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** $\mathbf{R}$ is given by a (finite) set of (relation) schemata.
  Continent: . . . ; Country: . . . ; City: . . . ; encompasses: . . .

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:

Continent	
<u>Name</u>	**Area**
VARCHAR(20)	NUMBER
Europe	9562489.6
Africa	3.02547e+07
Asia	4.50953e+07
America	3.9872e+07
Australia	8503474.56

Relations: Example

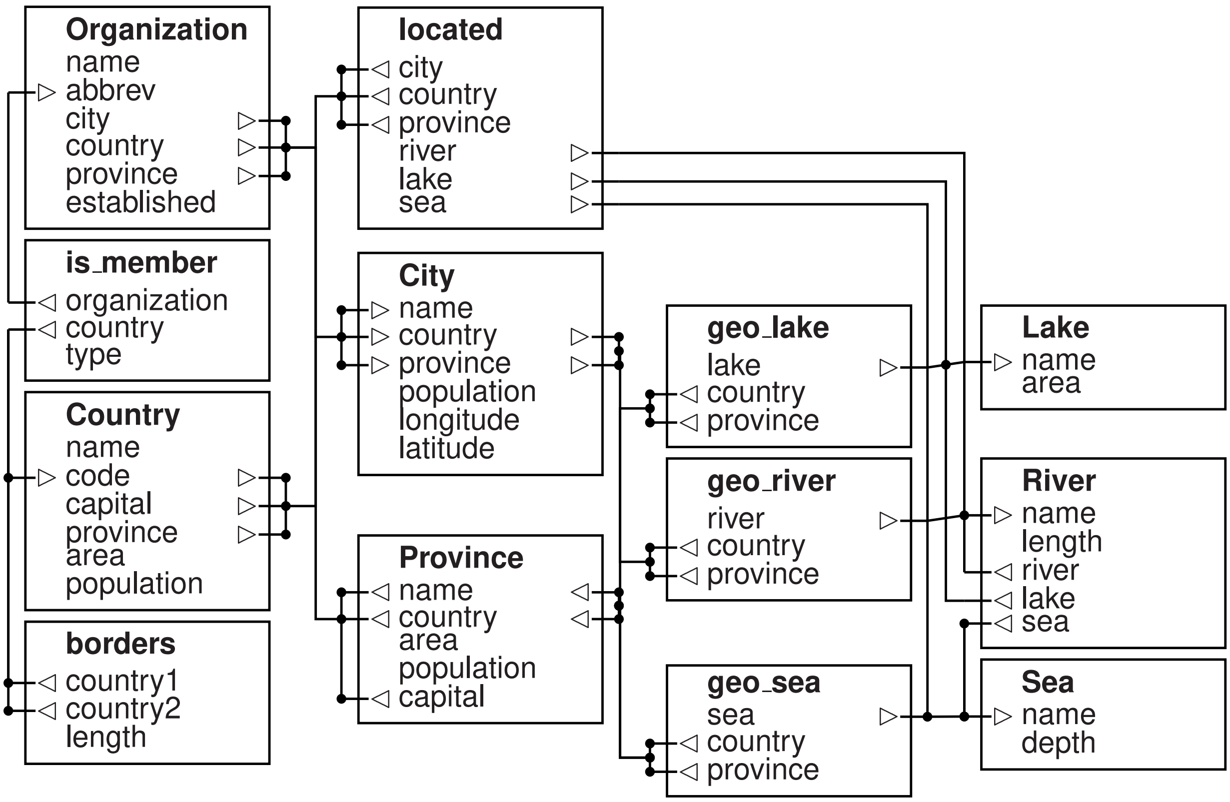
Continent	
Name	**Area**
Europe	9562489.6
Africa	3.02547e+07
Asia	4.50953e+07
America	3.9872e+07
Australia	8503474.56

Country				
Name	**code**	**Population**	**Capital**	**...**
Germany	D	83536115	Berlin	
Sweden	S	8900954	Stockholm	
Canada	CDN	28820671	Ottawa	
Poland	PL	38642565	Warsaw	
Bolivia	BOL	7165257	La Paz	
..	..	..	..	

