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
- **"First Normal Form":** only domains of atomic datatypes, no records, lists, sets etc.
- A **(relational) database schema** $\mathbf{R}$ 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:

Continent	
name	**area**
VARCHAR(20)	NUMBER
Europe	10523000
Africa	30221500
Asia	44614500
North America	24709000
South America	17840000
Australia	9000000

Relations: Example

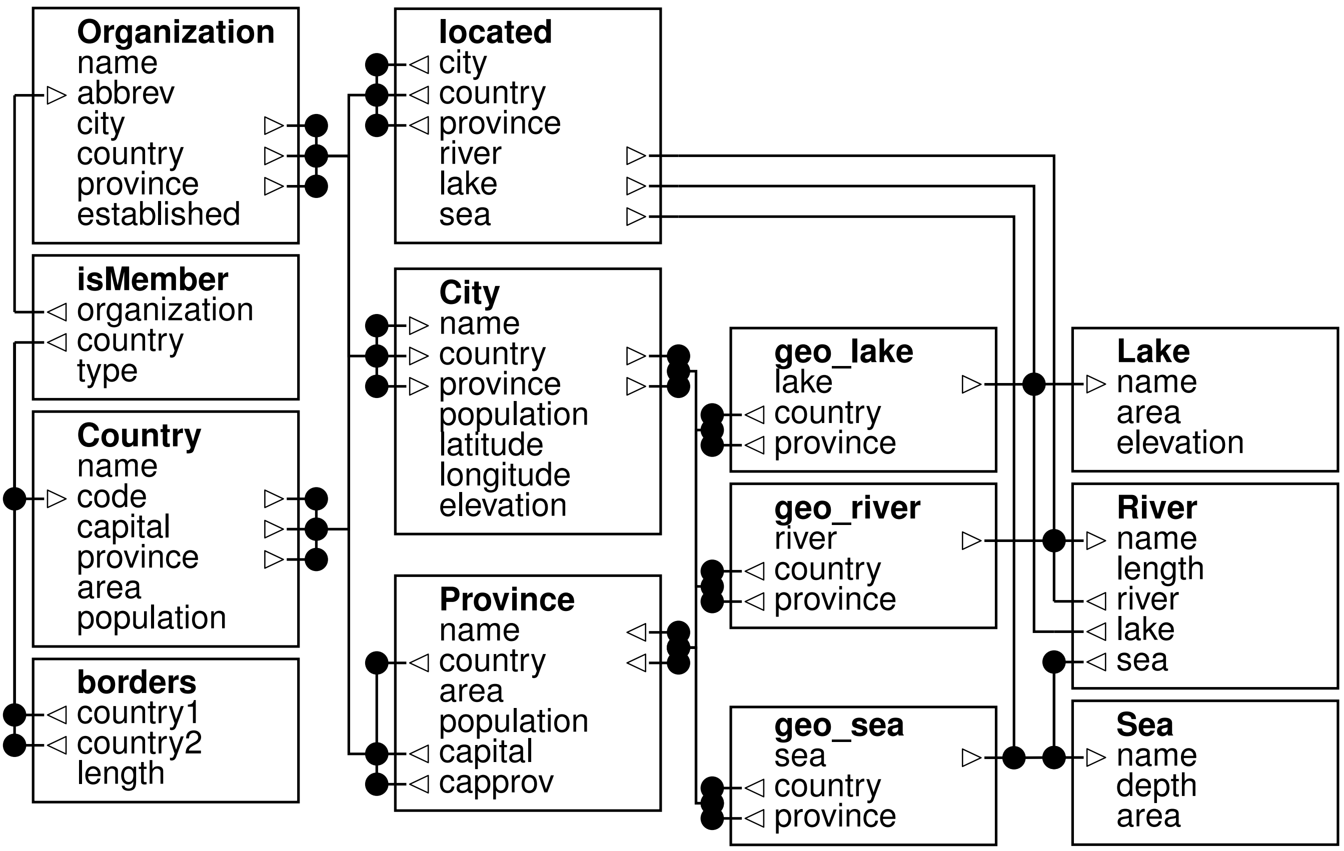
Continent	
name	**area**
Europe	10523000
Africa	30221500
Asia	44614500
North America	24709000
South America	17840000
Australia	9000000

Country				
name	**code**	**population**	**area**	**...**
Germany	D	83536115	356910	
Sweden	S	8900954	449964	
Russia	R	143666931	17075200	
Poland	PL	38642565	312683	
Bolivia	BOL	1098580	7165257	
..	..	..	..	

