2.2 Relational Model (RM)

- *Relational Model* by Codd (1970): mathematical foundation: set theory,
- only a single structural concept *Relation*,
- entity/object types and relationship types are uniformly modeled by *relation schemata*.
- properties of entities/objects and relationships are represented by attributes (in the relation schemata).
- a relation schema consists of a name and a set of attributes,
  Continent: name, area
- each attribute is associated with a *domain* that specifies the allowed values of the attribute. Often, attributes also can have a *null value*.
  Continent: name: VARCHAR(25), area: NUMBER
- A *(relational) database schema* \( R \) is given by a (finite) set of (relation) schemata.
  Continent: . . . ; Country: . . . ; City: . . . ; encompasses: . . . ; isMember: . .
- for every relation, a set of (primary) key attributes is distinguished

2.2.1 Relations

- A *(database) state* associates each *relation schema* to a *relation*.
- elements of a relation are called *tuples*.
  Every tuple represents an entity or a relationship. *(name: Asia, area: 4.5E7)*
- relations are unordered. Columns are also unordered.

**Example:**

<table>
<thead>
<tr>
<th>Continent</th>
<th>name</th>
<th>area</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VARCHAR(20)</td>
<td>NUMBER</td>
</tr>
<tr>
<td>Europe</td>
<td>9562489.6</td>
<td></td>
</tr>
<tr>
<td>Africa</td>
<td>3.02547e+07</td>
<td></td>
</tr>
<tr>
<td>Asia</td>
<td>4.50953e+07</td>
<td></td>
</tr>
<tr>
<td>America</td>
<td>3.9872e+07</td>
<td></td>
</tr>
<tr>
<td>Australia</td>
<td>8503474.56</td>
<td></td>
</tr>
</tbody>
</table>
Relations: Example

<table>
<thead>
<tr>
<th>Continent</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>area</td>
<td></td>
</tr>
<tr>
<td>Europe</td>
<td>9562489.6</td>
<td></td>
</tr>
<tr>
<td>Africa</td>
<td>3.02547e+07</td>
<td></td>
</tr>
<tr>
<td>Asia</td>
<td>4.50953e+07</td>
<td></td>
</tr>
<tr>
<td>America</td>
<td>8.9872e+07</td>
<td></td>
</tr>
<tr>
<td>Australia</td>
<td>8503474.56</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Country</th>
<th>code</th>
<th>population</th>
<th>area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Germany</td>
<td>D</td>
<td>83536115</td>
<td>356910</td>
</tr>
<tr>
<td>Sweden</td>
<td>S</td>
<td>8900954</td>
<td>449964</td>
</tr>
<tr>
<td>Russia</td>
<td>R</td>
<td>143666931</td>
<td>17075200</td>
</tr>
<tr>
<td>Poland</td>
<td>PL</td>
<td>38642565</td>
<td>312683</td>
</tr>
<tr>
<td>Bolivia</td>
<td>BOL</td>
<td>1098580</td>
<td>7165257</td>
</tr>
</tbody>
</table>

• with referential integrity constraints (to be explained later)
• references to keys

Graphical representation of the relational schema of the MONDIAL database (excerpt):
DEVELOPMENT OF A DATABASE APPLICATION

(cf. 3-Level-Architecture, Slide 6 and Slide 44)

Conceptual Design: structuring of the requirements for the representation of the relevant excerpt of the real world:
- independent from the database system to be used (phys. level),
- independent from the detailed views of the users (external schema),
results in the conceptual schema, in general an ER schema (or specified in UML).

Implementation Design: Mapping from the conceptual schema to the notions of the database system to be used.
The result is the logical schema, usually a relational schema (or an object-oriented schema, or – in earlier times – a network database schema).
- this mapping is described next,
- then realize it in the database (SQL) ...

DEVELOPMENT OF A DATABASE APPLICATION (CONT’D)

Physical Design: definition of the actual storage and appropriate auxiliary data structures (for enhanced efficiency).
- don’t worry: creating the logical schema in an SQL database automatically creates a structure on the physical level
  (this is the advantage of having the relational model as a kind of an abstract datatype that is implemented in a standardized way by relational databases).

Detailed Physical Design: optionally/later: finetuning of the physical level.

Implementation of the External Level:
- clarify the requirements on the external level by using the conceptual model, adapt to daily users’ needs (forms, presentations, reports, data exchange interfaces, ...),
- implement the external level based on the logical model.

