Databases – Conceptual to Relational Model

Jörg Endrullis

VU University Amsterdam

Translation :: Basic Translation

From Conceptual to Relational Model

Basic idea

Entity sets and relationship sets are represented as tables.

From Conceptual to Relational Model

Basic idea

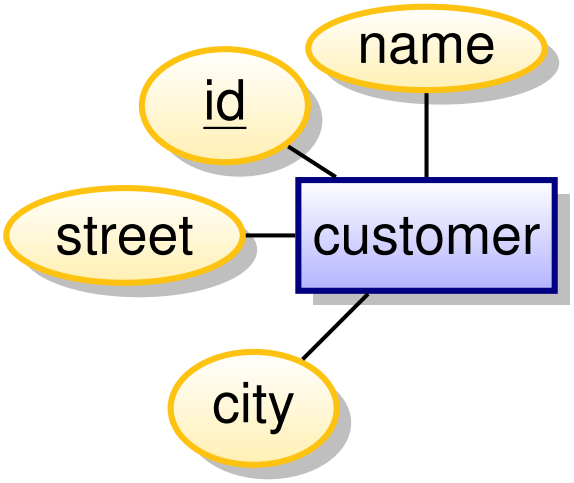
Entity sets and relationship sets are represented as tables.

Roughly:

- **one table for each entity set**
  (name of the table is name of the entity set)
- **one table for each relationship set**
  (name of the table is name of the relationship set)
- **columns roughly correspond to the attributes**

Representing Entity Sets

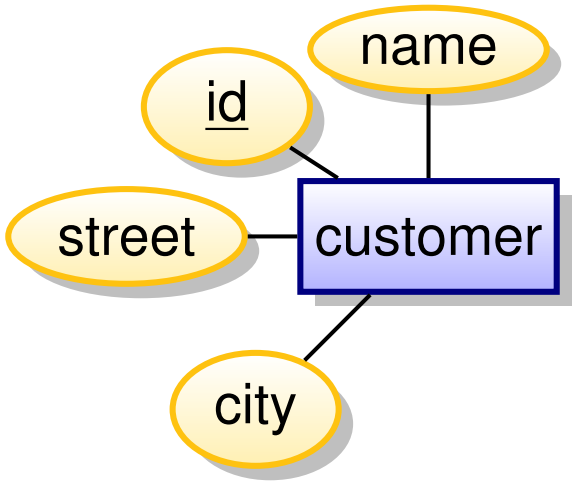
A **strong entity set** becomes



Representing Entity Sets

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

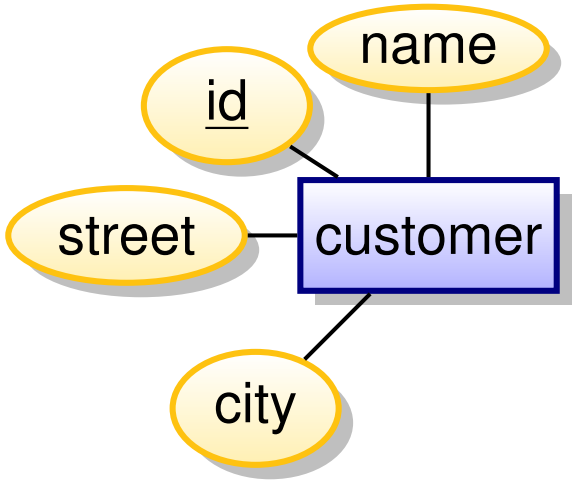
- columns for the attributes



Representing Entity Sets

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

- columns for the attributes

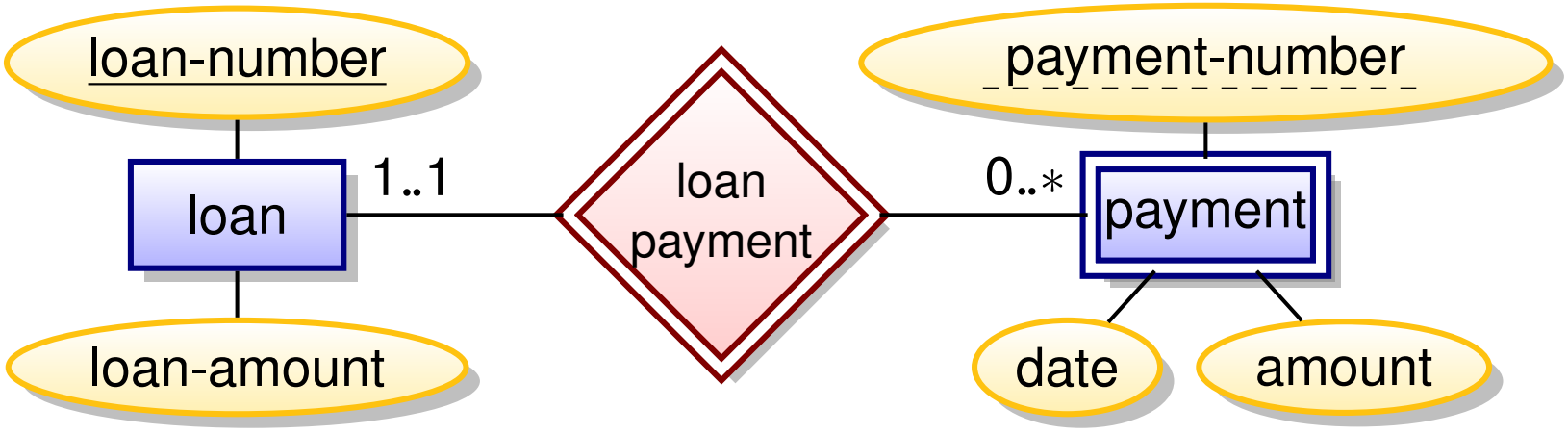


Customer

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

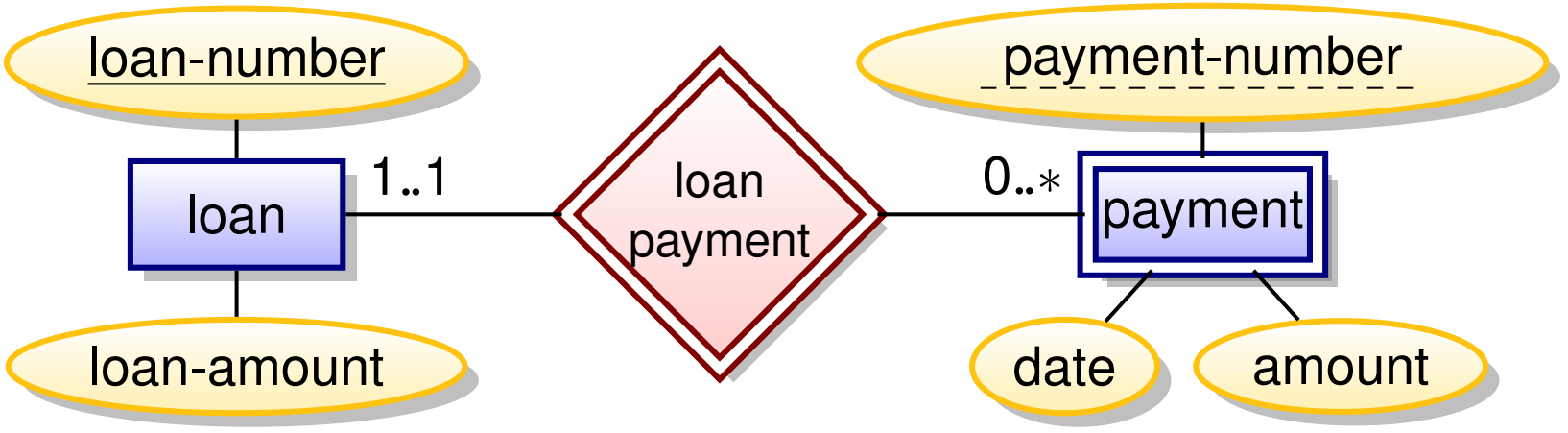
A **weak entity set** becomes



Representing Weak Entity Sets

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

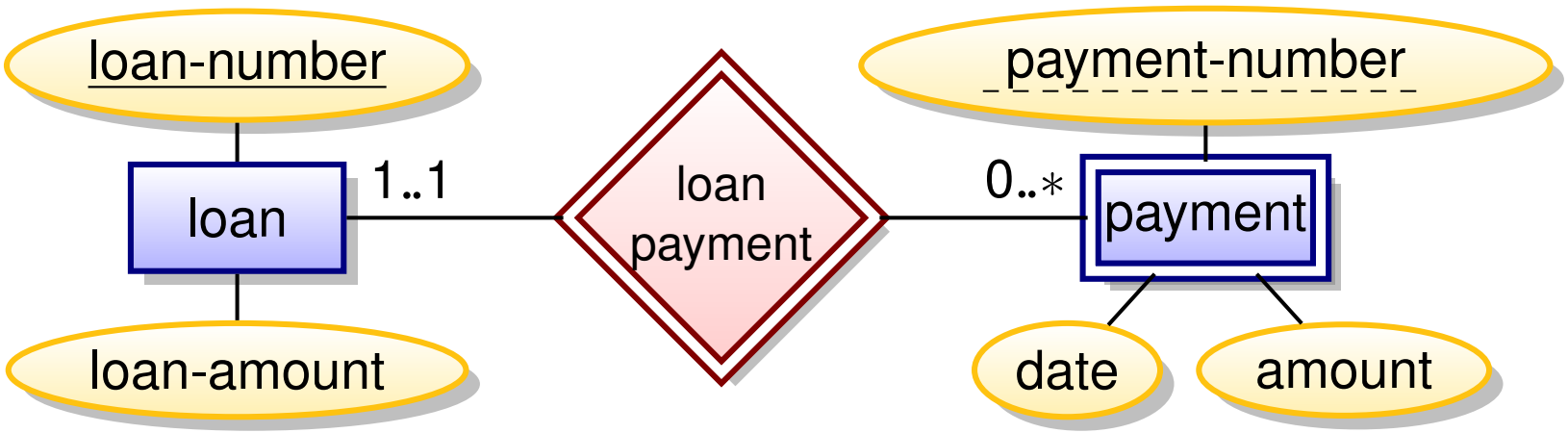
- columns for the attributes, and
- columns for the primary keys of the identifying entity



