Databases

Lecture 5 - Constraints & Notations

Emily Lucia Antosch

HAW Hamburg

10.03.2025

Contents

1. Introduction

1.1 Where are we right now?

- Last time, we looked at how we can use relationships in the database design stages
- Today, we'll be discussing
  - how we can expand on that knowledge
  - what multivalued and derived attributes are
  - the syntax for creating constraints more in-depth.

1.1 Where are we right now?

1. Introduction
2. Basics
3. SQL
4. Entity-Relationship-Model
5. **Relationships**
6. **Constraints**
7. More SQL
8. Subqueries & Views
9. Transactions
10. Database Applications
11. Integrity, Trigger & Security

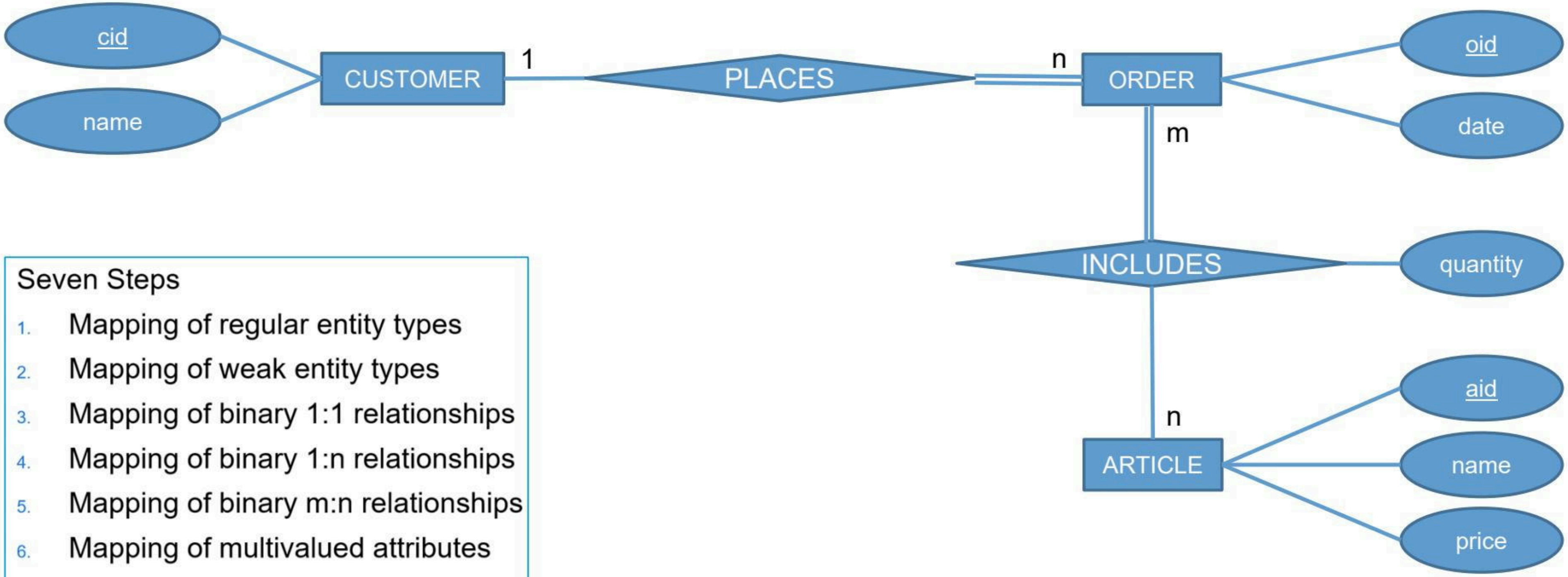
1.2 What is the goal of this chapter?

- At the end of this lesson, you should be able to
  - create constraints based on your logical and conceptual design
  - use domains
  - and pivot the type of constraints to fit your need.

2. Relationships

2.1 ERM

Convert ERD to RM

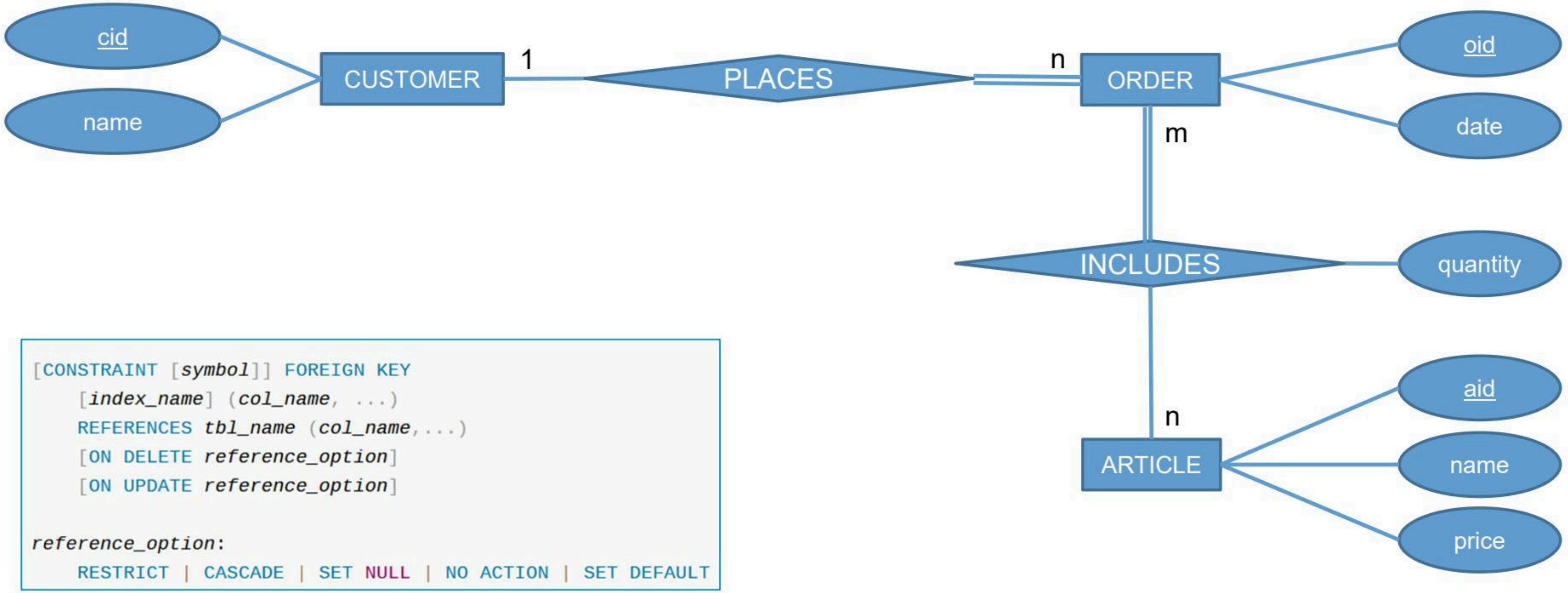


Seven Steps

1. Mapping of regular entity types
2. Mapping of weak entity types
3. Mapping of binary 1:1 relationships
4. Mapping of binary 1:n relationships
5. Mapping of binary m:n relationships
6. Mapping of multivalued attributes
7. Mapping of n-ary relationships

2.1 ERM

Convert ERD to RM: SQL



```
[CONSTRAINT [symbol]] FOREIGN KEY
    [index_name] (col_name, ...)
    REFERENCES tbl_name (col_name,...)
    [ON DELETE reference_option]
    [ON UPDATE reference_option]

reference_option:
    RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT
```

2.1 ERM

Weak Entity Types

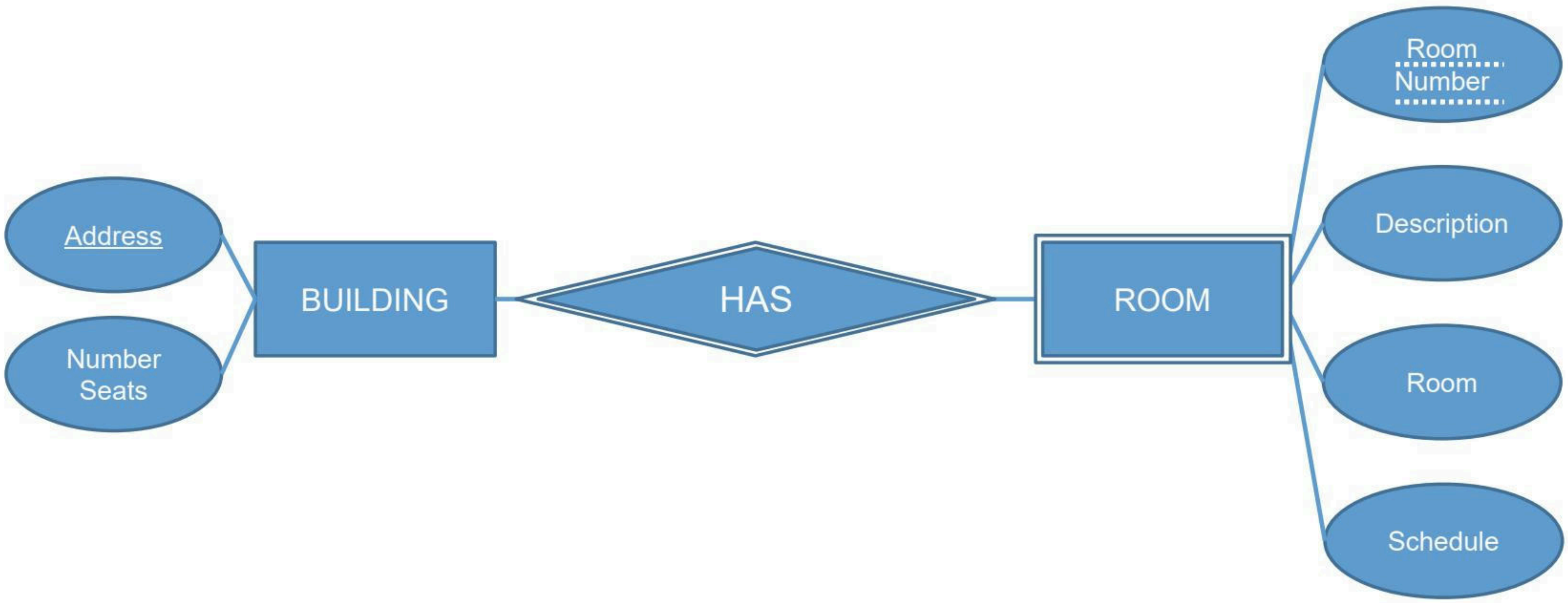
- An entity type without a key attribute is called a **weak entity type**
- Weak entities are identified by being related to specific entities from another entity type in combination with one of their attribute values;
- This other entity type is called **the identifying or owner entity type**, and the relationship type that relates a weak entity type to its owner **the identifying relationship**
- A weak entity type always has a total participation constraint (existence dependency) with respect to its identifying relationship

2.1 ERM

- Represented by double rectangles and by having their identifying relationship placed in double diamonds
  - The partial key attribute is underlined with a dashed or dotted line
- Example: Room vs. Building
  - Need for identify room: Room number and Building number!

2.1 ERM

Weak Entity Types



2.1 ERM

Weak Entity Types

- The company is organized into departments
- Each department has a unique name, a unique number, a manager (employee) with start date, and several locations
- A department controls a number of projects, each with unique name, unique number, single location
- We store each employee's name, ssn, address, salary, sex, birthdate
- An employee is assigned to one department, but may work on several projects, also from other departments
- We keep track of the hours per week per project

2.1 ERM

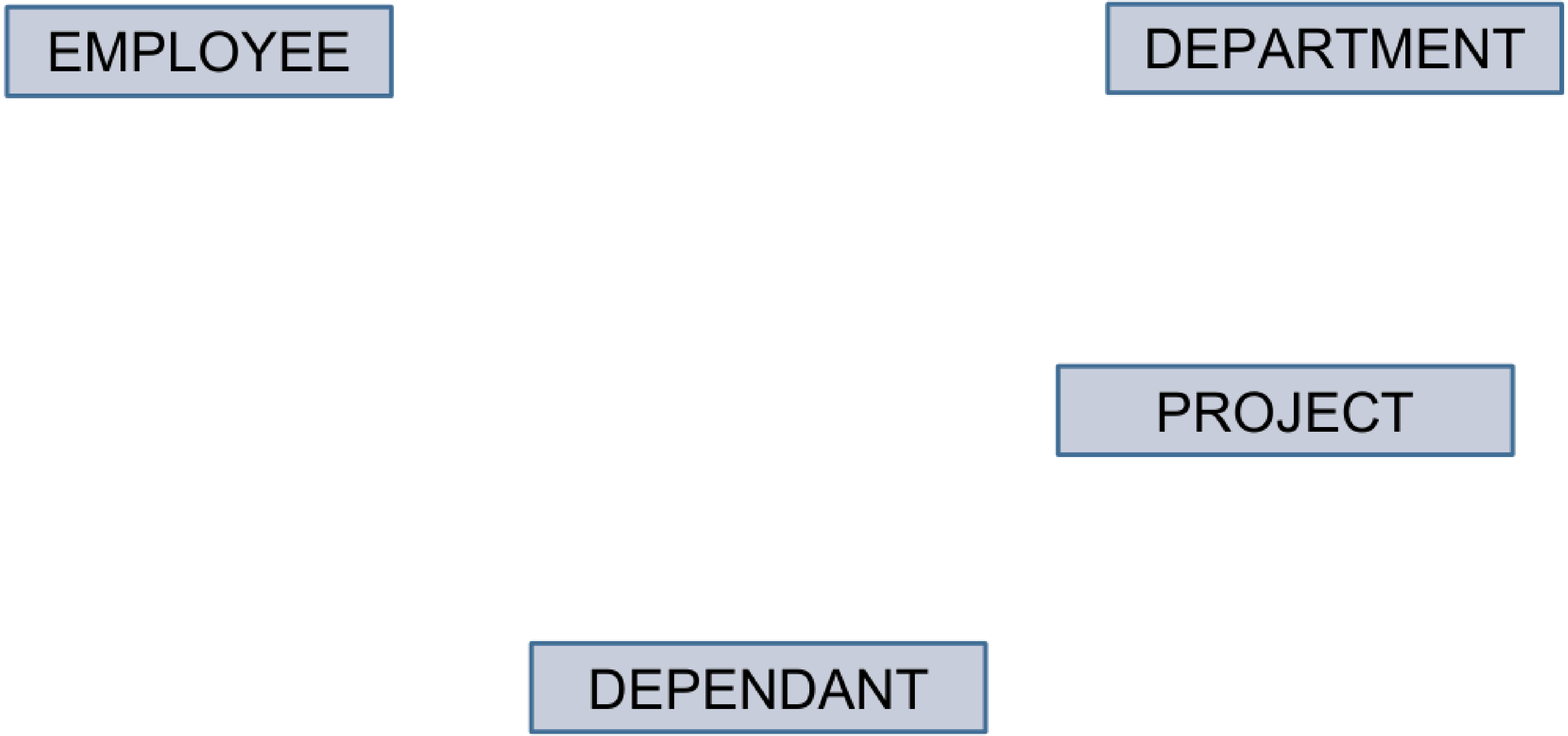
- We also keep track of the supervisor
- We want to keep track of each employee’s dependents for insurance purposes, namely first name, sex, birth date, and relationship to employee. Source: Elmasri, Fundamentals

Weak Entity Types

? Question

What could be a weak entity type?

2.1 ERM



EMPLOYEE

DEPARTMENT

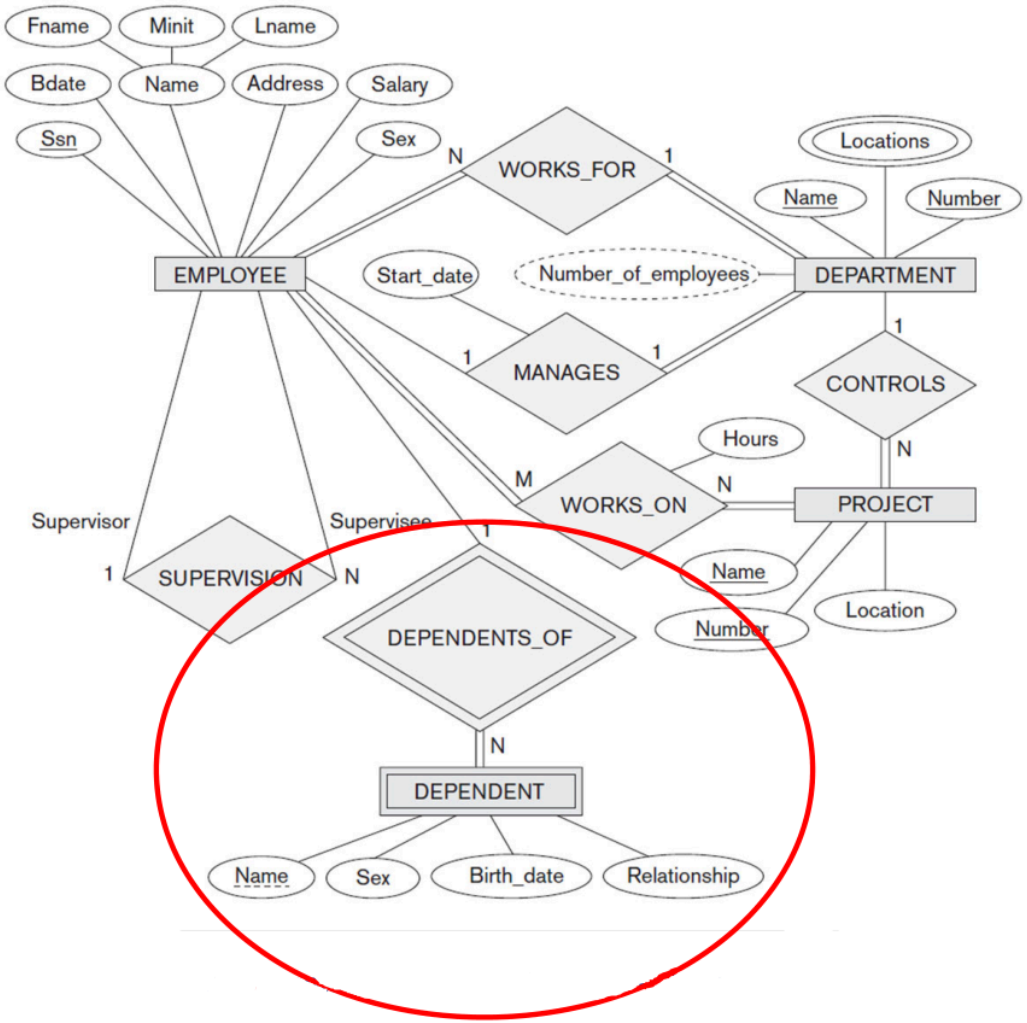
PROJECT

DEPENDANT

2.1 ERM

Weak Entity Types

2.1 ERM


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
MANAGES
1
1
CONTROLS
Hours
N
M
WORKS_ON
N
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
Name
DEPENDENTS_OF
Number
Location
N
DEPENDENT
Name
Sex
Birth_date
Relationship

2.1 ERM

Multivalued Attributes

- Single-valued vs multivalued attributes
  - Single-valued: Most attributes have a single value for a particular entity
  - Multivalued: In some cases, an attribute can have a set of values for the same entity
  - Multivalued attributes are displayed in double ovals

2.1 ERM

Multivalued Attributes

? Question

What could be a multivalued attribute?

- The company is organized into departments
- Each department has a unique name, a unique number, a manager (employee) with start date, and several locations
- A department controls a number of projects, each with unique name, unique number, single location

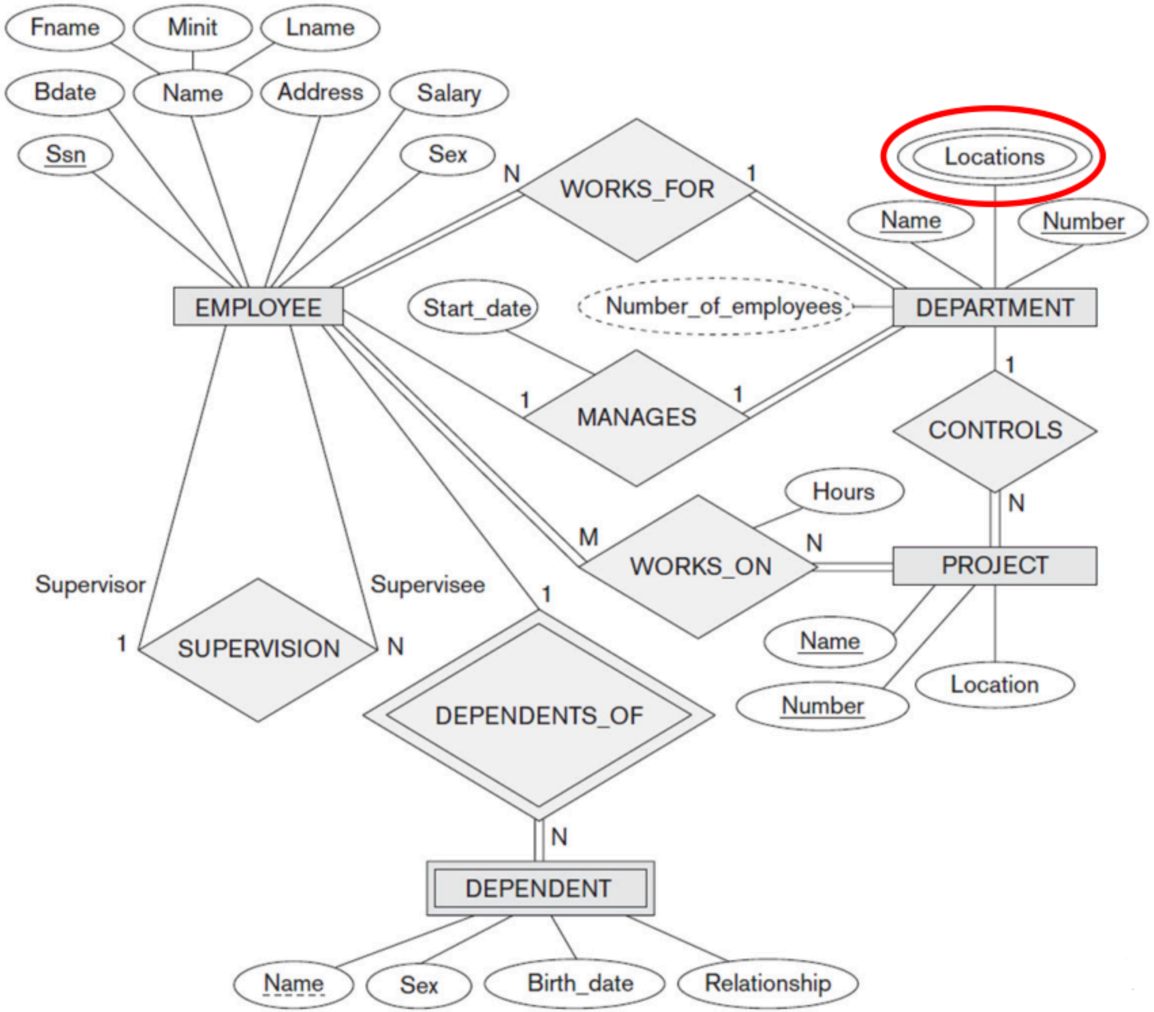
2.1 ERM

- We store each employee’s name, ssn, address, salary, sex, birthdate
- An employee is assigned to one department, but may work on several projects, also from other departments
- We keep track of the hours per week per project
- We also keep track of the supervisor
- We want to keep track of each employee’s dependents for insurance purposes, namely first name, sex, birth date, and relationship to employee. Source: Elmasri, Fundamentals

2.1 ERM

Multivalued Attributes

2.1 ERM


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
MANAGES
1
1
CONTROLS
Hours
N
M
WORKS_ON
N
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
DEPENDENTS_OF
Name
Number
Location
N
DEPENDENT
Name
Sex
Birth_date
Relationship

2.1 ERM

Complex Attributes

- Composite and multivalued attributes can be nested arbitrarily
- The combination of composite and multivalued attributes is called **Complex Attribute**

Example

Person can have more than one residence and each residence can have a single address and multiple phones

2.1 ERM

Derived Attributes

- Stored vs. derived attributes
  - Some attribute values can be derived from related entities
  - For example: Age can by derived from birthdate

2.1 ERM

Derived Attributes

? Question

Can you think of a derived attribute?