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):



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).

2.3 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 R_{ER} a relationship type in the ERM.

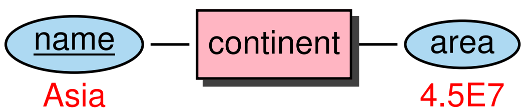
- Entity types: $(E_{ER}, \{A_1, \ldots, A_n\}) \rightarrow E(A_1, \ldots, A_n)$,
- For weak entity types, the key attributes of the identifying entity type must be added.
- Relationship types:
  $(R_{ER}, \{RO_1 : E_1, \ldots, RO_k : E_k\}, \{A_1, \ldots, A_m\}) \rightarrow$
  $B(E_1\_K_{11}, \ldots, E_1\_K_{1p_1},\ \ldots,\ E_k\_K_{k1}, \ldots, E_k\_K_{kp_k},\ A_1, \ldots, A_m)$,
  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

$(E_{ER}, \{A_1, \ldots, A_n\}) \rightarrow E(A_{i_1}, \ldots, A_{i_k})$

where $\{A_{i_1}, \ldots, A_{i_k}\} \subseteq \{A_1, \ldots, A_n\}$ are the scalar (i.e., not multivalued) attributes of E_{ER} – multivalued attributes are mapped separately.



Continent

name	area
VARCHAR(20)	NUMBER
Europe	10523000
Africa	30221500
Asia	44614500
North America	24709000
South America	17840000
Australia	9000000

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

MULTIVALUED ATTRIBUTES

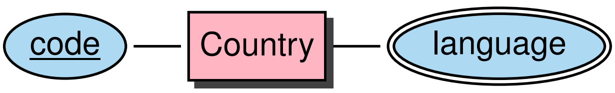
... one thing left:

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.
(renaming 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.

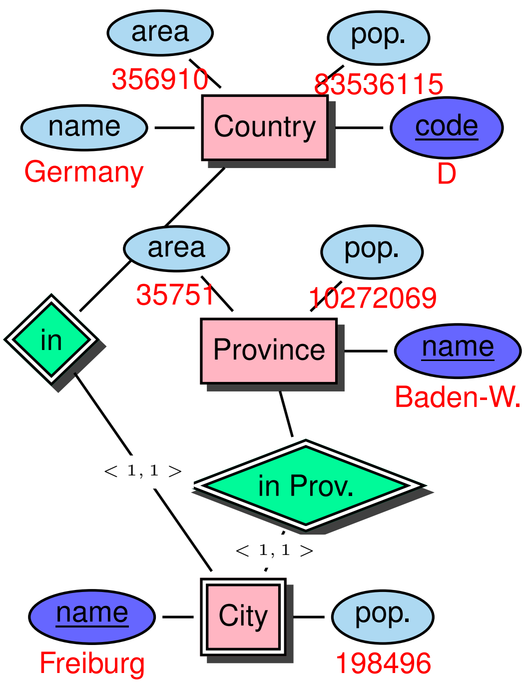


Languages

country	language
D	German
CH	German
CH	French
..	..

WEAK ENTITY TYPES

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

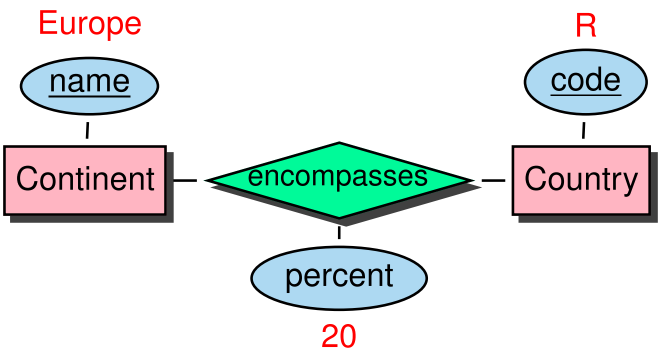


City

name	country	province	population	...
Freiburg	D	Baden-W.	198496	..
Berlin	D	Berlin	3472009	..
Freiburg	CH	FR	NULL	..
Cordoba	E	Andalucia	328326	..
Cordoba	RA	Cordoba	1207774	..
..	..	..	..	..