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

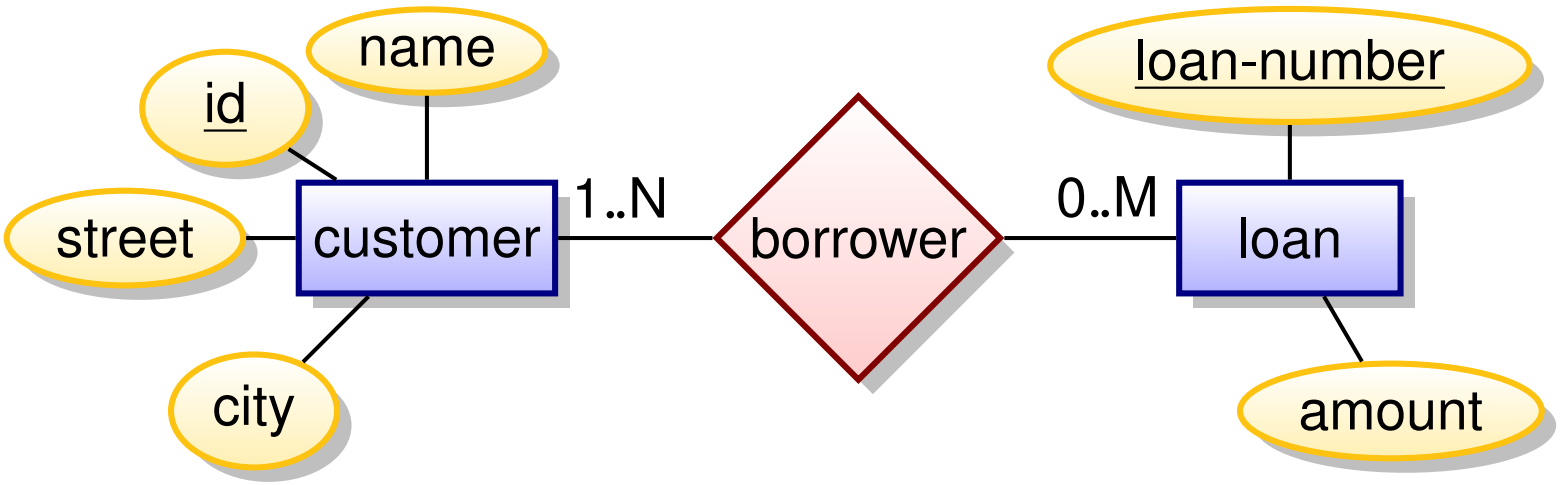


Payment

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

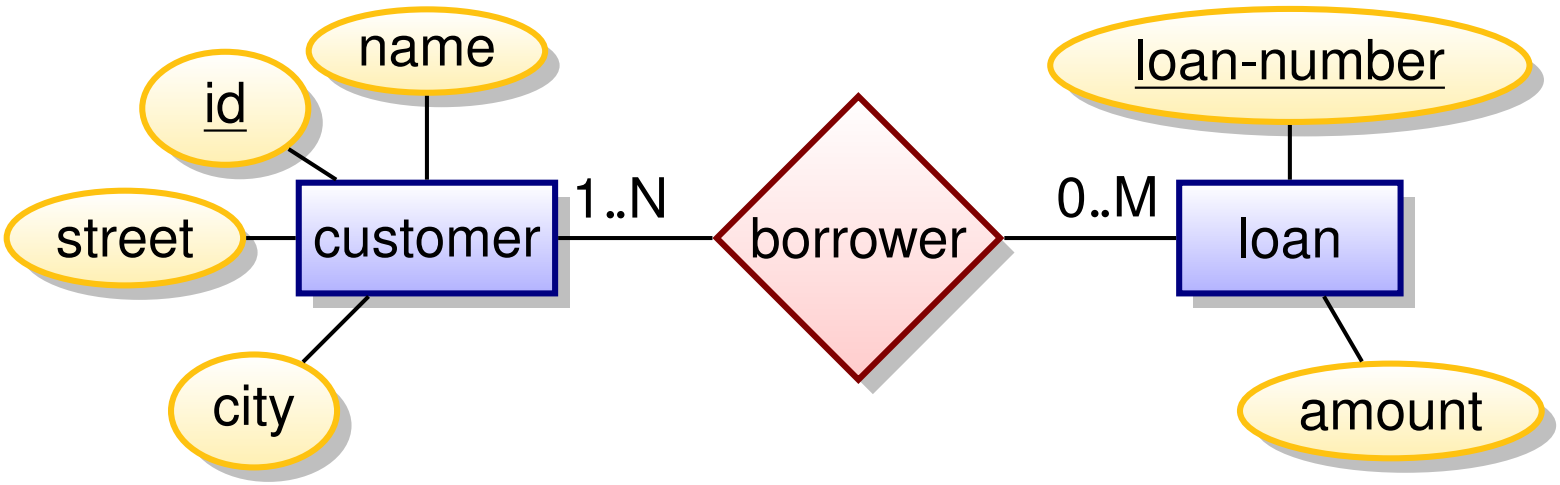
A many-to-many **relationship set** becomes



Representing Relationship Sets

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

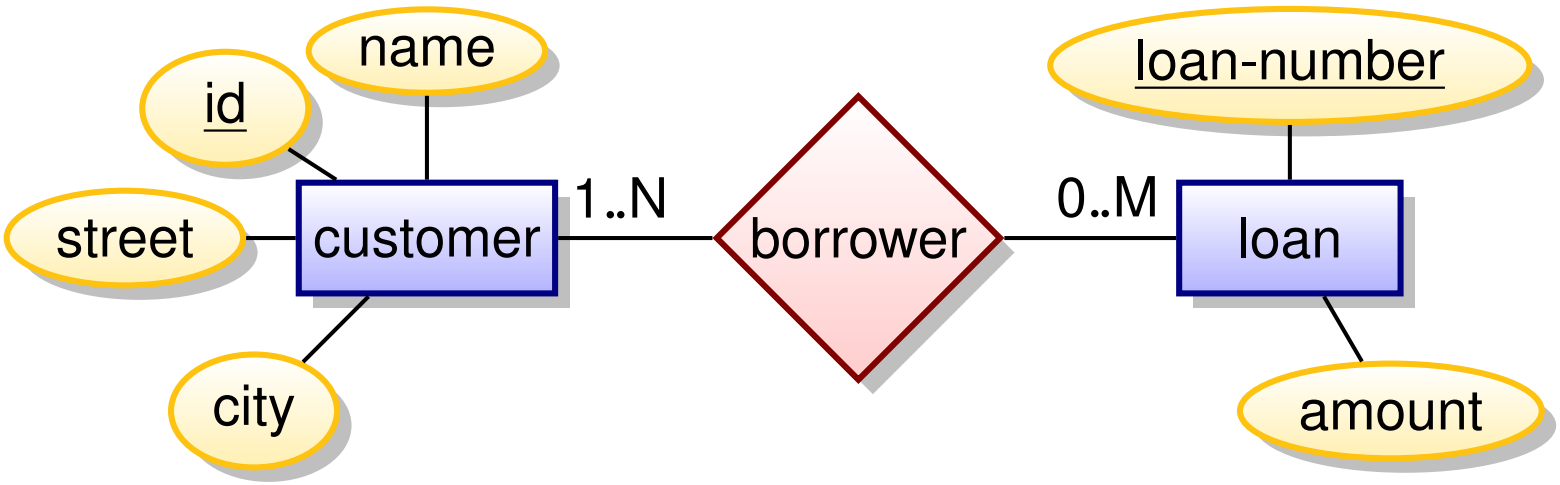
- columns for the attributes of the relationship set, and
- for the primary keys of the participating entity sets.