- The company is organized into departments
- Each department has a unique name, a unique number, a manager (employee) with start date, and several locations
- A department controls a number of projects, each with unique name, unique number, single location

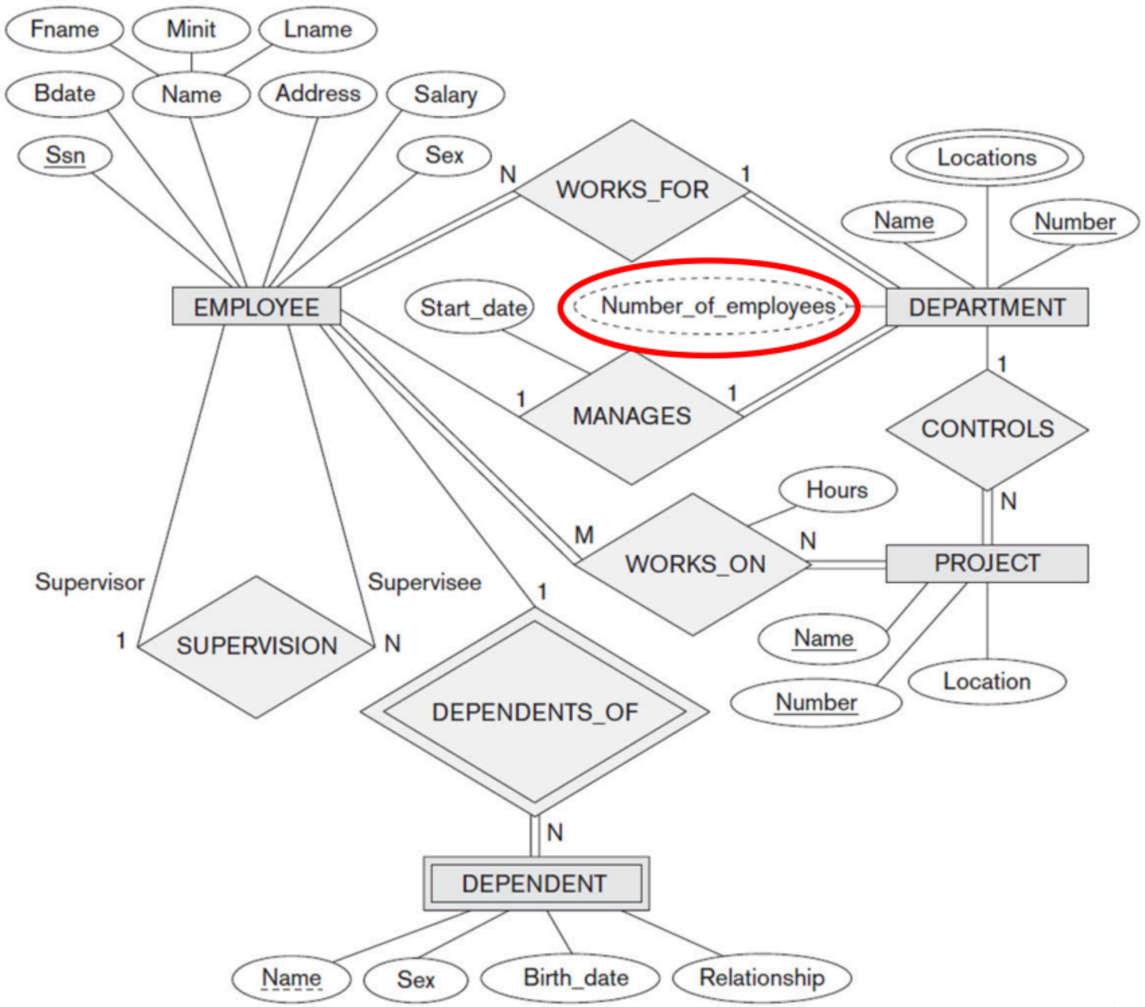
2.1 ERM

- We store each employee’s name, ssn, address, salary, sex, birthdate
- An employee is assigned to one department, but may work on several projects, also from other departments
- We keep track of the hours per week per project
- We also keep track of the supervisor
- We want to keep track of each employee’s dependents for insurance purposes, namely first name, sex, birth date, and relationship to employee. Source: Elmasri, Fundamentals

2.1 ERM

Derived Attributes

2.1 ERM


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
MANAGES
1
1
CONTROLS
Hours
N
M
WORKS_ON
N
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
Name
Location
Number
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
Birth_date
Relationship

2.1 ERM

Mapping of ERM

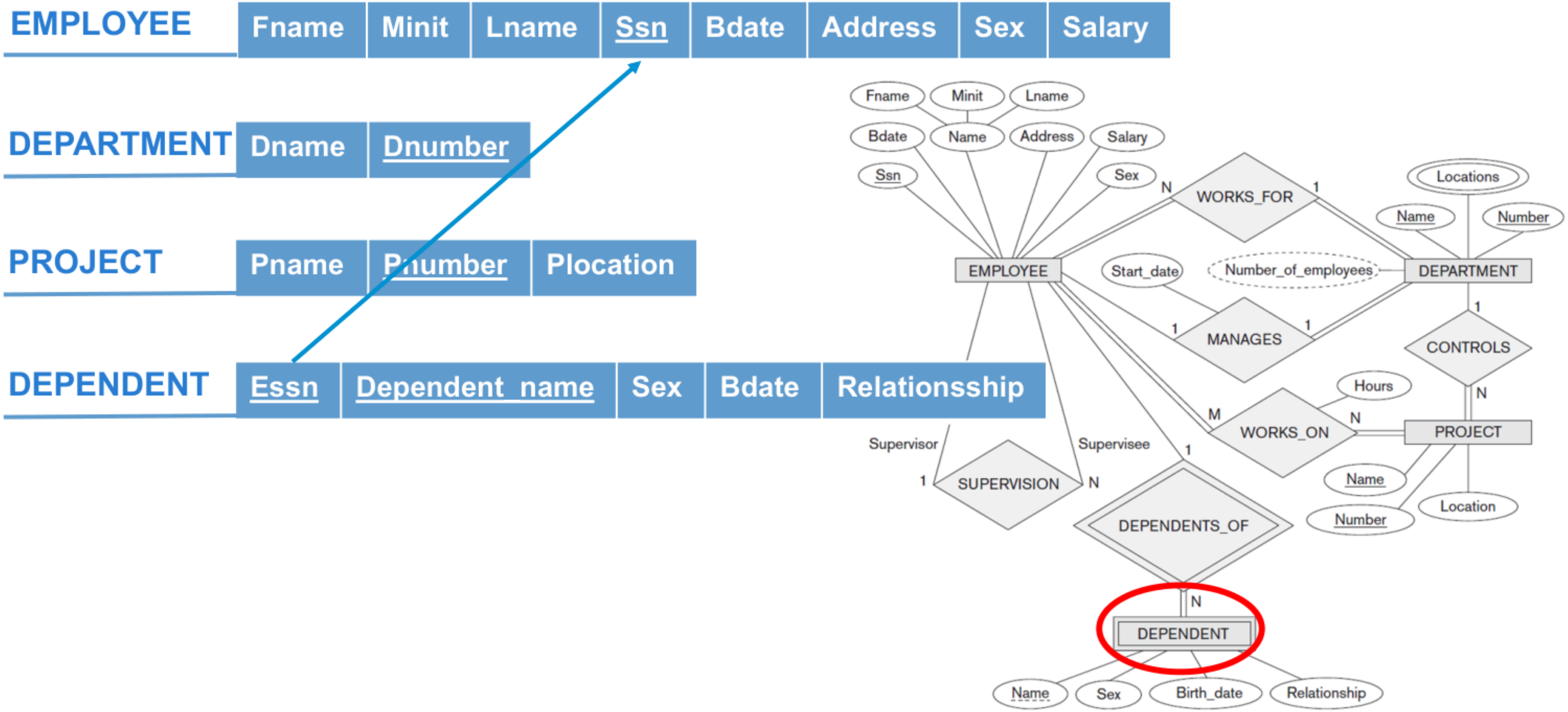
- Seven Steps
  1. Mapping of regular entity types ✓
  2. Mapping of weak entity types
  3. Mapping of binary 1:1 relationships ✓
  4. Mapping of binary 1:n relationships ✓
  5. Mapping of binary m:n relationships ✓
  6. Mapping of multivalued attributes
  7. Mapping of n-ary relationships

2.1 ERM

Mapping of ERM: 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); this takes care of mapping the identifying relationship type of W
- 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

Mapping of ERM: Weak Entity Types



EMPLOYEE	Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary

DEPARTMENT	Dname	Dnumber

PROJECT	Pname	Pnumber	Plocation

DEPENDENT	Essn	Dependent_name	Sex	Bdate	Relationsship

2.1 ERM

Info

- The attribute `SSN` is renamed to `ESSN`, although this is not necessary.
- The primary key is the combination `{ESSN, DEPENDENT_NAME}`

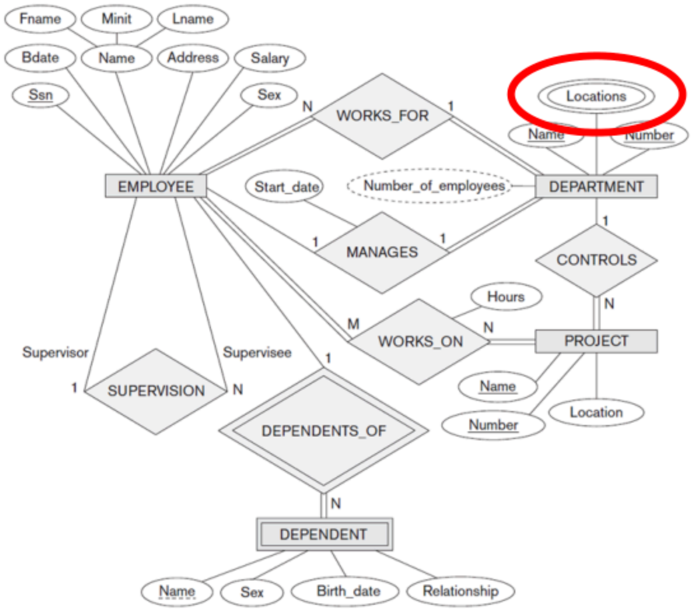
Mapping of ERM: Multivalued Attributes

- 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 a multivalued attribute
- The primary key of R is the combination of A and K
- If the multivalued attribute is composite, we include its simple components

2.1 ERM

Mapping of ERM: Multivalued Attributes

DEPT_ LOCATIONS	Dnumber	Dlocation



2.1 ERM

- Attribute `Dlocation` represents the multivalued attribute `LOCATIONS` of `DEPARTMENT`
- Attribute `Dnumber` represents the primary key of `DEPARTMENT`
- The primary key of `DEPT_LOCATIONS` is the combination of `{Dnumber, Dlocation}`
- A separate tuple will exist in `DEPT_LOCATIONS` for each location that a department has

2.1 ERM

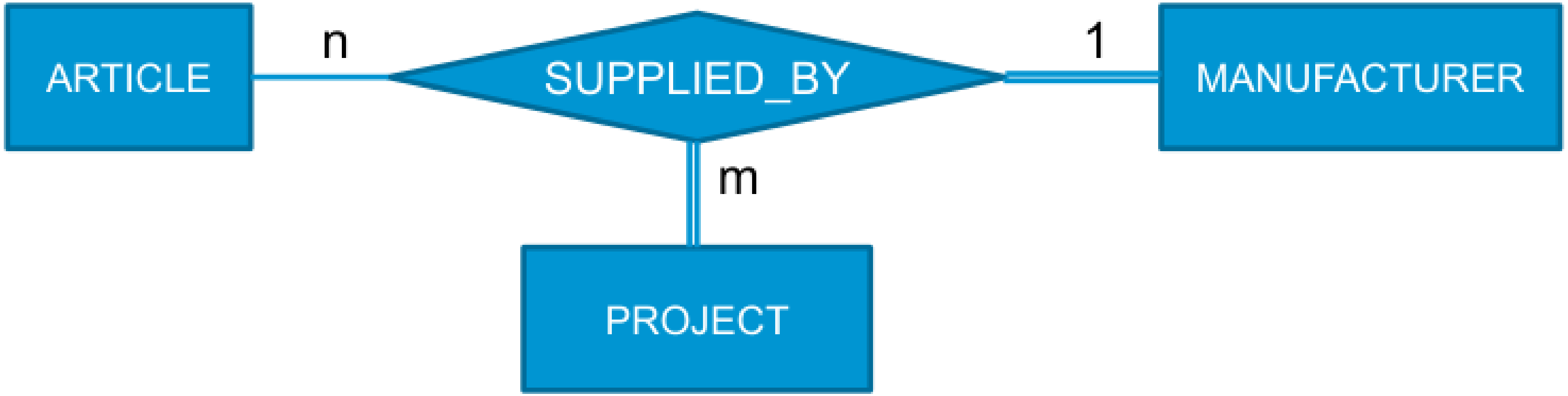
Mapping of ERM

- Seven Steps
  1. Mapping of regular entity types ✓
  2. Mapping of weak entity types ✓
  3. Mapping of binary 1:1 relationships ✓
  4. Mapping of binary 1:n relationships ✓
  5. Mapping of binary m:n relationships ✓
  6. Mapping of multivalued attributes ✓
  7. Mapping of n-ary relationships

Ternary Relationship Types

- Example:
  - Manufacturers supply items for projects.
  - A manufacturer must supply at least one item.
  - An article from in-house production does not have to be supplied for a project but can be supplied for many projects.
  - A project uses at least one item.
  - An item is supplied by only one manufacturer for a project.

2.1 ERM



2.1 ERM

Ternary Relationship Types

? Question

Cardinality: Can an entity of entity type A and an entity of entity type B be related to multiple entities of entity type C?

? Question

Participation: Must an entity type A be related to at least one entity type B and one entity type C?

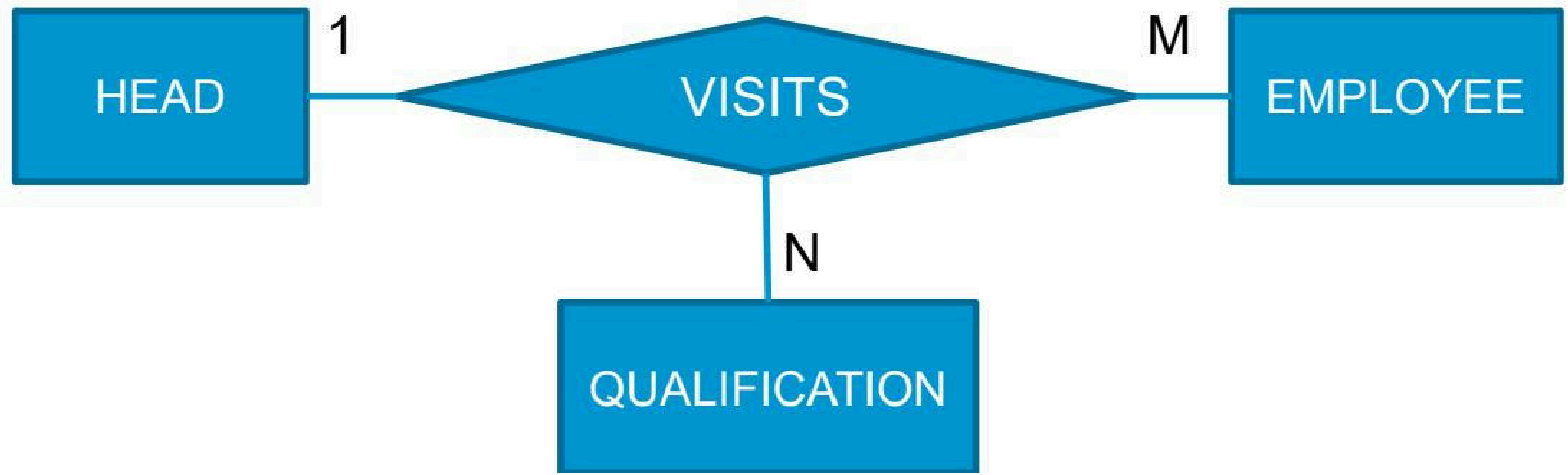
2.1 ERM

Ternary Relationship Types

- Example:
  - To prevent students from concentrating on one professor, they may only work with one professor on one seminar topic.
  - In addition, a student can only work on a seminar topic with one professor.
  - However, a professor may assign a seminar topic more than once.
  - Students must attend seminars, but seminar topics do not have to be chosen.

2.1 ERM

Ternary Relationship Types

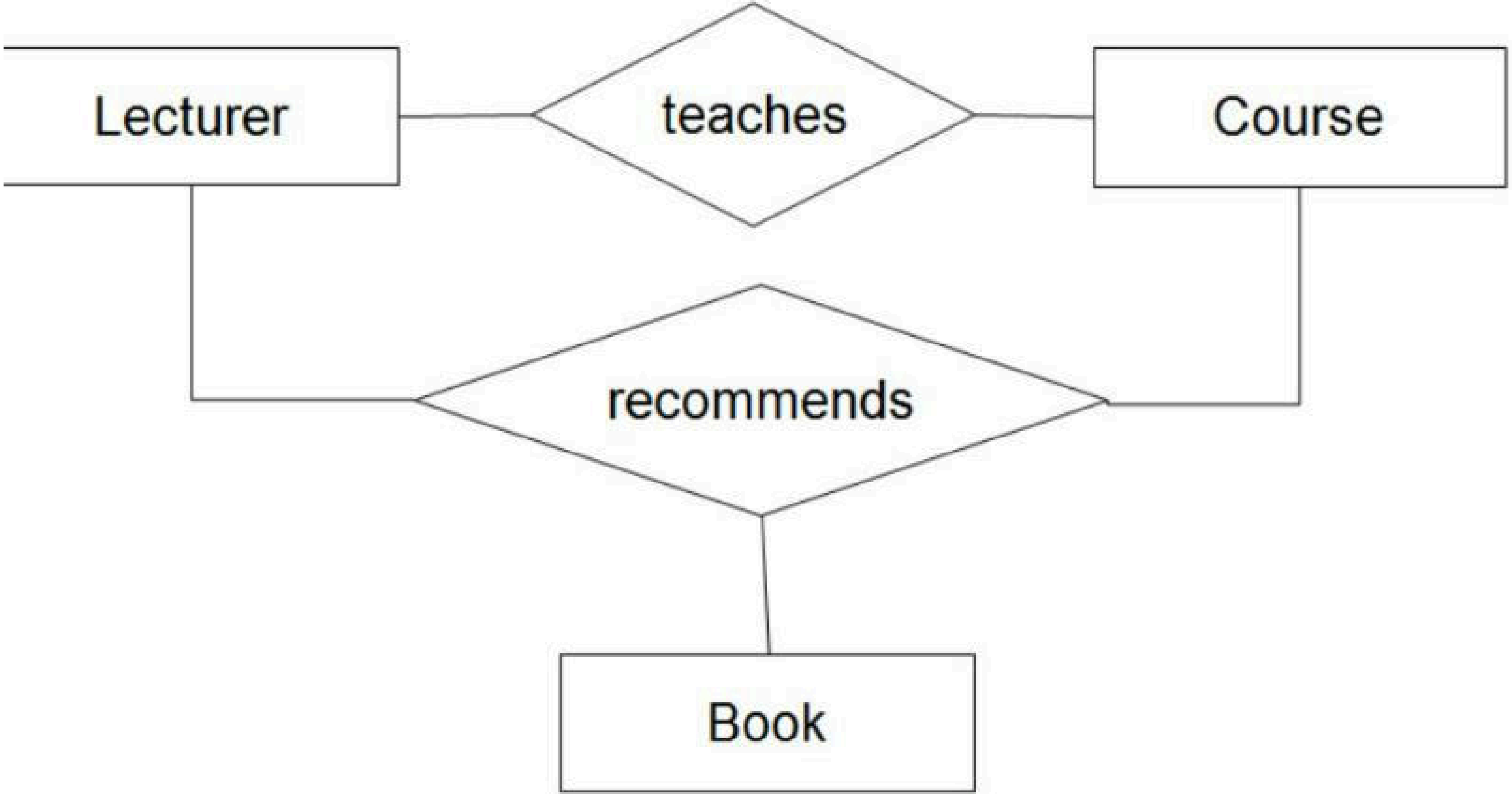


2.1 ERM

Ternary Relationship Types

- Higher degree relationship type: Literature recommendations for specific courses

2.1 ERM

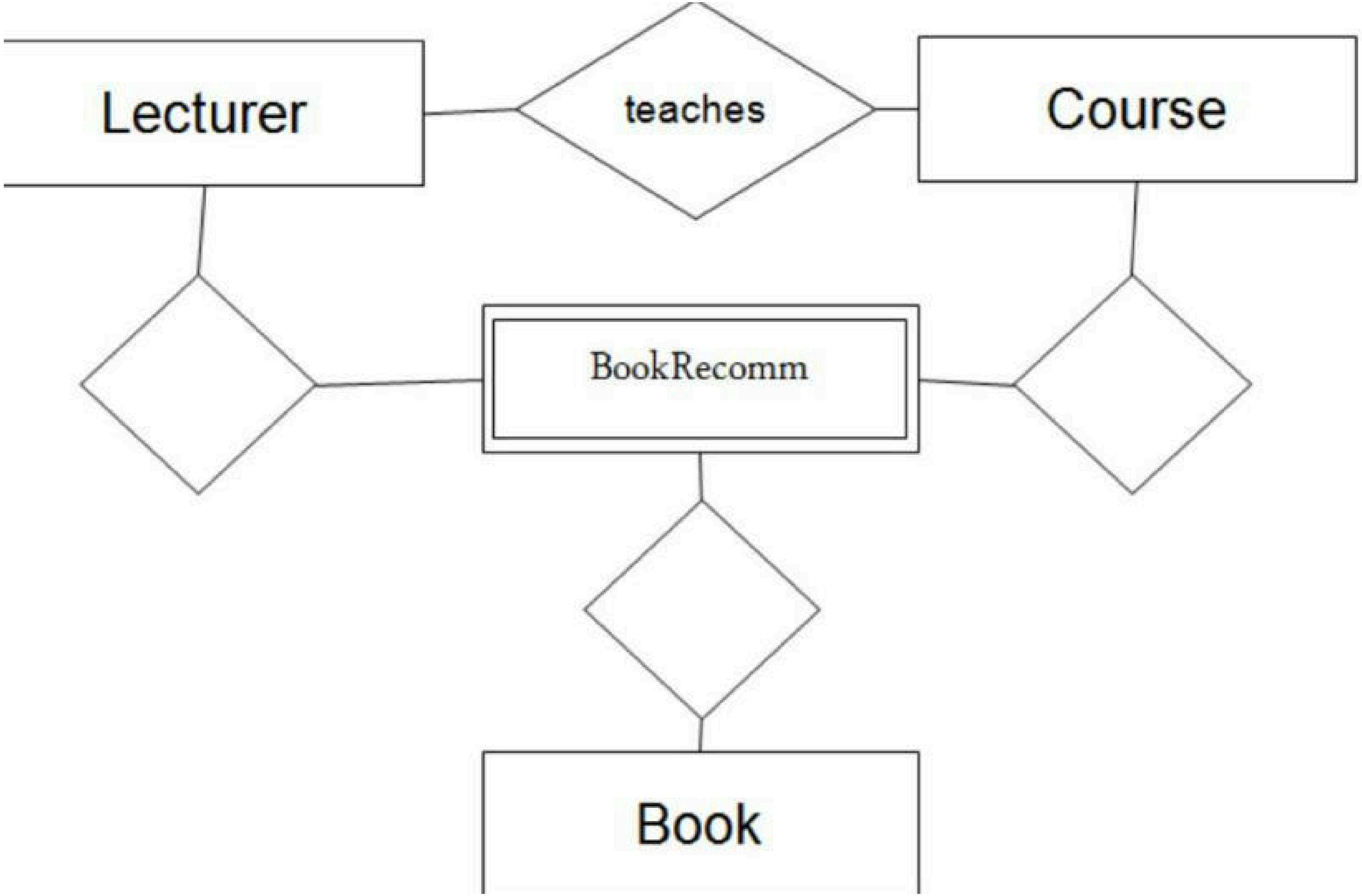


2.1 ERM

Ternary Relationship Types

- In many CASE tools, only binary relationship types can be represented
- Ternary relationship type is replaced by (weak) entity type + relationship types

2.1 ERM



2.1 ERM

Mapping of ERM

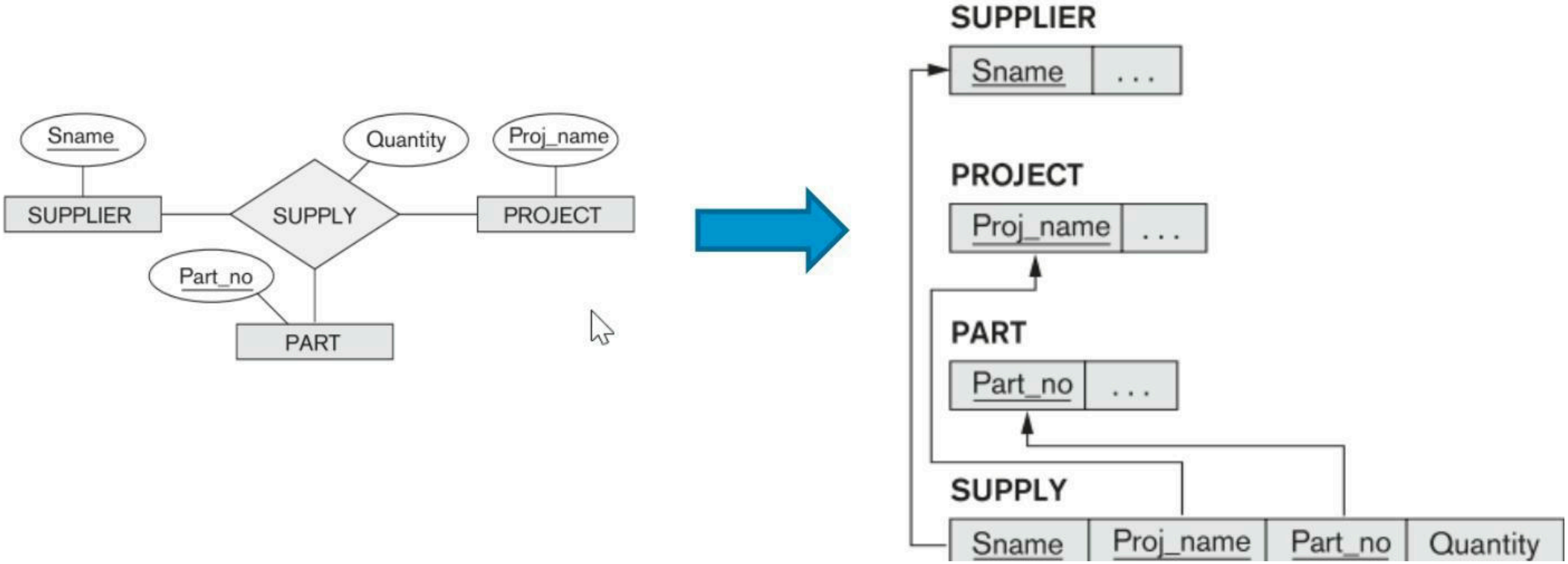
- Seven Steps

1. Mapping of regular entity types ✓
2. Mapping of weak entity types ✓
3. Mapping of binary 1:1 relationships ✓
4. Mapping of binary 1:n relationships ✓
5. Mapping of binary m:n relationships ✓
6. Mapping of multivalued attributes ✓
7. Mapping of n-ary relationships

Mapping of ERM: Mapping of n-ary relationships

- For each n-ary relationship type R, where $n > 2$, 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
- Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S
- The primary key of S is usually a combination of all the foreign keys that reference the relations representing the participating entity types

Mapping of ERM: Mapping of n-ary relationships



SUPPLIER

Sname | . . .

PROJECT

Proj_name | . . .

PART

Part_no | . . .

SUPPLY

Sname | Proj_name | Part_no | Quantity

2.1 ERM

Mapping of ERM

- Seven Steps
  1. Mapping of regular entity types ✓
  2. Mapping of weak entity types ✓
  3. Mapping of binary 1:1 relationships ✓
  4. Mapping of binary 1:n relationships ✓
  5. Mapping of binary m:n relationships ✓
  6. Mapping of multivalued attributes ✓
  7. Mapping of n-ary relationships ✓

Mapping of ERM: Summary

ER Model	Relational Model
Entity type	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 attributes
Multivalued attribute	Relation and foreign key
Value set	Domain
Key attribute	Primary key

3. Constraints

3.1 Theory

Basics

- Three categories
  1. Constraints that are inherent in the data model **inherent model-based constraints** or **implicit constraints** Example: no duplicate tuples in a relation
  2. Constraints that can be directly expressed in schemas of the data model **schema-based constraints** or **explicit constraints** Example: Domain constraints, key constraints, constraints on NULL, entity integrity constraints and referential integrity constraints

3.1 Theory

3. Constraints that cannot be directly expressed in the schemas of the data model, and hence must be expressed and enforced by the application programs **application-based** or **semantic constraints** or **business rules**

3.2 SQL

Basics

- Syntax for creating an empty table

```sql
CREATE TABLE < relationname >
         (<column> <type> [ DEFAULT expr ]
                  [ [NOT] NULL ] [ colconstraint ] *
         [,{<column> <type> [ DEFAULT expr ]
                  [ [NOT] NULL ] [ colconstraint ] *
          | <tableconstraint> } ] *
         );
```

3.2 SQL

- Some constraints (e.g., `UNIQUE`, `NOT NULL`) can be defined as column constraints or as table constraints

3.3 ERM

Key Attributes

- How can we identify an actual entity within an entity set?
- Attributes must be used
  - Key Attributes (also called identifying attributes)
- Sometimes several attributes together form a key attribute (identifying attribute), meaning that the combination of the attribute values must be distinct for each entity
  - If a set of attributes possesses this property, the proper way to represent this in the ER model that is to define a **composite attribute** and designate it as a key attribute of the entity type

3.3 ERM

  - Notice that such a composite key attributes must be minimal; that is, all component attributes must be included in the composite attribute to have the uniqueness property

- Key attributes are underlined
- If two attributes are underlined separately, then each is an identifying attribute on its own

3.4 RM

Primary Key

- Primary Key
  - Also called **Entity Integrity Constraint**
  - PK values must be unique and cannot be NULL!
  - Notation: <u>underlined</u>

ISBN	Title	Author	Publisher	Year	Price
978-1-292-09761-9	Fundamentals of Database Systems	Ramez Elmasri	Prentice Hall	2016	59.99
978-0321197849	An Introduction	C. J. Date	Pearson	2003	69.92

3.4 RM

ISBN	Title	Author	Publisher	Year	Price
	to Database Systems				

3.5 Domains

Basics

- A domain D is a set of atomic values
- Atomic means that each value is indivisible
- A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn
- It is also useful to specify a name for the domain, to help in interpreting its values

3.5 Domains

Example

- `Names`: The set of character strings that represent names of persons
- `Employee_age`: Possible ages of employees in a company; each must be an integer value between 16 and 70

3.5 Domains

Basics

- Domain constraints specify that within each tuple, the value of each attribute A must be an atomic value from the domain $\mathrm{dom}(A)$
- Data types
  - Numeric data types: short integer, integer, long integer
  - Real numbers: float, double
  - Characters
  - Booleans
  - Fixed-length or variable-length strings
  - Date

3.5 Domains

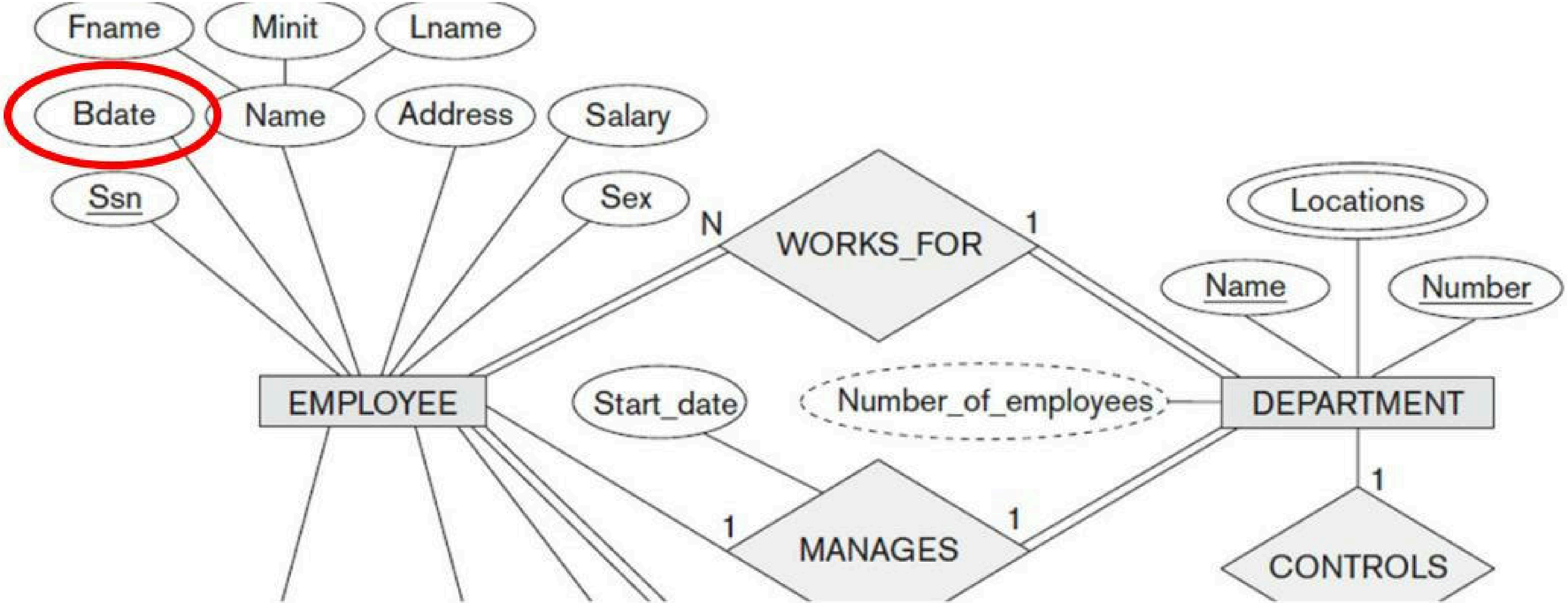
- Also, possible subrange of values from a data types and enumerated data types

3.5 Domains

Basics

- For example: Employee must be between 16 and 70 years old
- Value sets are not displayed in ER diagrams, and are typically specified using the basic data types available in most programming languages

3.5 Domains


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS

3.5 Domains

Basics

- Example:
  - `Employee_age`: integer number between 16 and 70
  - `Mobile_Number`: (dd)ddd-ddddddd d is a decimal digit
- A domain is thus given a name, data type, and format

3.5 Domains

Basics

- `CREATE DOMAIN` is part of the ANSI Standard But almost all RDBMS ignore this command
- A domain is simply a self-defined data type More precise it is a limitation of the values range of a data type

3.5 Domains

Example

- You want to save the age of a person, you could use `INTEGER`
- Now you could save a value from −2,147,483,648 to 2,147,483,647
- No person will reach an age of 2,147,483,647 years, neither can someone be younger than 0

- By creating a domain, you could design a useful value range (e.g., 0 => column =< 100).

3.5 Domains

- For using a domain, you just type the domain-name instead of a data type

3.5 Domains

Basics

- Based on base data type … with additional constraints

SQL

```sql
CREATE DOMAIN <name> [ AS ] datatype [ DEFAULT expession] [ constraint [ . . . ] ] ; [ CONSTRAINT constraint\_name] { NOT NULL | NULL | CHECK ( expression )
```

3.5 Domains

Example

SQL

```sql
CREATE DOMAIN nnint AS INT NOT NULL ;
CREATE DOMAIN posint AS INT CHECK ( VALUE >= 0 ) ;
CREATE DOMAIN dayofweek AS VARCHAR CHECK ( VALUE IN ( 'Monday' , 'Tuesday' , ...)) ;
CREATE DOMAIN SSN TYPE AS CHAR(9);
```

3.5 Domains

SQL - Check

- One option to implement domains in mySQL
- `colconstraint` = Column constraint
  - `CHECK`
    - Constraint can restrict attribute or domain values using the `CHECK` clause following an attribute or domain definition

Example

SQL

```sql
Dnumber INT NOT NULL CHECK (Dnumber > 0
AND Dnumber < 21);
```

3.5 Domains

SQL - Check

- Tuple-based constraints
  - Semantical Integrity
  - Apply to each tuple individually and are checked whenever a tuple is inserted or modified
- Makes sure that condition is met … or `NULL`!
- Typical use case: Range checking

Example

```sql
CONSTRAINT chk\_age CHECK (age ≥ 18)
```

SQL

3.5 Domains

- Also, complex conditions possible (verify relationships with other rows and/or tables)

Example

SQL

```sql
    CHECK (Dept_create_date <= Mgr_start_date);
```

3.5 Domains

SQL - Check

- The `CHECK` clause can also be used in conjunction with the `CREATE DOMAIN` statement if supported by the DBMS

- For example, we can write the following statement:

```sql
CREATE DOMAIN D_NUM AS INTEGER CHECK (D_NUM > 0 AND D\_NUM < 21 );
```

3.5 Domains

Constraints on `NULL`

- Constraint on attributes specifies whether NULL values are or are not permitted
- `NULL`: special attribute value
  - Value unknown (exists but is intentionally withheld)
    - A person's date of birth is not known, so it is represented by NULL in the data base
  - Unavailable or withheld value (exists but is not known)
    - A person has a home phone but does not want it to be listed, so it is withheld and represented as `NULL` in the database.

3.5 Domains

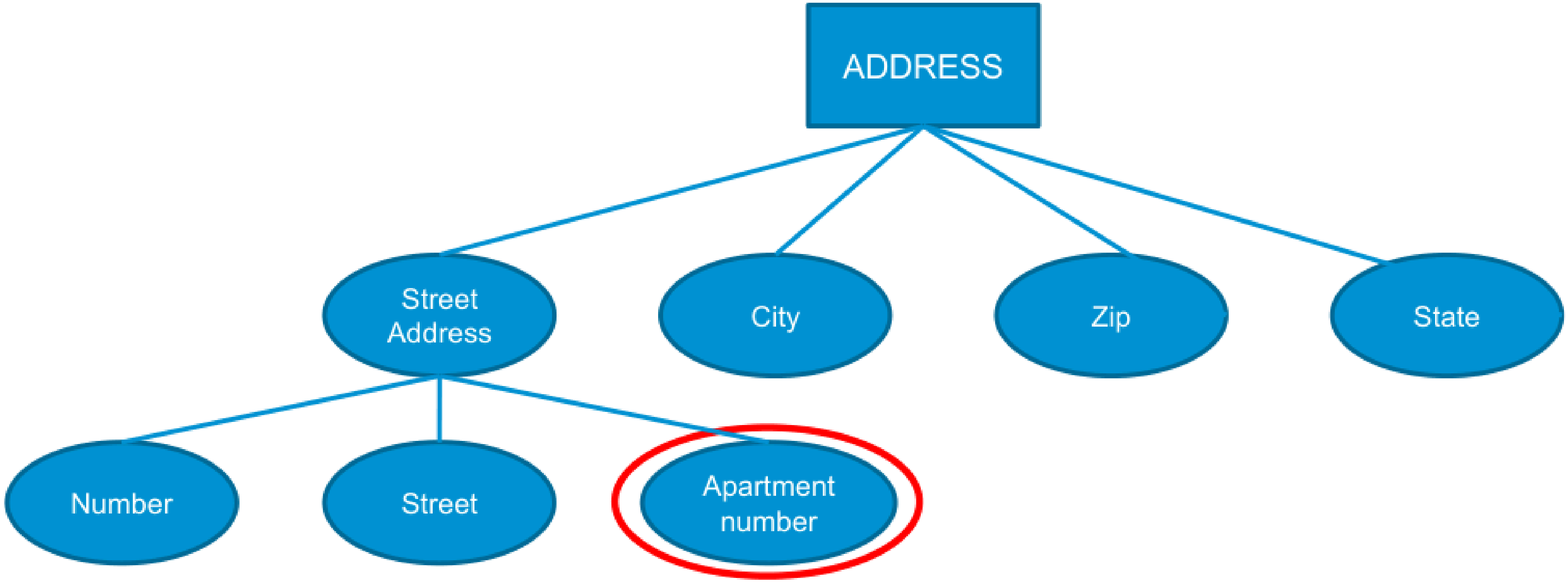
- ▸ Not applicable (the attribute is undefined for this tupel)
  - – Student’s name has no middle initials, student has no academic degree, …
- Example: primary key

3.5 Domains

Constraints on `NULL`

- `NULL` values
  - A particular entity may not have an applicable value for an attribute, e.g. apartment number in address

3.5 Domains



- `NULL` can also be used if we do not know the value of an attribute for a particular entity

3.5 Domains

Constraints on NULL

- Syntax for creating an empty table:

SQL

```sql
CREATE TABLE < relationname >
         (<column> <type> [ DEFAULT expr ]
                  [ [NOT] NULL ] [ colconstraint ] *
         [,{<column> <type> [ DEFAULT expr ]
                  [ [NOT] NULL ] [ colconstraint ] *
          | <tableconstraint> } ] *
         ) ;
```

3.5 Domains

Constraints on NULL

- `NULL`:
  - "no information available"
  - "no information available yet"
  - "unknown"
  - "not applicable"

- Examples:
  - Birthdate
  - Apartment Number
  - Minit

3.5 Domains

Constraints on `NULL`

- `NOT NULL` mandatory field, a value is needed
- Default: `NULL` optional field, `NULL` is allowed
- This is always implicitly specified for the attributes of the primary key of each relation

> **! Memorize**
>
> Attention: Most attributes should be `NOT NULL`!

3.5 Domains

UNIQUE

- UNIQUE constraint prevents duplicates in a column, i.e., a duplicate entry is not valid in a unique column
- A **unique key** is a candidate key
- All the candidate keys of a relation can uniquely identify the records of the relation, but only one of them is used as the primary key of the relation
- Example: primary key

3.5 Domains

`UNIQUE`

- Additional identifying attributes: alternate (secondary) keys
- In SQL: `UNIQUE`
- Unique elements can be `NULL`
  - … in some implementations
  - Thinking of key: should not be nullable
    - Unique: could be null (several nulls allowed!)

3.5 Domains

- As column constraint

```sql
Dname VARCHAR(15) UNIQUE;
```

- As table constraint

```sql
CREATE TABLE Department
( Dname VARCHAR(15) NOT NULL Dnumber INT NOT NULL …
PRIMARY KEY ( Dnumber ) ,
UNIQUE ( Dname ),
...);
```

3.5 Domains

Relationship Types

- Cardinality
  - Specifies the maximum number of relationship instances that an entity can participate in
  - Cardinality ratios
    - 1:1
    - 1:N
    - M:N
  - Cardinality ratios for binary relationships are represented on ER diagrams by displaying 1, M, and N on the diamonds

3.5 Domains

- Notice that in this notation, we can either specify no maximum (N) or a maximum of one (1) on participation

3.5 Domains

Relationship Types

- Participation
  - Specifies whether the existence of an entity depends on its being related to another entity via the relationship type
  - Also called **minimum cardinality constraint**
  - Two types
    - Total: every entity in the total set of all entities of an entity type A must be related to an entity of entity type B via a relationship
      - Total participation is also called **existence dependency**

3.5 Domains

    - Is displayed as a double line connecting the participating entity type to the relationship
  - Partial: some or part of the entities of an entity type A are related to some entities of an entity type B via a relationship
    - Is displayed by a single line connecting the participating entity type to the relationship336

3.5 Domains

Relationship Types

- Cardinality
  - specifies the maximum number of relationship instances that an entity can participate in
- Participation
  - specifies if the existence of an entity depends on its being related to another entity via the relationship type
  - **minimum cardinality constraint**

3.5 Domains

Referential Integrity Constraint

- It is defined between two relations
- It is used to maintain the consistency among tules in the two relations: a tuple in one relation that refers to another relation must refer to an existing tuple in that relation
- **Foreign key**: a set of attributes FK in relation schema R_1 is a foreign key of R_1 that references relation R_2 if it satisfies the following rules:
  1. The attributes in FK have the same domain(s) as the primary key attributes PK of R_2; the attributes FK are said to **reference** or **refer to** the relation R_2.

3.5 Domains

2. A value of FK in a tuple t_1 of the current state $r_1(R_1)$ either occurs as a value of PK for some tuple t_2 in the current state $r_2(R_2)$ or is `NULL`. In the former case, we have $t_1[\mathrm{FK}] = t_2[\mathrm{FK}]$, and we say that the tuple t_1 references or refers to the tuple t_2.

3.5 Domains

Foreign Key - Syntax

- As Column Constraint
  - Only if the foreign key is one single attribute (and not combined)

```SQL
[CONSTRAINT < constraintname > ]
        REFERENCES < tablename >[( column )] [< action >]
```

- As Table Constraint

```SQL
[CONSTRAINT < constraintname >]
        FOREIGN KEY (< column list >)
```

3.5 Domains

```
      REFERENCES < tablename >[(< column list >)]
              [< action >]
```

3.5 Domains

Foreign Key - Syntax

SQL

```sql
CREATE TABLE Department
( Dname VARCHAR(15) NOT NULL,
Dnumber INT NOT NULL,
Mgr_ssn CHAR(9) REFERENCES Employee(Ssn),
Mgr_start_date DATE,
PRIMARY KEY (Dnumber),
UNIQUE (Dname));
```

3.5 Domains

Foreign Key - Syntax

SQL

```sql
CREATE TABLE Department
( Dname VARCHAR(15) NOT NULL,
Dnumber INT NOT NULL,
Mgr_ssn CHAR(9) NOT NULL,
Mgr_start_date DATE,
PRIMARY KEY(Dnumber),
UNIQUE(Dname),
FOREIGN KEY (Mgr_ssn) REFERENCES Employee (Ssn));
```

3.5 Domains

Foreign Key - Syntax

- `< action >`:
  - How to react on changes to the referenced table
- The default action: reject the update operation (`RESTRICT` option)

SQL

```sql
action ::= ON {UPDATE | DELETE}
{NO ACTION | SET NULL | SET DEFAULT | CASCADE}
```

3.5 Domains

Foreign Key

- Options:
  - `SET NULL` Value of foreign key is set to `NULL`
  - `SET DEFAULT` Value of foreign key is set to a default value
  - `CASCADE` Value of foreign key is updated
- For example:
  - `ON DELETE CASCADE` Delete all referencing tuples
  - `ON UPDATE CASCADE` Change Value of the foreign key attribute(s)
- General Rule for using `CASCADE`:
  - For “relationship” relations
  - For multivalued attributes

3.5 Domains

- For relations that represent weak entity types

3.5 Domains

Foreign Key

SQL

```sql
  CREATE TABLE Employee
(...,
  Dno INT NOT NULL DEFAULT 1,
  CONSTRAINT EMPPK PRIMARY KEY (Ssn),
  CONSTRAINT EMPSUPERFK FOREIGN KEY (Super_ssn) REFERENCES Employee(Ssn)
    ON DELETE SET NULL ON UPDATE CASCADE,
  CONSTRAINT EMPDEPTFK FOREIGN KEY(Dno) REFERENCES Department(Dnumber)
    ON DELETE SET DEFAULT
```

3.5 Domains

```
      ON UPDATE CASCADE);
```

3.5 Domains

Foreign Key

- All constraints get an identifier … if not by you, then by the system
- Problems with system generated identifiers Bad error messages
- Maybe we want to alter or drop the constraint later? Then we need its name!
- Exception: `NOT NULL` constraints no need for identifier
- The names of all constraints within a particular schema must be unique

3.5 Domains

Foreign Key

SQL

```sql
 CREATE TABLE Employee
(...,
  Dno INT NOT NULL DEFAULT 1,
  CONSTRAINT EMPPK PRIMARY KEY (Ssn),
  CONSTRAINT EMPSUPERFK FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
    ON DELETE SET NULL
    ON UPDATE CASCADE,
  CONSTRAINT EMPDEPTFK FOREIGN KEY(Dno) REFERENCES Department(Dnumber)
```

3.5 Domains

```
   ON DELETE SET DEFAULT
   ON UPDATE CASCADE
);
```

3.5 Domains

Other Constraints

- Semantic integrity constraints
  - Example: The maximum number of hours an employee can work on all projects per week is 56
  - Realization:
    - Within the application programs or
    - Constraint specification language, e.g., trigger and assertions
- Functional dependencies constraint
  - It establishes a functional relationship among two sets of attributes X and Y

3.5 Domains

- This constraint specifies that the value of X determines a unique value of Y in all states of a relation
- It is denoted as a functional dependency $X \longrightarrow Y$

3.5 Domains

Overview

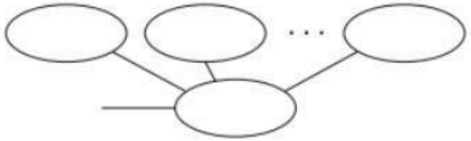
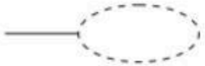
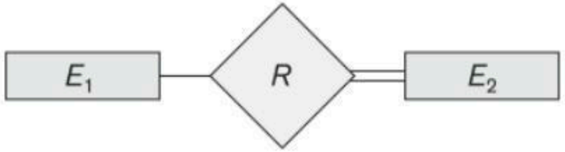
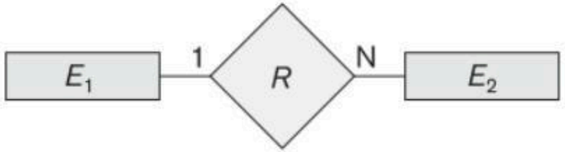
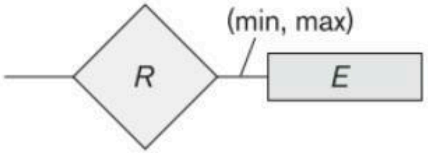
Constraint	Number of affected Relations
Domain constraints	1
Constraints on `NULL`	1
Entity integrity constraints (primary key)	1
Referential integrity constraints	$\overline{<} 1$
Semantic integrity constraints	$\overline{<} 1$
Functional dependencies constraint	$\overline{<} 1$

4. Notation and Guidelines

4.1 ERM

Basics

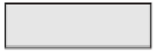
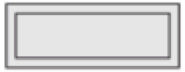
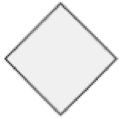
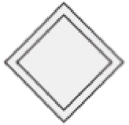
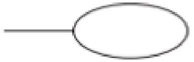
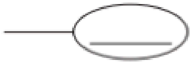
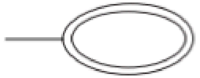
4.1 ERM

Symbol	Meaning
	Composite Attribute
	Derived Attribute
	Total Participation of E_2 in R
	Cardinality Ratio 1: N for E_1:E_2 in
	Structural Constraint (min, max) on Participation of E in R

4.1 ERM

Basics

4.1 ERM

Symbol	Meaning
	Entity type
	Weak entity type
	Relationship type
	Identifying relationship type
	Attribute
	Key Attribute
	Multivalued Attribute

Basics

- Names
  - Entity type and relationship type names:
    - uppercase letters
  - Attribute names
    - initial letter capitalized
  - Role names
    - lowercase letters
- Binary relationship names to make the ER diagram of the schema readable from left to right and from top to bottom

4.1 ERM

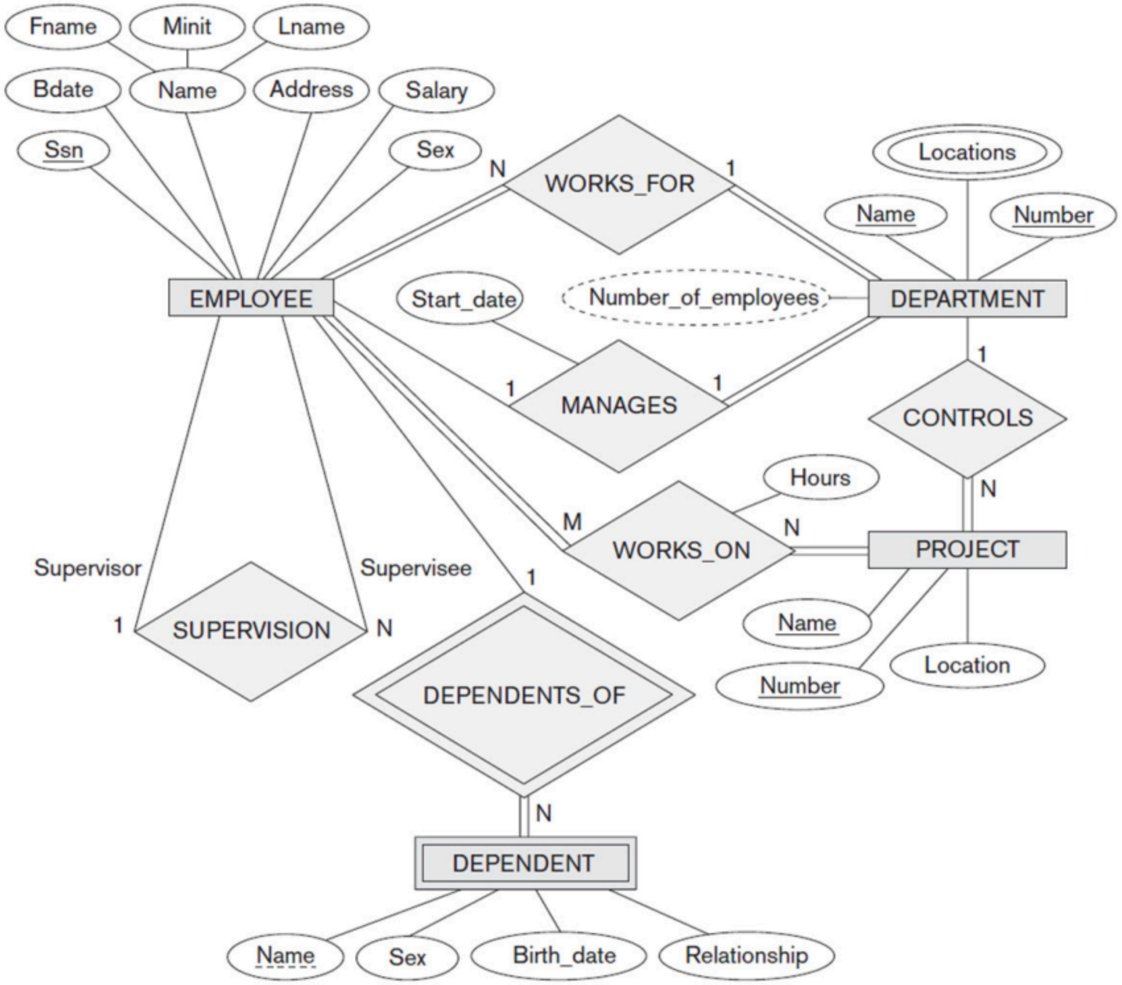
Basics

- In general, the schema design process should be considered an iterative refinement process
- An attribute may be refined to a relationship if it is a reference to another entity type
- If an attribute exists in several entity types, it may be promoted to an independent entity type
- If an entity type A exists in the initial design with a single attribute and is related to only one other entity type B, the entity type A may be reduced or demoted to an attribute of entity type B

4.1 ERM

Basics

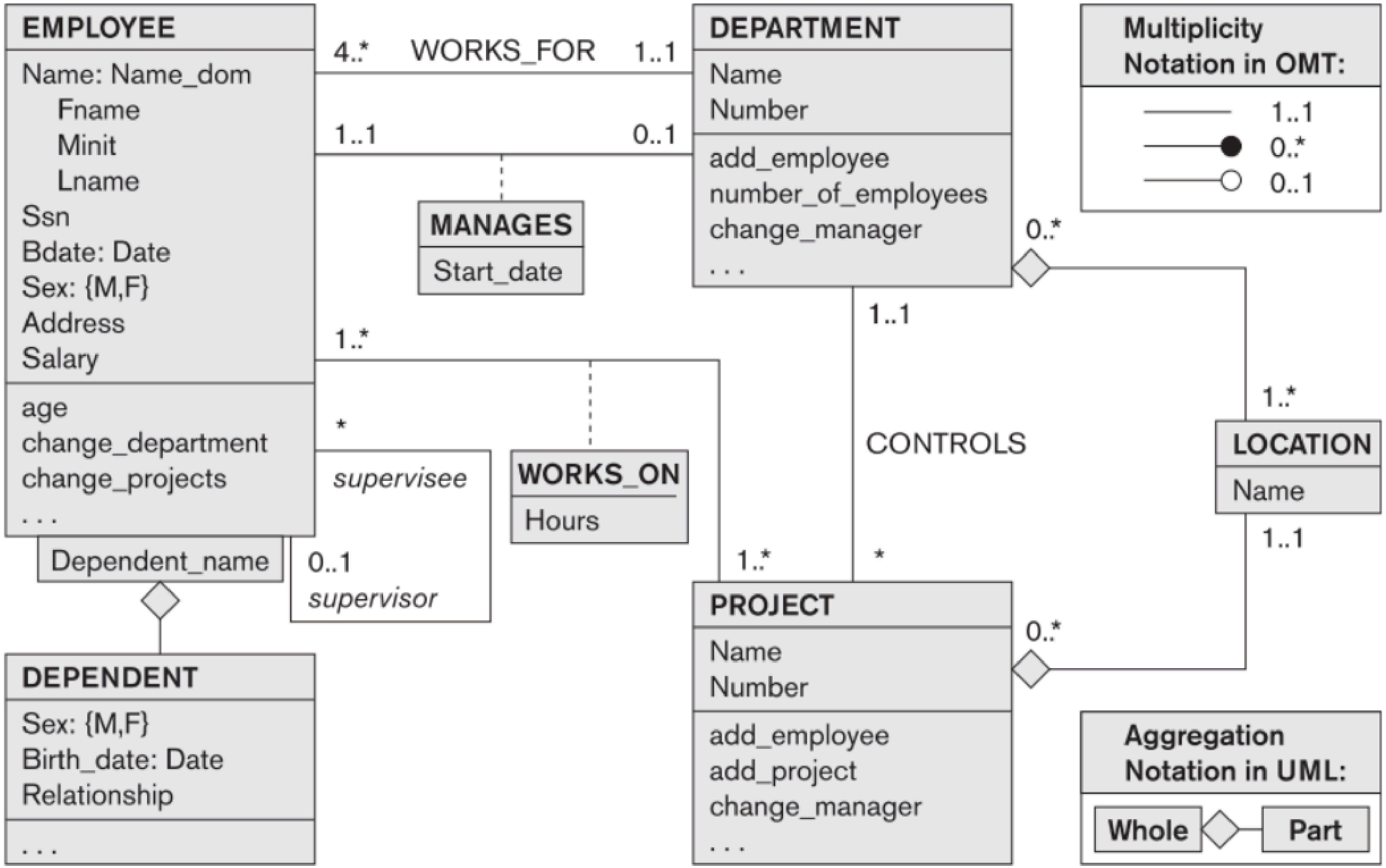
4.1 ERM


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS
Hours
N
M
N
WORKS_ON
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
Name
Number
Location
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
Birth_date
Relationship

4.1 ERM

UML Notation

4.1 ERM


EMPLOYEE
Name: Name_dom
Fname
Minit
Lname
Ssn
Bdate: Date
Sex: {M,F}
Address
Salary
age
change_department
change_projects
. . .
Dependent_name
DEPENDENT
Sex: {M,F}
Birth_date: Date
Relationship
. . .
4..*
WORKS_FOR
1..1
1..1
0..1
MANAGES
Start_date
1..*
*
supervisee
0..1
supervisor
WORKS_ON
Hours
1..*
DEPARTMENT
Name
Number
add_employee
number_of_employees
change_manager
. . .
0..*
1..1
CONTROLS
*
PROJECT
Name
Number
add_employee
add_project
change_manager
. . .
0..*
1..*
LOCATION
Name
1..1
Multiplicity
Notation in OMT:
1..1
0..*
0..1
Aggregation
Notation in UML:
Whole
Part

4.1 ERM

MC Notation

- Participation constraints
- Relationships can be mandatory or optional
- Types
  - Exactly one element: 1
  - One or no element: c (or 1c)
  - No or many elements: mc (or nc)
  - One or many elements: m (or n)

4.1 ERM

ℹ Info

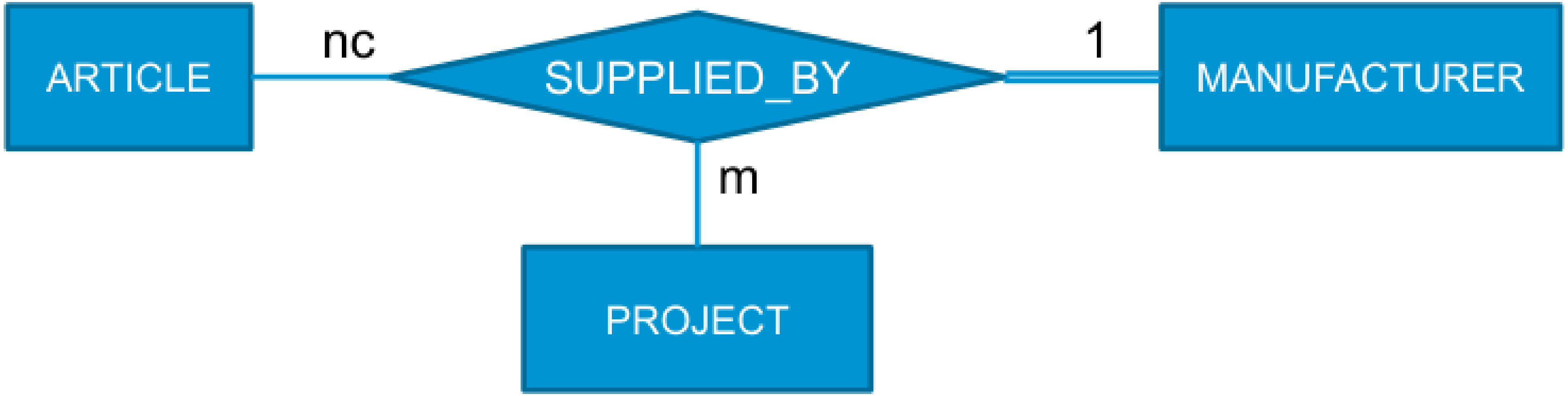
Also called must-can notation!

MC Notation - Tenary Relationship

Example

- Manufacturers supply items for projects.
- A manufacturer must supply at least one item.
- An article from in-house production does not have to be supplied for a project but can be supplied for many projects.
- A project uses at least one item.
- An item is supplied by only one manufacturer for a project.

4.1 ERM



4.1 ERM

MC Notation - Tenary Relationship

? Question

Cardinality: Can an entity of entity type A and an entity of entity type B be related to multiple entities of entity type C?

? Question

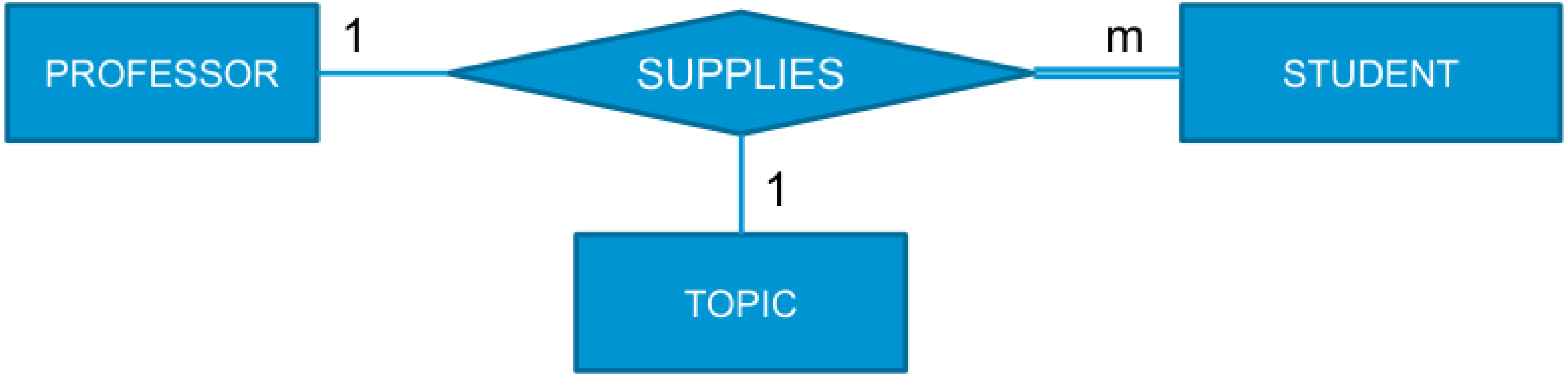
Participation: Must an entity type A be related to at least one entity type B and one entity type C?

4.1 ERM

MC Notation

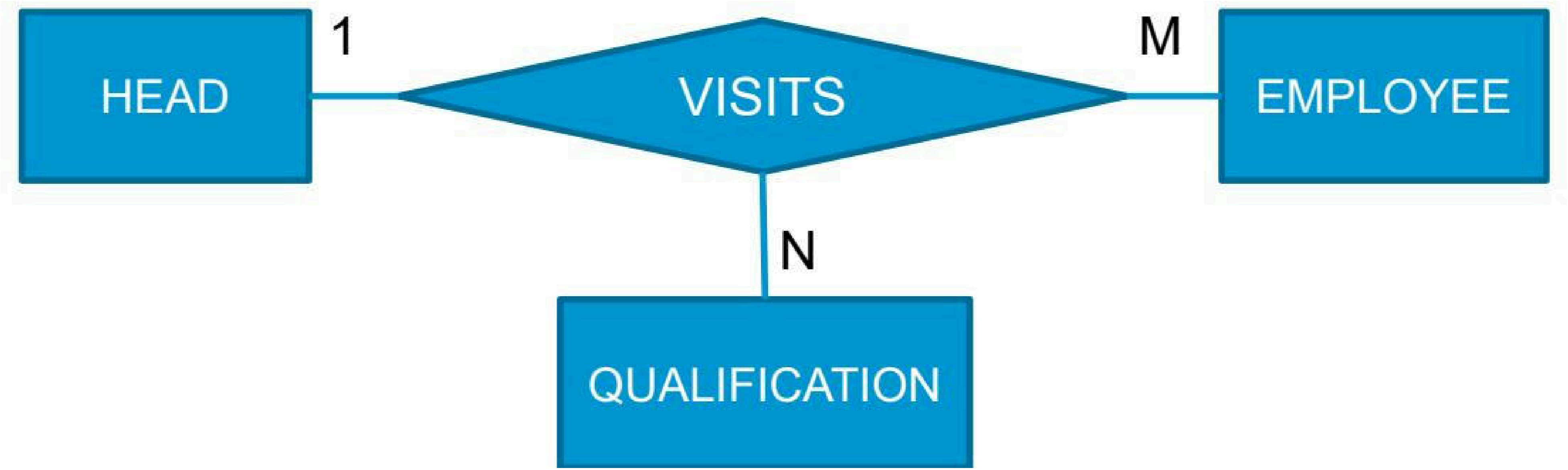
- To prevent students from concentrating on one professor, they may only work with one professor on one seminar topic.
- In addition, a student can only work on a seminar topic with one professor.
- However, a professor may assign a seminar topic more than once.
- Students must attend seminars, but seminar topics do not have to be chosen.

4.1 ERM



4.1 ERM

MC Notation - Tenary Relationship



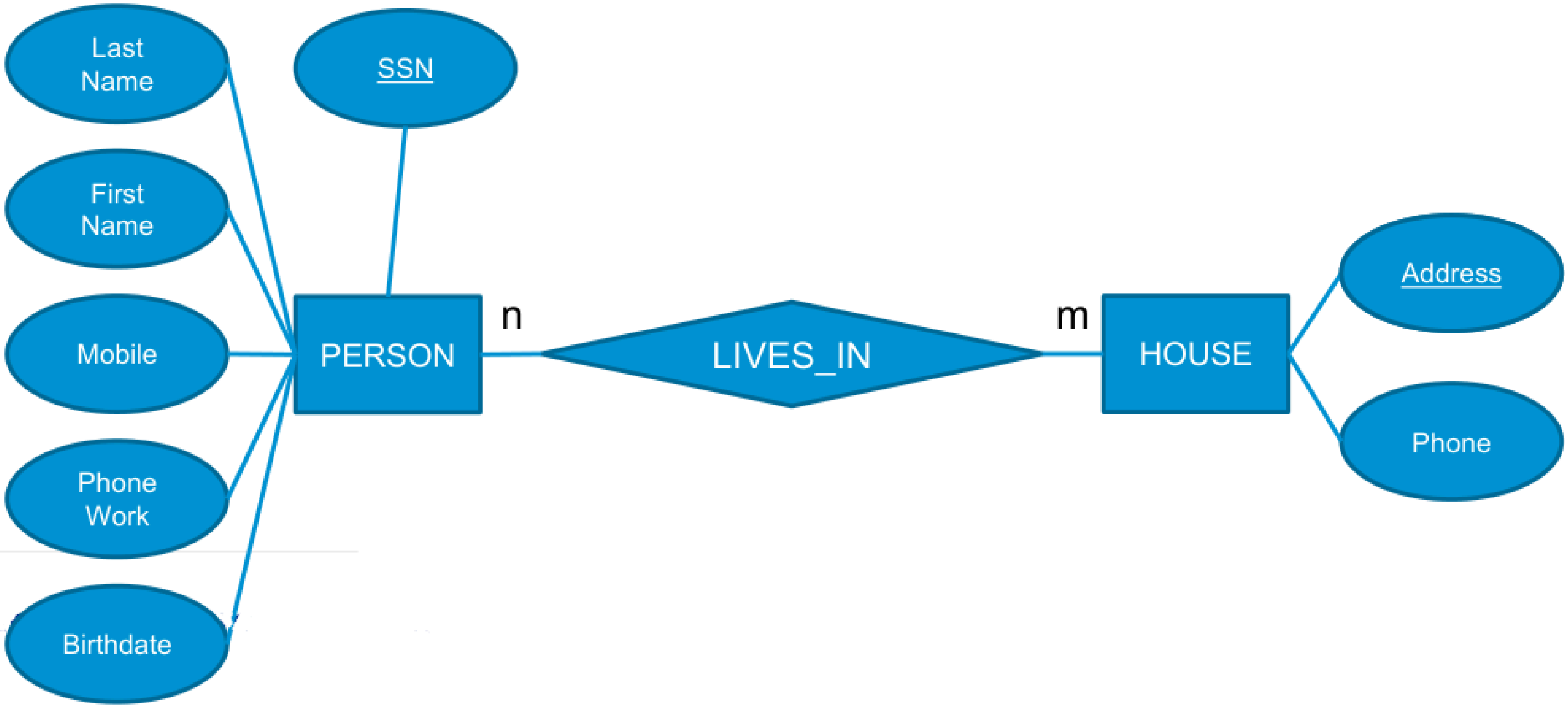
4.1 ERM

Comparison ERM and RM

- ERM:
  - Conceptual Database Design
  - Describes a collection of **entities**, also called as real-world **objects** and **relations** between those entities
  - Basic elements: **entity type**, **relationship type** and **attributes**
  - **Constraints** like **Cardinality**, **Participation ratio** and **Keys**

4.1 ERM

ERM:

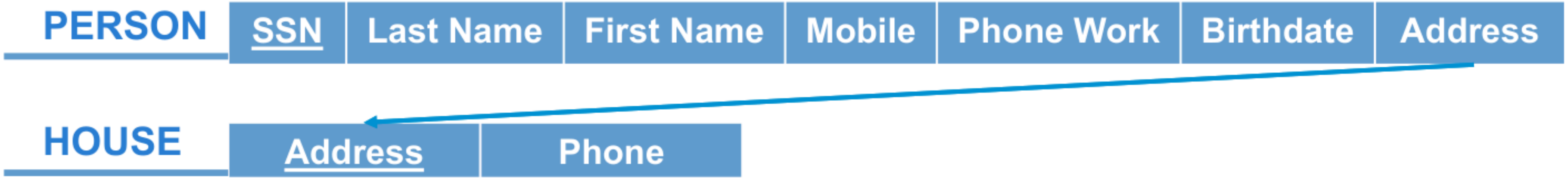

Last Name
SSN
First Name
Mobile
Phone Work
Birthdate
PERSON
n
LIVES_IN
m
HOUSE
Address
Phone

Comparison ERM and RM

- Relational Model:
  - Logical Database Design
  - Describes data and relation among those data by tables
  - Basic elements: Relations and Attributes
  - Constraints: Domain constraints, key constraints, constraints on `NULL`, entity integrity constraints and referential integrity constraints

4.1 ERM

Relational Model

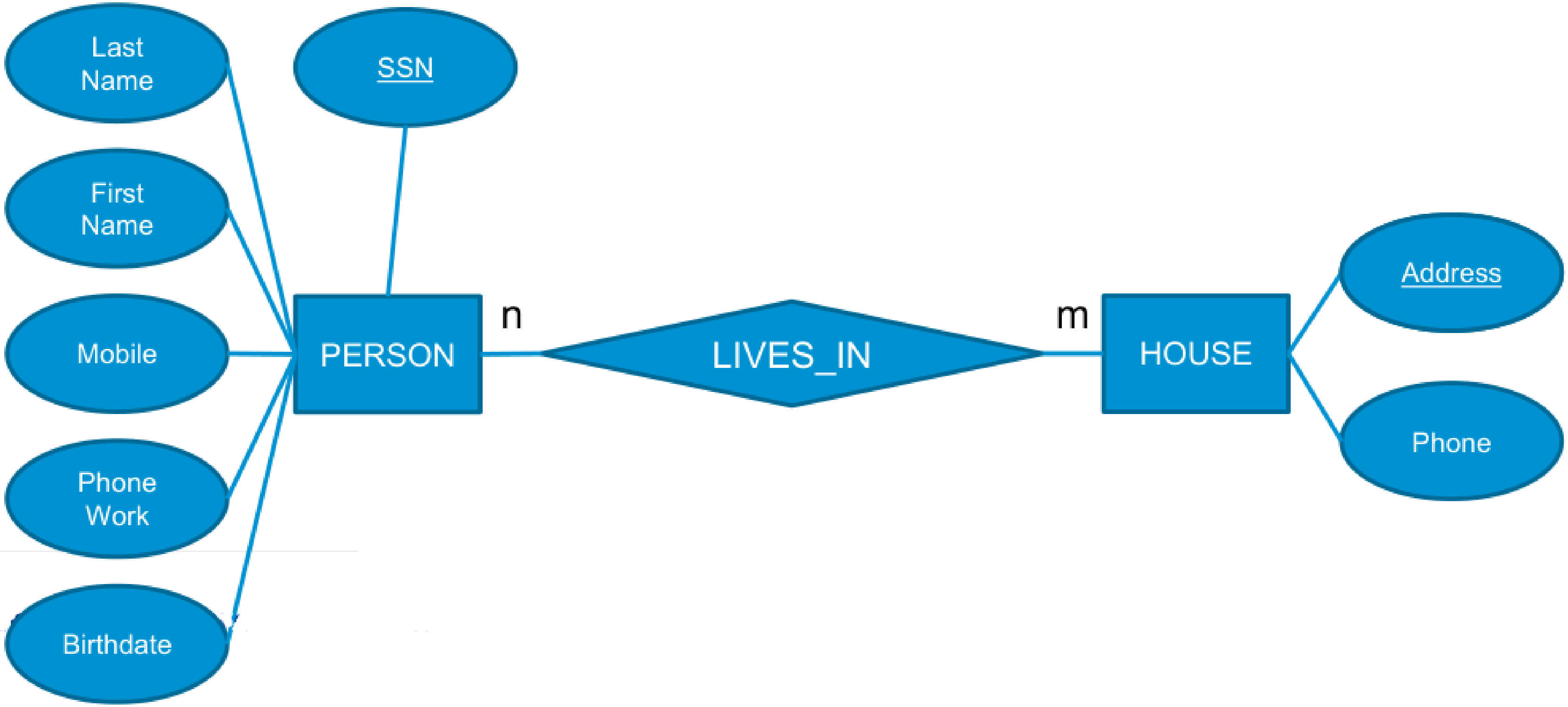


PERSON: SSN | Last Name | First Name | Mobile | Phone Work | Birthdate | Address

HOUSE: Address | Phone

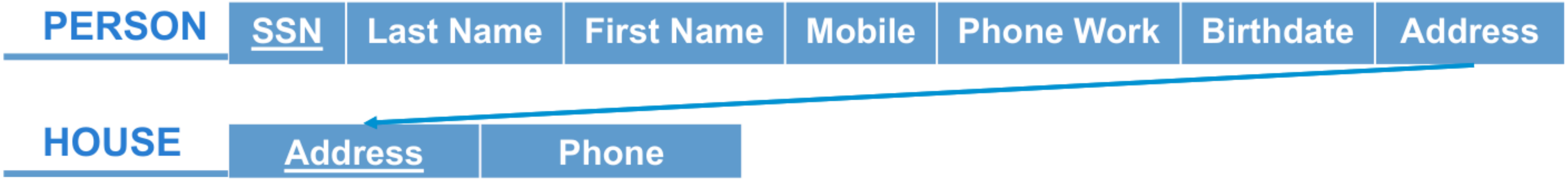
Comparison ERM and RM

ERM:



4.1 ERM

Relational Model



PERSON	SSN	Last Name	First Name	Mobile	Phone Work	Birthdate	Address

HOUSE	Address	Phone

4.1 ERM

Comparison ERM and RM

Aspect	ERM	RM
Basic	It represents the collection of objects called entities and relation between those entities	It represents the collection of tables and the relation between those tables
Describe	ERMs describe data as entity set, relationship set and attributes	Relational model describes data in a table as domains, attributes, tuples

4.1 ERM

Aspect	ERM	RM
Relationship	In an ERM, it is easier to understand the relationships between entities	Comparatively, it is less easy to derive a relation between tables in relational model
Mapping	ERM describes mapping cardinalities	Relational model does not describe mapping cardinalities

4.1 ERM

Mapping of ERM to RM

Main rules

- Entity types
  - Mapped to relations
  - Relations contain the attributes
  - Composite attributes: set of simple attributes
- Relationship Types
  - Foreign keys or
    - Relations plus Foreign keys

4.1 ERM

Mapping of ERM to RM

- Seven Steps
  1. Mapping of regular entity types
  2. Mapping of weak entity types
  3. Mapping of binary 1:1 relationships
  4. Mapping of binary 1:n relationships
  5. Mapping of binary m:n relationships
  6. Mapping of multivalued attributes
  7. Mapping of n-ary relationships

4.1 ERM

Design Guidelines

5. License Notice

5.1 Attribution

- This work is shared under the CC BY-NC-SA 4.0 License and the respective Public License.
- https://creativecommons.org/licenses/by-nc-sa/4.0/
- This work is based off of the work by Prof. Dr. Ulrike Herster.
- Some of the images and texts, as well as the layout were changed.
- The base material was supplied in private, therefore the link to the source cannot be shared with the audience.