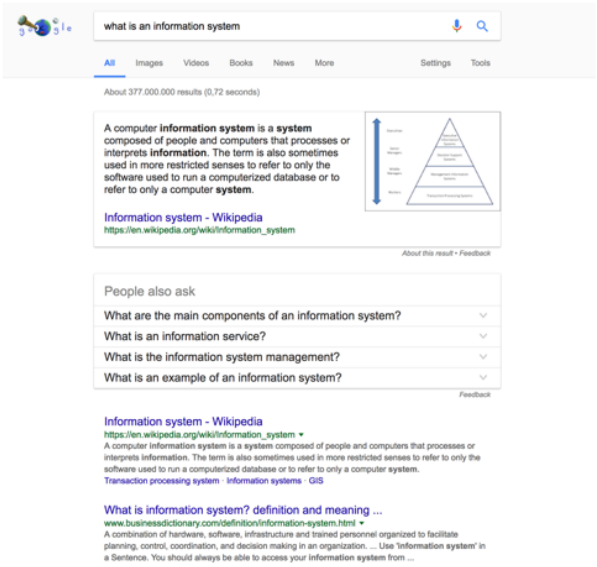
Introduction:
Distributed Information Systems
An Overview

Overview

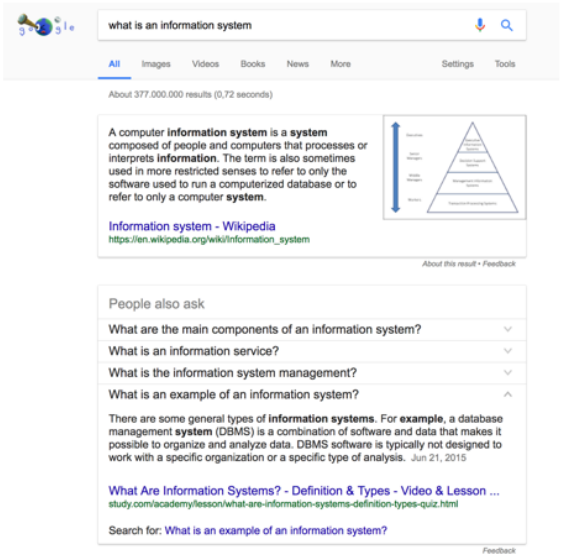
1. What is an Information System?
2. Data Management
3. Information Management
4. Distributed Information Management

1. WHAT IS AN INFORMATION SYSTEM?

Ask Google & Wikipedia



Ask Google & Wikipedia



Information Systems

Name	Literals	Categories	Similarity	DF	Entropy	Hashtag?	Harvesting?	Wiki/Wn
big data	de: "big data" en: "big data", "bigdata"	Information Systems	0.934	3437		Yes	No	
information systems	en: "information systems"	Information Systems	0.934	1303		Yes	No	
Business Intelligence	de: "business intelligence" en: "Business Intelligence",	Information Systems; Academic	0.901	101		Yes	No	
knowledge management	en: "knowledge management"	Information Systems	0.897	435		Yes	No	
information security	en: "information security"	Information Systems	0.878	518		Yes	No	
emerging technology	en: "emerging technologies", "emerging	Information Systems	0.871	564		No	No	???
data analytics	en: "data analytics"	Information Systems	0.869	91		Yes	No	
e-business	en: "e-business"	Information Systems	0.866	159		No	No	???
change management	en: "change management"	Information Systems	0.866	256		No	No	???
cloud computing	de: "cloud computing" en: "cloud computing"	Information Systems	0.863	390		Yes	No	
data mining	en: "data mining"	Information Systems	0.863	612		No	No	???
supply chain	de: "lieferkette", "supply-chain-management"	Information Systems	0.863	951		Yes	No	
data management	en: "data management"	Information Systems	0.862	365		Yes	No	
Internet of Things	de: "internet der dinge", "iot", "yacht"	Information Systems	0.862	1151		Yes	No	
data science	de: "data science" en: "data science", "data	Information Systems	0.861	741		Yes	No	
data integration	en: "data integration"	Information Systems	0.854	149		No	No	???
analytics	de: "analytisch", "analytisches	Information Systems	0.853	2536		Yes	No	
decision support systems	en: "decision support systems"	Information Systems	0.852	113		No	No	???
information management	en: "information management"	Information Systems	0.850	519		No	No	???

Objectives:

Understand the difference among an IS and other IT systems.

Understand the role of IS for managing models

Understand different aspects of reality represented in information systems

Understand the constituents of a model

Understand that functions in information system are often explictly represented

Understand the concept of interpretation of a model

Understand that having a correct model is a very difficult problem

Information Systems

Examples, e.g. at EPFL

- course catalogue, finance, library system, news, search engine, genome information, campus map

Computer systems that are not IS?

- IP telephony, room temperature control, computer game, Matlab

In this part of the lecture we would like to understand the basic concept of information system. We are surrounded today by information systems and everybody has an intuitive understanding what an information systems is and does (a system that is treating information). A large organization such as EPFL is running dozens if not hundreds of information systems. With the advent of the Web and more recently mobile computing and the resulting democratization of information technology and integration of information technology in every day's life a plethora of new information systems have been emerging. Increasingly information systems are not only interacting with humans, but also are among each other, with sensors gathering data from the environment, algorithms making decisions and different systems exchanging their data. The recent trend of generating and analysing increasing amounts of data, the so-called Big Data is even more demonstrating the growing importance of information systems. The following are some types of information systems that are common:

The classical information systems

- Organizational databases
- Business process management systems
- Geographic information systems
- Text retrieval systems

More recent types of information systems

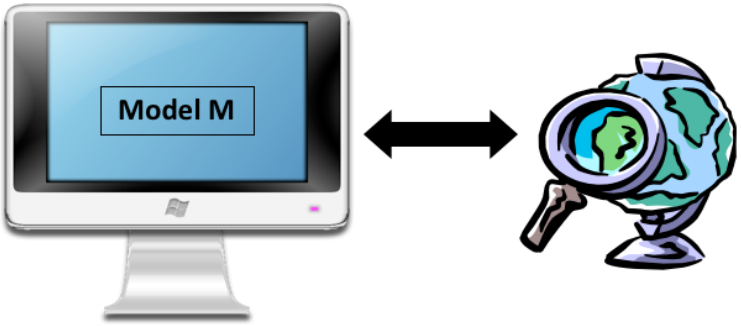
- Social Networks
- Query-Answering System
- Recommender Engines
- Business intelligence (data warehousing and mining systems)
- Bioinformatics systems (e.g. genome or protein sequence retrieval)
- Environmental monitoring systems (disaster warning, meteo website)
- Web community systems (recommender systems, social networks)
- Publish-subscribe and data dissemination systems (e.g. RSS, mobile broadcast)
- Etc.

However, not every computing system is considered as an information systems. Systems that are used for communication through different media (e.g. IP telephony), games or simulations are not unanimously considered as information systems. What is the distinctive feature of an information system? In the following we will provide a more

precise characterization of the concept of information system for the purposes of this course and is we understand the concept in the context of this course.

What is an Information System?

An information system is a **software** that manages a **model** of some aspect of the **real world** within a (distributed) computer system for a given **purpose**.



There exists no generally accepted, formal definition of what an information system is. For sure, information systems are software systems. In addition, one can quite safely state that all information systems represent within a computer system a model of a part of the world. And this model is needed to fulfill some purpose. We base our definition on this : “An information system is a **software** that manages a **model** of (some aspect of) the **real world** within a (distributed) computer system (for a given **purpose**)”.

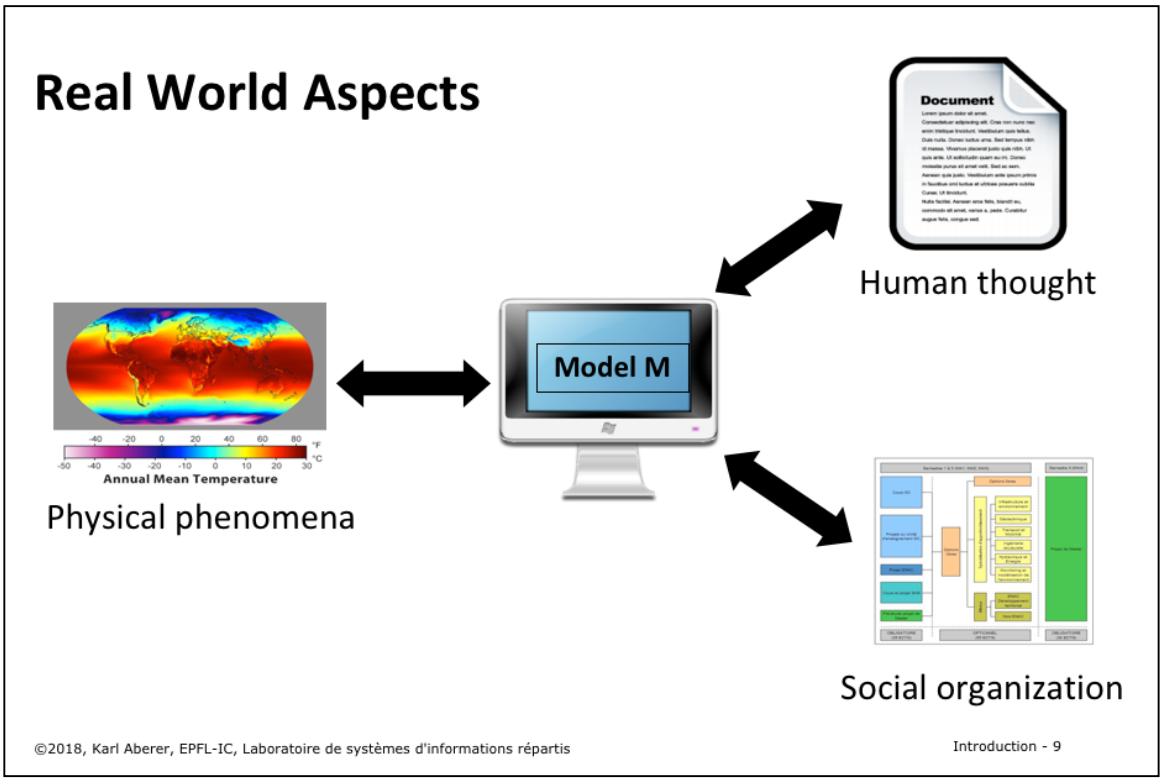
This definition involves a number of concepts that require further explanation:

real world: The notion of real world refers not necessarily to our physical environment only. It can be anything from abstract concepts (e.g. a legal information system) to technical systems including computer system or networks itself (e.g. information systems for network management).

purpose: every information system has an entity (human, computer) that makes use of it. It does so, in order to perform a certain task related to some aspect of the real world (e.g. making a decision, performing a computation etc.).

aspect: this implies that there exist many different ways to represent the real world and same aspects of the real world in information systems, depending on the purpose.

model: what a model is and what is its role we will explore in more detail in the following.



We can identify broadly three types of real-world aspects that are captured by information systems:

1. Physical phenomena: these information systems measure the environment and create models of physical phenomena. Typical examples are meteorological information systems, or geo-information systems.
2. Social organization: these information systems capture the roles, relationships, activities etc. in social organizations, such as businesses and institutions. This type of information systems is probably the earliest one that had wide-spread use, for applications such as finance, logistics etc.
3. Human thought: these information systems model human thought and reasoning processes. They enable to capture the meaning of text and other media, assess the importance and quality of information, but also model human traits such as sentiments or opinions. Information retrieval systems, of which web search engines are a specific example, are the typical representative of this class of systems.

Broadly information system represent physical phenomena, the social organization of humans and the result of human thought processes.

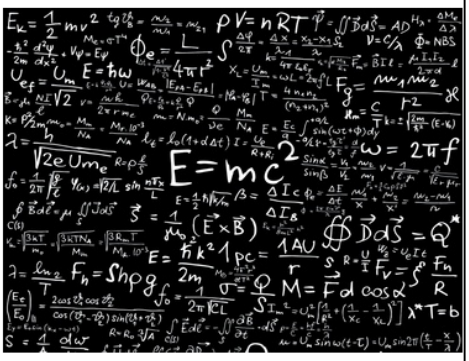
Exercise: Classify common information systems according to these categories? Identify the **purpose** for which these different information systems are used.

Models

A **model** is a mathematical structure consisting of a set of

- **Constants** (or identifiers)
- **Functions** (or relations)
- **Axioms** (or constraints)

The set of constants and functions must be consistent with the axioms



Information systems use models to represent some aspect of reality. But what is a model? The answer is simple: any (mathematical) structure can serve as a model.

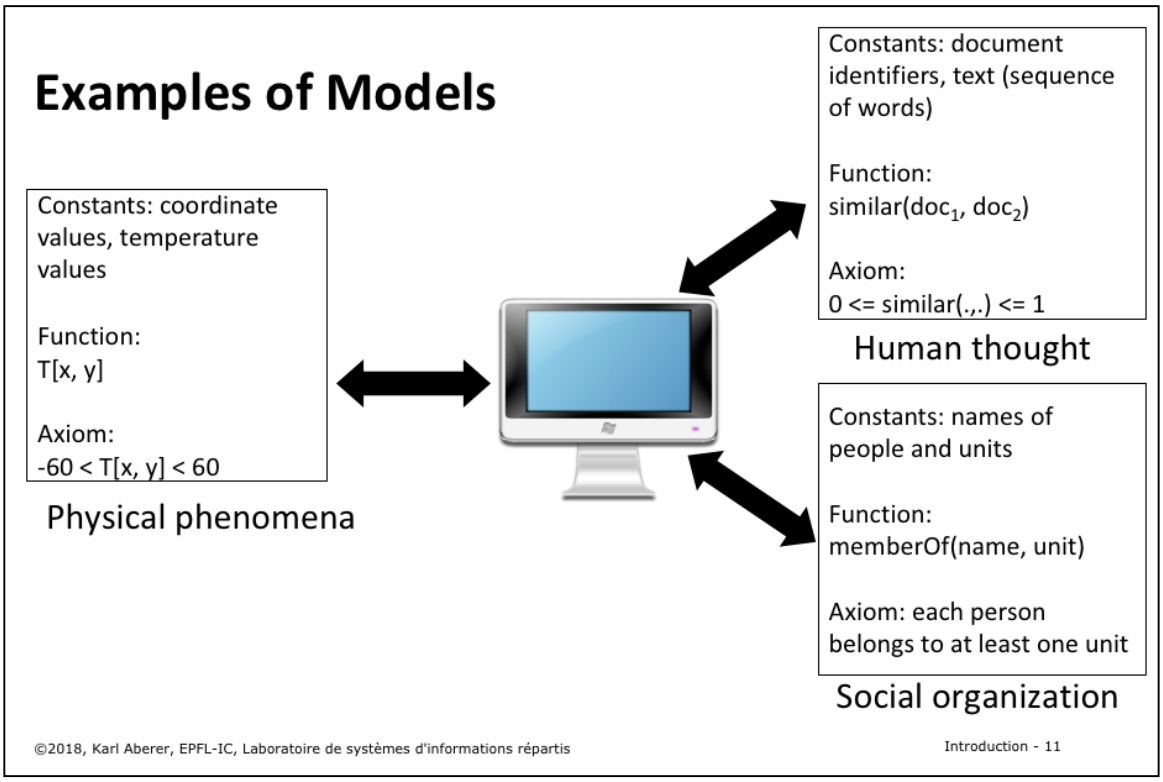
A mathematical structure consists of constants, functions and axioms. The constants are used to give names to things (indeed everyTHING can receive a name – and a central task of information systems is to provide names, respectively identifiers, to real world objects), the functions to provide properties of these things, and the axioms provide rules or constraints that state which properties are possible and not.

