Databases

Jörg Endrullis

VU University Amsterdam

2015

From Conceptual to Relational Model

Basic idea

From Conceptual to Relational Model

Basic idea

- entity sets and relationship sets are represented as tables

From Conceptual to Relational Model

Basic idea

- entity sets and relationship sets are represented as tables
- for each entity set and relationship set there is a table
  - name of the table = name of the entity or relationship set

From Conceptual to Relational Model

Basic idea

- entity sets and relationship sets are represented as tables
- for each entity set and relationship set there is a table
  - name of the table = name of the entity or relationship set
- each table has a number of columns (with unique names)
  - usually the columns correspond to the attributes

Representing Entity Sets

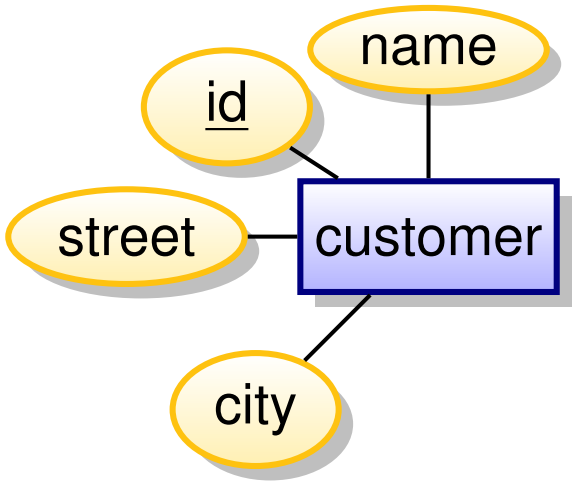
A **strong entity set** becomes a table with

- columns for the attributes

Representing Entity Sets

A **strong entity set** becomes a table with

- columns for the attributes



customer

id	name	street	city
1	Smith	North	Pittsburgh
2	Jones	Alma	Philadelphia
3	Brown	Main	New York
4	Ford	Main	Washington

Representing Weak Entity Sets

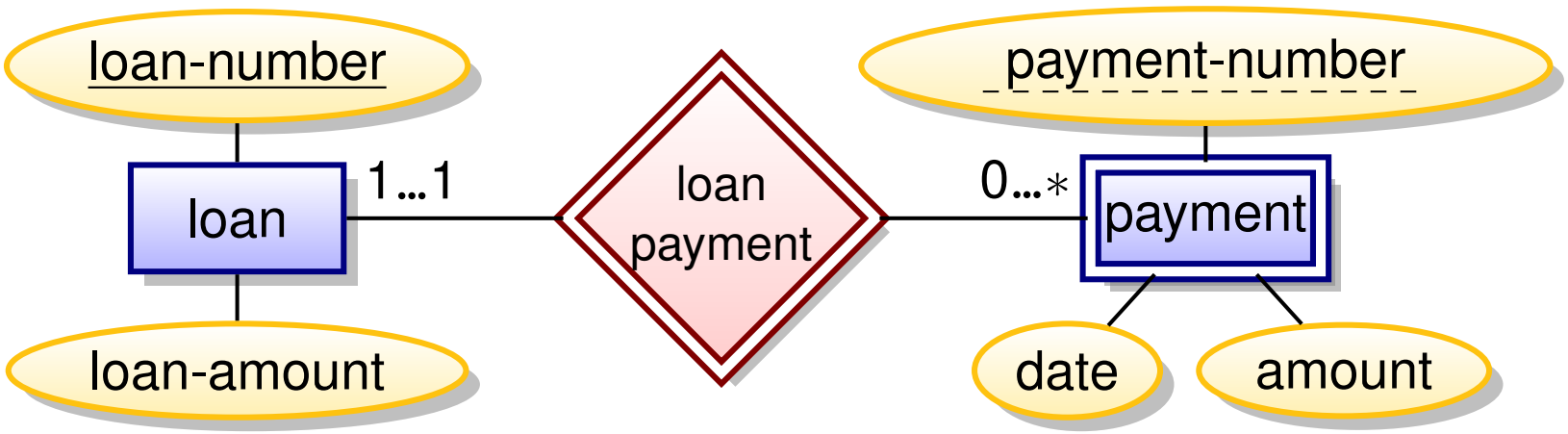
A **weak entity set** becomes a table that includes

- columns for the attributes, and
- columns for the primary keys of the identifying entity

Representing Weak Entity Sets

A **weak entity set** becomes a table that includes

- columns for the attributes, and
- columns for the primary keys of the identifying entity



payment

loan-number→loan	payment-number	date	amount
L-11	1	19-05-2013	125
L-14	2	01-02-2014	1000
L-17	1	05-07-2012	50
L-20	5	17-11-2013	750

Representing Relationship Sets

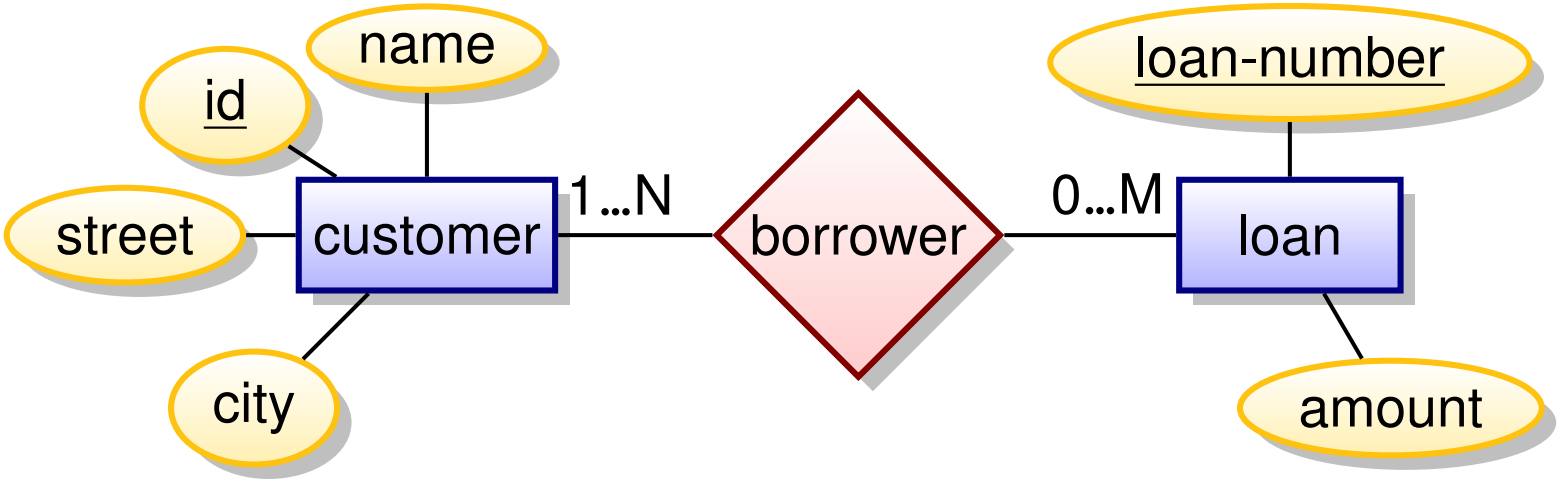
A many-to-many **relationship set** becomes a table with

- columns for the attributes of the relationship, and
- for the primary keys of the participating entity sets.

