AALBORG UNIVERSITY

Database System (SW5)

6. ER Model (part 2)

Tiantian Liu
Department of Computer Science
Aalborg University
Fall 2025

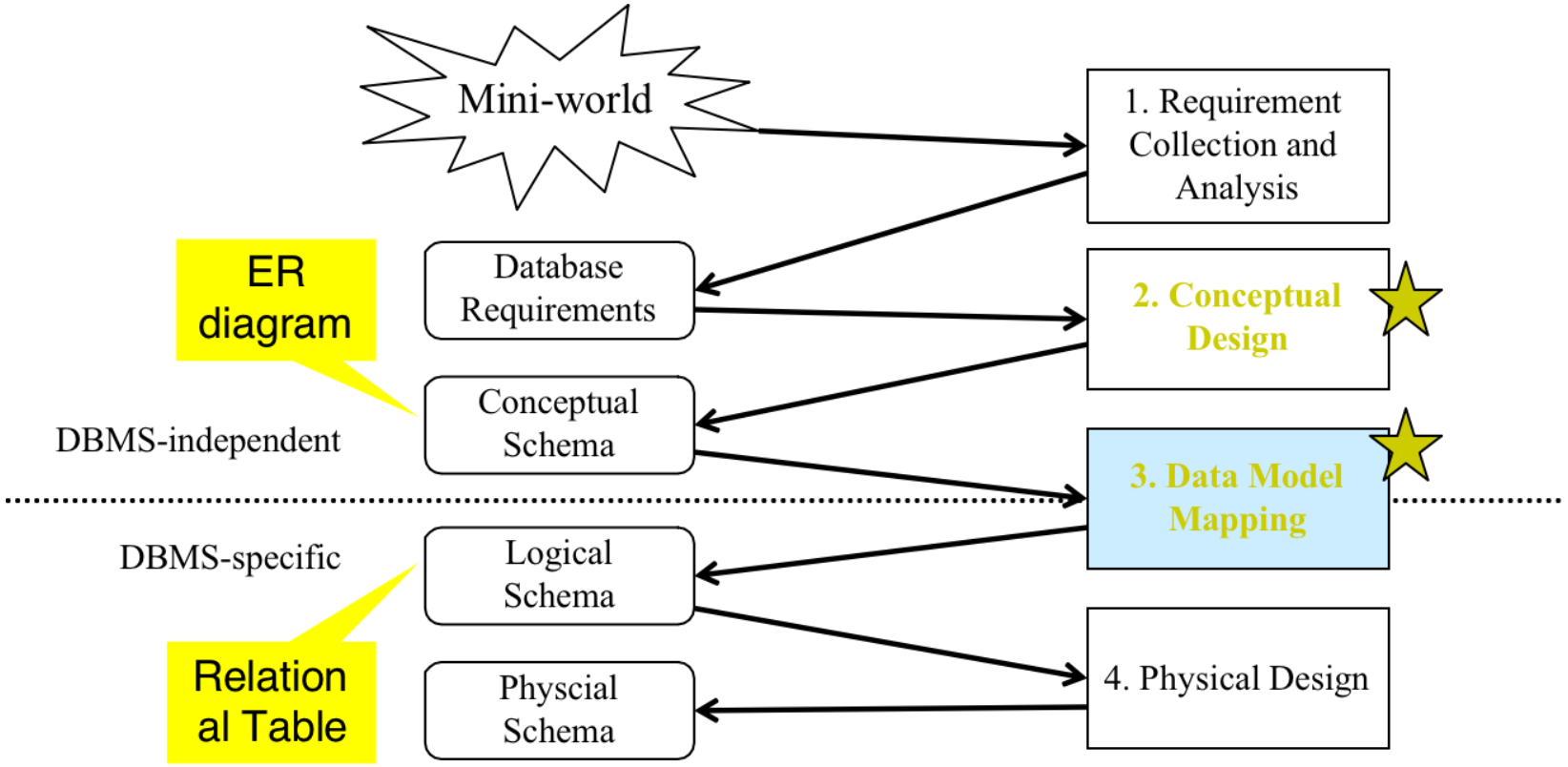
Agenda

AALBORG UNIVERSITY

- Map entity types to relations
- Map relationship types to relations
- Map complex cases to relations
- Relational modelling of Specialization

Data Modelling Process

AALBORG UNIVERSITY



Mapping an ER Model to Relations

AALBORG UNIVERSITY

- A set of relations are created

M1: Regular entity types

M2: Weak entity types

M3: Binary M:N relationship types

M4: Binary 1:N relationship types

M5: Binary 1:1 relationship types

M6: Complex relationship types

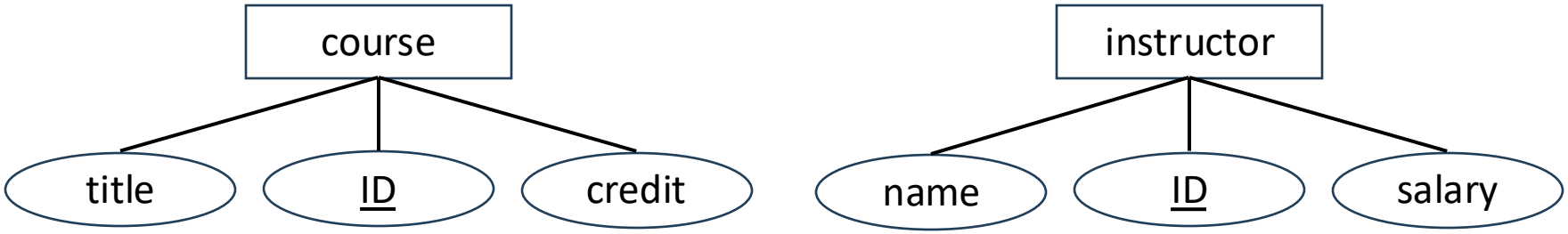
M7: Complex attributes

M1: Regular Entity Types

AALBORG UNIVERSITY

- **Create** a relation for each regular entity type.
  - The relation has one column for each simple attribute of its corresponding entity type.
  - The primary key for the relation is the primary key of the entity type.
  - If there are no attributes other than the primary key and the entity participates totally in a relationship, then the relation can be eliminated.

M1: Example

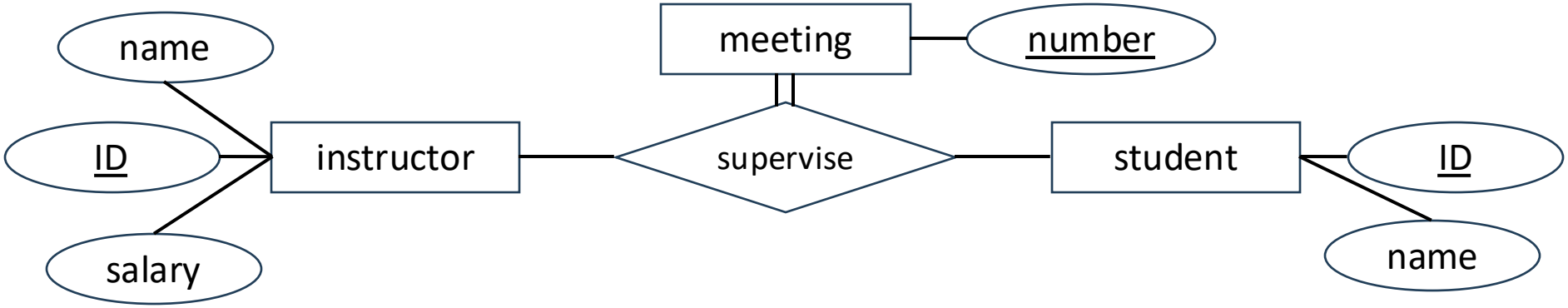


- ***course(<u>ID</u>, title, credit)***
- ***instructor(<u>ID</u>, name, salary)***

The order of attributes doesn't matter in this context!

The attribute domains don't matter either.

M1: Example



- ***instructor(<u>ID</u>, name, salary)***
- ***student(<u>ID</u>, name)***

In this case, *meeting* relation can be eliminated.

We can maintain the number in *supervise* relation.

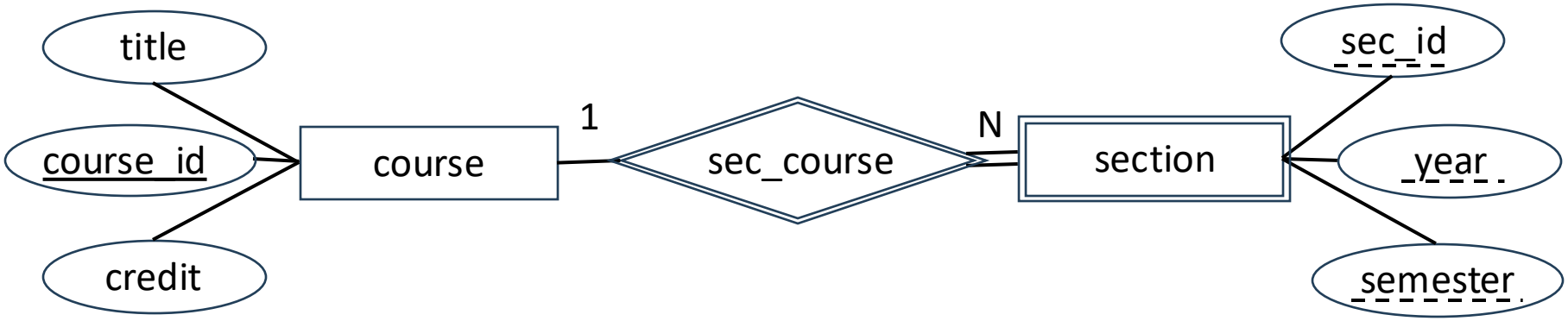
M2: Weak Entity Types

AALBORG UNIVERSITY

- **Create** a relation for each weak entity type.
  - The relation has one column for each simple attribute of its corresponding entity type.
  - The relation also includes the primary key from the strong entity type it depends as the foreign key.
  - The relation's primary key is composed of the foreign key and part of its own attributes.
  - Weak entity types and their identifying relationship types can always be merged.
  - If the weak entity type is related to the strong entity type via 1-1 relationship, the relation can be eliminated. The attributes of the weak entity type will join the table created for the strong entity type.

M2: Example

AALBORG UNIVERSITY

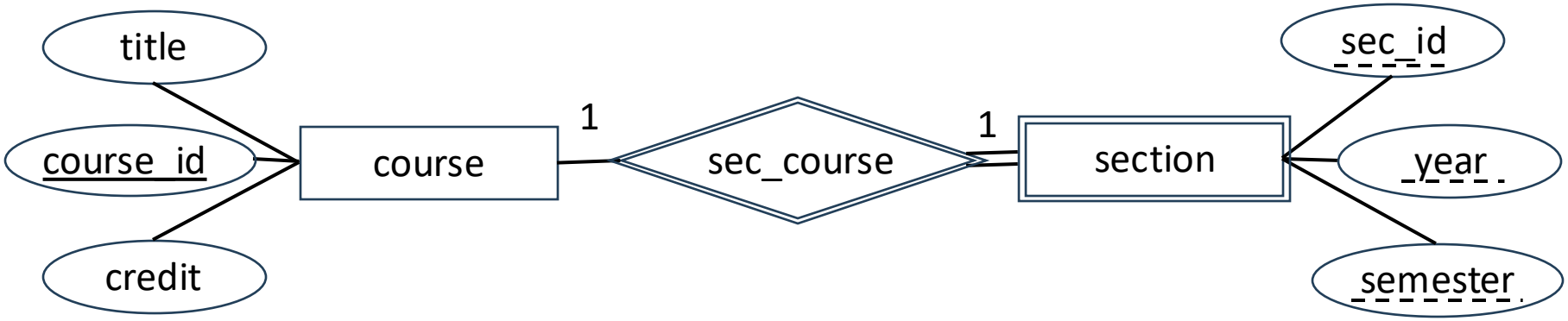


- *course*(*course_id*, *title*, *credit*)
- ***section*(*course_id-->course, sec_id, year, semester*)**

The primary key of *section* includes *course*'s primary key.