Representing Relationship Sets

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

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



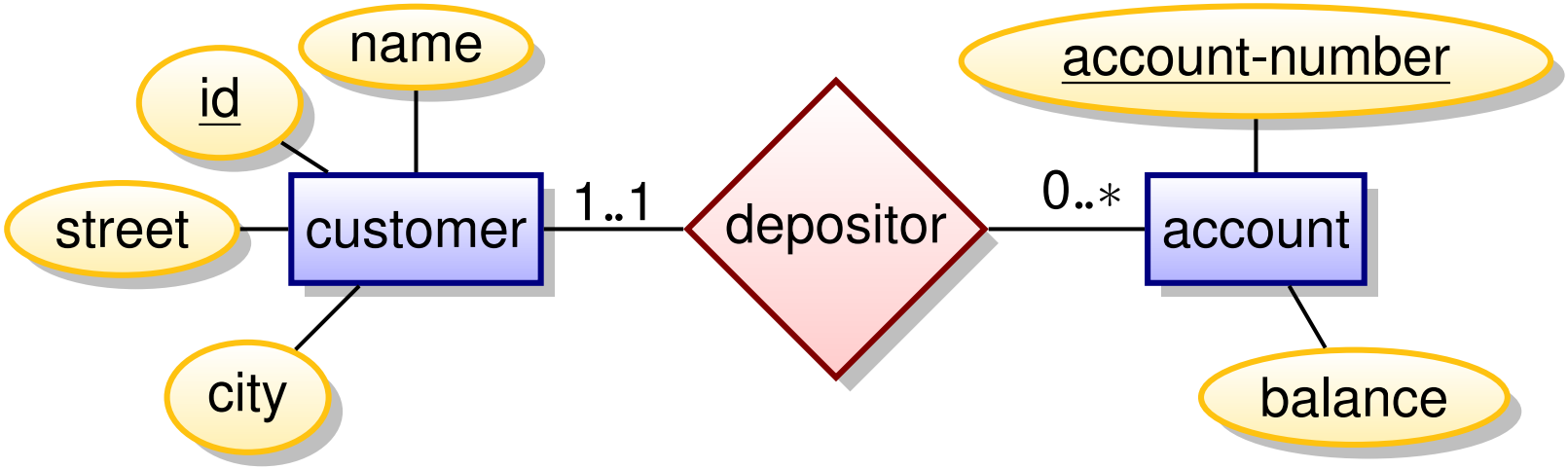
Borrower

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

Translation :: Eliminating Tables

Eliminating Tables

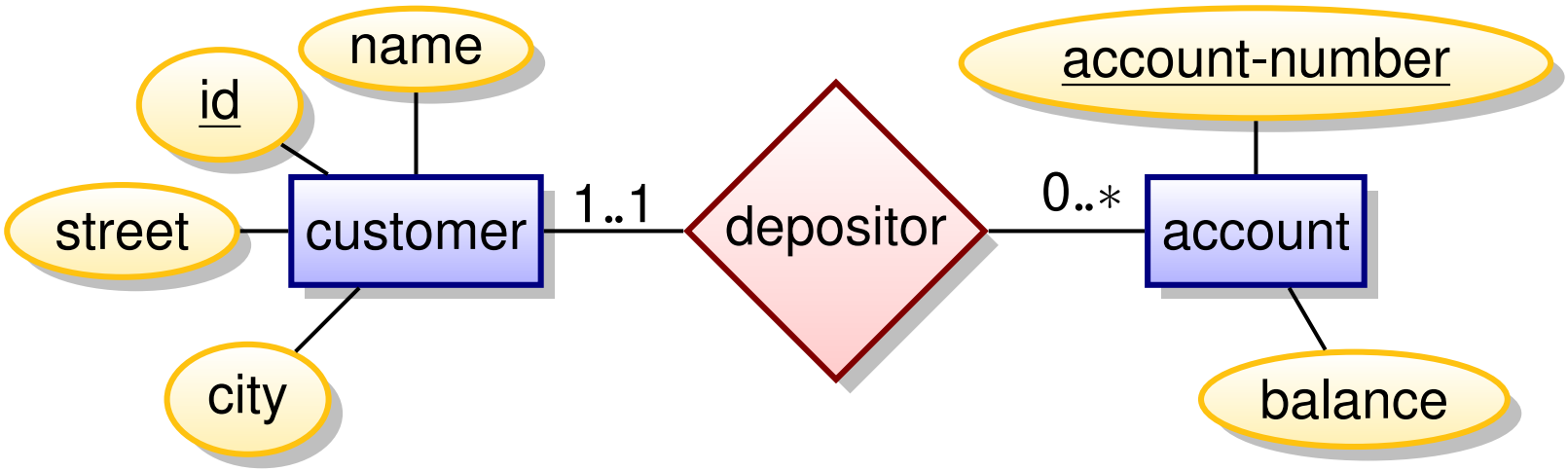
Many-to-(zero or)one relations can be represented by:



Eliminating Tables

Many-to-(zero or)one relations can be represented by:

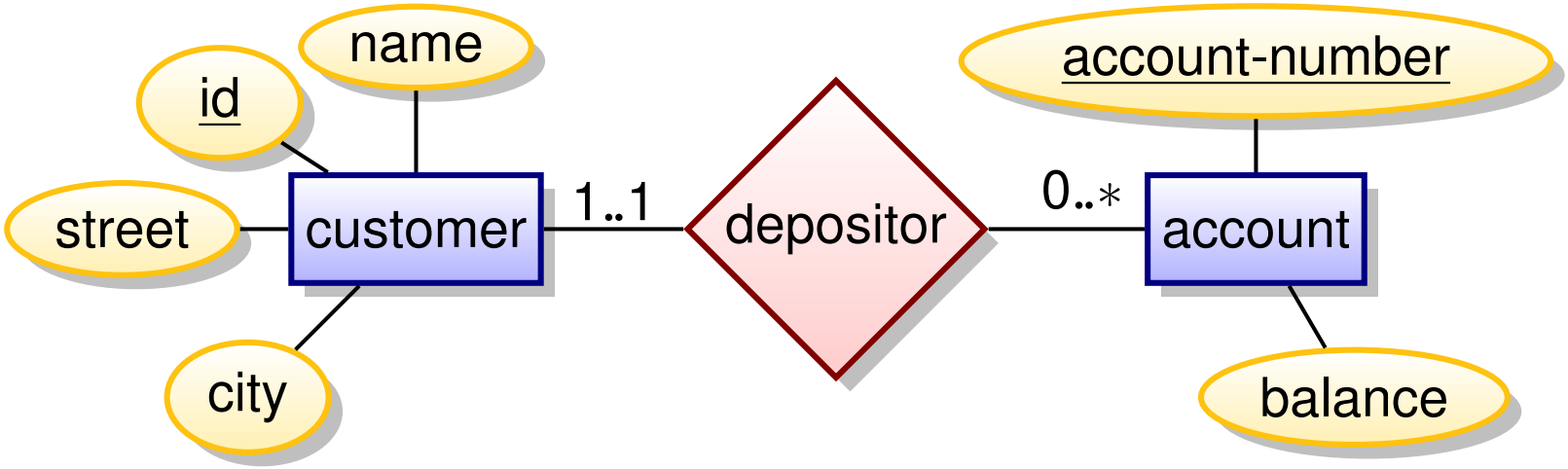
- adding an extra extra attribute/column to the many-side with the primary key of the one-side



Eliminating Tables

Many-to-(zero or)one 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 $\rightarrow$ **Customer**	**account-number**	**balance**
12-0202	83828	125
01-1823	29281	1000

Eliminating Tables

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.

If participation is **total** (1..1), declare foreign key `not null`.

Eliminating Tables

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.

If participation is **total** (1..1), declare foreign key `not null`.

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

Eliminating Tables

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.

If participation is **total** (1..1), declare foreign key `not null`.

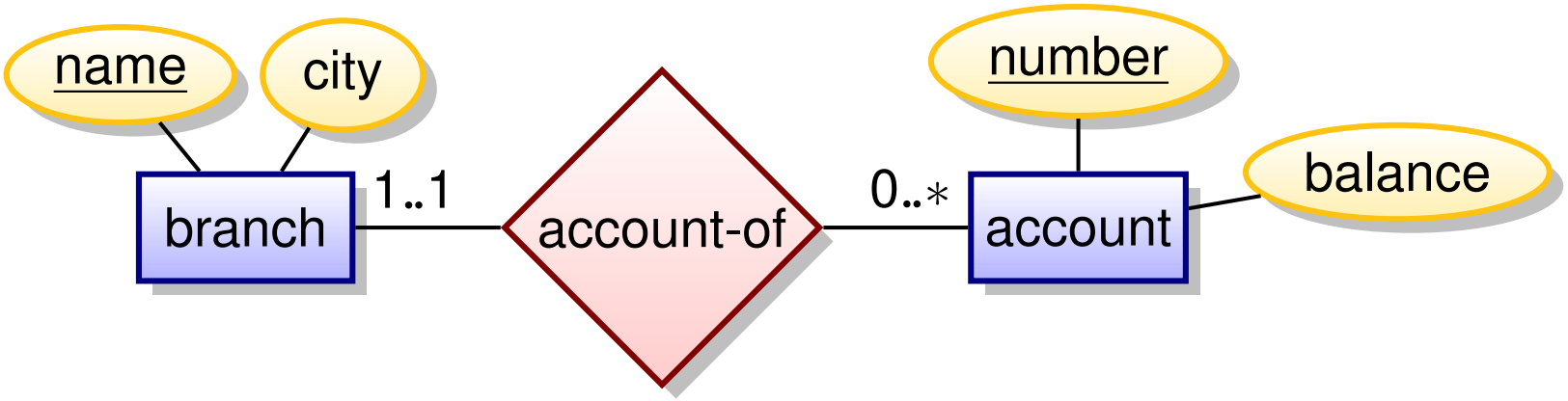
For **one-to-one** (0..1 or 1..1) relationship sets either side can be extended with the key of the other.

Tables for relationship sets linking **weak entity sets** to the identifying entity set can always be eliminated.

No extra table is needed! The table of the weak entity set already contains the key of the identifying entity set.

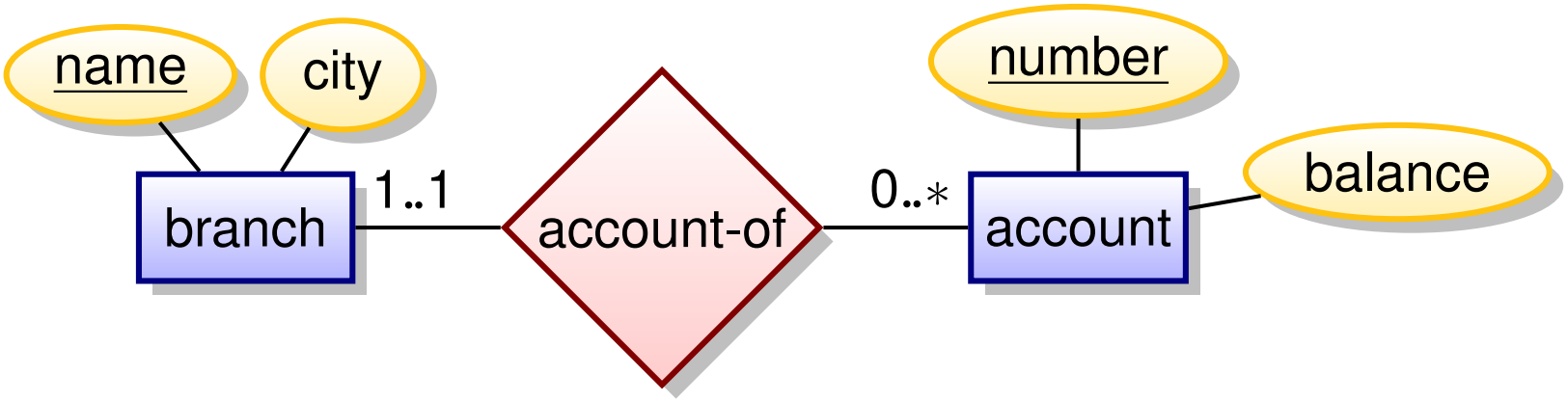
For instance the payment table already contains the full information that would appear in the loan-payment table (loan-number and payment-number).

Eliminating Tables



Basic translation

Eliminating Tables



Basic translation

Branch

name	**city**
branch1	Amsterdam
branch2	Utrecht

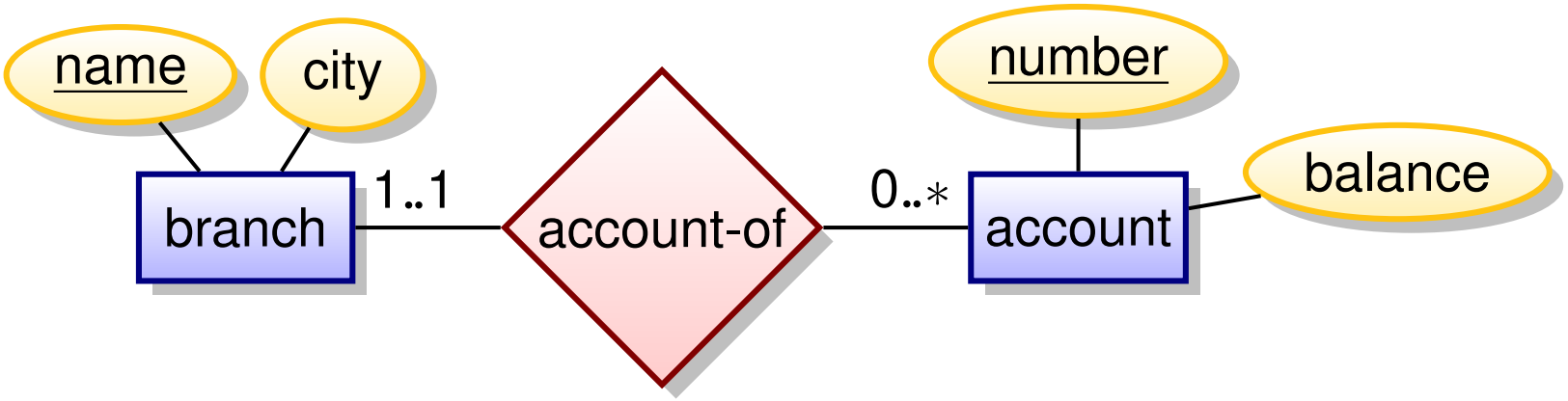
Account-of

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

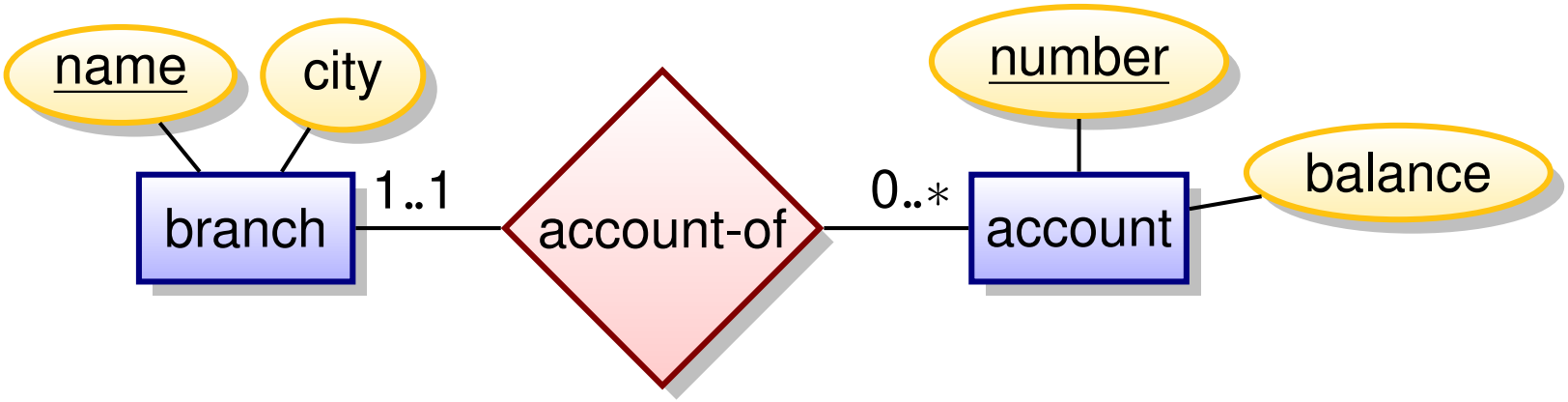
number → **Account**	**name** → **Branch**
83828	branch1
29281	branch2

Account

number	**balance**
83828	125
29281	1000

Optimised translation

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

Translation :: Cardinalities and Constraints

Cardinalities and Constraints

When translating entity sets and relationship sets to tables:

- every table should have a **primary key** (if possible)
- declared **foreign key constraints** for each relation

Cardinalities and Constraints

When translating entity sets and relationship sets to tables:

- every table should have a **primary key** (if possible)
- declared **foreign key constraints** for each relation

Foreign keys should be declared

- **not null**, or not,
- **unique**, or not,

to model the cardinality limits as good as possible.

Cardinalities and Constraints

When translating entity sets and relationship sets to tables:

- every table should have a **primary key** (if possible)
- declared **foreign key constraints** for each relation

Foreign keys should be declared

- **not null**, or not,
- **unique**, or not,

to model the cardinality limits as good as possible.

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

Each row is a relationship among all participating entity sets.

Cardinalities and Constraints

When translating entity sets and relationship sets to tables:

- every table should have a **primary key** (if possible)
- declared **foreign key constraints** for each relation

Foreign keys should be declared

- **not null**, or not,
- **unique**, or not,

to model the cardinality limits as good as possible.

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

Each row is a relationship among all participating entity sets.

Attributes should be declared **not null** and/or **unique** if appropriate.

Cardinalities and Constraints

Which min/max cardinalities can be enforced and how?

- A 0..$*$ to 0..$*$ B:
- A 0..1 to 0..$*$ B:
- A 1..1 to 0..$*$ B:
- A 0..1 to 0..1 B:
- A 0..1 to 1..1 B:
- A 1..1 to 1..1 B:
- A *M*..*N* to 1..$*$ B:

Cardinalities and Constraints

Which min/max cardinalities can be enforced and how?

- A $0..*$ to $0..*$ B: yes
- A $0..1$ to $0..*$ B:
- A $1..1$ to $0..*$ B:
- A $0..1$ to $0..1$ B:
- A $0..1$ to $1..1$ B:
- A $1..1$ to $1..1$ B:
- A $M..N$ to $1..*$ B:

Cardinalities and Constraints

Which min/max cardinalities can be enforced and how?

- A 0..∗ to 0..∗ B: yes
  A separate relationship set table.
- A 0..1 to 0..∗ B:
- A 1..1 to 0..∗ B:
- A 0..1 to 0..1 B:
- A 0..1 to 1..1 B:
- A 1..1 to 1..1 B:
- A *M*..*N* to 1..∗ B:

Cardinalities and Constraints

Which min/max cardinalities can be enforced and how?

- A 0..∗ to 0..∗ B: yes
  A separate relationship set table.
- A 0..1 to 0..∗ B: yes
- A 1..1 to 0..∗ B:
- A 0..1 to 0..1 B:
- A 0..1 to 1..1 B:
- A 1..1 to 1..1 B:
- A *M*..*N* to 1..∗ B:

Cardinalities and Constraints

Which min/max cardinalities can be enforced and how?

- A 0..$*$ to 0..$*$ B: yes
  A separate relationship set table.
- A 0..1 to 0..$*$ B: yes
  Add key of A as foreign key to B.
- A 1..1 to 0..$*$ B:
- A 0..1 to 0..1 B:
- A 0..1 to 1..1 B:
- A 1..1 to 1..1 B:
- A *M*..*N* to 1..$*$ B:

Cardinalities and Constraints

Which min/max cardinalities can be enforced and how?

- A 0..$*$ to 0..$*$ B: yes
  A separate relationship set table.
- A 0..1 to 0..$*$ B: yes
  Add key of A as foreign key to B.
- A 1..1 to 0..$*$ B: yes
- A 0..1 to 0..1 B:
- A 0..1 to 1..1 B:
- A 1..1 to 1..1 B:
- A *M*..*N* to 1..$*$ B:

Cardinalities and Constraints

Which min/max cardinalities can be enforced and how?

- A 0..∗ to 0..∗ B: yes
  A separate relationship set table.
- A 0..1 to 0..∗ B: yes
  Add key of A as foreign key to B.
- A 1..1 to 0..∗ B: yes
  Add key of A as foreign key to B
- A 0..1 to 0..1 B:
- A 0..1 to 1..1 B:
- A 1..1 to 1..1 B:
- A *M*..*N* to 1..∗ B:

Cardinalities and Constraints

Which min/max cardinalities can be enforced and how?

- A 0..∗ to 0..∗ B: yes
  A separate relationship set table.
- A 0..1 to 0..∗ B: yes
  Add key of A as foreign key to B.
- A 1..1 to 0..∗ B: yes
  Add key of A as foreign key to B with constraint `not null`.
- A 0..1 to 0..1 B:
- A 0..1 to 1..1 B:
- A 1..1 to 1..1 B:
- A *M*..*N* to 1..∗ B:

Cardinalities and Constraints

Which min/max cardinalities can be enforced and how?

- A 0..∗ to 0..∗ B: yes
  A separate relationship set table.
- A 0..1 to 0..∗ B: yes
  Add key of A as foreign key to B.
- A 1..1 to 0..∗ B: yes
  Add key of A as foreign key to B with constraint `not null`.
- A 0..1 to 0..1 B: yes
- A 0..1 to 1..1 B:
- A 1..1 to 1..1 B:
- A *M*..*N* to 1..∗ B:

Cardinalities and Constraints

Which min/max cardinalities can be enforced and how?

- A 0..∗ to 0..∗ B: yes
  A separate relationship set table.
- A 0..1 to 0..∗ B: yes
  Add key of A as foreign key to B.
- A 1..1 to 0..∗ B: yes
  Add key of A as foreign key to B with constraint `not null`.
- A 0..1 to 0..1 B: yes
  Add key of A (or B) as foreign key to B (or A)
- A 0..1 to 1..1 B:
- A 1..1 to 1..1 B:
- A *M*..*N* to 1..∗ B:

Cardinalities and Constraints

Which min/max cardinalities can be enforced and how?

- A 0..∗ to 0..∗ B: yes
  A separate relationship set table.
- A 0..1 to 0..∗ B: yes
  Add key of A as foreign key to B.
- A 1..1 to 0..∗ B: yes
  Add key of A as foreign key to B with constraint `not null`.
- A 0..1 to 0..1 B: yes
  Add key of A (or B) as foreign key to B (or A) with constraint `unique`.
- A 0..1 to 1..1 B:
- A 1..1 to 1..1 B:
- A *M*..*N* to 1..∗ B:

Cardinalities and Constraints

Which min/max cardinalities can be enforced and how?

- A 0..∗ to 0..∗ B: yes
  A separate relationship set table.
- A 0..1 to 0..∗ B: yes
  Add key of A as foreign key to B.
- A 1..1 to 0..∗ B: yes
  Add key of A as foreign key to B with constraint `not null`.
- A 0..1 to 0..1 B: yes
  Add key of A (or B) as foreign key to B (or A) with constraint `unique`.
- A 0..1 to 1..1 B: yes
- A 1..1 to 1..1 B:
- A *M*..*N* to 1..∗ B:

Cardinalities and Constraints

Which min/max cardinalities can be enforced and how?

- A 0..∗ to 0..∗ B: yes
  A separate relationship set table.
- A 0..1 to 0..∗ B: yes
  Add key of A as foreign key to B.
- A 1..1 to 0..∗ B: yes
  Add key of A as foreign key to B with constraint `not null`*.*
- A 0..1 to 0..1 B: yes
  Add key of A (or B) as foreign key to B (or A) with constraint `unique`*.*
- A 0..1 to 1..1 B: yes
  Add key of B as foreign key to A
- A 1..1 to 1..1 B:
- A *M*..*N* to 1..∗ B:

Cardinalities and Constraints

Which min/max cardinalities can be enforced and how?

- A 0..∗ to 0..∗ B: yes
  A separate relationship set table.
- A 0..1 to 0..∗ B: yes
  Add key of A as foreign key to B.
- A 1..1 to 0..∗ B: yes
  Add key of A as foreign key to B with constraint `not null`*.*
- A 0..1 to 0..1 B: yes
  Add key of A (or B) as foreign key to B (or A) with constraint `unique`*.*
- A 0..1 to 1..1 B: yes
  Add key of B as foreign key to A with constraints `unique` *&* `not null`*.*
- A 1..1 to 1..1 B:
- A *M*..*N* to 1..∗ B:

Cardinalities and Constraints

Which min/max cardinalities can be enforced and how?

- A 0..∗ to 0..∗ B: yes
  A separate relationship set table.
- A 0..1 to 0..∗ B: yes
  Add key of A as foreign key to B.
- A 1..1 to 0..∗ B: yes
  Add key of A as foreign key to B with constraint `not null`*.*
- A 0..1 to 0..1 B: yes
  Add key of A (or B) as foreign key to B (or A) with constraint `unique`*.*
- A 0..1 to 1..1 B: yes
  Add key of B as foreign key to A with constraints `unique` *&* `not null`*.*
- A 1..1 to 1..1 B: yes
- A *M*..*N* to 1..∗ B:

Cardinalities and Constraints

Which min/max cardinalities can be enforced and how?

- A 0..∗ to 0..∗ B: yes
  A separate relationship set table.
- A 0..1 to 0..∗ B: yes
  Add key of A as foreign key to B.
- A 1..1 to 0..∗ B: yes
  Add key of A as foreign key to B with constraint `not null`*.*
- A 0..1 to 0..1 B: yes
  Add key of A (or B) as foreign key to B (or A) with constraint `unique`*.*
- A 0..1 to 1..1 B: yes
  Add key of B as foreign key to A with constraints `unique` *&* `not null`*.*
- A 1..1 to 1..1 B: yes
  Join tables of A and B.
- A *M*..*N* to 1..∗ B:

Cardinalities and Constraints

Which min/max cardinalities can be enforced and how?

- A 0..∗ to 0..∗ B: yes
  A separate relationship set table.
- A 0..1 to 0..∗ B: yes
  Add key of A as foreign key to B.
- A 1..1 to 0..∗ B: yes
  Add key of A as foreign key to B with constraint `not null`*.*
- A 0..1 to 0..1 B: yes
  Add key of A (or B) as foreign key to B (or A) with constraint `unique`*.*
- A 0..1 to 1..1 B: yes
  Add key of B as foreign key to A with constraints `unique` *&* `not null`*.*
- A 1..1 to 1..1 B: yes
  Join tables of A and B.
- A *M*..*N* to 1..∗ B: no

Cardinalities and Constraints

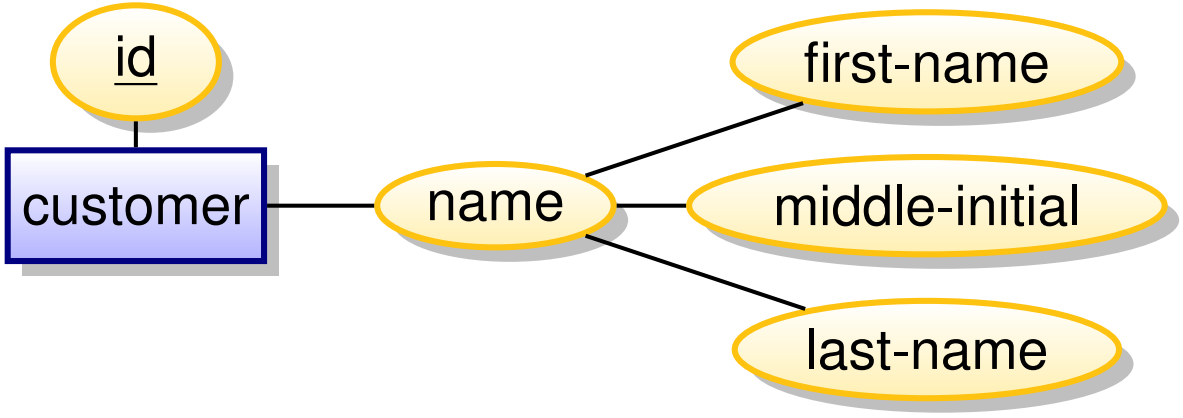
Which min/max cardinalities can be enforced and how?

- A 0..∗ to 0..∗ B: yes
  A separate relationship set table.
- A 0..1 to 0..∗ B: yes
  Add key of A as foreign key to B.
- A 1..1 to 0..∗ B: yes
  Add key of A as foreign key to B with constraint `not null`*.*
- A 0..1 to 0..1 B: yes
  Add key of A (or B) as foreign key to B (or A) with constraint `unique`*.*
- A 0..1 to 1..1 B: yes
  Add key of B as foreign key to A with constraints `unique` *&* `not null`*.*
- A 1..1 to 1..1 B: yes
  Join tables of A and B.
- A *M*..*N* to 1..∗ B: no
  Workaround: approximate the cardinality limit 1..∗ *with* 0..∗*.*

Translation :: Composite & Multi-Valued Attributes

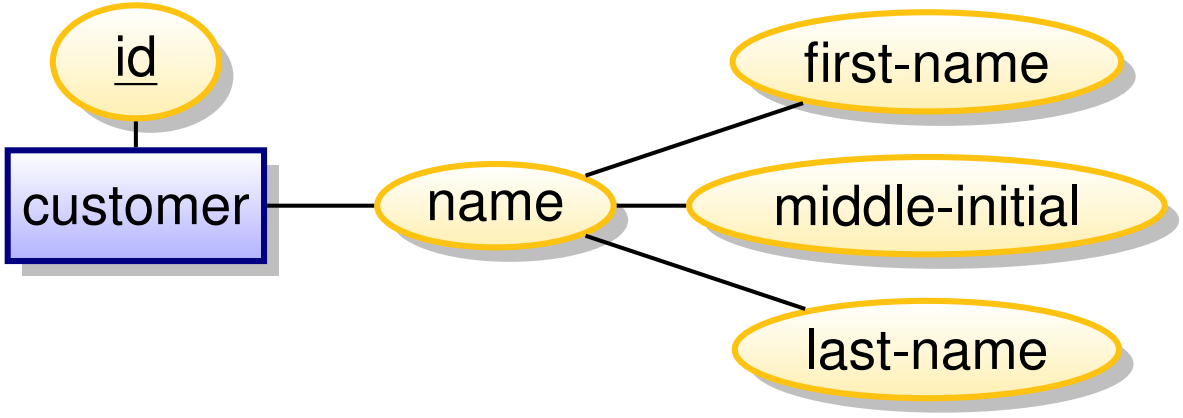
Composite Attributes

Composite attributes are



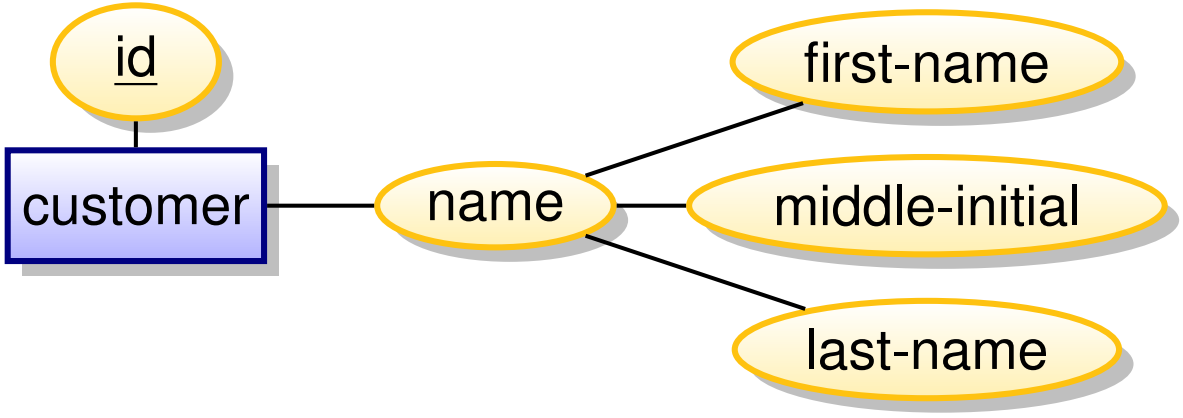
Composite Attributes

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.

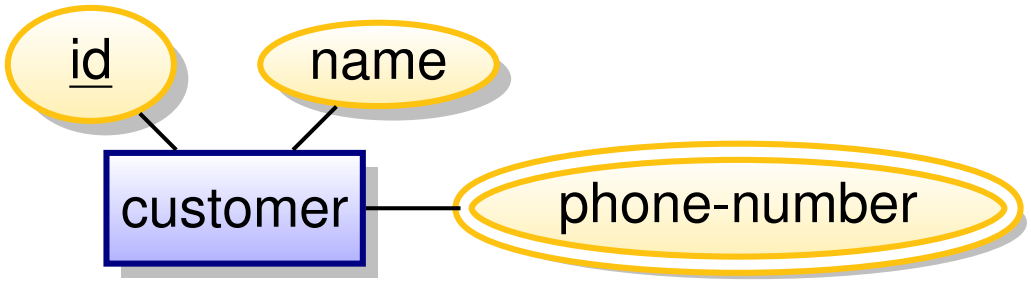


Customer

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

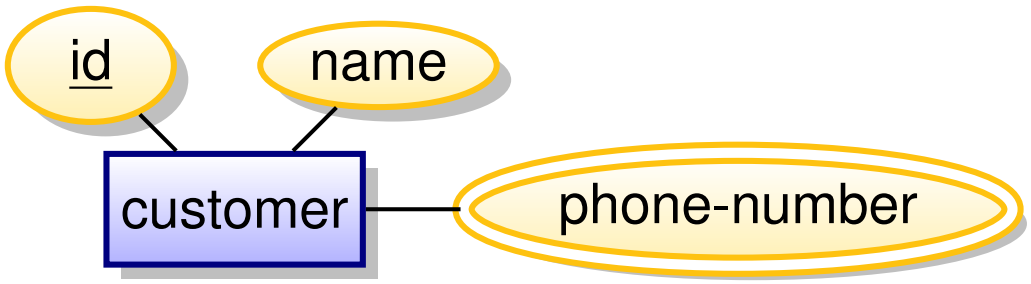


Multi-Valued Attributes

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

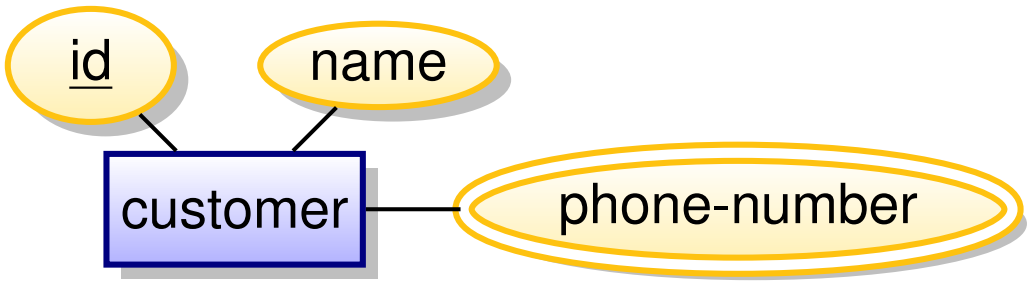


Multi-Valued Attributes

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

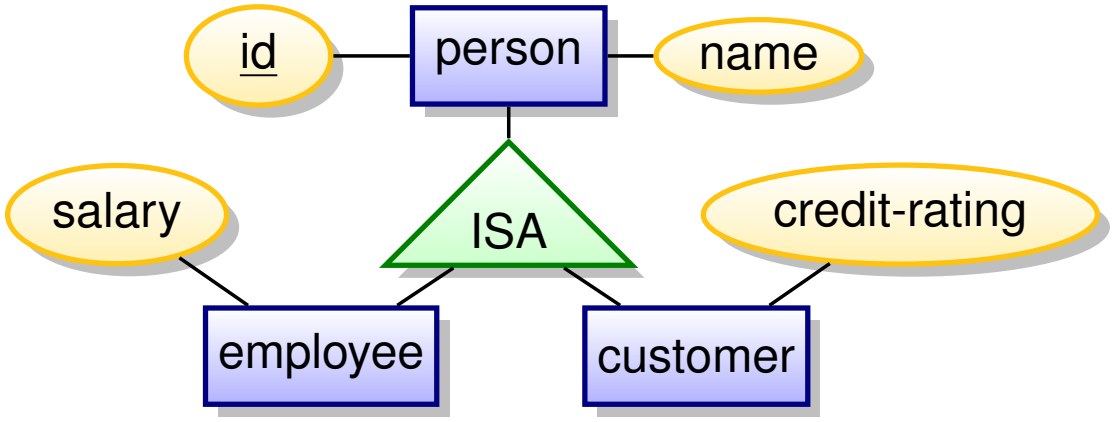
id	name
1	Smith
2	Jones
3	Brown
4	Ford

Phone-number

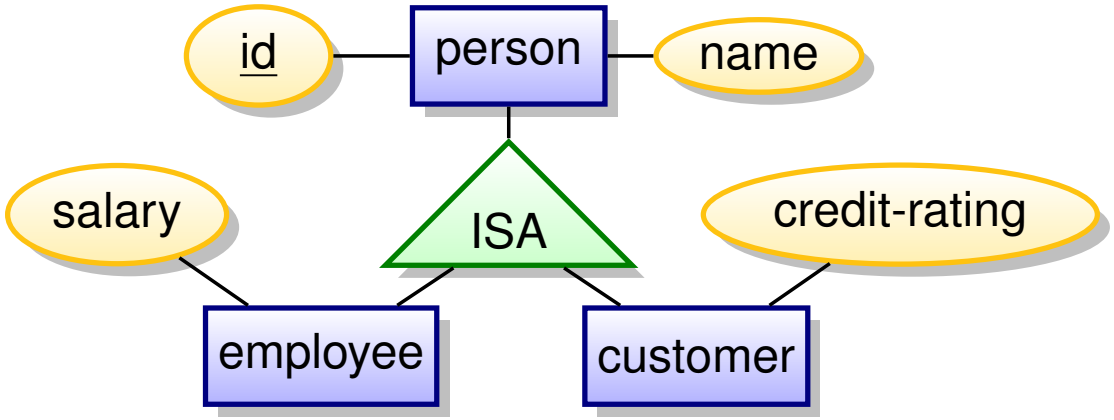
id → Customer	number
1	06-19348472
1	0346-928475
3	06-13783933
3	0238-187333
3	0192-937189

Translation :: ISA

ISA to Relational Model



ISA to Relational Model



Method 1: hierarchy of tables

- a table for the higher-level entity set
- 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

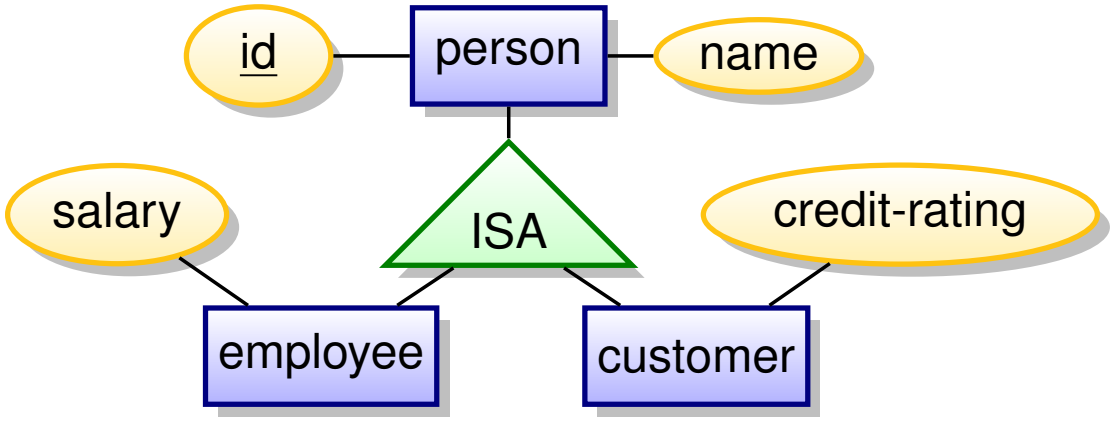
id → Person	salary
1	4000

Customer

id → Person	credit-rating
2	42

Minor 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*):

Table for the **generalised entity set** 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.

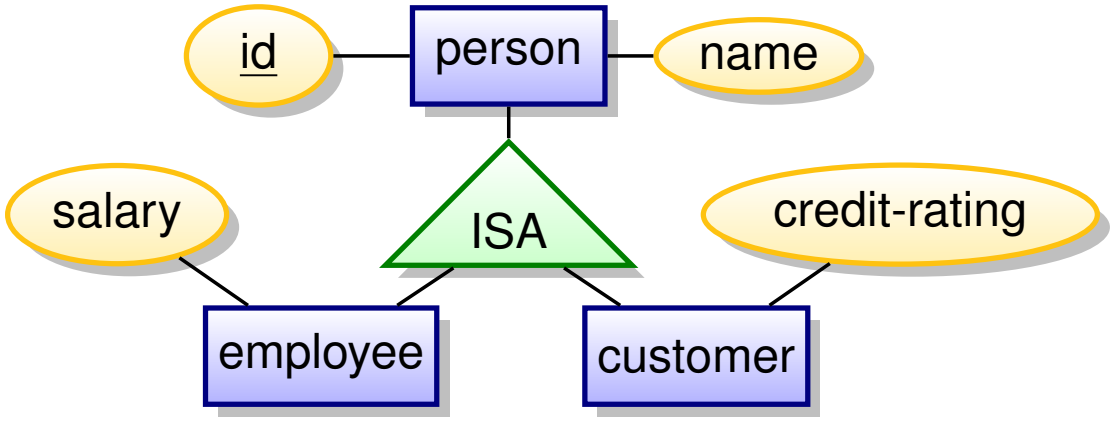
If specialisation is total then we need no table for the generalised entity (*person*):

Table for the **generalised entity set** can be defined **as a view** containing the union of the specialisation tables

Drawback:

- explicit table for the generalised entity might be needed for foreign key constraints.
- 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

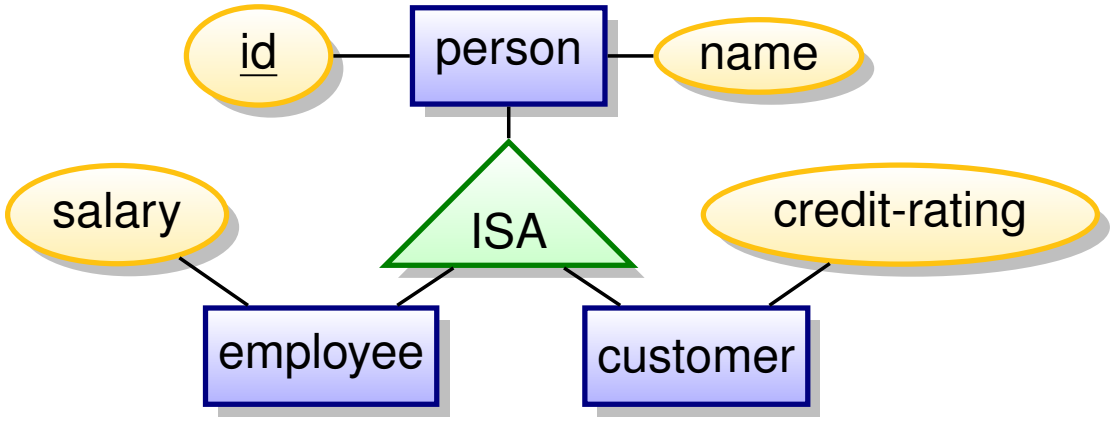


Method 3: one table with null values

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 non-applicable attributes

For instance, salary will be null for customers.

Translation :: Primary Keys

Primary Keys

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

What would be a good primary key?

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

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

Primary Keys

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

What would be a good primary key?

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

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

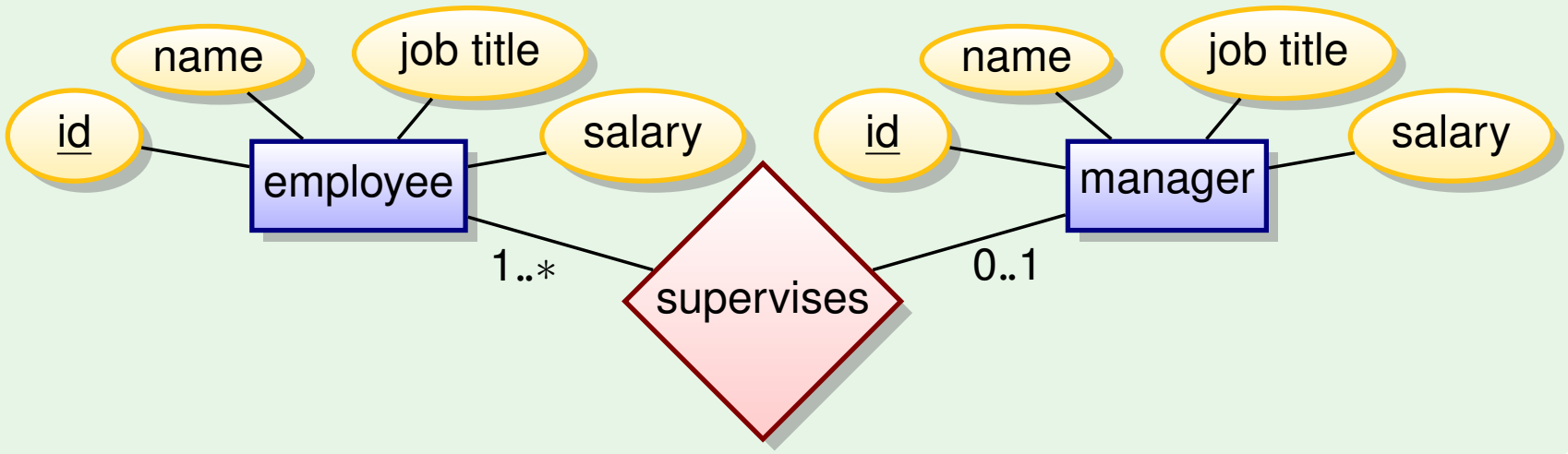
It is often good to introduce an artificial **internal key**:

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

Translation :: Recursive Relations

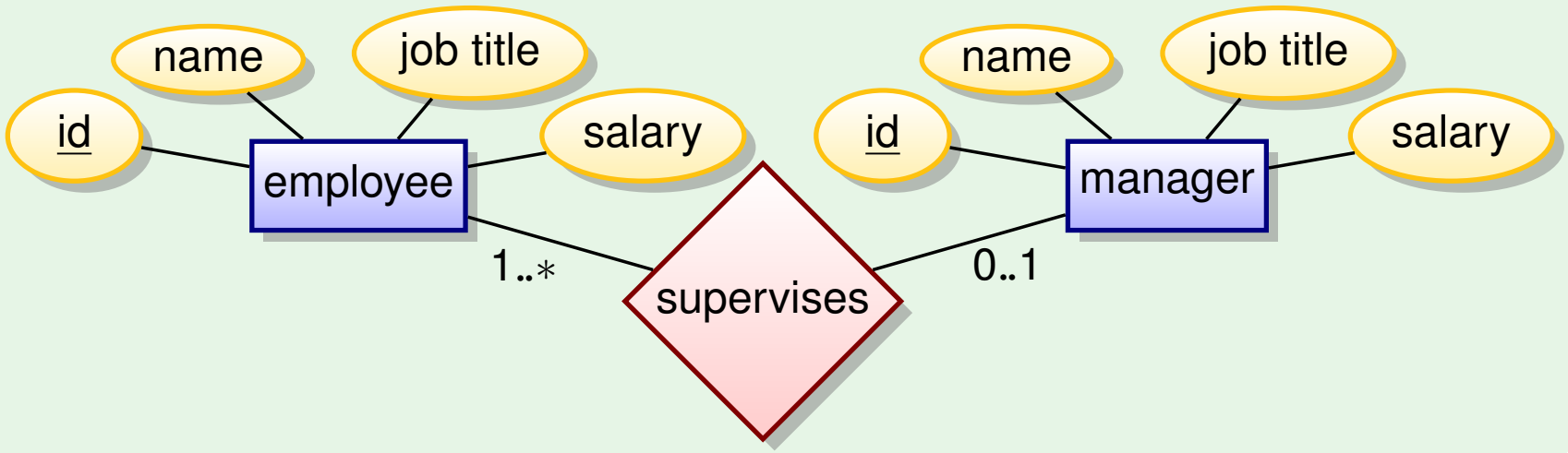
Recursive Relations

Example: an employee is supervised by a manager.



Recursive Relations

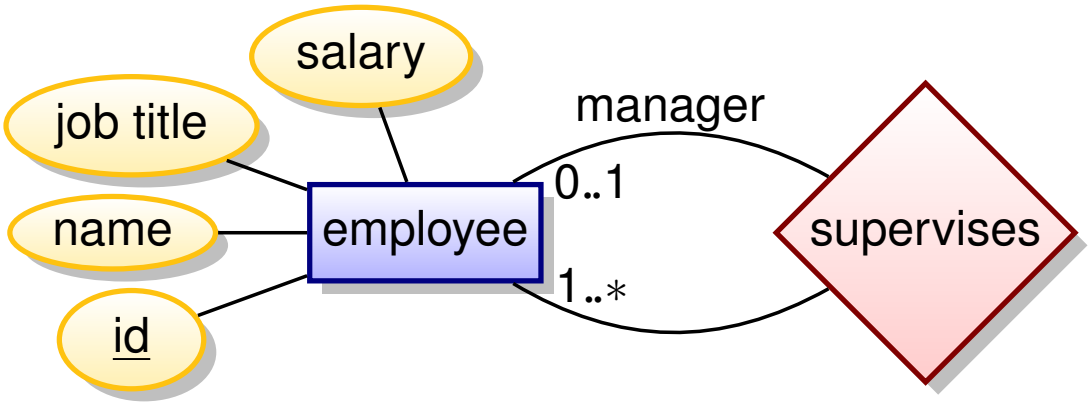
Example: an employee is supervised by a manager.



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

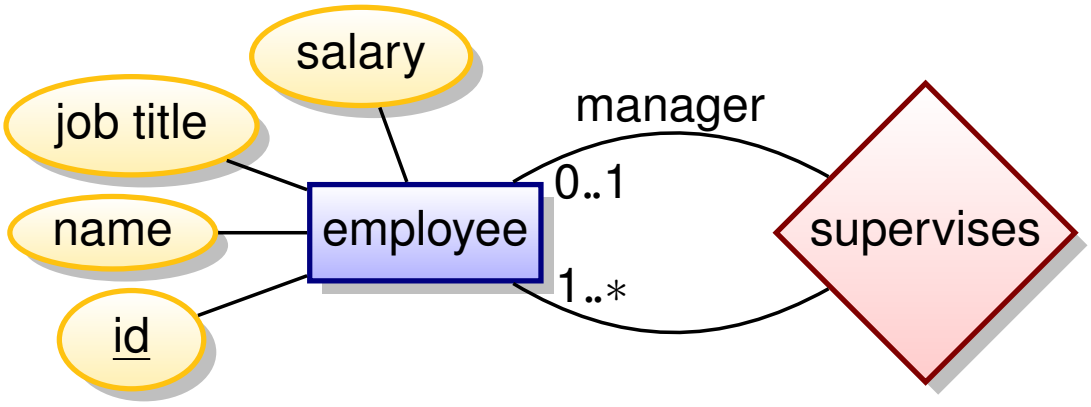
Recursive Relations

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



Recursive Relations

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

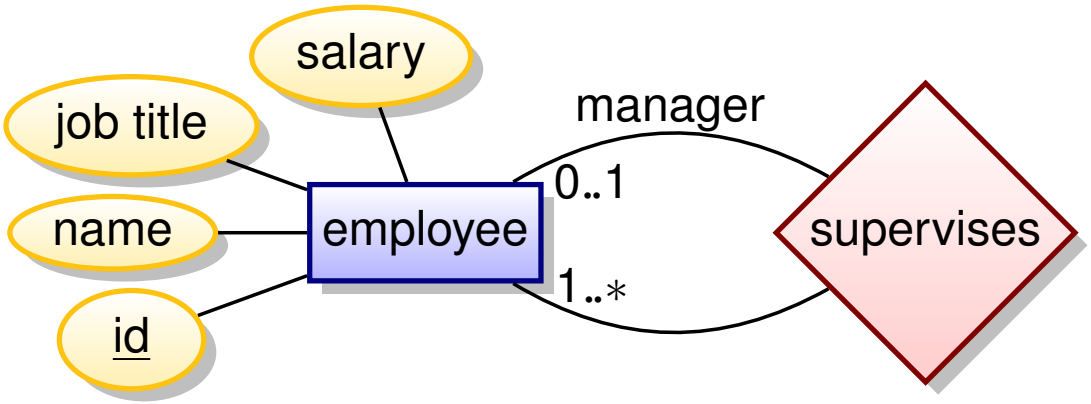


A **recursive relation** translates to a foreign key that refers to the same table.

Employee				
id	**name**	**jobTitle**	**salary**	**supervisedBy** $\rightarrow$ **id**
1	James	...	...	2
2	Harrison	...	...	null

Recursive Relations

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



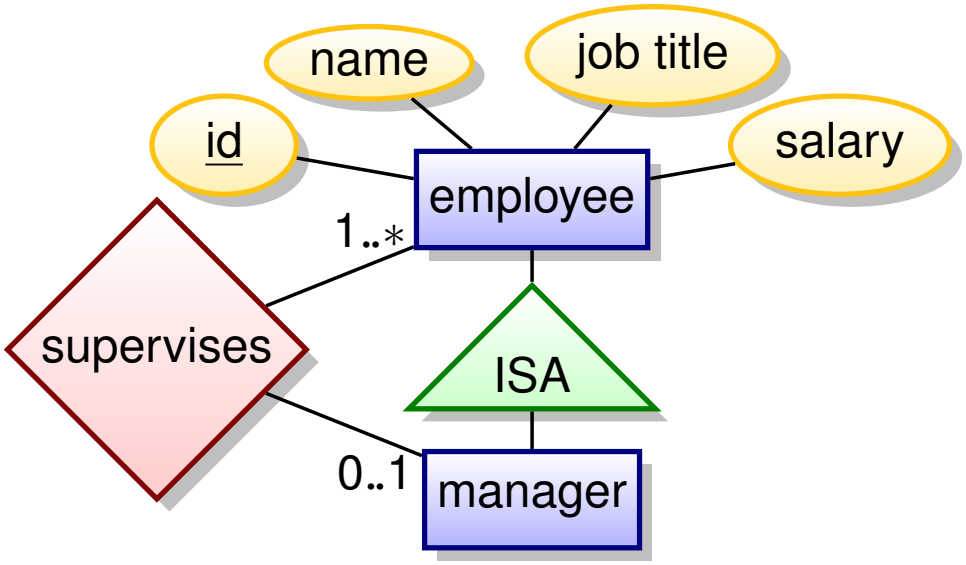
A **recursive relation** translates to a foreign key that refers to the same table.

Employee				
id	**name**	**jobTitle**	**salary**	**supervisedBy** $\rightarrow$ **id**
1	James	...	...	2
2	Harrison	...	...	null

A **recursive many-to-many relation** requires a separate table with two foreign keys to the parent table (the usual translation).

Recursive Relations

The following diagram is also correct:



Can be translated as:

Employee				
id	**name**	**jobTitle**	**salary**	**supervisedBy** $\rightarrow$ **Manager**

Manager
id $\rightarrow$ **Employee**

If the manager has no additional attributes, then it is better to eliminate the table (translation as on the last slide).