Some information systems support only very limited or specific kinds of models, others very generic models. Very often first order logics is used for information systems as very generic approach. Models based on first order logic allow the to represent the important entities, their relationships and properties of the real world in a generic approach. However, with the increasing need to process information for specific needs (e.g. processing text, images, sensor data), also more specific mathematical models are increasingly used, such as graph models, vector spaces, probabilistic models, differential equations, simulation programs etc.

Here are some examples of formal models used in information systems:

- Entity-Relationship models have been among the earliest conceptual models used for information systems. They have been derived from knowledge representation mechanisms developed in AI.
- OWL: is a generalization of the entity relationship model enabling logical inference (for concept classes). It has become the basic model for the Semantic Web.
- Graph models: used for social network data, biological network data, communications network data
- Vector space models: used to represent feature spaces of text and media content
- Probabilistic models: used to represent uncertainty in content and sensor data
- Differential equations and simulation programs: used to represent behaviors of complex systems
- Process models have been developed to capture the structure and dynamics of business processes, also called workflows.

Some you have already encountered in courses on data management or software engineering, the use of some others we will demonstrate later in this course.



For our three running examples we can provide concrete instances of possible models. A temperature map we can model, for example, as a two-dimensional matrix. An organizational structure is best captured using relationships among entities, whereas documents can be modelled by their degree of similarity.

Exercise: determine for each of the examples which type of mathematical structure is being used.

Representation of Functions

Functions can be represented by giving a specification or algorithm (implicitly) or by enumeration (explicitly).

We call enumerated functions **data**.

Example:

- Implicit representation: $f(x) = x^2$
- Explicit representation: $f(1) = 1, f(2) = 4, f(3) = 9$

Functions are a key constituent of information systems. They relate objects with their properties and different objects among each other. An interesting and central question is of how functions are represented.

Actually, there exists two fundamentally different ways to do this: either by explicitly enumerating function values for given function arguments, or by providing a specification or algorithm to compute the function value from a given function argument. Both ways are actually used in information systems.

Functions in Information Systems

Information systems strongly rely on explicit representation

- many aspects of the world are not algorithmically defined, e.g., birthdate of a person
- difference to simulation systems

Computed functions (implicit representation) play nevertheless an important role

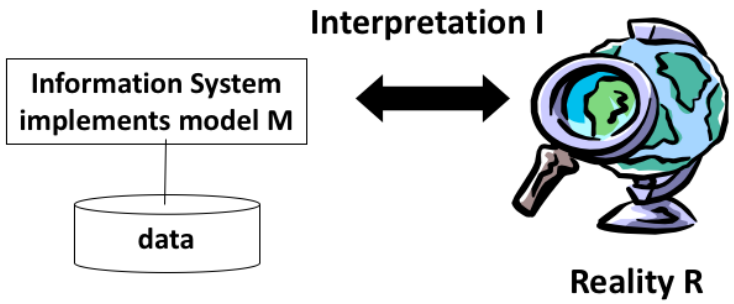
- queries, views, user-defined functions etc.

However, since many facts about the real-world cannot by specified by an algorithm (e.g. computing the birthday from the name of a person), the explicit representation of functions play a particularly central role in information systems. Explicitly enumerated functions we also call commonly **data**.

Nevertheless also implicitly represented functions play an important role in information systems (computing the age of a person from its birthday). Such functions appear under many different names, such as queries, views, user-defined functions etc.

How do we know that a model is a model?

The model is linked by an interpretation (relationship) to the real world

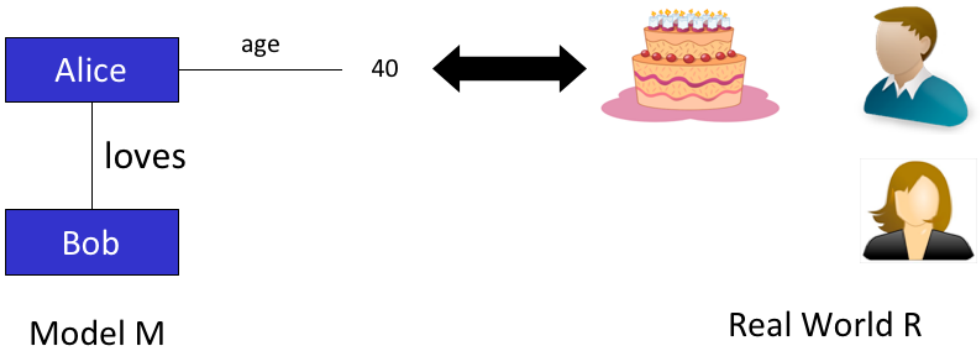


When building a model of the real world in an information system, a natural question to ask is of how we can know that the model is indeed a model of the real world. In an abstract sense the answer to this question is very simple: we have to provide an interpretation function, that maps every constant of the model to some real world object, and the functions in the model preserve all relationships that occur in the real-world. Such an (interpretation) function is also called a homomorphic function (a function that preserves the “form”).

Interpretation

The interpretation relationship is **homomorphic**

- I: M ➔ R
- preserves relationships



Formally: an interpretation relationship I that maps constants of a (real-world) domain R to a (model) domain M (I: M -> R) is homomorphic iff

I(f(x1,…,xn)) = f_real(I(x1),…,I (xn))

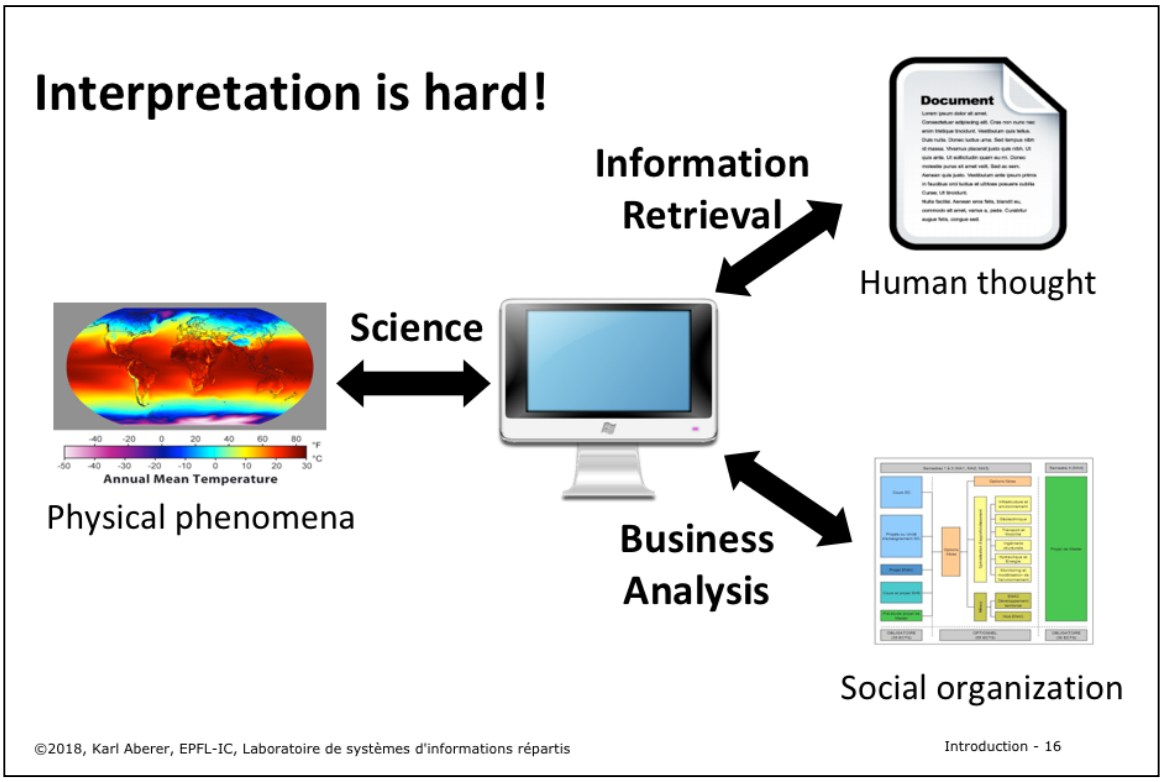
where f_real is the function in the real-world that is represented by f.

In the example if the constant “John” maps to a person named John and the constant “Ann” maps to a person named Ann, and the two persons are married, then the function husband(“Ann”) should have the value “John” in the information system:

I(“John”) = John

I(“Ann”) = Ann

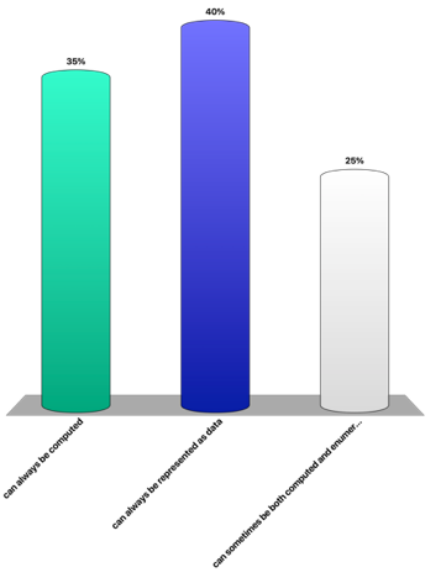
I(husband(“Ann”)) = I(“John”) = John = husband_real (Ann) = husband_real (I (“Ann”))



Since there is no way to formally verify functions in the real world (e.g. whether two people are married), indirect methods are required that help to verify, or at least make plausible, that a model represents correctly the real world. With respect to physical phenomena it is indeed Science that endeavours to create models of reality (e.g. physical laws) using the scientific method. With respect to capturing human thought, expressed for example as written text, the field of information retrieval has established methods to verify whether computer models appropriately represent, or at least mimick, the human reasoning processes. For traditional business information systems, it is the profession of business analysts that translates real world requirements and organizational structures into models for information systems. These approaches always implement some form of feedback mechanism, in which the models that are developed are verified with respect to reality. E.g., business analysts present the models to potential users and refine them based on the feedback. Scientists verify their models with respect to experimental data and assume they are valid as long they are not falsified (as we know since Popper we can never proof that a model is correct!).

Functions in models …

1. can always be computed
2. can always be represented as data
3. can sometimes be both computed and enumerated



Can you give a concrete example of a function in an information system, where the function can be alternatively computed or enumerated?

Interpretation relationships ..

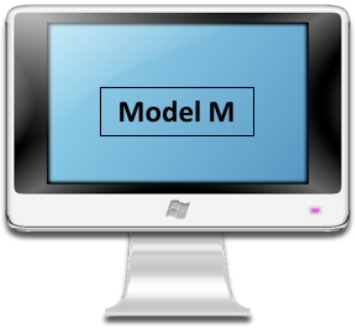
1. are always computable
2. are sometimes computable
3. are uniquely defined

Can you give an example of a computable interpretation relationship?

2. DATA MANAGEMENT

Data Models

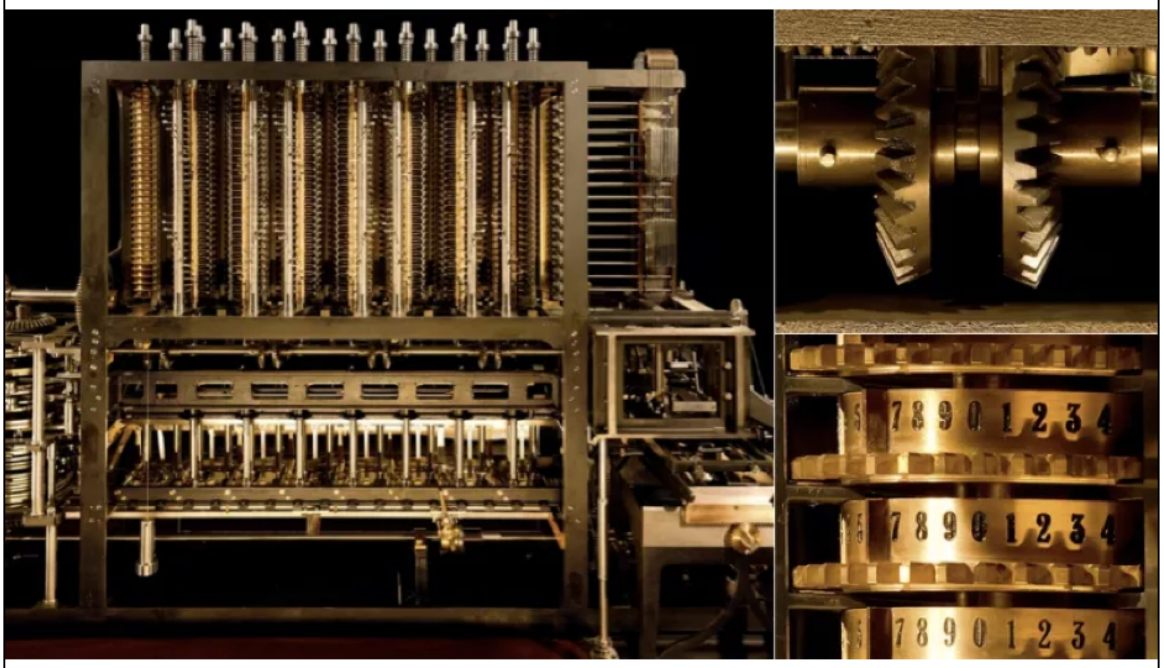
How is a model represented in a computer system?