The *section* entity type and *sec_course* relationship type are merged.

M2: Example



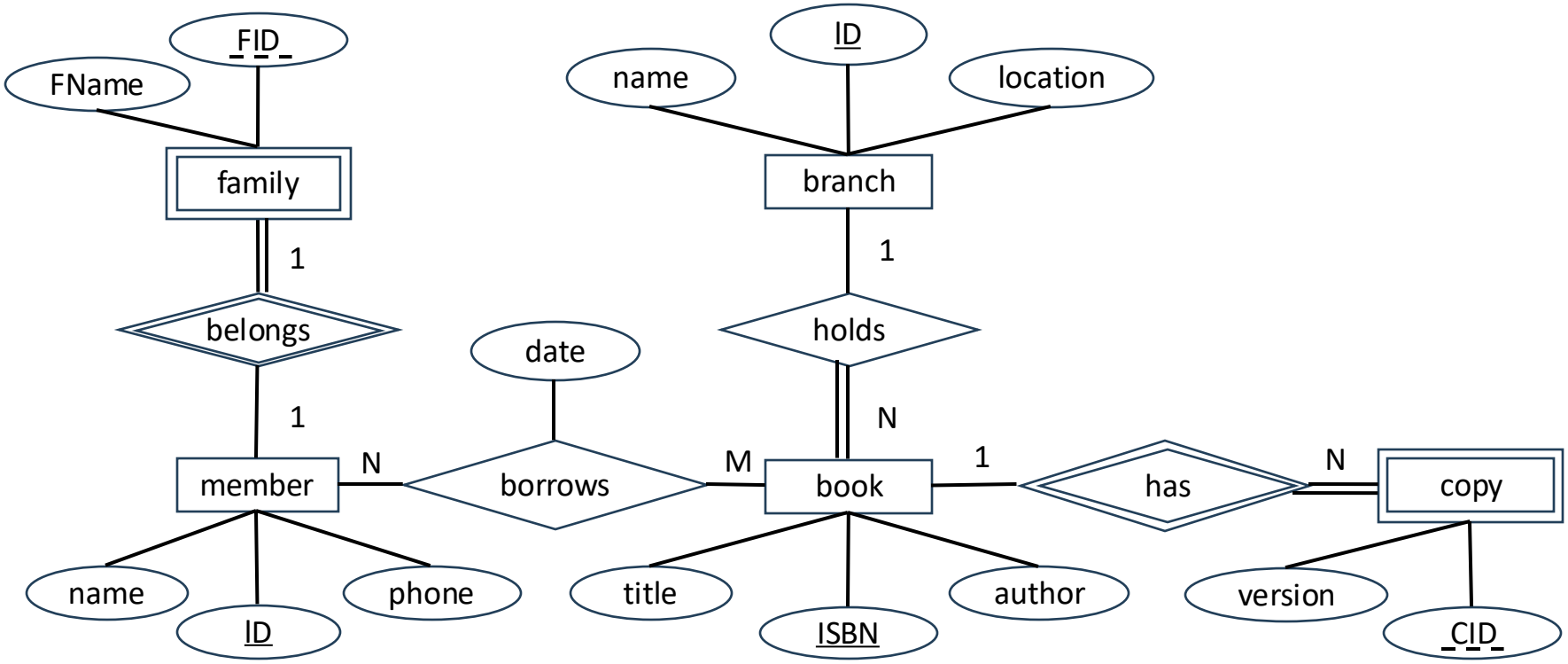
- ***course(course_id, title, credit, sec_id, year, semester)***

The *section* is related to the strong entity type via 1-1 relationship, the relation can be eliminated.

The attributes of the weak entity type join the relation created for the strong entity type.

Exercise 1

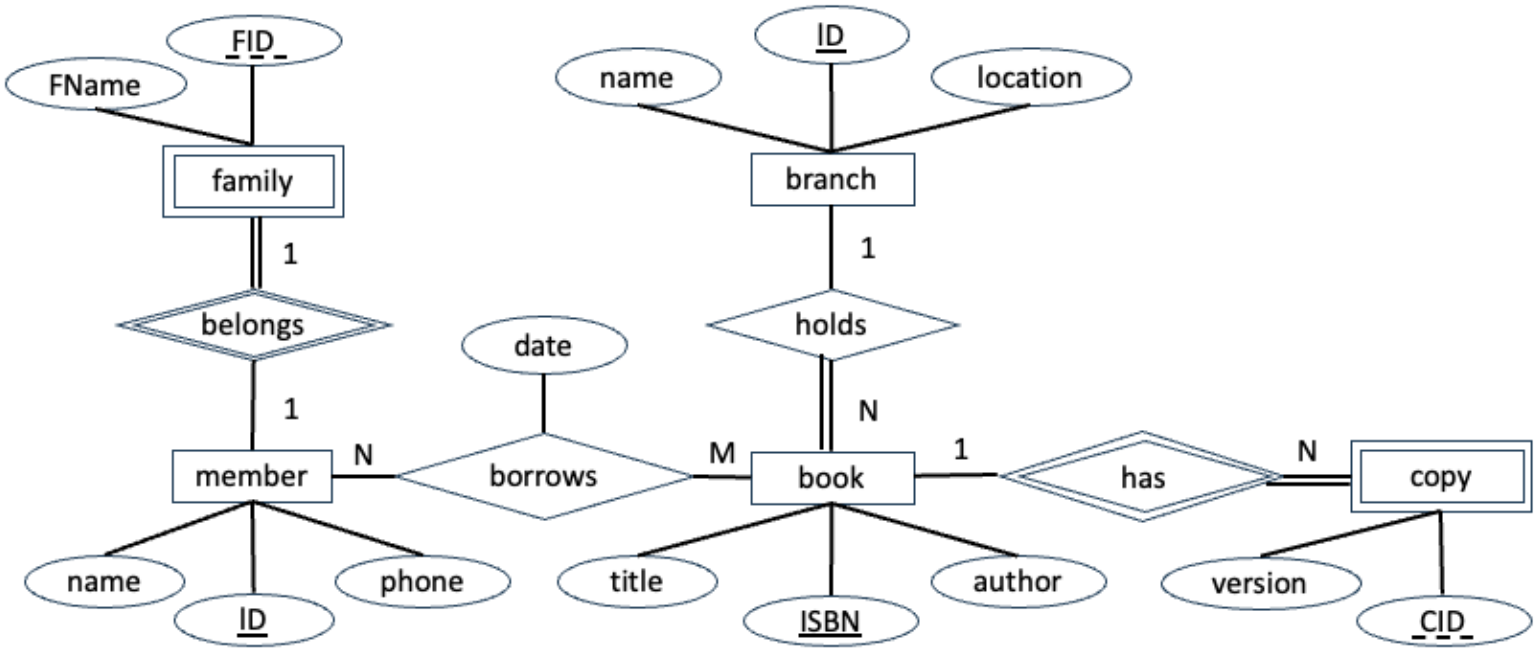
- Create relations for regular/weak entity types.



Exercise 1--Solution

AALBORG UNIVERSITY

- *branch*(*ID*, *name*, *location*)
- *book*(*ISBN*, *title*, *author*)
- *member*(*ID*, *name*, *phone*, *FID*, *FName*)
- *copy*(*ISBN-->book*, *CID*, *version*)



Agenda

AALBORG UNIVERSITY

- Map entity types to relations
- Map relationship types to relations
- Map complex cases to relations
- Relational modelling of Specialization

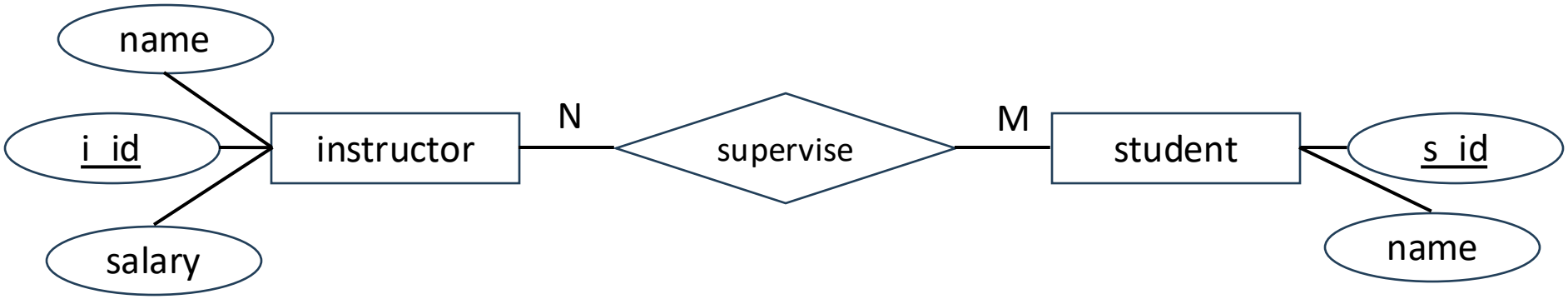
M3: Binary M:N relationship types

AALBORG UNIVERSITY

- **Create** a relation for each binary M:N relationship type.
  - The relation's columns are the primary keys of the participating entity types. These are also foreign keys.
  - The relation's primary key is the union of the primary keys of the participating entity types.
  - Include also columns for each of the simple attributes of the relationship type.

M3: Example

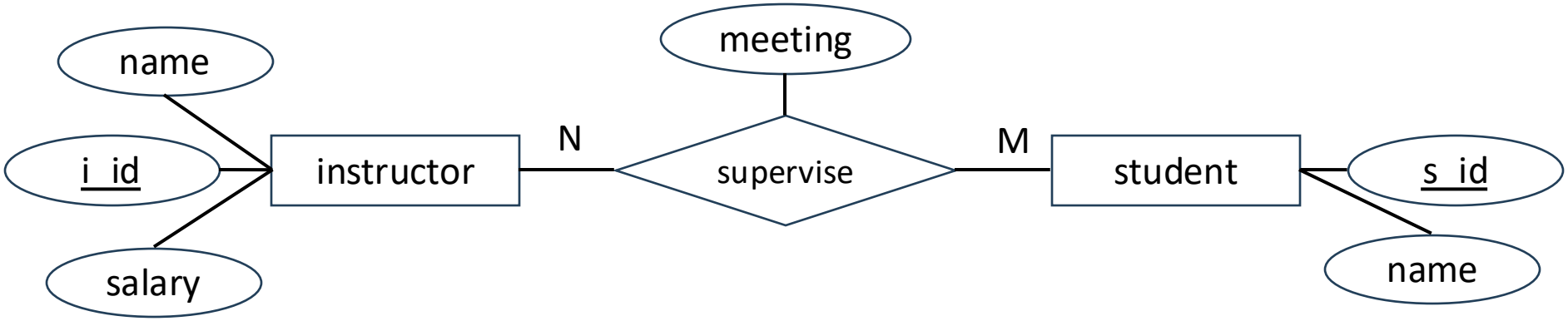
AALBORG UNIVERSITY



- *instructor*(*i_id*, *name*, *salary*)
- *student*(*s_id*, *name*)
- ***supervise*(*i_id-->instructor, s_id-->student*)**

The relation's primary key is the union of the primary keys of the participating entity types.

