Information Systems

German-Russian Institute of Advanced Technology (GRIAT)

Summer term 2015

Rita Schindler | TU Ilmenau, Germany
www.tu-ilmenau.de/dbis

Goal of this Lecture

- Knowledge about database systems, their foundations
  - modeling & design, query formulation
- ... their implementation
  - storage & index structures, query optimization & processing, transaction processing
- ... and application
  - particularly, in data warehousing

- Prerequisites:
  - Basic understanding of relational databases, SQL

Overview

- Introduction & Conceptual Design

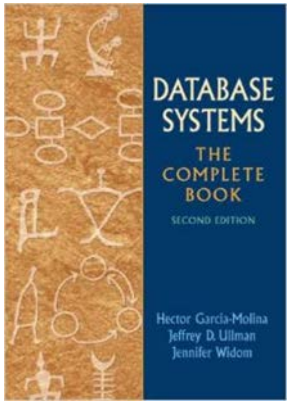
- Relational Database Theory
- SQL

- Storage and Index Structures
- Query Processing
- Transaction Processing & Recovery

- Architecture & Design of Data Warehouses
- Querying Data Warehouses
- Indexing & Materialized Views

Course Material

- Handouts
- Exercises, tasks and questions at the end of each chapter
- Recommended readings:
  - Garcia-Molina, Ullman, Widom
    Database Systems – The Complete Book
    Pearson/Prentice Hall, 2. ed., 2009
  - . . .



  - C. J. Date
    Introduction to Database Systems
    Addison-Wesley, 8. edition, 2003

Список Литературы

Издательская группа "Диалектика-Вильямс"

http://www.dialektika.com

Подразделение:

Издательский дом "Вильямс"

http://www.williamspublishing.com/

Гектор Гарсиа-Молина, Джеффри Д. Ульман, Дженнифер Уидом

Системы баз данных. Полный курс

1088 стр.; 2004; **Вильямс**

ISBN 5-8459-0384-X, 0-1303-1995-3;

К. Дж. Дейт

Введение в системы баз данных. Восьмое издание

1328 стр.; 2008; **Вильямс**.

ISBN 978-5-8459-0788-2, 0-321-19784-4;

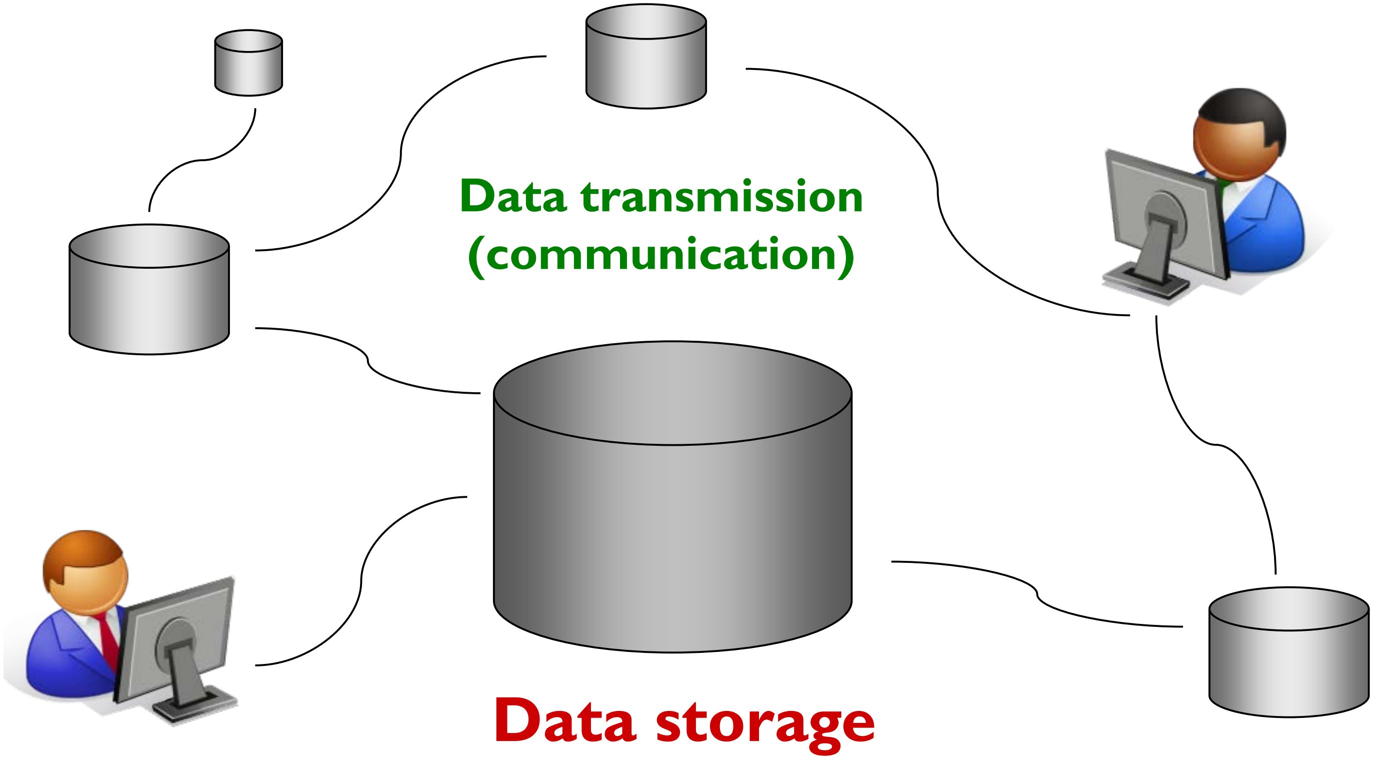
Information Systems

Chapter 1:
Foundations &
Conceptual Database Design

Rita Schindler | TU Ilmenau, Germany
www.tu-ilmenau.de/dbis

Information Systems

Information exchange: Overcoming of space and time



Information Systems (cont.)

- Communication and data storage – **core components** of the most complex networked systems of computer science

  Example **Amazon**

  - Online-Marketplace
  - Technology:
    - Linux,
    - Oracle DB-Software,
    - three databases with capacities of 7.8 TB, 18.5 TB, 24.7 TB

- Only their combination provides the most service offerings

- Our focus: data storage

Databases

Databases today are essential to every business.

Whenever you visit a major Web site –

- Google, Yahoo!, Amazon.com, . . .
- or thousands of smaller sites that provide information –

there is a database behind the scene serving up the information you request

The power of databases

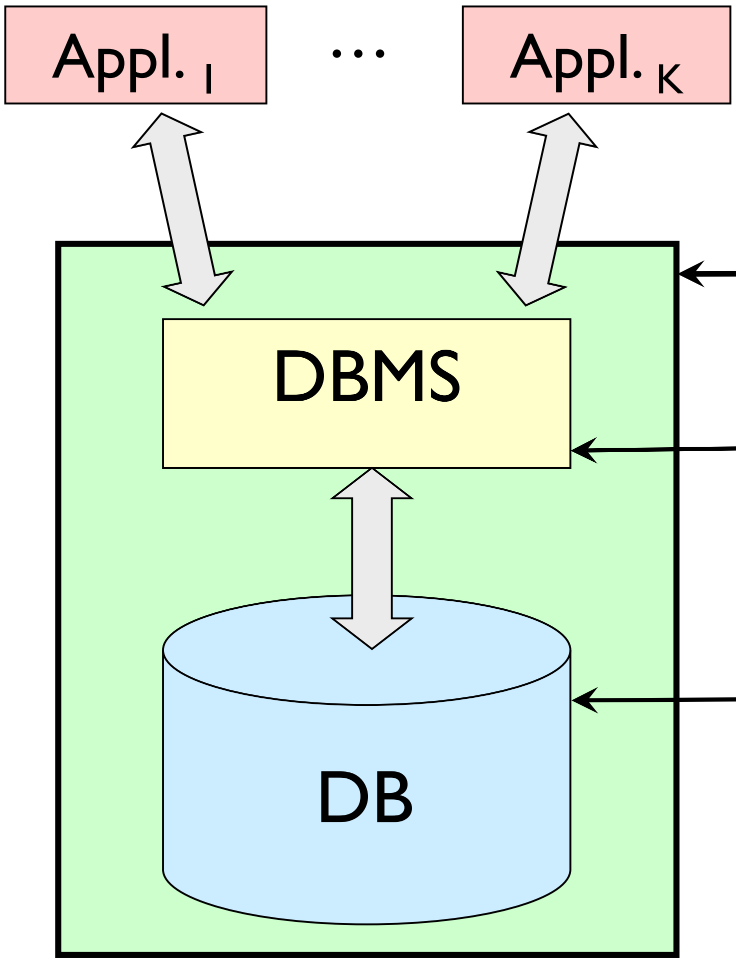
- comes from a body of knowledge and technology that has developed over several decades
- is embodied in a specialized software package

What is a Database System?

- A Database System (DBS) consists of a Database (DB) and a Database Management System (DBMS)
- A Database is a (typically very large) integrated collection of interrelated data which are stored in files.
  - Data describe information and activities about one or more related organizations (portion of the real world).

    For example, an university database might contain information about
    - entities (e.g., students, courses, faculties, . . .)
    - relationships among entities (e.g., ’Bill is taking Database Systems course’)
- A Database Management System
  - is a collection of software packages designed to store, access, and manage databases.
  - It provides users and applications with an environment that is convenient and efficient to use.

What is a Database System?



Applications
(communicate with DBMS)

Database System

=

Database Management System
(Software package)

+

Database
(actual data collection)

Purpose of a Database System

- Data Integration: All data are uniformly managed.
- Efficient Data Access: Database languages are provided to store, access and manage data.
- Data Dictionary: Contains all data about objects and structures of the database (metadata)
- User/Application-Views: Different views for different users and applications
- Integrity Constraints: are enforced by the DBMS.
- Security Mechanisms: to protect data from security threats
- Transactions: combine sets of operations on data into logical units
- Synchronization: of concurrent user transactions
- Recovery: of data after system crash
- ad-hoc queries, report generation, interfaces to other database systems, interfaces for application programming, . . .

History of Relational Database Systems

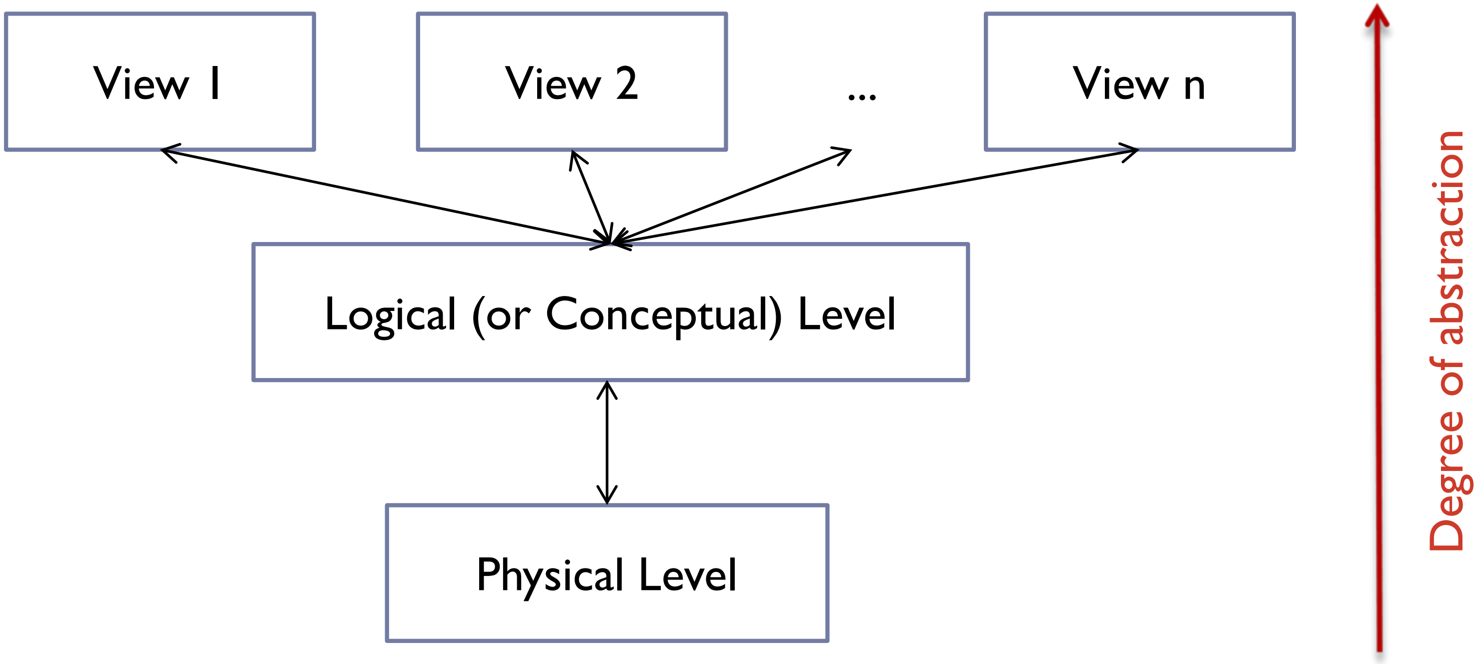
- 1970: Ted Codd (IBM) → relational model as conceptual foundation of relational DBS
- 1974: System R (IBM) → first prototype of a RDBMS
  - only two modules: RDS (optimizing SQL processor), RSS (access method); approx. 80.000 LOC (PL/1, PL/S, Assembler), approx. 1,2 MB code size
  - query language SEQUEL
  - first installation 1977
- 1975: University of California at Berkeley (UCB) → Ingres
  - query language QUEL
  - predecessor of Postgres, Sybase, . . .
- 1979: Oracle Version 2
- Today: 24 Bill. $ market

Some of the Largest Database Systems

- eBay Data Warehouse: 6,5 PB (= 6, 5 · 1015 Bytes)
  - Teradata DBMS, >100 servers, 17 trillion records, 50 TB/day
- WalMart Data Warehouse: 2,5 PB
  - Teradata DBMS, NCR MPP hardware;
  - Product information (sales etc.) from 2.900 stores; 50.000 queries/week
- Facebook 400 TB
  - Hadoop/Hive, 610 nodes, 15 TB/day
- US Library of Congress 10-20 TB
  - not fully digitized

Different Views of Data

- A major purpose of a DBMS is to provide users with an abstract view of data,
- i.e. it hides details of how data are stored and maintained on a computer.



Different Views of Data (cont.)

