Databases

Lecture 4 - Relationships

Emily Lucia Antosch

HAW Hamburg

19.04.2025

Contents

1. Introduction

1.1 Where are we right now?

- Last time, we looked at how to logically and conceptually design databases.
- You also learned about how to use simple SQL language to map a conceptually designed database to an actual one.
- Today, we'll look at
  - what relationships are in terms of databases,
  - how we can use constraints to enforce our relationships and,
  - how we can implement that in SQL.

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
  - define relationships and identify which type of relationship that is and,
  - decide how to implement that relationship in SQL.

2. Relationships

2.1 Relationship Types

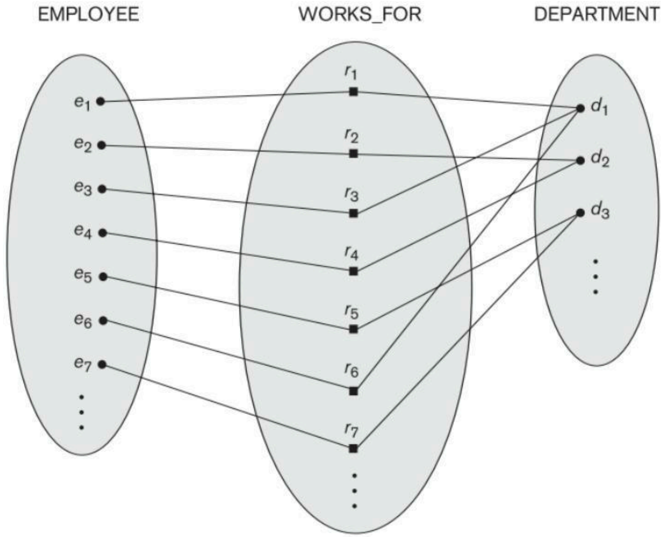
Basics

- Describe relationships between entity types characterized by a verb
- Often 2 naming possibilities:
  - teaches vs. is taught by
  - Relationship has always two (or more) directions
- May have attributes
- Number of participating entity types (degree):
  1. Unary relationship type (e.g., Employee supervises another employee)
  2. Binary relationship type (e.g., Employee works for one department)
  3. Ternary relationship type (e.g., Lecturer recommends books for one specific course)
  4. Higher degrees…

2.1 Relationship Types

Basics

- Each relationship instance in R is an association of entities, where the association includes exactly one entity from each participating entity type
- In an ERM, relationship types are displayed as diamond-shaped boxes, which are connected by straight lines to the entity types

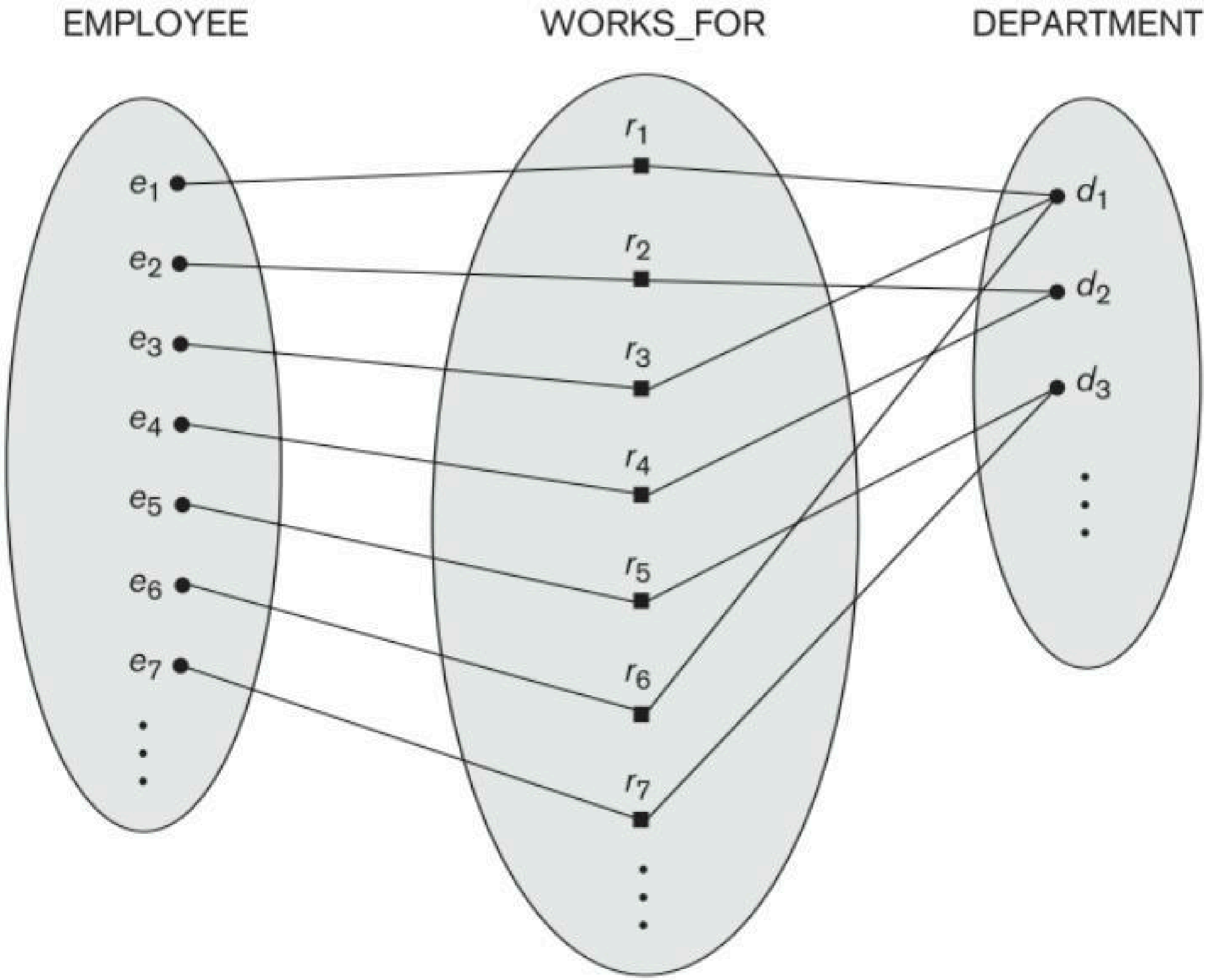


2.1 Relationship Types

Entity Type

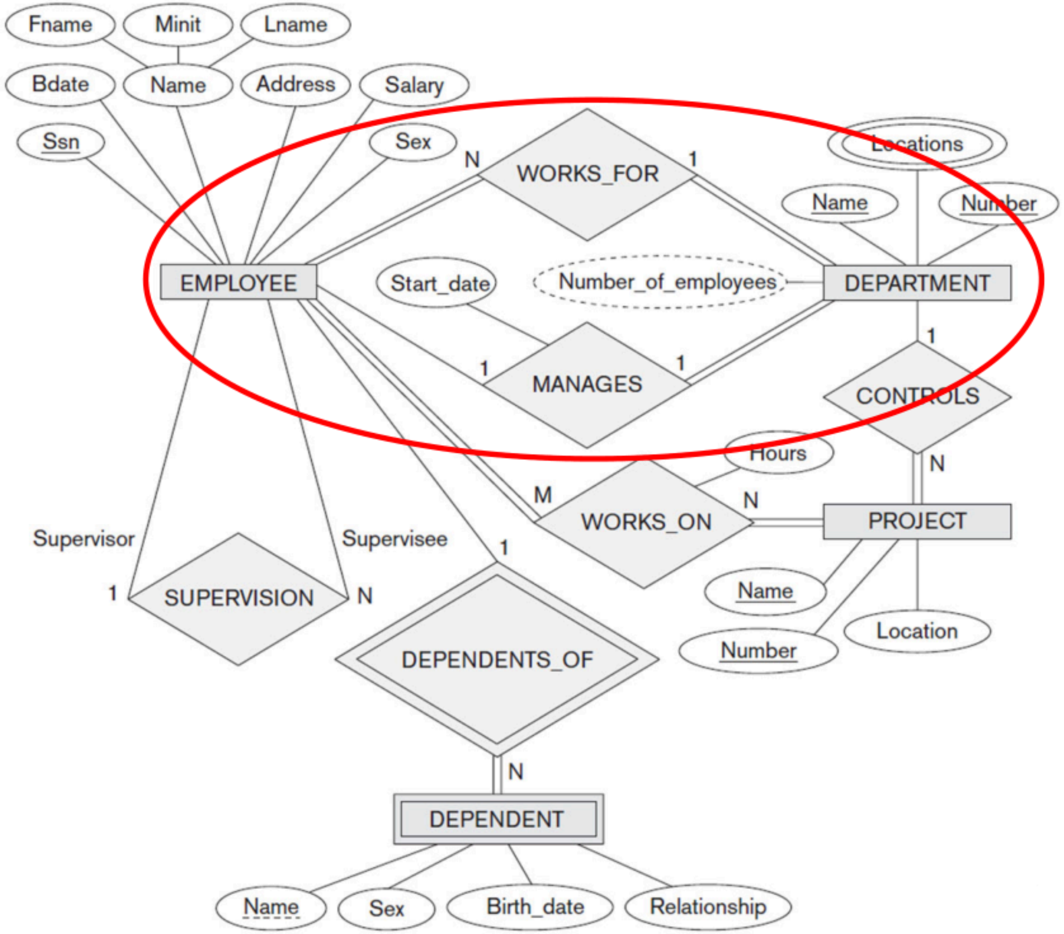
- Represented as rectangle in ERM
- Singular noun
- Attribute Type
    - Represented as ovals
    - Noun
- Relationship Type
    - Represented as diamond in ERM
    - Always between entity types
    - Verb
    - Has cardinalities