M3: Example



- ***supervise(i_id-->instructor, s_id-->student, meeting)***

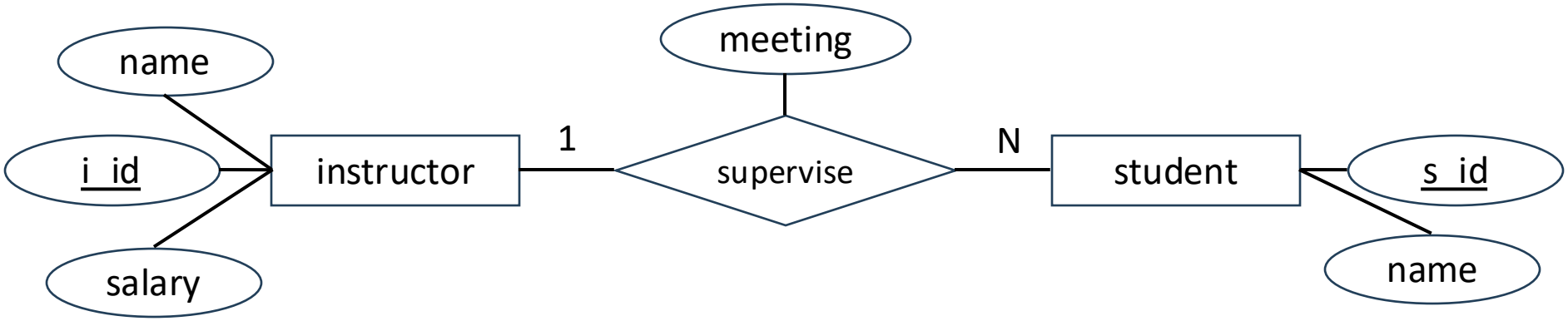
Include columns for each of the simple attributes of the relationship type.

M4: Binary 1:N relationship types

AALBORG UNIVERSITY

- For each regular binary 1:N relationship type, there are two options.
- **Option 1**: **Create** a separate table for the relationship.
  - Primary key is simply the primary key of the "many" side (N-side).
  - Add columns for each of the simple attributes of the relationship type.
- **Option 2**: **Extend** the N-side table. (Improved)
  - Include the primary key from the 1-side table as a foreign key;
  - Add columns for each of the simple attributes of the relationship type.

M4: Option 1--Example



- *instructor*(*i_id*, *name*, *salary*)
- *student*(*s_id*, *name*)
- ***supervise*(*s_id-->student*, *i_id-->instructor*, *meeting*)**

- **Create** a separate table for the relationship.
  - Primary key is simply the primary key of the "many" side (N-side).
  - Add columns for each of the simple attributes of the relationship type.

M4: Option 1--Example

AALBORG UNIVERSITY

i_id	*name*	*salary*
10101	Srinivasan	65000
12121	Wu	90000
15151	Mozart	40000
22222	Einstein	95000
32343	EI Said	60000
33456	Gold	87000
45565	Katz	75000
58583	Califieri	62000
76543	Singh	80000
76766	Crick	72000
83821	Brandt	92000
98345	Kim	80000

instructor

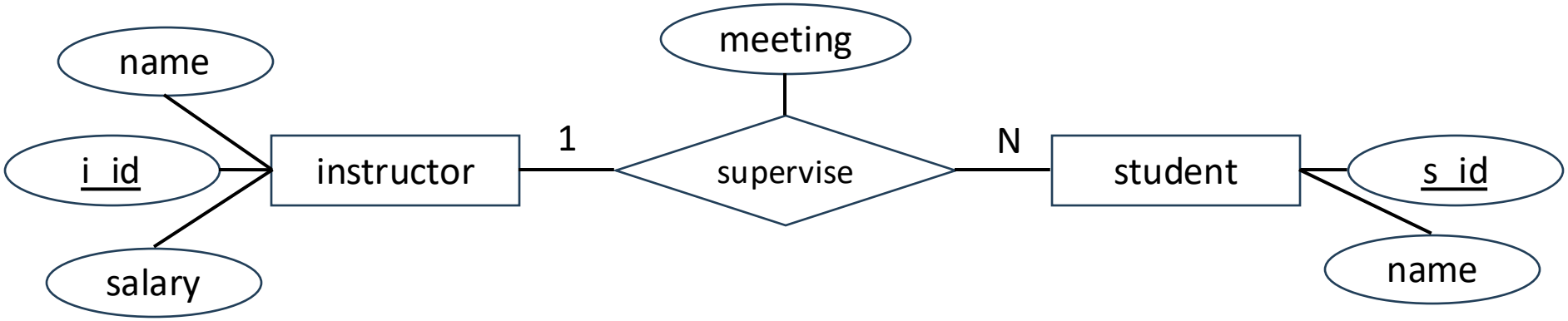
s_id	*name*
1001	Alice
1002	Anna
1003	Bob
1004	Cathy
1005	Christian
1006	David

student

s_id	*i_id*	*meeting*
1001	12121	10
1002	33456	9
1003	22222	12
1004	22222	8
1005	76766	10
1006	45565	11

supervise

M4: Option 2--Example



- *instructor*(*i_id, name, salary*)
- ***student(s_id, name, i_id-->instructor, meeting)***

- **Extend** the N-side table. (Improved)
  - Include the primary key from the 1-side table as a foreign key;
  - Add columns for each of the simple attributes of the relationship type.

M4: Option 2--Example

AALBORG UNIVERSITY

i_id	*name*	*salary*
10101	Srinivasan	65000
12121	Wu	90000
15151	Mozart	40000
22222	Einstein	95000
32343	EI Said	60000
33456	Gold	87000
45565	Katz	75000
58583	Califieri	62000
76543	Singh	80000
76766	Crick	72000
83821	Brandt	92000
98345	Kim	80000

instructor

s_id	*name*
1001	Alice
1002	Anna
1003	Bob
1004	Cathy
1005	Christian
1006	David

student

+

s_id	*i_id*	*meeting*
1001	12121	10
1002	33456	9
1003	22222	12
1004	22222	8
1005	76766	10
1006	45565	11

supervise

s_id	*name*	*i_id*	*meeting*
1001	Alice	12121	10
1002	Anna	33456	9
1003	Bob	22222	12
1004	Cathy	22222	8
1005	Christian	76766	10
1006	David	45565	11

student

merged

Extend N-side

M4: Option 2--Example

AALBORG UNIVERSITY

i_id	*name*	*salary*	*s_id*	*meeting*
10101	Srinivasan	65000	null	null
12121	Wu	90000	1001	10
15151	Mozart	40000	null	null
22222	Einstein	95000	1003	12
22222	Einstein	95000	1004	8
32343	EI Said	60000	null	null
33456	Gold	87000	1002	9
45565	Katz	75000	1006	11
58583	Califieri	62000	null	null
76543	Singh	80000	null	null
76766	Crick	72000	1005	10
83821	Brandt	92000	null	null
98345	Kim	80000	null	null

instructor

s_id	*name*
1001	Alice
1002	Anna
1003	Bob
1004	Cathy
1005	Christian
1006	David

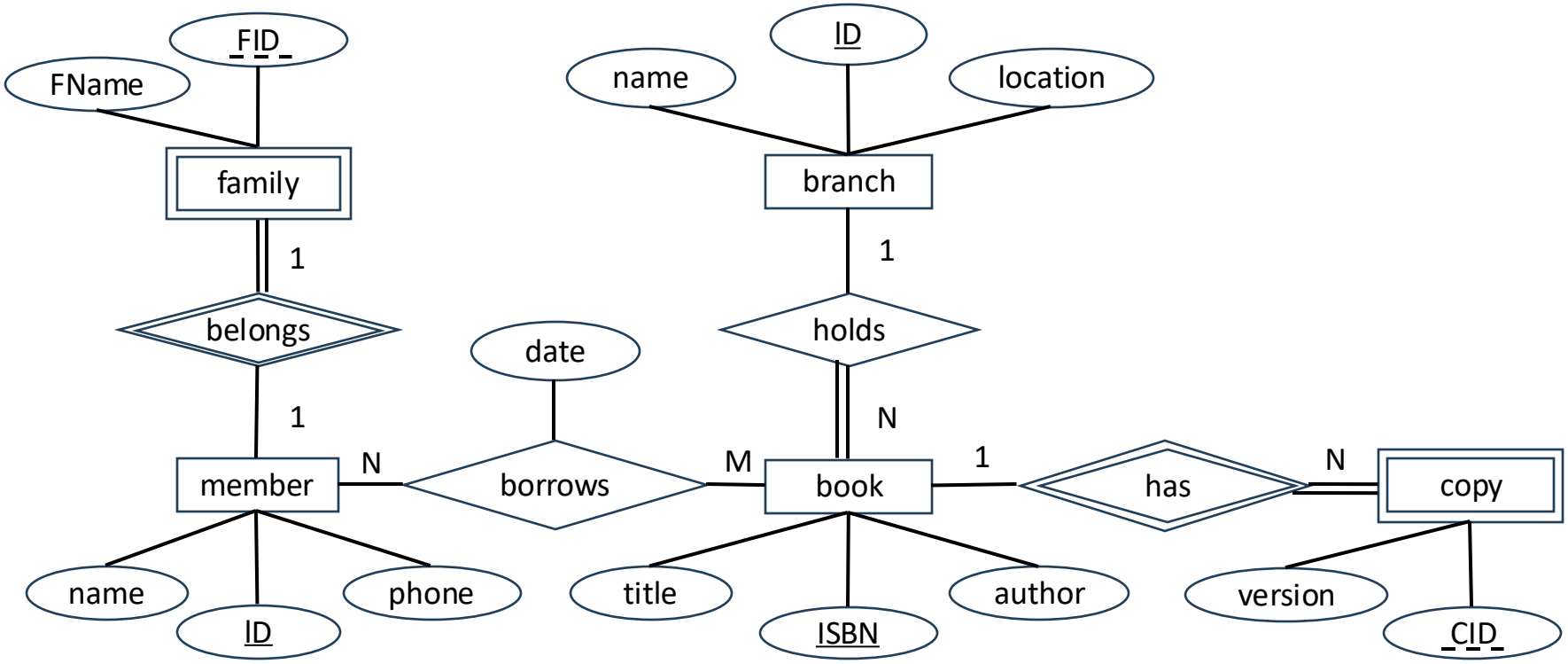
student

Why not 1-side ?