A model M is represented using a **data model D**.

A data model D uses **data structures and operations** for the representation of the constants, data, functions and constraints of a model M within a computer system.

So far we have considered models as abstract mathematical formalisms, consisting of constants, functions and axioms. For being able to handle a model in a computer system, we need to represent the model within the computer system. For that purpose we need a representation mechanism that can be “understood” and processed by a computer, in other words we need a formalism that can be used to represent a model. Such a formalism is called a **data model**. A data model consists of data structures and operations that a computer can represent and execute.



Example: Associative Array

Represent a function f: K → V

Abstract data type (the mathematical structure)

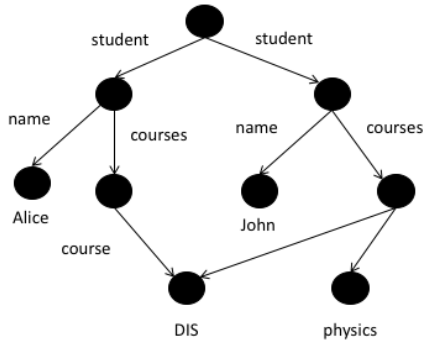
- Representation: {(key, value)}
- Operations: add(key, value), search(key), delete(key)
- Constraint: every key occurs only once

Implementation: hash tables, binary search trees, tries, linked lists, …

The study of data structures is at the heart of computer science. For example, the **associative array** is an abstract data structure (abstract data type) that is used to represent functions. Thus it is evidently of central interest for representing functions of models. Associative array is a data structure that manages a set of key-value pairs (representing the function) and that supports a set of operations to manipulate the associative array, such as adding, deleting and modifying the elements of the associative array. It is called abstract data type, since different (physical) implementations of the same data structure are possible. The different implementations exhibit all the same logical behaviour, but may have different performance characteristics.

Exercise: review different implementations of associative arrays, such as hash tables and tries.

Other Popular Data Structures



Labelled Graphs

STUDENTID	NAME	COURSE
1234	John	DIS
3456	Bob	physics
2345	Alice	DIS

Relational Tables

Other frequently used data structures used as data models are labelled graphs and relational tables. These data structures allow to represent relationships among entities in a generic way. Relational tables are at the core of the relational data model of SQL database systems, graph-oriented data models are becoming increasingly popular in the context of managing linked information in the Web and data from social networks.

Database

The collection of data represented in a data model D is called a **database DB.**

A computer system that is designed to (generically) manage databases is called a **database management system DBMS**.

We already mentioned that the explicit representation of functions by enumeration, resulting in data, plays an important role in information systems. In the context of data management such a set of data is called a **database**. A computer system that is designed to manage databases is called a **database management system DBMS**. Thus database management systems can be used to manage databases, and thus to realize information systems. The inverse is not necessarily true. Many information systems that use database do this without using a specific database system.

Data Modeling Languages (1)

The use of the term data model is ambiguous.

It may also designate a language used to specify data models, consisting of

1. Data Definition Language (DDL)
2. Data Manipulation Language (DML)

The specification of a data model using a DDL is called a **database schema**, or simply **schema S**.

In this context we have to be very careful about terminology, as data model is interchangeable used to designate two different things: (1) a data model used to represent a model within a computer system (this is the sense we will use in the following) and (2) a formalism respectively language to specify a whole class of data models. Data modelling formalisms consist of two parts: A data definition language DDL enables the specification of data models, consisting of possible data structures and integrity constraints. The data manipulation language allow to specify the functions in the data model.

A specification of a data model using a DDL is called a database scheme, respectively simply schema.

Data Modeling Languages (2)

A data modeling language specifies three main components:

- **Data structures**: a collection of data structures, which are used to represent databases.
- **Integrity constraints**: a language to express rules the data in databases has to observe.
- **Manipulation**: a collection of operators, which can be applied to the data structures, to update, transform and query the data, in a database.

The three components of a data model formalism are the following: (1) the structural part describes of how constants and functions are represented as data structures. This enables the representation of facts in a database. (2) integrity constraints that have to be respected by the facts can be expressed using a specific language, (3) the data manipulation operators enable manipulation of the databases, e.g. adding and removing of facts, or querying, i.e. computing functions that answer questions of users.

Example: Relational Schema

Data Definition Language

```
CREATE TABLE Student
        (Studentid NOT NULL PRIMARY KEY,
        Name, Course)
```

Data Manipulation Language

```
SELECT Name FROM Student WHERE Course = « DIS »
```


The probably best known data modelling formalism is the relational data model (note the use of the term data model!). In this small example we see of how to use the DDL to define a table with a specific structure (e.g. three fields) and how to define an integrity constraint for this data structure, i.e. that the Studentid field needs to be unique. Using the DML a query is formulated. A query is nothing else than a function that is computed on the data structure.

Database Management Systems

The software that implements a data modeling formalism is the **database management system**

Database management systems allow to implement many information systems. They are designed to be **application- or domain-independent**.

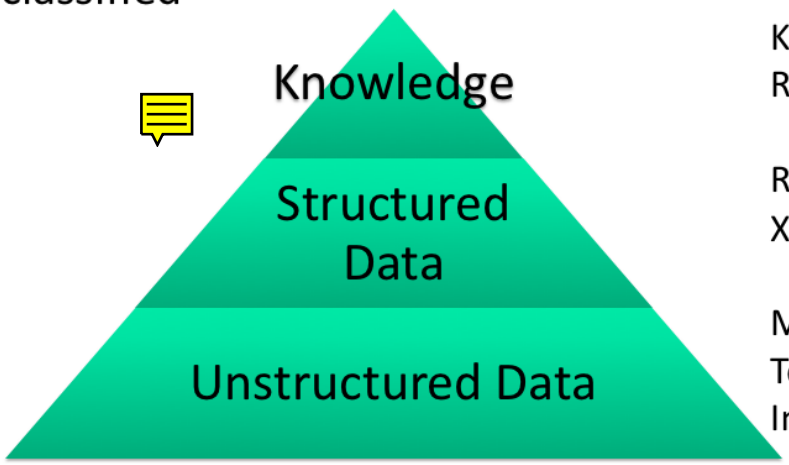
The data is stored using encodings of data structures exploiting available storage media (e.g. main memory or disk) and their addressing mechanisms.

Data modeling formalisms are implemented (typically) by database management systems. Database systems are a software that is designed to support storage and manipulation large databases using a specific data modeling formalism, such as the relational data model. Many of the models used in information systems, as discussed earlier, can be represented by data models specified in data modeling formalisms, and thus database systems are often used for the implementation of information systems. However, there many information systems that implement their native data management, one example being text retrieval or Web retrieval systems.

Database management systems are designed to support the management of very large databases efficiently. To that end they exploit different techniques to encode the data in data structures on storage media, that take advantage of the technical characteristics of the implementation platform.

Levels of Abstraction

Depending on the “degree of abstraction” data is classified



Knowledge Graphs
Rules

Relational data
XML and RDF data

Measurement data
Text
Images and Video

There exist some conventions to distinguish data into different levels of abstractions. One can think of these levels of abstraction as levels of increasing interpretation of “raw data” that has been recorded or captured. These interpretations are usually obtained either by automated or human analysis of the data, or a combination of the two.

Levels of Abstraction

Unstructured data

- Data captured from measurements and human input
- Structure given by static data types (e.g. time series)

Structured data

- Data is structured according to a schema

Knowledge

- Schema can evolve dynamically as knowledge expands

The classification is not strict!

Unstructured data is usually referred to as data that originates more or less directly from some direct interactions with the real world environment, be it through measurements (sensors, images, videos), human inputs (natural language, text, query logs), machine input (system logs) or similar. The data is in fact structured (e.g. an image is an array of pixels, or a text is a sequence of characters), but the structure is considered to carry little “semantic” meaning.

Structured data is data that follows a static schema, that implements a interpretable model of some domain of interest. The most prominent example being the relational data model, where applications or users define tables with specific meaning. Such meaning could be extracted from unstructured data, e.g. a table could contain the list of all objects recognized in an image.

Knowledge is also structured data, but with a more open scope of applications. One may think that the schema of knowledge

data can by dynamically extended, and new knowledge can be derived from existing knowledge through inference. Since the representation of knowledge is deemed to be more flexible, usually grpah-based data models are used, that allow to represent in an unconstrained way new relationships.

Note: the notion of “knowledge management” is used in practice in with a very different meaning. It refers to information systems that manage the knowledge of an organization, consisting, e.g., of its documents and information on the skills of its collaborators.

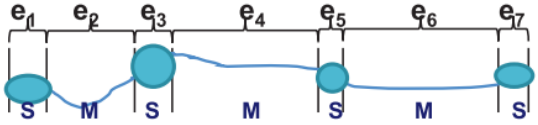
Example

Unstructured data: a GPS trace

a sequence of GPS point (x,y,t)

Bob's and Alice's GPS trace

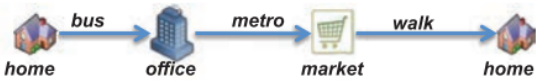
Structured data: Road segments, Places



a sequence of episodes

Places that Bob and Alice visit

Knowledge: Concepts and Inferences



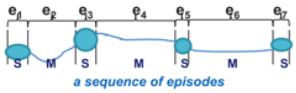
episodes with semantic annotations

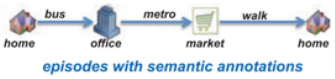
Bob and Alice are frequently together in Ouchy, thus *Bob loves Alice*

This examples illustrates how such abstractions look like in practice. We can consider GPS traces as raw, unstructured data. Unstructured expresses here the fact that apart from the physical location we have no further understanding of the meaning of the data. By using automated analysis (e.g. segmentation methods) and background knowledge (e.g. maps) one can extract information about places (e.g. where GPS coordinates did not change for a period) and roads / paths where movement is detecting. Aligning such structured data with maps can give an interpretation of where the person was and by what means it moved, which then constitutes knowledge. Inferences on such knowledge might then reveal relationships among persons and produce high level knowledge such as "Bob loves Alice".

Storing Data in a DBMS

Model (Information System)





episodes with semantic annotations

Data (Database System)

x	y	t
12	13	5:00
12	14	5:01
13	15	5:02

place	x	y	d
p111	12	13	2:00
p112	13	15	1:00

segment	xs	ys	xe	ye
s221	12	13	13	15
s222	13	15	20	27

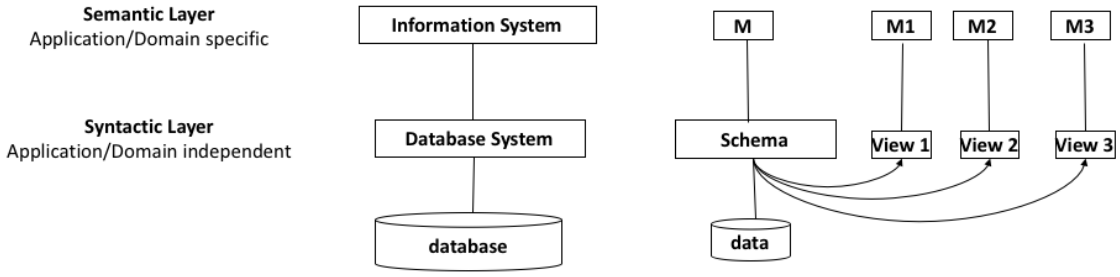
Subject	Relation	Object
home	ISA	Place
bus	ISA	Vehicle
Bob	ISAT	home

From the perspective of data management the distinction among the different levels of abstraction is not significant. Apart from the fact that for certain types of data, certain types of data management systems might be more applicable and efficient than others, any DBMS can in principle manage data at any abstraction level. This is illustrated here for the example where all data could be managed in a relational database management system.

Logical Data Independence

The same data can be interpreted in different ways

- Views support different schemas for accessing the same data stored in a database
- Corresponds to supporting different models based on the same data
- Important to support evolution of schemas



One of the central ideas in the design of database systems (and information systems in general) is data independence. Data independence in general means that an abstract representation of data is used that is independent of the underlying implementation of the data at the lower abstraction level.

In database management systems **views** are used to realize the concept of **logical data independence**. Views are a mechanism to derive support different conceptual models (higher abstraction level) using the same logical database schema (lower abstraction level). This is for example important to support evolution of the database. The logical database schema may schema, e.g. as a result of changes in the organizational structure of an enterprise). This should not affect current applications that rely on the existing logical database schema. This the changes in the evolution are “hidden” by creating a view.

Example: Logical Data Independence

Original logical schema

- Manager(Name, Unit)

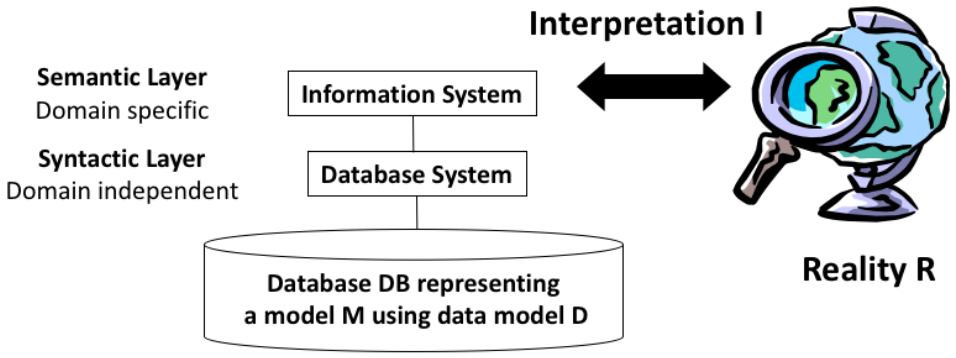
New logical schema

- Manager(FirstName, LastName, Unit)

View to support old logical schema

- CREATE VIEW old.Manager AS
  SELECT concat(FirstName,' ',LastName) as Name, Unit FROM new.Manager

Database Systems



Having introduced the notion of data model, we can refine our view of information systems. An information system is (typically) based on a database management system. The information system is concerned with providing a model for a real-world aspect, whereas the database system is concerned with the efficient management of the data structures required to represent the model. In many practical systems this separation is not that explicit. But it is very helpful to have such an architectural view, for better understanding the architecture of systems and the separation of different concerns (i.e. those related to a specific application, and those that are domain independent). We can understand the separation among the concept of information systems and database systems also as one among syntax and semantics.

