Part 5: Introduction to Logical Design

References:

- Elmasri/Navathe:Fundamentals of Database Systems, 3rd Edition, 1999. Chapter 3, "Data Modeling Using the Entity-Relationship Model"
- Silberschatz/Korth/Sudarshan: Database System Concepts, 3rd Ed., Ch. 2, "Entity-Relationship Model".
- Ramakrishnan: Database Management Systems, Mc-Graw Hill, 1998, Ch. 14, "Conceptual Design and the ER-Model"
- Kemper/Eickler: Datenbanksysteme (in German), Ch. 2, Oldenbourg, 1997.
- Rauh/Stickel: Konzeptuelle Datenmodellierung (in German), Teubner, 1997.
- Teorey: Database Modeling and Design, Third Edition, 1999.
- Barker: CASE*Method, Entity Relationship Modelling, Oracle/Addison-Wesley, 1990.
- Lipeck: Skript zur Vorlesung Datenbanksysteme (in German), Univ. Hannover, 1996.

Objectives

After completing this chapter, you should be able to:

- translate a given Entity-Relationship-diagram into the relational model.

  I.e. determine an equivalent relational database schema (including keys and foreign keys).

- explain what constructs (cardinalities) cannot be directly translated.
- recognize typical structures (such as many-to-many relationships) in relational database schemas.

Overview

General Remarks (1)

- In order to develop a relational schema, one usually first designs an ER-schema, and then transforms it into the relational model, because the ER-model
  - ◇ allows better documentation of the relationship between the schema and the real world.

    E.g. entity types and relationships are distinguished.
  - ◇ has a useful graphical notation.
  - ◇ has constructs like inheritance which have no direct counterpart in the relational model.

    The difficult conceptual design can be simplified a bit by first using the extended possibilities.

General Remarks (2)

- Given an ER-schema S_E, the goal is to construct a relational schema S_R such that there is a one-to-one mapping τ between the states for S_E and S_R.

  I.e. each possible DB state with respect to S_E has exactly one counterpart state with respect to S_R and vice versa.

- States that are possible in the relational schema but meaningless with respect to the ER-schema must be excluded by integrity constraints.

  E.g., in the ER-model, relationships can be always only between currently existing entities. In the relational model, "dangling pointers" must be explicitly excluded by means of foreign key constraints.

