

Constraints of binary relationship sets

- ❑ **1:1-relationship** (one-to-one relationship)

if for a binary relationship set $R(E_1, E_2)$ each entity in E_1 is associated with *at most* one entity in E_2 , and vice versa

- ❑ **1:m-relationship** (one-to-many relationship)

if for a binary relationship set $R(E_1, E_2)$ each entity in E_1 is associated with any number (zero or more) of entities in E_2 , and each entity in E_2 is associated with *at most* one entity in E_1

- ❑ **m:1-relationship** (many-to-one relationship)

analogous to the 1:m-relationship

- ❑ **m:n-relationship** (many-to-many relationship)

if for a binary relationship set $R(E_1, E_2)$ each entity in E_1 is associated with any number (zero or more) of entities in E_2 , and vice versa

- ❑ constraints considered as partial functions, e.g.




for 1:1-relationship: has_husband: wives \rightarrow husband, has_wife: husband \rightarrow wives

for 1:m- und m:1-relationships: employed_by: persons \rightarrow companies

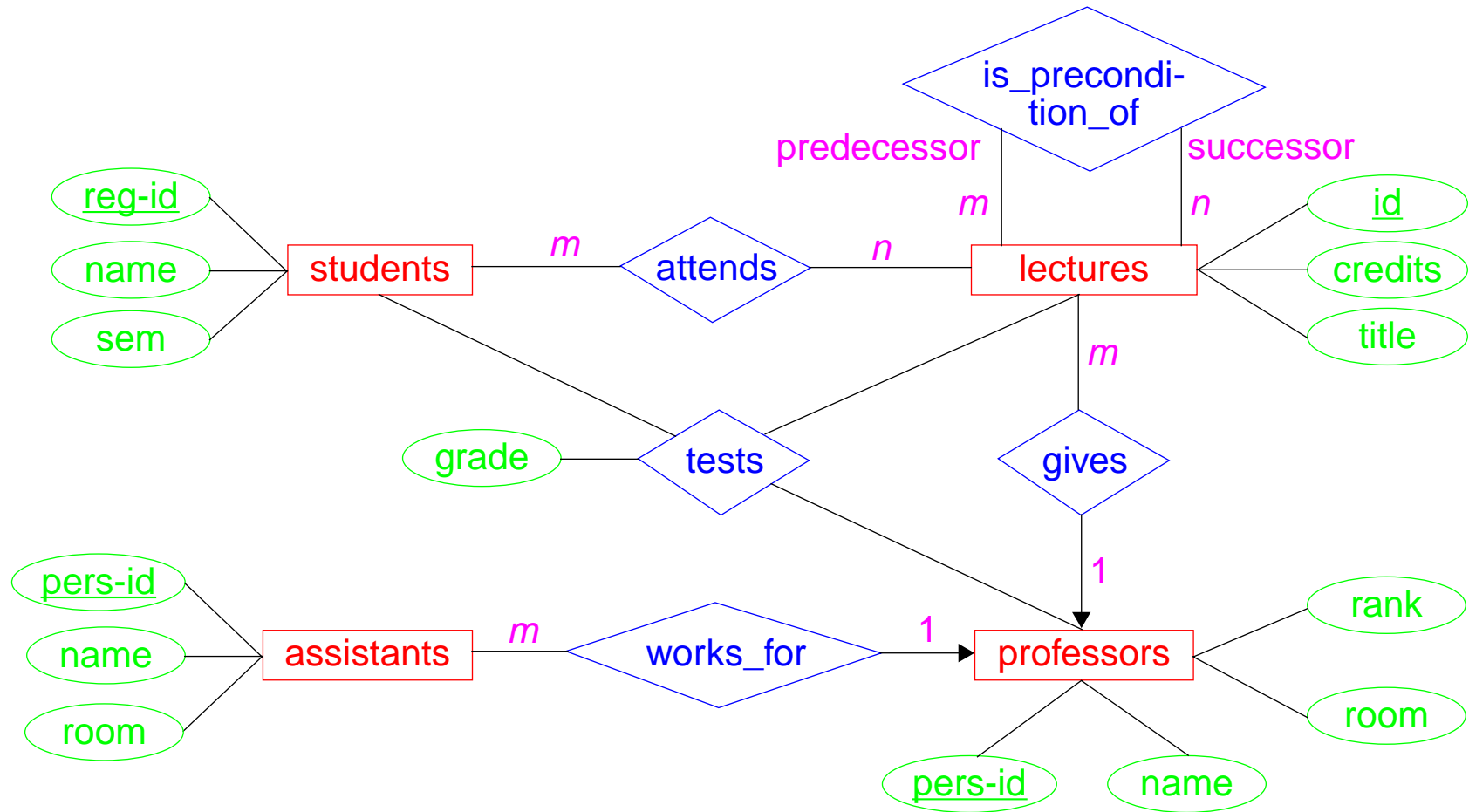
E-R diagrams

- ❑ graphical representation of entity sets, relationship sets, and their attributes by means of a graph


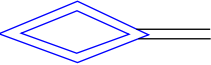
Notations

- ❑ rectangles represent entity sets: 
- ❑ ellipses represent attributes: 
 - they are connected with their entity set by undirected edges
 - key attributes are underlined