Representing Relationship Sets

A many-to-many **relationship set** becomes a table with

- columns for the attributes of the relationship, and
- for the primary keys of the participating entity sets.



borrower

id	loan-number
12-0202	L-11
01-1823	L-14
22-7361	L-17
05-1912	L-20

Eliminating Tables

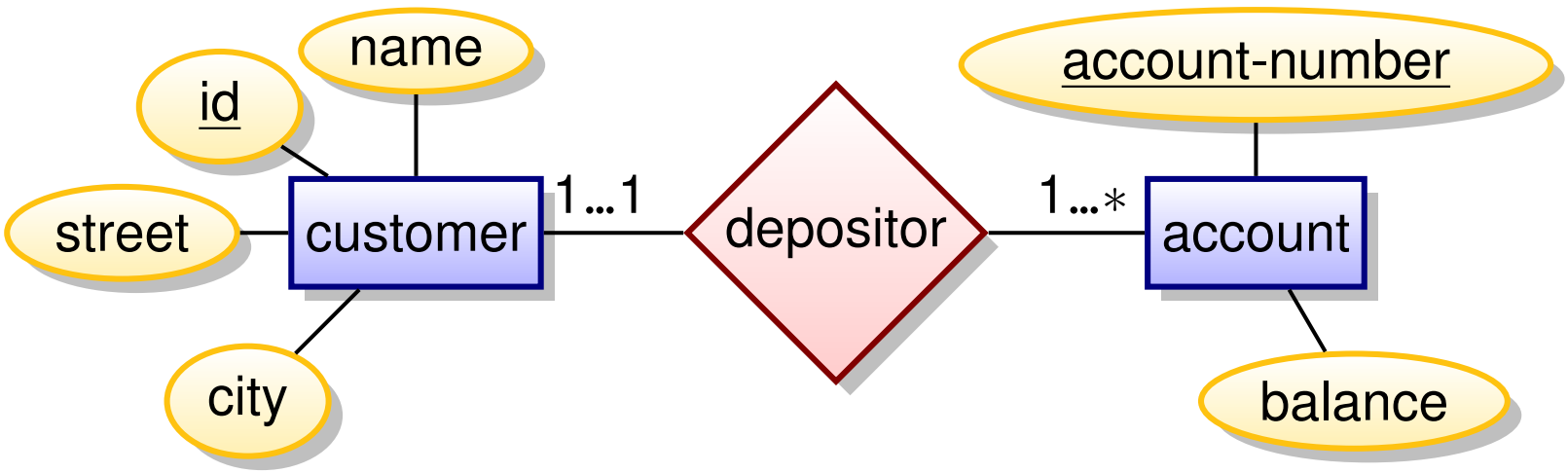
Many-to-one & **one-to-many** relations can be represented by:

- adding an extra extra attribute/column to the many-side with the primary key of the one-side

Eliminating Tables

Many-to-one & **one-to-many** relations can be represented by:

- adding an extra extra attribute/column to the many-side with the primary key of the one-side



For example, instead of creating a table for the relationship set *depositor*, add a the attribute *id* of *customer* to *account*.

account

id→customer	account-number	balance
12-0202	83828	125
01-1823	29281	1000

Eliminating Tables

- For **one-to-one** (0…1 or 1…1) relationship sets either side can be extended with the key of the other.

Eliminating Tables

- For **one-to-one** (0…1 or 1…1) relationship sets either side can be extended with the key of the other.
- If participation is **partial** (0…1) then replacing the table by an attribute will result in **null values** for the entities that do not participate in the relationship set.

Eliminating Tables

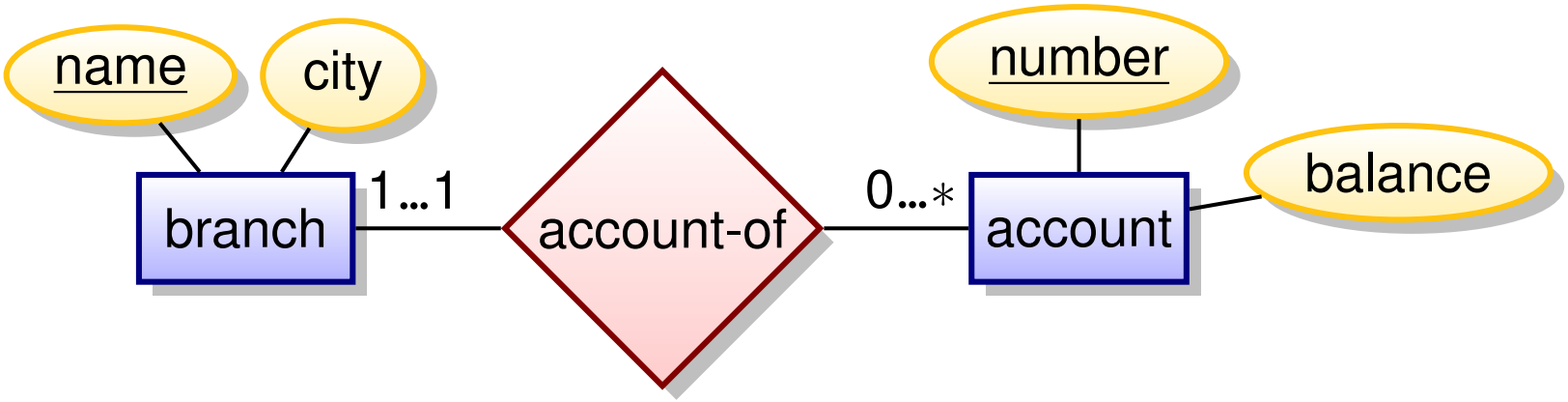
- For **one-to-one** (0…1 or 1…1) relationship sets either side can be extended with the key of the other.
- If participation is **partial** (0…1) then replacing the table by an attribute will result in **null values** for the entities that do not participate in the relationship set.
- Tables for relationship sets linking **weak entity sets** to the identifying entity set can always be eliminated.

Eliminating Tables

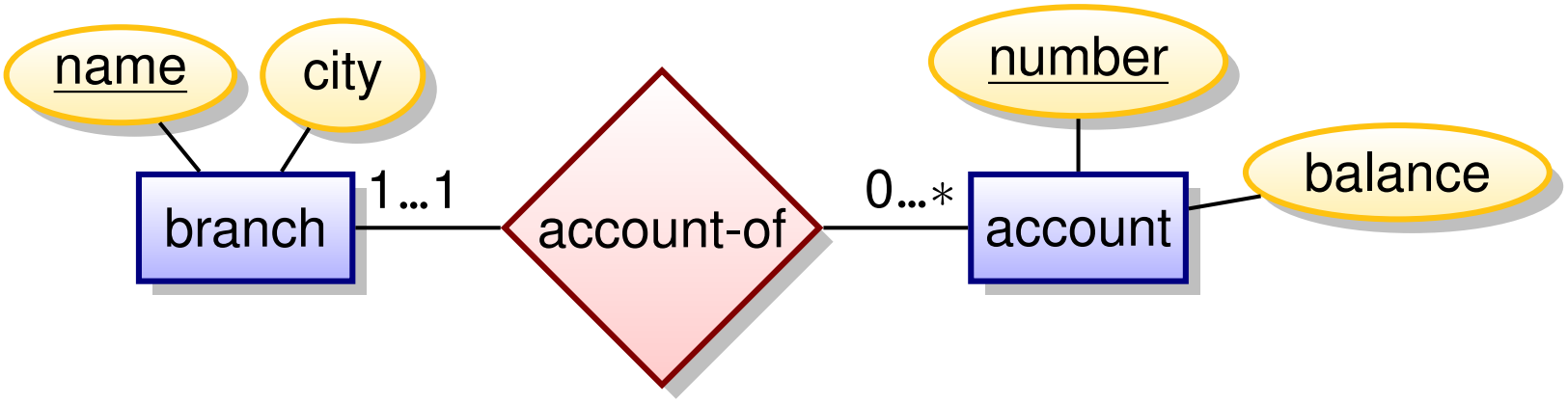
- For **one-to-one** (0…1 or 1…1) relationship sets either side can be extended with the key of the other.
- If participation is **partial** (0…1) then replacing the table by an attribute will result in **null values** for the entities that do not participate in the relationship set.
- Tables for relationship sets linking **weak entity sets** to the identifying entity set can always be eliminated.

