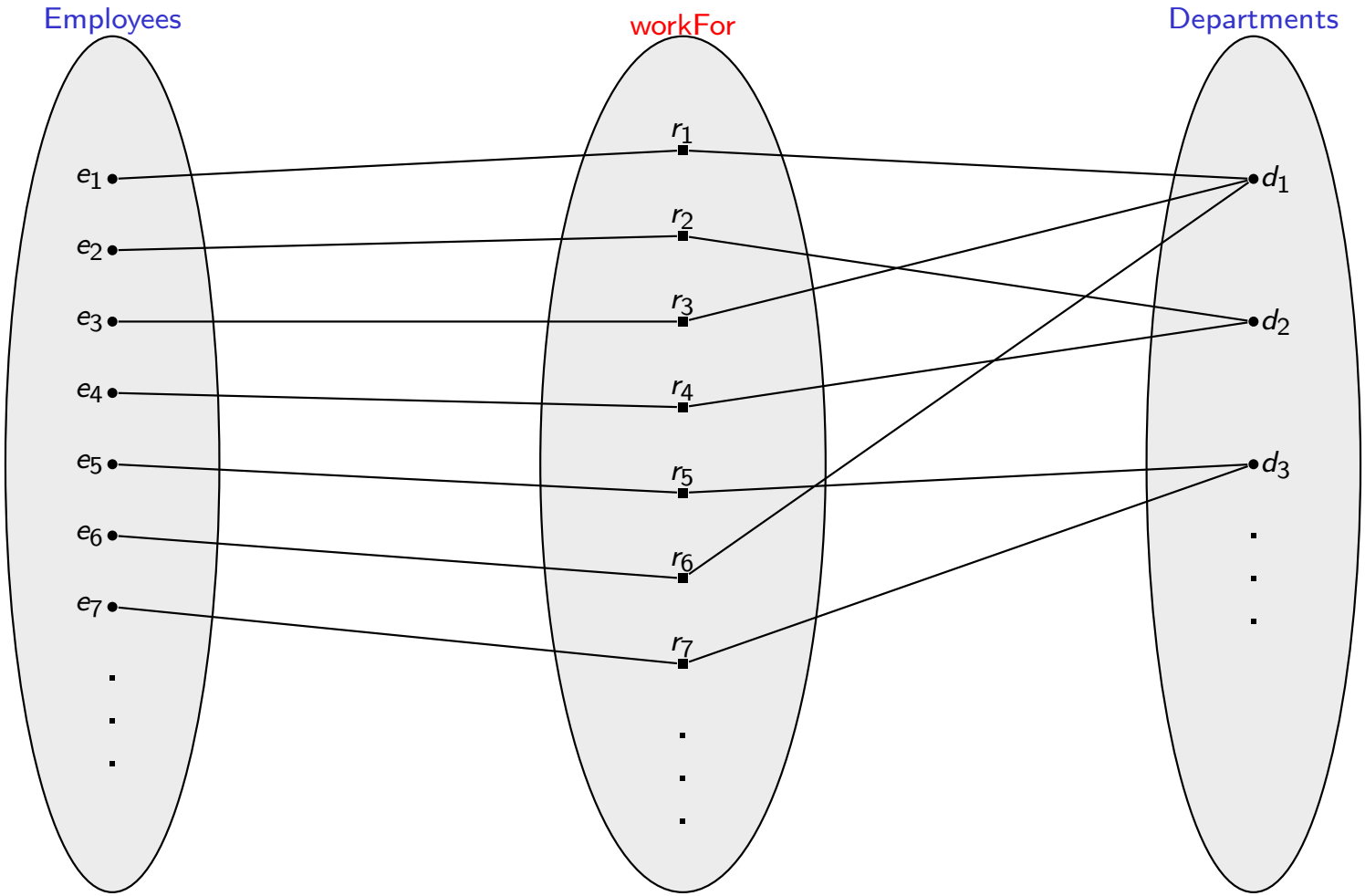


- entity: e_1
- relationship: $r_1 = \text{workFor}(e_1, d_1)$
- entity set: Employees
- relationship set: workFor

Structural Constraints on Relationships

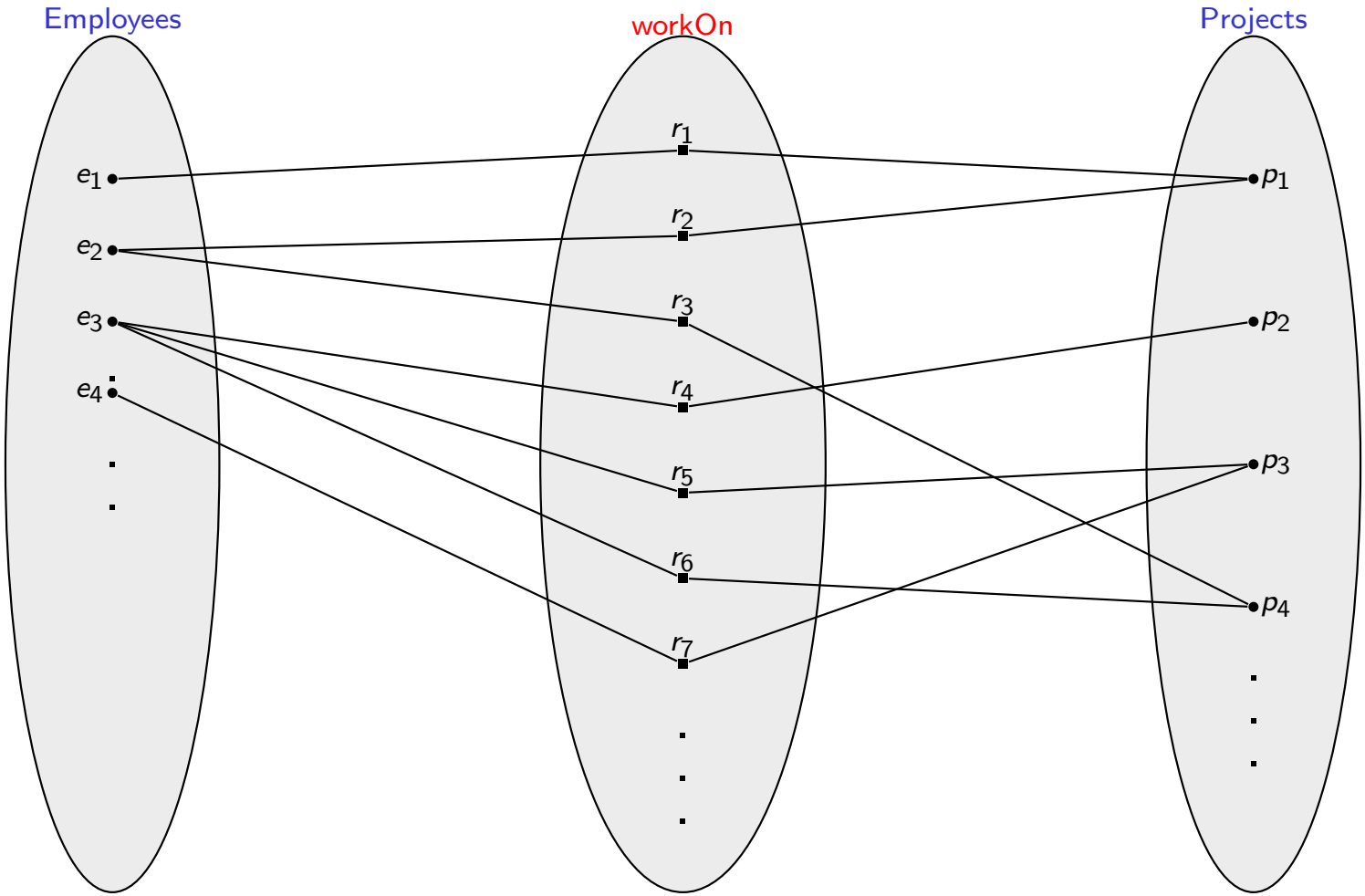
- Structural constraints on relationship types limit the possible combinations of entities in relationship sets.
- A **cardinality constraint** specifies *maximum* participation
  - One-to-one (1:1)
  - One-to-many (1:N) or Many-to-one (N:1)
  - Many-to-many (M:N)
- A **participation constraint** specifies *minimum* participation (also called existence dependency)
  - zero (optional participation, not existence-dependent)
  - one or more (mandatory participation, existence-dependent)

Many-to-one (N:1) Relationship



- Employees:Departments = N:1
- An employee works for at most 1 department
- A department employs at most N employees

Many-to-many (M:N) Relationship

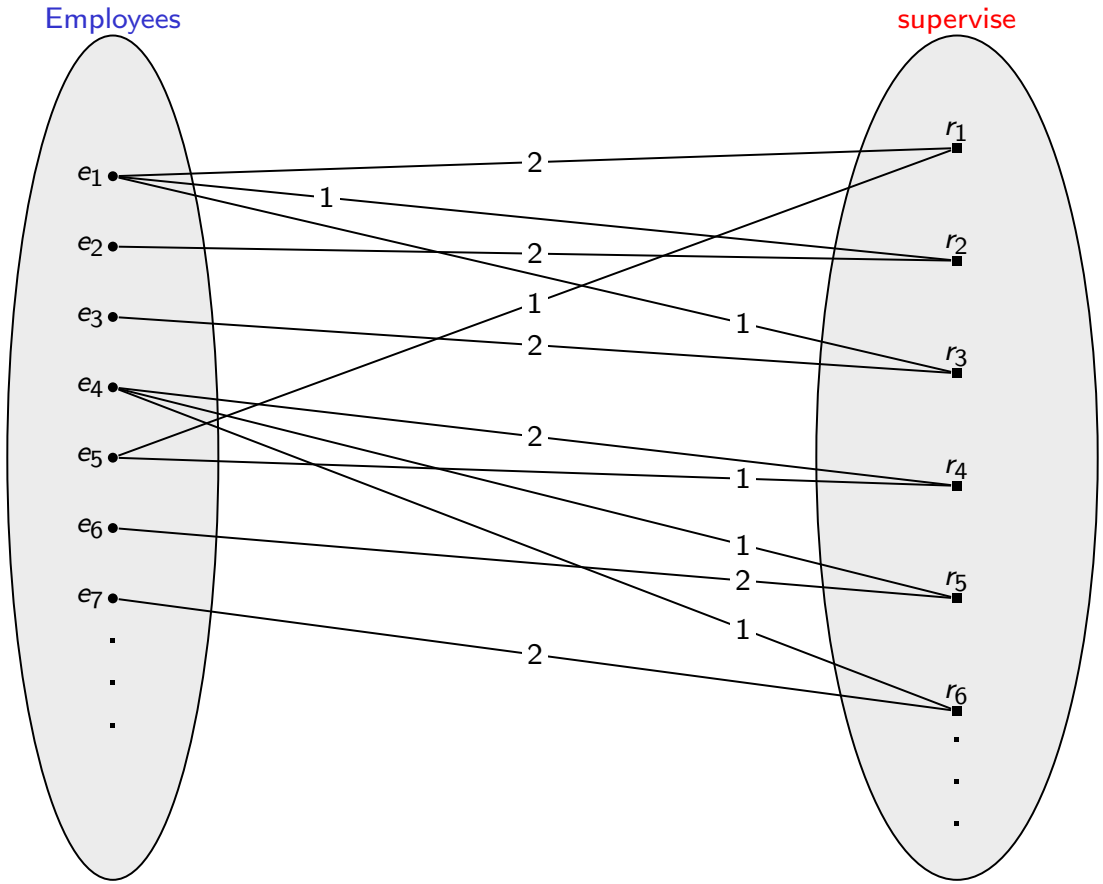


- Employees:Projects = M:N
- An employee works on at most N projects
- A project is assigned to at most M employees

Recursive Relationships/1

- In a recursive relationship type:
  - The same entity type participates in different roles.
  - For example, the supervise relationships between
    - an employee in role of supervisor (or boss) and
    - another employee in role of subordinate (or worker).
- In ER diagrams we need to display role names to distinguish participations.