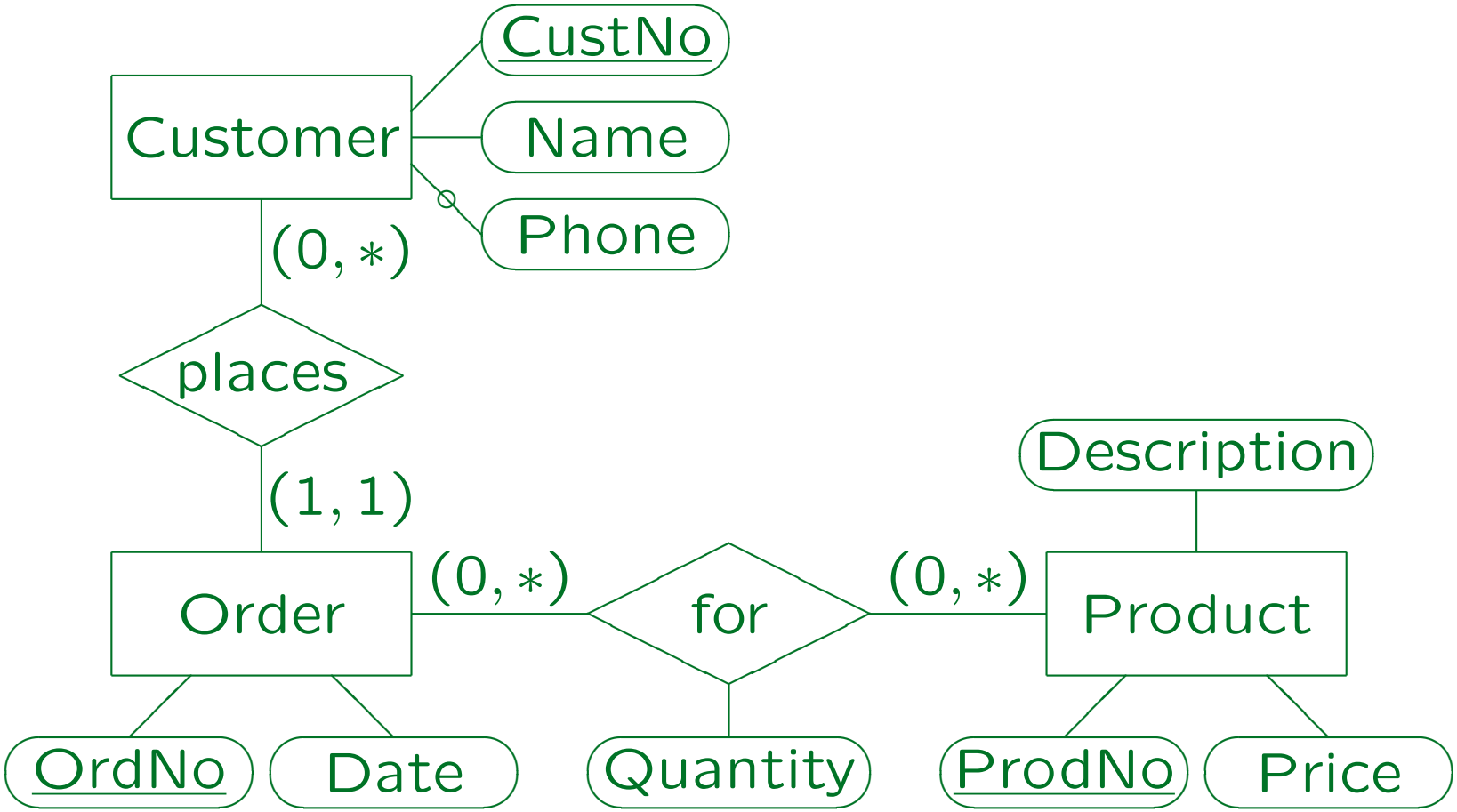
General Remarks (3)

- In addition, it must be possible to translate queries referring to S_E into queries with respect to S_R, evaluate them in the relational system, and then translate the answers back.
- I.e. it must be possible to simulate the designed ER-database with the actually implemented relational database.

  Any schema translation must explain the correspondance of schema elements such that, in our case, a query intended for the ER-schema can also be formulated with respect to the relational schema.

Overview

Example


CustNo
Customer
Name
Phone
(0, ∗)
places
(1, 1)
Description
Order
(0, ∗)
for
(0, ∗)
Product
OrdNo
Date
Quantity
ProdNo
Price

Step 1: Entities (1)

- First create a table for each entity. The name of this table is the name of the entity type.

  Alternatively, the plural form can be used.

- The columns of this table are the attributes of the entity type.

  Optional attributes translate into columns allowing null values.

- The primary key of the table is the primary key of the entity type.

  If the entity type has no key, add an artificial key.

Step 1: Entities (2)

Customers		
CustNo	Name	Phone
10	Jones	624-9404
11	Smith	

Orders	
OrdNo	Date
200	2/15/00
201	2/16/00

Products		
ProdNo	Description	Price
1	Apple	0.50
2	Kiwi	0.25
3	Orange	0.60

Step 2: One-To-Many Rel. (1)

- If a relationship has the maximum cardinality 1 on one side, it is one-to-many. E.g. "places" is one-to-many from "Customer" to "Order".

  If it has maximum cardinality 1 on both sides, it is actually one-to-one (see below).

- In this case you add the key of the "one" side (Customer) as a column to the "many" side (Order).
- This column will be a foreign key referencing the row which corresponds to the related entity.

Step 2: One-To-Many Rel. (2)

- Result in the example:

`Orders(OrdNo, Date, CustNo→Customers)`

Orders		
OrdNo	Date	CustNo
200	2/15/00	11
201	2/16/00	11

Customers		
CustNo	Name	Phone
10	Jones	624-9404
11	Smith	

Step 2: One-To-Many Rel. (3)

- If the minimum cardinality is 1 (as in this example), null values are not allowed for the new foreign key column.
- If the minimum cardinality should be 0, null values must be allowed for the foreign key column.

  It is null for entities not participating in the relationship.