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 $E_i, 1 \leq i \leq k$.
(it is allowed to rename, e.g., to use *Country* for *Country.Code*)



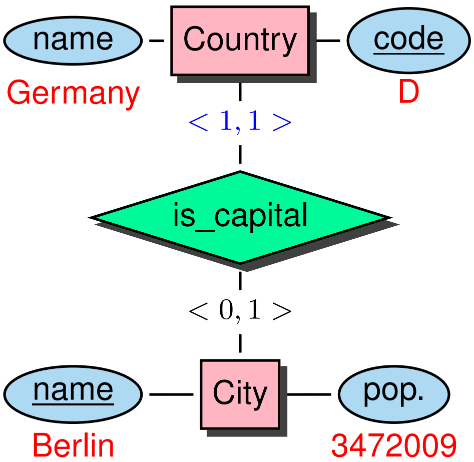
encompasses		
country	**continent**	**percent**
VARCHAR(4)	VARCHAR(20)	NUMBER
R	Europe	20
R	Asia	80
D	Europe	100
...	...	...

- Note: for 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)

Example: the "capital" relationship can be merged into the "country" table (or, less intuitively, into the "city" table, where the <0,1> indicates that it would generate lots of NULL values).

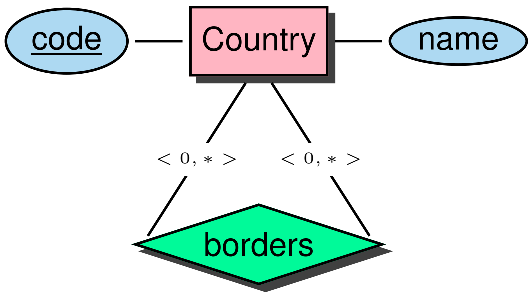


Country					
name	**code**	**population**	**capital**	**province**	**...**
Germany	D	83536115	Berlin	Berlin	..
Austria	A	8023244	Vienna	Vienna	..
Canada	CDN	28820671	Ottawa	Quebec	..
Bolivia	BOL	7165257	La Paz	Bolivia	..
..	..	..	..	..	..

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

Recursive Symmetric Relationship Types

- recursive non-symmetric relationship types (river-flowsInto): use role names as column names.
- recursive symmetric relationship types (borders): invent column names
  - Symmetric storage would introduce redundancy and risk for inconsistencies. Store only one direction and create a symmetric SQL view from it (belongs to the "external level" of the 3-level-architecture)



borders		
country1	**country2**	**length**
D	F	451
~~F~~	~~D~~	~~450~~
D	CH	334
CH	F	573
:	:	:

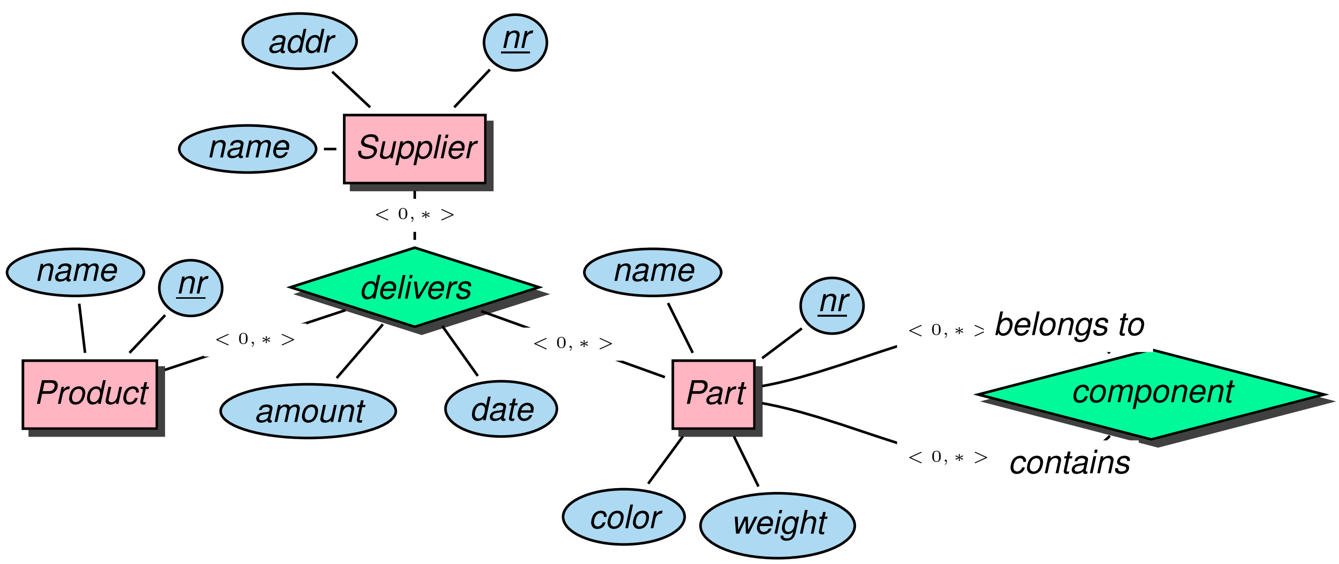
SQL view (see later)

```
CREATE VIEW symmborders AS
  (SELECT * FROM borders) UNION (SELECT country2, country1, length FROM borders)
```


Exercise

Exercise 2.4

Give a relational schema for the following ER schema:



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** $\mathbf{R}$ 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 attribute $A \in \bar{X}$ is associated to a (non-empty) **domain** of atomic values, called dom(A).
- dom$(\bar{X})$:= dom$(A_1) \times \ldots \times$ dom(A_m).
- Example: Continent(name, area)
  dom(continent.name) = VARCHAR, dom(continent.area) = NUMBER
  dom(Continent) = VARCHAR $\times$ 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**.

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.)
  Note: as a set of tuples, there cannot be two tuples that have all the same values.
- a **tuple** μ over $\bar{X}$ is a mapping $\mu : \bar{X} \to \text{dom}(\bar{X})$; or, for each individual attribute, $\mu : A \to \text{dom}(A)$.
  Tup$(\bar{X})$ denotes the set of all tuples over $\bar{X}$.

  Example: Consider tuples in the Continent(name, area) table:
  $\mu =$ name $\mapsto$ "Asia", area $\mapsto$ 44614500
  with μ(name) = "Asia", μ(area) =44614500
- 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.
  ⇒ **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:
⇒ **relational calculus**

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

Keys

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., μ_1 and μ_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,
- no "embedded relations" (i.e., partial functional dependencies): 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. $\bar{X}'$.
  [3rd Normal Form, cf. Slide 374; Example see Slide 67]

Keys: Additional Formal Requirements

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:

{code} $\rightarrow$ *{name},* *{name}* $\rightarrow$ *{code}*
{code} $\rightarrow$ *{area, population, capital, capprov}*
{code} $\rightarrow$ *{name, code, area, population, capital, capprov}*
{name} $\rightarrow$ *{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**:
  $\bar{K} \subseteq \bar{X}$ is a possible key of $R(\bar{X})$ if $\bar{K} \rightarrow \bar{X}$.

Additionally:

- keys must be minimal, i.e., no subset $\bar{K}' \subsetneq \bar{K}$ satisfies the key property:
  there is no subset $\bar{K}' \subsetneq \bar{K}$ s.t. $\bar{K}' \rightarrow \bar{X}$.
  (otherwise: take $\bar{K}'$ as key)
- *every* single attribute should be *fully dependent* on the *complete* key: for every $A \in (\bar{X} \setminus \bar{K})$: there is no subset $\bar{K}' \subsetneq \bar{K}$ s.t. $\bar{K}' \rightarrow A$.
  (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

Country (bad schema)							
Name	**Code**	**Language**	**Percent**	**Population**	**Area**	**Capital**	**Province**
Germany	D	German	100	83536115	356910	Berlin	Berlin
Switzerland	CH	German	65	7207060	41290	Bern	BE
Switzerland	CH	French	18	7207060	41290	Bern	BE
Switzerland	CH	Italian	12	7207060	41290	Bern	BE
:	:	:	:	:	:	:	:

- 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, <u>Code</u>, Population, Capital, Province) and Languages(<u>Code</u>, <u>Language</u>, 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?

Inclusion Constraints and Referential Integrity

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.

$r_1 = \mathcal{S}(R_1)$ 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 $R_2.\bar{Y}_2$.
  - $\bar{Y}_1$ 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.

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

  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 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?* □