  The table of the weak entity set already contains the key of the identifying entity set.
  - E.g. the *payment* table already contains the full information that would appear in the *loan-payment* table.
    (that is, *loan-number* and *payment-number*)

Eliminating Tables



Eliminating Tables

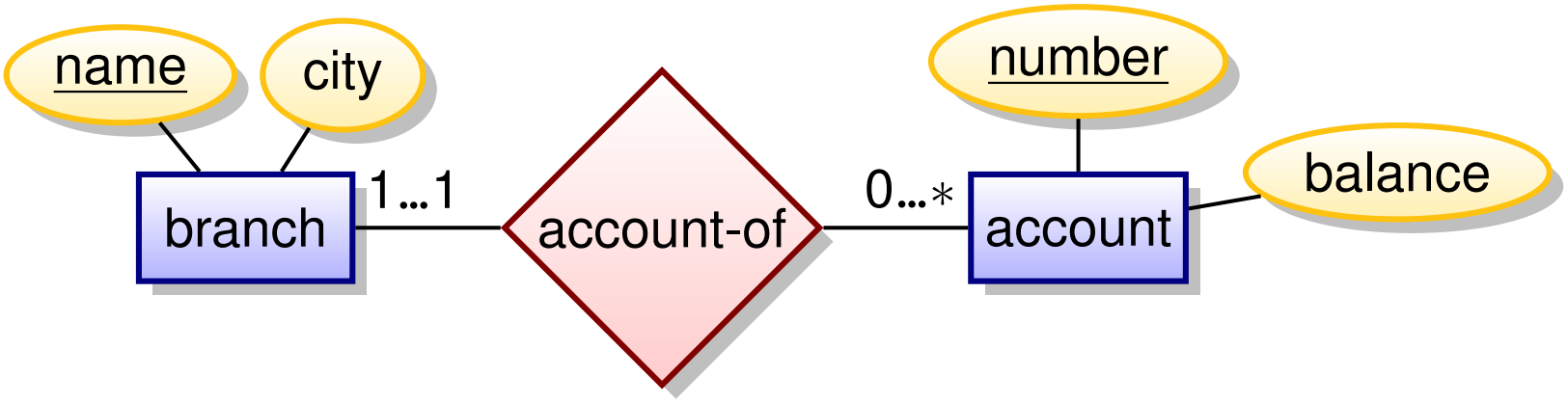


Basic translation

branch

name	city
branch1	Amsterdam
branch2	Utrecht

Eliminating Tables



Basic translation

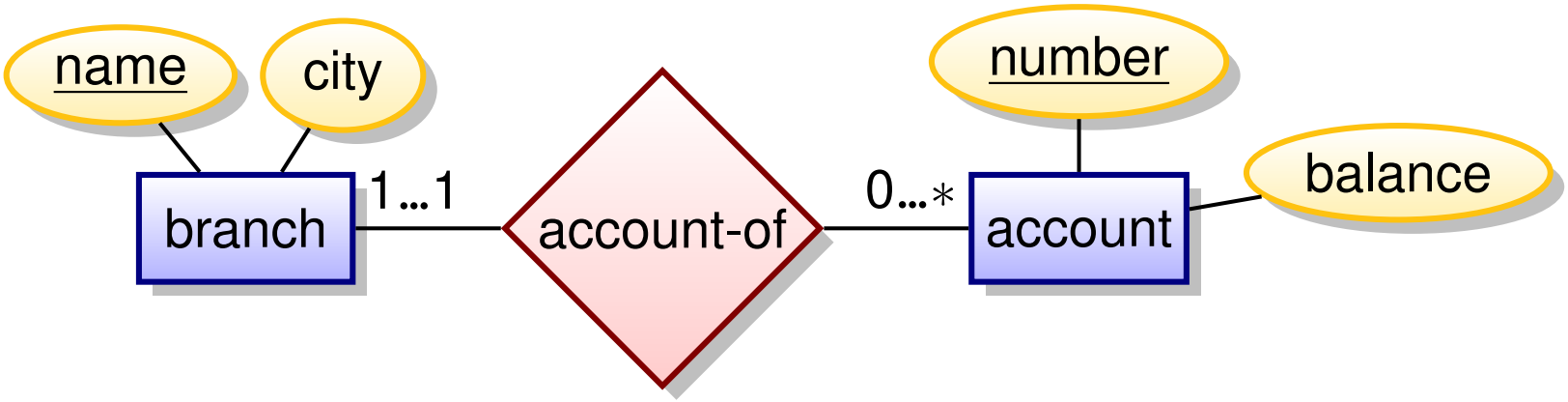
branch

name	**city**
branch1	Amsterdam
branch2	Utrecht

account-of

number →**account**	**name** →**branch**
83828	branch1
29281	branch2

Eliminating Tables



Basic translation

branch

name	**city**
branch1	Amsterdam
branch2	Utrecht

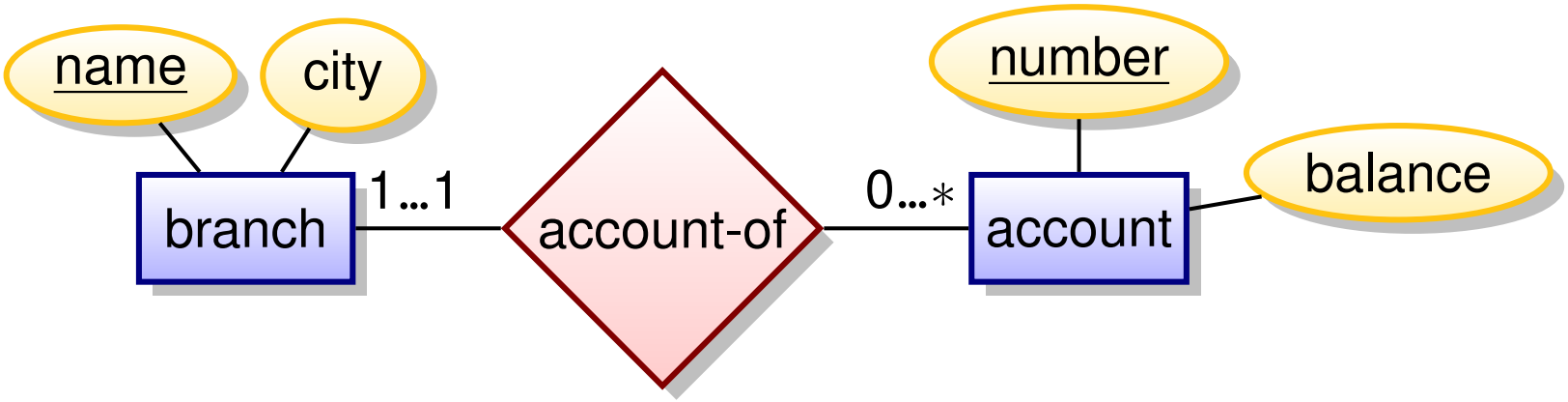
account-of

number →**account**	**name** →**branch**
83828	branch1
29281	branch2

account

number	**balance**
83828	125
29281	1000

Eliminating Tables



Basic translation

branch

name	**city**
branch1	Amsterdam
branch2	Utrecht

account-of

number →**account**	**name** →**branch**
83828	branch1
29281	branch2

account

number	**balance**
83828	125
29281	1000

Optimised translation

branch

name	**city**
branch1	Amsterdam
branch2	Utrecht

account

name→**branch**	**number**	**balance**
branch1	83828	125
branch2	29281	1000

Key Constraints

When translating entity sets and relationship sets to tables:

- every table should have a primary key (if possible)
- declared foreign key references for each relationship
- declared whether foreign keys are `NULLABLE` or not

Key Constraints

When translating entity sets and relationship sets to tables:

- every table should have a primary key (if possible)
- declared foreign key references for each relationship
- declared whether foreign keys are `NULLABLE` or not

Moreover, attributes should be declared unique (if there cannot be duplicates).

For example:

- All columns in tables from relationship sets are *not nullable*.

  Each row is a relationship among all participating entity sets.

Key Constraints

Which min/max cardinalities can be enforced and how?

- A 0…1 to 0…1 B:
- A 0…1 to 1…1 B:
- A 0…1 to 0…∗ B:
- A 0…1 to 1…∗ B:
- A 1…1 to 1…1 B:
- A 1…1 to 0…∗ B:
- A 1…1 to 1…∗ B:
- A 0…∗ to 0…∗ B:
- A 0…∗ to 1…∗ B:
- A 1…∗ to 1…∗ B:

Key Constraints

Which min/max cardinalities can be enforced and how?

- A 0…1 to 0…1 B: yes
- A 0…1 to 1…1 B:
- A 0…1 to 0…∗ B:
- A 0…1 to 1…∗ B:
- A 1…1 to 1…1 B:
- A 1…1 to 0…∗ B:
- A 1…1 to 1…∗ B:
- A 0…∗ to 0…∗ B:
- A 0…∗ to 1…∗ B:
- A 1…∗ to 1…∗ B:

Key Constraints

Which min/max cardinalities can be enforced and how?

- A 0…1 to 0…1 B: yes
  Use unique constraint on the foreign keys.
- A 0…1 to 1…1 B:
- A 0…1 to 0…∗ B:
- A 0…1 to 1…∗ B:
- A 1…1 to 1…1 B:
- A 1…1 to 0…∗ B:
- A 1…1 to 1…∗ B:
- A 0…∗ to 0…∗ B:
- A 0…∗ to 1…∗ B:
- A 1…∗ to 1…∗ B:

Key Constraints

Which min/max cardinalities can be enforced and how?

- A 0…1 to 0…1 B: yes
  Use unique constraint on the foreign keys.
- A 0…1 to 1…1 B: yes
- A 0…1 to 0…∗ B:
- A 0…1 to 1…∗ B:
- A 1…1 to 1…1 B:
- A 1…1 to 0…∗ B:
- A 1…1 to 1…∗ B:
- A 0…∗ to 0…∗ B:
- A 0…∗ to 1…∗ B:
- A 1…∗ to 1…∗ B:

Key Constraints

Which min/max cardinalities can be enforced and how?

- A 0…1 to 0…1 B: yes
  Use unique constraint on the foreign keys.
- A 0…1 to 1…1 B: yes
  Add key of B as foreign key to A with constraints
- A 0…1 to 0…∗ B:
- A 0…1 to 1…∗ B:
- A 1…1 to 1…1 B:
- A 1…1 to 0…∗ B:
- A 1…1 to 1…∗ B:
- A 0…∗ to 0…∗ B:
- A 0…∗ to 1…∗ B:
- A 1…∗ to 1…∗ B:

Key Constraints

Which min/max cardinalities can be enforced and how?

- A 0…1 to 0…1 B: yes
  Use unique constraint on the foreign keys.
- A 0…1 to 1…1 B: yes
  Add key of B as foreign key to A with constraints unique & not nullable.
- A 0…1 to 0…∗ B:
- A 0…1 to 1…∗ B:
- A 1…1 to 1…1 B:
- A 1…1 to 0…∗ B:
- A 1…1 to 1…∗ B:
- A 0…∗ to 0…∗ B:
- A 0…∗ to 1…∗ B:
- A 1…∗ to 1…∗ B:

Key Constraints

Which min/max cardinalities can be enforced and how?

- A 0…1 to 0…1 B: yes
  Use unique constraint on the foreign keys.
- A 0…1 to 1…1 B: yes
  Add key of B as foreign key to A with constraints unique & not nullable.
- A 0…1 to 0…∗ B: yes
- A 0…1 to 1…∗ B:
- A 1…1 to 1…1 B:
- A 1…1 to 0…∗ B:
- A 1…1 to 1…∗ B:
- A 0…∗ to 0…∗ B:
- A 0…∗ to 1…∗ B:
- A 1…∗ to 1…∗ B:

Key Constraints

Which min/max cardinalities can be enforced and how?

- A 0…1 to 0…1 B: yes
  Use unique constraint on the foreign keys.
- A 0…1 to 1…1 B: yes
  Add key of B as foreign key to A with constraints unique & not nullable.
- A 0…1 to 0…∗ B: yes
  Add key of A as foreign key to B.
- A 0…1 to 1…∗ B:
- A 1…1 to 1…1 B:
- A 1…1 to 0…∗ B:
- A 1…1 to 1…∗ B:
- A 0…∗ to 0…∗ B:
- A 0…∗ to 1…∗ B:
- A 1…∗ to 1…∗ B:

Key Constraints

Which min/max cardinalities can be enforced and how?

- A 0…1 to 0…1 B: yes
  Use unique constraint on the foreign keys.
- A 0…1 to 1…1 B: yes
  Add key of B as foreign key to A with constraints unique & not nullable.
- A 0…1 to 0…∗ B: yes
  Add key of A as foreign key to B.
- A 0…1 to 1…∗ B: no
- A 1…1 to 1…1 B:
- A 1…1 to 0…∗ B:
- A 1…1 to 1…∗ B:
- A 0…∗ to 0…∗ B:
- A 0…∗ to 1…∗ B:
- A 1…∗ to 1…∗ B:

Key Constraints

Which min/max cardinalities can be enforced and how?

- A 0…1 to 0…1 B: yes
  Use unique constraint on the foreign keys.
- A 0…1 to 1…1 B: yes
  Add key of B as foreign key to A with constraints unique & not nullable.
- A 0…1 to 0…∗ B: yes
  Add key of A as foreign key to B.
- A 0…1 to 1…∗ B: no
- A 1…1 to 1…1 B: yes
- A 1…1 to 0…∗ B:
- A 1…1 to 1…∗ B:
- A 0…∗ to 0…∗ B:
- A 0…∗ to 1…∗ B:
- A 1…∗ to 1…∗ B:

Key Constraints

Which min/max cardinalities can be enforced and how?

- A 0…1 to 0…1 B: yes
  Use unique constraint on the foreign keys.
- A 0…1 to 1…1 B: yes
  Add key of B as foreign key to A with constraints unique & not nullable.
- A 0…1 to 0…∗ B: yes
  Add key of A as foreign key to B.
- A 0…1 to 1…∗ B: no
- A 1…1 to 1…1 B: yes
  Join tables of A and B.
- A 1…1 to 0…∗ B:
- A 1…1 to 1…∗ B:
- A 0…∗ to 0…∗ B:
- A 0…∗ to 1…∗ B:
- A 1…∗ to 1…∗ B:

Key Constraints

Which min/max cardinalities can be enforced and how?

- A 0…1 to 0…1 B: yes
  Use unique constraint on the foreign keys.
- A 0…1 to 1…1 B: yes
  Add key of B as foreign key to A with constraints unique & not nullable.
- A 0…1 to 0…∗ B: yes
  Add key of A as foreign key to B.
- A 0…1 to 1…∗ B: no
- A 1…1 to 1…1 B: yes
  Join tables of A and B.
- A 1…1 to 0…∗ B: yes
- A 1…1 to 1…∗ B:
- A 0…∗ to 0…∗ B:
- A 0…∗ to 1…∗ B:
- A 1…∗ to 1…∗ B:

Key Constraints

Which min/max cardinalities can be enforced and how?

- A 0…1 to 0…1 B: yes
  Use unique constraint on the foreign keys.
- A 0…1 to 1…1 B: yes
  Add key of B as foreign key to A with constraints unique & not nullable.
- A 0…1 to 0…∗ B: yes
  Add key of A as foreign key to B.
- A 0…1 to 1…∗ B: no
- A 1…1 to 1…1 B: yes
  Join tables of A and B.
- A 1…1 to 0…∗ B: yes
  Add key of A as foreign key to B with constraint
- A 1…1 to 1…∗ B:
- A 0…∗ to 0…∗ B:
- A 0…∗ to 1…∗ B:
- A 1…∗ to 1…∗ B:

Key Constraints

Which min/max cardinalities can be enforced and how?