- Update anomaly:
  - What happens when Einstein's salary updated?
- Deletion anomaly
  - What happens if Einstein no longer taught?
- Insert anomaly:
  - Curie is new and does not yet supervise any students

Exercise 2

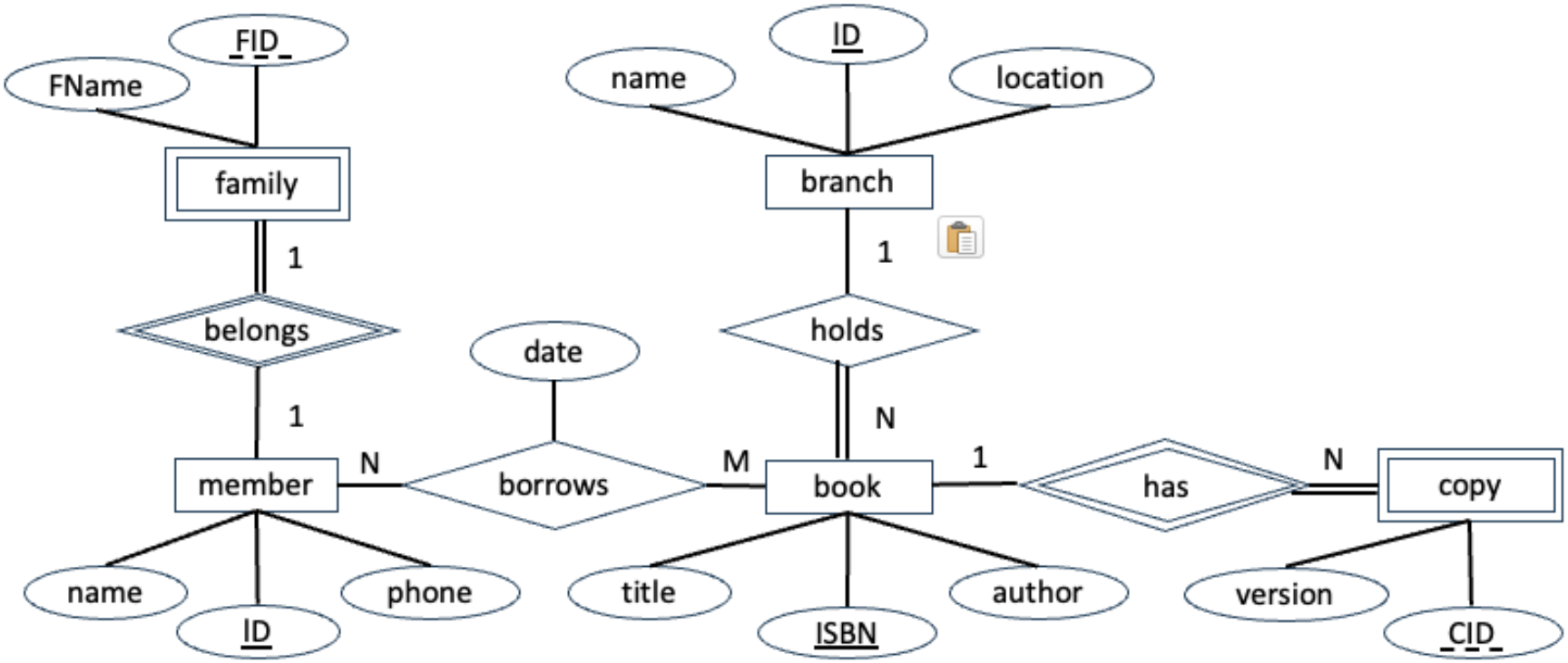
- Create relations for relationship types or revise the existing relations for entity types.



Exercise 2--Solution

AALBORG UNIVERSITY

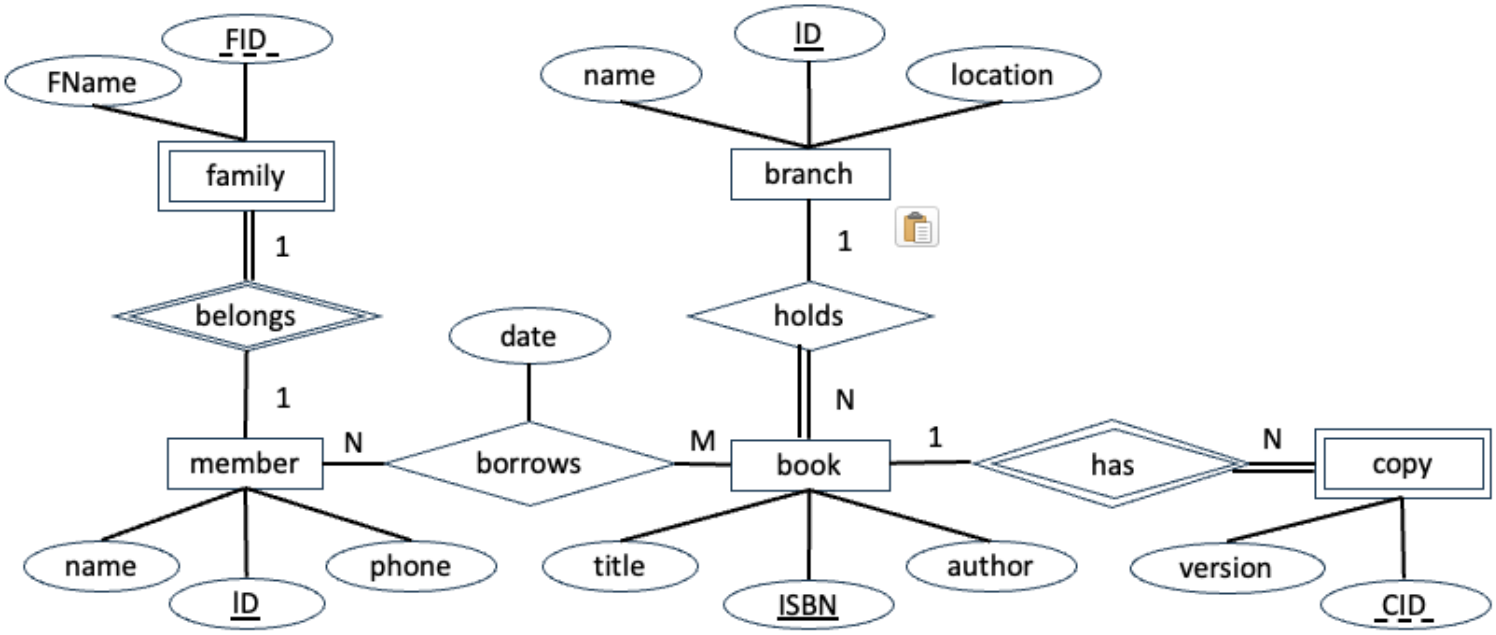
- *borrows*(*ID-->member*, *ISBN-->book*, *date*)
- *book*(*ISBN*, *title*, *author*, *branchID-->branch*)



Exercise 2--Solution

AALBORG UNIVERSITY

- *branch*(*ID*, *name*, *location*)
- *book*(*ISBN*, *title*, *author*, *branchID-->branch*)
- *member*(*ID*, *name*, *phone*, *FID*, *FName*)
- *copy*(*ISBN-->book*, *CID*, *version*)
- *borrows*(*ID-->member*, *ISBN-->book*, *date*)

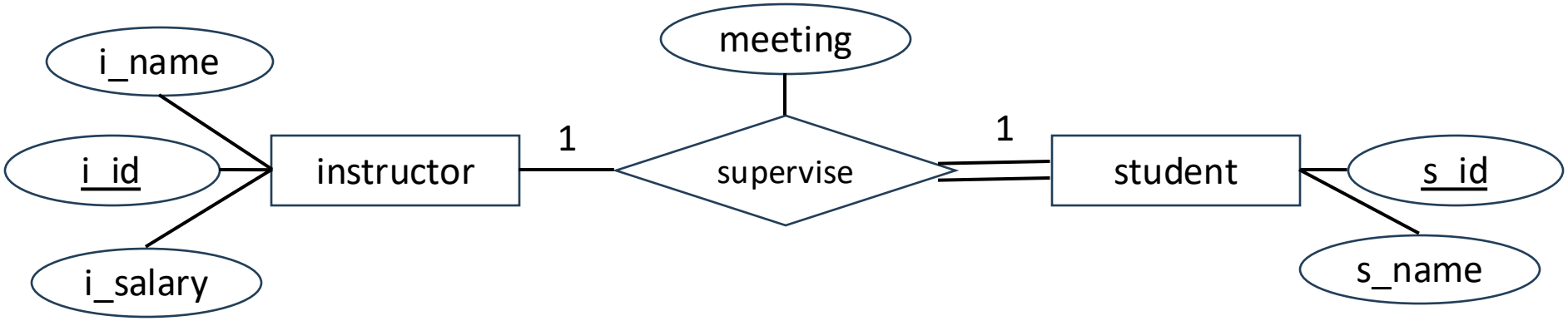


M5: Binary 1:1 relationship types

AALBORG UNIVERSITY

- For each regular binary 1:1 relationship type, **extend** a relation.
  - It is best to extend a table of an entity type with total participation to mitigate redundancy; otherwise, many rows will have NULL values
- The relation you extend will have a foreign key that references the primary key of the other entity type
- Add also columns for each of the simple attributes of the relationship type

M5: Example



instructor: partial participation

student: total participation

- ***student(s_id, s_name, i_id-->instructor, meeting)***
- *Instructor(i_id, i_name, i_salary)*

It is best to **extend** the relation of the entity type with total participation

M5: Example (Cont.)

AALBORG UNIVERSITY

i_id	*i_name*	*i_salary*
10101	Srinivasan	65000
12121	Wu	90000
15151	Mozart	40000
22222	Einstein	95000
32343	EI Said	60000
33456	Gold	87000
45565	Katz	75000
58583	Califieri	62000
76543	Singh	80000
76766	Crick	72000
83821	Brandt	92000
98345	Kim	80000

instructor

s_id	*s_name*
1001	Alice
1002	Anna
1003	Bob
1004	Cathy
1005	Christian
1006	David

student

+

s_id	*i_id*	*meeting*
1001	12121	10
1002	33456	9
1003	22222	12
1004	83821	8
1005	76766	10
1006	45565	11

supervise

merged

s_id	*s_name*	*i_id*	*meeting*
1001	Alice	12121	10
1002	Anna	33456	9
1003	Bob	22222	12
1004	Cathy	83821	8
1005	Christian	76766	10
1006	David	45565	11

student

Extend *student*

M5: Example (Cont.)

AALBORG UNIVERSITY

i_id	*i_name*	*i_salary*	*s_id*	*meeting*
10101	Srinivasan	65000	null	null
12121	Wu	90000	1001	10
15151	Mozart	40000	null	null
22222	Einstein	95000	1003	12
32343	EI Said	60000	null	null
33456	Gold	87000	1002	9
45565	Katz	75000	1006	11
58583	Califieri	62000	null	null
76543	Singh	80000	null	null
76766	Crick	72000	1005	10
83821	Brandt	92000	1004	8
98345	Kim	80000	null	null

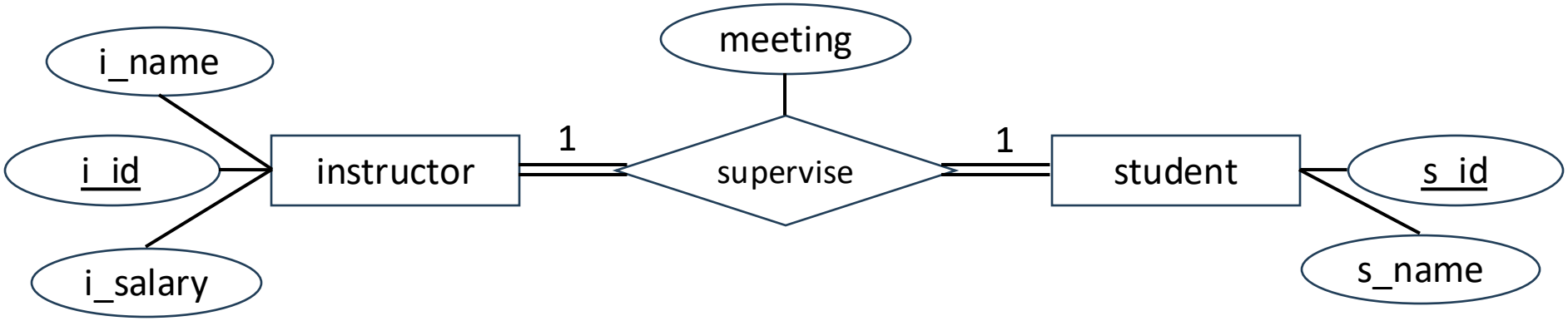
instructor

s_id	*s_name*
1001	Alice
1002	Anna
1003	Bob
1004	Cathy
1005	Christian
1006	David

student

Why not *instructor* ?