What is not specified in a data definition language?

1. The structure of a relational table
2. The query of a user
3. A constraint on a relational table
4. The definition of a view

Logical data independence is not intended to …

1. support different conceptual models on the same database
2. create a conceptual model over different databases
3. change a conceptual model without affecting the database schema

Key Data Management Tasks

Efficient management of large amounts of data

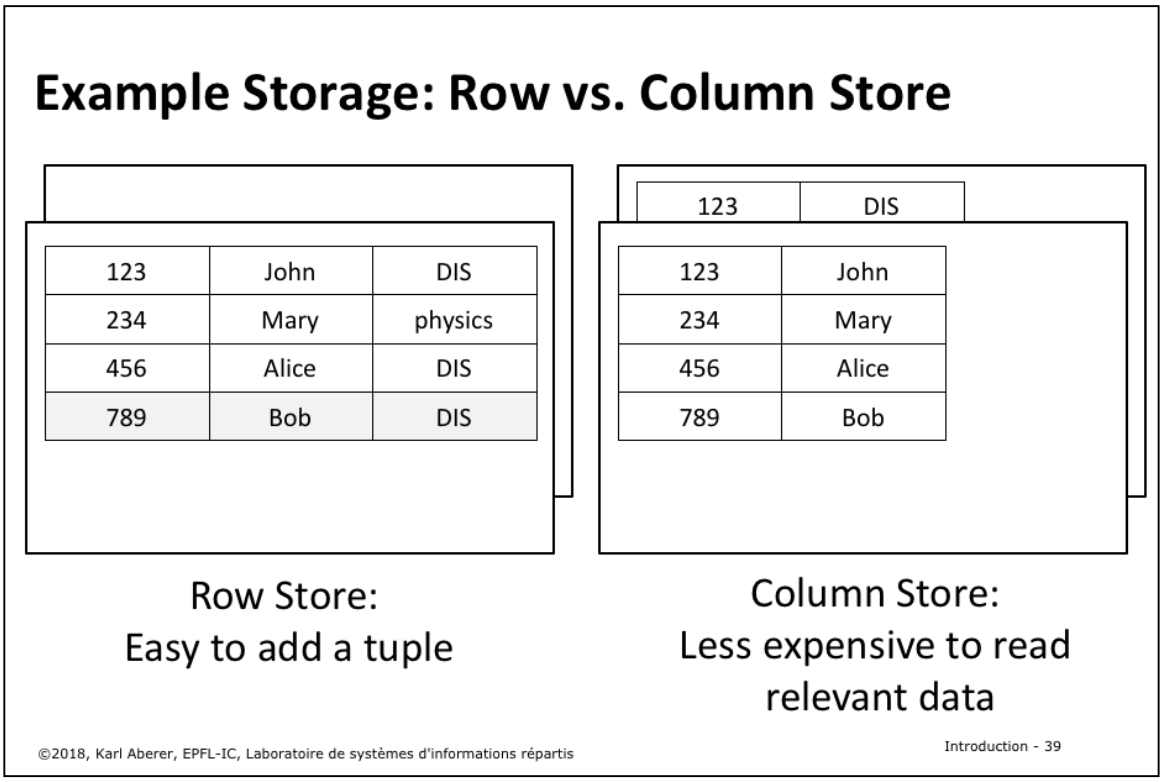
- efficient storage and indexing
- efficient search and aggregation

Ensuring **persistence** and **consistency** of data under updates and failures

- Persistence = data stored independent of lifetime of programs
- Consistency = data correct independent of type of failures

Perform both tasks by exploiting different media (e.g. memory, disk, flash, tape)

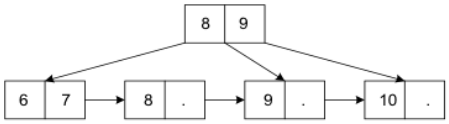
When handling large amounts of data, we face two key challenges: first, the management should be performed efficiently. This concerns questions of how the data is mapped to the different available storage media, and what auxiliary data structures, so-called indices, are created to support the efficient access and questions of how operations on the data, e.g. for search and aggregation, are performed algorithmically in an efficient way. Second, the data needs to be kept safely. Different to data that is managed within the lifetime of a program, so-called transient data, data in information systems is maintained independently of the lifetime of any program. This property is called **persistence** of the data. Ensuring persistence and consistency of data, under all possible failures and when the data is stored in a variety of storage media us a non-trivial problem.



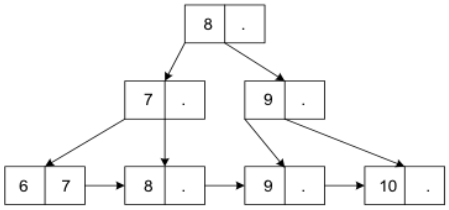
A very basic issue in the design of storage layouts is the question whether for table data a row-oriented or column-oriented storage approach is being used. Traditionally relational database management systems adopted a row-oriented approach, where complete tuples are stored together within a block of the underlying storage system (e.g. a disk). This makes it easy to add or modify a tuple, since all the values of the tuple are stored in the same block. Recently, with the rise of Big Data and data analytics, more and more applications use such data to perform aggregation operations on specific attributes (e.g. to produce statistics). For efficiently performing such operations it is much better to store all the values of a specific attribute of a table together, so that fewer storage blocks need to be accessed to obtain all the values of a single attribute. On the other hand, this makes updates more expensive, since now the values of the same tuple are distributed over different storage blocks. Dealing with such trade-offs between different physical storage schemes and data access patterns, is at the heart of data management technology.

Example Indexing: B+-Tree

insertion of 6



rectification of tree



One example of a typical problem related to supporting the efficient access to data is the design of indexing structures that perform will for very large databases and that can be efficiently managed on available storage media, e.g. disks with a fixed block size of storage. An example of a typical and very popular solution to this problem are B+-Trees. B+-Trees are designed to satisfy two main properties: the tree nodes can be stored efficiently on the available disk blocks (this is for example achieved by having a high fanout in the tree), and the tree remains balanced independent of how the data (more precisely the search keys) are distributed in order to guarantee a low depth of the tree and thus efficient search and update operations in logarithmic time. In order to guarantee these properties, updates to the B+-Tree might require additional operations to restructure the tree. In the example, after insertion, the maximal fanout condition is violated, and thus a new level needs to be introduced in the tree.

Optimizing Access to Databases

Choose indices and storage according to

- Different types of accesses: predicates on attributes, paths, …
- Different access patterns: frequency of queries, parameters, …
- **Physical database design**

For a logical operation choose

- among different ways exploiting the indexing structures and storage layout
- **Declarative Query Optimization**

Optimizing access to data implies two main tasks: (1) the choice of the data structures and storage layout that matches best the requirements of applications. This optimization is called **physical database design** and is based on known characteristics of the applications, such as the type of accesses and their frequencies. This optimization is performed **before** the accesses to the database are executed. (2) the choice of the best algorithm, given some storage and indexing scheme, when a concrete operation such as a query is to be executed. This optimization is called declarative query optimization and is performed **at the time** the access to the database is executed.

Problem: Safe storage and updates

Maintain consistent state

- in the presence of multiple users (Concurrency)
- in the presence of failures (Recovery)

Example

- T1 and T2 are different users, what goes wrong ?

Transactions		**State**
T1:	**T2:**	**account**
read(account)		20
account := account + 10	**read**(account)	20
	account := account + 20	
	write(account)	40
	commit	
write(account)		30
Commit		

The second major technical challenge for database systems is the support of consistent access, even when multiple applications access the system simultaneously and failures in the system occur. Here we illustrate a potential problem a parallel access to a account database by two applications might pose and how inconsistencies can be introduced as a result. One possible solution would be to introduce locks, such that as long as one application is accessing an account, no other application can access it. This might in practice, however, be too restrictive and many smarter solutions have been proposed to that end in the area of data management within the field of transaction management.

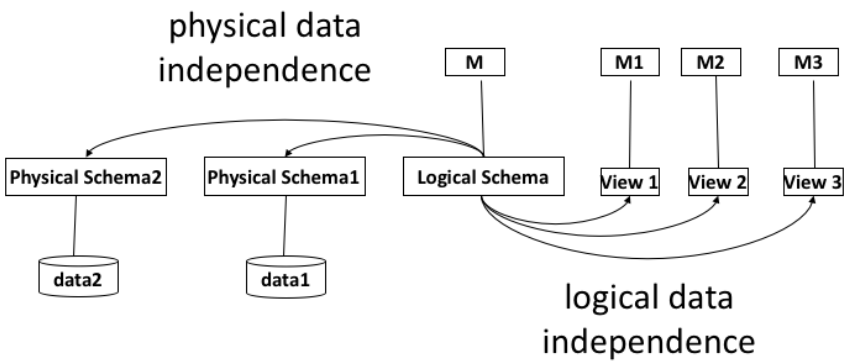
Transaction Management

Support for applications the following abstractions

1. Other users do not influence own transactions (**isolation**)
2. System failures do not affect the execution of operations
   - Transactions are completely executed or not at all (**atomicity**)
   - Once transactions are executed their result is never lost (**durability**)

The previous example illustrated one problem that can occur in the access to databases, that different applications can interfere when accessing the same data and thus violate consistency. Transaction management provides mechanisms, such as locking schemes, for avoiding this problem and to provide **isolation** among applications. In other words, transaction management is supporting the abstraction that each application appears as being the only application accessing a database. Another problem that may occur when updating a database are failures, resulting in potentially incomplete operations or in the loss of the results of already completed operations. Transaction management supports to this end two other properties. The first is **atomicity**, that is a transaction, that transforms a database from one consistent state to another, is either completely executed or not at all, and thus potentially inconsistent intermediate states cannot occur. The second is **durability**, which guarantees that results of completed transactions are never lost. Both properties can be assured by implementing, for example, suitable logging schemes for operations executed on a database.

Physical Data Independence

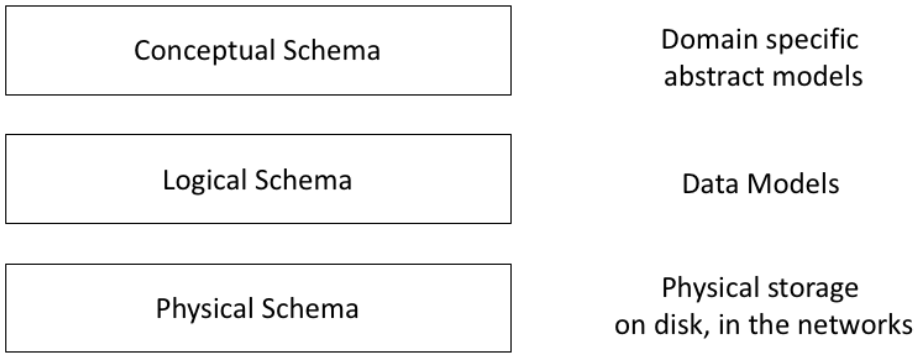


Different physical realizations of the database implementation (storage layout, query execution) do not affect the result

The second central data independence concept in databases, after logical data independence, is physical data independence. Physical data independence expresses the concept that the same logical database, e.g. having the same database schema, can be physically realized in many different ways, e.g. using different physical database designs, and the same logical operations, e.g. queries and updates, can be executed in physically different ways, using declarative query optimization and transaction management. Physical data independence relieves developers of database applications that focus mainly on the modeling of the application, i.e. in engineering an information system, from dealing with physical optimizations that are important for performance purposes.

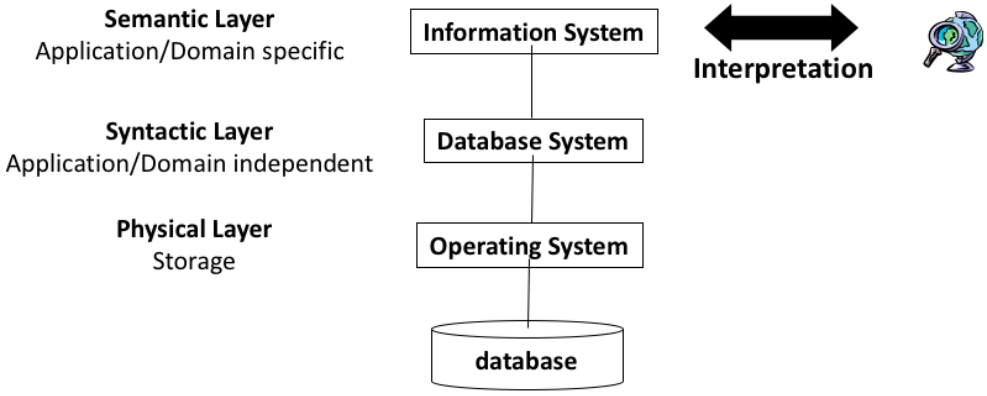
Modelling Architecture

Defined as standard by ANSI in 1975



The distinction between application-specific models of information systems, data models used for their implementation in a database systems, and different physical realizations of data models, dates back as early as 1975 where it has been standardized in the ANSI modeling architecture.

Refined View of an Information System



Factoring in the design principles of data management systems we can provide now a refined view of a typical information system architecture. The semantic layer is concerned with the modeling of the real world. In order to implement the model the syntactic layer provides a data model in which the semantic model can be implement and that is supported by a generic software (typically a database management system) that supports a wide range of information systems. The physical layer deals with the problem of representing the constructs of the data model efficiently in the computing system, using the typical mechanisms as provided by its operating system, such as access to storage and network resources.