Recursive Relationships/2



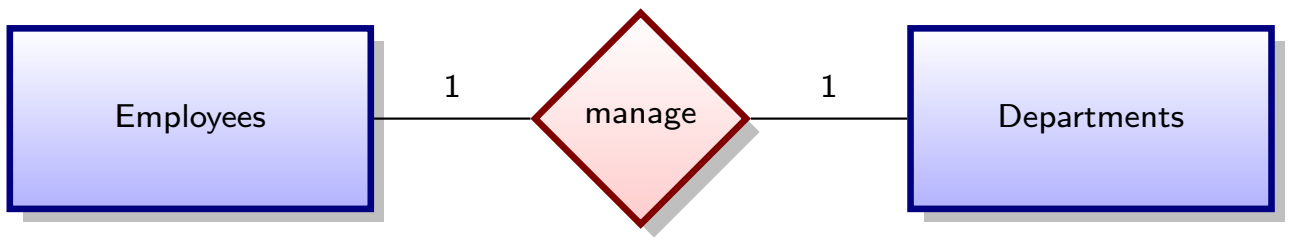
- label 1 stands for *supervisor* role
- label 2 stand for *subordinate* role
- e_1 is the supervisor of e_2
- e_1 is supervised by e_5

Attributes of Relationship Types

- A relationship type can have attributes:
  - For example, HoursPerWeek of workOn
  - Its value for each relationship instance describes the number of hours per week that an employee works on a project.
    - A value of HoursPerWeek depends on a particular (employee, project) combination
  - Most relationship attributes are used with M:N relationships
    - In 1:N relationships, they can be transferred to the entity type on the N-side of the relationship

Notation for Constraints on Relationships/1

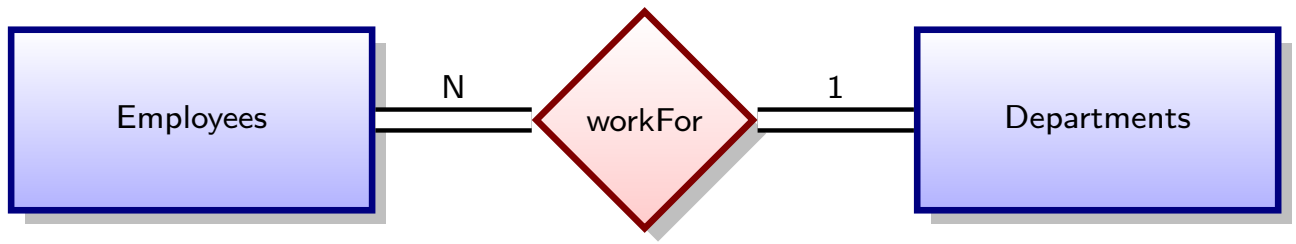
- **Cardinality constraints** (of a binary relationship): 1:1, 1:N, N:1, or M:N
- Cardinality constraints are shown by placing appropriate numbers on the relationship edges.



- Reading is from left-to-right and top-down (usually; there are always exceptions) and the terms are chosen accordingly.
- Reading: An employee manages 1 department
- Inverse: A department is managed by 1 employee

Notation for Constraints on Relationships/2

- **Participation constraint** on each participating entity type: total (called existence dependency) or partial.
- Total participation is shown by double line, partial participation by single line.

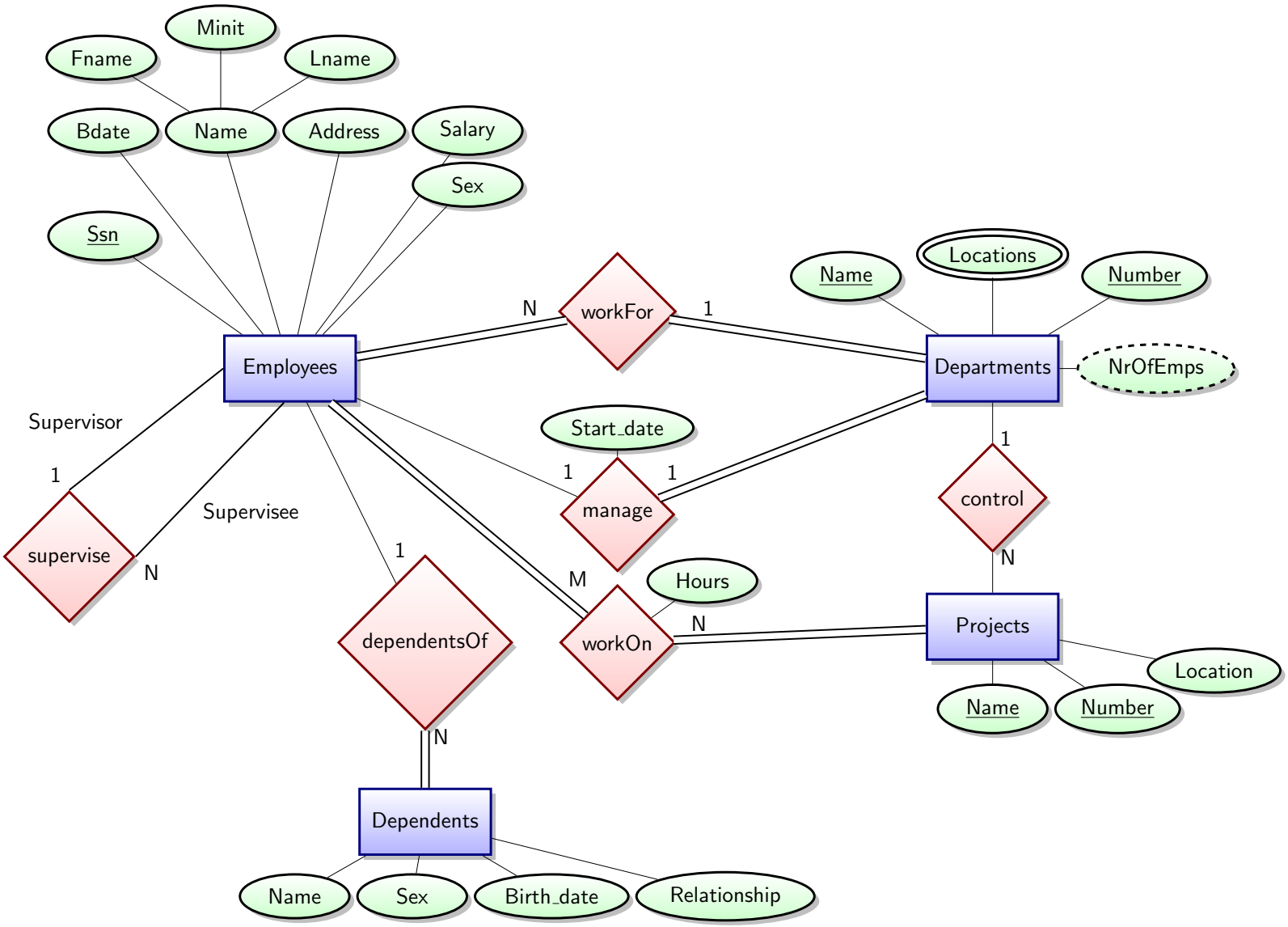


- A total participation constraint specifies a minimum participation:
  - An employee must workFor a department
  - A department must giveWorkTo N employees
- Structural constraints are easy to specify for binary relationship types but get more subtle for higher order relationship types.

Relationships for the COMPANY Database/1

- By examining the requirements, six relationship types are identified
- All are *binary* relationships (degree 2)
- Relationship types with participating entity types:
  - workFor (between Employees and Departments)
  - manage (also between Employees and Departments)
  - control (between Departments and Projects)
  - workOn (between Employees and Projects)
  - supervise (between Employees (as subordinate) and Employees (as supervisor))
  - dependentsOf (between Employees and Dependents)

Relationships for the COMPANY Database/2



Discussion of Relationship Types

- In the refined design, some attributes from the initial entity types are refined into relationships:
  - Manager of Departments -> manage
  - Projects of Employees -> workOn
  - Department of Employees -> workFor
  - etc
- More than one relationship type can exist between the same participating entity types:
  - manage and workFor are distinct relationship types between Employees and Departments.
  - These relationship types have different meanings and different relationship instances.

Review 6.2

Assume a company employs agents to sell products to customers. Agents are based in a city and they work on commission. Customers negotiate individual discounts. For products we record production location, price, and sales quantity. Design an appropriate ER schema.

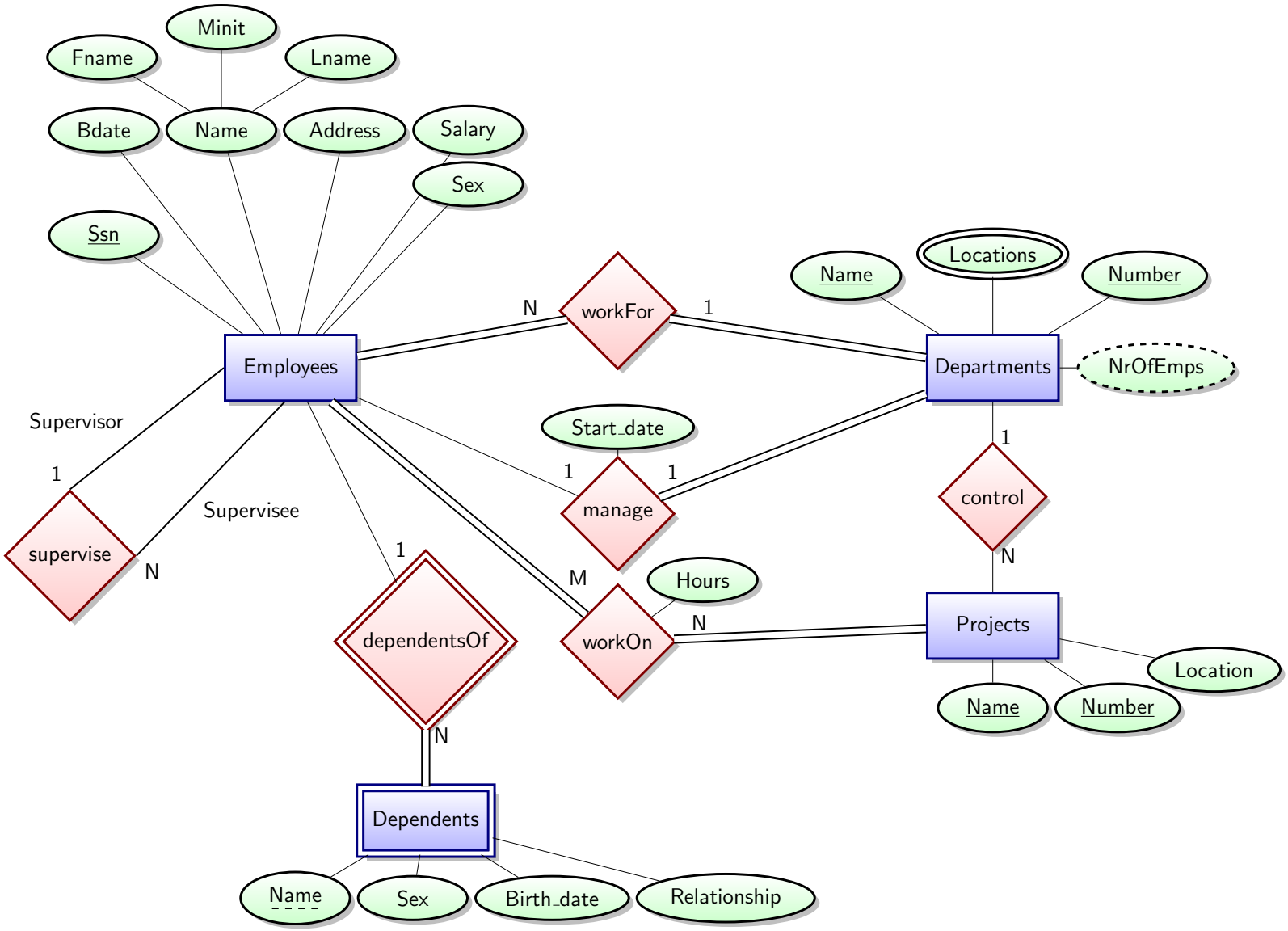
Weak Entities and N-ary Relationships

- Weak Entities
- N-ary Relationships

Weak Entity Types

- A **weak entity type** is an entity type that does not have a key attribute.
- A weak entity must participate in an **identifying relationship type** with an owner or identifying entity type.
- Entities are identified by the combination of:
  - A partial key of the weak entity type
  - The particular entity they are related to in the identifying entity type
- Example:
  - A Dependent entity is identified by the dependent's first name, and the specific Employee with whom the dependent is related.
  - Name of Dependent is the *partial key*
  - Dependent is a *weak entity type*
  - Employee is its identifying entity type via the identifying relationship type dependentOf.

The COMPANY Database



Review 6.3

Construct an ER schema for a hospital with patients and medical doctors. For each patient we keep a log of various tests and examinations.

Review 6.4

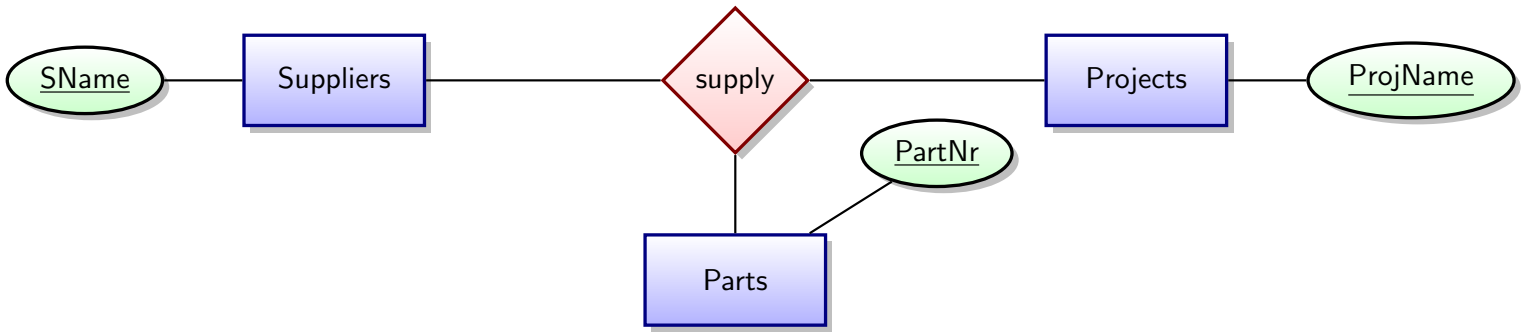
Assume we want to schedule classrooms for the final exams at a university. The examination unit are sections of courses. Propose an ER diagram.

Review 6.5

An airline reservation system keeps track of passengers identified by ticket number, flights identified by flight number, gates identified by gate number, and seat assignments. Design an appropriate ER schema.

Relationships of Higher Degree/1

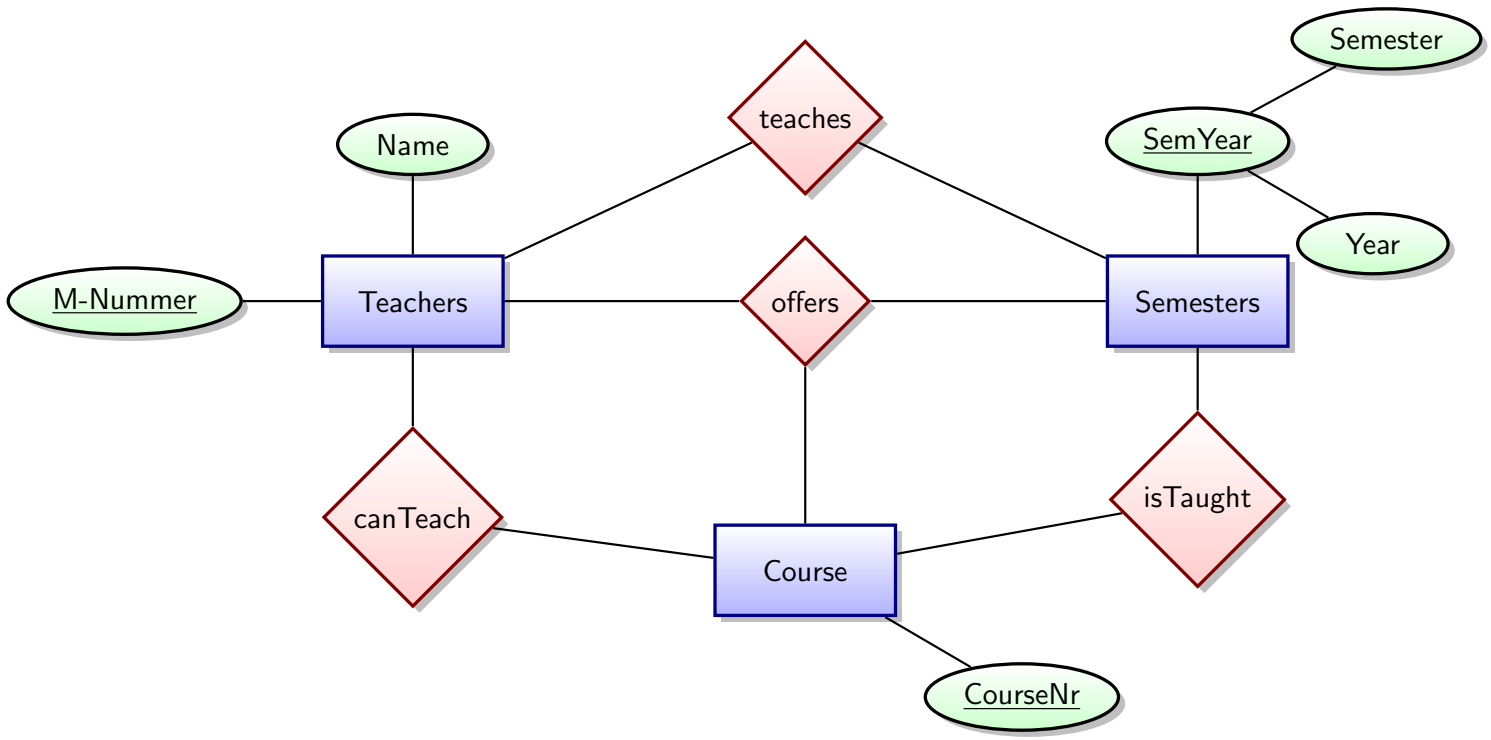
- Relationship types of degree 2 are called binary.
- Relationship types of degree 3 are called ternary and of degree n are called n-ary.



- Constraints are harder to specify for higher-degree relationships (n>2) than for binary relationships.
- In an n-ary relationship a cardinality constraint specifies how often an entity may occur for one specific instance of all other entities.
- In general, an n-ary relationship is not equivalent to n binary relationships.
- If needed, the binary and n-ary relationships can all be included in the schema design.