M5: Single Relation-Example



instructor: total participation

student: total participation

- ***r(s_id, s_name, i_id, i_name, i_salary, meeting)***

Only correct in case of total participation of both involved entity types

M5: Single Relation-Example (Cont.)

AALBORG UNIVERSITY

i_id	*i_name*	*i_salary*
12121	Wu	90000
22222	Einstein	95000
33456	Gold	87000
45565	Katz	75000
76766	Crick	72000
83821	Brandt	92000

instructor

s_id	*s_name*
1001	Alice
1002	Anna
1003	Bob
1004	Cathy
1005	Christian
1006	David

student

s_id	*i_id*	*meeting*
1001	12121	10
1002	33456	9
1003	22222	12
1004	83821	8
1005	76766	10
1006	45565	11

supervise

M5: Single Relation-Example (Cont.)

AALBORG UNIVERSITY

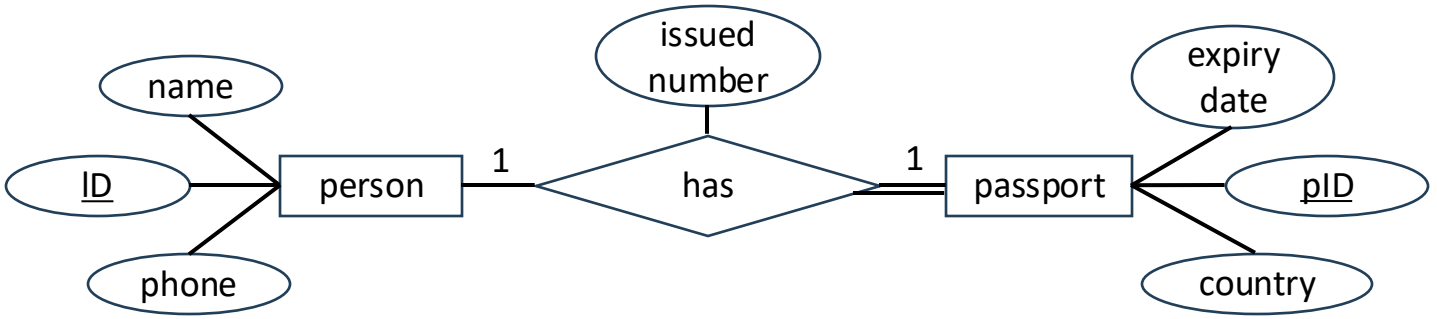
- *r*(*s_id*, *s_name*, *i_id*, *i_name*, *i_salary*, *meeting*)

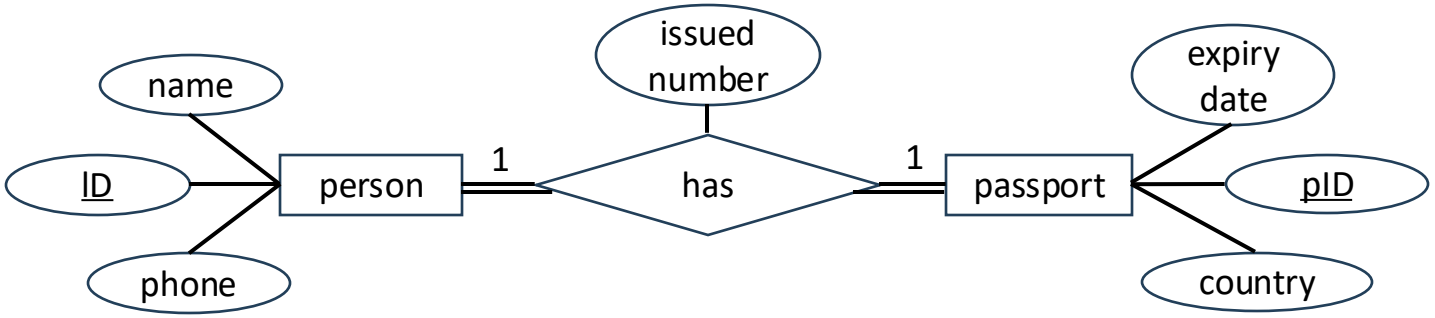
s_id	*s_name*	*i_id*	*i_name*	*i_salary*	*meeting*
1001	Alice	12121	Wu	90000	10
1002	Anna	33456	Gold	87000	9
1003	Bob	22222	Einstein	95000	12
1004	Cathy	83821	Brandt	92000	8
1005	Christian	76766	Crick	72000	10
1006	David	45565	Katz	75000	11

Exercise 3

AALBORG UNIVERSITY

- Create relations for the ER diagrams.

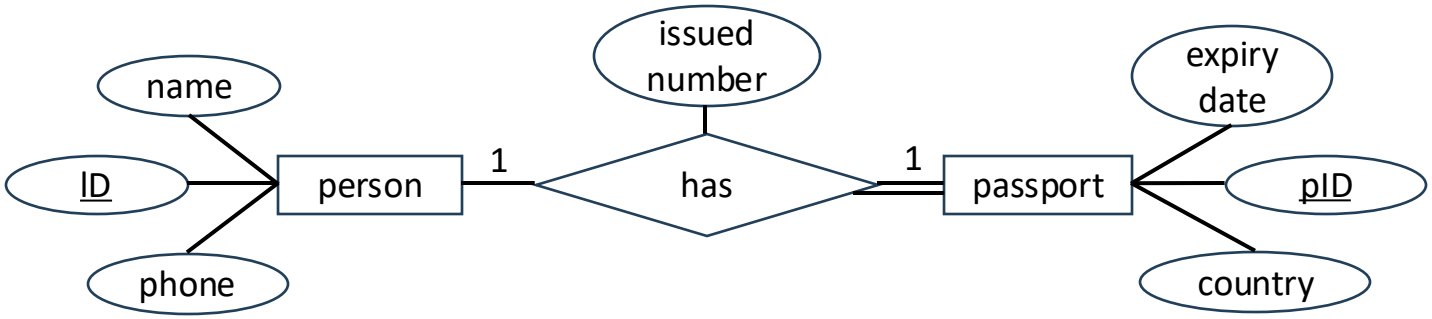




Exercise 3--Solution

AALBORG UNIVERSITY

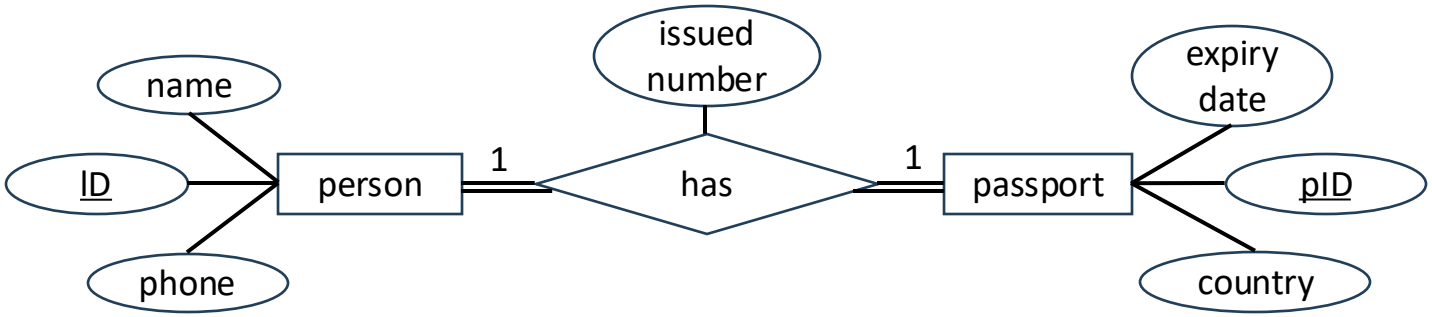
- *person*(*ID*, *name*, *phone*)
- *passport*(*pID*, *country*, *expiry date*, *ID-->person*, *issued number*)



Exercise 3--Solution

AALBORG UNIVERSITY

- Option 1
  - *person*(*ID*, *name*, *phone*)
  - *passport*(*pID*, *country*, *expiry date*, *ID-->person*, *issued number*)
- Option 2
  - *person*(*ID*, *name*, *phone*, *pID-->passport*, *issued number*)
  - *passport*(*pID*, *country*, *expiry date*)
- Option 3
  - *r*(*ID*, *name*, *phone*, *pID*, *country*, *expiry date*, *issued number*)



Summary of Mapping Relationships

AALBORG UNIVERSITY

- M:N relationship type
  - New relation with relationship type's attributes
    - Add attributes referencing the primary keys of the involved entity type relations
    - Primary key: set of foreign keys
- 1:N relationship type
  - Add information to the entity type relation of the "N"-side:
    - Add foreign key referencing the primary key of the "1"-side entity type relation
    - Add attributes of the relationship type
- 1:1 relationship type
  - Add information to one of the involved entity type relations:
    - Add foreign key referencing the primary key of the other entity type relation
    - Add attributes of the relationship type

Agenda

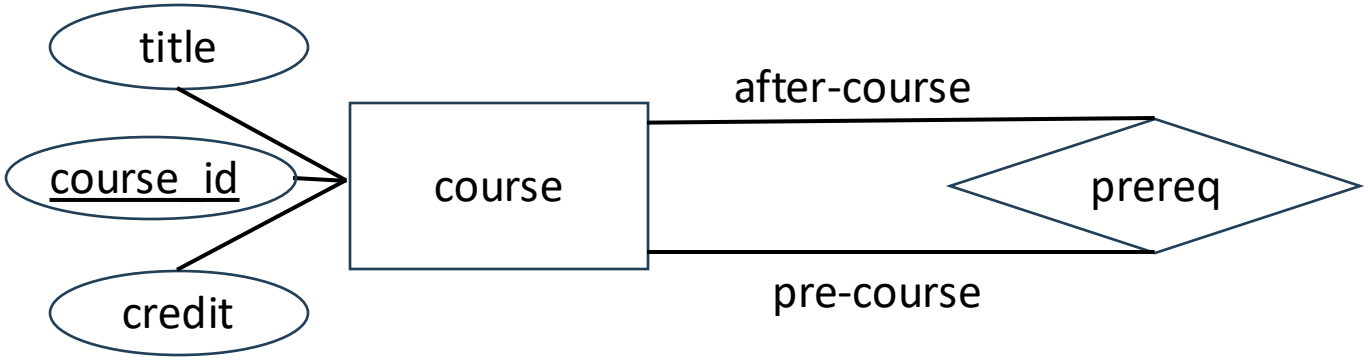
AALBORG UNIVERSITY

- Map entity types to relations
- Map relationship types to relations
- Map complex cases to relations
- Relational modelling of Specialization

M6: Complex Relation Types

AALBORG UNIVERSITY

- Recursive relationship types



- Mapping just like standard N:M relationship types and renaming of foreign keys
- ***course(course_id, title, credit)***
- ***prereq(after-course_id-->course, pre-course_id-->course)***