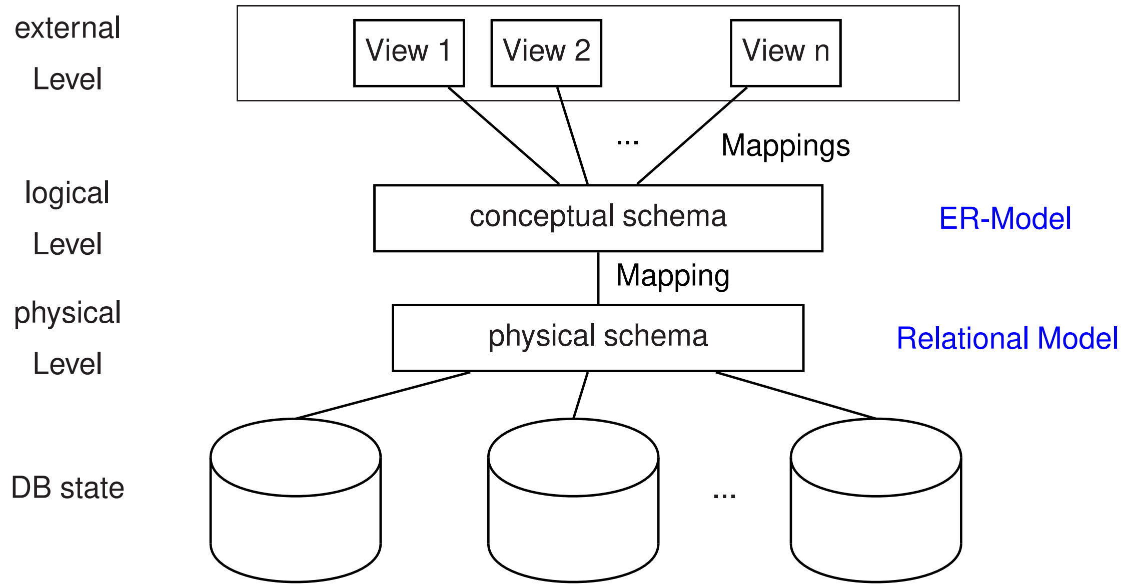
encompasses		
Country	**Continent**	**Percent**
VARCHAR(4)	VARCHAR(20)	NUMBER
R	Europe	20
R	Asia	80
D	Europe	100
...	...	...

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

Graphical representation of the relational schema of the MONDIAL database (excerpt):



3-LEVEL ARCHITECTURE OF A DBMS (ANSI/SPARC)



2.2.2 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** $\mathbf{R}$ 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 $A \in \bar{X}$ is associated to a (non-empty) **domain**, called dom(A).
- dom$(\bar{X})$:= dom$(A_1) \times \ldots \times$ dom(A_m).

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**.

RELATIONAL DATABASES – FORMALIZATION: SEMANTICS

- A **(relational) database** (or, more explicitly, a **database state**) $\mathcal{S}$ (over $\mathbf{R} = \{R_1(\bar{X}_1), \ldots, R_n(\bar{X}_n)\}$) is a **relational structure** over $\mathbf{R}$.
- A **relational structure** $\mathcal{S}$ associates each $R_i(\bar{X}_i)$ to a **relation** $\mathcal{S}(R_i)$ over $\bar{X}_i$.
- elements of a relation are called **tuples**.
  (every tuple represents an entity or a relationship.)
- a **tuple** μ over $\bar{X}$ is a mapping $\mu : \bar{X} \to$ dom$(\bar{X})$; or, for each individual attribute, $\mu : A \to$ dom(A).
  Tup$(\bar{X})$ denotes the set of all tuples over $\bar{X}$.
- a **relation** r over $\bar{X}$ is a finite *set* $r \subseteq$ Tup$(\bar{X})$ – usually represented by a table.
- Rel$(\bar{X}) := 2^{Tup(\bar{X})}$ is the set of all relations over $\bar{X}$.