- A 0…1 to 0…1 B: yes
  Use unique constraint on the foreign keys.
- A 0…1 to 1…1 B: yes
  Add key of B as foreign key to A with constraints unique & not nullable.
- A 0…1 to 0…∗ B: yes
  Add key of A as foreign key to B.
- A 0…1 to 1…∗ B: no
- A 1…1 to 1…1 B: yes
  Join tables of A and B.
- A 1…1 to 0…∗ B: yes
  Add key of A as foreign key to B with constraint not nullable.
- A 1…1 to 1…∗ B:
- A 0…∗ to 0…∗ B:
- A 0…∗ to 1…∗ B:
- A 1…∗ to 1…∗ B:

Key Constraints

Which min/max cardinalities can be enforced and how?

- A 0…1 to 0…1 B: yes
  Use unique constraint on the foreign keys.
- A 0…1 to 1…1 B: yes
  Add key of B as foreign key to A with constraints unique & not nullable.
- A 0…1 to 0…∗ B: yes
  Add key of A as foreign key to B.
- A 0…1 to 1…∗ B: no
- A 1…1 to 1…1 B: yes
  Join tables of A and B.
- A 1…1 to 0…∗ B: yes
  Add key of A as foreign key to B with constraint not nullable.
- A 1…1 to 1…∗ B: no
- A 0…∗ to 0…∗ B:
- A 0…∗ to 1…∗ B:
- A 1…∗ to 1…∗ B:

Key Constraints

Which min/max cardinalities can be enforced and how?

- A 0…1 to 0…1 B: yes
  Use unique constraint on the foreign keys.
- A 0…1 to 1…1 B: yes
  Add key of B as foreign key to A with constraints unique & not nullable.
- A 0…1 to 0…∗ B: yes
  Add key of A as foreign key to B.
- A 0…1 to 1…∗ B: no
- A 1…1 to 1…1 B: yes
  Join tables of A and B.
- A 1…1 to 0…∗ B: yes
  Add key of A as foreign key to B with constraint not nullable.
- A 1…1 to 1…∗ B: no
- A 0…∗ to 0…∗ B: yes
- A 0…∗ to 1…∗ B:
- A 1…∗ to 1…∗ B:

Key Constraints

Which min/max cardinalities can be enforced and how?

- A 0…1 to 0…1 B: yes
  Use unique constraint on the foreign keys.
- A 0…1 to 1…1 B: yes
  Add key of B as foreign key to A with constraints unique & not nullable.
- A 0…1 to 0…∗ B: yes
  Add key of A as foreign key to B.
- A 0…1 to 1…∗ B: no
- A 1…1 to 1…1 B: yes
  Join tables of A and B.
- A 1…1 to 0…∗ B: yes
  Add key of A as foreign key to B with constraint not nullable.
- A 1…1 to 1…∗ B: no
- A 0…∗ to 0…∗ B: yes *(relationship set table without constraints)*
- A 0…∗ to 1…∗ B:
- A 1…∗ to 1…∗ B:

Key Constraints

Which min/max cardinalities can be enforced and how?

- A 0…1 to 0…1 B: yes
  Use unique constraint on the foreign keys.
- A 0…1 to 1…1 B: yes
  Add key of B as foreign key to A with constraints unique & not nullable.
- A 0…1 to 0…∗ B: yes
  Add key of A as foreign key to B.
- A 0…1 to 1…∗ B: no
- A 1…1 to 1…1 B: yes
  Join tables of A and B.
- A 1…1 to 0…∗ B: yes
  Add key of A as foreign key to B with constraint not nullable.
- A 1…1 to 1…∗ B: no
- A 0…∗ to 0…∗ B: yes *(relationship set table without constraints)*
- A 0…∗ to 1…∗ B: no
- A 1…∗ to 1…∗ B:

Key Constraints

Which min/max cardinalities can be enforced and how?

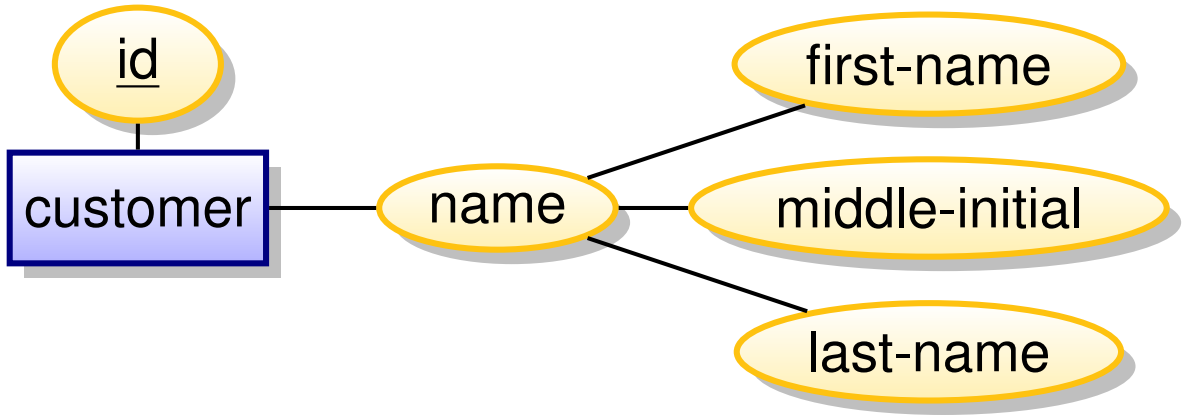
- A 0…1 to 0…1 B: yes
  Use unique constraint on the foreign keys.
- A 0…1 to 1…1 B: yes
  Add key of B as foreign key to A with constraints unique & not nullable.
- A 0…1 to 0…∗ B: yes
  Add key of A as foreign key to B.
- A 0…1 to 1…∗ B: no
- A 1…1 to 1…1 B: yes
  Join tables of A and B.
- A 1…1 to 0…∗ B: yes
  Add key of A as foreign key to B with constraint not nullable.
- A 1…1 to 1…∗ B: no
- A 0…∗ to 0…∗ B: yes *(relationship set table without constraints)*
- A 0…∗ to 1…∗ B: no
- A 1…∗ to 1…∗ B: no

Composite Attributes

Composite attributes are **flattened out** by creating a separate column for each component attribute.

Composite Attributes

Composite attributes are **flattened out** by creating a separate column for each component attribute.



customer

id	first-name	middle-initial	last-name
1	James	null	Smith
2	Joe	J	Jones
3	Jack	F	Brown
4	Harrison	null	Ford

Multi-Valued Attributes

Multi-valued attribute *A* of an entity set *E* is represented by a **separate table** *T* with:

Multi-Valued Attributes

Multi-valued attribute A of an entity set E is represented by a **separate table** T with:

- columns for the primary key of E, and

Multi-Valued Attributes

Multi-valued attribute A of an entity set E is represented by a **separate table** T with:

- columns for the primary key of E, and
- a column for the attribute value

Multi-Valued Attributes

Multi-valued attribute A of an entity set E is represented by a **separate table** T with:

- columns for the primary key of E, and
- a column for the attribute value

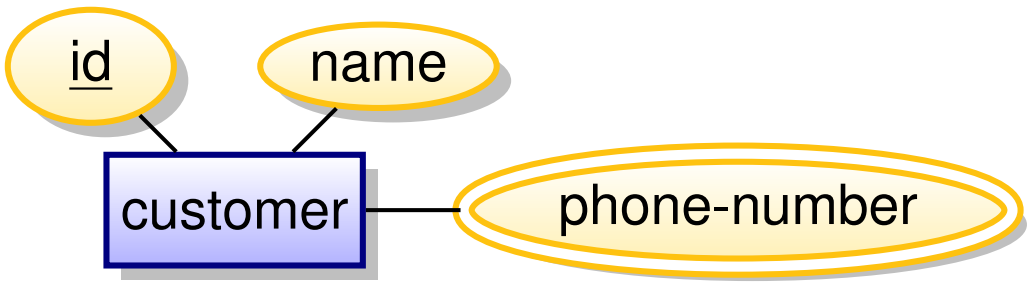
Each single value of the multi-valued attributes gets its own row.

Multi-Valued Attributes

Multi-valued attribute *A* of an entity set *E* is represented by a **separate table** *T* with:

- columns for the primary key of *E*, and
- a column for the attribute value

Each single value of the multi-valued attributes gets its own row.



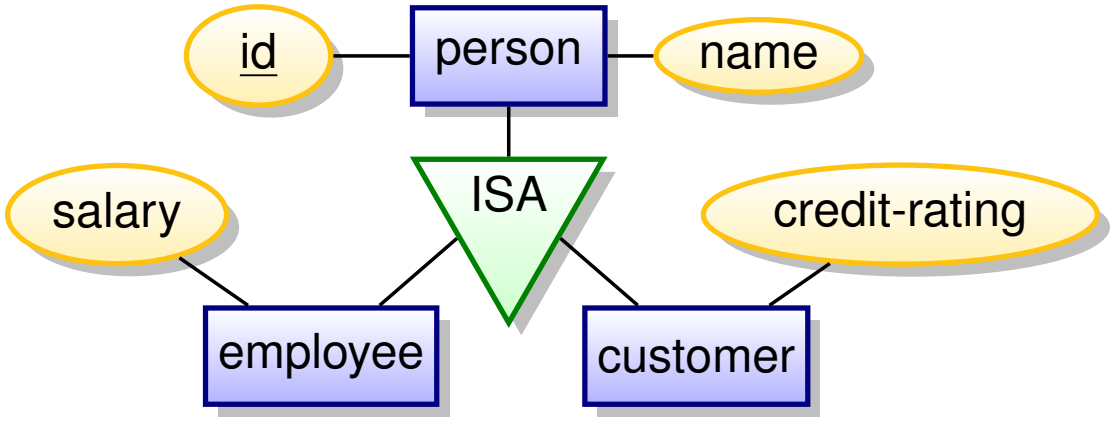
customer

id	name
1	Smith
2	Jones
3	Brown
4	Ford

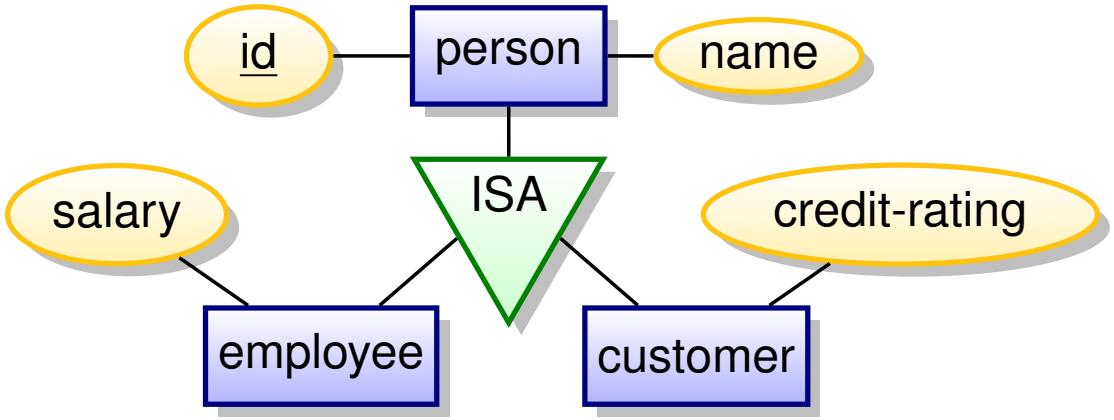
phone-number

id→customer	number
1	06-19348472
1	0346-928475
3	06-13783933
3	0238-187333
3	0192-937189

ISA to Relational Model



ISA to Relational Model



Method 1: hierarchy of tables

- from a table for the higher-level entity set
- form a table for each lover-level entity set; include primary key of higher-level entity set and local attributes

person

id	name
1	James
2	Jones

employee

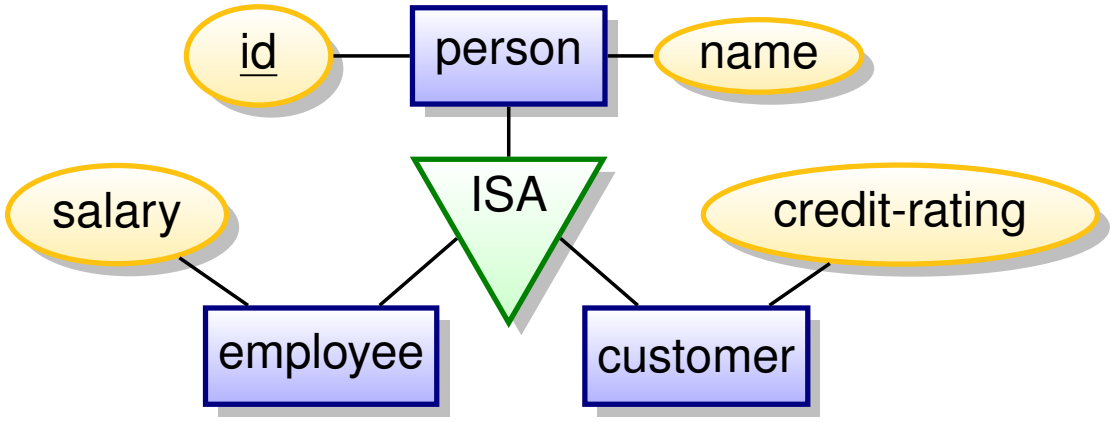
id→person	salary
1	4000

customer

id→person	credit-rating
2	42

- drawback: requires accessing multiple tables

ISA to Relational Model



Method 2: many tables

Form a table for each entity set with all local and inherited attributes.

employee		
id	**name**	**salary**
1	James	4000

customer		
id	**name**	**credit-rating**
2	Jones	42

Typically, we also need a table for person, but. . .

ISA to Relational Model

Method 2: many tables

Form a table for each entity set with all local and inherited attributes.

- **if specialisation is total** then we need no table for the generalised entity (*person*)

ISA to Relational Model

Method 2: many tables

Form a table for each entity set with all local and inherited attributes.

- **if specialisation is total** then we need no table for the generalised entity (*person*)
  - the table for the **generalised entity** can be defined **as a view** containing the union of the specialisation tables

ISA to Relational Model

Method 2: many tables

Form a table for each entity set with all local and inherited attributes.

- **if specialisation is total** then we need no table for the generalised entity (*person*)
  - the table for the **generalised entity** can be defined **as a view** containing the union of the specialisation tables
  - however, the **explicit table might be needed for foreign key constraints**

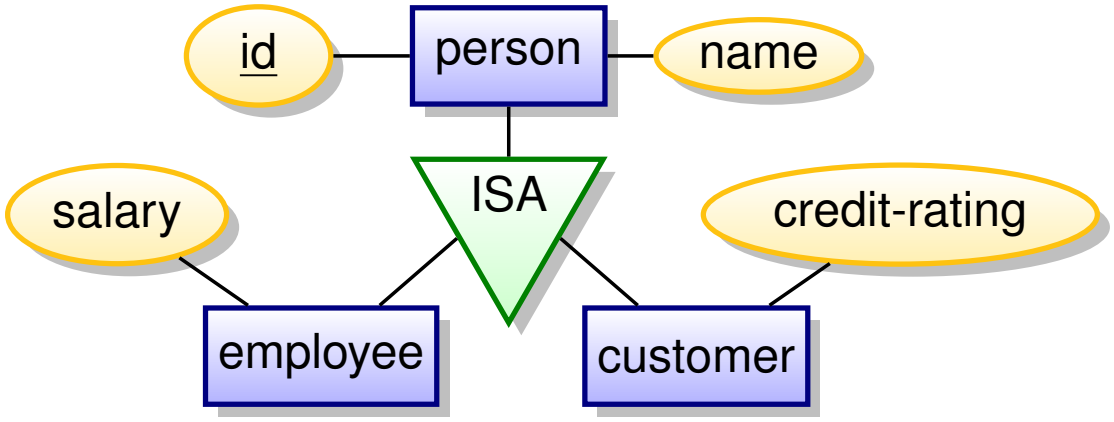
ISA to Relational Model

Method 2: many tables

Form a table for each entity set with all local and inherited attributes.

- **if specialisation is total** then we need no table for the generalised entity (*person*)
  - the table for the **generalised entity** can be defined **as a view** containing the union of the specialisation tables
  - however, the **explicit table might be needed for foreign key constraints**
- drawback: attributes are stored redundantly if an entity belongs to several specialised entity sets (overlapping ISA)
  - e.g. name and address are stored multiple times for someone who is customer and employee

ISA to Relational Model

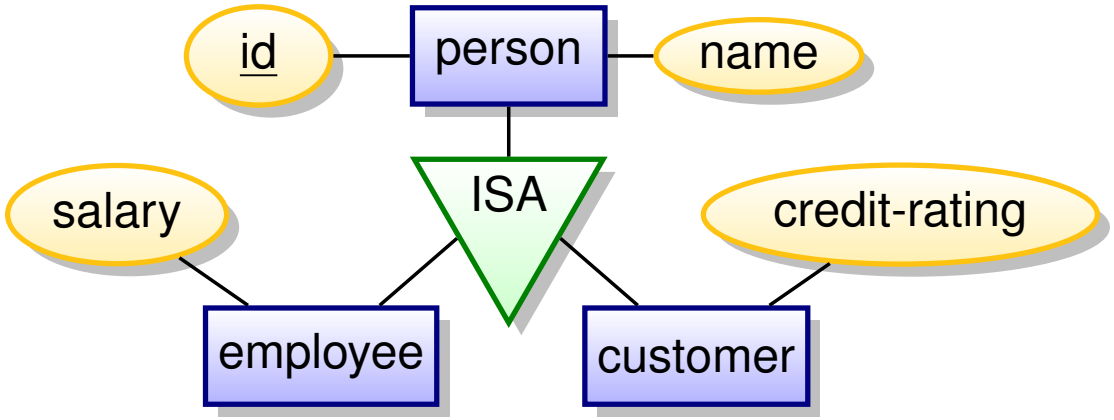


Method 3: one table with null values

From a single table with all local and specialised attributes.

person			
id	**name**	**salary**	**credit-rating**
1	James	4000	null
2	Jones	null	42

ISA to Relational Model



Method 3: one table with null values

From a single table with all local and specialised attributes.

person			
id	**name**	**salary**	**credit-rating**
1	James	4000	null
2	Jones	null	42

- advantage: no joins
- drawback: null values for entities that do not have the corresponding attribute

Primary Keys

customer				
first-name	**last-name**	**phone**	**street**	**city**
Tom	James	06-73917384	Main	London
Jow	Jones	06-18384405	Slater	Paris

What would be a good primary key?

Primary Keys

customer

first-name	last-name	phone	street	city
Tom	James	06-73917384	Main	London
Jow	Jones	06-18384405	Slater	Paris

What would be a good primary key?

Is $\{$ *first-name*, *last-name*, *phone* $\}$ a good key?

Primary Keys

customer				
first-name	**last-name**	**phone**	**street**	**city**
Tom	James	06-73917384	Main	London
Jow	Jones	06-18384405	Slater	Paris

What would be a good primary key?

Is $\{$ *first-name*, *last-name*, *phone* $\}$ a good key?

- is it really unique?

Primary Keys

customer				
first-name	**last-name**	**phone**	**street**	**city**
Tom	James	06-73917384	Main	London
Jow	Jones	06-18384405	Slater	Paris

What would be a good primary key?

Is $\{$ *first-name*, *last-name*, *phone* $\}$ a good key?

- is it really unique?
- the phone number can change

Primary Keys

customer				
first-name	**last-name**	**phone**	**street**	**city**
Tom	James	06-73917384	Main	London
Jow	Jones	06-18384405	Slater	Paris

What would be a good primary key?

Is { *first-name*, *last-name*, *phone* } a good key?

- is it really unique?
- the phone number can change

It is often good to introduce an artificial **internal key**:

- e.g. *customer-id*

Primary Keys

customer				
first-name	**last-name**	**phone**	**street**	**city**
Tom	James	06-73917384	Main	London
Jow	Jones	06-18384405	Slater	Paris

What would be a good primary key?

Is { *first-name*, *last-name*, *phone* } a good key?

- is it really unique?
- the phone number can change

It is often good to introduce an artificial **internal key**:

- e.g. *customer-id*
- advantage: unique, does not change

Primary Keys

customer				
first-name	**last-name**	**phone**	**street**	**city**
Tom	James	06-73917384	Main	London
Jow	Jones	06-18384405	Slater	Paris

What would be a good primary key?

Is { *first-name*, *last-name*, *phone* } a good key?

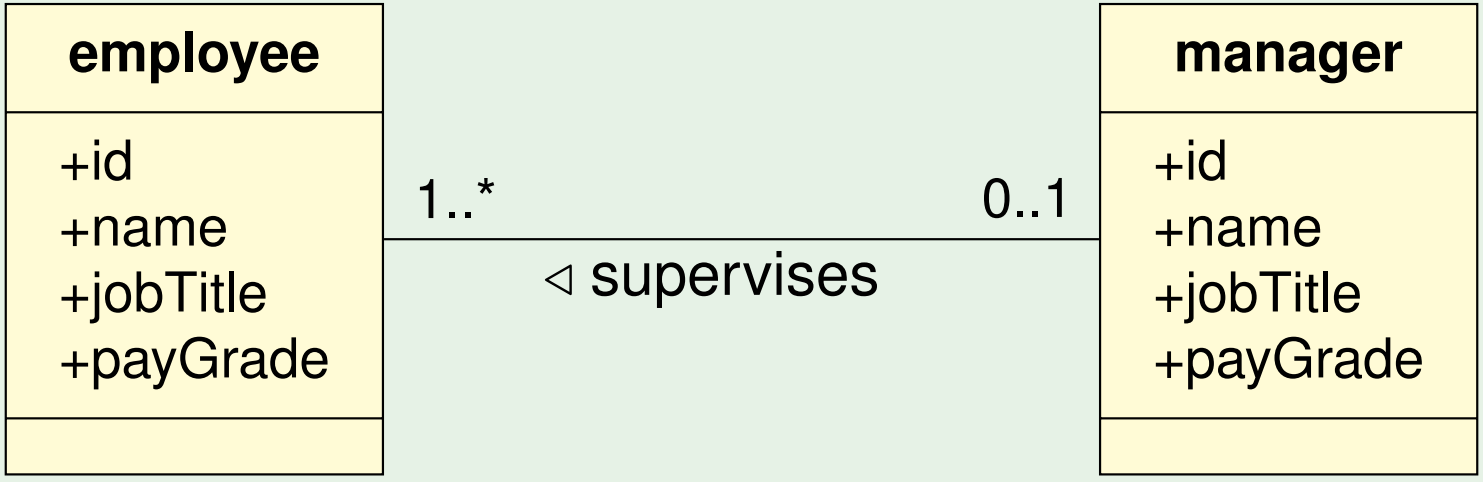
- is it really unique?
- the phone number can change

It is often good to introduce an artificial **internal key**:

- e.g. *customer-id*
- advantage: unique, does not change
- disadvantage: no descriptive meaning

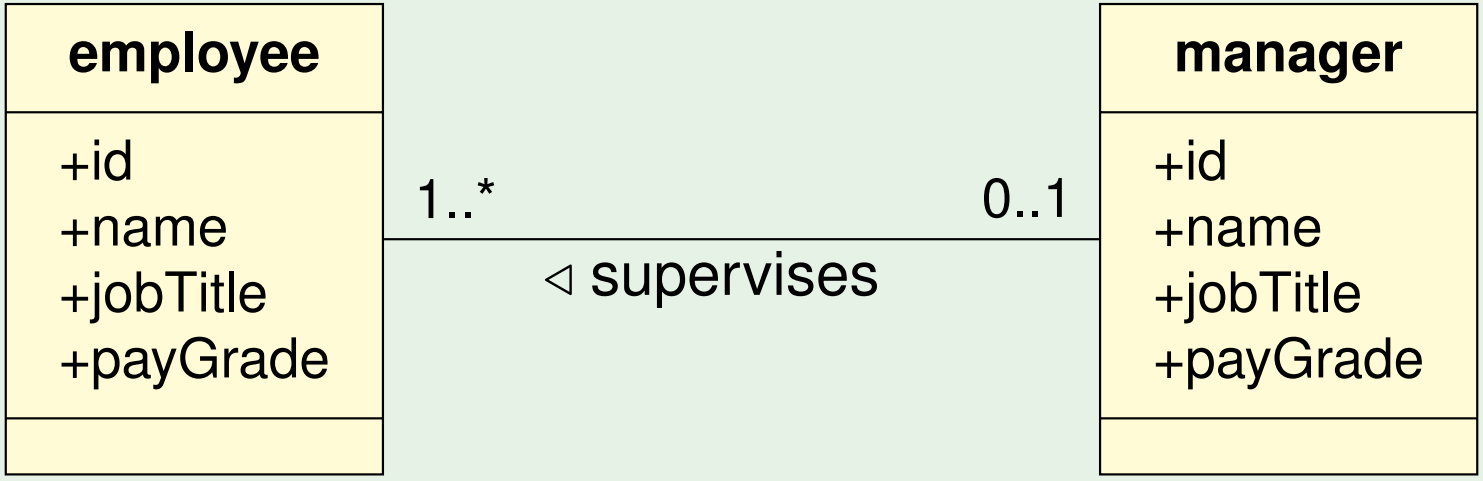
Recursive Associations

Example: an employee is supervised by a manager.



Recursive Associations

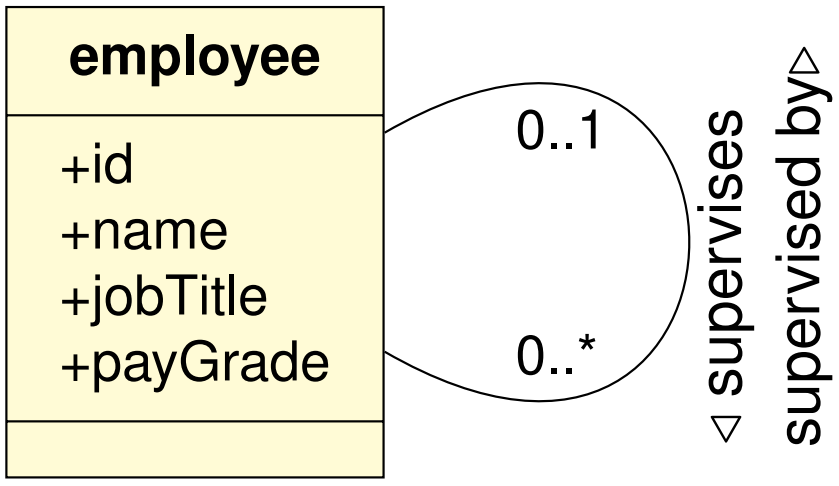
Example: an employee is supervised by a manager.



This diagram is wrong since a manager happens to be an employee as well.

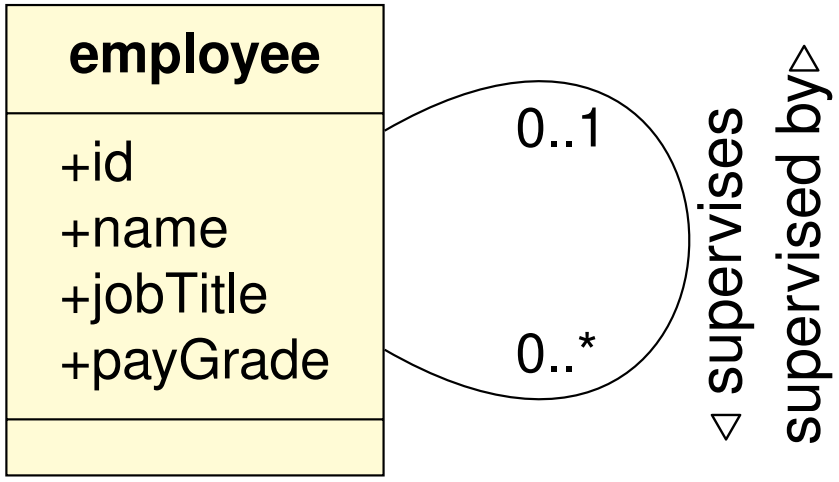
Recursive Associations

The correct way is to use a **recursive association**:



Recursive Associations

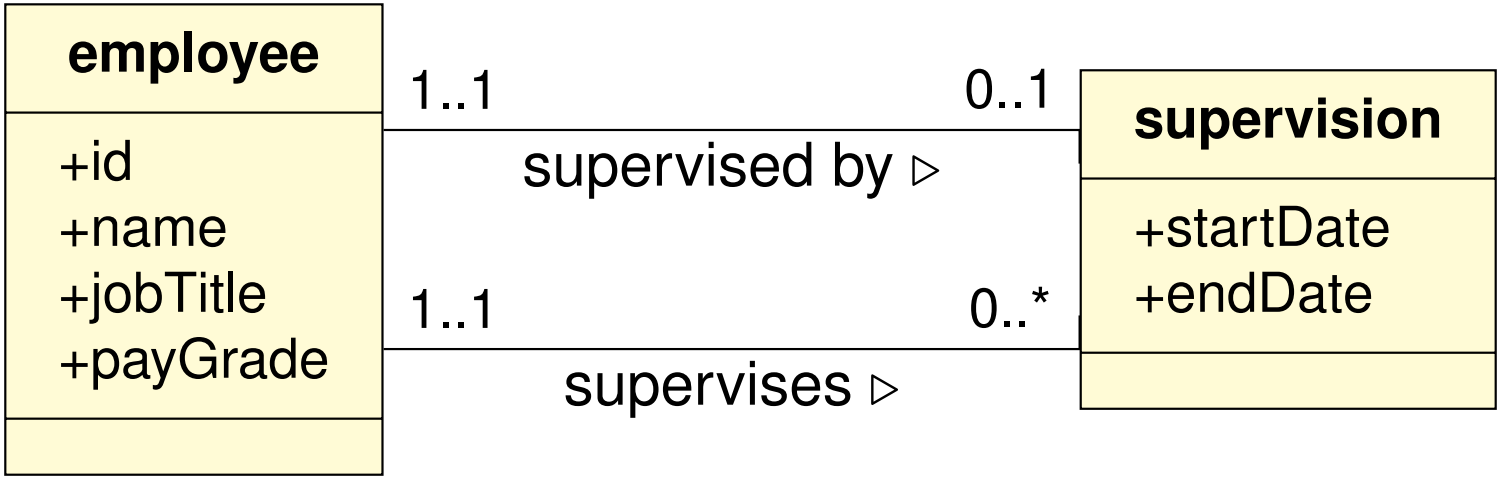
The correct way is to use a **recursive association**:



A **recursive association** translates to a foreign key that refers to the same table.

employee				
id	**name**	**jobTitle**	**payGrade**	**supervisedBy→id**
1	James	. . .	. . .	2
2	Harrison	. . .	. . .	null

Recursive Associations



A **recursive association with attributes** requires a separate table with two foreign keys to the parent table.

From Conceptual to Relational Model: Objectives

After completing this chapter, you should understand:

- **How to translate a conceptual to a relational model**
  - identifying keys
  - internal/external keys
  - (foreign) key constraints
  - multi-valued attributes
  - weak entity sets vs. composition
  - ‘is a’
  - representing cardinalities
  - recursive relationships
  - optimisation: removing relationship tables