2.1 Relationship Types



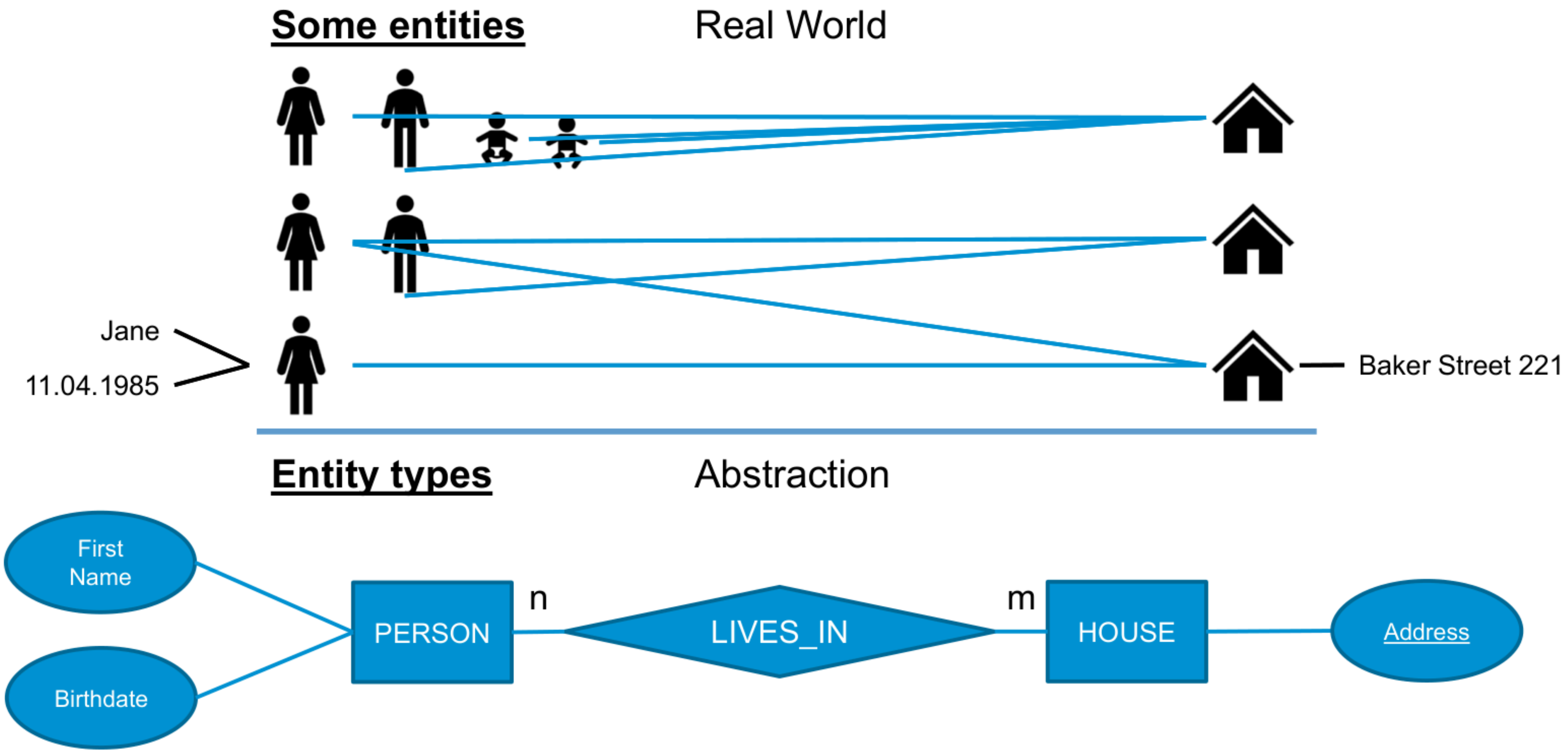
2.1 Relationship Types

Company Example



2.1 Relationship Types

Abstraction



2.1 Relationship Types

Role Names

- The role name signifies the role that a participating entity from the entity type plays in each relationship instance, and helps to explain what the relationship means

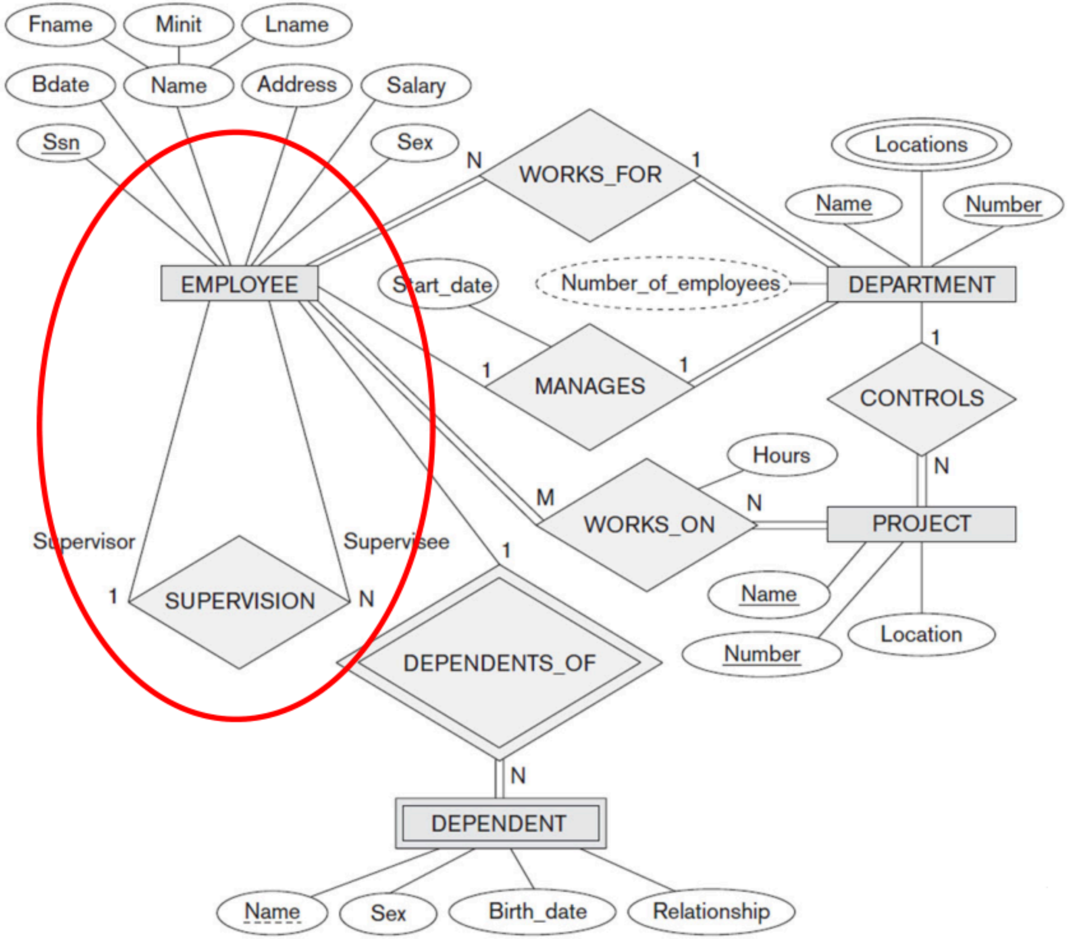
2.1 Relationship Types

Recursive Relationship Types

- In some cases, the same entity type participates more than once in a relationship type in different roles
- In such cases the role name becomes essential for distinguishing the meaning of the role that each participating entity plays
- Such relationship types are called recursive relationship types

2.1 Relationship Types

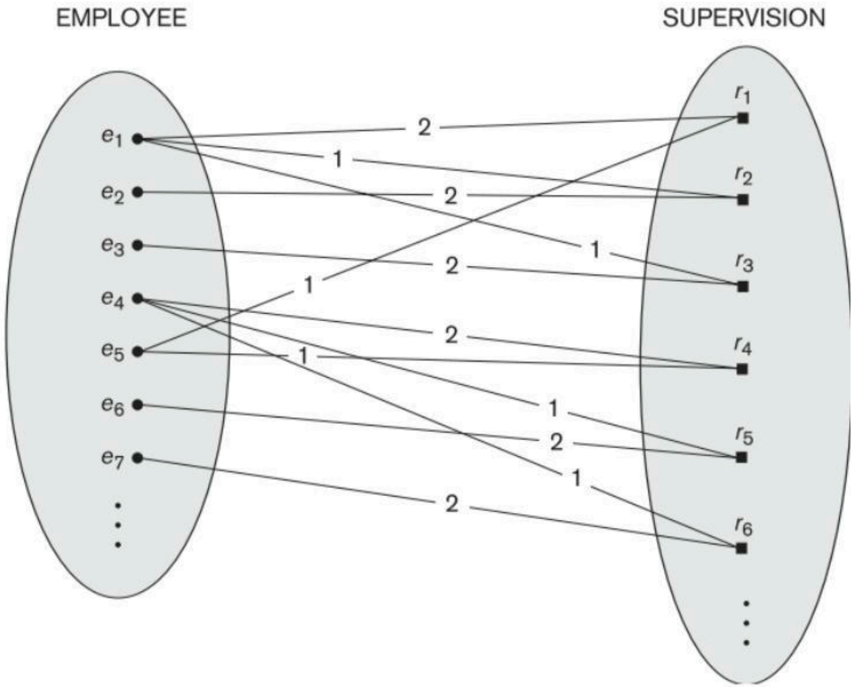
Recursive Relationship Types



2.1 Relationship Types

Example: Employee in 2 roles

- Supervisor (boss) role name 1
- Supervisee (subordinate) role name 2



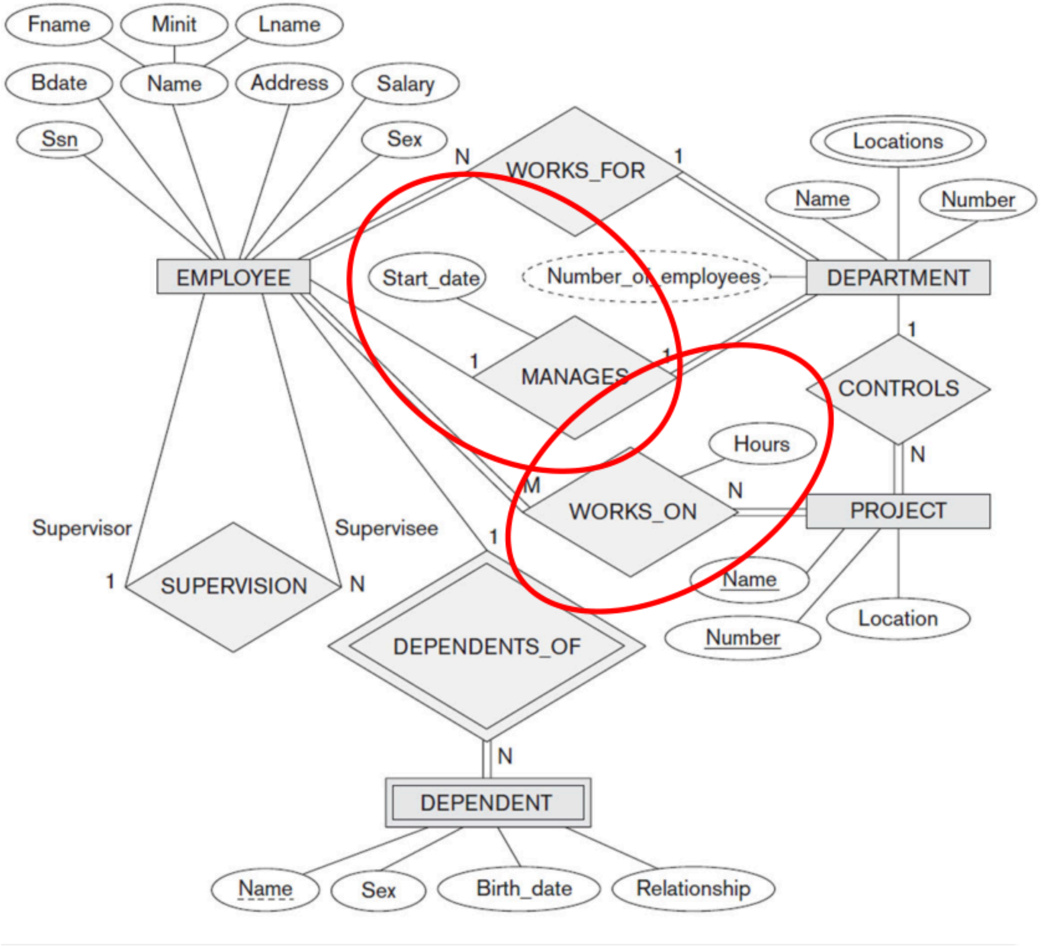
2.1 Relationship Types

Constraints

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

2.1 Relationship Types

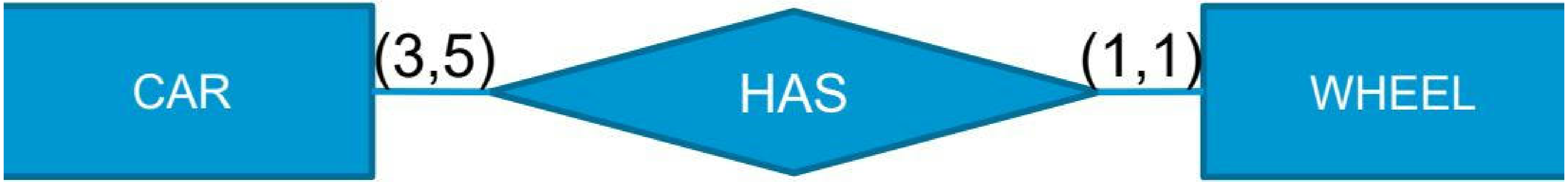
Company Example



2.1 Relationship Types

Constraints

- Cardinality
- (min,max) Notation
- Example
  - A car has at least 3 and at most 5 wheels Every wheel is associated to exactly one car

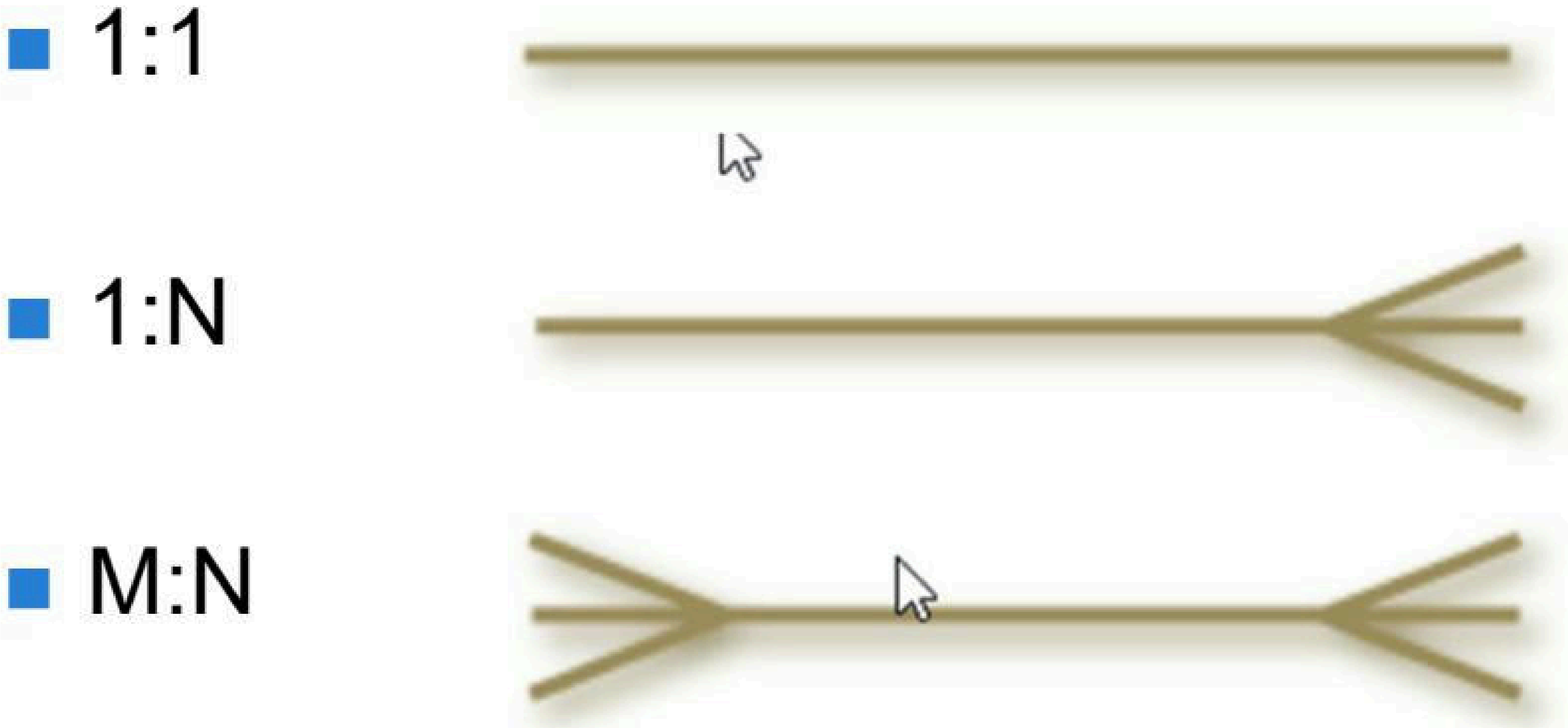


- Attention: In UML, (min,max) is placed on the opposite sites!
- Problem: General case cannot be easily implemented in RDBMS

2.1 Relationship Types

Constraints

- Other notations (e.g., in tools like draw.io)



- 1:1
- 1:N
- M:N

2.1 Relationship Types

Constraints

- 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
    - 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 relationship

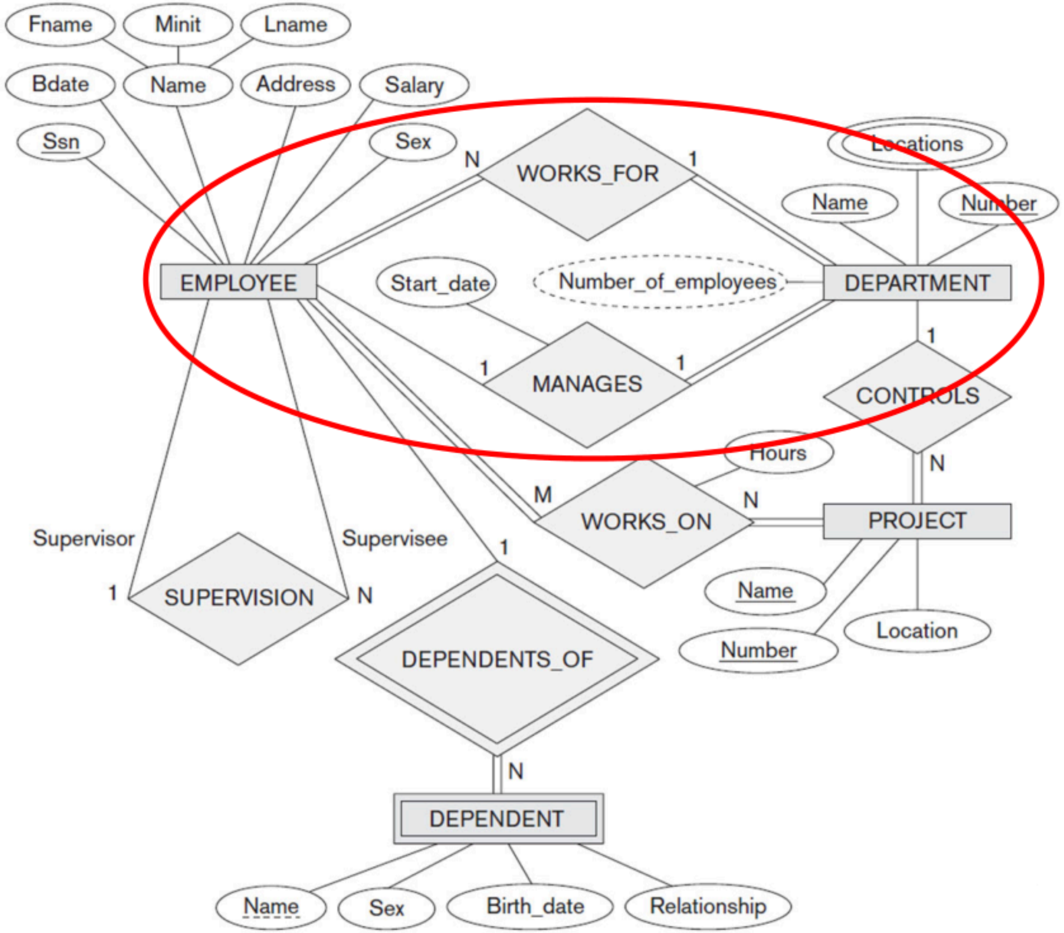
2.1 Relationship Types

Constraints

- 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

2.1 Relationship Types

Company Example



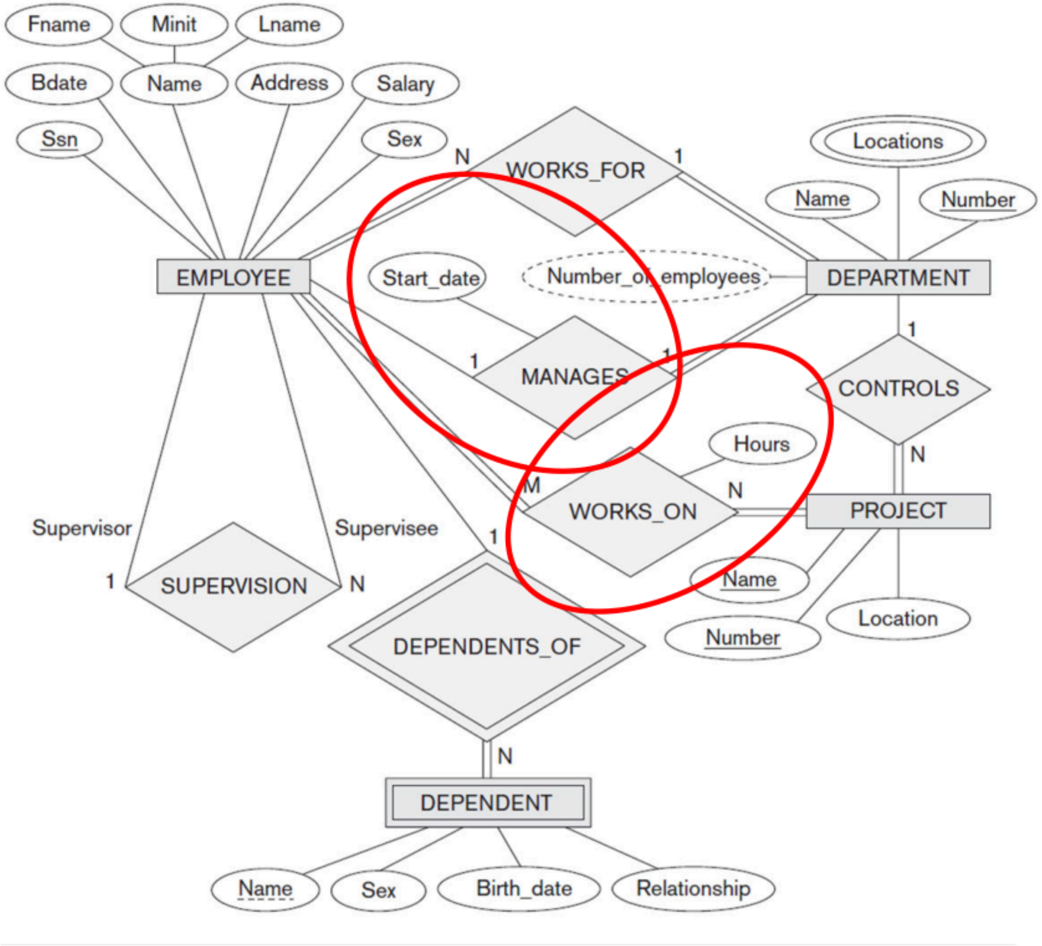
2.1 Relationship Types

Attributes

- Relationship types can also have attributes
- Notice that attributes of 1:1 or 1:N relationship types can be migrated to one of the participating entity types
- For M:N relationship types, some attributes may be determined by the combination of participating entities in a relationship instance, not by any single entity Such attributes must be specified as relationship attributes

2.1 Relationship Types

Company Example



2.1 Relationship Types

How to define them?

- Relationship between entity types
- Good naming
- More than one relationship?
  - Maybe different meanings, roles
  - Example for role: Supervisor, Supervisee
- Cardinalities
- Mandatory/optional
- Attributes for Relationship Type?

2.2 Relational Model

Constraints

- 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, primary key (entity integrity constraints), constraints on NULL, and referential integrity constraints
  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

2.2 Relational Model

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
  - 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

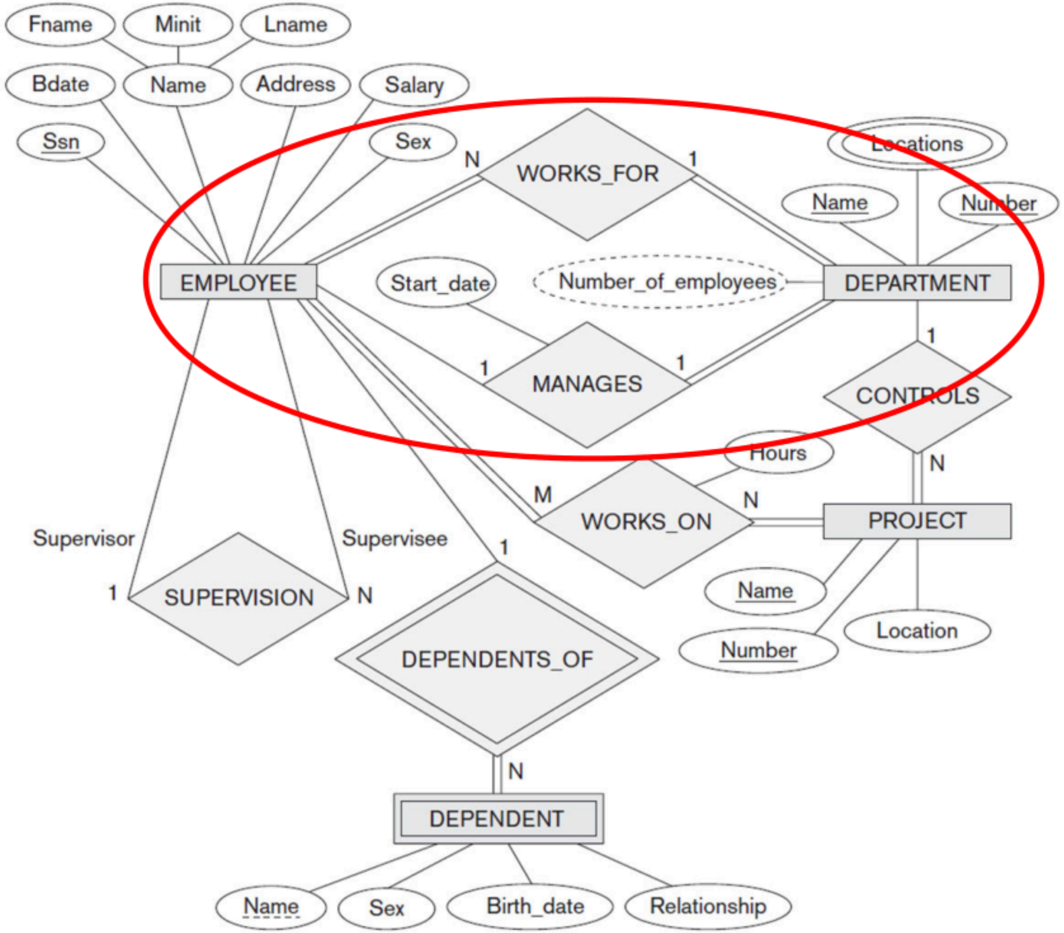
2.2 Relational Model

Referential Integrity Constraints

- A foreign key can refer to its own relation
- Foreign keys are depicted with a directed arrow: The arrowhead may point to the primary key
- All integrity constraints can be defined with the DDL, thus the DBMS can automatically enforce them

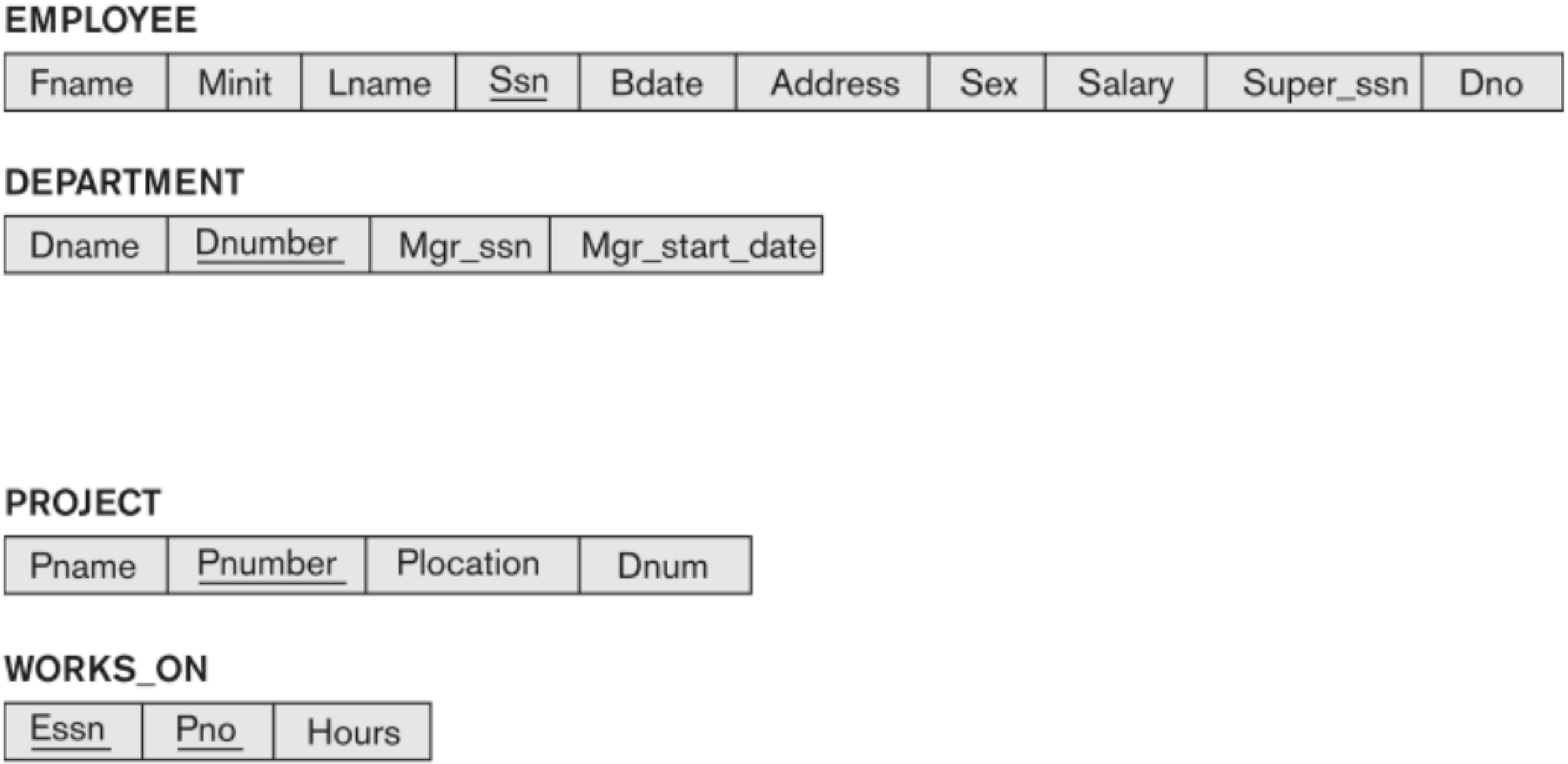
2.2 Relational Model

Company Example



2.2 Relational Model

Company Example



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

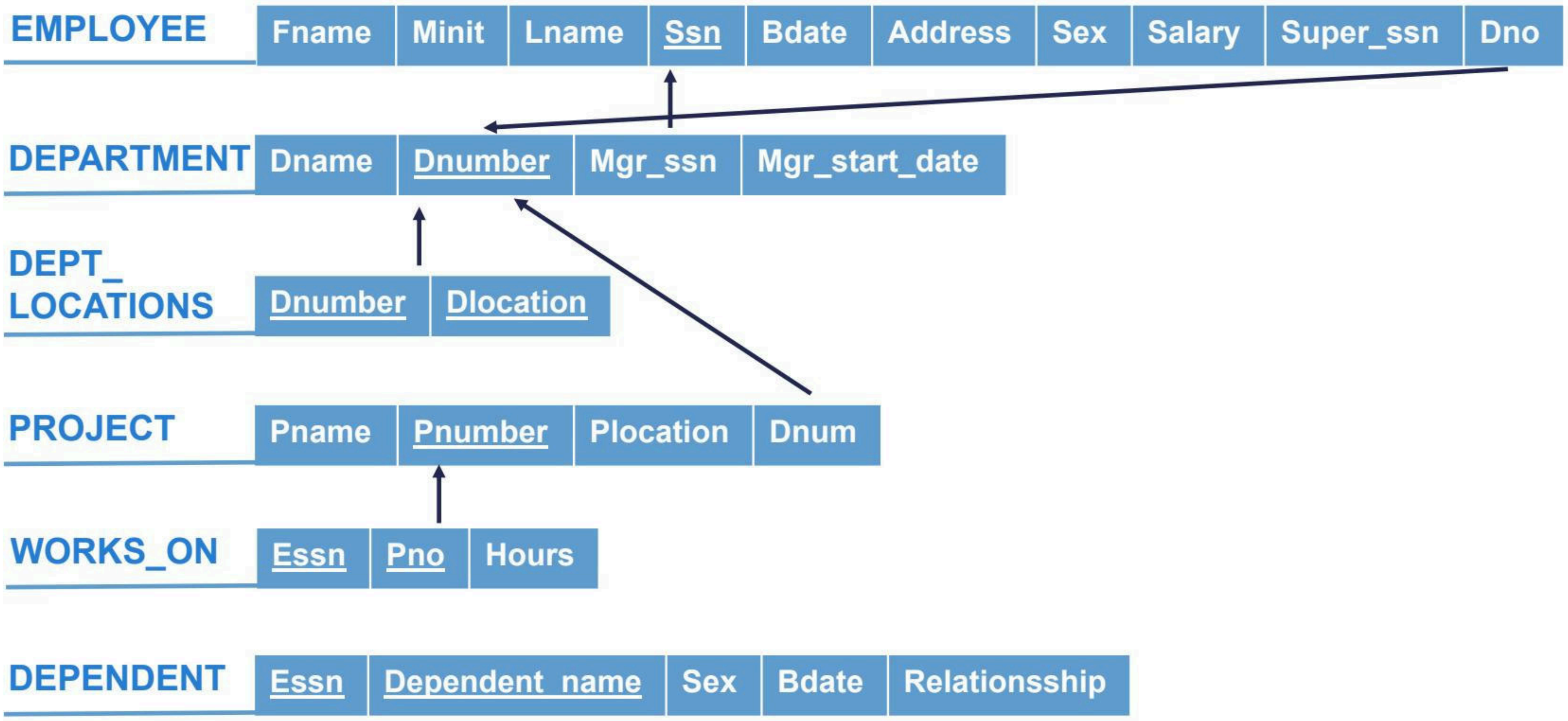
2.2 Relational Model

Notation of Foreign Keys

- There are several ways for the notation of relational schemas, especially for representing foreign keys
  - Option 1:
    - Foreign Keys can be represented with arrows
    - This notation is used in the lecture slides and in the book “Fundamentals of Database Systems” from Elmasri and Navathe
    - Advantage: Each FK-arrow connects the referencing attribute and referenced attribute, so the involved relations are obvious
    - Option 2:
      - Foreign Keys can be represented with addition (FK) within the referencing attribute
      - This notation is used in the laboratory of Mr. Ocker
      - Advantage: This notation is more readable for large, complex schemas
- Both notations are correct and may be used within the examination