Step 2: One-To-Many Rel. (4)

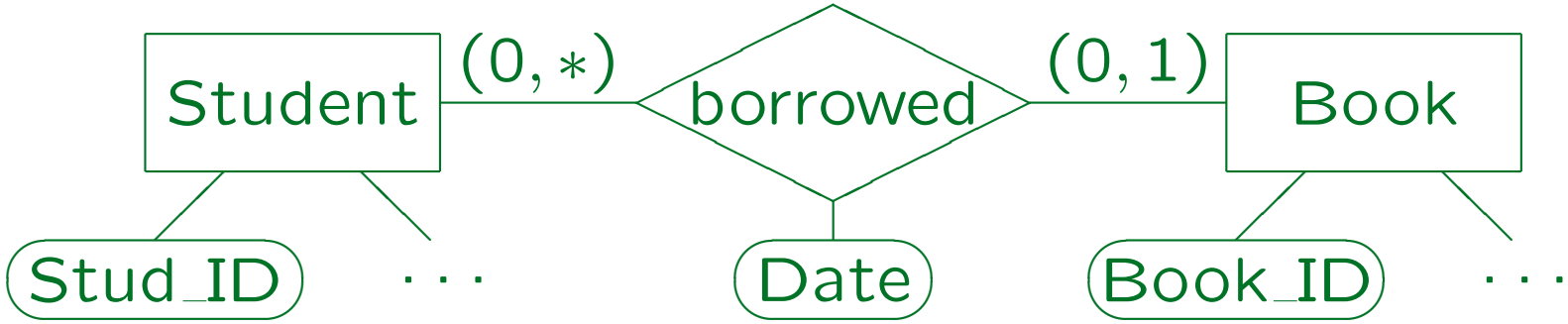
- The relationship name can be used in the column name, e.g.

  `Orders(OrdNo, Date, placed_by→Customers)`

- This excludes natural joins (connecting tables by equal column names, see Chapter 6), but this operation is not important in SQL (matter of style).
- Of course, all columns in the table must have unique names. If the added foreign key has the same name as an existing column, one or both must be renamed.

Step 2: Relationship Attributes

- Relationships can have attributes, e.g.:



- Such attributes are stored together with the pointer to the related entity:

  Books(Book_ID, ..., Stud_ID$^{O}\rightarrow$Students, Date^{O})

  "Stud_ID" and "Date" can be null, since not every book is borrowed, but they can either be both null or both not null ($\rightarrow$ CHECK-constraint).

Step 2: A Variant

- One-to-Many Relationships with cardinality $(0,1)$ can be translated into a table of their own:

  borrowed_by(Book_ID→Books, Stud_ID→Students, Date)

- The key values of the related entities are stored, plus attribute values of the relationship.
- The key attributes of the side with the $(0,1)$ cardinality become the key of this relation.

  Every book can be borrowed only once at the same time.

- This does not work with the cardinality (1,1).

Step 3: Many-To-Many R. (1)

- A relationship is many-to-many if it has the maximum cardinality $*$ on both sides (e.g. "for").
- Many-to-many relationships become their own tables.
- The columns of this table are the keys of the participating entity types, which together form the key of this table.
- These columns are at the same time foreign keys, referencing the tables for the entity types.

Step 3: Many-To-Many R. (2)

- Relationship attributes are added as columns. They are not part of the key, e.g.

  `for(OrdNo→Orders, ProdNo→Products, Quantity)`

- Note that the key of "for" must really consist of both, "OrdNo" and "ProdNo".

  Since one order can request multiple products, "OrdNo" alone cannot be key. Since the same product may be subject of multiple orders, also "ProdNo" alone does not suffice.

- Tables can be renamed. E.g. "`Order_Details`" is a better table name than "`for`".

Step 3: Many-To-Many R. (3)

for		
OrdNo	ProdNo	Quantity
200	1	1
200	2	1
201	1	5

Orders		
OrdNo	Date	CustNo
200	2/15/00	11
201	2/16/00	11

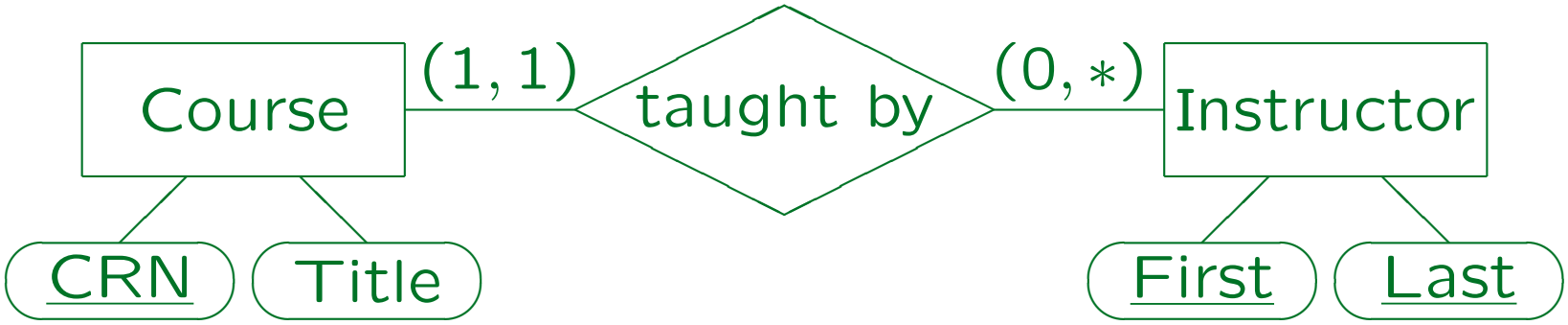
Products		
ProdNo	Description	Price
1	Apple	0.50
2	Kiwi	0.25
3	Orange	0.60

Step 3: Many-To-Many R. (4)

- Minimum cardinalities other than 0 for many-to-many relationships cannot be enforced by the standard constraints of the relational model.
- E.g. it would make sense to require that every order must be "for" at least one product. But this becomes a general constraint in the relational model.
- Of course, since this is important for the validity of the database state, one can specify the cardinality and later do checks in application programs.

  It only cannot be specified declaratively in the `CREATE TABLE`.

Composite Foreign Keys



- A composite foreign key is used to reference a table with a composite key:

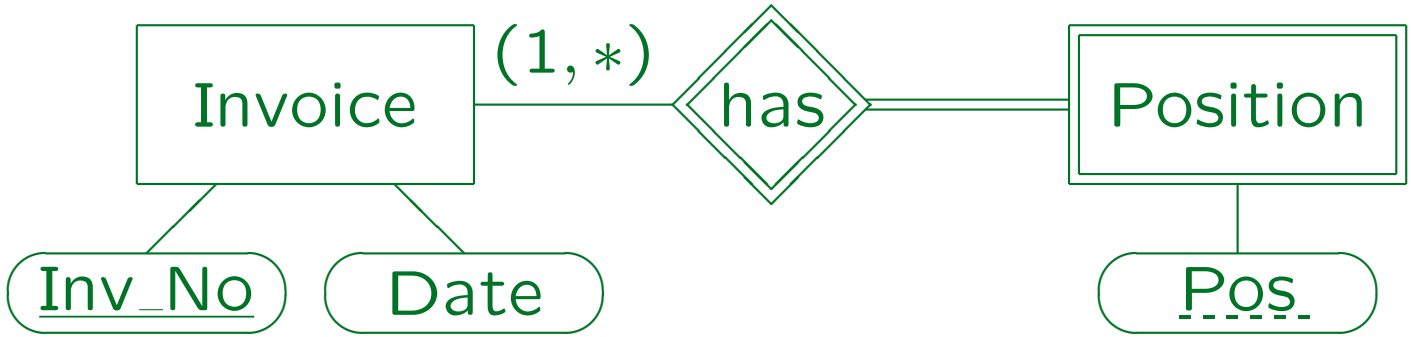
  Course(CRN, Title, (First, Last) → Instructor)

- If the minimum cardinality would be 0, "First" and "Last" can be null, but only together.

Overview

1. Goals of Logical Design
2. Basic ER-Constructs
3. Weak Entities
4. One-to-One Relationships
5. Final Steps, Limitations

Step 1B: Weak Entities (1)



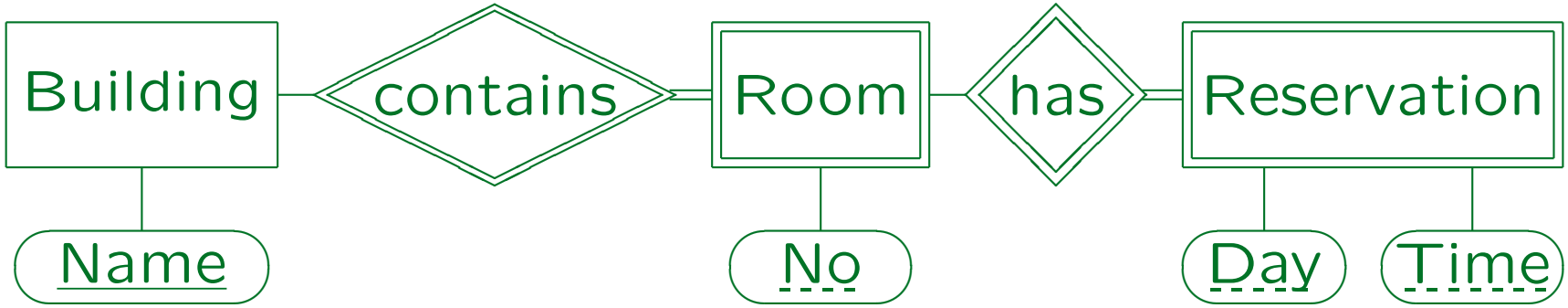
- When a weak entity is translated, the key attributes of the owner entity must be added as a foreign key:

  $$\text{Position}(\underline{\text{Inv\_No}} \rightarrow \text{Invoice}, \underline{\text{Pos}}, \ldots)$$

- This automatically implements the relationship.

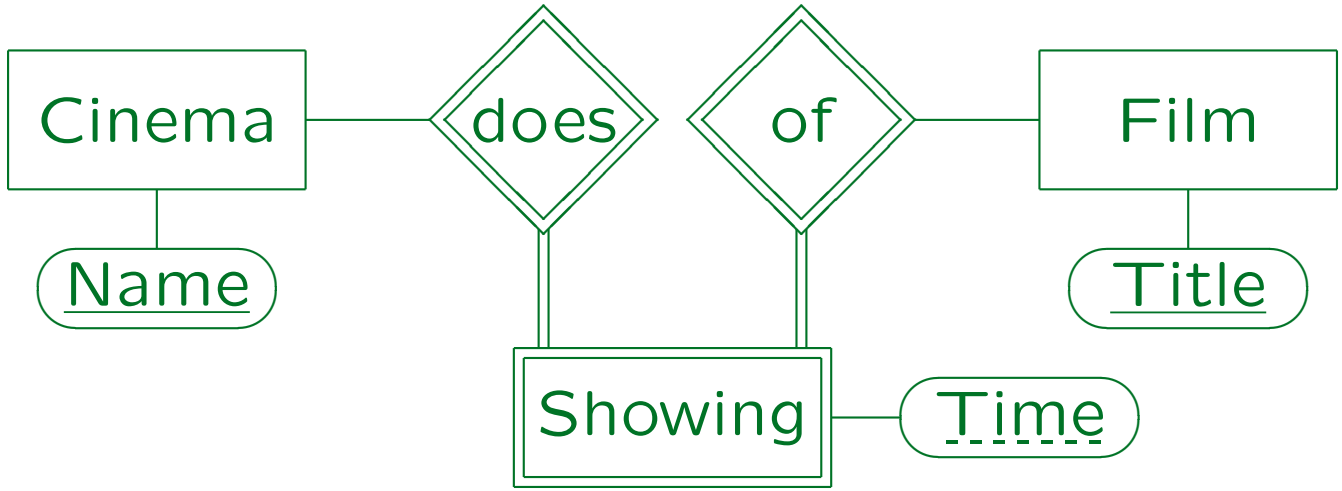
  Such relationships must be ignored in Step 2.
  It makes sense to specify "`DELETE CASCADES`" for the foreign key.
  Note that there is no "dashed underlining" in the relational model.

Step 1B: Weak Entities (2)



- If a weak entity is itself parent of another weak entity, the inherited key attributes are passed further:
  Buildings(Name)
  Rooms(Name → Buildings, No)
  Reservations((Name, No) → Rooms, Day, Time)
- Exercise: Does Reservations need to declare Name as a foreign key referencing Buildings?