- Physical Level describes how data records are actually stored and how files and indexes are organized and used
- Logical Level (sometimes also called Conceptual Level) describes what data are stored in the DBS in terms of entities and relationships; emphasis on logical structure of the database
- View Level describes how users and applications see the data

- Abstraction is achieved by describing each level in terms of a database schema, which, in turn, is based on a data model

Data Models, Schemas, and Instances

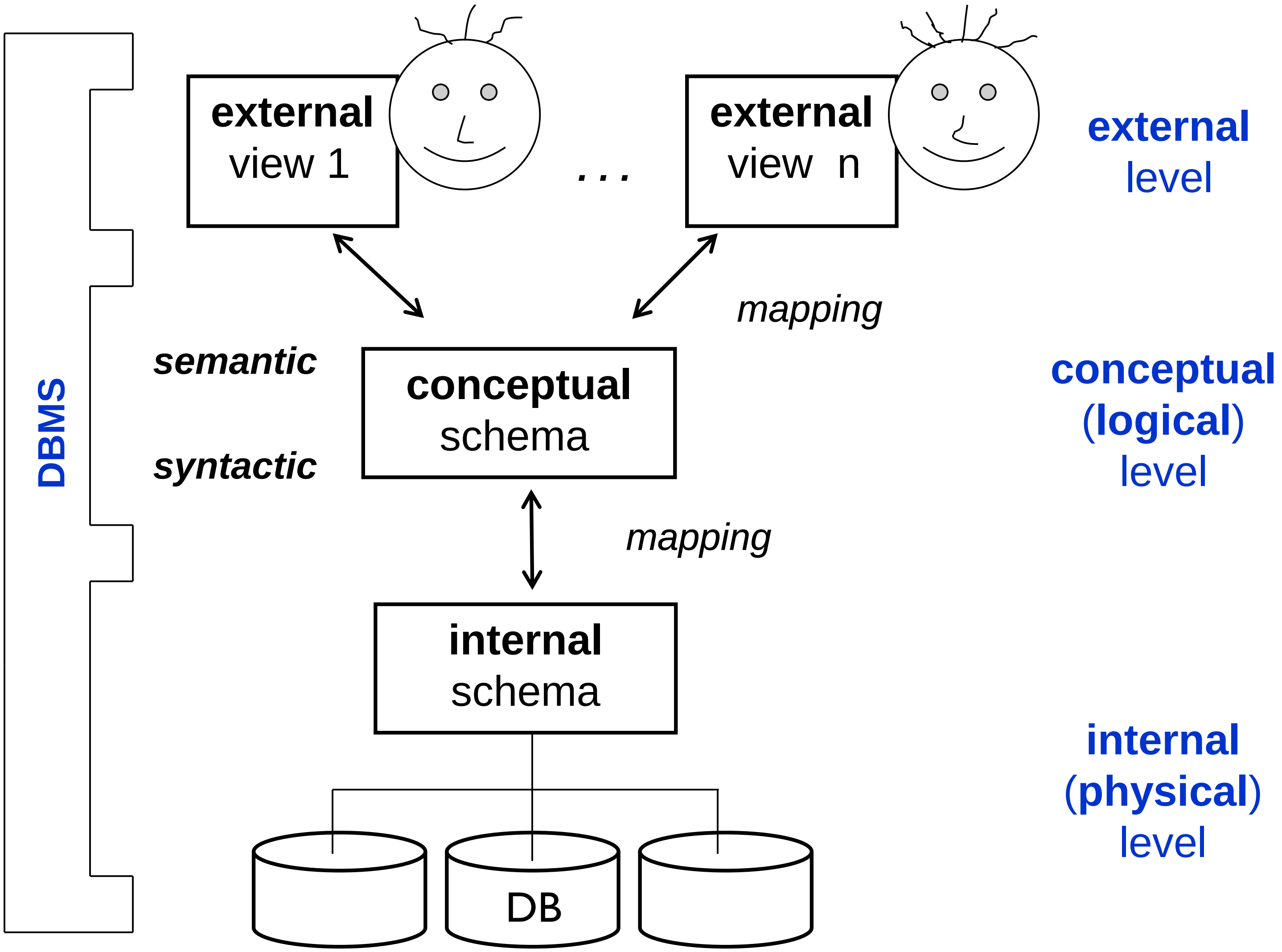
- A Data Model is a collection of concepts for describing
  - data and relationships among data
  - data semantics and data constraints

- Object-Based logical Models
  - Entity-Relationship (ER) Model
  - Object-Oriented (OO) Model
- Record-Based logical Models
  - Relational Model
  - Network Model
  - Hierarchical Model

Data Models, Schemas, and Instances (cont.)

- A database schema is a description of a particular collection of data, using a given data model.
- An instance of a database schema is the actual content of the database at a particular point in time.
- Schemas exist at different levels of abstraction
  - **Conceptual** (or Logical) Schema: typically builds the basis for designing a database (⇒ main focus in this course)
  - View (or **External**) **Schemas**: typically determined during requirements analysis (often require integration into one conceptual schema)
  - **Physical Schema**: storage structures associated with relations

Three Level Schema Architecture



Data Independence

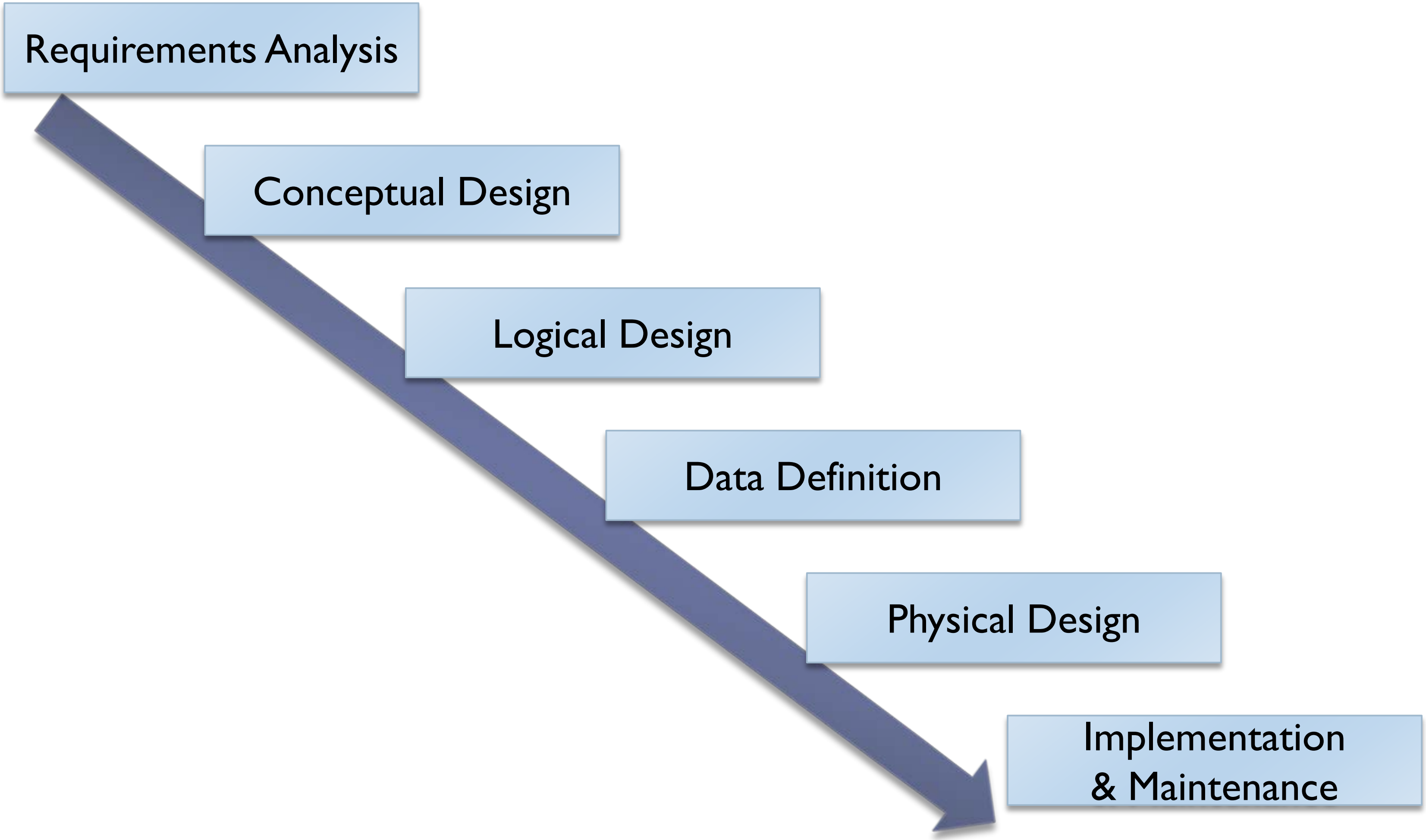
- Ability to modify definition of schema at one level without affecting a schema definition at a higher level

- Achieved through the use of three levels of data abstraction (also called three level schema architecture)
  - Logical Data Independence:
    - Ability to modify logical schema without causing application programs to be rewritten
  - Physical Data Independence:
    - Ability to modify physical schema without causing logical schema or applications to be rewritten (occasionally necessary to improve performance)

Database Languages

- A Database Management System offers two different types of languages (for the user)
  - Data Definition Language (DDL)
    - Specification language (notation) for defining a database schema; includes syntax and semantics
    - DDL compiler generates set of tables stored in the DBMS's data dictionary (contains metadata, i.e. data about data)
    - Data storage and definition language – special type of DDL in which storage structures and access methods used by the DBS are specified
  - Data Manipulation Language (DML)
    - Language for accessing and manipulating the data that is organized according to underlying data model
    - Two classes of languages
      - □ Procedural – user specifies how required data is retrieved
      - □ Declarative – user specifies what data is required without specifying how to get those data
    - (declarative ≙ non-procedural)

Database Design



Requirements Analysis

Conceptual Design

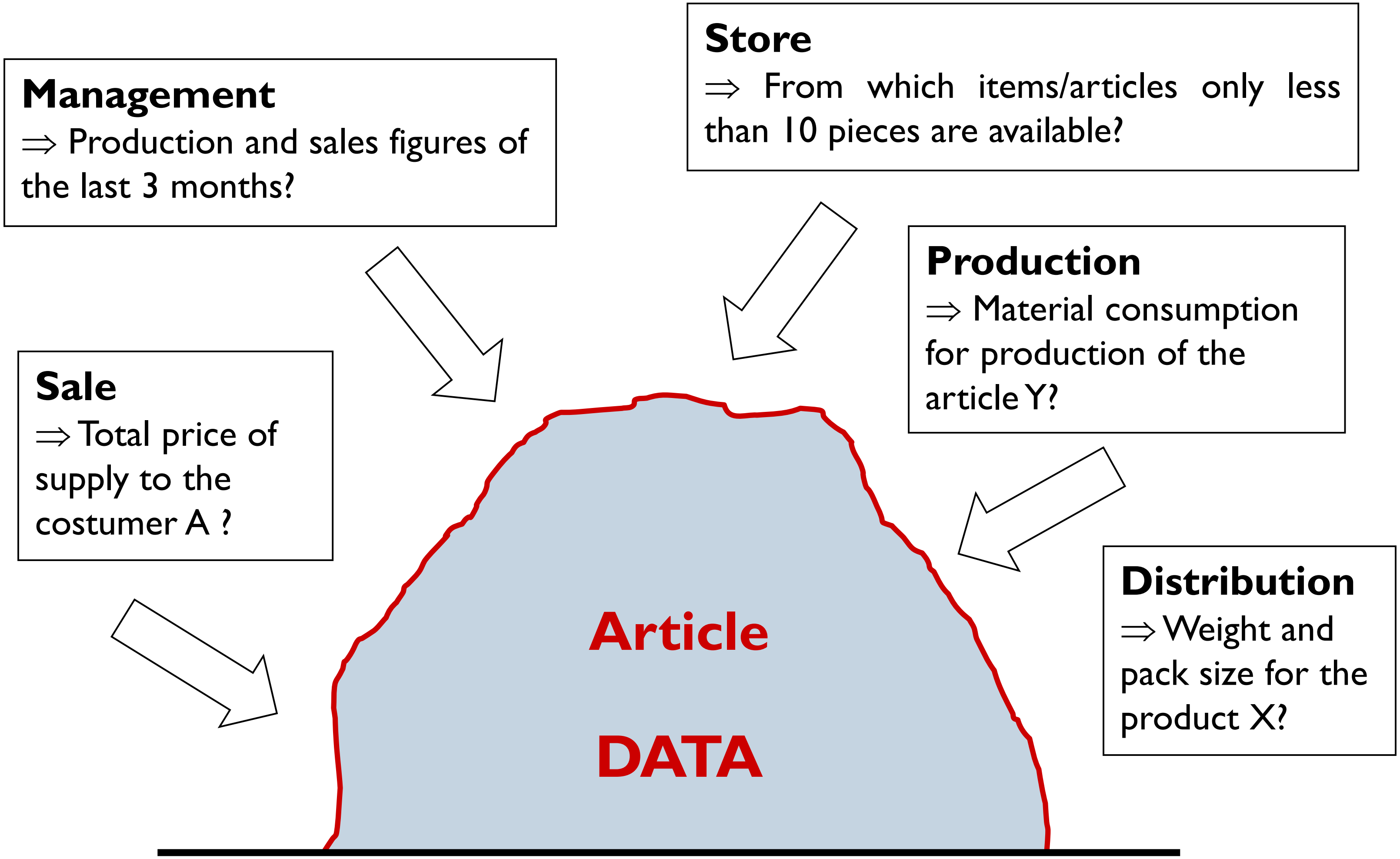
Logical Design

Data Definition

Physical Design

Implementation
& Maintenance

Design steps by example (I.)



Management
⇒ Production and sales figures of the last 3 months?

Store
⇒ From which items/articles only less than 10 pieces are available?

Production
⇒ Material consumption for production of the article Y?

Sale
⇒ Total price of supply to the costumer A ?

Distribution
⇒ Weight and pack size for the product X?

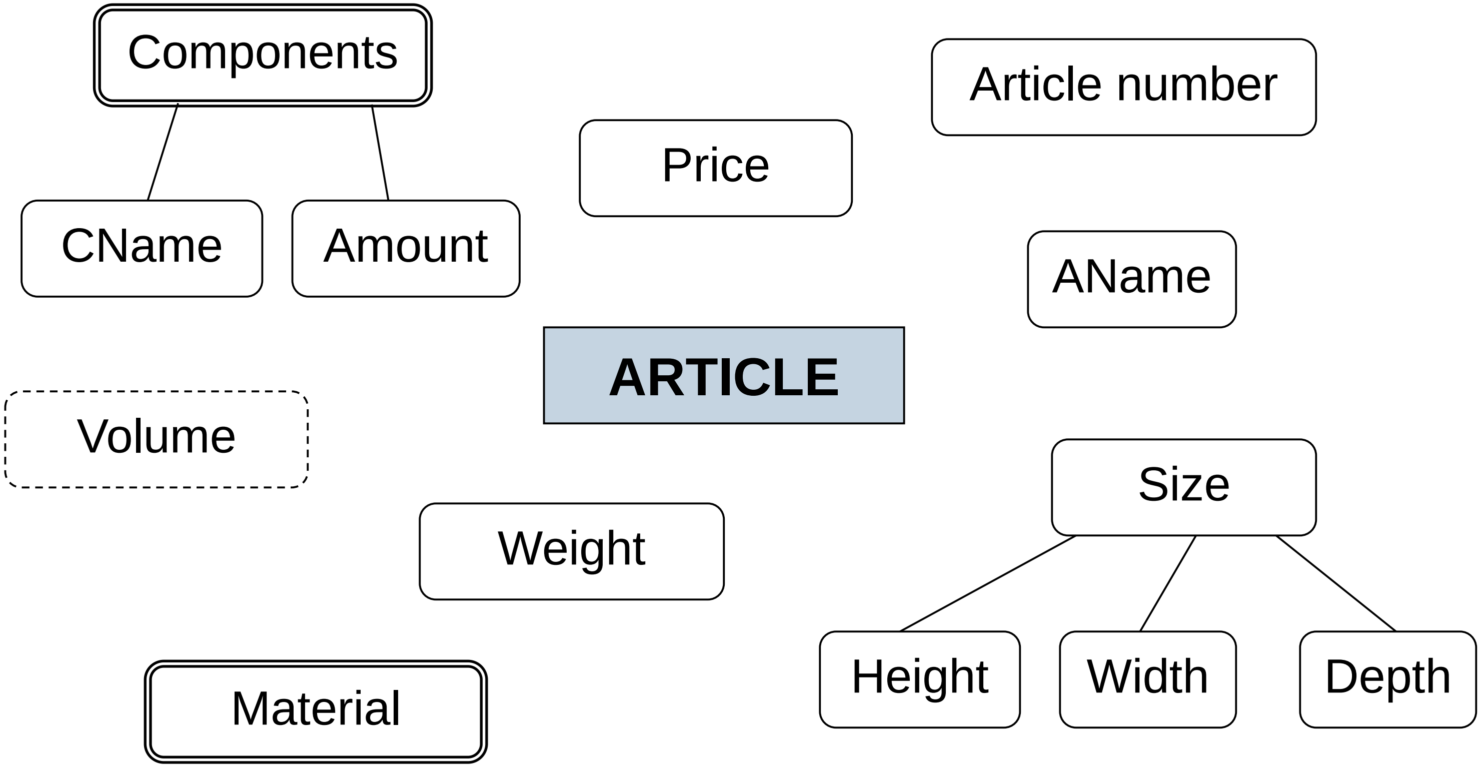
Article

DATA

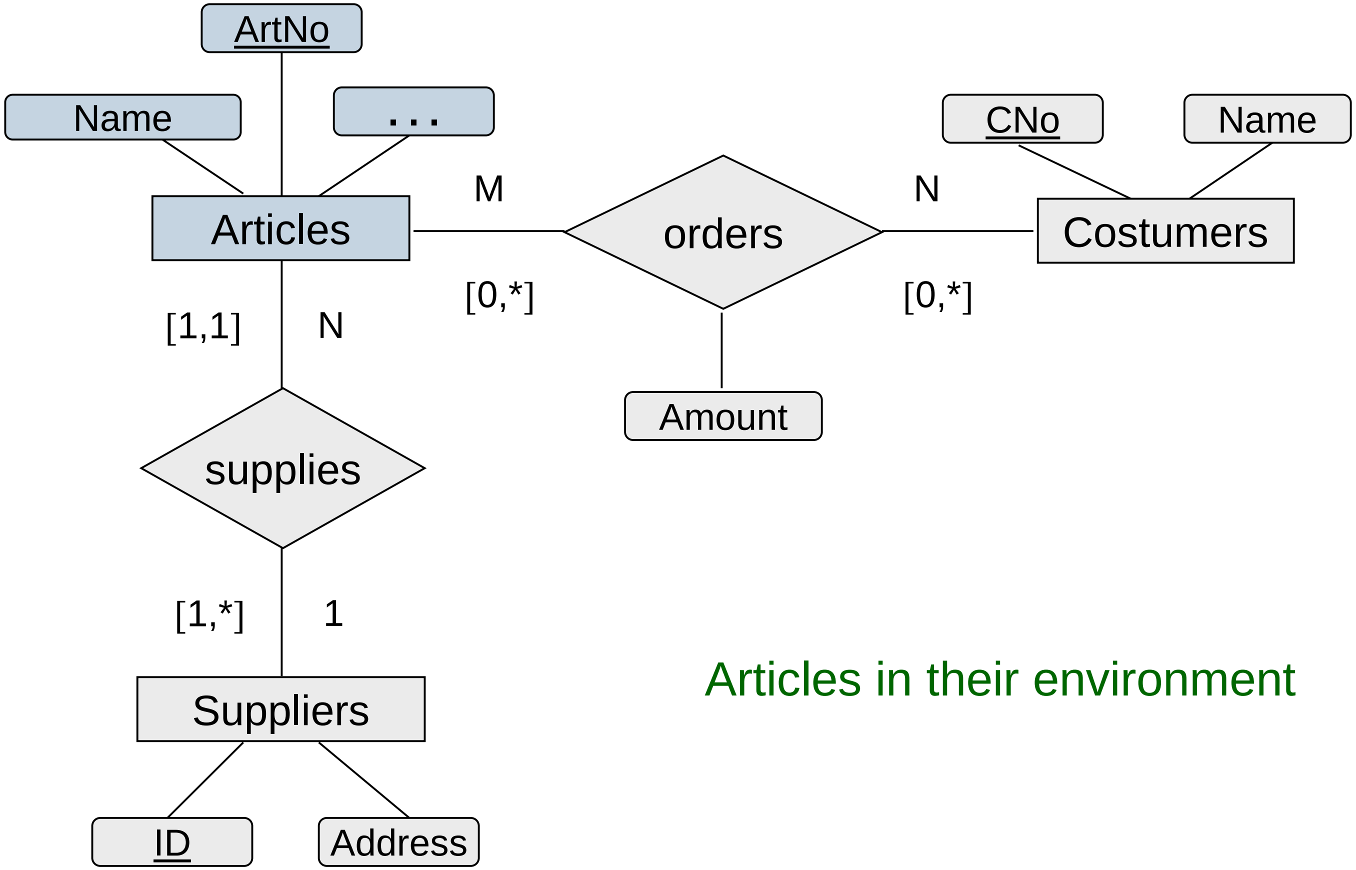
Producer of furniture

Design steps by example (2.)

Features of interest (abstraction) of the articles (Notation according to [EN02])



Design steps by example (3.)



Articles in their environment

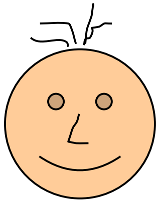
Design steps by example (4.)

Articles

ArtNo	Name	Price	. . .	Store	Quantity	. . .
1357	Table	74.90		A13	13	
2468	Bed	111.00		C06	7	
1298	Shelf	89.90		B01	56	

Orders

CNo	ArtNo	Amount
34567	1357	1
31246	2468	2
34567	1298	2



Total price for the orders of each customer?

```
SELECT  CNo , SUM ( Amount * Price )
  FROM  Articles  A ,  Orders  O
WHERE  A . ArtNo = O . ArtNo
GROUP  BY  CNo  ;
```

Exercises / Questions

- We can store (very) large collections of data in a traditional file system or in a database system. What are the distinctions?

- A database system consists of two parts. Can you characterize these?

Conceptual Database Design

- The first step is the abstract representation of the structure of a database.
- Questions that are addressed during conceptual design:
  - What are the entities and relationships of interest (mini-world)?
  - What information about entities and relationships among entities needs to be stored in the database?
  - What are the constraints (or business rules) that (must) hold for the entities and relationships?
- Note: Design is independent of all physical considerations (DBMS, OS, . . .).

Entity-Relationship Data Model

- The most common model for this phase of design is the entity-relationship model.
- Entity-Relationship model (ER model)
  - 1976 P. P. Chen
  - “The Entity-Relationship Model – Toward a Unified View of Data”
    ACM Transactions on Database Systems,
    Vol. 1, March 1976, Pages 9-36
  - Today there are many extensions of the model.

- A database schema in the ER model can be represented pictorially (Entity-Relationship diagram)

Example

The tourist industry wants to set up a supraregional data collection containing tour operators, offered journeys, and other information which may be of interest to clients.

For the journeys the destination and a short description are of importance. Moreover detailed information about the stages of a journey, i.e. concerning the places and the number of overnight stays there are meanigful.

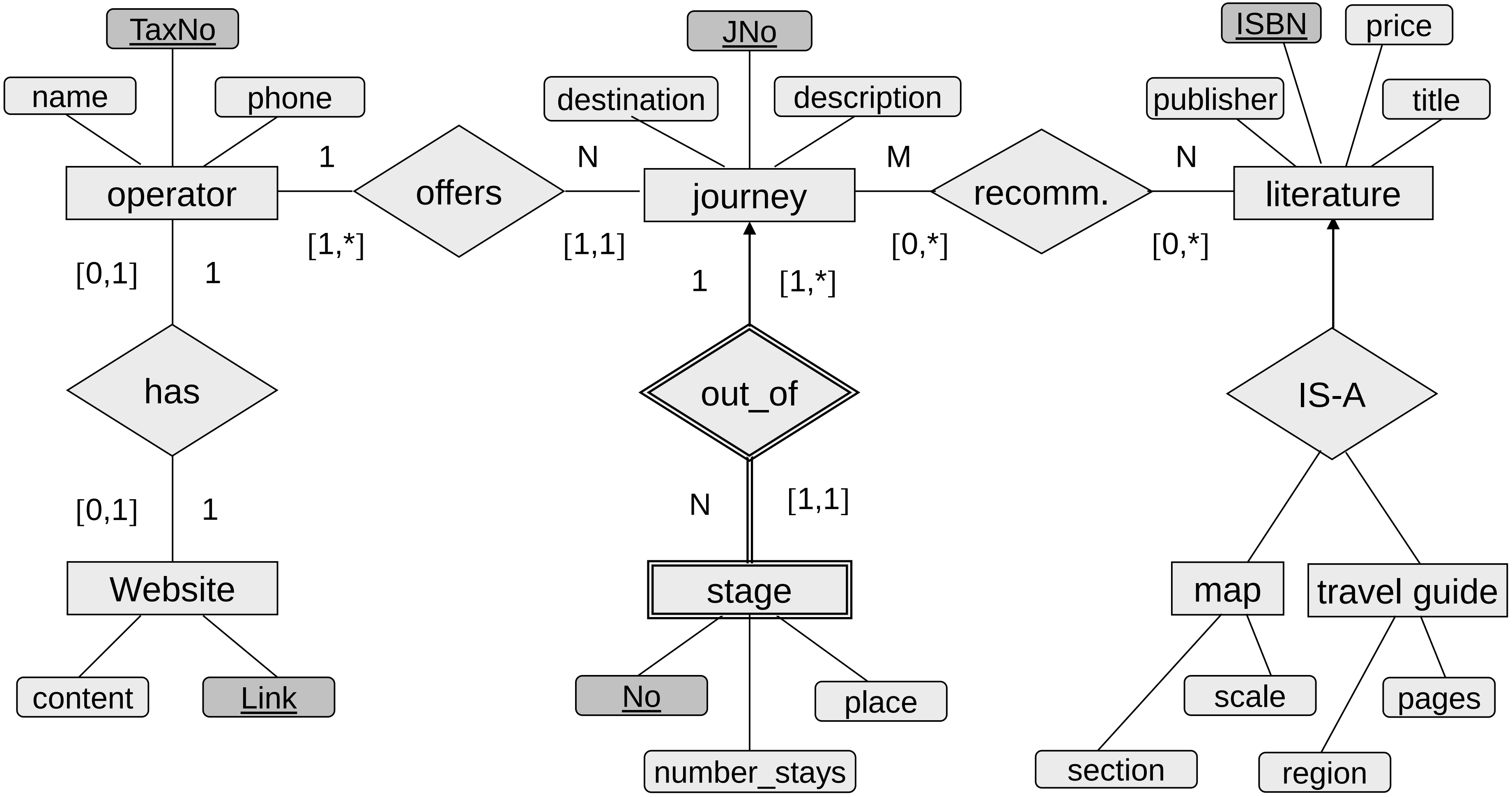
Naturally details – like the name and the telephone number – of the tour operators and their Web appearances are needed. Of interest are the link and the content itself.

Also travel literature – deviced into travel guides and maps shall be contained in the data collection. Out of these literature recommendations for the different journeys will be derived.

Concerning literature ISBN, title, publisher, and price shall be the characteristic features. For Maps still the section and the scale should be added.

Concerning travel guides the city or region described and the number of pages (hints for content and weight in the luggage) should be added.

Example


TaxNo
name
phone
operator
1
offers
[1,*]
N
[1,1]
JNo
destination
description
journey
M
recomm.
[0,*]
N
[0,*]
ISBN
price
publisher
title
literature
[0,1]
1
has
[0,1]
1
Website
content
Link
1
[1,*]
out_of
N
[1,1]
stage
No
place
number_stays
IS-A
map
travel guide
scale
pages
section
region

Entities

journey

values	properties
33557	number
Norway	destination
journey by mailboat	description

Entity e_i

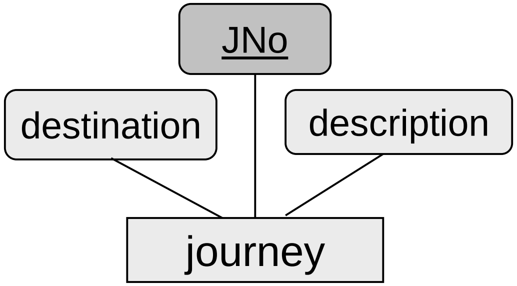
- real-world object or thing with an independent existence and which is distinguishable from other objects.
- Examples are a person, car, book, … , examen, order, …

Attributes

Attribute a_i

- An entity is represented by a set of attributes (its descriptive properties), e.g., name, age, salary, price etc.
- Attribute values that describe each entity become a major part of the data eventually stored in a database.
- Attribute Types:
  - simple (atomic) or composite
  - single-valued or multi-valued
  - stored or derived
  - each of the above can be an **optional** attribute, in case an entity may not have an applicable value for an attribute. → NULL
- With each attribute a domain is associated, i.e., a set of permitted values for an attribute.
- Possible domains are *number*, *string*, *date* etc.

Entity Types and Sets



- Entity Type E
  - Collection of entities that all have the same attributes, e.g., persons, cars, customers etc.
- Entity Set E
  - Collection of entities of a particular entity type at any point in time; entity set is typically referred to using the same name as entity type.

Key attributes

- Entities of an entity type need to be distinguishable.

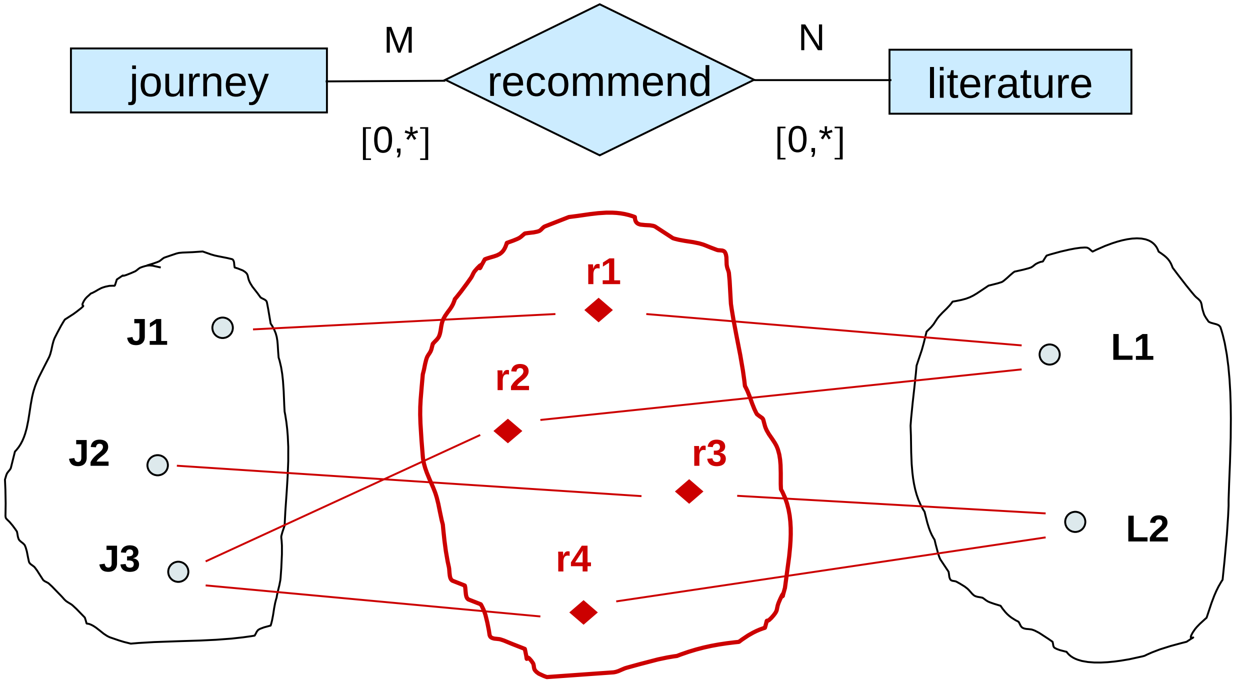
- A key of an entity type E is a set K of one or more attributes whose values uniquely determine each entity in an entity set.
  - For given any two distinct entities e_1 and e_2 in E, e_1 and e_2 cannot have identical values for each of the attributes in the key K.
  - It is possible for e_1 and e_2 to agree in some of this attributes, but never in all attributes.

- There also can be more then one possible key for an entity set. Then it is customary to pick one key as the „preferred key“, and to act as if that were the only key.
- If there are no attributes with the key property, we define an artificial key.

Relationship Types and Sets

Relationship Type R

- describes a set of similar relationships
- An n-ary relationship type R links n entity types $E_1, \ldots,$ E_n.



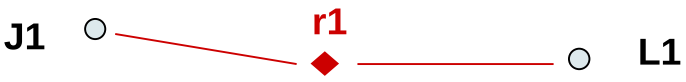
Relationships

Relationship (instance)

- association among two or more entities, e.g., "customer 'Bill' orders product 'SkyPhone' "
- Each relationship in a relationship set R of relationship type involves entities $e_1 \in E_1, \ldots, e_n \in E_n$

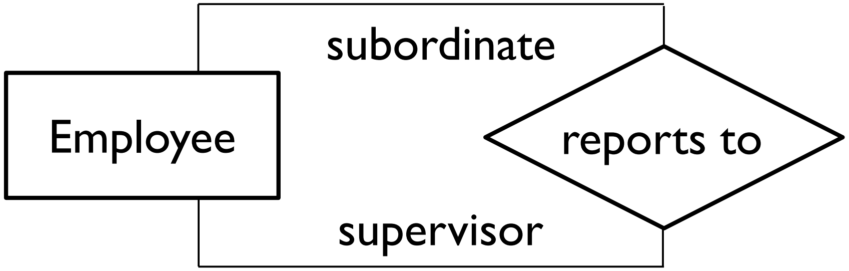
$$R \subseteq \{ (e_1, \ldots, e_n) \mid e_1 \in E_1, \ldots, e_n \in E_n \} ,$$

where $(e_1, \ldots, e_n)$ is a relationship.



Relationships, Types, Sets

- Degree of a relationship: refers to the number of entity types that participate in the relationship type (binary, ternary, . . .).
- Roles: The same entity type can participate more than once in a relationship type.

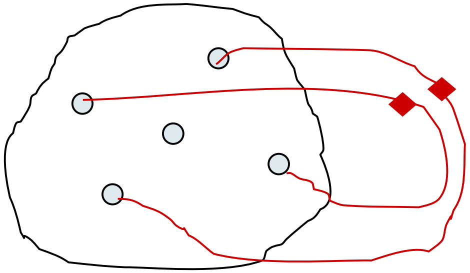


Employee

subordinate

reports to

supervisor



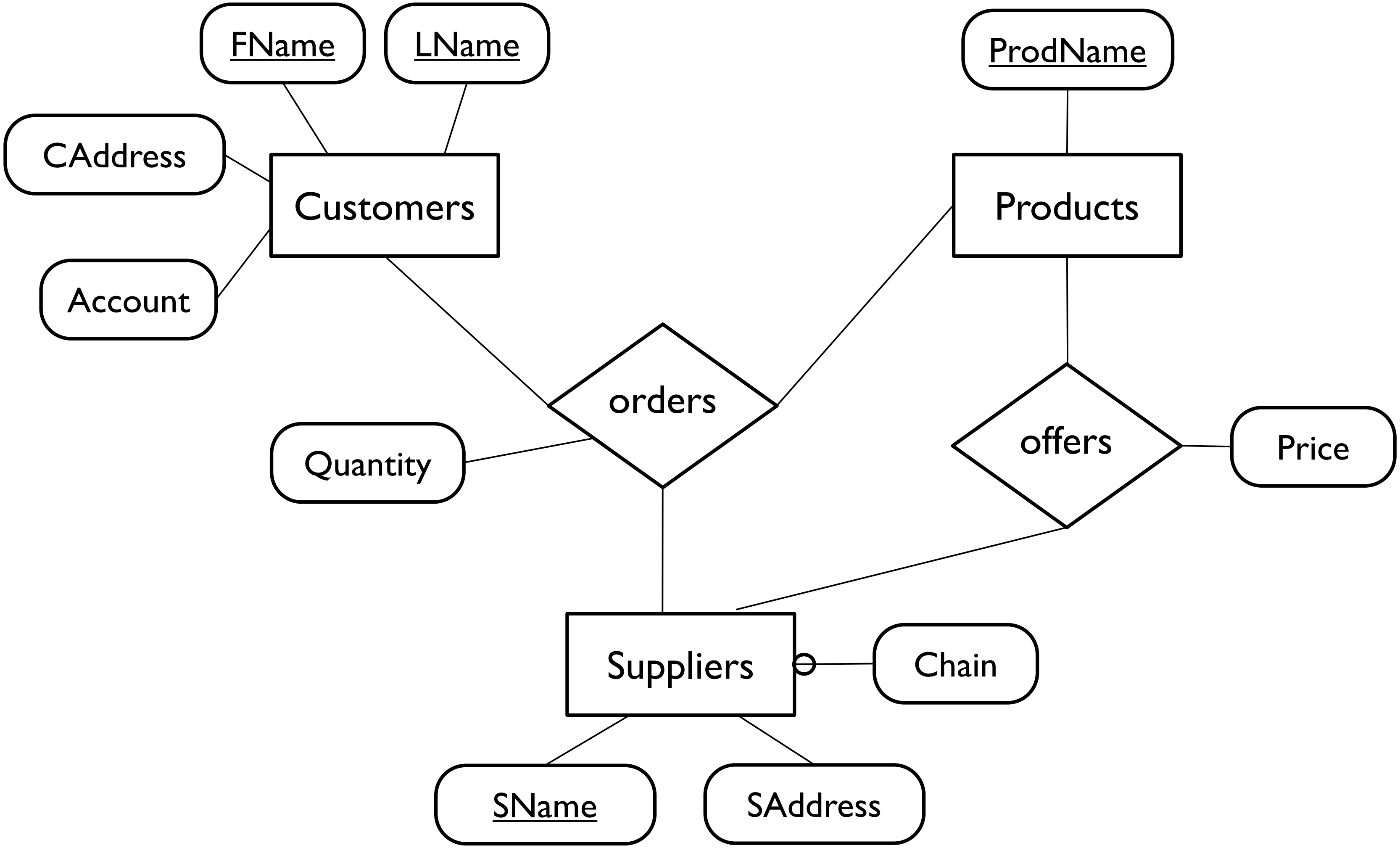
- Role labels clarify semantics of a relationship, i.e., the way in which an entity participates in a relationship. ⟼ recursive relationship.

Relationships, Types, Sets (cont.)

Relationship Attributes

- A relationship type can have attributes describing properties of a relationship.
  - "customer 'Bob' ordered product 'SkyPhone' on October 15, 2014, for $650".
  - These are attributes that cannot be associated with participating entities only, i.e., they make only sense in the context of a relationship.
- Note that a relationship does not have key attributes! The identification of a particular relationship in a relationship set occurs through the keys of participating entities.

Example of an Entity-Relationship Diagram



Example of an Entity-Relationship Diagram (cont.)

- Rectangles represent entity types
- Ellipses represent attributes
- Diamonds represent relationship types
- Lines link attributes to entity types and entity types to relationship types
- Key attributes are underlined
- Empty Circle at the end of a line linking an attribute to an entity type represents an optional (null) attribute (not mentioned in textbook)
- Not in the above diagram: Double Ellipses represent multi-valued attributes

Constraints on Relationship Types

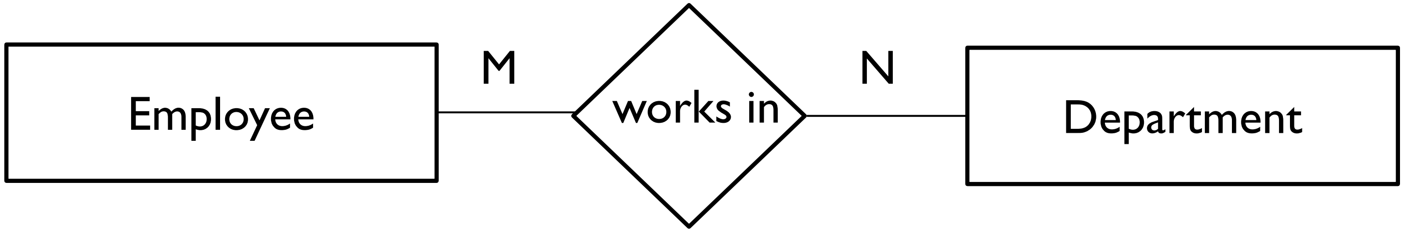
- Limit the number of possible combinations of entities that may participate in a relationship set.
- There are two types of constraints:
  - cardinality ratio and
  - participation constraints
- Very useful concept in describing binary relationship types.

For binary relationships, the cardinality ratio must be one of the following types:

Constraints on Relationship Types

For binary relationships, the cardinality ratio must be one of the following types:

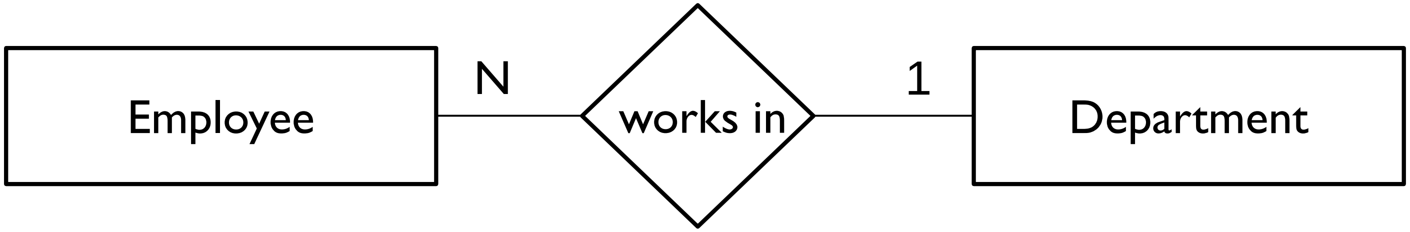
- Many-To-Many, N:M (default)



  - Meaning: An employee can work in many departments (≥ 0), and a department can have several employees (N,M ≡ any number).

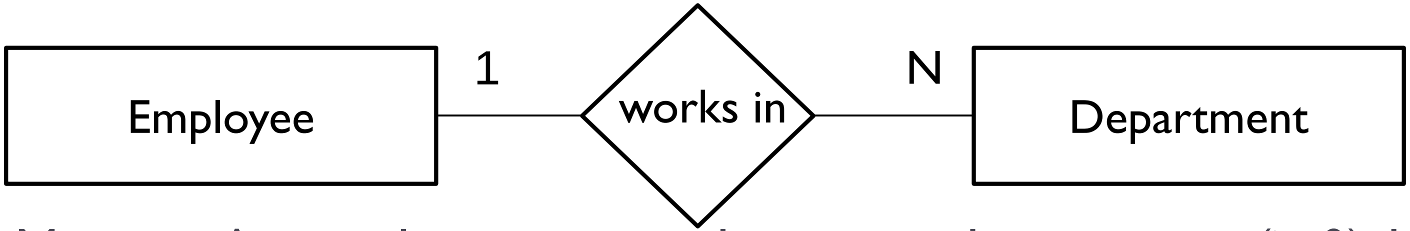
Constraints on Relationship Types (cont.)

- Many-To-One, N:1



  - Meaning: An employee can work in at most one department (≤ 1), and a department can have several employees.

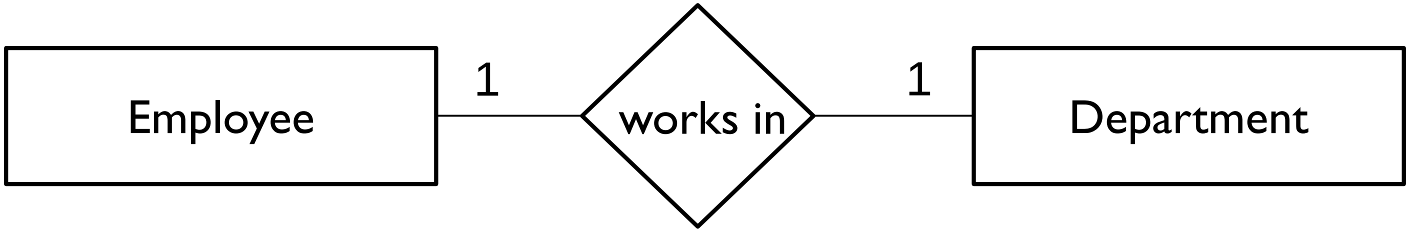
- One-To-Many, 1:N



  - Meaning: An employee can work in many departments (≥ 0), but a department can have at most one employee.

Constraints on Relationship Types (cont.)

- One-To-One, 1:1



  - Meaning: An employee can work in at most one department, and a department can have at most one employee.

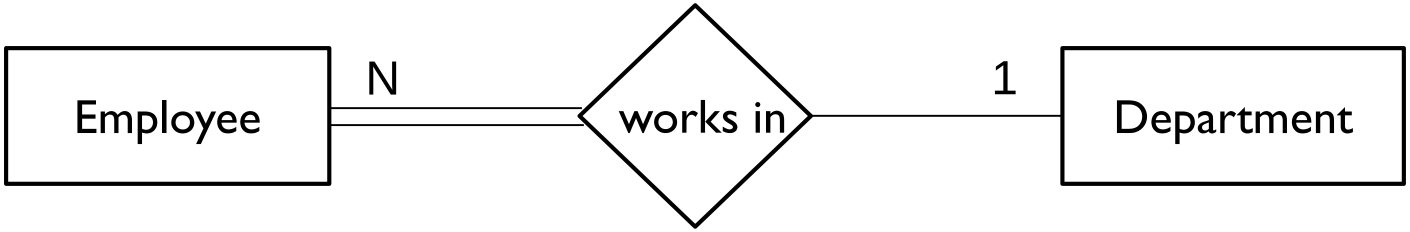
Constraints on Relationship Types (cont.)

- A 1:N relationship type (and the counterpart N:1) is also often called a functional relationship.

- Cardinality ratio of a relationship can affect the placement of a relationship attribute. E.g., in case of a N:1 relationship type, one can place a relationship attribute at a participating entity type.

Constraints on Relationship Types (cont.)

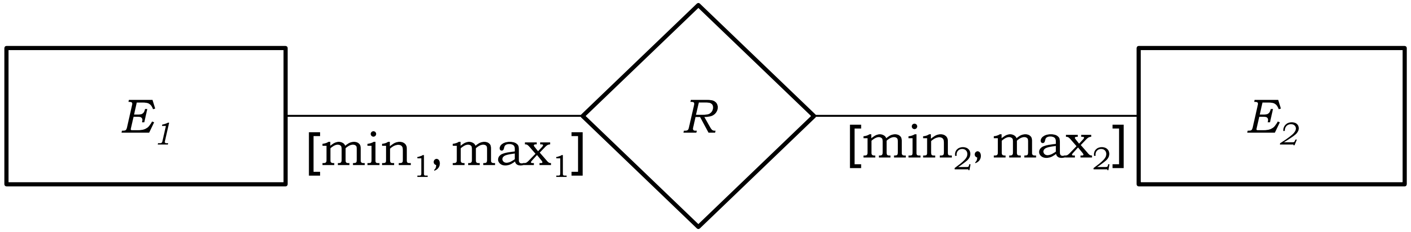
- Participation constraint: specifies whether the existence of an entity $e \in E$ depends on being related to another entity via the relationship type R.
  - total: each entity $e \in E$ must participate in a relationship, it cannot exist without that participation (total participation aka existence dependency).



  - partial: default; each entity $e \in E$ can participate in a relationship

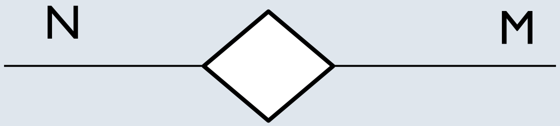
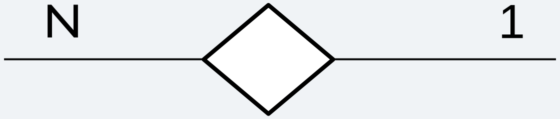
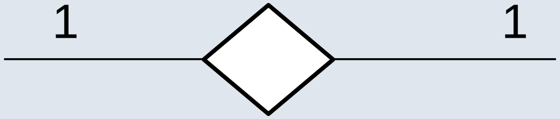
Constraints on Relationship Types (cont.)

- Instead of a cardinality ratio or participation constraint, more precise cardinalities can be associated with relationship types:


E_1
$[\mathrm{min}_1, \mathrm{max}_1]$
R
$[\mathrm{min}_2, \mathrm{max}_2]$
E_2

- Each entity $e_1 \in E_1$ must participate in relationship set R at least min_1 and at most max_1 times (analogous for $e_2 \in E_2$).

Frequently used cardinalities

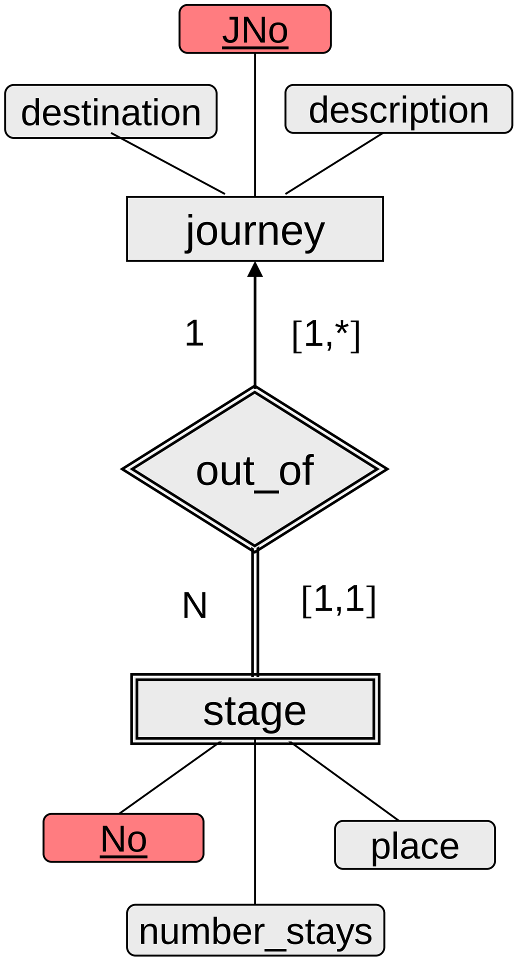
Relationship	$[min_1, max_1]$	$[min_2, max_2]$	pictorial notation
N:M	[0,*]	[0,*]	
N:1	[0,1]	[0,*]	
1:1	[0,1]	[0,1]	

Weak Entitity Sets

- Entity type, at whose identification (key) is involved a relationship (N:1) to a owner entity type
- Existence of the weak entities depends from the existence of their owner entity

- partial key
  - attribute(s) for identifikation of the weak entities relating to **one** owner entity
- key
  - partial key of the weak entities plus key of the owner entity set

Example



JNo

destination

description

journey

1

[1,*]

out_of

N

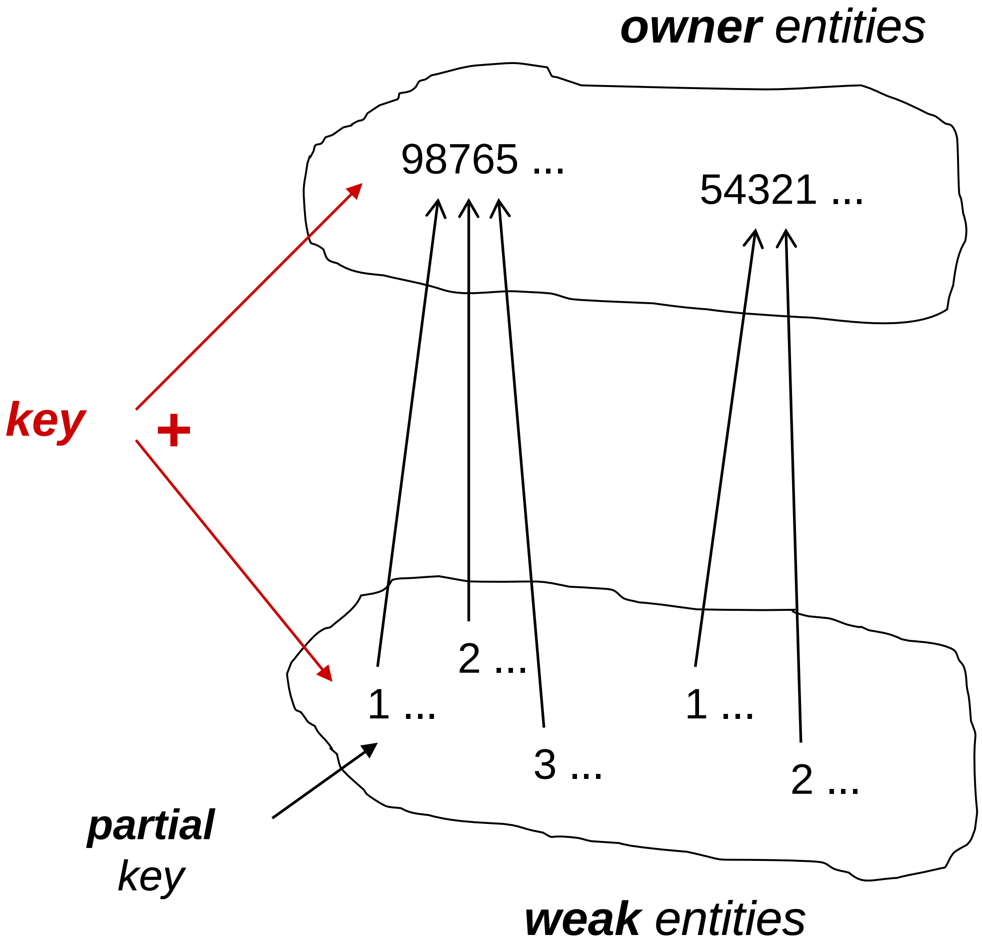
[1,1]

stage

No

place

number_stays



owner entities

98765 ...

54321 ...

key

+

1 ...

2 ...

3 ...

1 ...

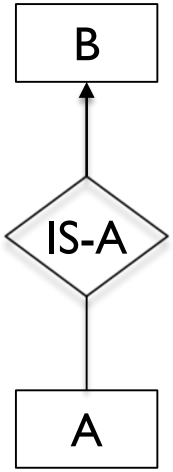
2 ...

partial
key

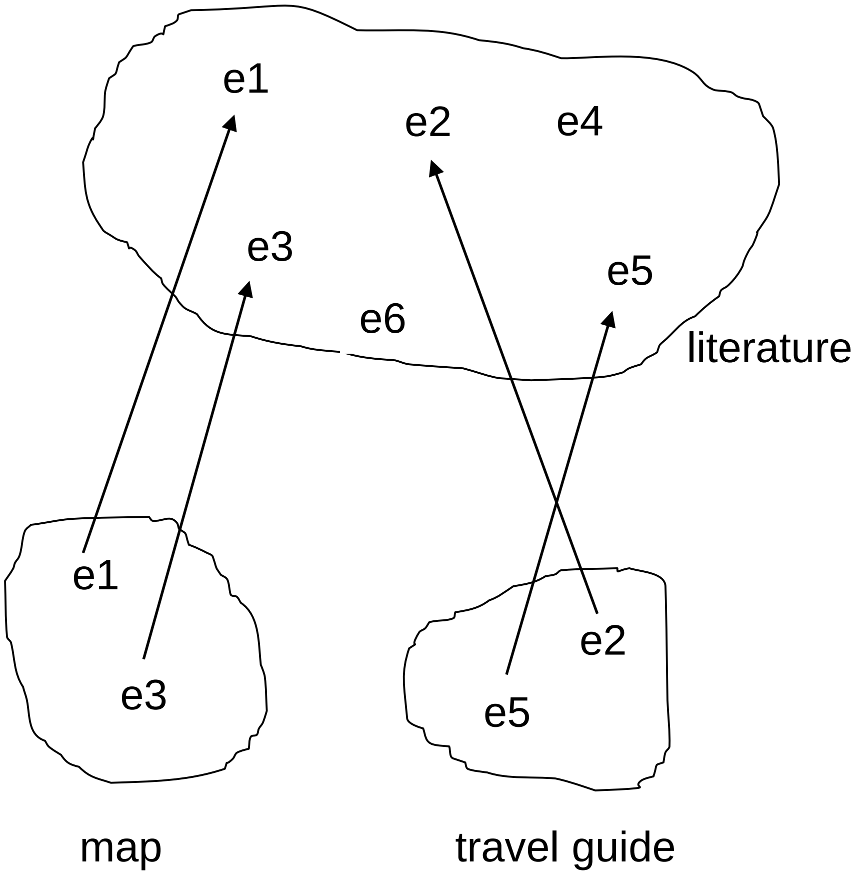
weak entities

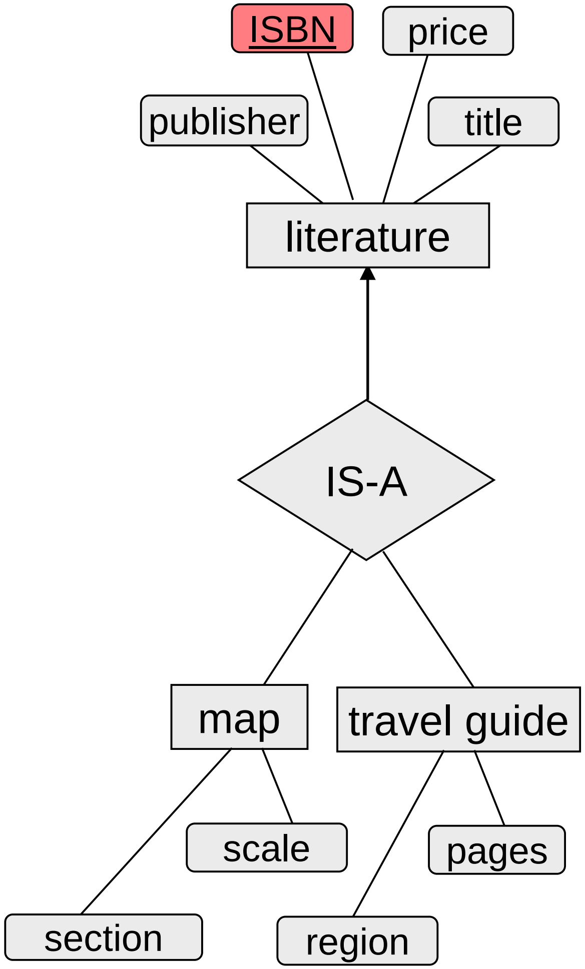
IS-A Hierarchies

- Two entity sets A and B are in the relationship IS-A, if entity set B is a generalisation of entity set A, or equivalently, A is a special kind of B.
  - Entity set A is a subset of entity set B.
  - A inherits the attributes of B, but also has additional attributes that don‘t make sense for those members of B that are not also members of A.
  - Technically, each entity a in set A is related to exactly one entity b in set B, such that a and b are really the same entity.
  - The key attributes for entity set A are actually attributes of entity set B. (A inherit the from B.)

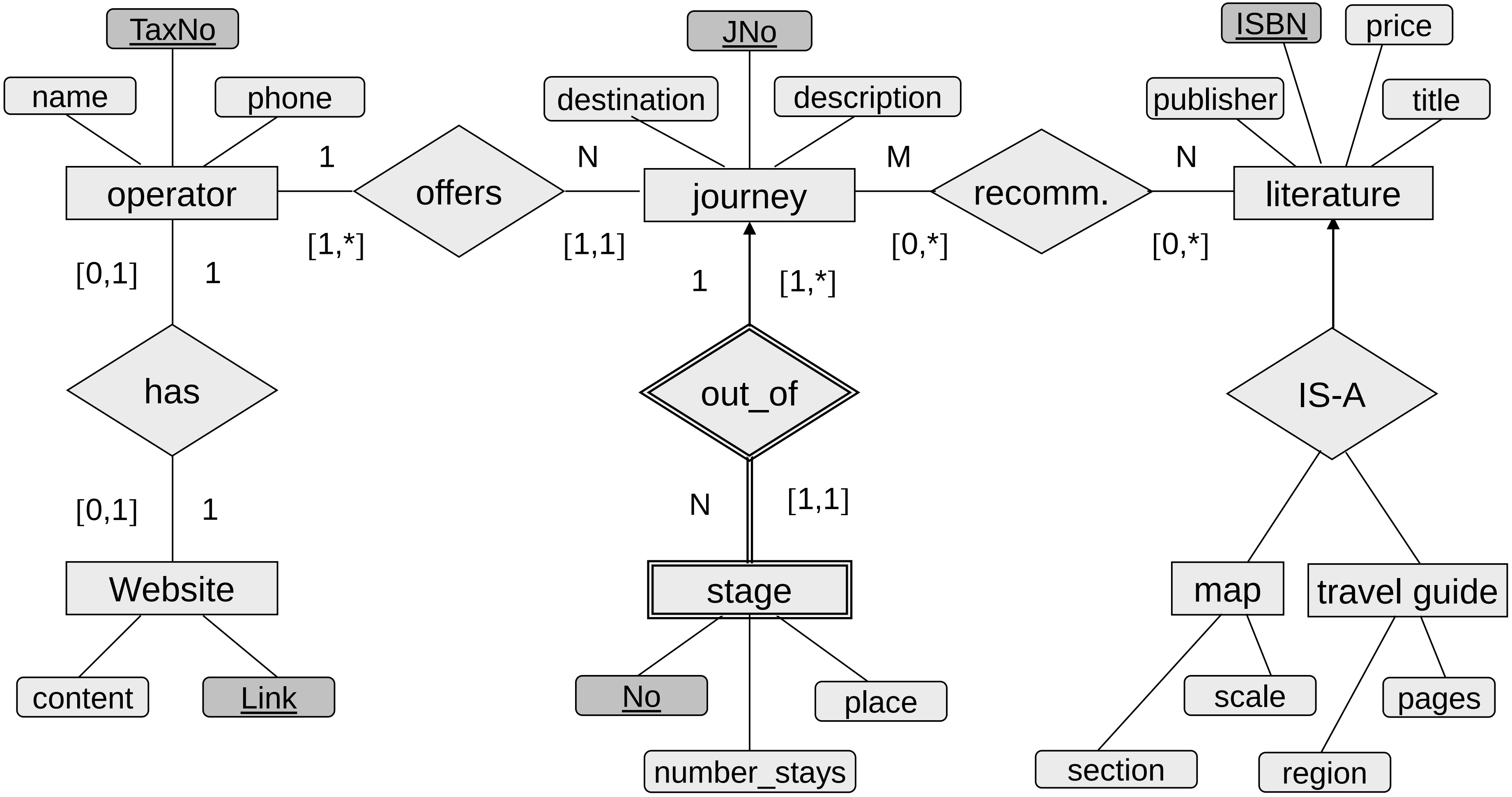


Example


e1
e2
e4
e3
e5
e6
literature
e1
e3
e2
e5
map
travel guide


ISBN
price
publisher
title
literature
IS-A
map
travel guide
scale
pages
section
region

Example

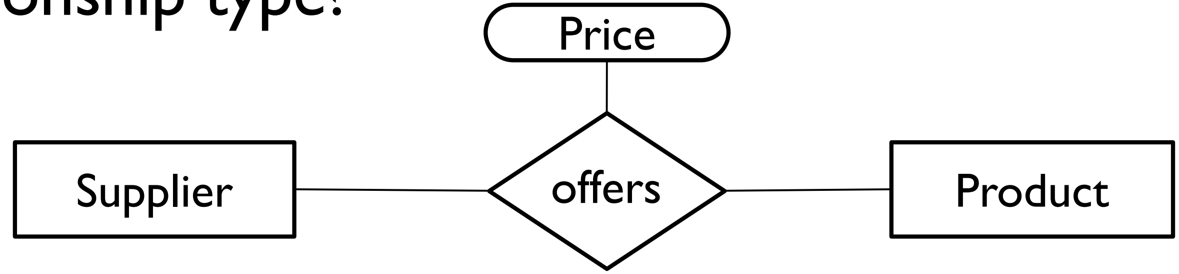

TaxNo
name
phone
operator
1
offers
[1,*]
N
[1,1]
JNo
destination
description
journey
M
recomm.
[0,*]
N
[0,*]
ISBN
price
publisher
title
literature
[0,1]
1
has
[0,1]
1
Website
content
Link
1
[1,*]
out_of
N
[1,1]
stage
No
place
number_stays
IS-A
map
travel guide
scale
pages
section
region

Design Choices for ER Conceptual Design

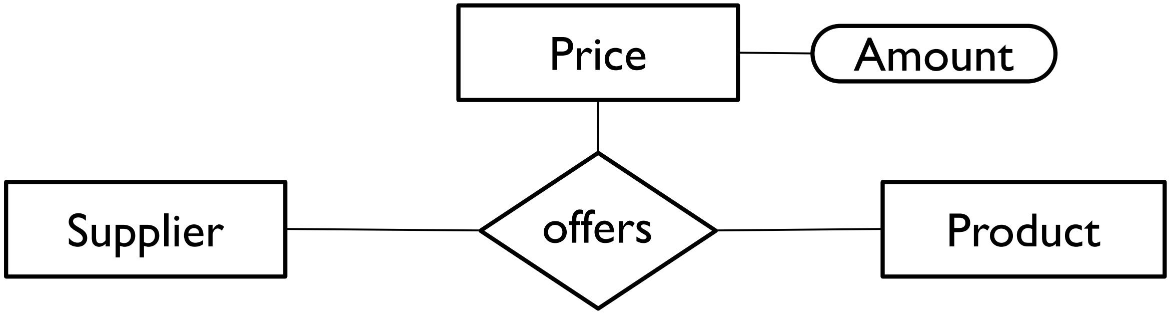
- It is possible to define entities and their relationships in a number of different ways (in the same model!).
  - Should a real-world concept be modeled as an entity type, attribute, or relationship type?
  - Is "Address" an attribute or an entity type? Decision depends upon the use one wants to make of address information. If one is interested in the structure, e.g., (City, Street, Zip-Code), Address must be modeled as an entity type or complex attribute.

Design Choices for ER Conceptual Design

- Should a concept be modeled as an entity type or relationship type?



  - Here a supplier cannot offer the same product for different prices! Why?
  - Modeling price as an entity type resolves this problem:



Steps in Designing an Entity-Relationship Schema

1. Identify entity types (entity type vs. attribute)
2. Identify relationship types
3. Identify and associate attributes with entity and relationship types
4. Determine attribute domains
5. Determine key attributes for entity types
6. Associate (refined) cardinality ratio(s) with relationship types

Exercises (1)

In the following table, E1 and E2 stand for entity types and R stands for a relationship type between two respective entity types. Characterize each relationship type by determining:

1. the cardinality ratio
2. the participation constraints for entities of E1 in R
3. the participation constraints for entities of E2 in R

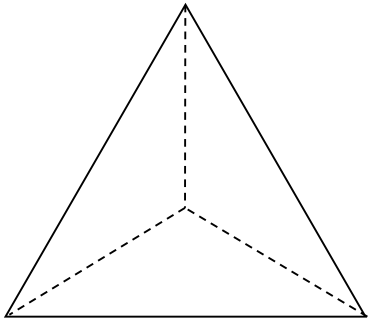
Discuss possible semantic alternatives (what might this relationship mean?) as well as necessary additional semantic information (what else do we need to know?).

Model these relationship types in ER diagrams and add attributes where necessary.

E1	R	E2	Comments
Manufacturers	supply	Parts	*Price* is specified
Employees	works_in	Rooms	No more than 3 in one room
Students	to-be-friends	Students	with attribut *since*
Students	are	Persons	
Supermarkets	have	Branches	(Chain stores)

Exercises (2)

A ***Polyhedron*** is defined by the hull of its bounding ***Faces.*** These faces are defined by their bounding ***Edges***. An edge is defined by its start and end **Points** in three-dimensional space. (You can imagine a tetrahedron as the simplest form of polyhedrons.)

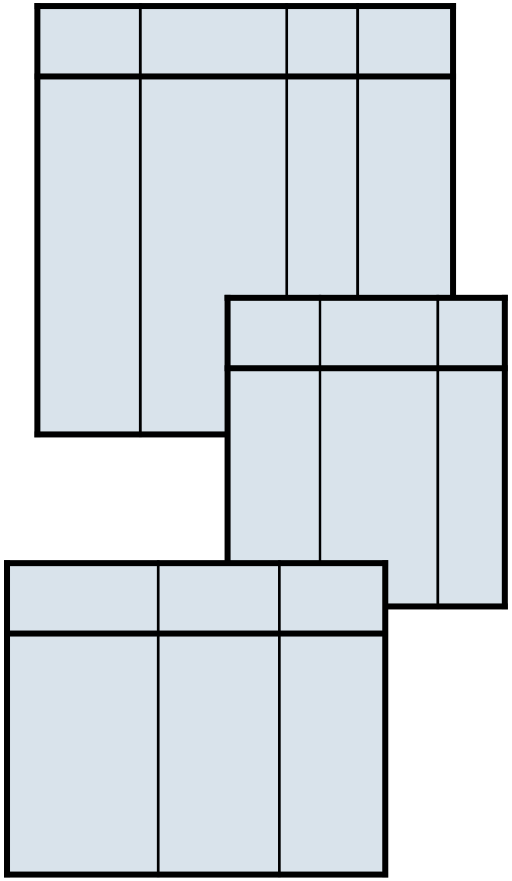


Design a suitable ER-diagram and specify the respective functionalities and cardinalities. (Attributes can be ignored.)

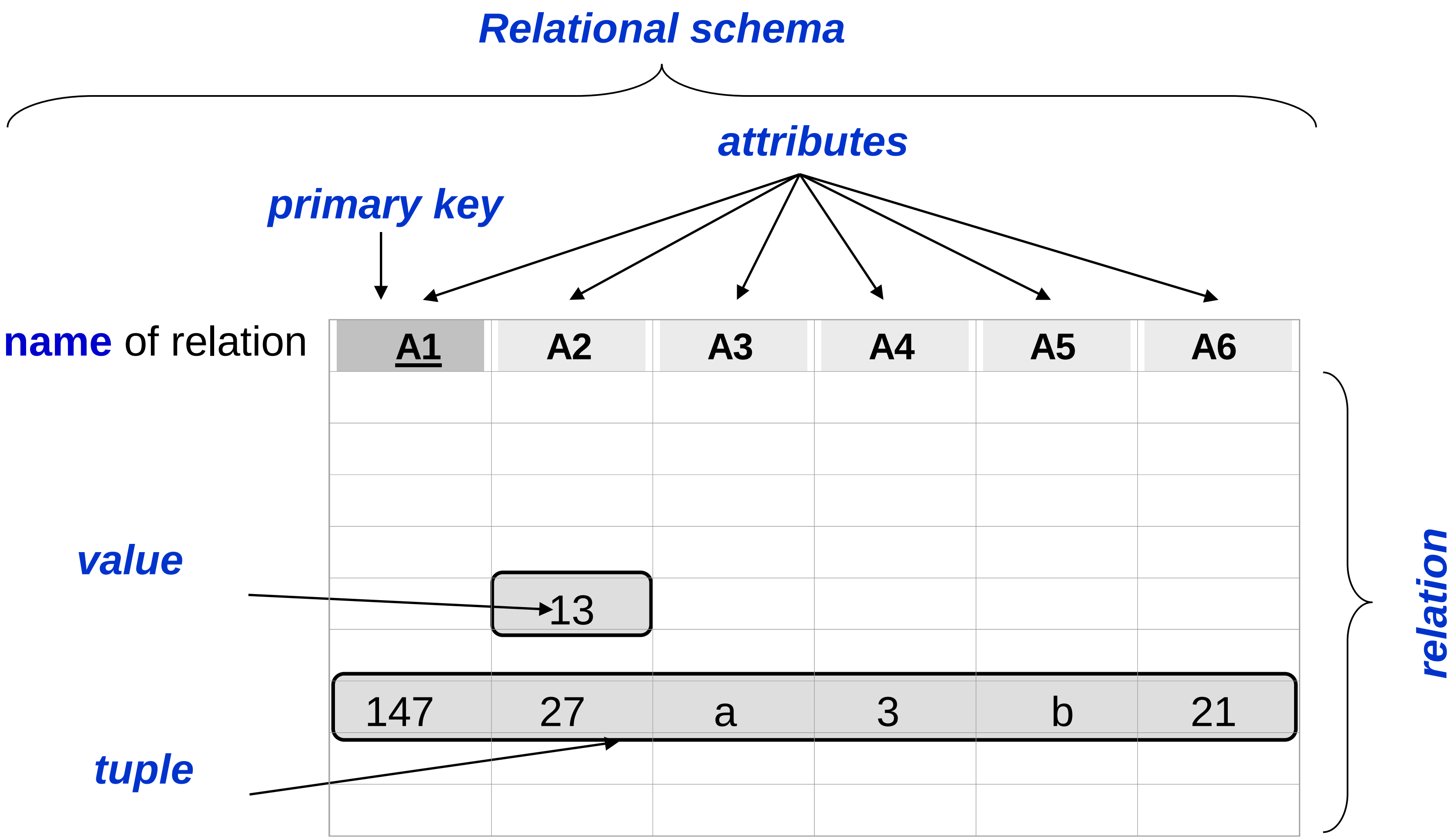
The relational data model

- Logical data model
- E. F. Codd (IBM) – 1970
- Represent data as two-dimensional tables called relations.
- Each row (tuple) of a table describes with the values in the columns

  an entity or a relationship



Relations



<u>A1</u>	A2	A3	A4	A5	A6
	13				
147	27	a	3	b	21

Basics of the Relational Model

Domains D_i

- set of permitted (atomic) values for an attribute A_i

Schema $R (A_1, A_2, \ldots, A_n)$

- descriptive attributes
  - R – name of relation
  - n – degree of relation (count of attributes)

Relation $r (R)$

- finite set of n-tuples $r = \{t_1, t_2, \ldots, t_m\}$
  - m – number of tuples

n-tuple t

- list (ordered set) of n values from the domains corresponding to attributes of the schema respectively
- $t_k = < v_1, v_2, \ldots, v_n >$ with $v_i \in D_i$, $i = 1, \ldots, n$, $k = 1, \ldots, m$

Keys

Key

- a set of attributes, whose values can identify the tuples (rows), but no subset has this property → a key must be minimal
- For a relation several candidate keys may exist.

Primary Key (PK)

- If there are more then one candidate keys, one of those is selected to be the primary key of the relation.

Foreign Key (FK)

- An attribut or attributes of one relation, referencing some key of a second („foreign“) relation
- Values of the foreign key appearing in the first relation must also appear in the referenced attributes of the second relation.

Translation of ER Schema into Tables

- An ER schema can be represented by a collection of tables which represent contents of the database (instance).
- Primary keys allow entity types and relationship types to be expressed uniformly as tables.
- For each entity and relationship type, a unique table can be derived which is assigned the name of the corresponding entity or relationship type.
- Each table has a number of columns that correspond to the (atomic) attributes and which have unique names. An attribute of a table has the same domain as the attribute in the ER schema.

- Translating an ER schema into a collection of tables is the basis for deriving a relational database schema from an ER diagram.

Translating Entity Types into Tables

- Given an entity type E_1 with (atomic) attributes $A_1, \ldots, A_n$ and associated domains $D_1, \ldots, D_n$.
- Translation of the entity type CUSTOMER into a table

FName	LName	CAddress	Account
Bob	Smith	Ilmenau	5,000
Michael	Gordon	Erfurt	2,000
…	…	…	…

```
create table Customers (
  FName varchar(40), LName varchar(40),
  CAddress varchar(100), Account real);
```

- A row in such a table corresponds to an entity from the entity set.

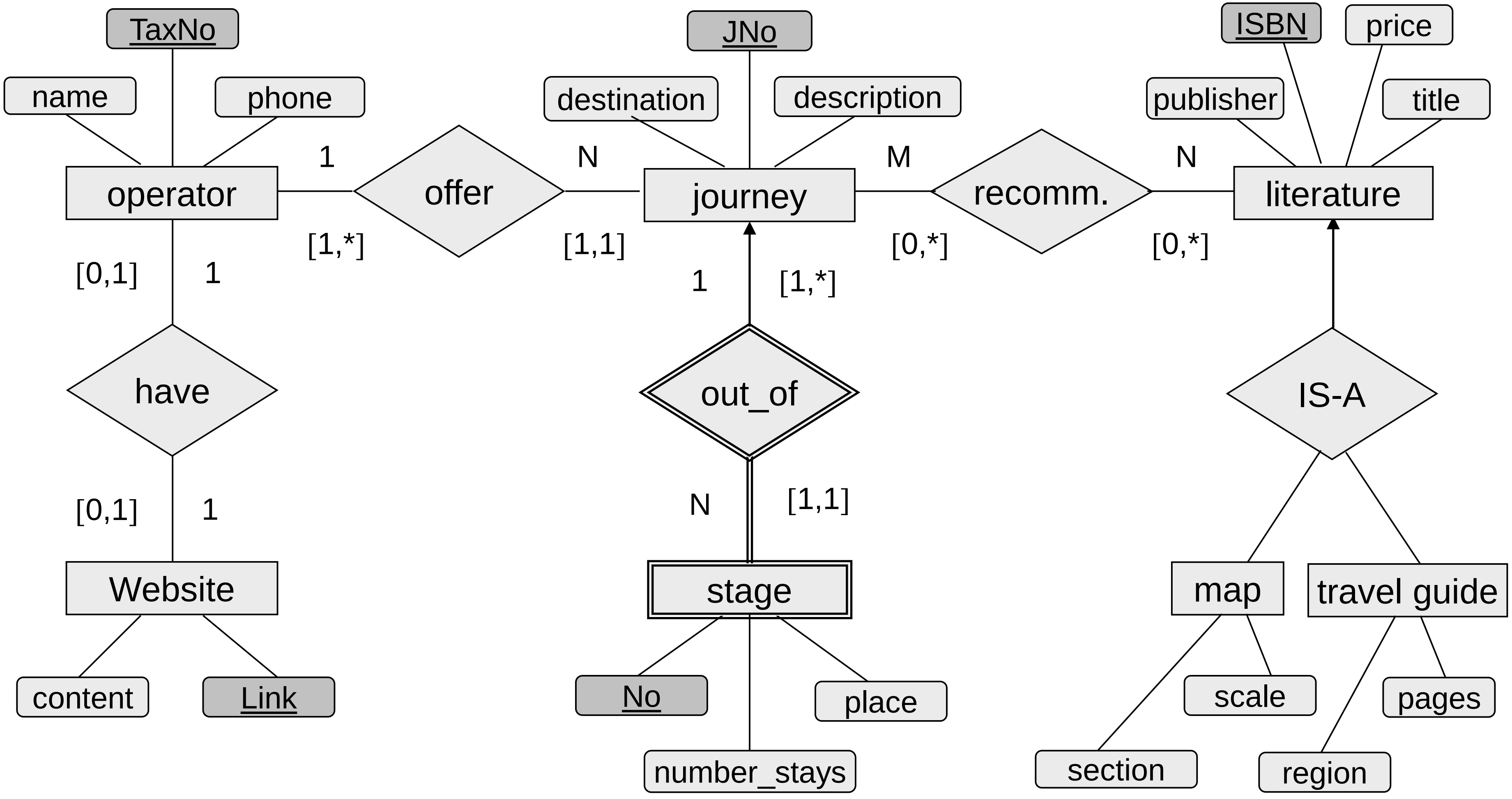
Translating Relationship Types into Tables

- A M:N relationship type is represented as a table with columns for the primary key attributes of the participating entity types, and any descriptive attributes of the relationship type.
- Example: Relationship type offers

ProdName	SName	Price
SkyPhone	Hal-Mart	650
Super Tablet	K+B	1,200
…	…	…

- ProdName and SName are the primary key attributes of the entity types PRODUCTS and SUPPLIERS respectively.
- Translation of 1:N and N:1 (functional) and 1:1 relationship types into tables sometimes can be optimized ⟼ no table for relationship type necessary!

Example Tourism



Example: entity types

Entity Type E

⇓

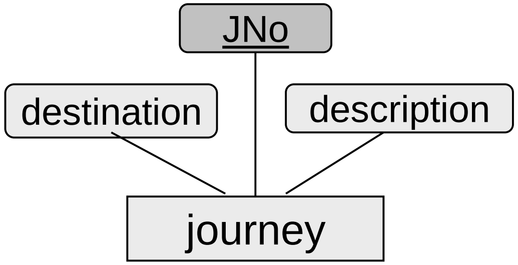
Relation

- Attributes

  attributes from E

- Primary key

  (one) minimal key from E



journey

JNo	destination	description
33557	Norway	journey by mailboat
72345	South Tyrol	hiking
12399	Verona	trip with visit an opera

Example

operator

TaxNo	name	phone

Website

Link	content

literature

ISBN	title	publisher	price

Example: relationship types

Relationship Type R

$\Downarrow$

Relation (in addition to the relations for the entity types)

- Attributes

  primary key attributes of the participating entity types (in function of foreign keys)

  plus

  attributes of the relationship itself

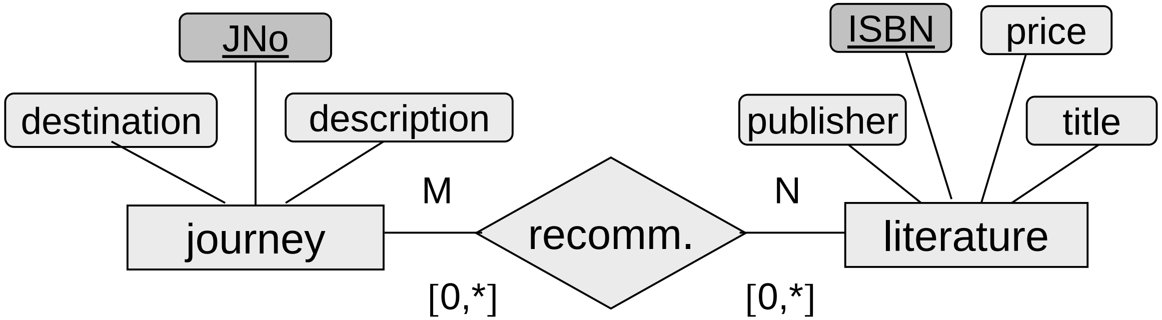
- Primary key

  M : N $\Rightarrow$ common primary key consisting of primary keys of both sides

  1 : N $\Rightarrow$ primary key of the N-Side

  1 : 1 $\Rightarrow$ one primary key (both are candidate keys !)

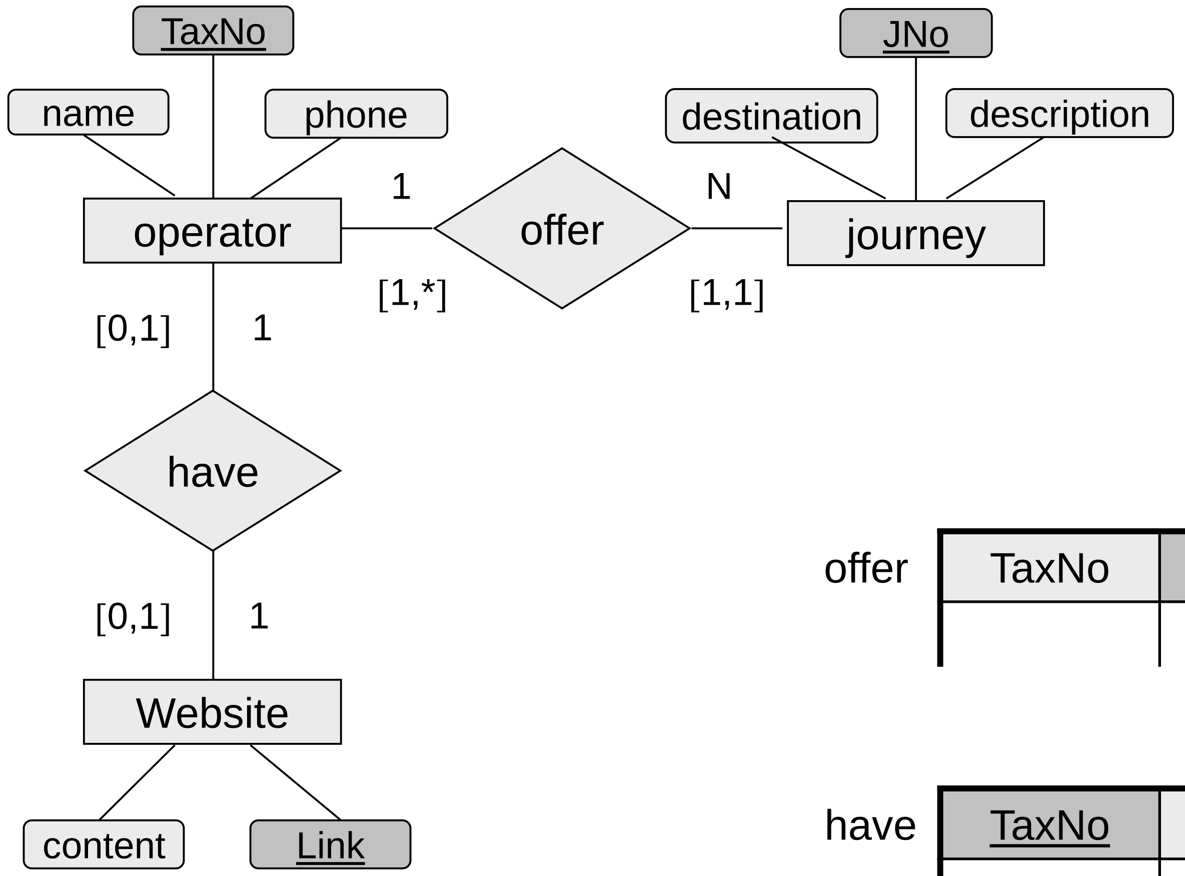
Example



recommend

JNo	ISBN

Example



offer

TaxNo	JNo

have

TaxNo	Link

Example: specialized entity types

Specialized entity type

⇓

Relation (for specialized entity type and IS-A relationship)

- Attributes

  attributes of specialized entity type

  plus key attribute(s) of the supertype

- Primary key

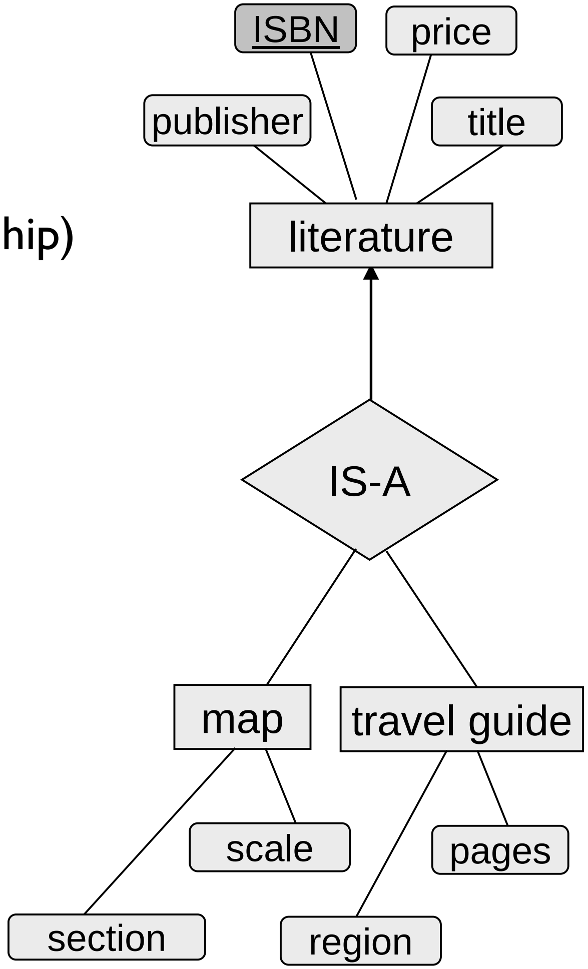
  primary key of the supertype

map

ISBN	section	scale

travel guide

ISBN	region	pages



Example: weak entity types

Weak entity type W

⇓

Relation (for weak entity type and its relationship to owner)

- Attributes

  attributes of the weak entity type

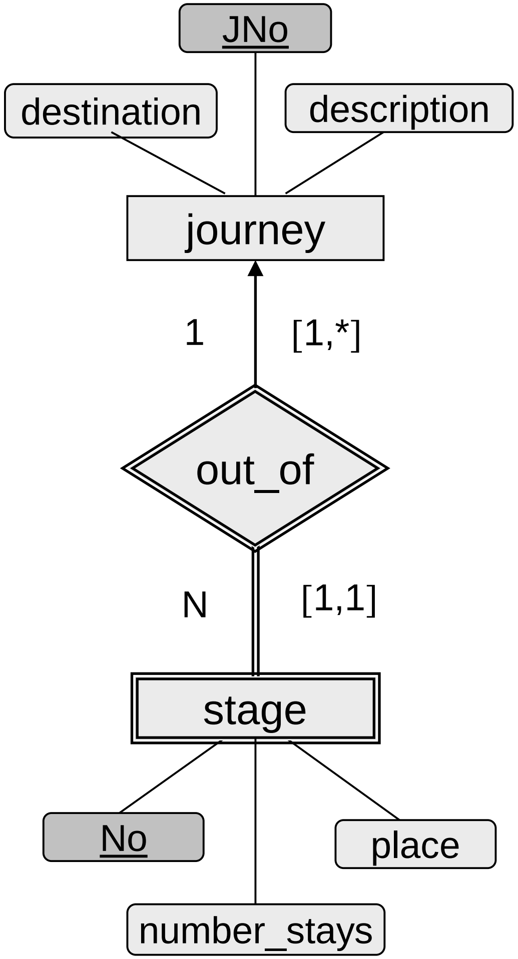
  plus

  primary key attribute(s) of the owner entity type

- Primary key

  common primary key consisting of partial key and the owner‘s primary key

Example



journey

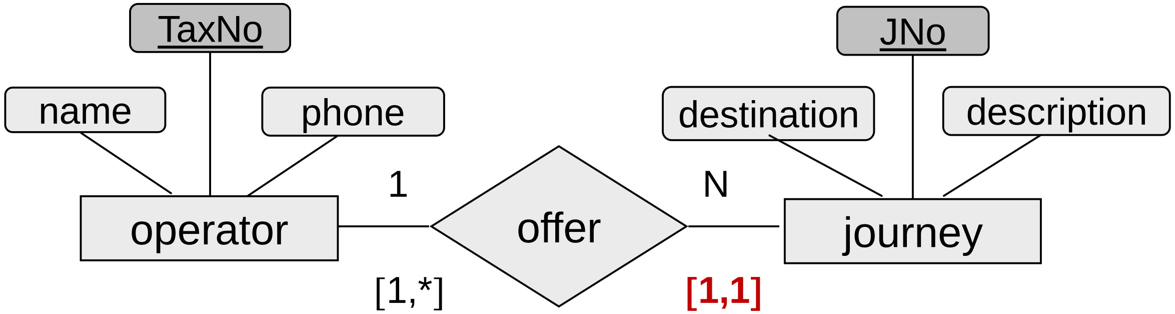
JNo	destination	description
33557	Norway	journey by mailboat
72345	South Tyrol	hiking
12399	Verona	trip with visit an opera

⤵

stage

JNo	No	place	number_stays
33557	1	Bergen	1
33557	2	Trondheim	2
33557	3	Kirkenes	2
12399	1	Verona	3

Example: cardinality constraint [1,1]



operator

TaxNo	name	phone

journey

JNo	destination	description

offer

TaxNo	JNo

operator

TaxNo	name	phone

journey

JNo	destination	description	**TaxNo**

Example: Database Schema

journey

JNo	destination	description	TaxNo

stage

JNo	No	place	number_stays

operator

TaxNo	name	phone

Website

Link	content

literature

ISBN	title	publisher	price

map

ISBN	section	scale

have

TaxNo	Link

travel guide

ISBN	region	pages

Summary of Conceptual Design

- Conceptual design follows requirements analysis, yields a high level description of data to be stored (conceptual level).
- ER model is a popular model for conceptual design, constructs are expressive, close to the way people think about applications; supported by many CASE tools.
- Basic constructs are entities, relationships, and attributes Some additional constructs: IS-A hierarchies, cardinality ratios, ...
- There are many variations on ER model constructs.

Summary of Conceptual Design (cont.)

- Several kinds of integrity constraints can be expressed in the ER model: key constraints, structural constraints, constraints on specializations
  - Some of them can be expressed in SQL when translating entity and relationship types into tables
  - Not all constraints can be expressed in the ER model
  - Constraints play an important role in determining a good database design for an application domain.

Summary of Conceptual Design (cont.)

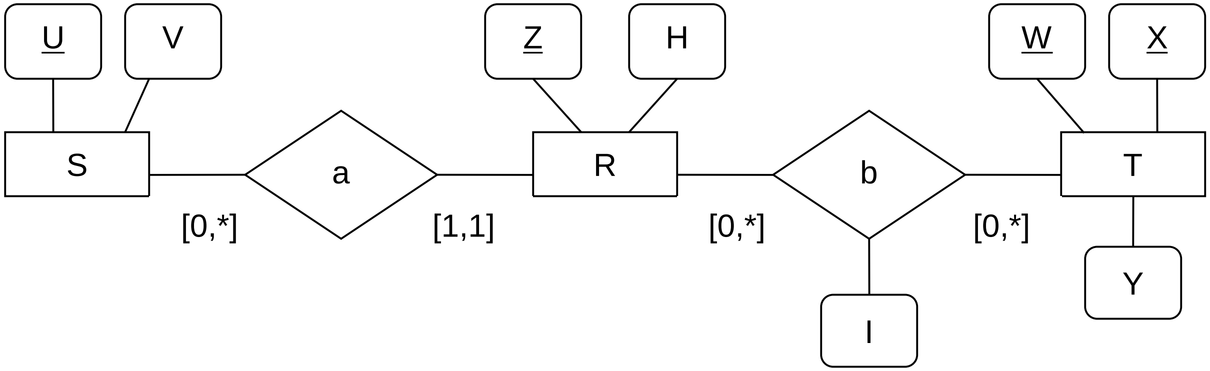
- ER design is subjective: There are many ways to model a given scenario! Analyzing alternative schemas is important!
- Entity type vs. attribute, entity type vs. relationship type, binary vs. n-ary relationship type, use of IS-A, generalization and specialization, . . .
- Ensuring a good database design includes analyzing and further refining relational schema obtained through translating ER schema.

Summary

- databases, DBMS, data model, and schema
- data independence by three level schema architecture
- database design process
- conceptual design using the ER model
- translating conceptual schema into relational schema (tables)

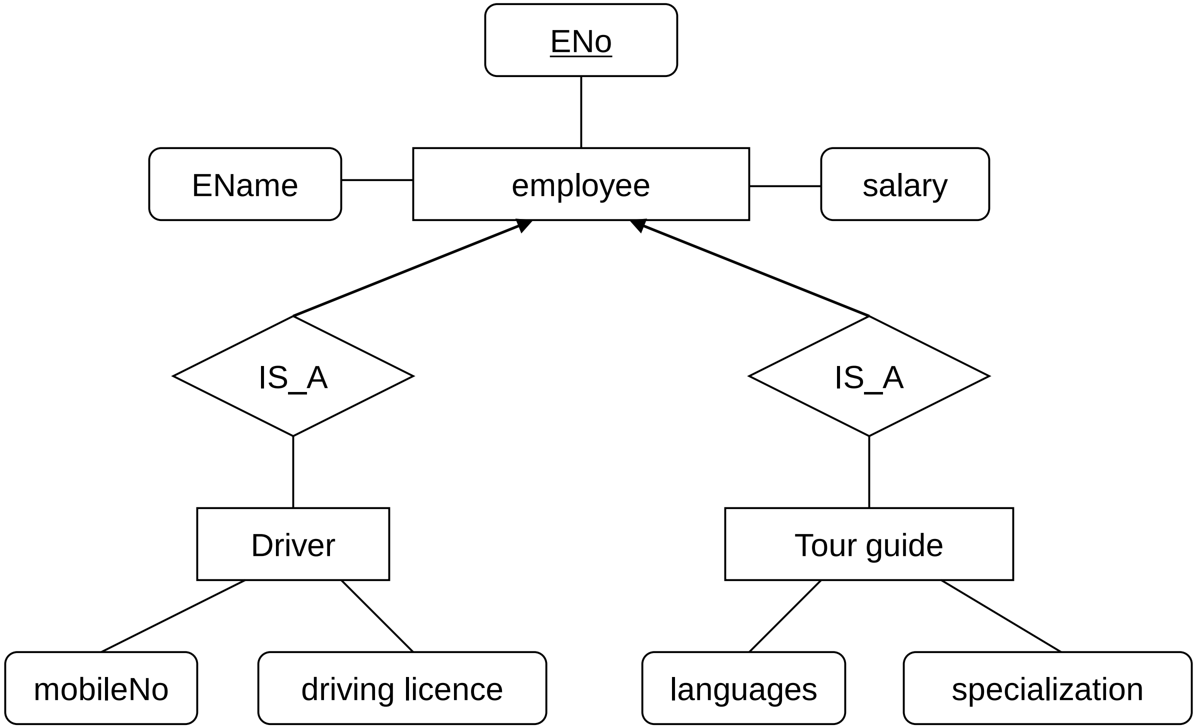
Exercises (1)

- Design a relational database schema for the following ER Diagram, i.e. the necessary relational schemas with their necessary attributes and their primary keys underlined.
- Consider possible optimizations to the database schema (reduction of tables).



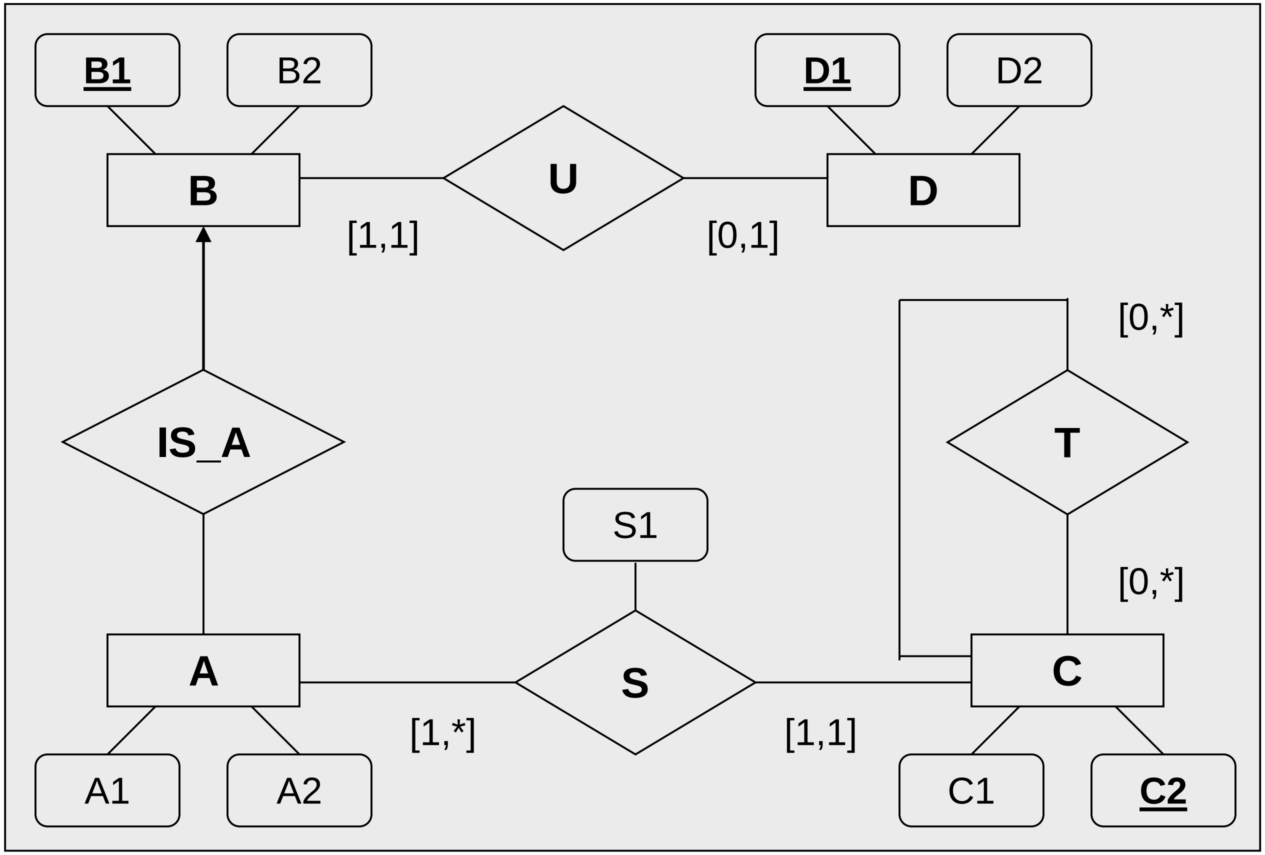
Exercises (2)

- Design a relational database schema for the following ER Diagram, i.e. the necessary relational schemas with their necessary attributes and their primary keys underlined.



Exercises (3)

- Design a relational database schema for the following ER Diagram.


B1
B2
D1
D2
B
U
D
[1,1]
[0,1]
[0,*]
IS_A
T
S1
[0,*]
A
S
C
[1,*]
[1,1]
A1
A2
C1
C2