PERSPECTIVES: RELATIONAL VS. SET THEORY

- Relations are sets of tuples.
  $\Rightarrow$ **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 autors use the term "interpretation" instead of "structure")

Relational theory is based on "classical" logic results:
$\Rightarrow$ **relational calculus**

- first-order logic
- finite model theory
- complexity results
- (deductive databases)

KEYS

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

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

If $\mu_1(K) = \mu_2(K)$ (i.e., μ_1 and μ_2 coincide in the values of K), then $\mu_1 = \mu_2$.

- 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.

Inclusion Constraints and Referential Integrity

Consider relation schemata $R_1(\bar{X}_1)$ und $R_2(\bar{X}_2)$. Let $\bar{Y}_1 \subseteq \bar{X}_1$ and $\bar{Y}_2 \subseteq X_2$ two attribute vectors of the same length.

$r_1 = \mathcal{S}(R_1)$ and $r_2 = \mathcal{S}(R_2)$ 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 $R_2.\bar{Y}_2$.
  - 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

... see later where referential integrity constraints come from.

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”
- a **partial tuple** over $\bar{X}$ is a mapping s.t.

$$(\forall A \in \bar{X})\ \mu(A) \in \text{dom}(A) \cup \{null\}.$$

A relation is called **partial** if it contains partial tuples.

2.2.3 Exercise

Exercise 2.4
Consider the relation schema $R(\bar{X})$, where $\bar{X} = \{A, B\}$ and dom(A) = dom$(B) = \{1, 2\}$.

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

Chapter 3
Database Design

Database design is part of the design phase in the software design process. It is based on the documents of the *requirements analysis*.

3.1 Database Design: Design Steps

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).

Detailed Physical Design: definition of the actual storage and appropriate auxiliary data structures (for enhanced efficiency).

"Classical" database design is restricted to the modeling of (static) structures, not considering the (dynamic) processes resulting from the execution (see UML and the *Information Systems* course).

3.2 Logical Schema: Mapping ERM to RM

Starting with the ER schema, the relational schema is designed.

[Overview slide]

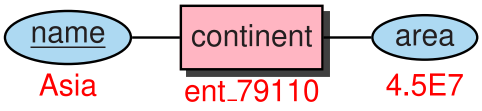
Let E_{ER} an entity type and B_{ER} a relationship type in the ERM.

- 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:
  $(B_{ER}, \{RO_1 : E_1, \ldots, RO_k : E_k\}, \{A_1, \ldots, A_m\}) \rightarrow$
  $B(E_1\_K_{11}, \ldots, E_1\_K_{1p_1},\ \ldots,\ E_k\_K_{k1}, \ldots, E_k\_K_{kp_k},\ A_1, \ldots, A_m)$,
  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 complexity, 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

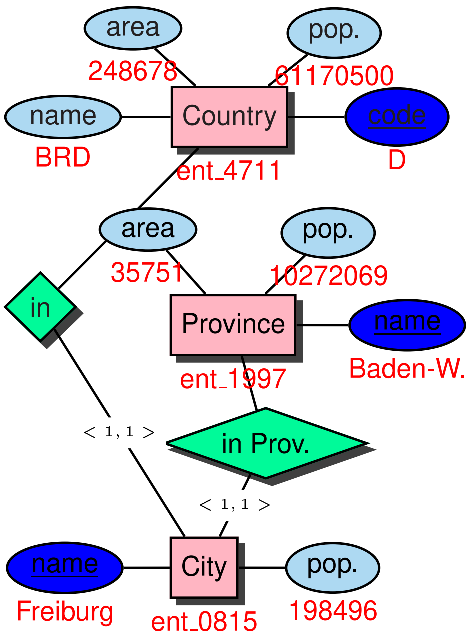
$(E_{ER}, \{A_1, \ldots, A_n\}) \rightarrow E(A_1, \ldots, A_n)$