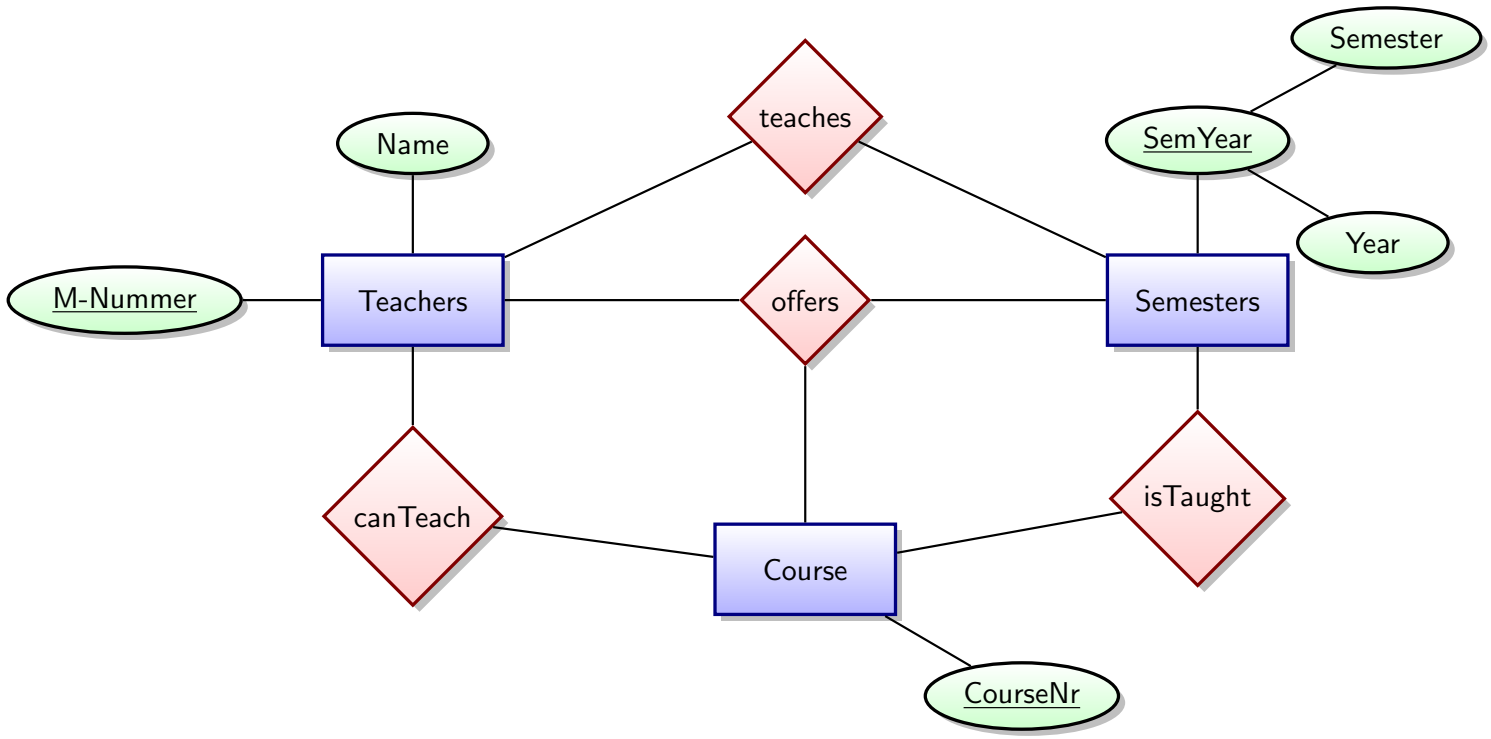
Relationships of Higher Degree/2

- An n-ary relationship is not equivalent to n binary relationships. The schema can include binary as well as n-ary relationships.
- The relationship offers cannot be derived from teaches, canTeach, isTaught.



Relationships of Higher Degree/3

- If a particular binary relationship can be derived from a higher-degree relationship at all times, then it is *redundant*.
- The teaches binary relationship can be derived from the ternary relationship offers (based on the meaning of the relationships).



Review 6.6

Use the ER model to model job applications. Candidates submit applications to companies and possibly get invited to an interview.

Alternative Diagrammatic Notation

- ER diagrams is one popular example for displaying database schemas.
- Many other notations exist in the literature and in various database design and modeling tools.
- UML class diagrams are another way of displaying ER concepts.

- Choose a systematic naming (there is no standard), e.g.,
  - Plural names for entity types
  - Upper case for entity type and lower case for relationship type
  - Initial letter capitalized for attribute names

Summary of Notation for ER Diagrams

Entität

existenzabhängige Entität

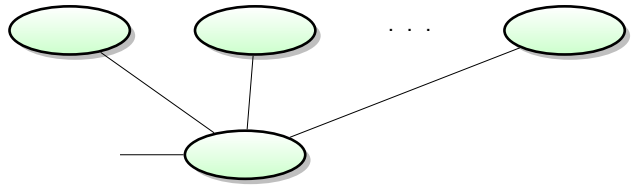
Beziehung

Beziehung zu übergeordnetem Entitätstyp

Attribute

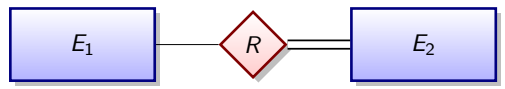
Schlüsselattribute

Mehrwertige Attribute

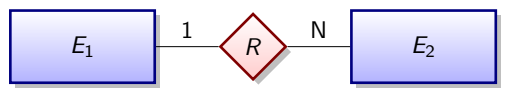


zusammengesetzte Attribute

abgeleitete Attribute



Totale Zugehörigkeit von E_2 in R



Kardinalitätseinschränkung 1:N für E_1:E_2 in R

Data Modeling Tools

- A number of popular tools that conceptual modeling and mapping into relational schema design.
  - Examples: ERWin, S-Designer (Enterprise Application Suite), ER-Studio, etc.
- Pros:
  - Serves as documentation of application requirements
  - Easy user interface: mostly graphics editor support
  - Simple graphical models are very intuitive
- Cons:
  - Graphical models easily get complex and ambiguous
  - Mostly represent a relational design in a diagrammatic form rather than a conceptual ER-based design

Subclasses and Superclasses

- Sub- and Superclasses
- Disjointness Constraint (disjoint, overlapping)
- Completeness Constraint (total, partial)

Subclasses and Superclasses/1

- An important extension that was not present in the initial ER model are subgroupings.
- An entity type may have additional meaningful subgroupings of its entities
- Example: Employees may be further grouped into:
  - Secretaries, Engineers, Technicians, ...
    – Based on the job of an employee
  - Managers
    – Employees who are managers
  - SalariedEmps, HourlyEmps
    – Based on the method of pay
- Extended ER diagrams represent these additional subgroupings, called *subclasses*.

Subclasses and Superclasses/2

- Each subgrouping is a subset of Employees
- Each subgrouping is a subclass of Employees
- Employees is the superclass for each of these subclasses
- These are called superclass/subclass relationships:
  - Employees/Secretaries
  - Employees/Technicians
  - Employees/Managers
  - ...
- Superclass/subclass relationships are also called IS-A relationships
  - Secretaries IS-A Employees
  - Technicians IS-A Employees
  - ...

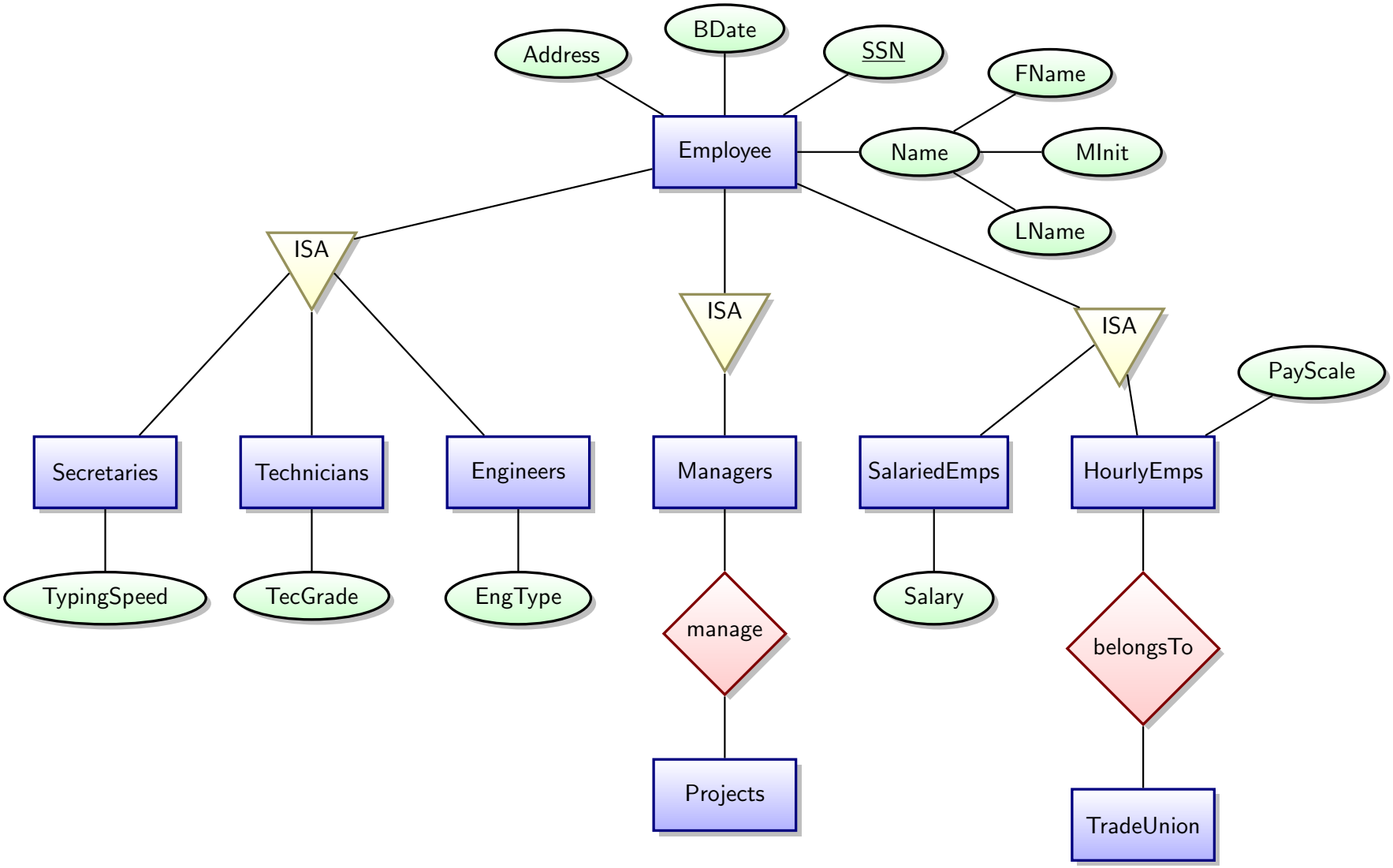
Subclasses and Superclasses/3

- An entity that is member of a subclass represents the same real-world entity as some member of the superclass:
  - The subclass member is the same entity in a *distinct specific role*
  - An entity cannot exist in the database merely by being a member of a subclass; it must also be a member of the superclass
- A member of the superclass can be optionally included as a member of any number of its subclasses
- Examples:
  - A salaried employee who is also an engineer belongs to two subclasses:
    - Engineers, and
    - SalariedEmps

Attribute Inheritance

- An entity that is member of a subclass *inherits*
  - All attributes of the entity as a member of the superclass
  - All relationships of the entity as a member of the superclass
- Example:
  - Secretaries (as well as Technicians and Engineers) inherit the attributes Name, SSN, . . . , from Employees
  - Every Secretaries entity will have values for the inherited attributes