Note:
“Classical” database design is restricted to the modeling of (static) structures, not considering the (dynamic) processes resulting from the execution (see UML).
Starting with the ER schema, the relational schema is designed.

Let $E_{ER}$ an entity type and $R_{ER}$ a relationship type in the ERM.

- **Entity types:** \((E_{ER}, \{A_1, \ldots, A_n\}) \rightarrow E(A_1, \ldots, A_n)\),
- For weak entity types, the key attributes of the identifying entity type must be added.
- **Relationship types:**
  \((R_{ER}, \{RO_1 : E_1, \ldots, RO_k : E_k\}, \{A_1, \ldots, A_m\}) \rightarrow B(E_1.K_{11}, \ldots, E_1.K_{1p_1}, \ldots, E_k.K_{k1}, \ldots, E_k.K_{kp_k}, A_1, \ldots, A_m)\),
  where \(\{K_{i1}, \ldots, K_{ip_i}\}\) are the primary keys of \(E_i, 1 \leq i \leq k\).
  - Renaming of foreign key attributes is allowed
    (e.g. coinciding attribute names in different referenced keys)

In case that \(k = 2\) and a \((1,1)\) relationship cardinality, the relation schema of the relationship type and that of the entity type may be merged.
- Aggregate types can be ignored if the underlying relationship type is mapped.

### Entity types

\((E_{ER}, \{A_1, \ldots, A_n\}) \rightarrow E(A_{i1}, \ldots, A_{ik})\)

where \(\{A_{i1}, \ldots, A_{ik}\} \subseteq \{A_1, \ldots, A_n\}\) are the scalar (i.e., not multivalued) attributes of \(E_{ER}\) – multivalued attributes are mapped separately.

<table>
<thead>
<tr>
<th>Continent</th>
<th>name</th>
<th>area</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VARCHAR(20)</td>
<td>NUMBER</td>
</tr>
<tr>
<td>Europe</td>
<td>9562489.6</td>
<td></td>
</tr>
<tr>
<td>Africa</td>
<td>3.02547e+07</td>
<td></td>
</tr>
<tr>
<td>Asia</td>
<td>4.50953e+07</td>
<td></td>
</tr>
<tr>
<td>America</td>
<td>3.9872e+07</td>
<td></td>
</tr>
<tr>
<td>Australia</td>
<td>8503474.56</td>
<td></td>
</tr>
</tbody>
</table>

The candidate keys of the relation are the candidate keys of the entity type.
MULTIVALUED ATTRIBUTES

... one thing left:
Attributes of relations must only be single values.
\( (E_{ER}, \{A_1, \ldots, A_i, \ldots, A_n\}) \) where \( A_i \) is a multivalued attribute
\( \rightarrow E_{A_i}(K_1, \ldots, K_p, A_i) \)
where \( \{K_1, \ldots, K_p\} \) are the primary keys of \( E \).
(repeating is allowed, especially if there is only one key attribute)
\( \{K_1, \ldots, K_p, A_i\} \) are the primary keys of the relation \( E_{A_i} \).

<table>
<thead>
<tr>
<th>Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>country</td>
</tr>
<tr>
<td>D</td>
</tr>
<tr>
<td>CH</td>
</tr>
<tr>
<td>CH</td>
</tr>
<tr>
<td>..</td>
</tr>
</tbody>
</table>

WEAK ENTITY TYPES

For weak entity types, the key attributes of the identifying entity type(s) must be added.
Relationship Types

\[(R_{ER}, \{RO_1 : E_1, \ldots, RO_k : E_k\}, \{A_1, \ldots, A_m\}) \rightarrow B(E_1_{K_11}, \ldots, E_1_{K_1p_1}, \ldots, E_k_{K_k1}, \ldots, E_k_{K_kp_k}, \ A_1, \ldots, A_m)\]

where \(\{K_{i_1}, \ldots, K_{i_p}\}\) are the primary keys of \(E_i, 1 \leq i \leq k\).

(it is allowed to rename, e.g., to use Country for Country.Code)

- Note: for references to weak entity types, the global key must be used (exercise: located_on as an \(n:m\) relationship between cities and islands).