2.2 Relational Model

Notation of Foreign Keys



EMPLOYEE	Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
DEPARTMENT	Dname	Dnumber	Mgr_ssn	Mgr_start_date						
DEPT_ LOCATIONS	Dnumber	Dlocation								
PROJECT	Pname	Pnumber	Plocation	Dnum						
WORKS_ON	Essn	Pno	Hours							
DEPENDENT	Essn	Dependent_name	Sex	Bdate	Relationsship					

2.2 Relational Model

Notation of Foreign Keys

EMPLOYEE: Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary | Super_ssn | Dno

DEPARTMENT: Dname | Dnumber | Mgr_ssn (FK) | Mgr_start_date

DEPT_ LOCATIONS: Dnumber (FK) | Dlocation

PROJECT: Pname | Pnumber | Plocation | Dnum (FK)

WORKS_ON: Essn | Pno (FK) | Hours

DEPENDENT: Essn | Dependent_name | Sex | Bdate | Relationsship

2.2 Relational Model

Mapping of ERM

- 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

2.2 Relational Model

Mapping of Binary 1:1

- 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. The foreign key approach
  - 2. The merged relationship approach
  - 3. The cross-reference or relationship relation approach

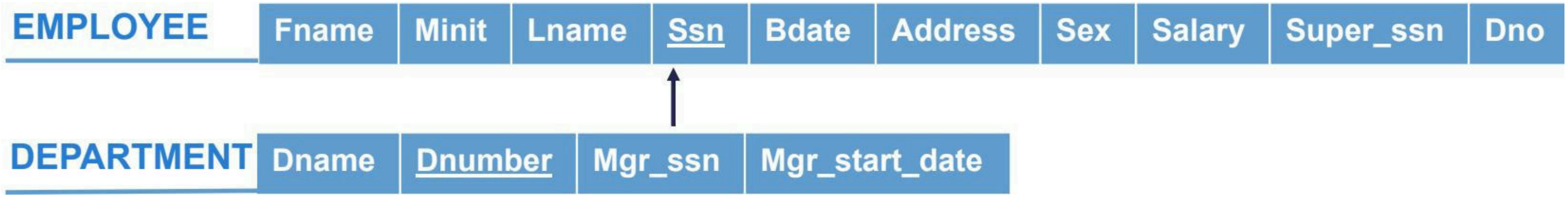
2.2 Relational Model

Mapping of Binary 1:1

1. The foreign key approach
   - Choose one of the relations S and include as a 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
   - Include all the simple attributes (or simple components of composite attributes) of the 1:1 relationship type R as attributes of S

2.2 Relational Model

Mapping of Binary 1:1



EMPLOYEE	Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

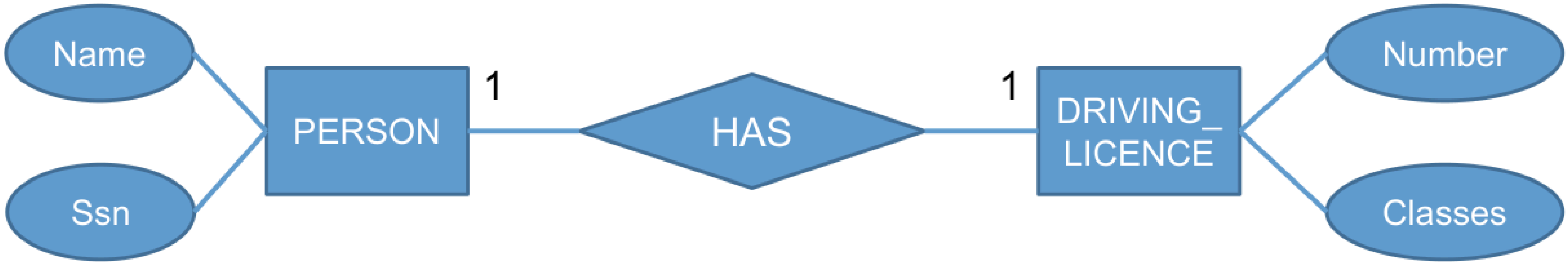
DEPARTMENT	Dname	Dnumber	Mgr_ssn	Mgr_start_date

- Mapping of relationship type MANAGES
  - DEPARTMENT serves as S
  - EMPLOYEE serves as T
- Attribute `SSN` is renamed in `MGR_SSN` in `DEPARTMENT`
- Attribute `START_DATE` is renamed in `Mgr_start_date` in `DEPARTMENT`
- It is also possible to include primary key of S as foreign key in T
- For the mapping, a UNIQUE-Constraint must be used!
  - Otherwise, an employee could manage several departments!

2.2 Relational Model

Mapping of Binary 1:1

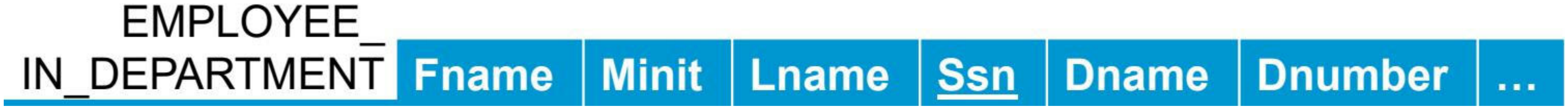
- For the mapping, a `UNIQUE`-Constraint must be used!



2.2 Relational Model

Mapping of Binary 1:1

2. Merged relation approach
   - Merge the two entity types and the relationship into a single relation
   - This is possible when both participations are total, as this would indicate that the two tables will always have the exact same number of tuples

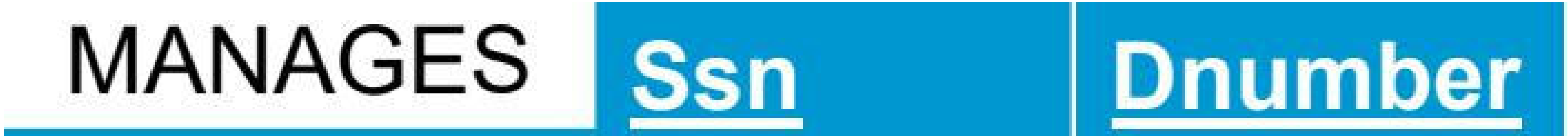


EMPLOYEE_IN_DEPARTMENT	Fname	Minit	Lname	<u>Ssn</u>	Dname	Dnumber	...

2.2 Relational Model

Mapping of Binary 1:1

3. The cross-reference or relationship relation approach
   - 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
   - This approach is required for binary M:N relationships
   - The relation R will include the primary key attributes of S and T as foreign keys to S and T
   - The primary key of R will be one of the two foreign keys, and the other foreign key will be a unique key of R



MANAGES	Ssn	Dnumber

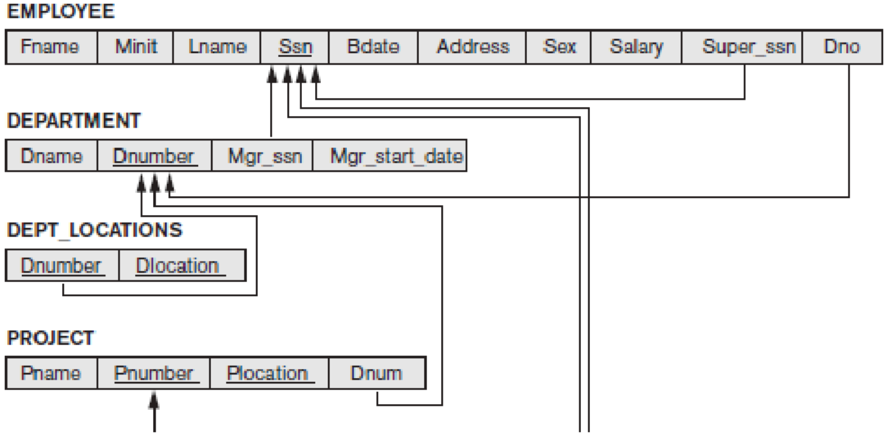
2.2 Relational Model

Mapping of Binary 1:N

- For each regular binary 1:N relationship type R, identify the relation S that represents the participating entity type at the N-side of the relationship type
- Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R
- Include any simple attributes (or simple components of composite attributes) of the 1:N relationship type as attributes of S
- An alternative approach: use the relationship relation and create a separate relation

2.2 Relational Model

Mapping of Binary 1:N



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

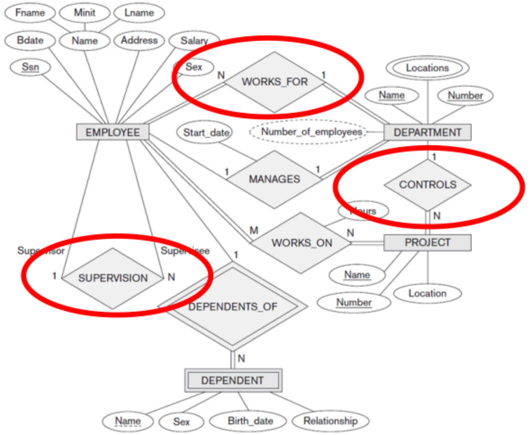
Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum



- Relationship type WORKS_FOR: Attribute Dno as foreign key in EMPLOYEE
- Relationship type SUPERVISION: Attribute Super_ssn as foreign key in EMPLOYEE
- Relationship type CONTROLS: Attribute Dnum as foreign key in PROJECT

2.2 Relational Model

Mapping of Binary 1:N - Total Participation

Total and Partial Participation should be mapped as well

- For participation definitions on the “1” side, a constraint assures the requirement
  - Total Participation 1:m → NOT NULL on FK
  - Partial Participation “0:m” → NULL on FK
- For participation definitions on the “m” side, there is a problem
  - These types (1:n vs. “1:0n”) are not distinguishable in Relational Model
  - These types of Total Participation cannot be implemented / enforced using SQL-DDL!

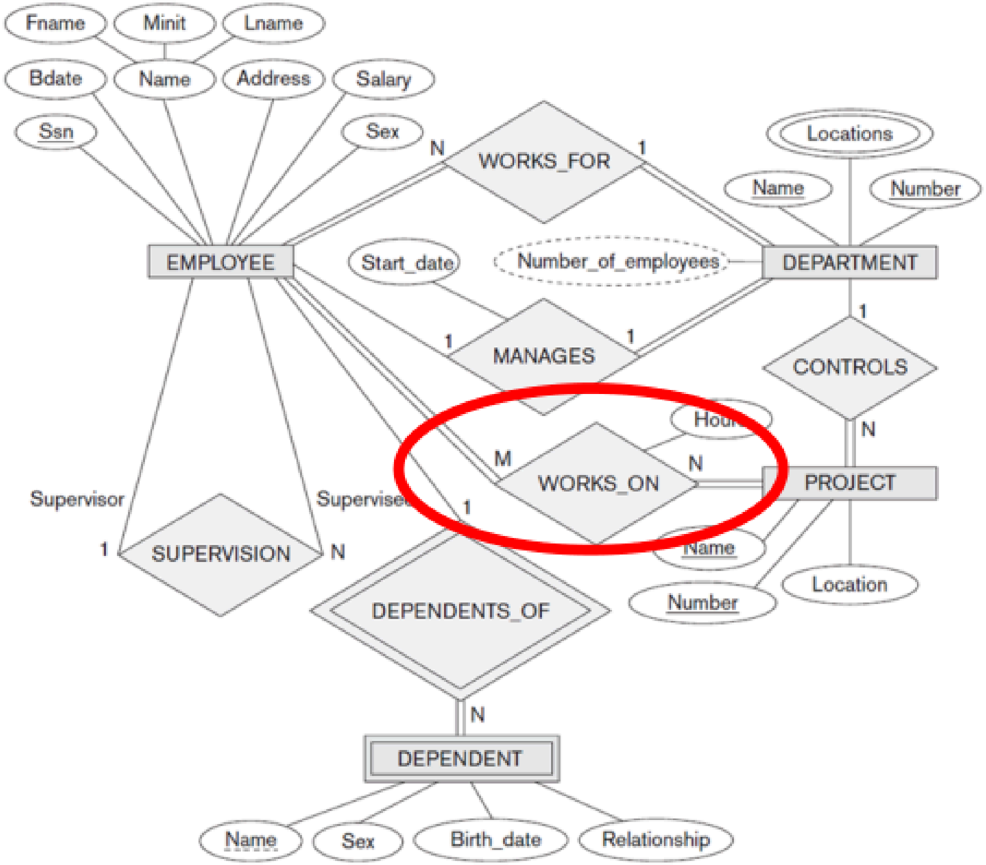
2.2 Relational Model

Mapping of Binary M:N

- For each 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
- Notice that a M:N relationship type can not be represented by a single foreign key attribute in one of the participating relations (as in 1:1 or 1:N relationship types) because of the M:N cardinality ratio

2.2 Relational Model

Mapping of Binary M:N



2.2 Relational Model

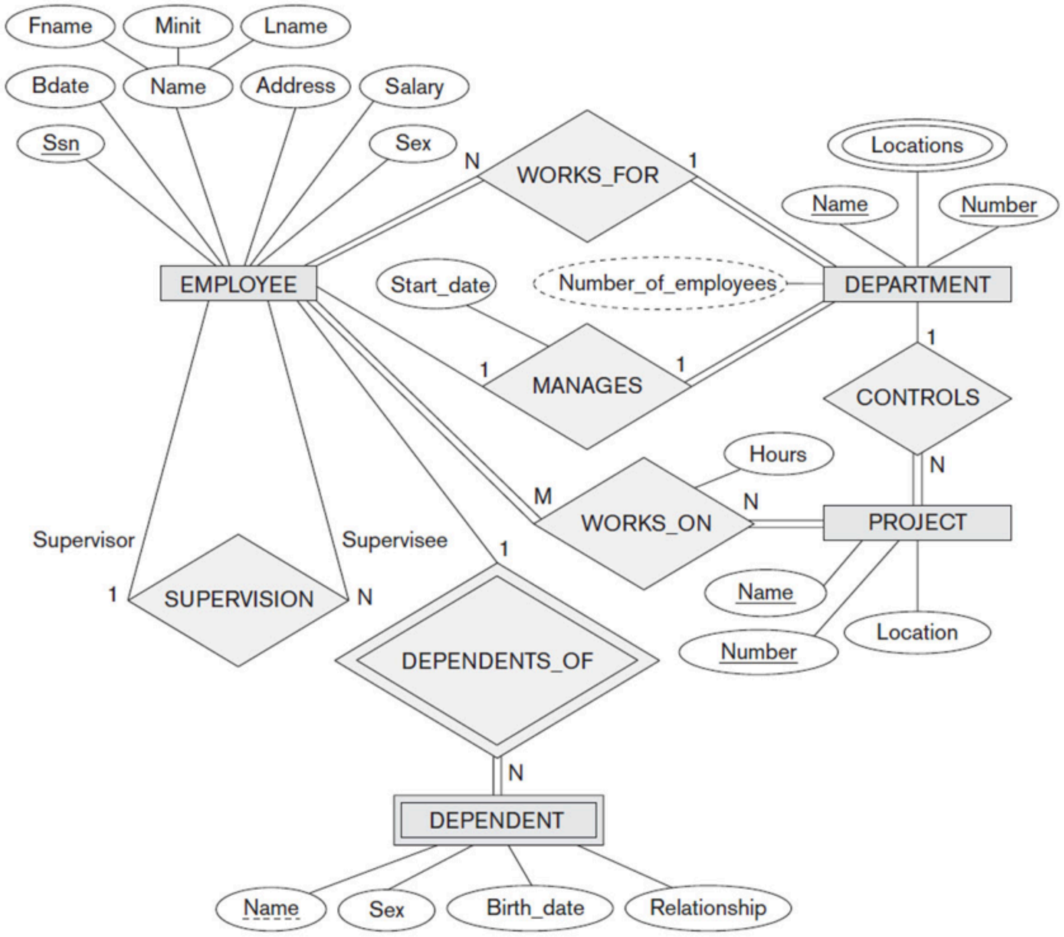
Mapping of Binary M:N

- Attribute `Ssn` is renamed in `Essn` in WORKS_ON
- Attribute `Pname` is renamed in `Pno` in DEPARTMENT
- Primary key is the combination `{Essn, Pno}`

> ℹ Info
>
> The existence dependency between `EMPLOYEE` and `PROJECT` should be specified on the foreign keys in the relation corresponding to the relationship `R` (`ON UPDATE` and `ON DELETE`)