Step 1B: Weak Entities (3)



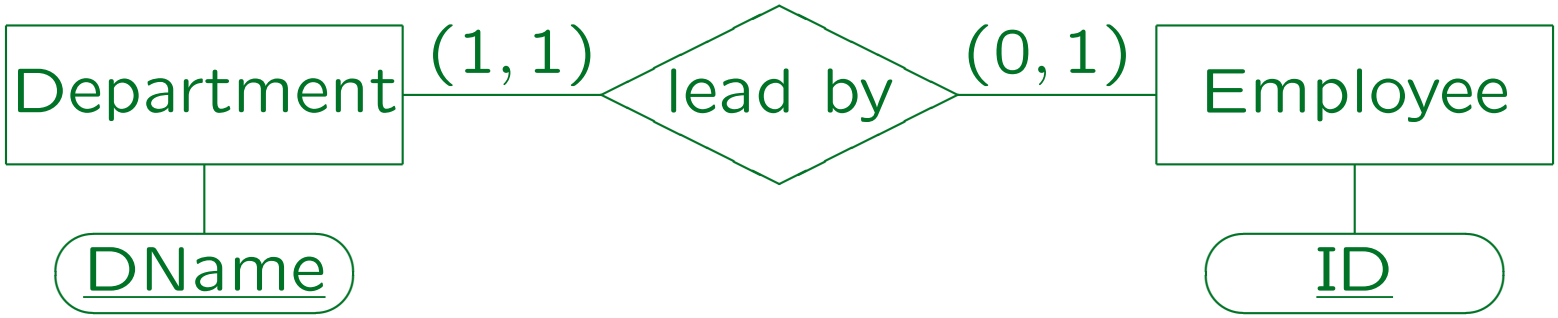
- Association entities inherit key attributes from more than one source:

  Cinemas(Name)
  Films(Title)
  Showings(Name → Cinemas, Title → Films, Time)

Overview

1. Goals of Logical Design
2. Basic ER-Constructs
3. Weak Entities
4. One-to-One Relationships
5. Final Steps, Limitations

Step 4: One-to-One Rel. (1)



- A relationship is one-to-one if it has maximum cardinality 1 on both sides.
- Basically, the translation is the same as for one-to-many relationships.

  However, there is an additional key constructed, see below.

Step 4: One-to-One Rel. (2)

- In this example, it is better to include the Employee key in the Department table, than vice versa, since Department has cardinality (1,1):

  Department(DName, . . . , Head $\rightarrow$ Employee)

- In this way, null values are avoided and the minimum cardinality 1 is enforced.

  If the department name is included in the Employee table, it can be null. In addition, a general constraint is required to ensure that every departments has a department head.

Step 4: One-to-One Rel. (3)

- "Head" is now also a key for the table "Department" (!), since an employee can be head of at most one department.
- It is only an alternative key, not part of the primary key.
- It enforces the maximum cardinality 1 on the Employee side.

Step 4: One-to-One Rel. (4)

- The key of any of the two tables can be included in the other table (as a possibly null foreign key).

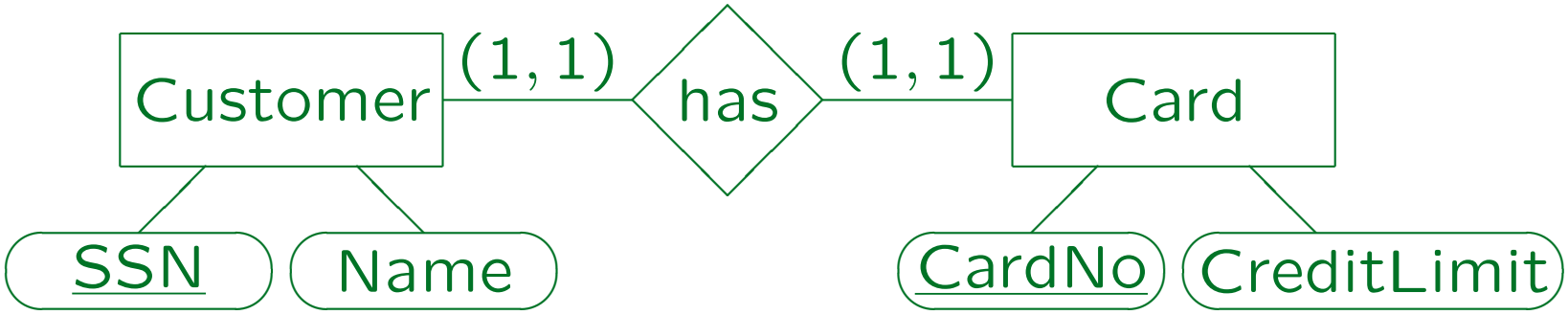
  However, it would be wrong to do both (redundancy).

- Or translate the relationship into a table on its own:

  Marriage(MName $\rightarrow$ Man, WName $\rightarrow$ Woman)

- Exercise: What is the key / keys?

Step 4: One-to-One Rel. (5)



- In order to enforce the minimum cardinality 1 on both sides, the tables must be merged:

  CustomerCard(SSN, Name, CardNo, CreditLimit)

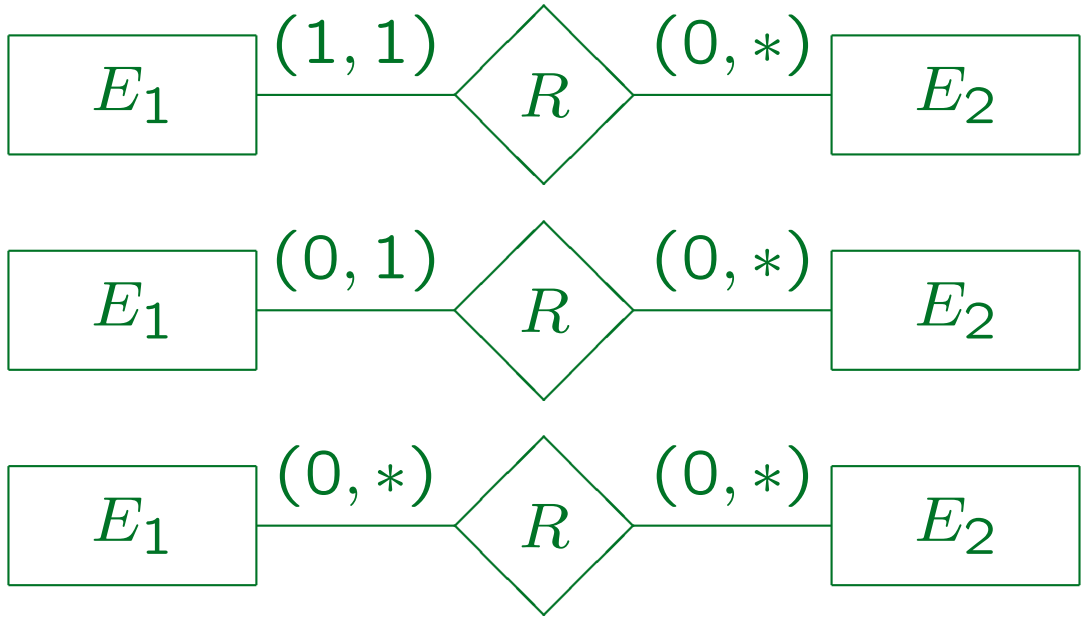
- SSN and CardNo are both keys. One is selected as primary key, the other is an alternative key.

Overview

1. Goals of Logical Design
2. Basic ER-Constructs
3. Weak Entities
4. One-to-One Relationships
5. Final Steps, Limitations

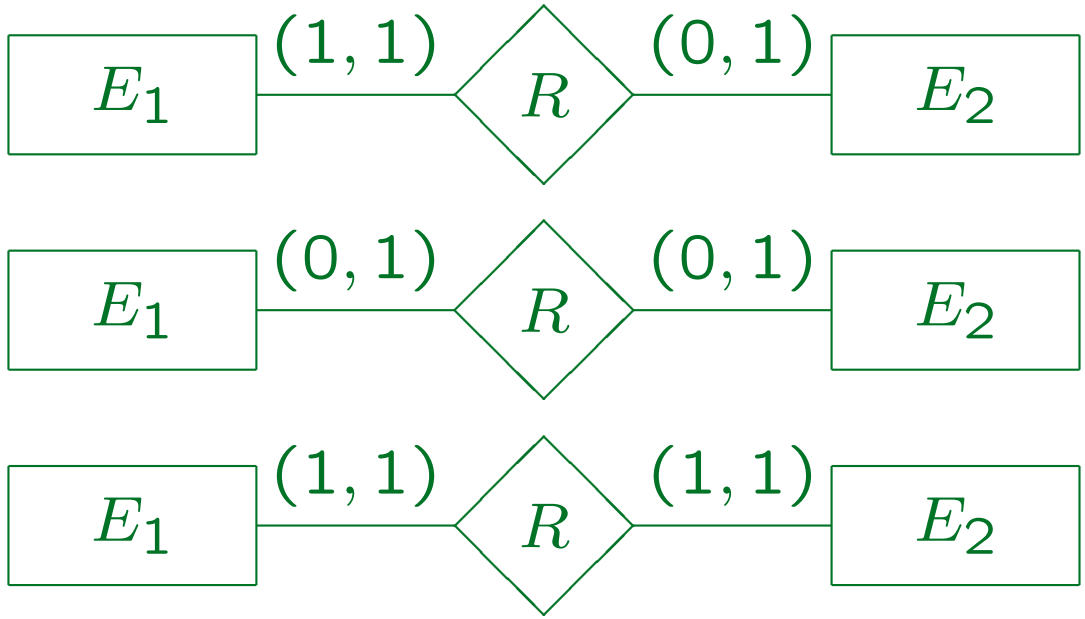
Limitations (1)

- The following cardinalities can be translated with the methods explained above (using only the standard constraints of the relational model):



Limitations (2)

- In addition, all kinds of one-to-one relationships can be handled:

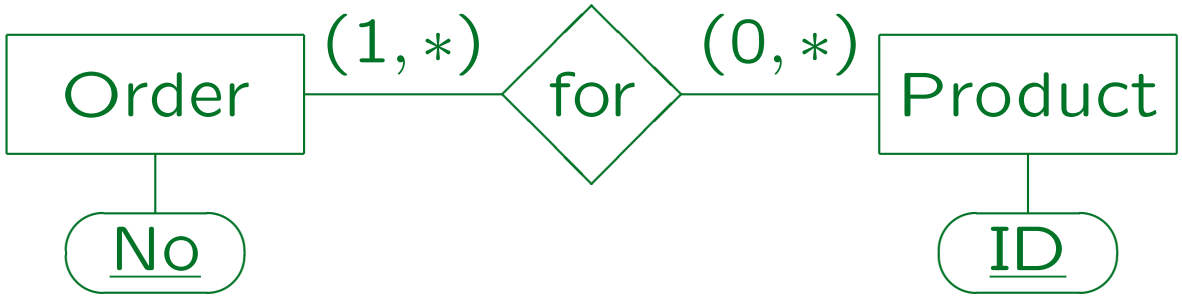


Limitations (3)

- If a relationship is none of these six cases, general constraints must be used, which can be implemented, e.g. via
  - ◇ Checks in application programs that are used to insert data.
  - ◇ Triggers, i.e. procedures stored in the database that are automatically executed e.g. for every tuple that is inserted or modified.
  - ◇ SQL queries that are executed from time to time and that print violations to the constraints.

Limitations (4)

- In particular, the cardinality $(1, *)$ sometimes makes sense, e.g. every purchase order should be for at least one item:



- The minimum cardinality 1 cannot be enforced declaratively in current DBMS.

  One uses the same translation as for $(0, *)$ and documents/specifies a general constraint in addition.

Step 5: Check (1)

- At the end, check the generated tables to see whether they make sense.
- E.g. fill them with a few example rows.
- If a correct ER-schema is correctly translated into the relational model, one will get a correct relational schema.
- However, a by-hand translation can result in mistakes, and the ER-schema can contain hidden flaws.

Step 5: Check (2)

- Sometimes tables are redundant and can be deleted.
- Think a last time about renaming tables or attributes.
- If two tables have the same key, consider merging them ("consider" does not mean to do it always!).
- Check the generated tables for a relational normal form (e.g. 3NF, BCNF, 4NF) (see Chapter 11).