M6: Complex Relation Types

AALBORG UNIVERSITY

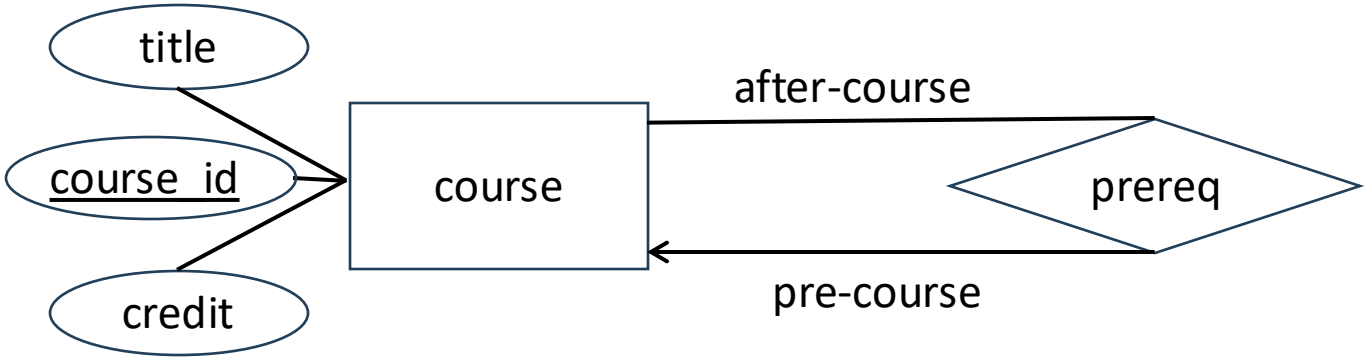
- Recursive relationship types

Module title (DK/ENG) and ECTS credits	**Database Systems /Databasesystemer (DBS), 5 ECTS**
Module description	Link: https://moduler.aau.dk/course/2025-2026/DSNSWCB514?lang=da-DK
Module coordinator	Tiantian Liu liutt@cs.aau.dk
Type and language	Course and in English
Objectives	The aim of the course is to teach concepts of database systems and data modeling
Academic content and conjunction with other modules/semesters	The academic content of the course includes: relational model, relational algebra, designing a database (ER model, conceptual design), normal forms (logical design), SQL (DDL, DML, TCL, DQL), physical design, query execution plans, transactions, concurrency control, and recovery
Scope and expected performance	Since it is a 5 ECTS course, the work load is expected to be 150 hours for the student The course will consist of lectures and exercises
Participants	CPH-SW5
Preprequisites for participation	Algorithms and data structures
Module activities (course sessions etc.)	Lectures, exercise sessions
Examination	Written or oral examination - Please note: FINAL examination form and aids (if any) will be posted on the course page)

M6: Complex Relation Types

AALBORG UNIVERSITY

- Recursive relationship types

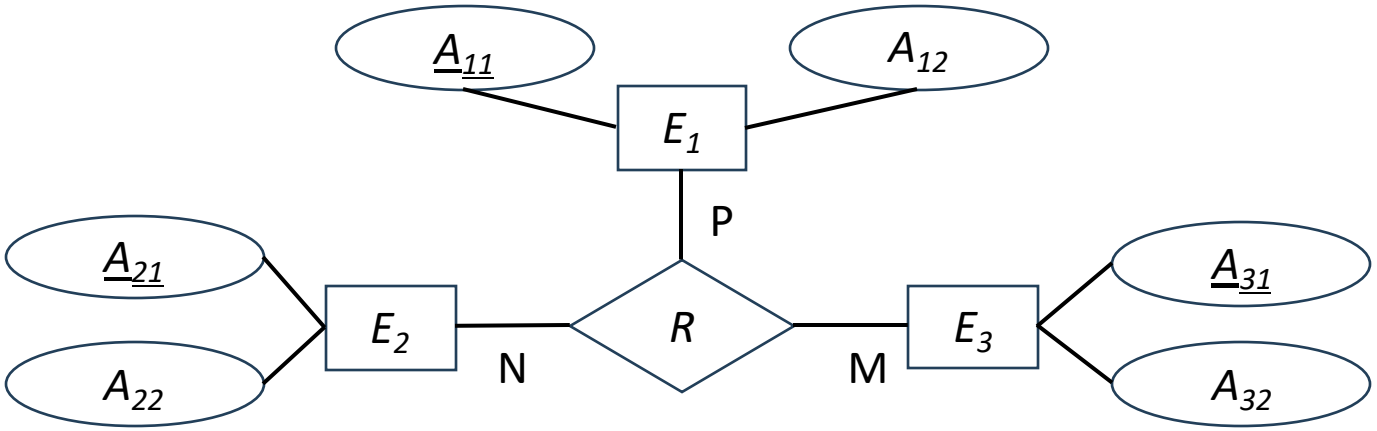


- Mapping just like standard 1:N relationship types and renaming of foreign keys
- ***course(course_id, title, credit, pre-course_id--> course)***

M6: Complex Relation Types

AALBORG UNIVERSITY

- N:M:P relationship types

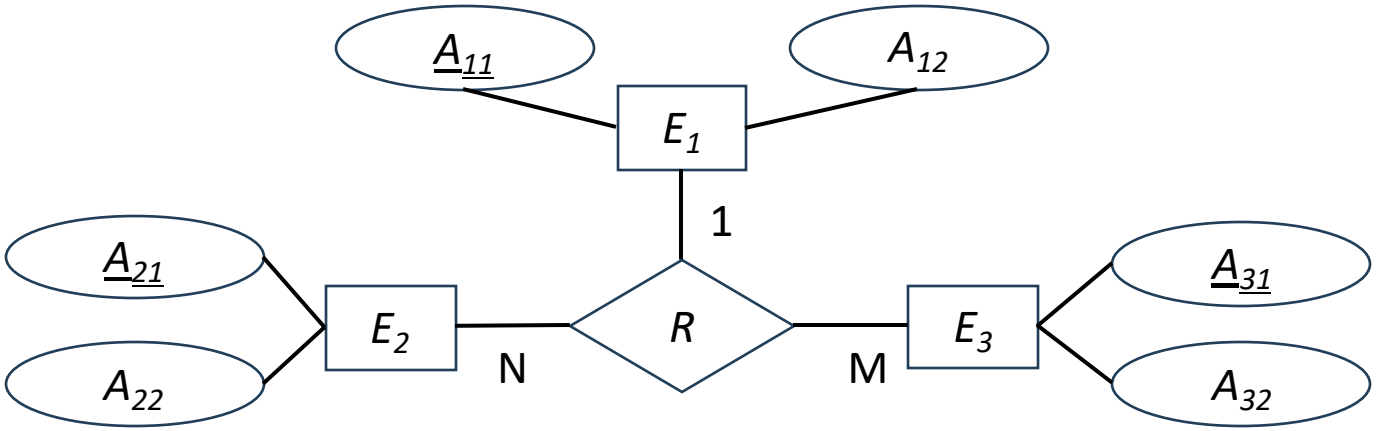


- $E_1(\underline{A_{11}}, A_{12})$
- $E_2(\underline{A_{21}}, A_{22})$
- $E_3(\underline{A_{31}}, A_{32})$
- $R(\underline{A_{11}}\text{-->}E_1, \underline{A_{21}}\text{-->}E_2, \underline{A_{31}}\text{-->}E_3)$

M6: Complex Relation Types

AALBORG UNIVERSITY

- N:M:1 relationship types

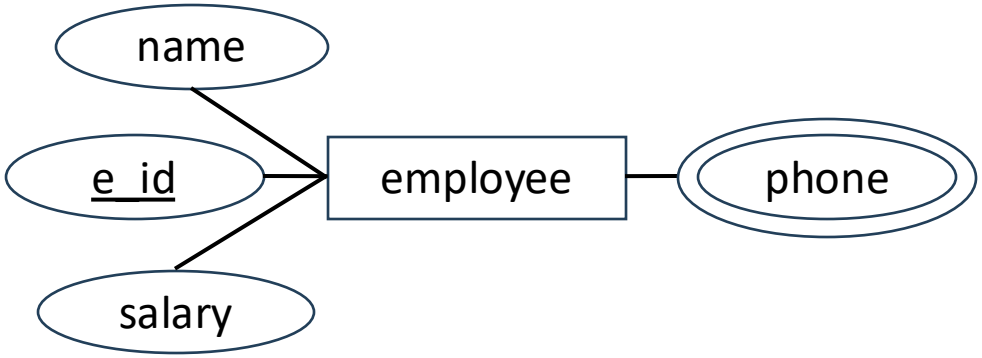


- $E_1(\underline{A_{11}}, A_{12})$
- $E_2(\underline{A_{21}}, A_{22})$
- $E_3(\underline{A_{31}}, A_{32})$
- $R(\underline{A_{21}}\text{-->}E_2, \underline{A_{31}}\text{-->}E_3, A_{11}\text{-->}E_1)$

M7: Complex Attributes

AALBORG UNIVERSITY

- Multi-valued attributes

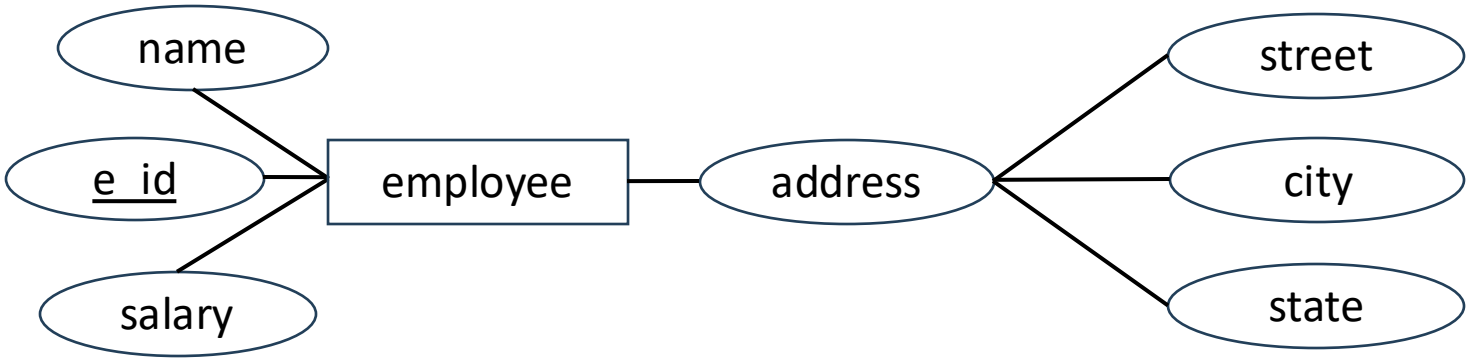


- Create a separate relation for each multi-valued attribute
- ***employee(e_id, name, salary)***
- ***phone(e_id-->employee, phoneNumber)***

M7: Complex Attributes

AALBORG UNIVERSITY

- Composite attributes

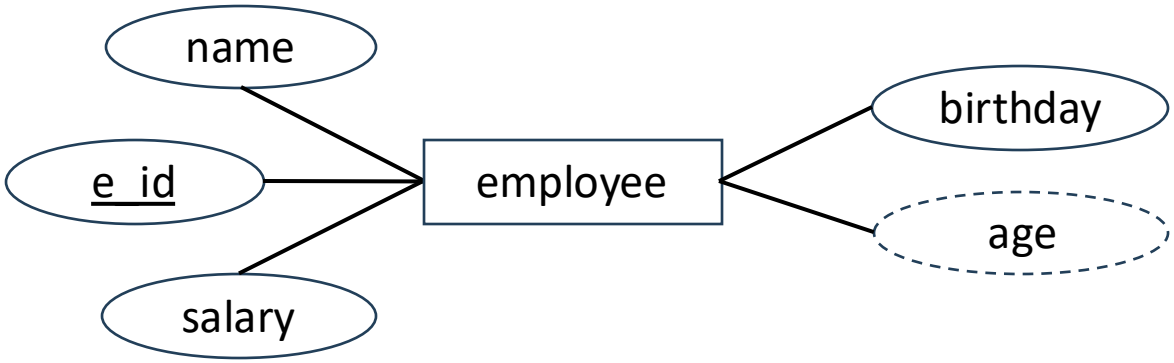


- Include the component attributes in the relation
- ***employee(e_id, name, salary, state, city, street)***

M7: Complex Attributes

AALBORG UNIVERSITY

- Derived attributes



- Ignored during mapping to relations, can be added later by using views.