Which is wrong? An index structure …

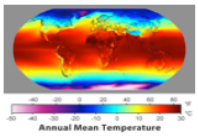
1. is defined as part of physical database design
2. is selected during query optimization
3. accelerates search queries
4. accelerates tuple insertion

Persistence means that …

1. a change of a transaction on a database is never lost after it is completed
2. the state of a database is independent of the lifetime of a program
3. the same logical database can be stored in different ways on a storage medium

3. INFORMATION MANAGEMENT

Information Management Tasks



model M

Retrieval: given a model, find some specific data

Data mining: given data, find a model the matches the data

Example:
Temperature model allows to find temperature at selected locations

data

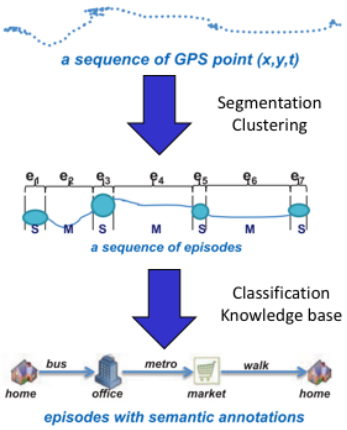
Example:
Given temperature measurements, find a function that interpolates for all locations

Since the central role of an information system is to create a model of reality based on data, the key information management tasks are related to the interplay between data and models. We can identify two directions for this interplay: from models to data, and from data to models. From models to data is commonly what we understand by **retrieval**, or information retrieval. Given a model of reality we would like to learn about specific aspects of reality. If we have a model of the temperature distribution in the world, we would like to retrieve the temperature at a specific location or a global average temperature. For Web search we would use a model of how search terms provided by user to a Web search engine relate to documents considered as being relevant by the user, to retrieve the results of a user query. Going from data to models is what we commonly understand as data mining. Often we find big data collections for which we do not have a proper or only incomplete interpretation. For example, we might have temperature measurements at given points, but do not understand the correlations among those measurements or the values at locations without measurements. If we have large document collections, we do not understand which are the topics that are covered by those documents. Data mining deals with the problem of providing algorithms that reveal hidden structures in data in order to create new models. Both information retrieval and data mining will be central topics that will be covered in this course.

Data Mining

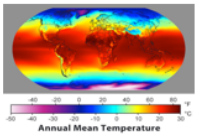
Creating “higher level abstractions” from “lower level data”

- Statistical and Machine Learning methods, as well as rule-based approaches
- Typically on large datasets
- Also called: data science, data analytics



We have earlier distinguished different levels of abstraction in data. Data mining is the means to derive higher level abstractions from data, or in other words, new models.

Information Management Tasks

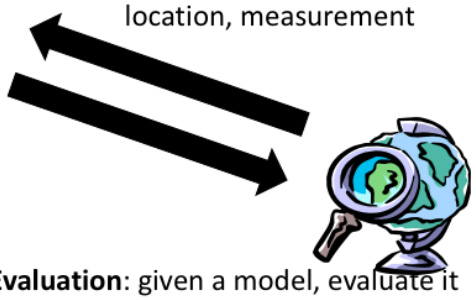


model M

data

Conceptual modeling: analyze the real world and specify a model

Example: define concepts temperature, location, measurement

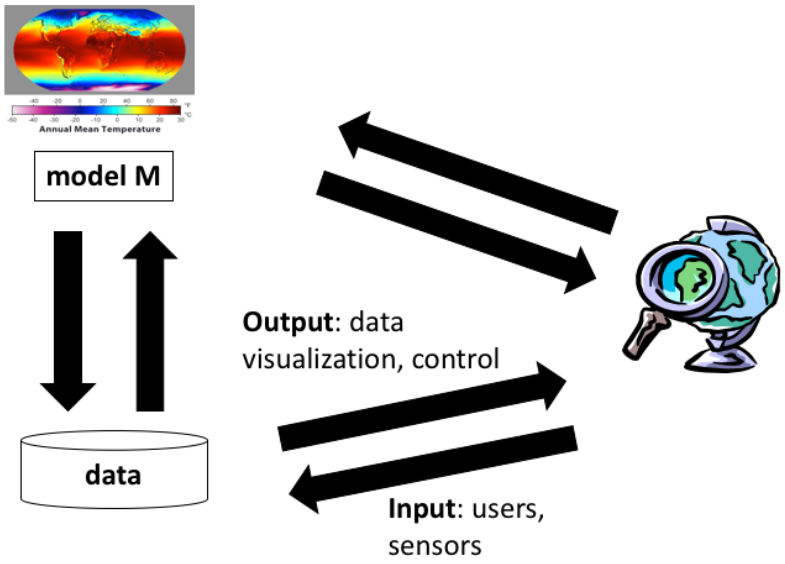


Evaluation: given a model, evaluate it against reality

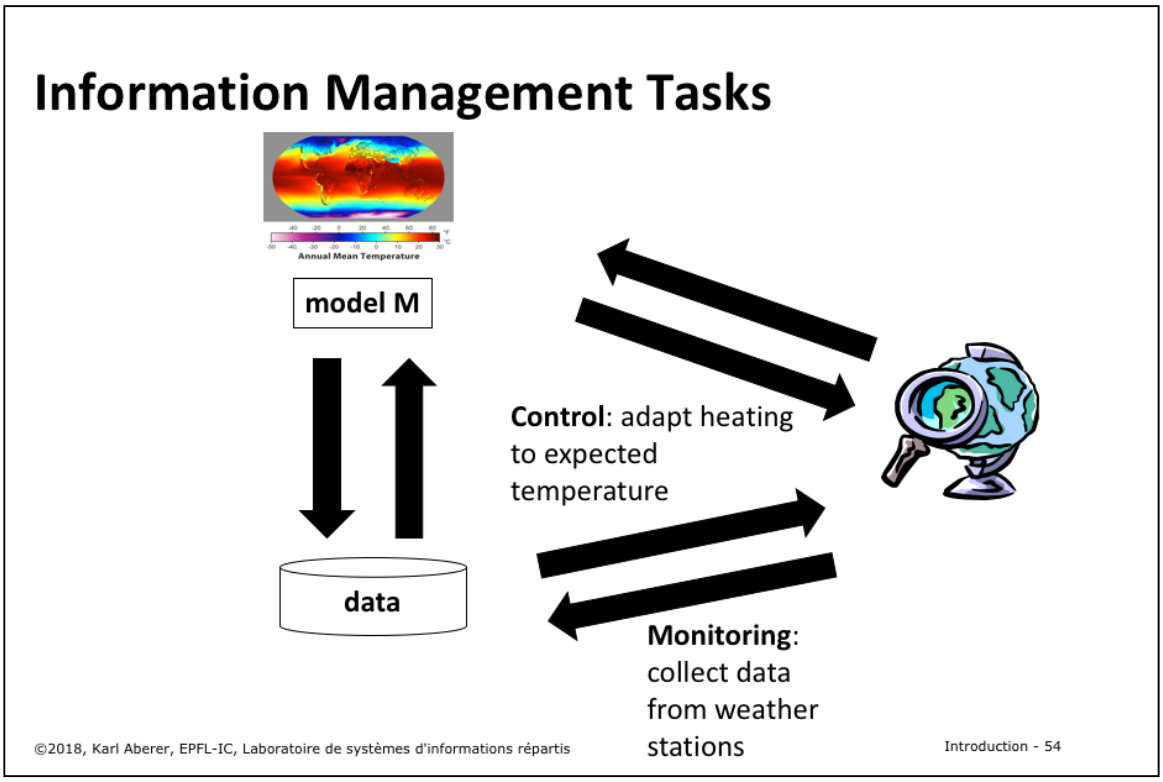
Example: compare predicted temperature to measurements

A second important task in information management aims at establishing the connection between the model used in an information system and the real world. In other words it is about establishing the interpretation relationships. This is possible only in indirect ways. Again we have two directions involved in the relationship among models and the real world. First, if we do not yet have a model, we need to observe and analyse the real world and (intellectually) derive models. For example, in the case of temperature modelling we would identify the concepts temperature, location and measurements as key concepts and represent them in a model. More generally conceptually modelling is widely used in the development of business information systems, where business analysts perform requirement analyses in order to determine what are the organizational structures and the processes within a business, and what are the processing needs. On the other hand, given a model we would like to verify whether the model is correct, for example, whether it produces prediction that correspond to reality. In our running example on temperature this could be achieved by comparing temperature values predicted by the model for certain locations with the true values. Evaluating models is both an important task for information retrieval (verifying that the models used are correct) and data mining (verified that newly generated models are correct). We will discuss detailed methods for performing these tasks later in the course.

Information Management Tasks

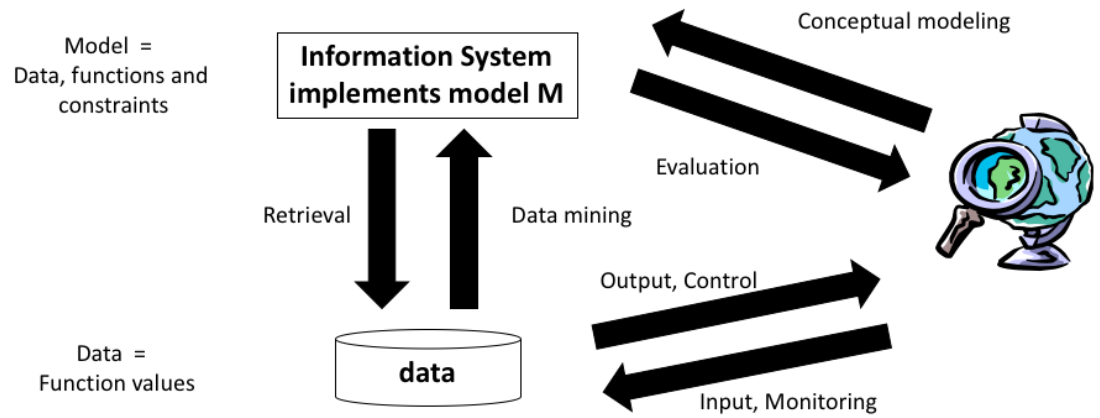


Regarding the interaction between an information system and the real world, we can consider the interaction with human users as well as direct interaction with the real world environment via sensors and control devices. With respect to data output for human users the visualization of large datasets is becoming increasingly important. With respect to human input a recent development is the use of input from large communities, so-called crowd-sourcing. Direct interactions with the real-world is what is called today the Internet of Things, where computers receive data through sensors and control directly devices.



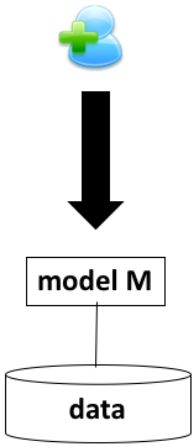
For completing the picture we could also consider the interaction between data processing systems and the real world. We may understand this relationship in the view of the increasing expansion of the Internet of Things, with devices and objects being directly connected to information systems. Through this connection the interaction is twofold: on the one hand the real-world devices are generating data that can then be further processed in the information systems, on the other hand data generated in the information system can be used to control real-world devices.

Information Management Tasks



Here we summarize the main information management tasks, and of how they interconnect the models, the data and the real world.

Utility



Users need information system to take decisions

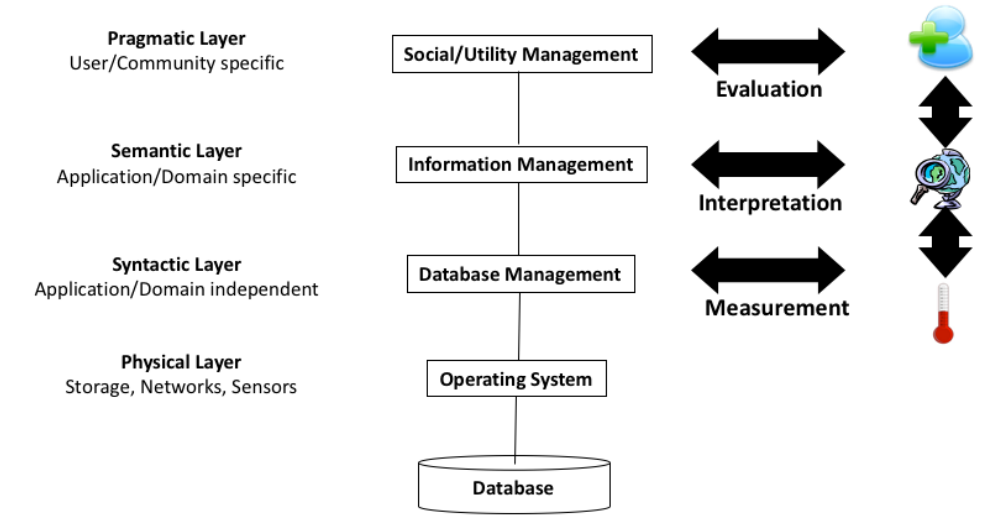
Utility of information linked to the value achieved

Value depends on
- Importance of decision
- Quality of decision

Quality of decision depends on quality and understandability of information!

Finally we are coming back to the issue of the purpose of an information system. Information systems are needed to make informed decisions. Thus their reason to exist is to provide useful information. The **utility of information** depends on the nature of the decision on the one hand, and on the quality of the decision that is possible based on the information on the other hand. Traditionally utility of information has been considered implicitly in the design and implementation. But as the awareness of the importance of information is growing, more and more we also see that utility of information, and related factors, such as quality are treated and managed explicitly. For example, data quality in databases is evaluated and monitored nowadays in many cases explicitly. Systems for rating and recommendation, e.g. in social networks, are another example where quality of information is handled explicitly.

Refined View of an Information System



Considering the information management tasks mentioned before we can further refine our view on information systems by several aspects. We can add an additional layer that corresponds to the users of the system, and that we call the pragmatic layer as users introduce the dimension of information utility. There exists no well-established notion for this layer. It can be considered as part of knowledge management, but this concepts is much wider, and it is implicitly found in many areas such as evaluation of information systems, data quality management, agent systems, social networks etc.

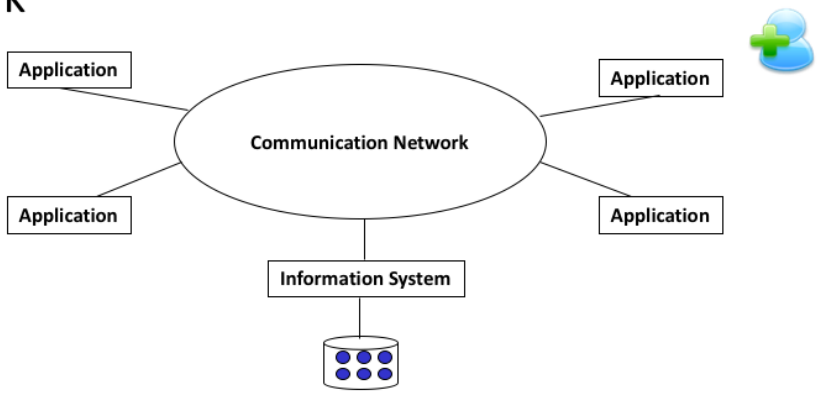
Grouping Facebook users according to their interest by analyzing the content of their tweets is …

1. a retrieval task
2. a data mining task
3. an evaluation task
4. a monitoring task

4. DISTRIBUTED INFORMATION SYSTEMS

Centralized Information System

Centralized Information System on Computer Network

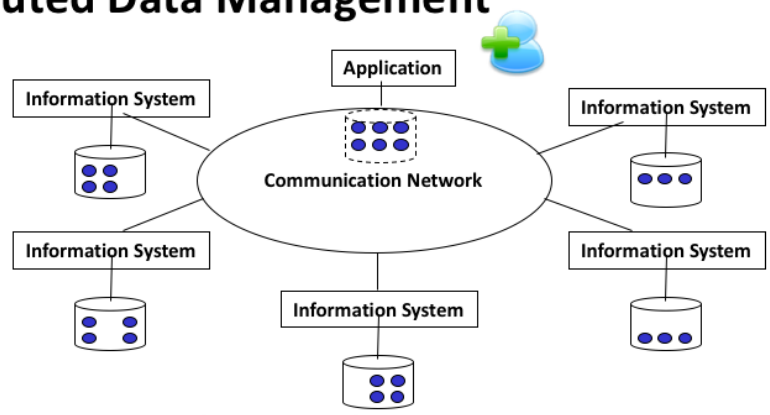


Except in the very early days, information systems had always been used in computer networks. This does not imply any significant additional problems beyond those we have discussed already, as long as the information system is centralized, i.e. running on one physical node under a single authority. The network just enables the interaction of a user with the information system from a remote location.

Physical Distribution

Use of distributed physical resources: locality of access, scalability, parallelism in the execution

Distributed Data Management



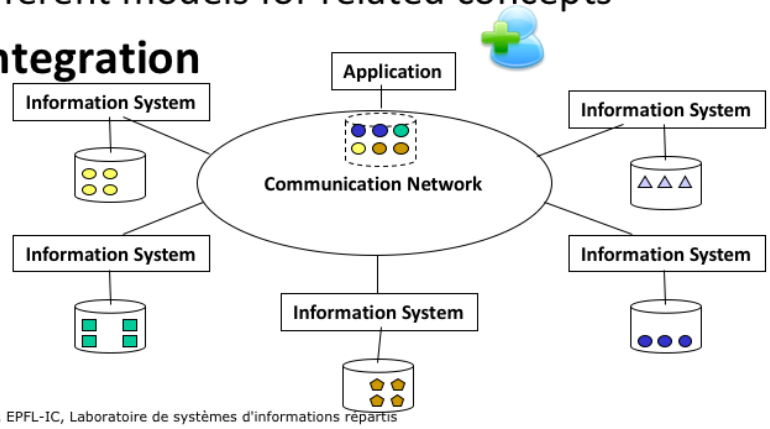
There exist however many reasons not to store all data in a single node of the network. Some of these reasons are related to the optimized use of available resources. We might want to move data in the network close to the node where it is accessed, we might want to take advantage of parallel processing of the data, and we might want to avoid bottlenecks in order to improve scalability of the system. All these are good reasons to distribute the data physically. However, physical distribution should be ideally fully transparent to the user. The user has still the impression of accessing a single information system that is running under a single authority. This model of distributed processing of data is the subject of **distributed data management**.

Heterogeneity – Logical Distribution

Use of different data models

- Independently developed information systems
- Different models for related concepts

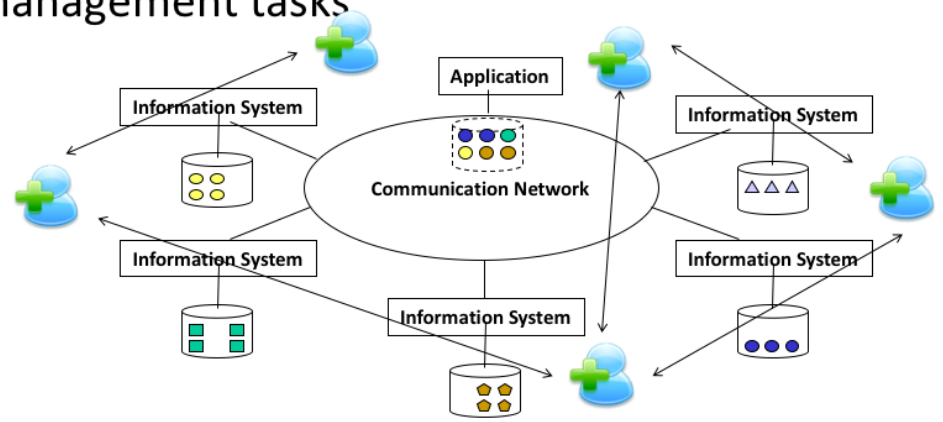
Data integration



Having a homogeneous view of a distributed information systems might not always be possible. If we want to access information in systems that have been developed by different entities, or if we want to integrate information from different information systems that have been independently developed, we can no longer assure that the information can be homogeneously accessed. The same information might be represented using different models and data structures, and the access methods might be different. In that case we are talking about **heterogeneous information systems**. The heterogeneity results from a distribution of the decision authority when designing the system. In order to overcome heterogeneity methods for integrating data and making information systems interoperable are needed.

Autonomy – Distribution of Control

Independent users have to collaborate, coordinate, negotiate, to perform information management tasks



Finally in heterogeneous information systems we have to deal with the problem that the different information systems are under the control of different **autonomous** authorities. This poses problems of coordination, mutual trust, and privacy protection. The information systems can be considered as independent agents, that may pursue common goals, while protecting their own interests.

Creating a web portal for comparing product prices is (primarily) a problem of …

1. Distributed data management
2. Heterogeneous data integration
3. Collaboration among autonomous systems

Distributed Data Management

Key Issues in Distributed Data Management

Where to store data in the network?

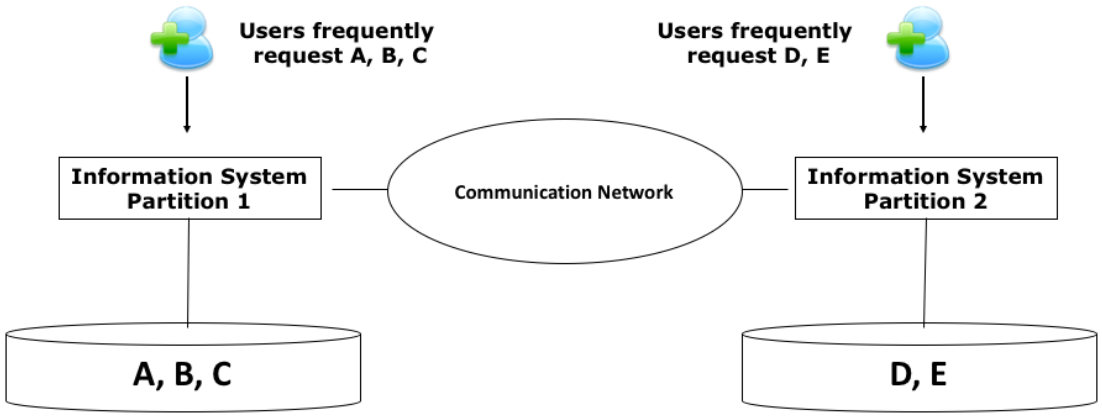
- **Partitioning** of data
- **Replication** and **caching**
- Considering typical access patterns and data distributions

How to access data in the network?

- **Push** vs. **pull** access (query vs. filtering)
- Indexing of data in the network
- Distribution of queries and filters
- Considering the communication model

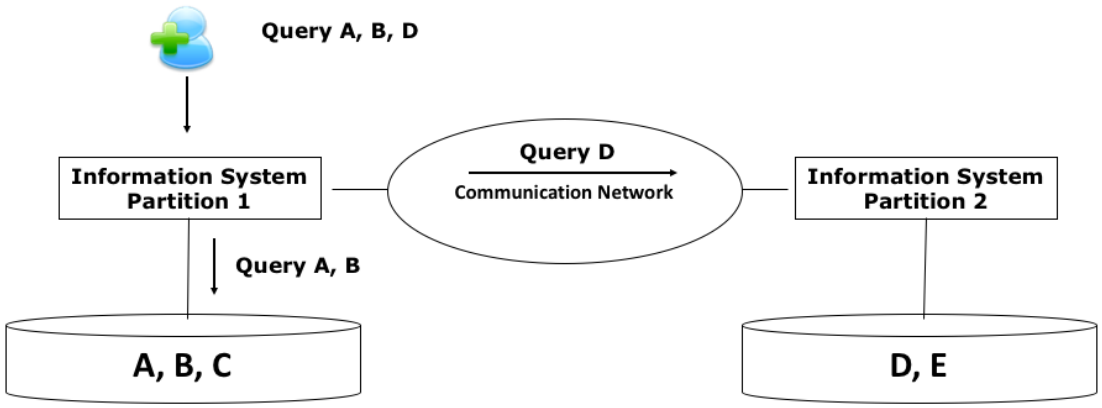
Distributed data management deals with similar questions as centralized data management, namely optimizing the storage of the data and the processing of accesses of the data. The new key element is that data can now be stored at different nodes in a network, and that the cost of data transmission over networks becomes an essential performance consideration. Since cost of data transmission is generally considered as expensive, in distributed data management often multiple copies of the same data are kept at different locations in order to speed up access or data is partitioned to bring the data closer to the application that use different parts of the data. This in turn implies new problems of keeping distributed data consistent while executing transactions that involve different nodes in the network. The area of distributed transaction management is dealing with this problem.

Data Partitioning

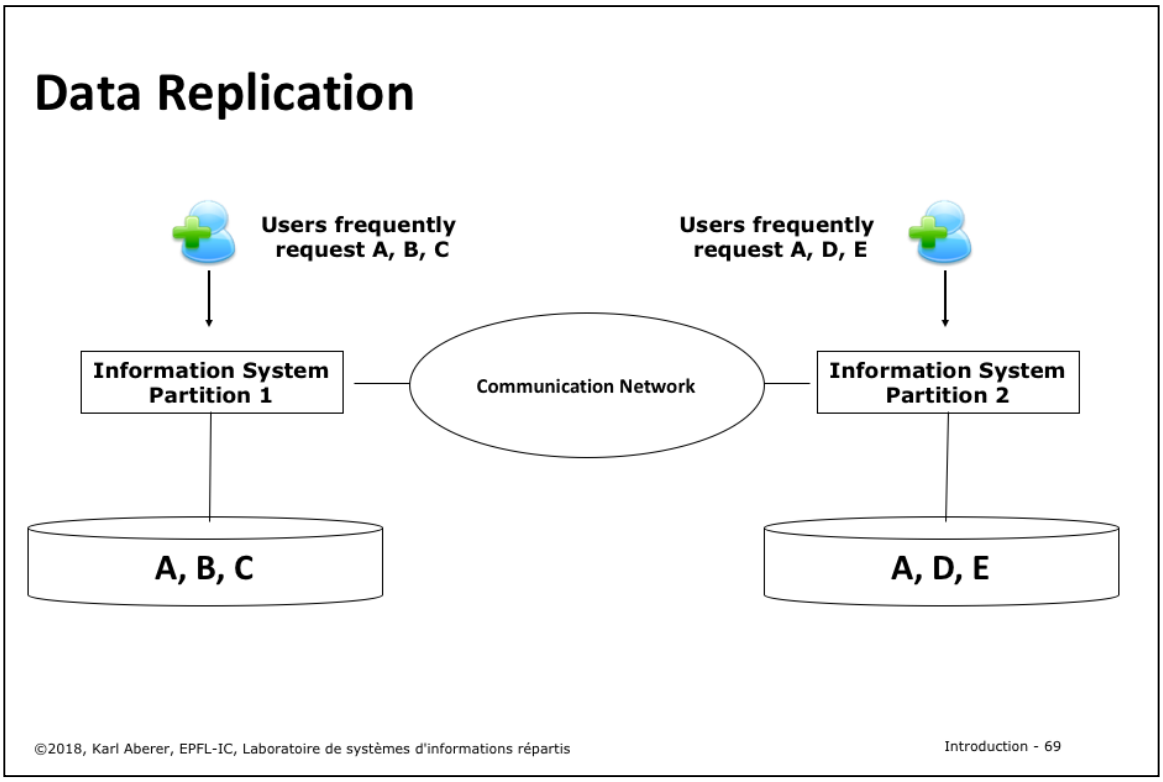


Since the cost of data transmission in communication networks is comparably high, it makes sense to move data to those nodes in the network where it is mostly used. Since for different data in the same database these can be different nodes, databases can be cut into different partitions that then are distributed in the network. Determining the optimized partitioning of a database is part of distributed physical database design, comparable to the design of storage layout schemes for centralized databases.

Distributed Query Processing

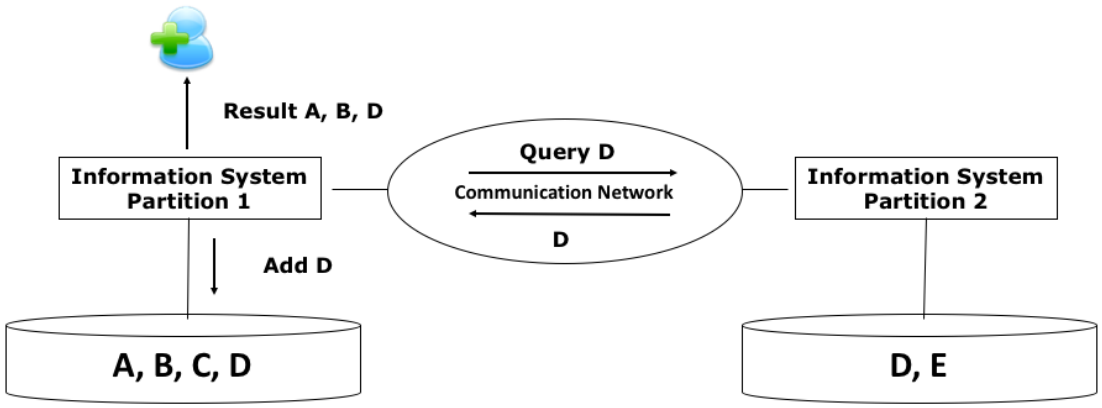


Once data from a database is distributed over different nodes in the network, certain queries can no longer be answered by accessing only a single node. Distributed query processing deals with the problem of analysing queries and deciding which data can be retrieved from which node. For executing a distributed query, messages are sent over the network (e.g. in form of sub-queries), to aggregate the data from the different nodes into a single result.



It may also be that the same data is used frequently in multiple places. In that sense having the data stored only in a single node may not be the best solution, and it is better to replicate the data into multiple places. Replication improves data access, by reducing the required network traffic, but comes at an additional cost. Updates become more expensive as they have to ensure that replicas are correctly updated and kept consistent.