- ❑ relationship sets are represented by diamonds: 
 - relationship sets are connected with their pertaining entity sets by edges
 - edges carry information about cardinality according to imposed constraints (frequently *arrow notation* is also used)
- ❑ a role of a relationship set is attached to the corresponding edge

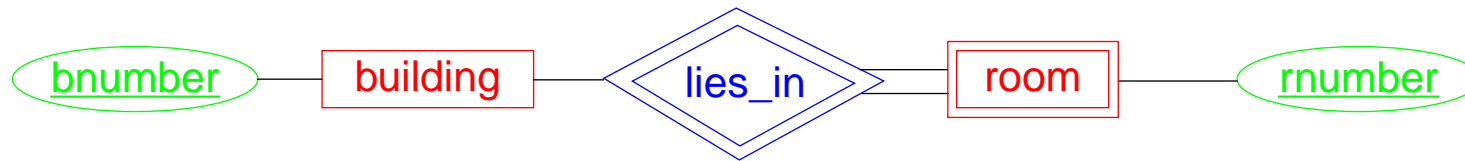
Example: conceptual university schema



Extensions

- existence dependent (**weak**) entity sets
 - assumption so far: entities exist autonomously and can be uniquely identified within an entity set by their key attributes (**strong** entity set)
 - in reality there are also **weak** entities that do not have sufficient attributes to form a key. These entities are
 - + dependent in their existence from another, superior entity and
 - + can be uniquely identified only in combination with the key of a superior entity
 - superior entity set is called **identifying** or **owner entity set**
 - graphical notation: 
- identifying relationship set
 - a weak entity set E_1 must be associated with an identifying entity set E_2 by an **identifying relationship set**, if the key of E_1 comprises the key of E_2 and if it contains one or more additional attributes of E_1
 - relationship from the weak entity set to the superior entity set has usually an $m:1$ -cardinality and more seldom a 1:1-cardinality
 - graphical notation: 

□ example:



□ total participation of an entity set in a relationship

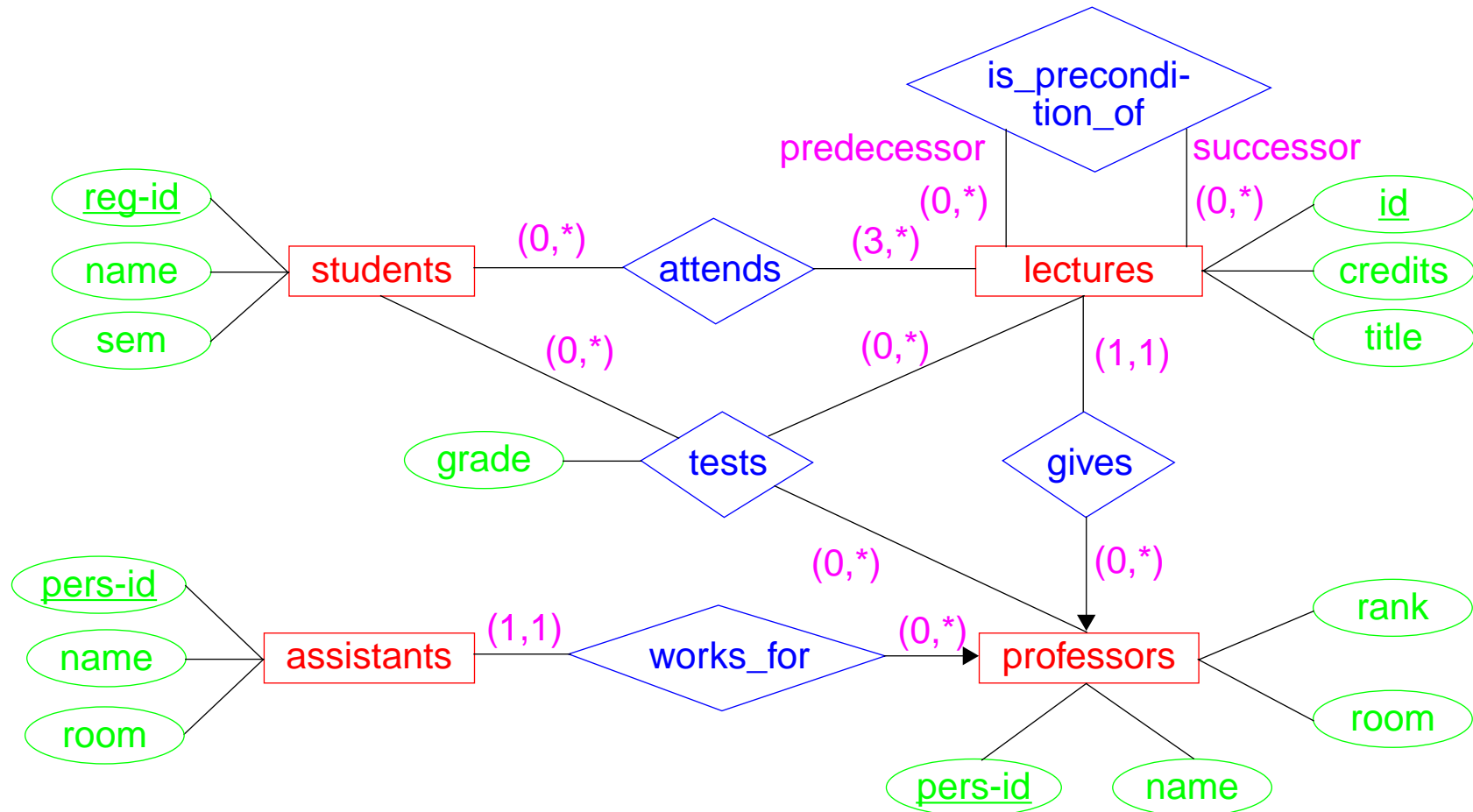
- all entities of an entity set E_1 are associated with another entity set E_2 by a relationship set R
- this holds, in particular, for weak entity sets
- example:



□ more precise characterization of cardinalities of relationship sets

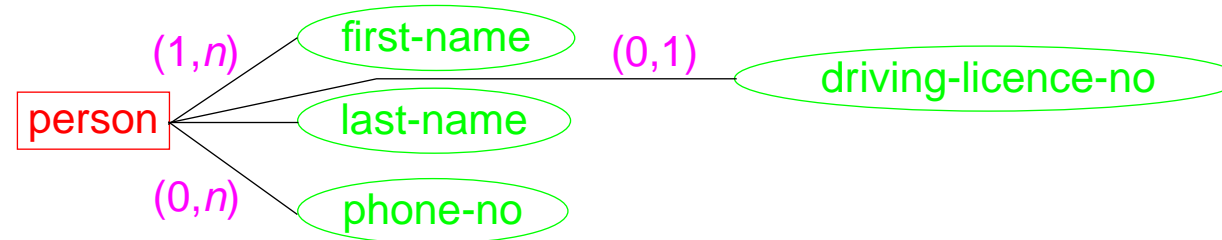
- (min, max) -notation
- for each entity set participating in a relationship set
 - + min expresses that each entity of this set is in relationship at least min times
 - + max expresses that each entity of this set is in relationship at most max times

- special cases
 - + $min = 0$: an entity does not have to be in relationship (optional)
 - + $max = *$: an entity may be in relationship arbitrarily many times
- example: conceptual university schema with (min, max)-notations



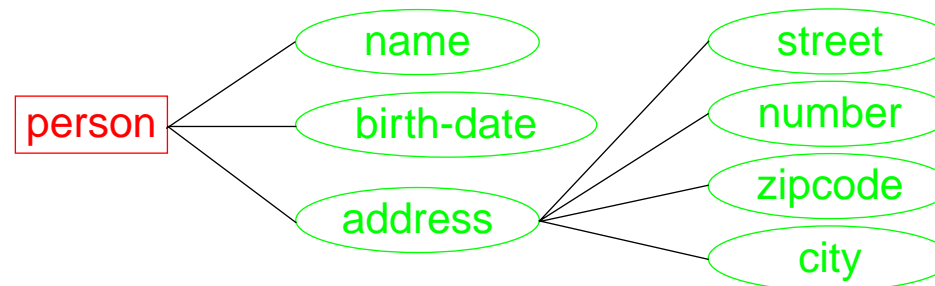
❑ multivalued attributes

- optional attribute: minimal cardinality is equal to 0
- simple attribute: cardinality is equal to 1
- prescribed attribute: minimal cardinality is equal to 1
- **multivalued attribute**: maximal cardinality is equal to n
- example:




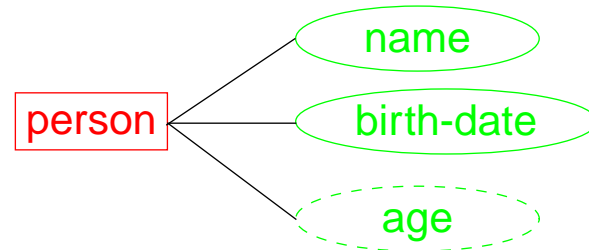
❑ **composite** attributes

- grouping of attributes of the same entity set or relationship set which are closely related
- antonym: simple attribute
- example:



❑ derived attributes

- attribute that can be derived from one or more attributes
- antonym: **base/stored** attribute
- graphical representation: 
- example:



Generalization

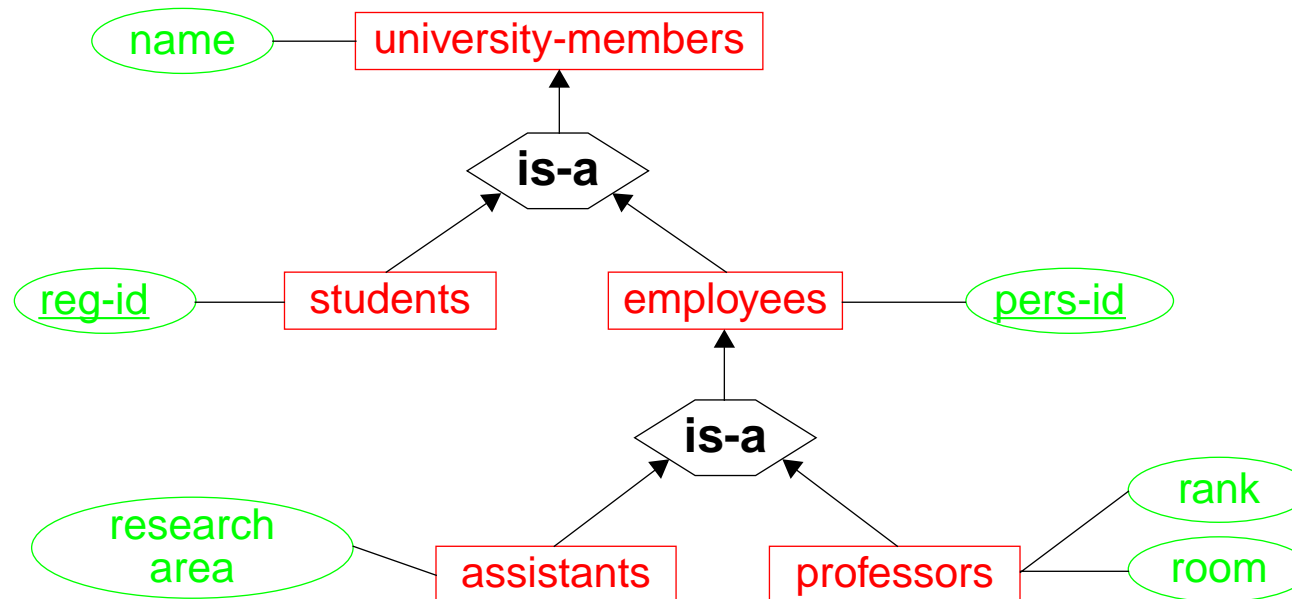
❑ goals

- abstraction at the set level: better (i.e., more understandable and more concise) structuring of entity sets
- abstraction at the instance level: similar entities are to be modeled by a common entity set

❑ „factoring“ (extracting) properties (attributes, relationships) of similar entity sets (**sub-class, subtypes, categories**) to a common **superclass (supertype)**

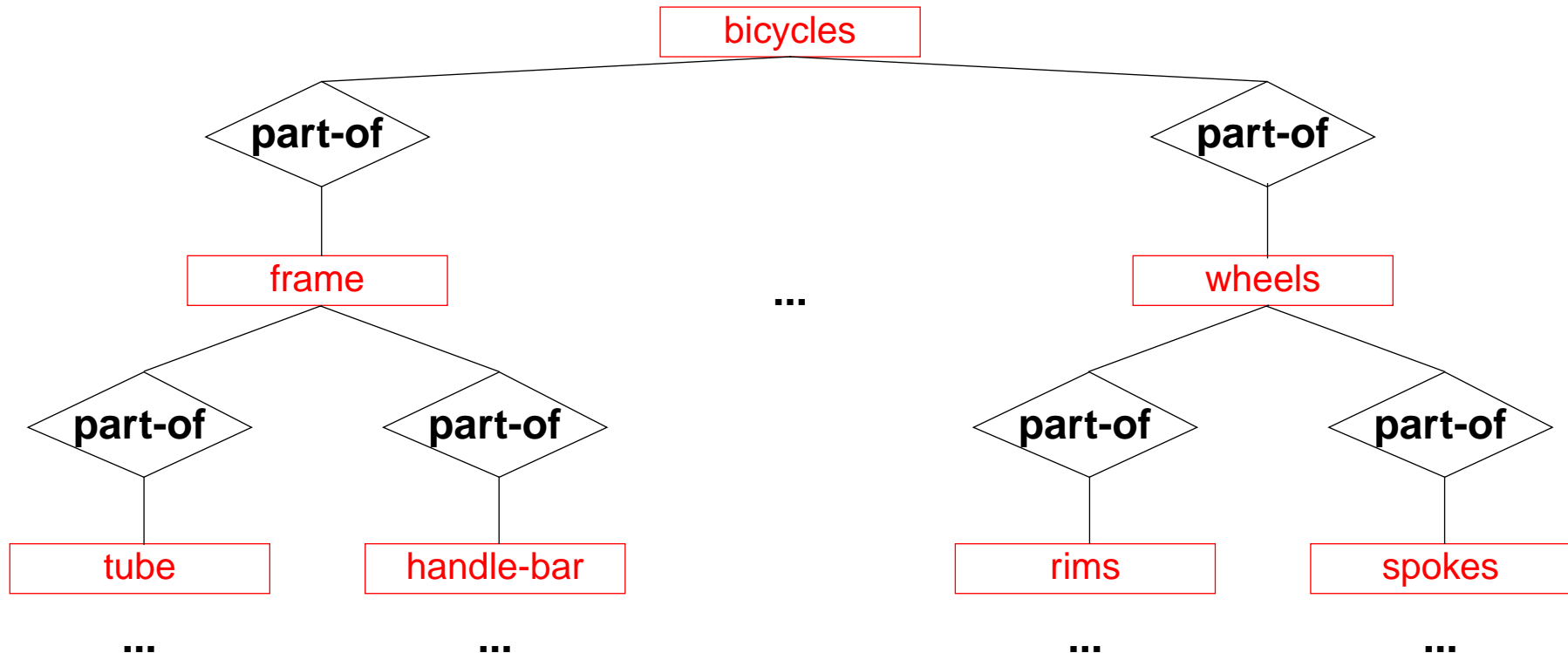
❑ properties that cannot be extracted remain with the respective subclass, i.e., the subclass is a **specialization** of the superclass