Relationship Types: 1:N-Relationships

In case that \(k = 2\) (binary relationship) and a (0,1)- or (1,1)-relationship cardinality (i.e., \(n:1\)-relations), the relation schema of the relationship type and that of the entity type can be merged (into the relation schema for the entity type)

Other examples: headquarters of organizations, flows_into (the latter is a bit more complex because a river flows into another river, a lake, or a sea).
**RELATIONSHIP TYPES**

In case that for some relationship type, the keys of involved entity types have coinciding names, the role specifications may be used to guarantee the uniqueness of key attributes in the relationship type.

<table>
<thead>
<tr>
<th>borders</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>country1</td>
<td>country2</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>CH</td>
<td></td>
</tr>
<tr>
<td>CH</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>..</td>
<td>..</td>
<td></td>
</tr>
</tbody>
</table>

**EXERCISE**

Exercise 2.4

*Give a relational schema for the following ER schema:*

- Supplier
- Product
- Part
- delivers
- component
- contains
- nr
- name
- addr
- amount
- date
- color
- weight
2.4 Relational Databases – Formalization

Syntax

(note the similarities with first-order logic)

• A (relational) signature is a set of relation schemata \( R_i(\bar{X}_i) \).

• a relation schema \( R(\bar{X}) \) consists of a name (here, \( R \)) and a finite set \( \bar{X} = \{A_1, \ldots, A_m\} \), \( m \geq 1 \) of attributes. \( \bar{X} \) is the format of the schema.

• a (relational) database schema \( R \) consists of a relational signature (i.e., a set of (relation) schemata), optionally with integrity constraints.

• alternative notations for relation schemata:
  – abbreviation: \( R(A_1, \ldots, A_n) \) instead of \( R(\{A_1, \ldots, A_n\}) \).
  – if the order of the attributes \( \{A_1, \ldots, A_m\} \) is relevant (i.e., for representation as a table), \( \bar{X} \) is denoted as a vector \([A_1, \ldots, A_m]\).

Relational Databases – Formalization: Domains

Consider a relation schema \( R(\bar{X}) \)

• each attribute \( A \in \bar{X} \) is associated to a (non-empty) domain, called \( \text{dom}(A) \).

• \( \text{dom}(\bar{X}) := \text{dom}(A_1) \times \ldots \times \text{dom}(A_m) \).

• Example: Continent(name, area)
  \( \text{dom(continent.name)} = \text{VARCHAR}, \text{dom(continent.area)} = \text{NUMBER} \)

Note the following:

• the assignment of domains to attributes belongs to the database schema.

• in first-order logic, the definition of the domain of a structure belongs to the semantics.
• A (relational) database (or, more explicitly, a database state) $S$ (over $R = \{R_1(\bar{X}_1), \ldots, R_n(\bar{X}_n)\}$) is a relational structure over $R$.

• A relational structure $S$ associates each $R_i(\bar{X}_i)$ to a relation $S(R_i)$ over $\bar{X}_i$.

• elements of a relation are called tuples.
  (every tuple represents an entity or a relationship.)

• a tuple $\mu$ over $\bar{X}$ is a mapping $\mu : \bar{X} \rightarrow \text{dom}(\bar{X})$; or, for each individual attribute, $\mu : A \rightarrow \text{dom}(A)$.
  $\text{Tup}(\bar{X})$ denotes the set of all tuples over $\bar{X}$.

Example: Consider tuples in the Continent(name, area) table:

\[
\begin{array}{|l|l|}
\hline
\text{name} & \text{Asia} \\
\text{area} & 4.50953e+07 \\
\hline
\end{array}
\]

with $\mu(\text{name}) = \text{"Asia"}$, $\mu(\text{area}) = 4.5E7$

• a relation $r$ over $\bar{X}$ is a finite set $r \subseteq \text{Tup}(\bar{X})$ – usually represented by a table.

• $\text{Rel}(\bar{X}) := 2^{\text{Tup}(\bar{X})}$ is the set of all relations over $\bar{X}$.

---

**PERSPECTIVES: RELATIONAL VS. SET THEORY**

• Relations are sets of tuples.
  ⇒ relational algebra

**PERSPECTIVES: RELATIONAL VS. FIRST-ORDER LOGIC**

• database schema = relational signature = first-order signature without function symbols

• database = relational structure = first-order structure (without function symbols)
  (some authors use the term “interpretation” instead of “structure”)

Relational theory is based on “classical” logic results:
  ⇒ relational calculus

  • first-order logic
  • finite model theory
  • complexity results
  • (deductive databases)
While in the ER model, the keys serve only for an intuitive modeling, in relational database design they play an important role for the database performance and for the ability of the database to incorporate and maintain key constraints.

The notion of keys is defined as for the ER model:

For a set $\bar{K} \subseteq \bar{X}$ of attributes of a relation schema $R$, a relation $r \in \text{Rel}(\bar{X})$ satisfies the key constraint $\bar{K}$ if for all tuples $\mu_1, \mu_2 \in r$:

If $\mu_1(\bar{K}) = \mu_2(\bar{K})$ (i.e., $\mu_1$ and $\mu_2$ coincide on the values of $\bar{K}$), then $\mu_1 = \mu_2$.

More Concrete Requirements on Keys

(to be formalized on the next slides)

keys should be minimal:

- no subset $\bar{K}' \subsetneq \bar{K}$ satisfies the key property,
- for no subset $\bar{X}' \subsetneq \bar{X}$ of the attributes of $R$, any subset $\bar{K}' \subsetneq \bar{K}$ satisfies the key property wrt. $\bar{X}$. [Example see Slide 67]

The relational model provides a more concise formalization of keys (cf. Slide 326 ff. on Normalization Theory for details). These are based on the definition of functional dependencies:

Given a relation $R(\bar{X}), \bar{V}, \bar{W} \subseteq \bar{X}$. 

$r$ satisfies the functional dependency (FD) $\bar{V} \rightarrow \bar{W}$ if for all tuples $\mu_1, \mu_2 \in r$,

$$\mu_1(\bar{V}) = \mu_2(\bar{V}) \Rightarrow \mu_1(\bar{W}) = \mu_2(\bar{W}).$$

(“$\bar{W}$ functionally depends on $\bar{V}$”)

**Example 2.4**

Consider the relation schema Country(name, code, area, population, capital, capprov).

The following functional dependencies hold wrt. the intended application domain:

- $\{\text{code}\} \rightarrow \{\text{name}\}$
- $\{\text{name}\} \rightarrow \{\text{code}\}$
- $\{\text{code}\} \rightarrow \{\text{area, population, capital, capprov}\}$
- $\{\text{code}\} \rightarrow \{\text{name, code, area, population, capital, capprov}\}$
- $\{\text{name}\} \rightarrow \{\text{name, code, area, population, capital, capprov}\}$
Keys (Cont’d)

• In general, there are more than one key (called candidate keys) for a relation schema $R$.

• One of these candidate keys is distinguished (by the designer) to be the primary key. In the schema, it is represented by underlining these attributes.

Keys: Additional Formal Requirements

• Formalization of the Key Constraint:
  $K \subseteq \bar{X}$ is a possible key of $R(\bar{X})$ if $K \rightarrow \bar{X}$.

Additionally:

• keys must be minimal, i.e., no subset $K' \subsetneq K$ satisfies the key property:
  there is no subset $K' \subsetneq K$ s.t. $K' \rightarrow \bar{X}$.
  (otherwise: take $K'$ as key)

• every single attribute should be fully dependent on the complete key: for every
  $A \in (\bar{X} \setminus K)$: there is no subset $K' \subsetneq K$ s.t. $K' \rightarrow A$.
  (otherwise: if there is some attribute that depends only on a part of the key, split this relationship into a separate table, cf. example on Slide 67 and section on Normalization Theory, Slide 326.)

Although looking formally, the second criterion is also easy to understand and prevents bad/dangerous database design.
Keys and Database Design: Example

<table>
<thead>
<tr>
<th>Country (bad schema)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: Germany</td>
</tr>
<tr>
<td>Code: D</td>
</tr>
<tr>
<td>Language: German</td>
</tr>
<tr>
<td>Percent: 100</td>
</tr>
<tr>
<td>Population: 83536115</td>
</tr>
<tr>
<td>Area: 356910</td>
</tr>
<tr>
<td>Capital: Berlin</td>
</tr>
<tr>
<td>Province: Berlin</td>
</tr>
<tr>
<td>Name: Switzerland</td>
</tr>
<tr>
<td>Code: CH</td>
</tr>
<tr>
<td>Language: German</td>
</tr>
<tr>
<td>Percent: 65</td>
</tr>
<tr>
<td>Population: 7207060</td>
</tr>
<tr>
<td>Area: 41290</td>
</tr>
<tr>
<td>Capital: Bern</td>
</tr>
<tr>
<td>Province: BE</td>
</tr>
<tr>
<td>Name: Switzerland</td>
</tr>
<tr>
<td>Code: CH</td>
</tr>
<tr>
<td>Language: French</td>
</tr>
<tr>
<td>Percent: 18</td>
</tr>
<tr>
<td>Population: 7207060</td>
</tr>
<tr>
<td>Area: 41290</td>
</tr>
<tr>
<td>Capital: Bern</td>
</tr>
<tr>
<td>Province: BE</td>
</tr>
<tr>
<td>Name: Switzerland</td>
</tr>
<tr>
<td>Code: CH</td>
</tr>
<tr>
<td>Language: Italian</td>
</tr>
<tr>
<td>Percent: 12</td>
</tr>
<tr>
<td>Population: 7207060</td>
</tr>
<tr>
<td>Area: 41290</td>
</tr>
<tr>
<td>Capital: Bern</td>
</tr>
<tr>
<td>Province: BE</td>
</tr>
</tbody>
</table>

- the database is redundant
- needs more space, less efficient to query
- update anomalies/risks: updating Swiss population requires to update all three lines, otherwise inconsistent information

Dependency analysis:

Keys: \{Code, Language\} or \{Name, Language\}, but e.g. already \{Code\} \rightarrow \{Population, Capital\}

Split into Country(\Name, \Code, \Population, \Capital, \Province) and Languages(\Code, \Language, \Percent).

Keys and Database Design

- A good ER model and straightforward translation as introduced in the previous section leads to a good relational design
- determining the keys is helpful in validating the design:
  - for tables obtained from translating entity types, the keys are the same as in the ER model (for weak entity types: including those of the identifying entity types; cf. Country)
  - the handling of multivalued attributes as shown on Slide 53 is a consequence of the functional dependency analysis (same case as in the above example)
  - for relations that represent relationship types: see exercise below.

Exercise: Keys of relations obtained from relationships

Discuss how the keys of the relations that are obtained from relationship types are determined. Which alternative scenarios have to be considered?

- consider binary relationships systematically,
- what about ternary relationships?
Consider relation schemata \( R_1(\bar{X}_1) \) and \( R_2(\bar{X}_2) \). Let \( \bar{Y}_1 \subseteq \bar{X}_1 \) and \( \bar{Y}_2 \subseteq \bar{X}_2 \) two attribute vectors of the same length.

\[ r_1 = S(R_1) \text{ and } r_2 = S(R_2) \]

satisfy an inclusion constraint \( R_1.\bar{Y}_1 \subseteq R_2.\bar{Y}_2 \) if and only if for all \( \mu_1 \in r_1 \) there is a \( \mu_2 \in r_2 \) s.t. \( \mu_1(\bar{Y}_1) = \mu_2(\bar{Y}_2) \).

**Referential Integrity**

- if \( \bar{Y}_2 \) is the key of \( R_2 \), there is a referential integrity constraint from \( R_1.\bar{Y}_1 \) to \( R_2.\bar{Y}_2 \).
  - \( \bar{Y}_1 \) is called a foreign key in \( R_1 \) that references \( R_2.\bar{Y}_2 \).
- encompasses.continent \( \subseteq \) Continent.name
- encompasses.country \( \subseteq \) Country.code

Referential integrity constraints result from incorporating the keys of the participating entities into the table that represents the relationship.

---

**Null Values – Unknown Values**

- up to now, tuples are total functions.
- if for some attribute, there is no value, a null value can be used.

Semantics:
  - “value exists, but is unknown”
    (e.g., geo-coordinates of some cities)
  - “value does not yet exist, but will exist in the future”
    (e.g., inflation of a newly founded country)
  - “attribute not applicable” (e.g. “last eruption date” for mountains other than volcanoes)

- a partial tuple over \( \bar{X} \) is a mapping s.t.

\[
\text{for all } A \in \bar{X}, \mu(A) \in \text{dom}(A) \cup \{\text{null}\}.
\]

A relation is called partial if it contains partial tuples.
2.4.1 Exercise

Exercise 2.5
Consider the relation schema $R(\bar{X})$, where $\bar{X} = \{A, B\}$ and $\text{dom}(A) = \text{dom}(B) = \{1, 2\}$.

- Give $\text{Tup}(\bar{X})$ and $\text{Rel}(\bar{X})$.
- $A$ is a key of $R$. Which relations $r \in \text{Rel}(\bar{X})$ violate the key constraint?