Overview of the Mapping

AALBORG UNIVERSITY

- **M1**: Regular entity types
  - Create a relation
- **M2**: Weak entity types
  - Create a relation
  - Primary key: the strong entity's primary key and part of its own attributes
- **M3**: Binary M:N relationship types
  - Create a relation
  - Primary key: the union of the primary keys of the participating entity types
- **M4**: Binary 1:N relationship types
  - Extend the N-side relation with foreign key

Overview of the Mapping (Cont.)

AALBORG UNIVERSITY

- **M5**: Binary 1:1 relationship types
  - Both two sides totally participating: extend one side relation with foreign key, or use one relation
  - Both two sides partially participating: extend one side relation with foreign key
  - One side totally participating and another side partially participating: extend totally participating side
- **M6**: Complex relationship types
  - Recursive relationship types: create a relation
  - N:M:P / N:M:1 relation types: create a relation

Overview of the Mapping (Cont.)

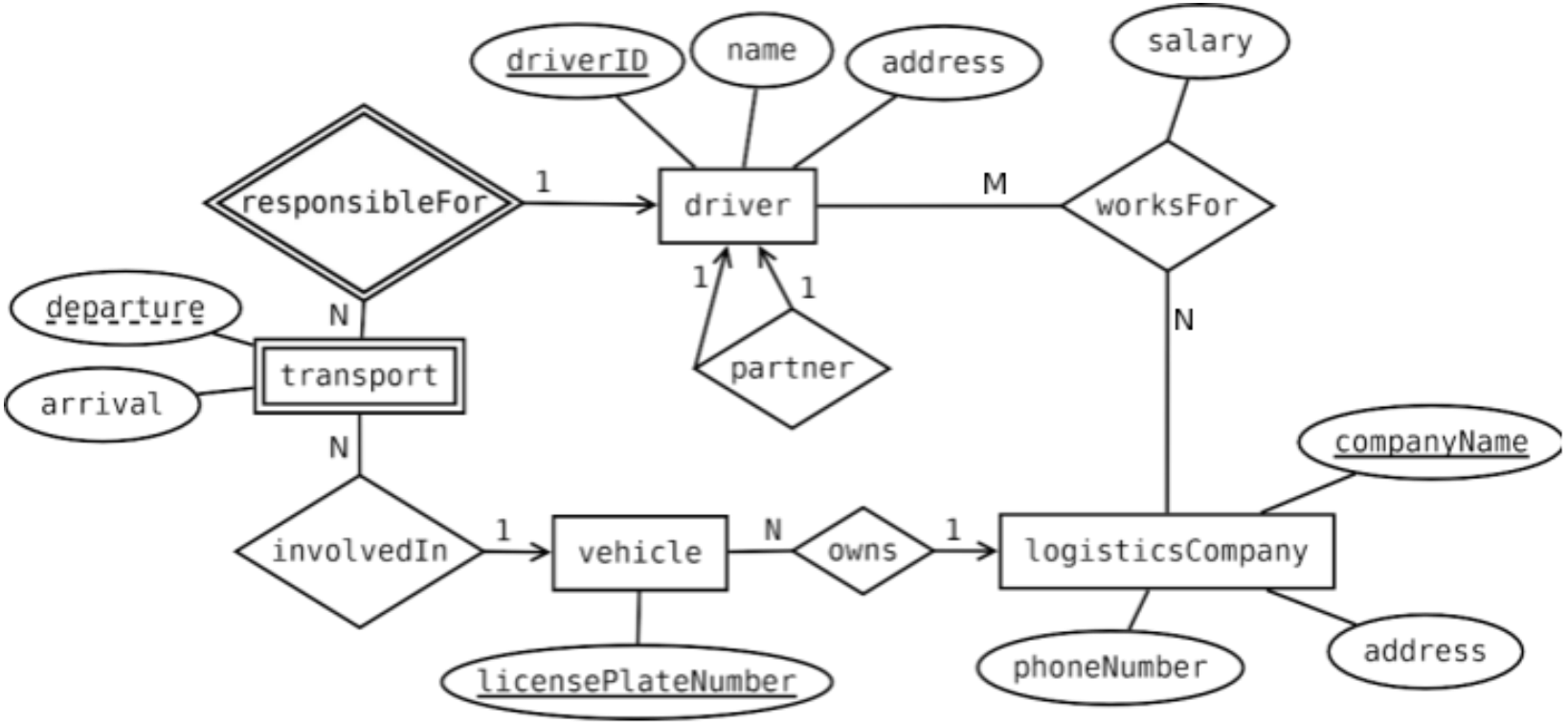
AALBORG UNIVERSITY

- **M7**: Complex attributes
  - Multi-valued attributes: create a separate relation for each multi-valued attribute
  - Composite attributes: Include the component attributes in the relation
  - Derived attributes: Ignored during mapping to relations, can be added later by using views.

Exercise 4

AALBORG UNIVERSITY

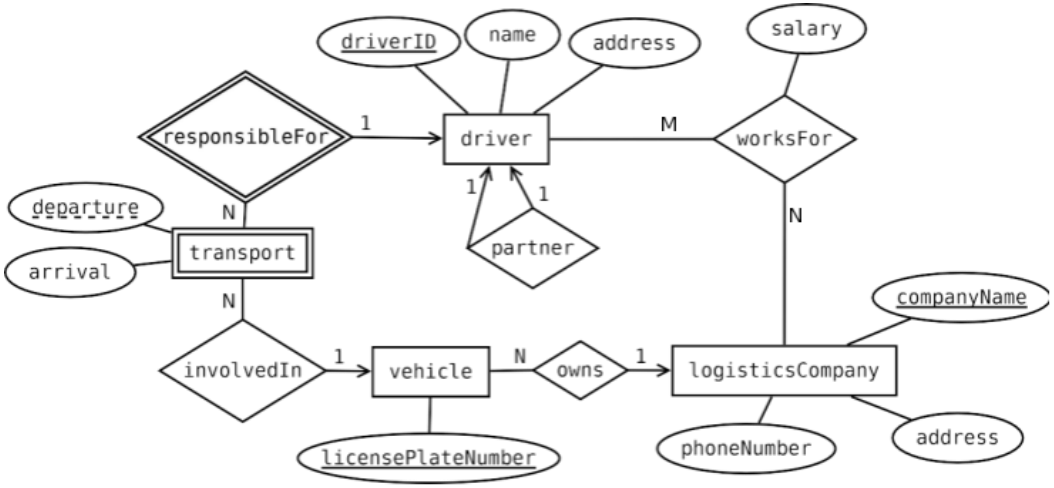
- Create relations for the ER diagrams.



Exercise 4--Solution

AALBORG UNIVERSITY

- *driver* (*driverID, name, address, partner-->driver*)
- *logisticsCompany* (*name, address, phoneNumber*)
- *worksFor* (*driverID-->driver, companyName-->logicsticsCompany, salary*)
- *vehicle* (*licensePlateNumber, companyName-->logisticsCompany*)
- *transport* (*driverID-->driver, departure, licensePlateNumber-->vehicle, arrival*)

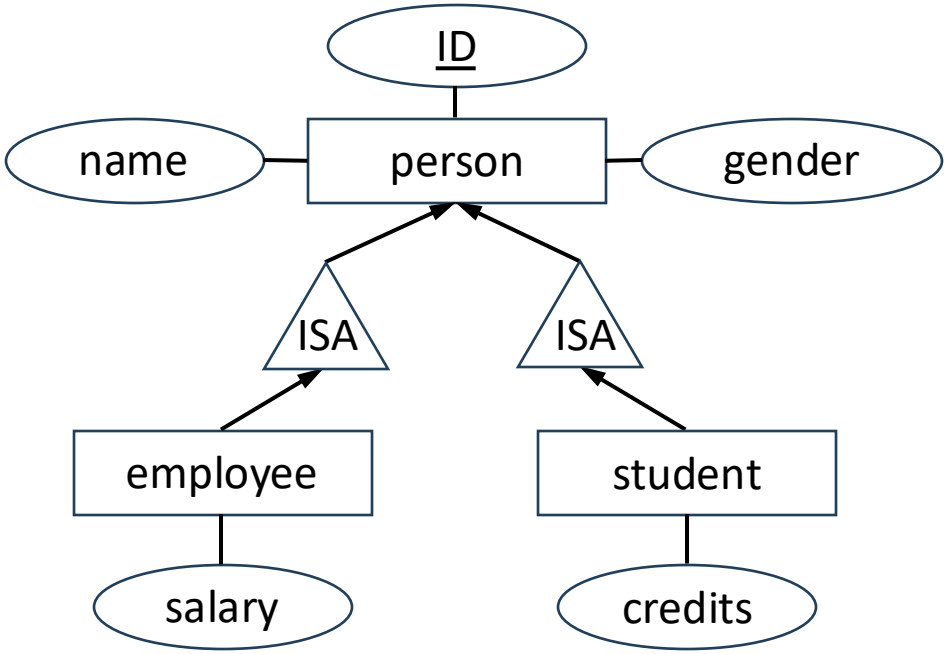


Agenda

AALBORG UNIVERSITY

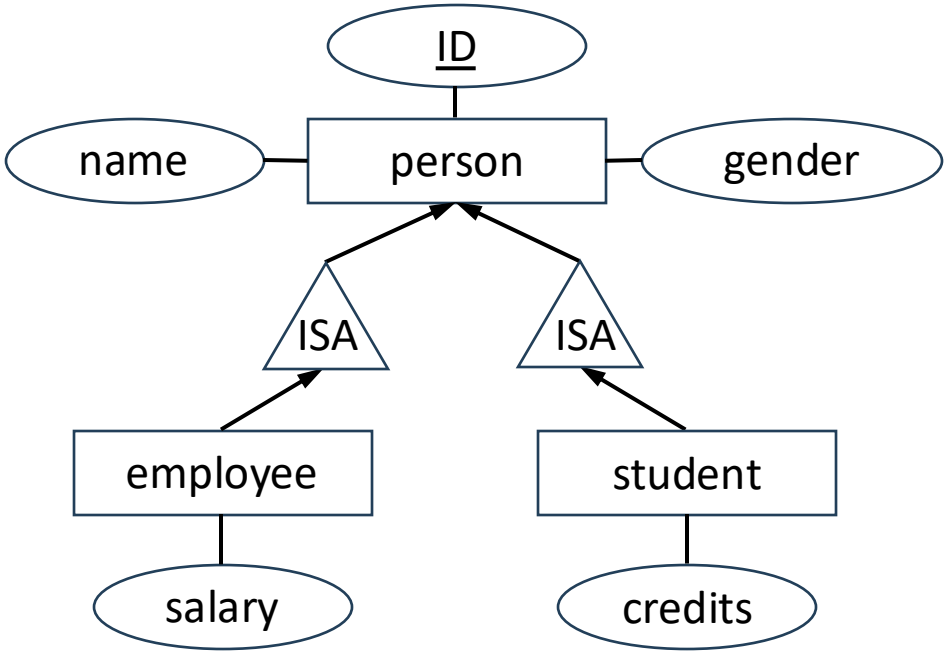
- Map entity types to relations
- Map relationship types to relations
- Map complex cases to relations
- Relational modelling of Specialization