- ❑ **inheritance** as the key concept of **generalization**: a subclass inherits all properties of a superclass
- ❑ entities of a subclass are implicitly considered as entities of the superclass, therefore **is-a** in the graphical representation
 - set of entities of the subclass is a subset of the set of entities of the superclass
- ❑ two special cases
 - **disjoint/overlapping specialization**: all subclasses of a superclass are pairwise disjoint/overlapping
 - **total specialization**: the superclass does not contain explicit elements, but is only given by the union of its subclasses (antonym: **partial specialization**)



Aggregation

- ❑ goal: distinct entity sets which together form a structured superclass are associated with each other
- ❑ an **aggregation** is a special relationship set which associates each superior entity set with several subordinate entity sets
- ❑ **part-of**-relationship
- ❑ example: construction of a bicycle



3. Relational Data Model

3.1 Introduction

- ❑ seminal article: E.F. Codd. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM 13(6):377-387 (1970)
- ❑ commercial DBMSs like Oracle, Informix, SQL Server, Sybase, DB/2 are based on the relational model
- ❑ reasons for the success of the relational data model
 - flat tables (relations) as the simple underlying data structure
 - no nested complicated structures
 - **set oriented** processing of data in contrast to **record oriented** processing prevailing until then (hierarchical model, network model)
 - simple comprehensibility also for the unskilled user
 - good performance for standard database applications
 - existence of a mature, formal theory (in contrast to other data models), in particular with respect to the design of relational databases and with respect to an efficient processing of user queries

3.2 Definition of the Relational Model

Basic structure

- Given n **domains** D_1, D_2, \dots, D_n
 - examples for domains: data types *integer*, *string[20]*, *real*, *bool*, *date*, ...
 - domains need not be disjoint, i.e., $D_i = D_j$ ist admissible for $i \neq j$
 - domains may contain only *atomic* values, they must not be structured
- a **relation (instance)** r_R ist defined as a subset of the Cartesian product of n domains:

$$r_R \subseteq D_1 \times D_2 \times \dots \times D_n \quad (r_R \text{ finite})$$
- r_R is an **occurrence (instance)** of a pertaining **relation schema** R (analogously to the programming language notions of *variable* and *type*).
- an element of the set R is called **tuple**, tuple has **arity** n
- example:
 - domains: $D_1 = \{a, b, c\}$, $D_2 = \{0, 1\}$
 - Cartesian product: $D_1 \times D_2 = \{(a, 0), (a, 1), (b, 0), (b, 1), (c, 0), (c, 1)\}$
 - examples for instances: $r_1 = \{(a, 0), (b, 0), (c, 0), (c, 1)\}$, $r_2 = \{(a, 0)\}$, $r_3 = \emptyset$

Schema definition

- ❑ distinction between the **schema** of a relation R , which is given by the n domains (data types), and the current **instance** of this relation schema, which is given by a subset of the Cartesian product
- ❑ schema analogously to the programming language notion of type definition
- ❑ a **relation schema** R , denoted by $R(A_1, A_2, \dots, A_n)$, consists of the **relation name** R and a list of attributes A_1, A_2, \dots, A_n
- ❑ each **attribute** A_i is the name of a role played by domain D_i in the relation schema R
 - D_i is also the domain (type) of A_i
 - notation: $D_i = \text{dom}(A_i)$
- ❑ for the schema $R(A_1, A_2, \dots, A_n)$ holds: $r_R \subseteq \text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times \text{dom}(A_n)$.
- ❑ we describe the schema of R also in the form $R(A_1 : D_1, A_2 : D_2, \dots, A_n : D_n)$.
- ❑ because we often do *not* make a clear distinction between the meta level (schema) and the instance level (occurrence), we also denote relation instances with the letter R

- representation of a relation as tables mit **rows** (tupels) und **columns**

R	A	N
	a	0
	a	1
	b	0
	b	1
	c	0
	c	1

A and N are attributes and have the function of column names

- example: relation Students(RegNo : *string*, Name : *string*, Age : *integer*, ...)

Students	RegNo	Name	Age	...
	123456	Meyer John	22	...
	456123	Smith Ben	23	...
	321654	Benson Jeff	27	...
	654321	Bates Allen	21	...