Example of Subclasses and Superclasses



Specialization and Generalization

- Specialization is the process of defining a set of subclasses of a superclass
- The set of subclasses is based upon some distinguishing characteristics of the entities in the superclass
- Example: SECRETARY, ENGINEER, TECHNICIAN is a specialization of EMPLOYEE based upon *job type*.
- We may have several specializations of the same superclass
- Generalization is the reverse of the specialization process
- Several classes with common features are generalized into a superclass.
- Example: CAR, TRUCK generalized into VEHICLE;
  - CAR, TRUCK become subclasses of the superclass VEHICLE.
  - We can view {CAR, TRUCK} as a specialization of VEHICLE
  - Alternatively, we can view VEHICLE as a generalization of CAR and TRUCK

Constraints on Specialization/1

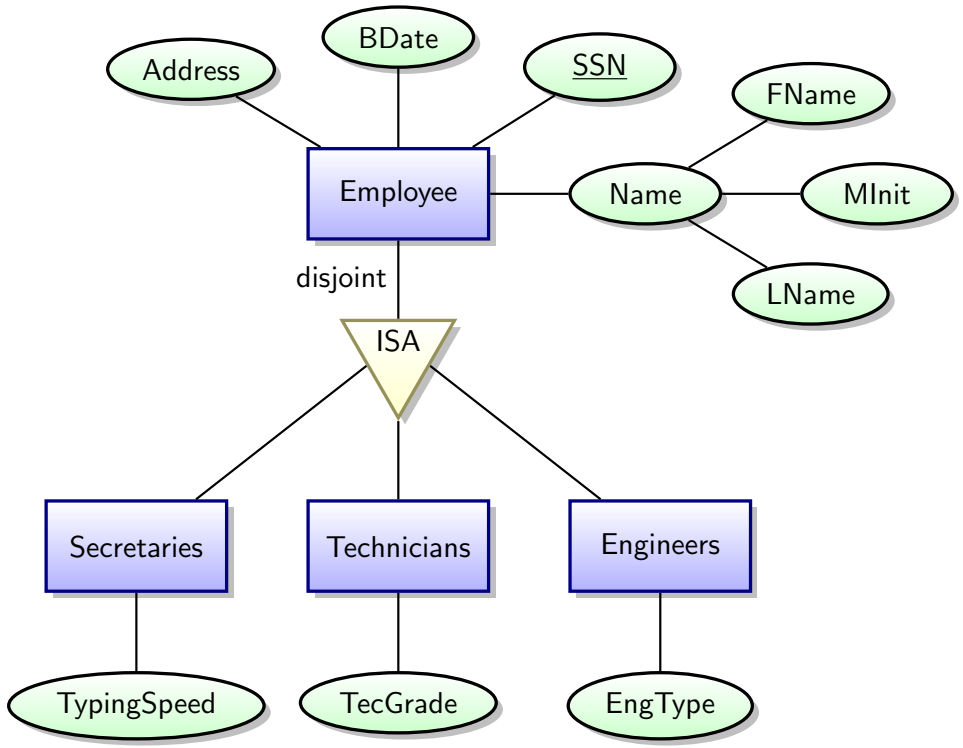
- Two basic constraints can apply to a specialization/generalization:
  - Disjointness Constraint
  - Completeness Constraint
- **Disjointness Constraint**:
  - Specifies that the subclasses of the specialization must be *disjoint*:
    - an entity can be a member of at most one of the subclasses of the specialization
    - Annotate edge in ER diagram with *disjoint*
  - If not disjoint, specialization is *overlapping*:
    - that is the same entity may be a member of more than one subclass of the specialization
    - Annotate edge in ER diagram with *overlapping* (or no annotation)

Constraints on Specialization/2

- **Completeness Constraint**:
  - *Total* specifies that every entity in the superclass must be a member of some subclass in the specialization/generalization
  - Shown in ER diagrams by a **double line**
  - *Partial* allows an entity not to belong to any of the subclasses
  - Shown in ER diagrams by a single line

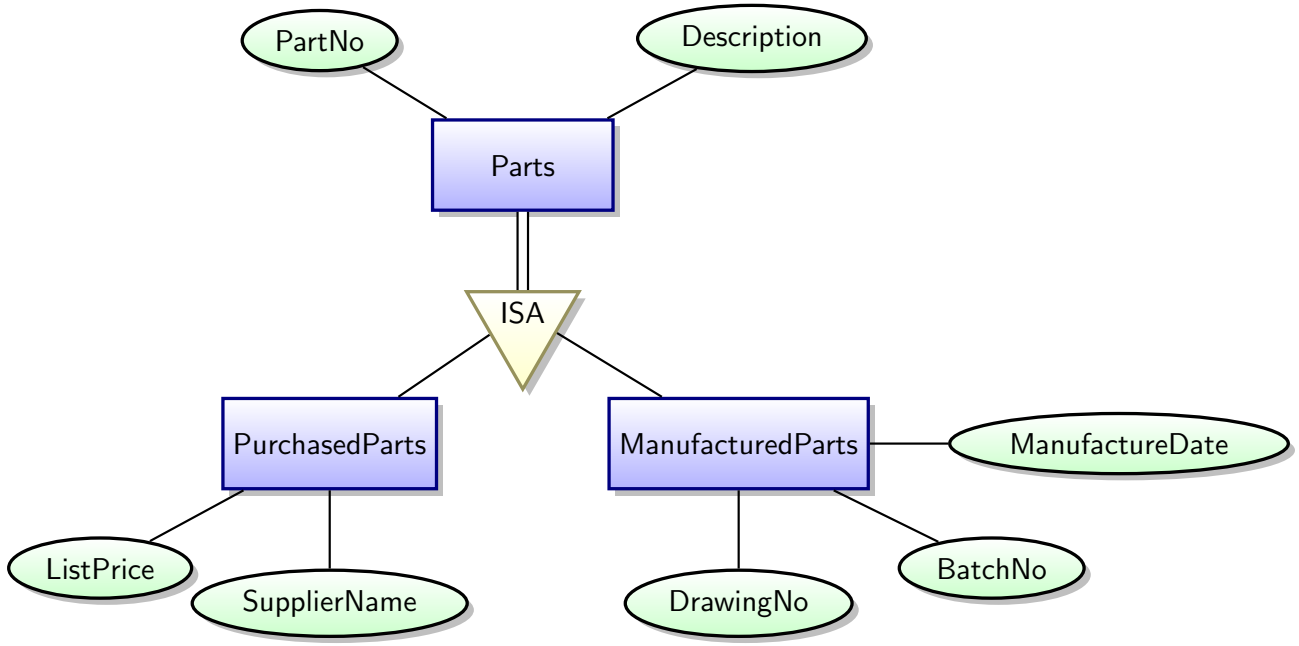
- Hence, we have four types of specialization/generalization:
  - Disjoint, total
  - Disjoint, partial
  - Overlapping, total
  - Overlapping, partial

Example of Disjoint Partial Specialization



- An employee my be neither a secretary nor a technician nor an engineer (partial specialization).
- An employee cannot be a secretary and technician or secretary and engineer or technician and engineer (disjoint specialization).

Example of Overlapping Total Specialization



- Every part must either be a manufactured part or a purchased part (total specialization).
- A part may be a manufactured part and a purchased part (overlapping specialization).

Review 6.7

Assume employees work on projects. Employees who work on projects are allowed to use machines. Use an ER diagram to model this.

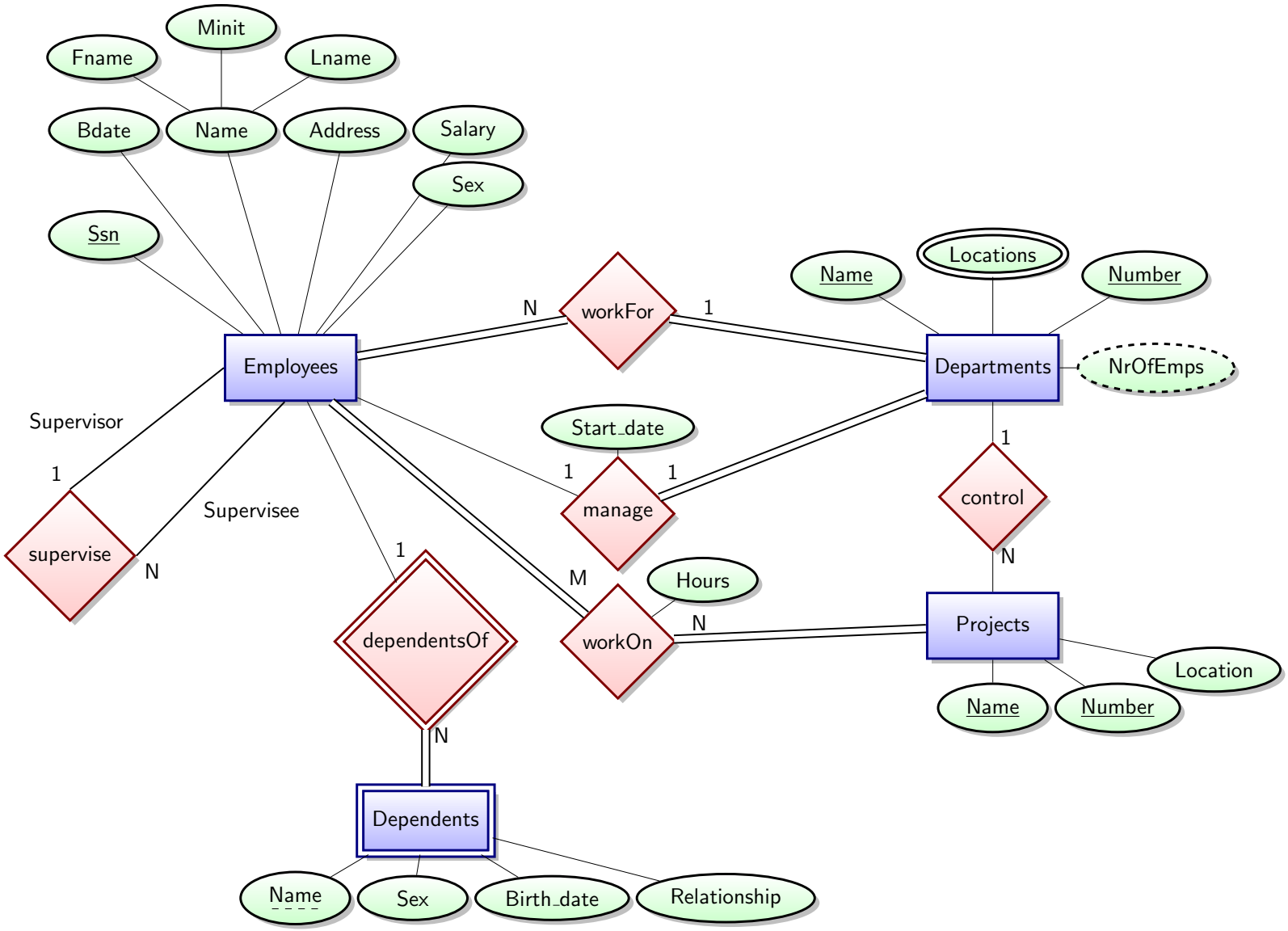
ER-to-Relational Mapping Algorithm

- mapping an ER Model to a Relational Model

ER-to-Relational Mapping Algorithm/1

- Algorithm to map a conceptual database design to a relational database design.
- ER-to-Relational Mapping Algorithm
  - Step 1: Mapping of Regular Entity Types
  - Step 2: Mapping of Weak Entity Types
  - Step 3: Mapping of Binary 1:1 Relation 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
  - Step 8: Mapping Specialization or Generalization

The ER schema for the COMPANY database

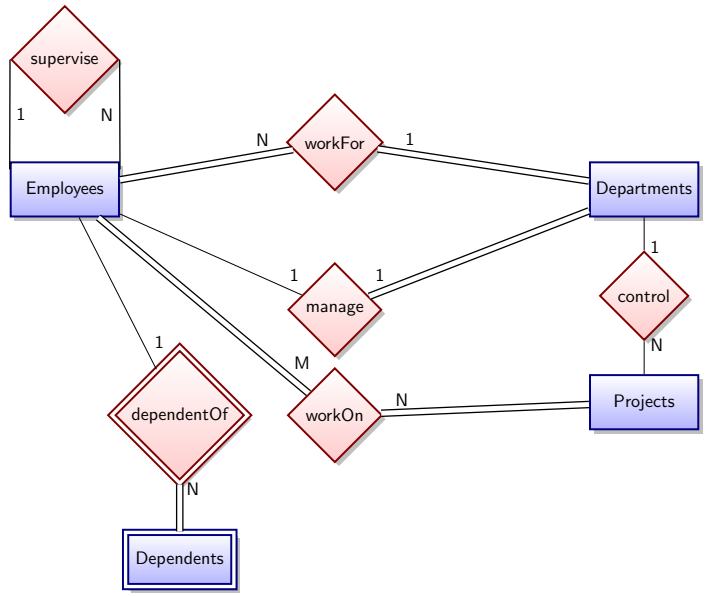


ER-to-Relational Mapping Algorithm/2

- **Step 1: Mapping of Regular Entity Types.**
  - For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E.
  - A composite attribute is flattened into a set of simple attributes.
  - Choose one of the key attributes of E as the primary key for R.
  - If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.

ER-to-Relational Mapping Algorithm/3

- **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.

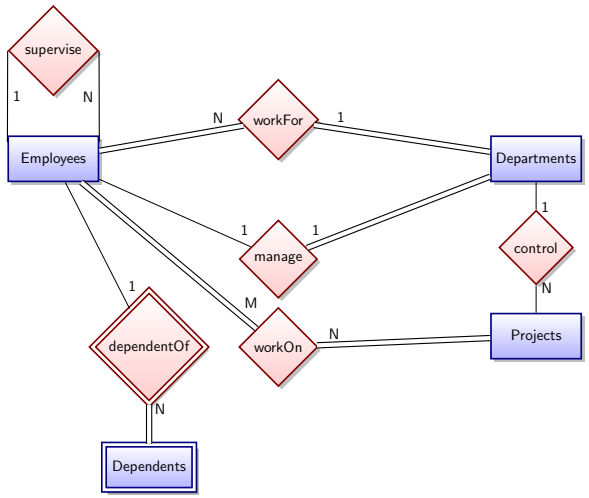


EMPLOYEE(FName, MInit, LName, SSN, BDate, Address, Sex, Salary)
DEPARTMENT(DName, DNumber)
PROJECT(PName, PNumber, PLocation)

ER-to-Relational Mapping Algorithm/4

- **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.
  - Also, include as foreign key attributes of R the primary key attributes of the relations that correspond to the owner entity types.
  - The primary key of R is the *combination of* the primary keys of the owners and the partial key of the weak entity type W, if any.

ER-to-Relational Mapping Algorithm/5



- **Example:** Create 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, DepName} because DepName is the partial key of DEPENDENT.

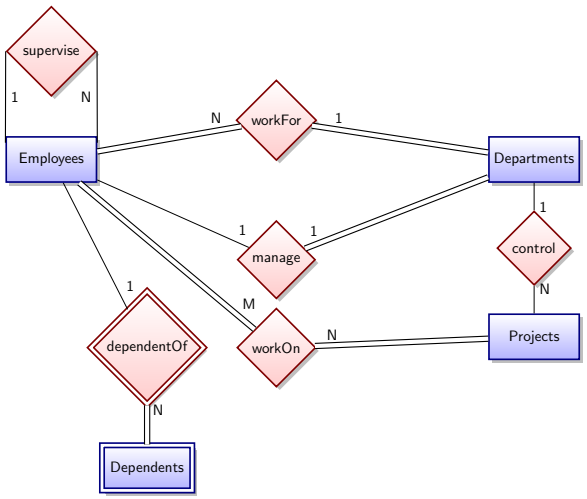
EMPLOYEE(FName, MInit, LName, SSN, BDate, Address, Sex, Salary)
DEPARTMENT(DName, DNumber)
PROJECT(PName, PNumber, PLocation)
DEPENDENT(ESSN, DepName, Sex, BDate, Relationship)

ER-to-Relational Mapping Algorithm/6

- **Step 3: Mapping of Binary 1:1 Relation Types**
  - For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.
- There are three possible approaches:
  1. **Foreign Key approach:** Choose one of the relations, say S, and include as foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S.
  2. **Merged relation option:** An alternate mapping of a 1:1 relationship type is possible by merging the two entity types and the relationship into a single relation. This may be appropriate when both participations are total.
  3. **Cross-reference or relationship relation option:** The third alternative is to set up a third relation R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types.

ER-to-Relational Mapping Algorithm/7

- **Example**: The 1:1 relation MANAGES is mapped by choosing the participating entity type DEPARTMENT to serve in the role of S, because its participation in the manage relationship type is total.



EMPLOYEE(FName, MInit, LName, SSN, BDate, Address, Sex, Salary)
DEPARTMENT(DName, DNumber, MgrSSN, MgrStartDate)
PROJECT(PName, PNumber, PLocation)
DEPENDENT(ESSN, DepName, Sex, BDate, Relationship)

Review 6.8

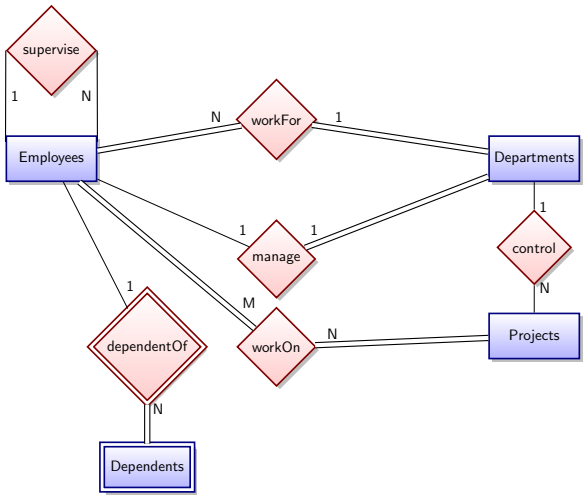
Illustrate the problems that occur if the 1:1 relationship manage is mapped through a foreign key in relation EMPLOYEE.

ER-to-Relational Mapping Algorithm/8

- **Step 4: Mapping of Binary 1:N Relationship Types**
  - For each binary 1:N relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R. S is the N-side.
  - Include as foreign key in S the primary key of relation T that represents the other entity type participating in R.
  - Include simple attributes of the 1:N relation type as attributes of S.

ER-to-Relational Mapping Algorithm/9

- **Example:** 1:N relationship types workFor, control and supervise.
  - For example for workFor we include the primary key DNumber of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it DNO.

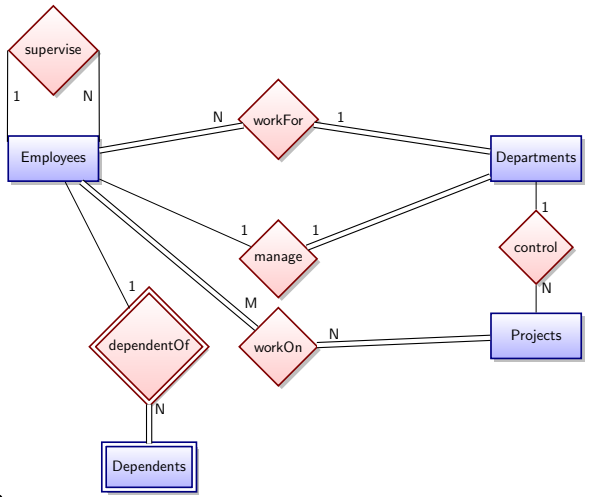


EMPLOYEE(FName, MInit, LName, SSN, BDate, Address, Sex, Salary, SuperSSN, DNo)
DEPARTMENT(DName, DNumber, MgrSSN, MgrStartDate)
PROJECT(PName, PNumber, PLocation, DNum)
DEPENDENT(ESSN, DepName, Sex, BDate, Relationship)

ER-to-Relational Mapping Algorithm/10

- **Step 5: Mapping of Binary M:N Relationship Types.**
  - For each regular binary M:N relationship type R, *create a new relation* S to represent R.
  - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; *their combination will form the primary key* of S.
  - Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S.

ER-to-Relational Mapping Algorithm/11



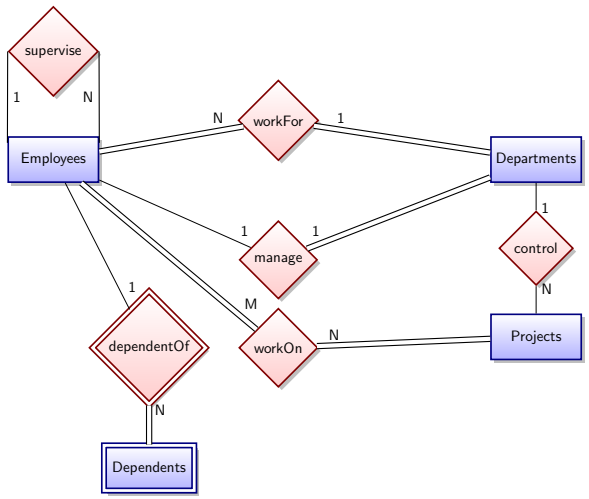
- **Example:** The M:N relationship type WORKSON from the ER diagram is mapped by creating a relation WORKSON in the relational database schema.
  - The primary keys of the PROJECT and EMPLOYEE relations are included as foreign keys in WORKSON and renamed PNo and ESSN, respectively.
  - Attribute Hours in WORKSON represents the Hours attribute of the relation type. The primary key of the WORKSON relation is the combination of the foreign key attributes ESSN, PNo.

EMPLOYEE(FName, MInit, LName, SSN, BDate, Address, Sex, Salary, SuperSSN, DNo)
DEPARTMENT(DName, DNumber, MgrSSN, MgrStartDate)
PROJECT(PName, PNumber, PLocation, DNum)
DEPENDENT(ESSN, DepName, Sex, BDate, Relationship)
WORKSON(ESSN, PNo, Hours)

ER-to-Relational Mapping Algorithm/12

- **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 of the relation that represents the entity 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.

ER-to-Relational Mapping Algorithm/13



- **Example:** The relation DEPTLOC 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.

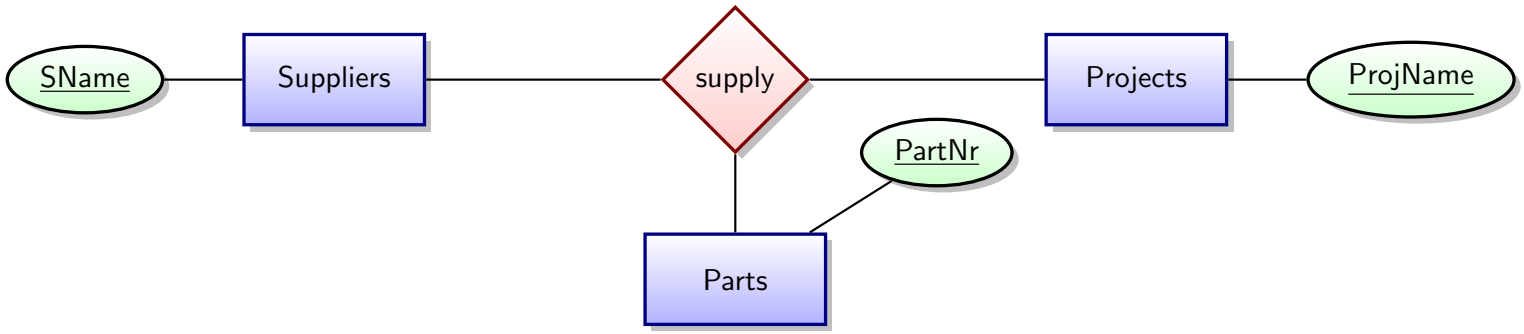
EMPLOYEE(FName, MInit, LName, SSN, BDate, Address, Sex, Salary, SuperSSN, DNo)
DEPARTMENT(DName, DNumber, MgrSSN, MgrStartDate)
PROJECT(PName, PNumber, PLocation, DNum)
DEPENDENT(ESSN, DepName, Sex, BDate, Relationship)
WORKSON(ESSN, PNo, Hours)
DEPTLOC(DNumber, DLocation)

ER-to-Relational Mapping Algorithm/14

- **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.

ER-to-Relational Mapping Algorithm/15

- **Example:** The relationship type SUPPLY



- 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

SUPPLIER(SName, ...)
PROJECT(ProjName, ...)
PART(PartNo, ...)
SUPPLY(SName, ProjName, PartNo, Quantity)

ER-to-Relational Mapping Algorithm/16

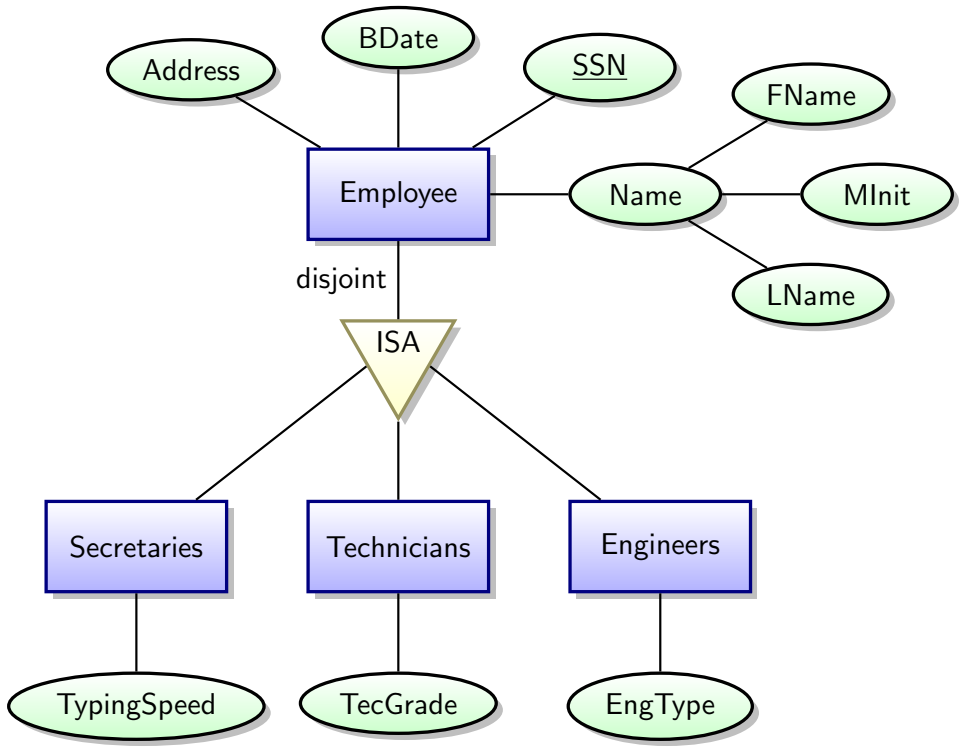
- **Step8: Options for Mapping Specialization or Generalization.**
  - Convert each specialization with m subclasses $S_1, S_2, \ldots, S_m$ and superclass C, where the attributes of C are $k, a_1, \ldots, a_n$ and k is the primary key, into relational schemas using one of the four following options:
    - Option 8A: Multiple relations for superclass and subclasses
    - Option 8B: Multiple relations for subclass relations only
    - Option 8C: Single relation with one type attribute
    - Option 8D: Single relation with multiple type attributes

ER-to-Relational Mapping Algorithm/17

- **Option 8A: Multiple relations for 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 attributes $Attrs(Li) = \{k\} \cup \{attributes\ of\ S_i\}$ and $PK(L_i) = k$.
  - This option works for any specialization (total or partial, disjoint of overlapping).

ER-to-Relational Mapping Algorithm/18

- **Example:** Specialization of EMPLOYEE



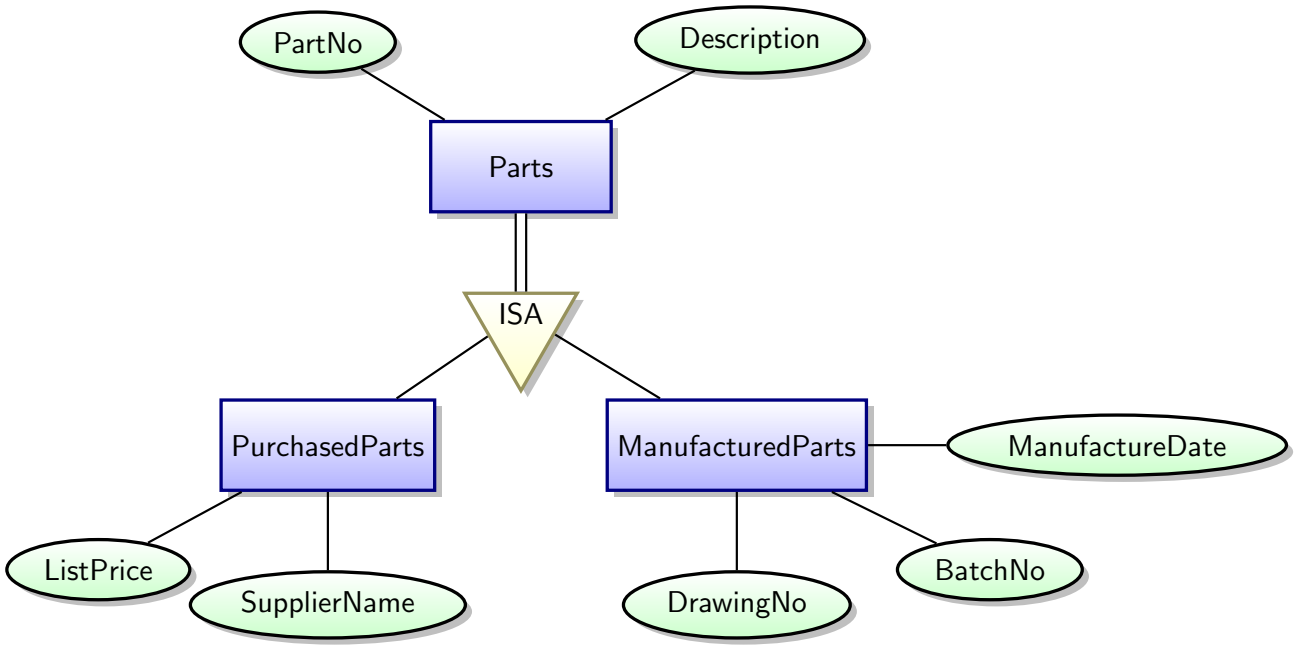
EMPLOYEE(SSN, FName, MInit, LName, BirthDate, Address, JobType)
SECRETARY(SSN, TypingSpeed)
TECHNICIAN(SSN, TGrade)
ENGINEER(SSN, EngType)

ER-to-Relational Mapping Algorithm/19

- **Option 8B: Multiple relations for subclass relations only**
  - Create a relation L_i for each subclass $S_i, 1 < i < m$, with attributes $Attr(L_i) = \{attributes\ \ of\ \ S_i\} \cup \{k, a_1, \ldots, a_n\}$ and $PK(L_i) = k$.
  - This option only works for a specialization whose subclasses are total (i.e., every entity in the superclass must belong to at least one of the subclasses).