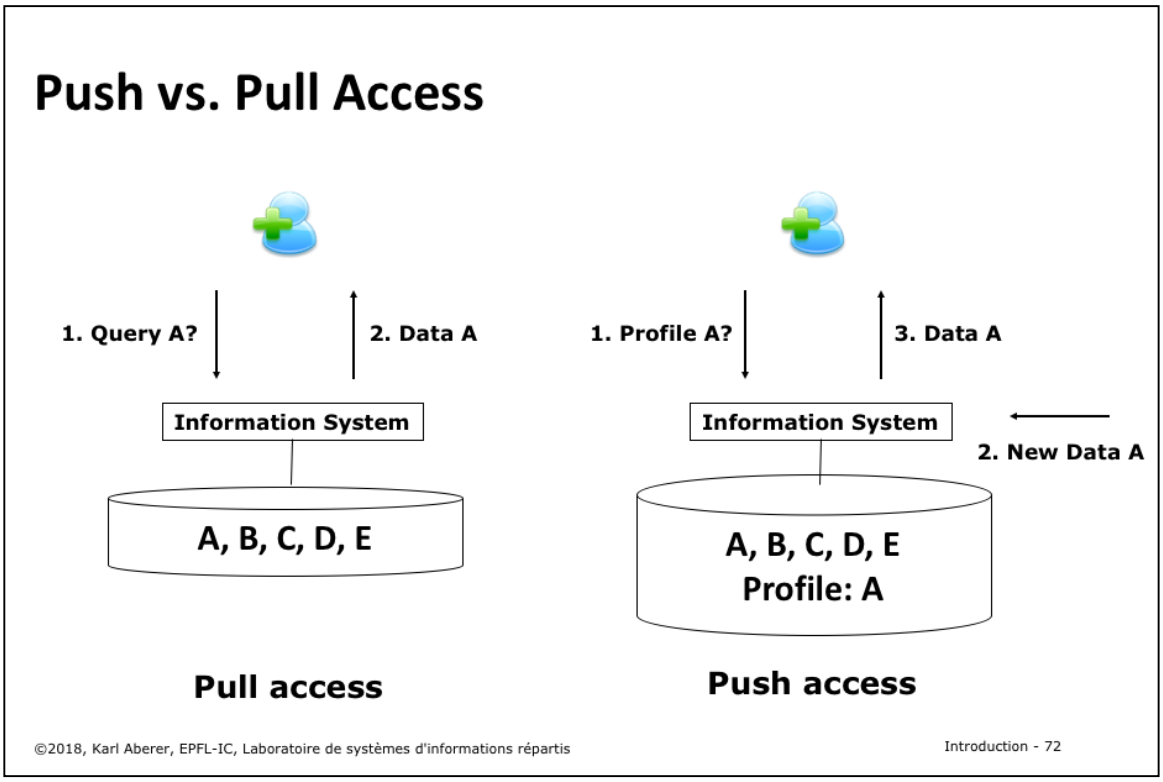
Data Caching



When data is transmitted in a network, e.g. during processing a query, it might be useful to keep a copy of the data that has been transmitted at the receiving node. This is called **caching**. Caching implies similar issues of maintaining consistency among different copies of the same data as replication. Different to replication with caching data is not proactively replicated, but as the by-product of other operations, in particular processing of queries. This may have the advantage that more likely data is replicated to a node that actually will be needed at that node (again).

If Google stores the search result of a Swiss client on a Swiss server after retrieving it from its US servers, it is doing ……

1. distributed query processing
2. data partitioning
3. data replication
4. data caching



Classical centralized information systems architectures are largely built upon the **client-server** model with a pull-based model of data access. Applications are running as clients of information system servers, and data is provided by the servers as response to data requests, e.g. queries. In distributed information systems information is not always requested by clients, but often proactively pushed to the clients by the information system. Reasons for this are both technological, e.g. exploiting better mobile networks by using braodcasting methods for information dissemination, or business related, e.g. for pushing advertisements to potential customers. Many protocols and systems, such RSS and Twitter, are using information push nowadays. Using information push changes dramatically of how data is stored, indexed and retrieved in information systems.

Communication Model

Standard model: **unicast**

- Point-to-point connection
- Request-reply protocol

Multicast

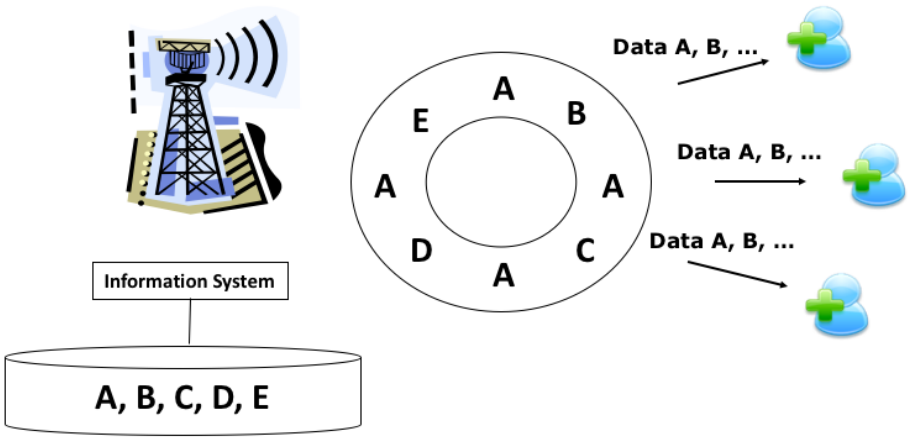
- Propagate requests to multiple receivers

Broadcast

- Users of wireless communication channels

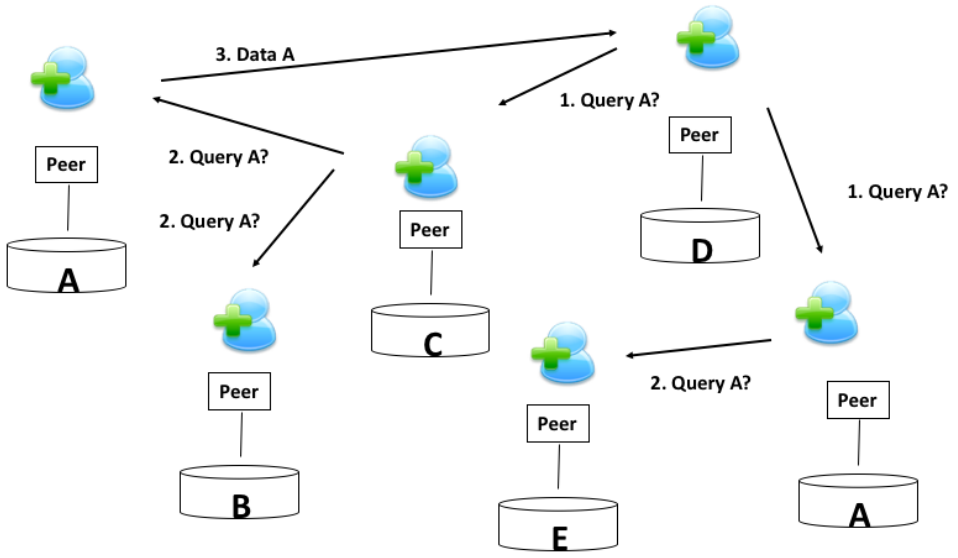
Apart from the standard unicast communication model in networks, implementing a request-reply protocol among two network nodes, networks permit also the use of other communication models. With multicast requests are transmitted to multiple receivers in the network at the same time. Multicast protocols typically create a tree-like structure along which the messages are transmitted in multiple hops to the receivers. Broadcast is a communication model that is particularly suitable for mobile communications. At no extra cost all nodes that are within reach of the mobile communication network can receive the same message from a server.

Example: Mobile Broadcast



One way to exploit the mobile broadcast communication model is transmitting all data of an information system to the receivers in the mobile network. The clients can listen to the data transmission and use the data need from the data stream that is disseminated via the mobile broadcast.

Multicast (Gossiping)



In P2P information systems no central server exists. One approach to access information in such a system is the use of gossiping, a multicast protocol. A peer sends a request to local neighborhood, his neighbors, which further spread the message till eventually some peer can respond to the request.

Information Dissemination

Control

- push – pull

Communication model

- unicast – broadcast - multicast

Event

- periodic - conditional (e.g. on change) - ad-hoc

To summarize, information dissemination in distributed information systems can be classified along three dimensions [Ozsu, Liu, Yang 1998]:

Control of the data exchange, communication model used, and the event triggering a data exchange. The events triggering an information exchange can be ad-hoc requests by applications or users (which corresponds to the traditional client-server model), a periodic event, or conditional events, e.g. triggered by an update to the data.

Which of those platforms is using multi-cast?

1. Bittorrent
2. Twitter
3. SMS
4. Google search

Heterogeneity

More Data - More Information?

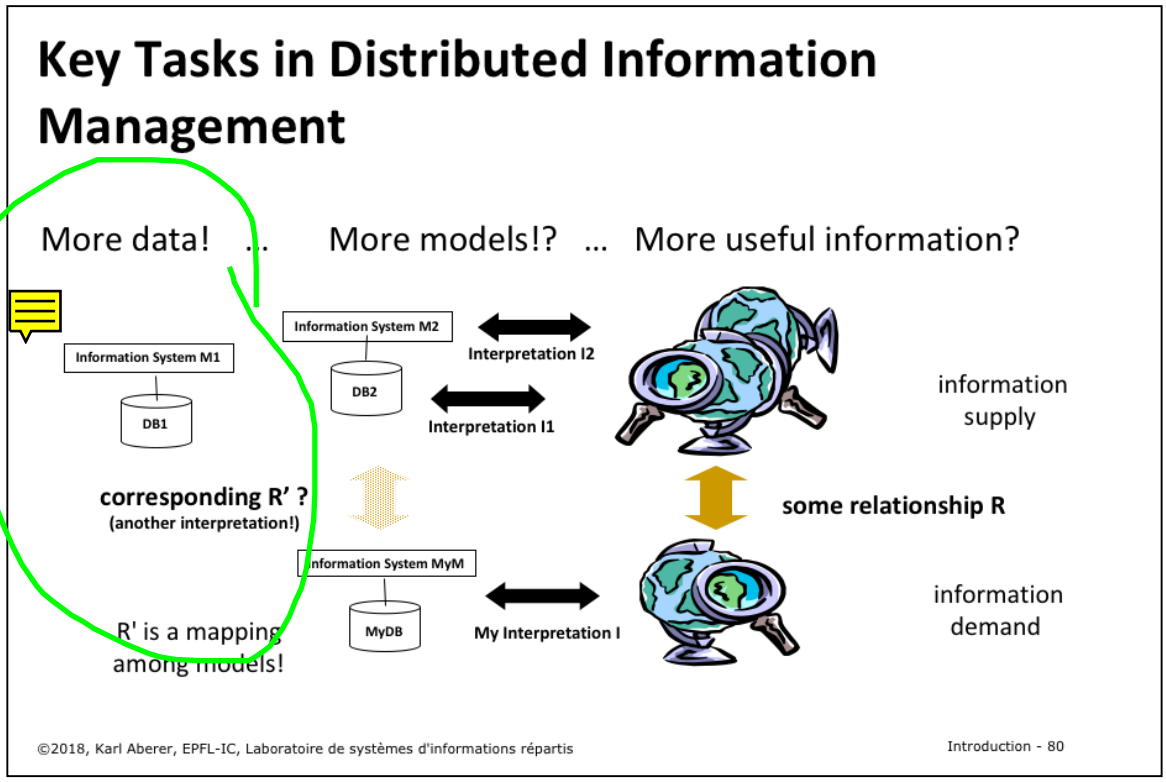
Data overload

- more data, disintermediation
- More useful information ?

Information starvation

- problem: *data supply* does not match *data demand*
- models used by data provider are different from models used by data consumer

We are living in the age of Big Data. More and more data is being produced, data is becoming more easily and directly accessible, data intermediaries are disappearing and users obtain direct access to data sources (disintermediation). The question is whether we have automatically more information. This is not clear. If the data cannot be properly interpreted and used by the data consumer, thus if the data supply does not match the data demand, the utility of the data remains limited. More data does not imply more information! The data needs to be made available according to a model that is useful for the data consumer. And this model might be largely different from the model used by the data producer. In fact, the data consumer might even not be able to understand that model. This is what is called information starvation.



This figure illustrates the problem of information starvation: there exist many information systems supplying data, but each having their own view on the world, which does not necessarily match the needs or understanding of a specific consumer. Every information system is interpreting its model differently with respect to the real world and relating to different views on the real world. Though there exists some relationships among all these views on the real world (let's denote it as R), and it surely implies some relationship R' among the different models used in the different information systems, the consumers of the information cannot easily understand the relationship R, and thus can also not easily relate their models to the models of others via the relationship R'. From the viewpoint of the data providers, introducing R' introduces a new interpretation of their data with respect to the model used by the data consumer.

The Problem

Semantic heterogeneity

- The same real world aspect can be modelled differently
- Relating different models (and thus different interpretations) requires often human intervention; human attention is a scarce resource !

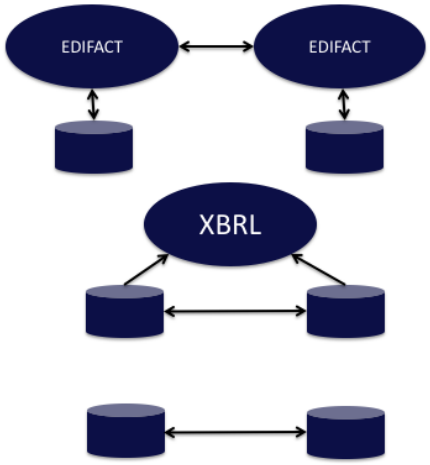
Relating different models used in information systems to each other is solving the problem of **semantic heterogeneity**. This problem is as hard as creating proper models for information systems and requires typically human intervention, thus the scarcest resource we have available.

Mapping: Three Approaches

Standardization
- Mapping through standards

Ontologies
- Mediated mapping

Mapping
- Direct mapping



Conceptually there exists three main possibilities of how to address semantic heterogeneity. A first approach is to map all the models everything to one common global model. This approach is taken with standardization. For example, EDIFACT is an international standard that models all concepts that are commonly used in business and trade. For exchanging information systems used in that domain map their data to EDIFACT and can thus exchange their information. A second approach, is to relate the model of an information system to a common model, frequently called ontology, and use this mapping to construct a direct mapping among the different models used in the information systems. A third approach consists of trying to construct directly a mapping among two information systems, without having any additional, shared knowledge in form of standards or ontologies among the two information systems.