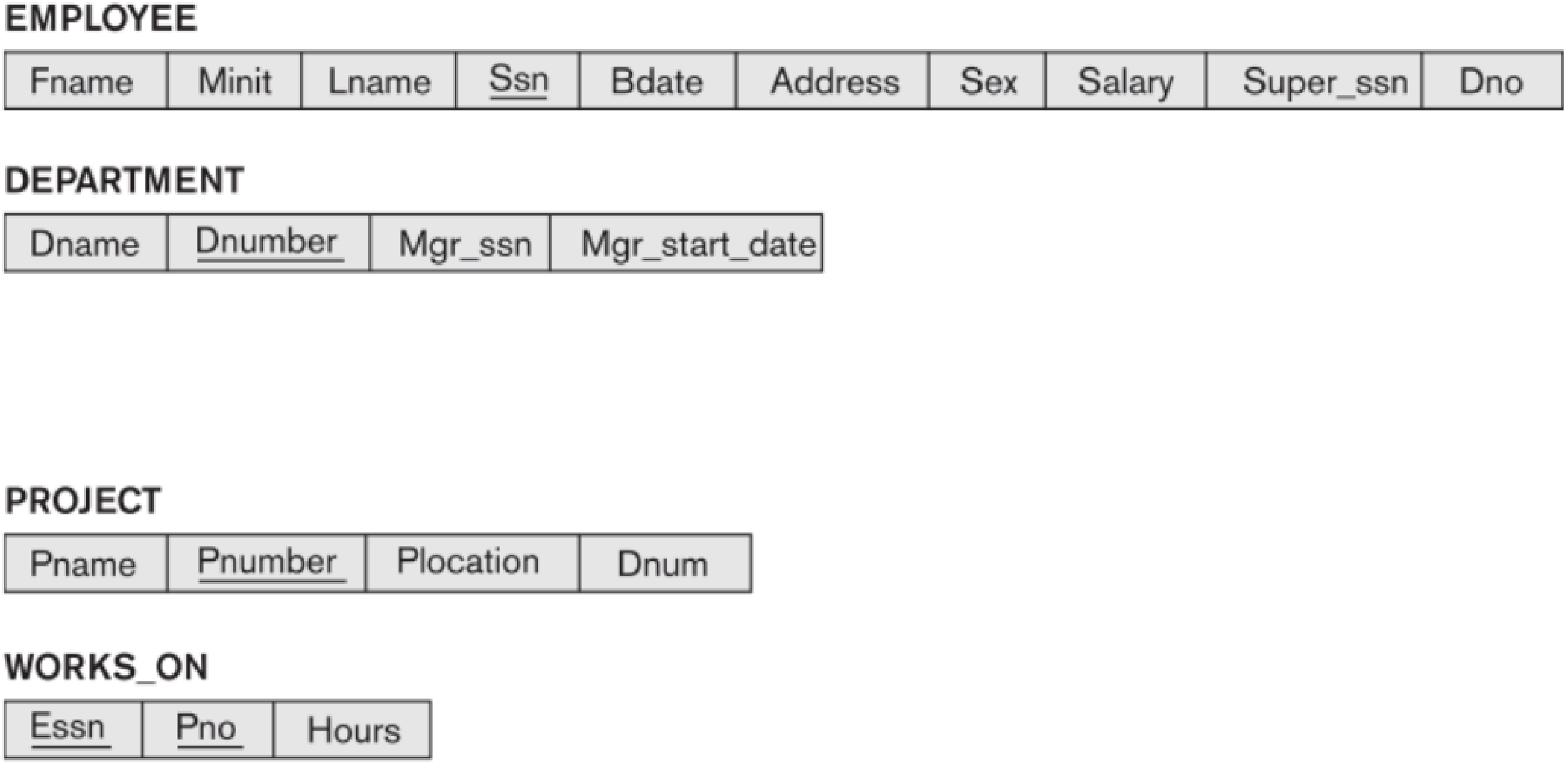
2.2 Relational Model

Example Company



2.2 Relational Model

Example Company



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

PROJECT

Pname	Pnumber	Plocation	Dnum

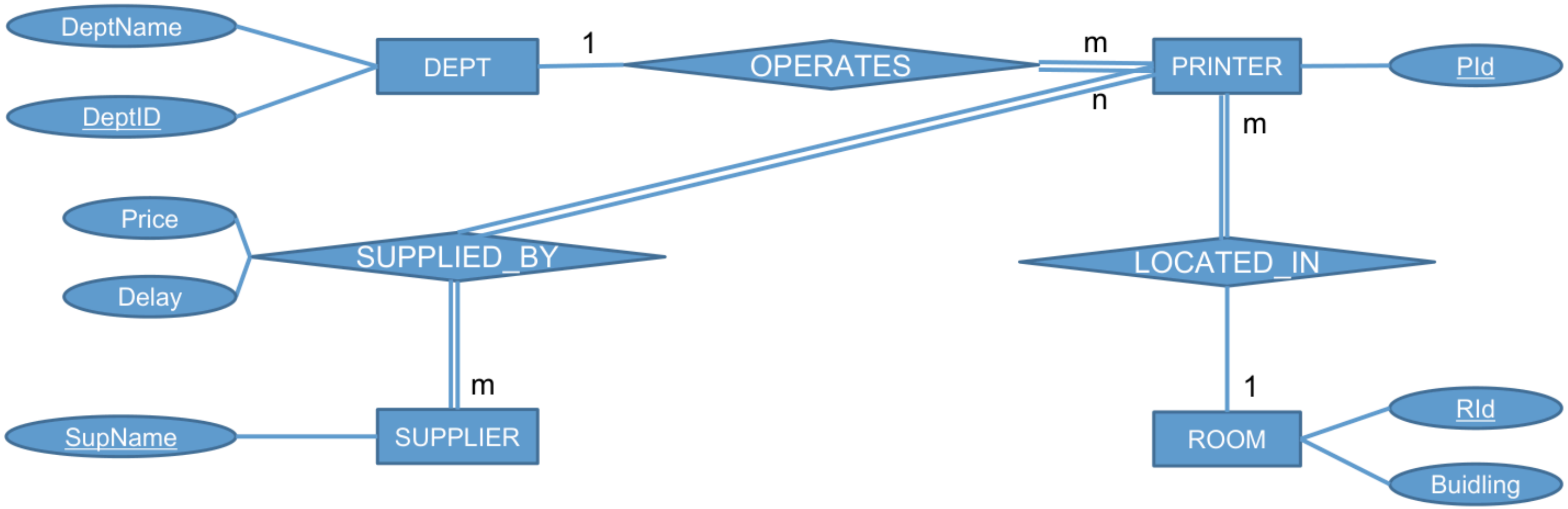
WORKS_ON

Essn	Pno	Hours

2.2 Relational Model

Assignment Office - Convert ERD to RM

Departments, identified by ID, operate a variety of printers, each located in a particular room in a particular building. Printers are supplied by a number of suppliers, identified by name, with each supplier charging a different price for a given printer, but also providing different delivery delays, measured in days. A given room can have any number of printers, including none.



2.2 Relational Model

Mapping of ERM to RM

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

2.2 Relational Model

Create Table - Constraints

- Referential integrity is specified via the FOREIGN KEY
- FK relates two tables
- Referenced table must exist already
- Referenced column must be UNIQUE
  - Best to use PK
  - If not PK: need to specify (column)

2.2 Relational Model

Create Table - Constraints 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 >)
  REFERENCES < tablename >[(< column list >)]
[< action >]
```

2.2 Relational Model

Create Table - Constraints Syntax

- Example column constraint:

```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));
```

2.2 Relational Model

Create Table - Constraints Syntax

- Example table constraint:

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 ) );
```

2.2 Relational Model

Create Table - Constraints

- `<action>`:
  - How to react on changes to the referenced table

- The default action: reject the update operation (RESTRICT option)

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

2.2 Relational Model

Create Table - Constraints

- 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
  - For relations that represent weak entity types

2.2 Relational Model

Create Table - Constraints

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 ON UPDATE CASCADE
  );
```

2.2 Relational Model

ALTER TABLE

- For modifying an existing relation
  - COLUMN: ADD, DROP, MODIFY
  - CONSTRAINT: ADD, DROP
  - TABLE: RENAME

2.2 Relational Model

ALTER TABLE - Column

Syntax for altering a table:

```sql
ALTER TABLE < relationname > . . .
```

```sql
ADD [ COLUMN ] < column > < type >
        [ < col\_constraint > [ . . .]
```

```sql
  DROP [COLUMN] <column> [RESTRICT | CASCADE]
```

```sql
  RENAME COLUMN <column> TO <new_column>
```

2.2 Relational Model

ALTER TABLE - Column

- Syntax for altering a table:

```sql
ALTER TABLE < relationname > . . .
```

Modification of columns vendor-specific: Oracle:

```sql
... MODIFY < column > < type > [< col\_constraints > [...]]
```

MySQL:

```sql
... CHANGE [ COLUMN ] < column > < type > ...
```

2.2 Relational Model

ALTER TABLE - Column

SQL

```sql
ALTER TABLE COMPANY.EMPLOYEE ADD COLUMN Job TEXT(12);
```

- Inserting values for the new column:
  - Default is `NULL` ⟶ `NOT NULL` constraint is not allowed
  - Using default clause
  - Using `UPDATE` individually on each tuple

2.2 Relational Model

ALTER TABLE - Constraints

Syntax for adding a new constraint:

```sql
ALTER table ADD < tableconstraint > ;
```

- Add a foreign key (instead of within create table statement):

```sql
   ALTER TABLE DEPARTMENT
   ADD CONSTRAINT DEPTMGRFK  FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn)
   ON DELETE RESTRICT ON UPDATE CASCADE;
```

2.2 Relational Model

ALTER TABLE - Constraints

Syntax for dropping an existing constraint:

```sql
ALTER TABLE < tablename > < alterstatement >
```

```sql
< alterstatement > ::=
      DROP PRIMARY KEY |
      DROP FOREIGN KEY < keyname > |
```

2.2 Relational Model

ALTER TABLE - Rename

- Syntax for renaming an existing table:
- Oracle, MySQL:

```sql
RENAME TABLE < relationname > TO < newrelationname >
```

- PostgreSQL, MySQL:

```sql
ALTER TABLE < name > RENAME TO < new_name >
```

2.2 Relational Model

Homework

- Company Example
  - Implement all relationship types from the ERM in your database
  - Think also about the cardinalities and participation constraints of these relationship types
  - What should be the behavior of these relations if data changes?
  - Try SQL statements for inserting, updating, and deleting data
- Implement the printer example in your database
- Think about your own, individual example (e.g., contact list)
  - Implement all relationship types from the ERM in your database
  - Think also about the cardinalities and participation constraints
  - What should be the behavior of these relations if data changes?
  - Try SQL statements for inserting, updating, and deleting data

3. License Notice

3.1 Attribution

The basis of this course stems from: Professor Dr. Ulrike Herster The following license applies to all unmodified and modified material used throughout this course.

! License