Relational Modelling of Specialization



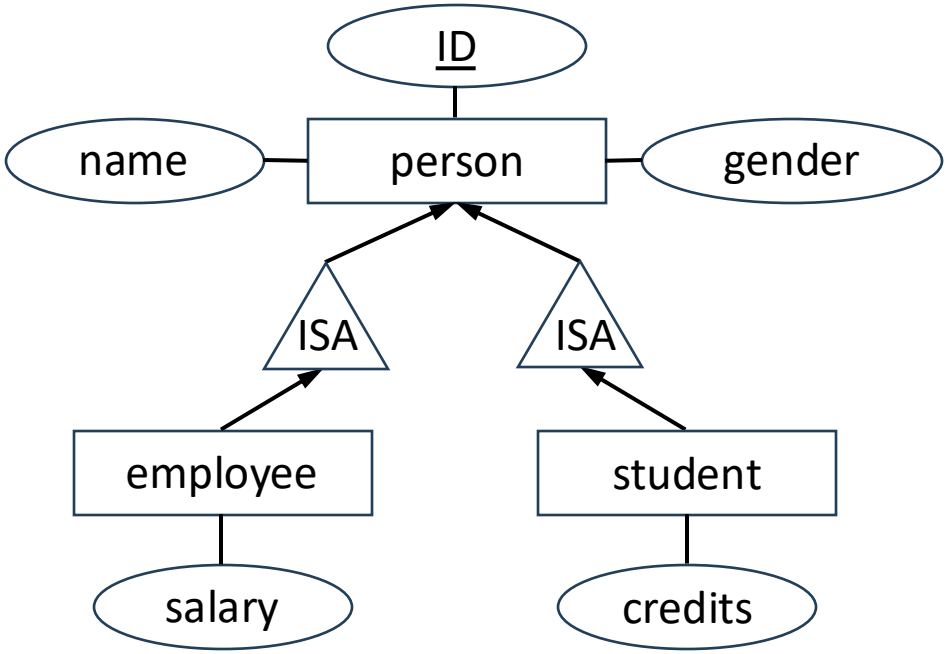
- The relational model does not support specialization and cannot express inheritance.
- Specialization is simulated.

Relational Modelling of Generalization



- How to map this information to relations?

Alternative 1: main classes



- A particular entity is mapped to a single tuple in a single relation (to its main class).
- ***person(ID, name, gender)***
- ***employee(ID, name, gender, salary)***
- ***student(ID, name, gender, credits)***

Alternative 1: main classes

AALBORG UNIVERSITY

ID	*name*	*gender*
12930	Sophia	F
18231	Olivia	F
72201	Ethan	M

person

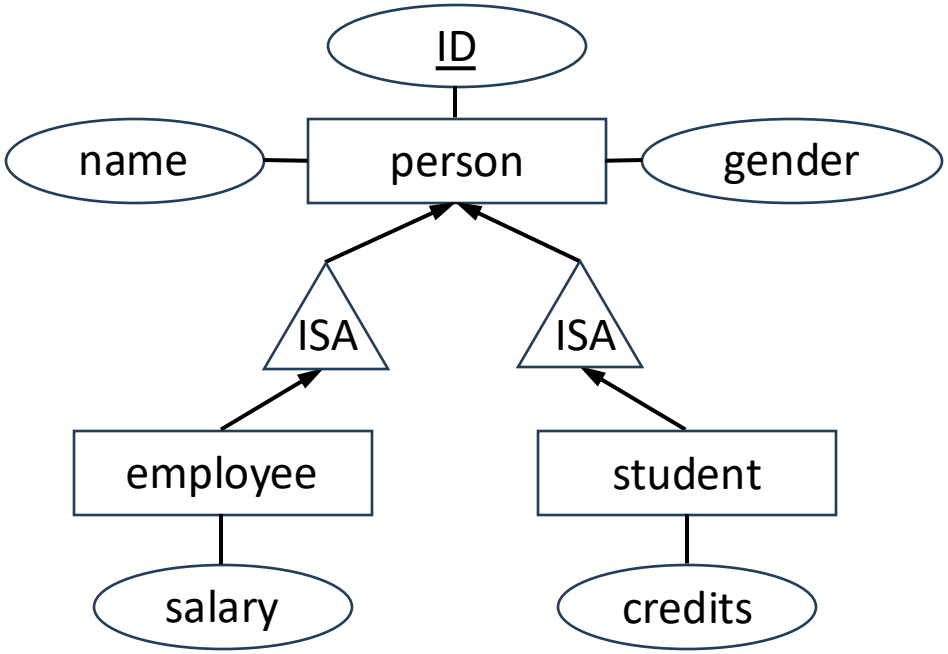
ID	*name*	*gender*	*salary*
12121	Wu	M	90000
76766	Crick	M	72000
83821	Brandt	F	92000

employee

ID	*name*	*gender*	*credits*
10001	Alice	F	100
10002	Anna	F	110
10006	David	M	125

student

Alternative 2: partitioning



- Parts of a particular entity are mapped to multiple relations, the key is duplicated.
- ***person(ID, name, gender)***
- ***employee(ID-->person, salary)***
- ***student(ID-->person, credits)***

Alternative 2: partitioning

AALBORG UNIVERSITY

ID	*name*	*gender*
10001	Alice	F
10002	Anna	F
10006	David	M
12121	Wu	M
12930	Sophia	F
18231	Olivia	F
72201	Ethan	M
76766	Crick	M
83821	Brandt	F

person

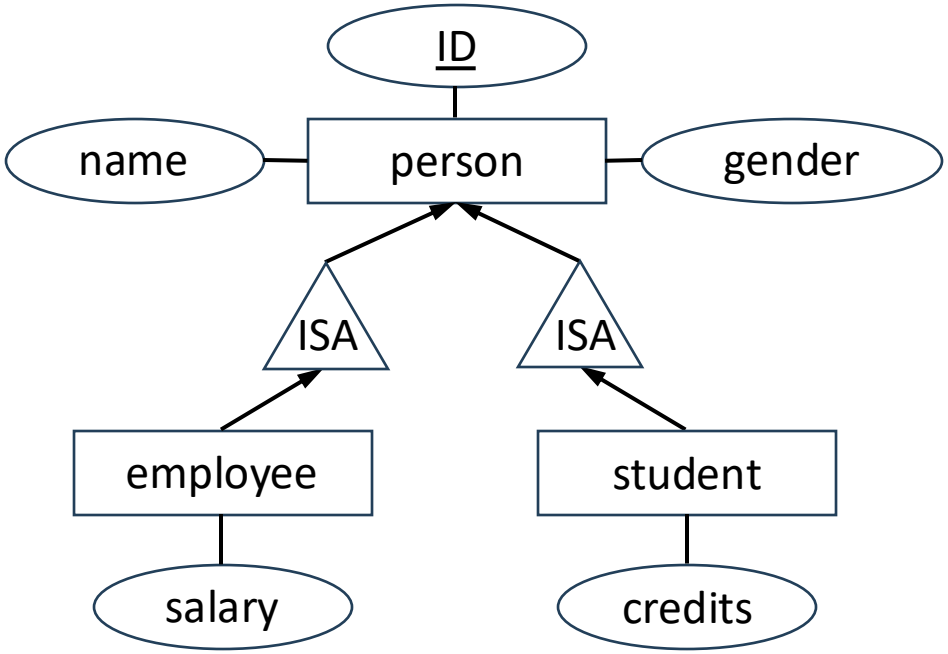
ID	*salary*
12121	90000
76766	72000
83821	92000

employee

ID	*credits*
10001	100
10002	110
10006	125

student

Alternative 3: full redundancy



- A particular entity is stored redundantly in the relations with all its inherited attributes.
- ***person(ID, name, gender)***
- ***employee(ID, name, gender, salary)***
- ***student(ID, name, gender, credits)***

Alternative 3: full redundancy

AALBORG UNIVERSITY

ID	*name*	*gender*
10001	Alice	F
10002	Anna	F
10006	David	M
12121	Wu	M
12930	Sophia	F
18231	Olivia	F
72201	Ethan	M
76766	Crick	M
83821	Brandt	F

person

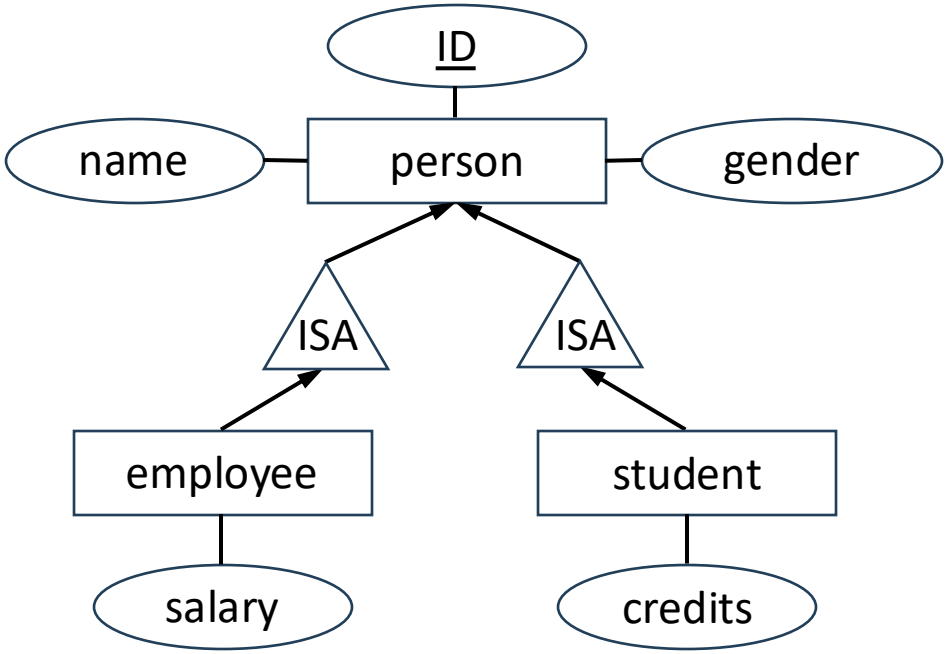
ID	*name*	*gender*	*salary*
12121	Wu	M	90000
76766	Crick	M	72000
83821	Brandt	F	92000

employee

ID	*name*	*gender*	*credits*
10001	Alice	F	100
10002	Anna	F	110
10006	David	M	125

student

Alternative 4: single relation



- All entities are stored in a single relation. An additional attribute encodes the membership in a particular entity type.
- ***person(ID, name, gender, type, salary, credits)***

Alternative 4: single relation

AALBORG UNIVERSITY

ID	*name*	*gender*	*type*	*salary*	*credits*
10001	Alice	F	student	null	100
10002	Anna	F	student	null	110
10006	David	M	student	null	125
12121	Wu	M	employee	90000	null
12930	Sophia	F	person	null	null
18231	Olivia	F	person	null	null
72201	Ethan	M	person	null	null
76766	Crick	M	employee	72000	null
83821	Brandt	F	employee	92000	null

person

Summary

AALBORG UNIVERSITY

- Review the ER model
- Map entity types to relations
- Map relationship types to relations
- Map complex cases to relations
- Relational modelling of Specialization

Next Lecture

AALBORG UNIVERSITY

- Relational Database Design
  - Understand the concepts of functional dependencies and normal forms
  - Describe the quality of a design by using normal forms
  - Improve a database design by decomposition of relations
- Physical Design
  - Understand how tables are stored in files
  - Understand basic indexing techniques
  - Understand the effects of storage and indexes on the performance of basic SQL queries