Continent	
Name	**Area**
VARCHAR(20)	NUMBER
Europe	9562489.6
Africa	3.02547e+07
Asia	4.50953e+07
America	3.9872e+07
Australia	8503474.56

Weak Entity Types

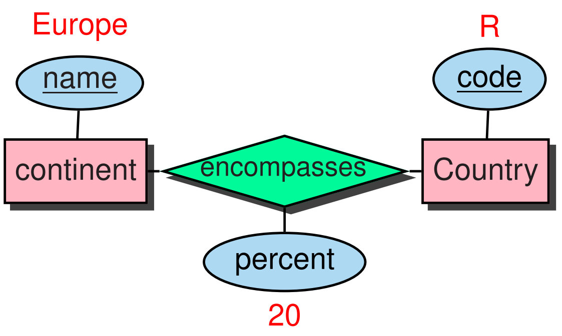
For weak entity types, the key attributes of the identifying entity type(s) must be added.



City				
Name	**Country**	**Province**	**Population**	**...**
Freiburg	D	Baden-W.	198496	..
Berlin	D	Berlin	3472009	..
..	..	..	..	..

Relationship Types

$(B_{ER}, \{RO_1 : E_1, \ldots, RO_k : E_k\}, \{A_1, \ldots, A_m\}) \rightarrow$
$B(E_1\_K_{11}, \ldots, E_1\_K_{1p_1}, \ldots, E_k\_K_{k1}, \ldots, E_k\_K_{kp_k},\ A_1, \ldots, A_m)$
where $\{K_{i1}, \ldots, K_{ip_i}\}$ are the primary keys of $E_i, 1 \leq i \leq k$.
(it is allowed to rename, e.g., to use *Country* for *Country.Code*)

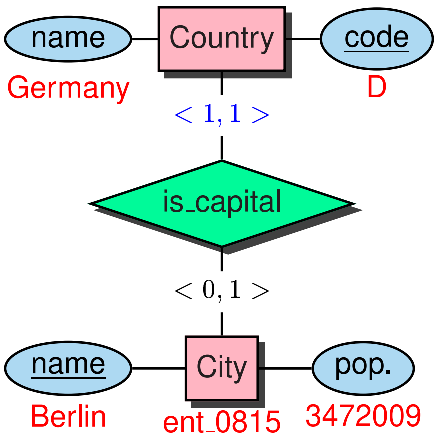


encompasses		
Country	**Continent**	**Percent**
VARCHAR(4)	VARCHAR(20)	NUMBER
R	Europe	20
R	Asia	80
D	Europe	100
...	...	...

- Note: for weak entity types, the global key must be used (exercise: is_capital)

Relationship Types: 1:n-Relationships

In case that $k = 2$ (binary relationship) and a (0,1)- or (1,1)-relationship complexity (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)

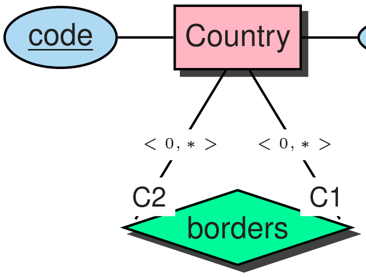


Country					
Name	**code**	**Population**	**Capital**	**Province**	**...**
Germany	D	83536115	Berlin	Berlin	..
Austria	A	8023244	Vienna	Vienna	..
Canada	CDN	28820671	Ottawa	Quebec	..
Bolivia	BOL	7165257	La Paz	Bolivia	..
..	..	..	..	..	..

Other examples: flows_into, headquarters of organizations

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.



borders	
Country1	Country2
D	F
D	CH
CH	F
..	..

EXERCISE

Exercise 3.1
Give a relational schema to the following ER schema: