The Entity-Relationship Model

- After completing this chapter, you should be able to

  ▷ explain the **three phases of database design,**

    Why are multiple phases useful?

  ▷ evaluate the significance of the **Entity-Relationship Model** (ER model) for DB design,

  ▷ enumerate the basic constructs of the ER model,

  ▷ develop **ER diagrams** (schemas in the ER model) for a given application,

  ▷ **translate** ER models into equivalent (as far as possible) relational models.

The Entity-Relationship Model

Overview

1. Database Design Overview

2. Basic ER Constructs

3. Kinds of Relationships (Cardinalities)

4. Keys, Weak Entities

5. Translation into the Relational Model

Database Design (1)

- Overall goal of DBMS usage: Efficiently develop programs to support given real-world tasks.

- These programs need to **store data persistently.**

- To develop these programs, apply proven methods of **software engineering**—specialized to support **data-intensive** programs.

> ### Database Design
> **Database Design** is the process of developing a **database schema** for a given application.
> DB design is a subtask of the overall software engineering effort.

Database Design (2)

- The specification of programs and data is intertwined:

  ▷ The schema should contain the data needed by the programs.

  ▷ Programs are often easy to develop once the structure of the data to be manipulated has been specified.

- Data, however, is an **independent resource:**

  ▷ Typically, additional programs will be developed later based on the collected data.

  ▷ Also, **ad-hoc queries** will be posed against the DB.

Database Design (3)

• During DB design, a **formal model of the relevant aspects
of the real world** ("mini world", "domain of discourse") must
be built.

  Once the DB is up and running, questions will be posed against
  the DB. Knowledge of these questions beforehand is important
  input to the DB design process and helps to identify the
  relevant parts of the real world.

• In some sense, the real world is the **measure of correctness**
for the DB schema: the possible DB states should correspond
to the states of the real world.

Database Design (4)

- DB design is not easy for a variety of reasons:

  ▷ **Expertise:** The designer must become an expert for the application domain or needs to talk to such experts. Depending on the domain, these experts might *not* be used to give a complete and formal account of their field.

  ▷ **Flexibility:** The real world typically permits exceptions and corner cases. Does the DB schema need to reflect these?

  ▷ **Size:** Database schemas may become huge (in terms of number of relations, attributes, constraints).

- Due to this complexity, DB design is a **multi-step** process.
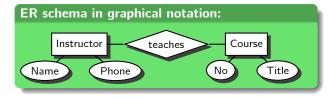
Database Design (5)

Three Phases of DB Design

① **Conceptual Database Design.** Produces the initial model of the mini world in a **conceptual data model** (*e.g.*, in the ER model).

② **Logical Database Design.** Transforms the conceptual schema into the data model supported by the DBMS (*e.g.*, the relational model).

③ **Physical Database Design.** Design indexes, table distribution, buffer size, *etc.*, to maximize performance of the final system (subject of "*Datenbanken II*").

Database Design (6)

- Why multiple design phases?

  ▷ Partition the problem, attack one sub-problem after the other.

    For example, during conceptual design there is no need to worry about performance aspects or limitations of the specific SQL dialect of the RDBMSs.

  ▷ DBMS features do not influence the conceptual design and only partially influence the logical design.

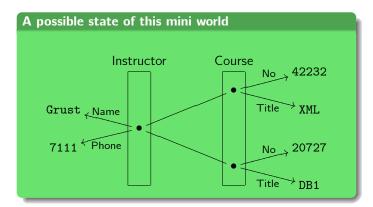   Thus, the conceptual design work is *not* invalidated, if a different DBMS is used later on.

Example (1)

- ER schema in graphical notation:

ER schema in graphical notation:

Instructor — teaches — Course

Name   Phone   No   Title

▷ This mini world talks about **instructors** and **courses.**

▷ Instructors **teach** courses.

▷ Instructors have a **name** and a **phone number.**

▷ Courses are **numbered** and have **titles.**

Example (2)

A possible state of this mini world

The ER Model (1)

- The **Entity-Relationship Model** is often referred to as a **semantic data model,** because it more closely resembles real world scenarios than, *e.g.*, the relational model.

  ▷ In the ER model, we model the concept of "Instructors." In the relational model we deal with names and phone numbers.

  ▷ In the ER model, there is a distinction between **entities** (objects) and **relationships** between such entities. In the relational model, both concepts are represented by relations.

The ER Model (2)

- Proposed by Peter Chen (1976), today at Louisiana State University (http://bit.csc.lsu.edu/~chen/chen.html).

- The ER model comes with a **graphical notation** which helps to establish quick overviews of database application schemas.

  Such **ER diagrams** are also a great way to communicate schemas to non-expert/future DB users.

- There are *no* "ER Model DBMS".

  Instead, we will describe a translation of ER model concepts into the relational model.

The Entity-Relationship Model

Overview

1. Database Design Overview

2. Basic ER Constructs

3. Kinds of Relationships (Cardinalities)

4. Keys, Weak Entities

5. Translation into the Relational Model

Basic ER Model Concepts (1)

- **Entities:**

  ▷ An **object** in the mini world about which information is to be stored. Examples: *persons, books, courses.*

  Note: entities do *not* have to correspond to objects of physical existence. Entities may also represent conceptual objects like, *e.g., vacations.*

  ▷ The mini world that has to be modelled can contain only a **finite number of objects.**

  ▷ It must be possible to distinguish entities from each other, *i.e.,* objects must have some **identity.**

  Examples: entity *book* identified by ISBN number, entity *vacations* identified by travel agency booking number.

Basic ER Model Concepts (2)

- **Attribute:**

  ▷ A **property** or feature of an entity (or relationship, see below).

    Example: the *title* of this *course* entity is "Foundations of Databases."

  ▷ The **value** of an attribute is an element of a data type like string, integer, date.

    These values have a **printable representation** (which entities have not).

Basic ER Model Concepts (3)

- **Relationship:**

  ▷ A **relation**—not in the strict relational model
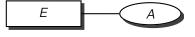   sense—**between pairs of entities** (a binary relationship).

    Example: Grust (an entity) *teaches* (a relationship) the
    course "Foundations of Databases" (an entity).

ER Diagrams (1)

- **Entity E:**

$$\boxed{E}$$

- **Attribute A of Entity Type E:**

$$\boxed{E} \!-\! \bigcirc\!A$$

- **Relationship R between Entity Types E_1 and E_2:**

$$\boxed{E_1} \!-\! \Diamond R \!-\! \boxed{E_2}$$

ER Diagrams (2)

- Relationships may also exist **entities of the same type** ("recursive relationship"):
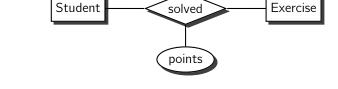

- In this case, **role names** have to be attached to the connecting edges.

  In the ER model, role names may be attached to any kind of relationship for documentation purposes.

ER Diagrams (3)

- Relationships may have attributes, too:

- ▷ This models the fact that a number of *points* is stored for every pair of a student X and an exercise Y such that X submitted a solution for Y.

Graphical Syntax

① An ER diagram contains

- **boxes, diamonds, ovals,** plus **interconnecting lines.**

② Boxes, diamonds, and ovals are each **labelled** by a string.

- Box labels are unique in the entire diagram.
- Oval labels are unique for a single box or diamond.
- Diamond labels are unique for a pair of connected boxes.

③ Interconnecting lines are only allowed between

- box—diamond, box—oval, diamond—oval.

④ A diamond has exactly two connecting lines to boxes. There may be any number of connections to ovals.

⑤ An oval has exactly one connecting line.

ER Example

Modelling a Mini World: Define an ER Diagram

- Information about researchers in the database field is to be stored.
- For each researcher, his/her last name, first name, e-mail address, and homepage URI are relevant.
- Researchers are affiliated with universities and assume a certain position (*e.g.,* professor, lecturer).
- Relevant university information are the name, homepage URI, and country.
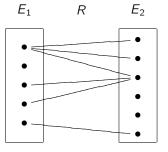- Researchers publish articles (of a given title) in journals.

The Entity-Relationship Model

Overview

1. Database Design Overview

2. Basic ER Constructs

3. Kinds of Relationships (Cardinalities)

4. Keys, Weak Entities

5. Translation into the Relational Model

Cardinalities (1)

- **General ER relationship:**

Cardinalities (2)

- In general, there is **no restriction** on how often an entity participates in an relationship R.

  ▷ An entity can be connected to one entity of the other type, to more than one, or it can have no R-partner at all.

- However, specific **application semantics** dictate to how many E_2 entities an E_1 entity can be related:

Cardinalities (3)

- The ER model introduces the **(min, max) notation** to specify an **interval** of possible participations in an relationship:

$$\boxed{E_1} \!-\! (m_1, n_1) \!-\! \langle R \rangle \!-\! (m_2, n_2) \!-\! \boxed{E_2}$$

- ▷ An entity of type E_1 may be related to at least m_1 and at most n_1 entities of type E_2.
- ▷ Likewise, m_2 is the minimum number and n_2 is the maximum number of E_1 entities to which an E_2 entity is related

Cardinalities (4)

- **Extensions:**
  - ▷ "$*$" may be used as maximum if there is **no limit.**
  - ▷ $(0, *)$ means no restriction at all (general relationship).

Cardinalities (5)

Marriage

| Man | — (0,1) — | is married to | — (0,1) — | Woman |

"*A man can be married to at most one woman and vice versa.*"

Airport Locations

| Airport | — (1,1) — | lies in | — (0,*) — | Country |

"*An airport lies in exactly one country. A country may have arbitrarily many airports (and maybe none at all).*"

Derive cardinalities from verbal specifications

"*Besides normal customers, the database may contain customers who have not yet ordered anything.*"

Order — (,) — ‹from› — (,) — Customer

Derive cardinalities from verbal specifications

"*An order can contain several products.*"

Order — (,) — ‹for› — (,) — Product

Selecting Cardinalities

- Sometimes a sketch of a valid database state may help in selecting the right cardinalities, *e.g.*, for the state sketched on slide 243, a viable cardinality for E_1—R may be $(0, 3)$.

- Application knowledge might lead to **weaker restrictions**, in this example $(0, 5)$ or $(0, *)$.

  A cardinality (a, b) is **weaker** than (c, d) if $a \leqslant c$ and $d \leqslant b$.

- In real applications, the cardinalities $(0, 1)$, $(1, 1)$, and $(0, *)$ are the most common and especially **easy to enforce in the relational model.**

Common Cases (1)

- Normally, the **minimum cardinality** will be 0 or 1, and the **maximum cardinality** will be 1 or $*$.

  ▷ Thus, only the $(0, 1), (1, 1), (0, *), (1, *)$ cardinalities are common in practice.

- To understand a relationship, one must know the cardinality specifications on **both sides.**

- The **maximum cardinalities** on each side are used to distinguish between **many-to-many**, **one-to-many** / **many-to-one**, and **one-to-one** relationships.
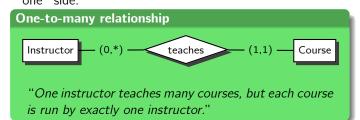
Common Cases (2)

- **Many-to-many** relationships:

  ▷ Both maximum cardinalities are $*$ (the minimum cardinalities are 0 or 1):

  > **Many-to-many relationship**
  >
  > Student — (0,*) — ◇ takes ◇ — (0,*) — Course

  ▷ This is the most general/least restrictive case of a relationship.
  ▷ When translated into the relational model, the representation of many-to-many relationships requires an extra table.

Common Cases (3)

- **One-to-many** relationships:

  ▷ Maximum cardinality 1 on the "many" side and ∗ on the "one" side:

  ### One-to-many relationship

  Instructor — (0,∗) — ⬦ teaches ⬦ — (1,1) — Course

  "*One instructor teaches many courses, but each course is run by exactly one instructor.*"

  ▷ One-to-many relationships do *not* require an extra table in an equivalent representation in the relational model.

Common Cases (4)

- **One-to-one** relationships:

  ▷ Maximum cardinality 1 on both sides:

  One-to-one relationship

  Employee — (0,1) — ⟨ is head of ⟩ — (1,1) — Department

  "*Each department has exactly one department head, some employees are the head of one department.*"

- Note how **mandatory** or **optional** participation in a relationship determines the **minimum cardinalities.**

Cardinalities: Alternative Notations (1)

- Widespread variants of notations for cardinalities:

  ▷ Leave particpiation unspecified:
    Cardinalities are either **many-to-many** (N:M),
    one-to-many (1:N), or **one-to-one** (1:1).

  One-to-many relationship

  Instructor —— 1 —— ⟨ teaches ⟩ —— N —— Course

Cardinalities: Alternative Notations (2)

- Sometimes found in software supporting visual ER diagram development (*e.g.*, in Oracle Designer^{TM}):

One-to-many relationship

Instructor ──── teaches ─────────── < Course
                        taught by

▷ The "crow foot" indicates the "many" side.

▷ Dashed lines indicate optional participiation.

▷ Relationship roles are given (for both sides).

The Entity-Relationship Model

Overview

1. Database Design Overview

2. Basic ER Constructs

3. Kinds of Relationships (Cardinalities)

4. Keys, Weak Entities

5. Translation into the Relational Model

Keys (1)

ER Key

A **key** K of an entity type E^a is an attribute of E which **uniquely identifies** the entities of this type. No two different entities share the same value for the key attribute. Composite keys are allowed.

---

[a]Only entity types can have key attributes.

Keys (2)

- The **translation of ER schemas into relational schemas** requires the declaration of ER keys.

- If there is no natural key, add **artificial identifiers** (*e.g.*, integers, remember attributes EMPNO, DEPTNO from Chapter 1) which then represent the entities.

Weak Entities (1)

- In many schemas, some entities describe a kind of **detail that cannot exist without a master** (or owner) **entity.**

  In such a case,

  ① there is a **relationship with cardinality** $(1, 1)$ **on the detail entity side,** and in addition

  ② the **key of the master is inherited and becomes part of the key of the detail entity.**

Weak Entities (2)

- Without a specific ER construct for this case, we would require the following additional constraint:

  ▷ If two entities are in "has" relationship,

  ▷ then their attribute "Inv_No" are required to have identical values.

  For example, invoice #12 cannot have position 2 in invoice #42 as detail.

- Such constraints occur if **an entity does not have a key by itself, but it is only unique in the context of some other (master) entity.**
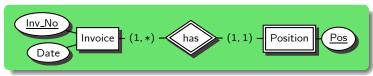
Weak Entities (3)

- **Note:** In such cases, keys are **always composite.**

  ▷ Examples:
    - A classroom is identified by a building *and* a room number.
    - A section in a book is identified by a chapter *and* a section title.
    - A web page URI is composed of a web server DNS address *and* a path on that server.

- There is also an **existence dependency.**

  If the building is pulled down, the classrooms automatically disappear. If the web server is shut down, all URIs on that server cease to function.

Weak Entities (4)

- In the ER model, such scenarios are modelled via **weak entities**[3].

- In ER diagrams, weak entities and their identifying relationships are indicated by **double lines:**

- For the weak entity, the **inherited part of the key is not shown.**

---

[3]Non-weak entities are also called strong entities.

Weak Entities (5)

Modelling with weak entities

Model a set of online quizzes (multiple choice tests).

- Each quiz is identified by a title, each question within a quiz is numbered, and each possible answer to a given question is referenced by a letter.

  For each question and answer, the associated text is stored. Answers are classified into correct and incorrect ones.
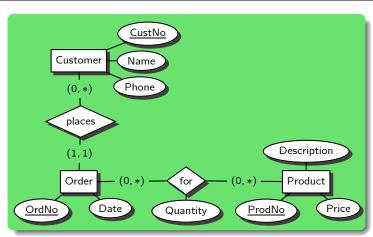
- What is the complete key for each of the occurring entity types?

The Entity-Relationship Model

Overview

1. Database Design Overview

2. Basic ER Constructs

3. Kinds of Relationships (Cardinalities)

4. Keys, Weak Entities

5. Translation into the Relational Model

ER Diagram Example

Step 1: Entities (1)

- **Transforming an ER entity E:**

  (1) Create a **table for each entity.**
  The table name is E (conventionally: $E + $'s').

  (2) The **columns** of this table are the **attributes of the entity type.**

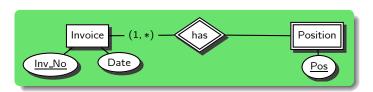 (3) The **primary key of the table** is the **primary key of the entity type.**

  If E's key is composite, so will be the relational key. If E has no key, add an **artifical key** to the table.

Step 1: Entities (2)

Customers		
CustNo	Name	Phone
10	Jones	624-9404
11	Smith	

Orders	
OrdNo	Date
200	2/15/04
201	2/16/04

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

Step 1B: Weak Entities

- When a **weak entity** is translated, the **key attributes of the owner entity** are added as a **key and foreign key:**

```
Position (Pos, Inv_No → Invoice, ...)
```

- ▷ This automatically implements the relationship.
- ▷ It makes sense to specify DELETE CASCADES for the foreign key: if an invoice is deleted, all its positions will be removed from the DB state, too.

Step 2: One-To-Many Relationships (1)

- **Transforming a relationship** R:

  (1) If R has **maximum cardinality 1 on one side**, R is **one-to-many**[4].

    Example: Customer–$(0, *)$–places–$(1, 1)$–Order, "one customer places many orders."

  (2) In this case, add the **key of the "one" side as a column to the "many" table** to implement R.

  (3) This column will be a **foreign key referencing a row in the table representing the related entity.**

---

[4]If R has maximum cardinality 1 on both sides, it is actually one-to-one, see below.

Step 2: One-To-Many Relationships (2)

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

Orders		
OrdNo	Date	CustNo
200	2/15/04	11
201	2/16/04	11

Customers		
CustNo	Name	Phone
10	Jones	624-9404
11	Smith	

- Convention: use relationship and role to name foreign key column:

```
Orders (OrdNo, Date, placed_by → Customers)
```
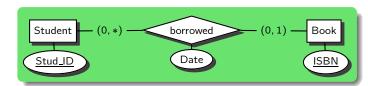
Step 2: One-To-Many Relationships (3)

- **Transforming a one-to-many relationship** R:

  ④ **If the minimum cardinality is 1** on the "many" side (see example), **null values are not allowed in the foreign key column** (column placed_by in example).

    If the minimum cardinality is 0, null values are allowed in the foreign key column.

    The foreign key is null for those entities that do not participate in R at all.

Step 2: Relationship Attributes

- To **transform one-to-many relationship attribute(s)**, *e.g.*

| Student — (0, ∗) — ⟨ borrowed ⟩ — (0, 1) — Book |
| Stud_ID — Date — ISBN |

▷ **store the relationship attribute(s) together with the reference to the related entity**, *e.g.*

```
Books (ISBN, ..., borrowed_by → Students, Date)
```

Step 2: One-To-Many Relationships: A Variant

- One-to-many relationships R **with cardinality (0,1)** can be translated into a table of their own:

```
borrowed_by (ISBN→Books, Stud_ID→Students, Date)
```

- The extra table holds the key values of the related entities plus the relationship attributes.
- The key attributes of the side with the (0,1) cardinality become the key of this relation.

  "Each book can be borrowed only once at the same time."

This does not model R correctly if the cardinality is (1,1)

Why?

Step 3: Many-To-Many Relationships (1)

- **Transforming a many-to-many relationship R:**

  (1) If R has **maximum cardinality** $*$ **on both sides,** R **is many-to-many.**

     Example: Order–$(1, *)$–for–$(0, *)$–Product, "an order contains many products, a product may be part of many orders."

  (2) R becomes its **own table.**

  (3) The columns of this table are the **keys of both participating entity types.**

  (4) These columns act as **foreign keys** referencing the entities and, at the same time, **together form a composite key for the extra table.**

Step 3: Many-To-Many Relationships (2)

- **Transforming a many-to-many-relationship R:**

  ⑤ Relationship attributes are added as columns to the table representing R.

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

Composite key?

Is it really necessary that both entity keys (here: OrdNo, ProdNo) form a composite key for the relationship table?

Step 3: Many-To-Many Relationships (3)

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

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

Orders		
OrdNo	Date	CustNo
200	2/15/04	11
201	2/16/04	11

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

Step 3: Many-To-Many Relationships (4)

- **Transforming a many-to-many relationship** R:

  ▷ **Note:** Minimum cardinalties other than 0 for R **cannot be enforced** by the relational model per se, *i.e.*, in terms of key constraints.

    Order–$(0, *)$–for–$(1, *)$–Product $\Leftrightarrow$ *"Every product occurs in at least one order."*

  ▷ If this is important to guarantee database consistency (valid DB state), this constraint needs to be **checked by the application program** or a **general RDBMS constraint mechanism.**
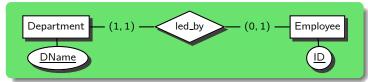
Step 4: One-To-One Relationships (1)

- **Transforming a one-to-one relationship** R:

Department — (1, 1) — led_by — (0, 1) — Employee

DName

ID

1. If R has **maximum cardinality 1 on both sides**, R **is one-to-one.**

2. We can essentially transform as if R were one-to-many, but additional key constraints are generated.

Step 4: One-To-One Relationships (2)

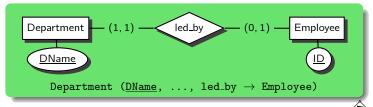- To which entity table shall we add the `led_by` attribute to represent the relationship?

Department — $(1,1)$ — led_by — $(0,1)$ — Employee

DName — ID

▷ Since we have Department–$(1,1)$–led_by ("*every department is led by exactly one employee*"), it makes sense to host the relationship in the Department table:

```
Department (DName, ..., led_by → Employee)
```

▷ We may declare the foreign key `led_by` as NOT NULL. (This is not possible if `led_by` is hosted in the Employee table.)

Step 4: One-To-One Relationships (3)

Department — $(1,1)$ — led_by — $(0,1)$ — Employee

DName

ID

```
Department (DName, ..., led_by → Employee)
```

- **Note:** led_by now also is a key for the Department table.

  led_by is a key for table Department

  Why is this the case in this example?

- This key constraint enforces the maximum cardinality of 1 (on the Employee side).
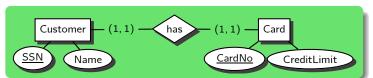
Step 4: One-To-One Relationships (4)

- Two variants:

  ① Any of the two (*not both!*) entity tables may host the `married` foreign key (null values allowed).

  ② Translate the relationship into a table of its own:

  ```
  married (MName → Man, WName → Woman)
  ```

  A one-to-one relationship in an extra table

  What would be the correct key(s) for table `married`?

Step 4: One-To-One Relationships (5)

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

```
CustomerCard (SSN, Name, CardNo, CreditLimit)
```

  ▷ No null values are allowed.
  ▷ Both, SSN and CardNo, are keys of this table. One is selected as primary key, the other is an secondary key.

Limitations (1)

• The following ER relationship cardinalities can be faithfully
  represented in the relational model:

Limitations (2)

- For all other cardinalities, the constraint mechanisms of the relational model will not suffice.


```
  Order  — (1, ∗) — ⟨ for ⟩ — (0, ∗) — Product
  ‾No‾                                    ‾ID‾
```

"Every purchase order includes at least one product item."

▷ Relational (foreign) key constraints *cannot* enforce the minimum cardinality 1.

⇒ Apply the translation method for (0, ∗) cardinalities and enforce the constraint in the application (or via an RDBMS **trigger** whenever a tuple in table Order is inserted).