ER-to-Relational Mapping Algorithm/20

- **Example:** Specialization of Parts



MANUFACTURED(PartNo,Desc,DrawNo,BatchNo,ManufDate)

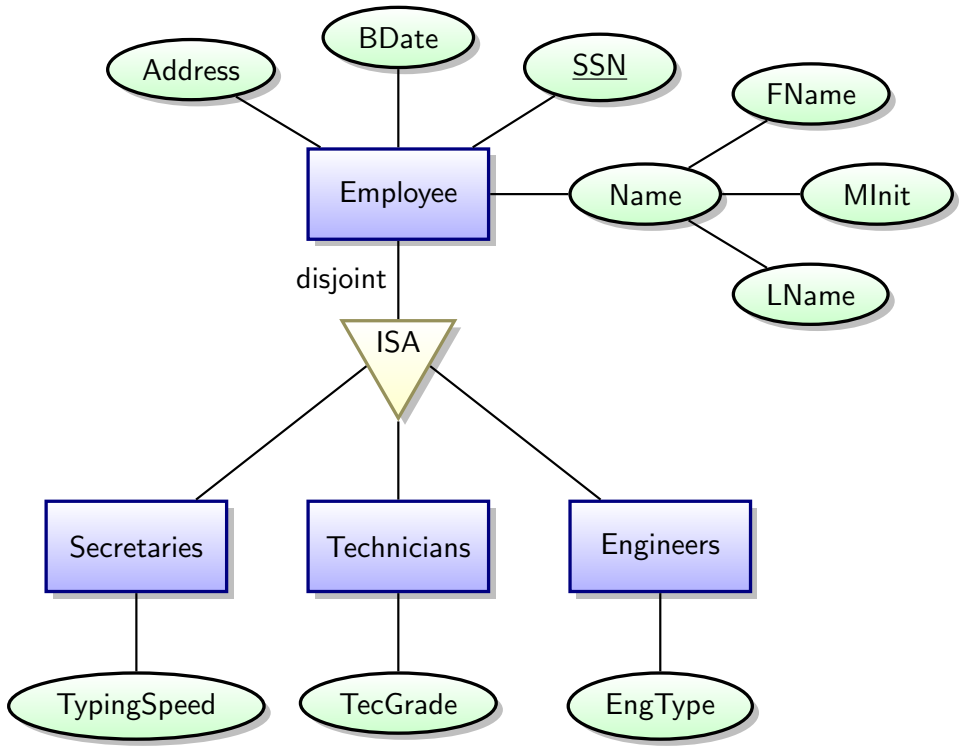
PURCHASED(PartNo,Desc,SuppName,ListPrice)

ER-to-Relational Mapping Algorithm/21

- **Option 8C: Single relation with one type attribute**
  - Create a single relation L with attributes $Attr(L) = \{k, a_1, \ldots, a_n, t\} \cup \{attributes\ of\ S_1\} \cup \{attributes\ of\ S_m\}$ and $PK(L) = k$.
  - The attribute t is called a type attribute that indicates the subclass to which each tuple belongs.
  - This option only works for a specialization whose subclasses are disjoint and might generate many null values for attributes of subclasses.

ER-to-Relational Mapping Algorithm/22

- **Example**: Use the JobType as a type attribute in EMPLOYEE to distinguish between secretary, technician, and engineer.



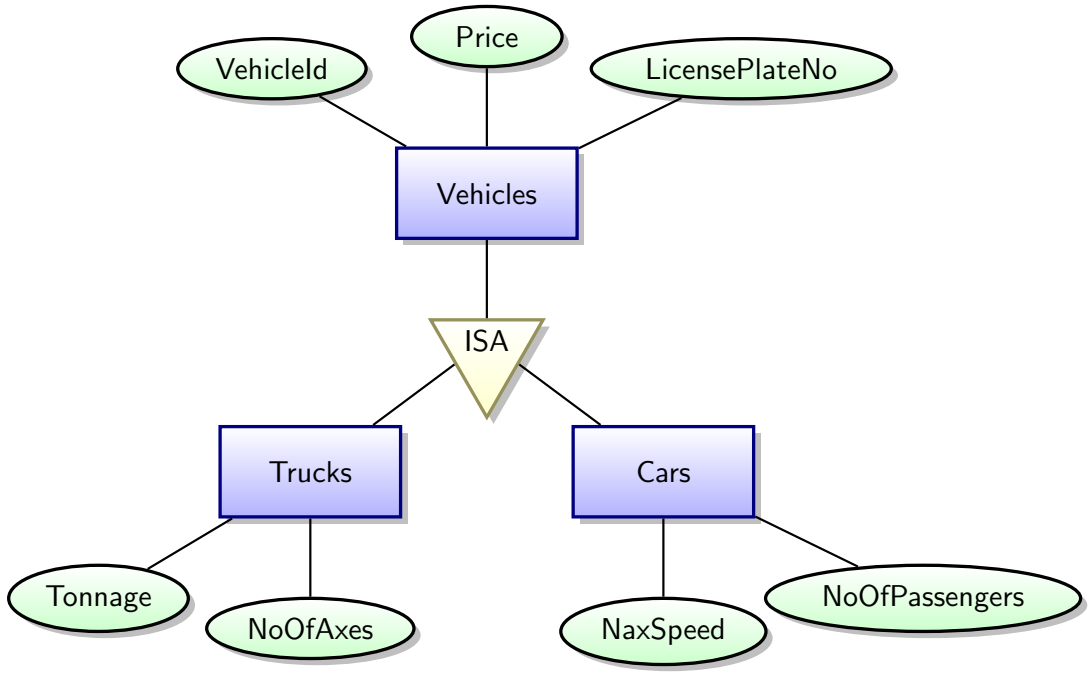
EMPLOYEE(SSN,FName,MInit,LName,...,JobType,TypingSpeed,TGrade,EngType)

ER-to-Relational Mapping Algorithm/23

- **Option 8D: Single relation with multiple type attributes**
  - Create a single relation L with attributes $Attr(L) = \{k, a_1, \ldots, a_n, t_1, ..., t_m\} \cup \{attributes\ of\ S_1\} \cup \{attributes\ of\ S_m\}$ and $PK(L) = k$.
  - Each t_i is a Boolean attribute that indicates whether a tuple belongs to subclass S_i.
  - This option works for a specialization whose subclasses are overlapping (but will also work for disjoint specialization).

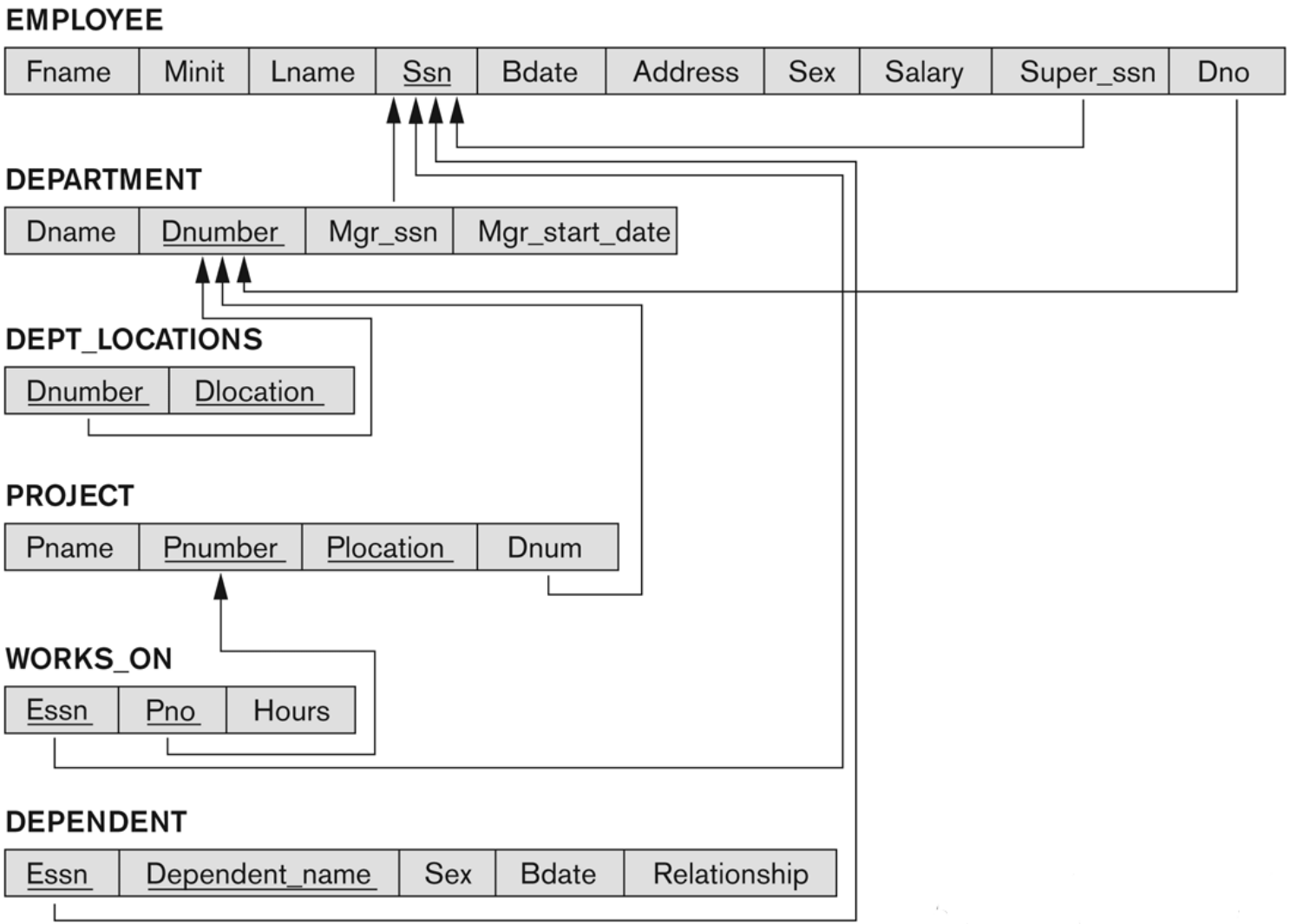
ER-to-Relational Mapping Algorithm/24

- **Example**:



VEHICLE(
VehicleID,Price,LicensePlateNo,
CarFlag,MaxSpeed,NoPassengers,
TruckFlag,NoAxes,Tonnage)

ER-to-Relational Mapping Algorithm/27



Summary of Mapping Constructs and 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
Key attribute	Primary (or secondary) key

Summary/1

- ER model concepts:
  - entity type, entity set, entity
  - attribute, attribute value
  - relationship type, relationship set, relationship
  - structural constraints: cardinality, participation
  - subclasses and superclasses
- During the conceptual modeling process a number of **clarifying decisions/assumptions** must be made
- It is important to make all necessary decisions (even if alternatives exist) and go on. **Make decisions explicit** (i.e., write them down).
- There is not a unique ER model.
- In order to clarify the semantics it can hep to consider relation instances

Summary/2

- ER-to-Relational Mapping Algorithm
  - Step 1: Mapping of Regular Entity Types
  - Step 2: Mapping of Weak Entity Types
  - Step 3: Mapping of Binary 1:1 Relation 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.
  - Step 8: Mapping Specialization or Generalization.
- The **mapping algorithm is mechanical** and without conceptual difficulty. Make sure you know how to apply it.