Direct Schema Mapping

Assume all data represented in canonical data model (e.g. relational)

1 – detect correspondences (schema matching)

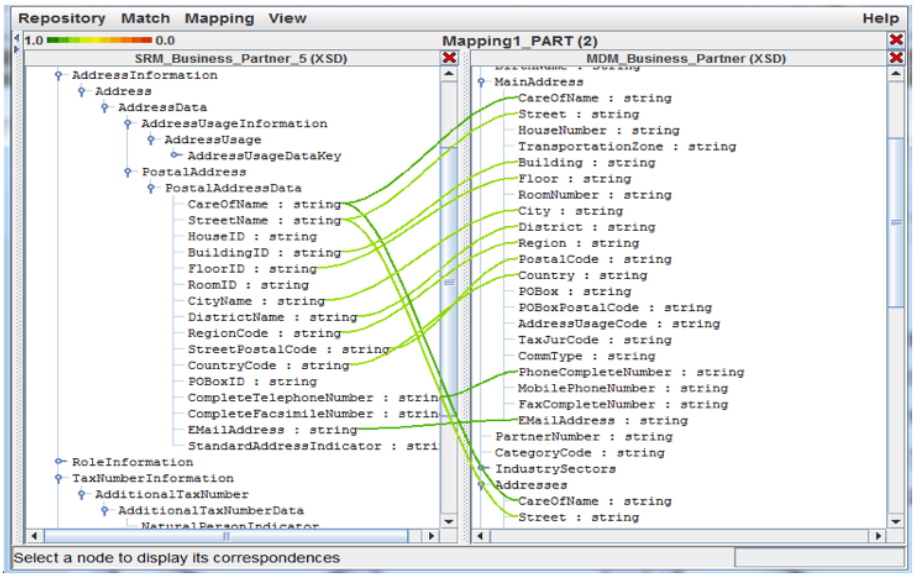
2 – resolve conflicts

3 – integrate schemas (schema mapping)

Mappings are frequently expressed as queries (e.g. SQL Query)

Creating direct mappings among information systems has been intensively explored for mapping data that is stored in relational and XML databases, and where the model is represented as a database schema. The problem is known as the **schema mapping** problem. Given two schemas of databases, first schema elements are identified that are likely to correspond to each other, i.e. that are likely representing the same real world aspect. This can be done by using many types of analysis using structural and content features of the database schema and the database. Tools for supporting this steps are called schema mapping tools. Once this step is performed some conflicting correspondences may occur, e.g. mapping the some concept in one schema to two different ones in the other. These not to be resolved, typically through decisions taken by humans. Once the correspondences are consistent, mappings can be automatically derived. The mappings are typically expressed as queries of the data manipulation language.

Schema Mapping Tools



This screenshot show a schema mapping tool that maps between different XML schemas. The lines indicate which attributes might be related to each other. One can easily see a lot of conflicts where the same attribute correspondes to multiple attributes in another schema.

Ontologies

Ontologies are an explicit specification of a conceptualization of the real world

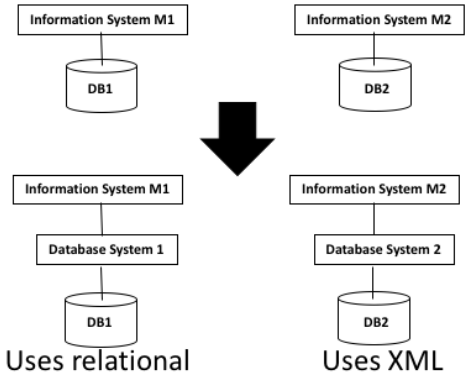
- different information systems agree on the same ontology
- relate their model/schema/data elements to the ontology
- mapping can be constructed via the ontology

One way to deal with semantic heterogeneity is to first agree on a common model of the real world. Such a common models are called ontologies. The provide a formal conceptual model of the world that has been agreed upon by a community. Forming such an agreement may be a difficult problem in itself! Once an ontology is established, it can be used like a “proxy” of the real world. Information systems can now specify the meaning of their models formally, by establishing an interpretation relationship with the ontology. Thus relationships among different models in different information systems becoming suddenly formally related to each other, which facilitates their integration by deriving mappings among the models.

More Problems?

Syntactic heterogeneity

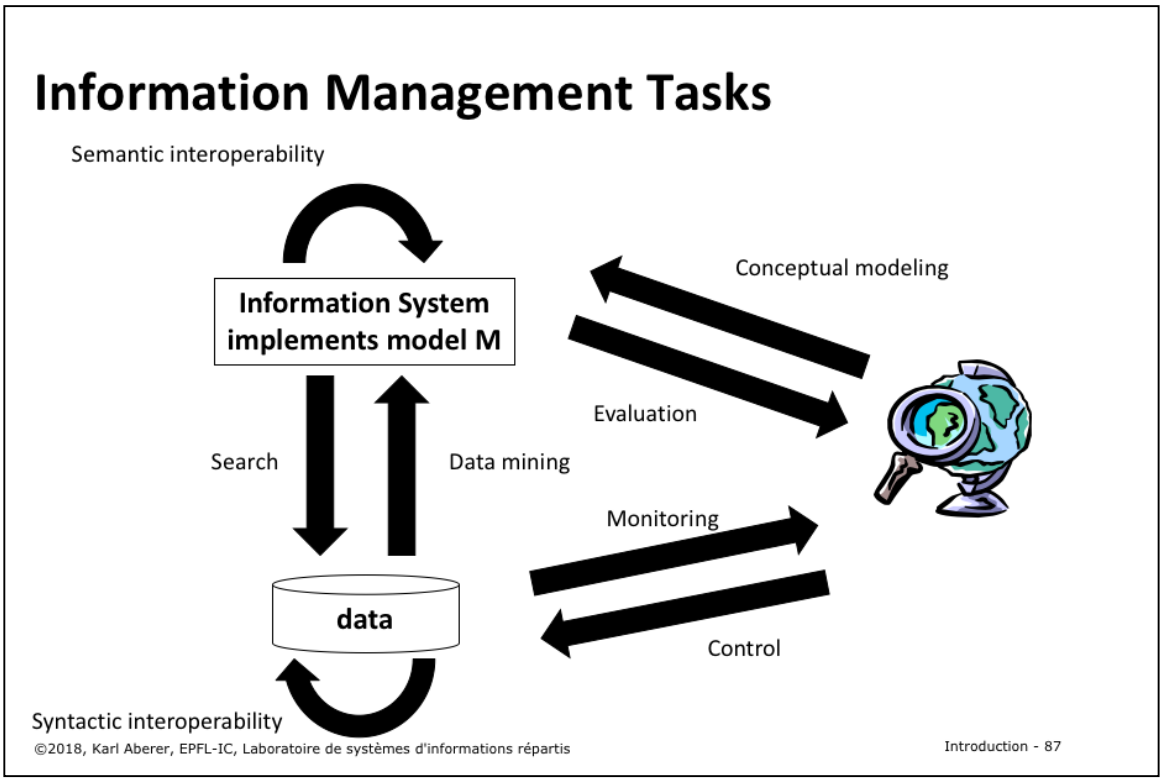
– The same data can be represented using different data models



Data Models might be different!

To complete the picture we have also to mention that heterogeneity among information systems cannot only occur due to the fact that the same real world aspects are modelled differently, but also due to the simple problem of using different underlying data models to represent the chosen model. In that case we are talking of **syntactic heterogeneity**. For overcoming syntactic heterogeneity mappings among different data models are needed. This problem is somewhat simpler than solving semantic heterogeneity, since the relationship among different data models can be treated completely within a formal context, but it is nevertheless not completely trivial, as different data models offer different data structures to store the same data and transformations might be algorithmically complex. The problem has been studied in different contexts including:

- Storage of programming language objects (e.g. Java) in database systems (e.g. relational)
- Integrating data from different types of data management systems, using different data modelling formalisms (e.g. relational, hierarchical, XML)
- Storing different types of data (e.g. XML, graphs, arrays) in generic databases management systems (e.g. relational)
- Exchanging data from database systems (e.g. relational) through document formats (e.g. XML)
- Representing graph-oriented models (e.g. RDF) as documents (e.g. XML)



Semantic and syntactic interoperability are two additional tasks in information management that we can add to our global picture.

Creating a web portal for comparing product prices requires to address

1. Syntactic heterogeneity
2. Semantic heterogeneity
3. Both

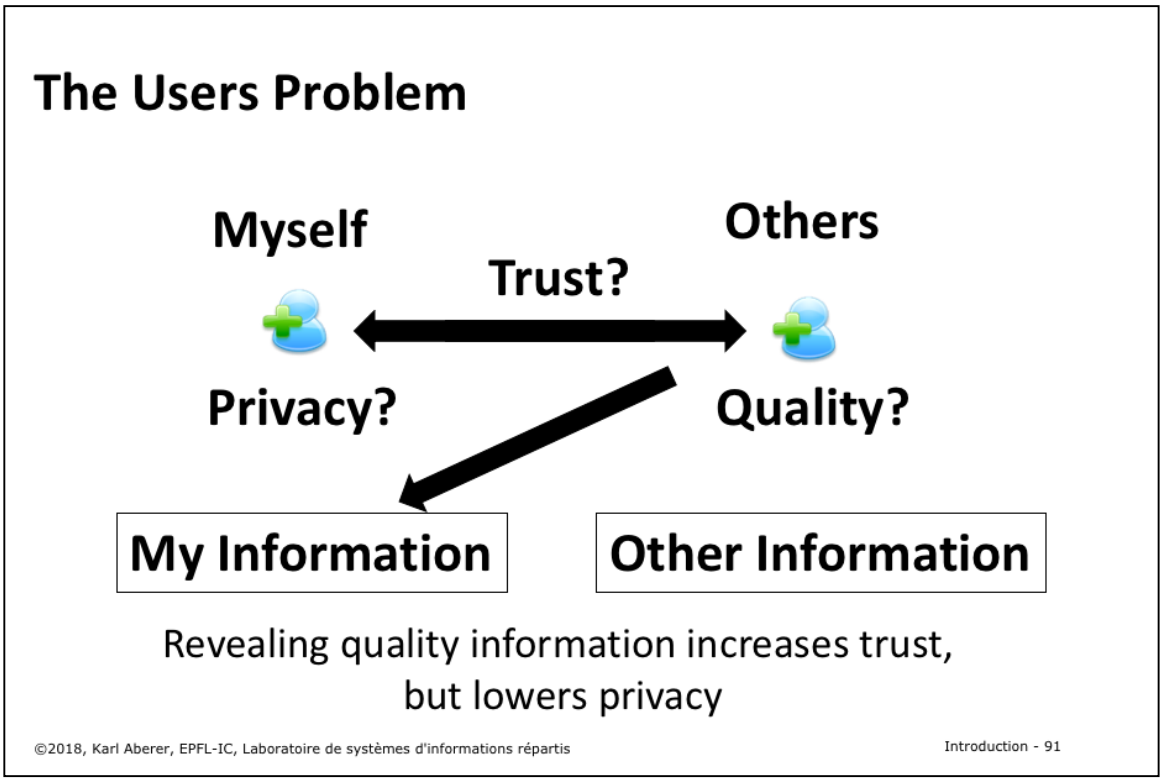
An ontology is a …

1. database
2. database schema
3. data model
4. data modeling formalism
5. model

Autonomy

Autonomy

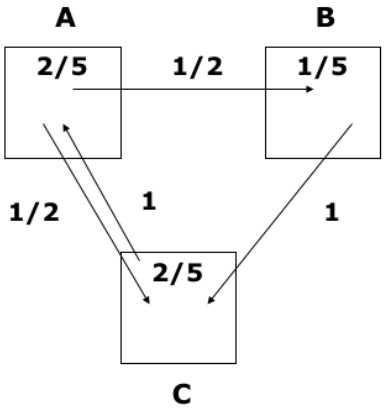
1. Is the cause of semantic heterogeneity
2. Requires mechanisms, such as transaction management, to coordinate access to data
3. Both of those

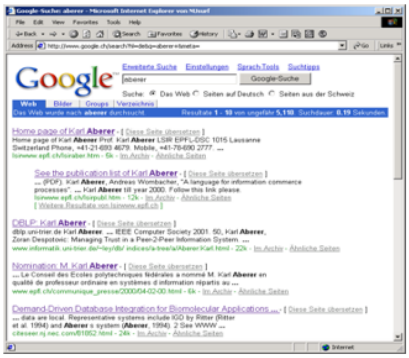


In an setting where different autonomous users exchange information issues related to the utility of information come into play. As users have their own private interest, they have to consider them in interactions with other users. For example, when receiving information from another user, a fundamental question is whether the information can be trusted. It might be that the other user might have an interested to provide wrong information in order to incite us to certain behaviours. When providing information to another user a different problem needs to be considered. Can we trust the other user to use the information correctly, or will he use it in ways that could be damaging to us. This is the privacy problem, and it is receiving in the information society huge attention. The two problems of information trust and privacy are also linked to each other. The more quality information we reveal the more trust we may expect, but the more we also put out privacy in danger.

Evaluating Quality of Information

Recommendations (e.g. Google PageRank)

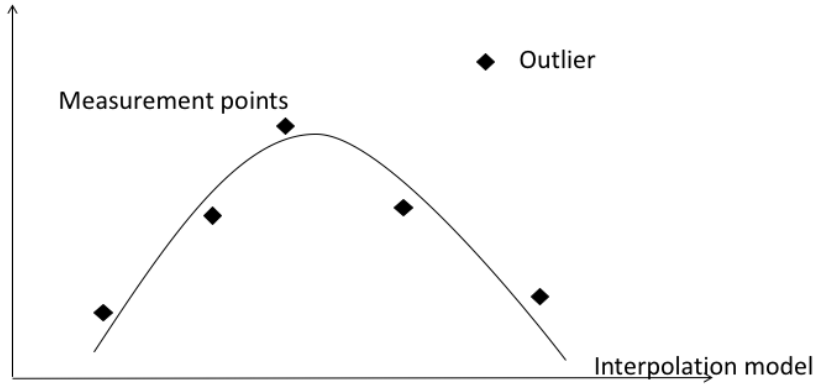




One way to evaluate the quality of information, and thus the level of trust we can have in a user providing information, is to share recommendations with other users on how they perceive the quality of this information. This is what, in principle, Google has been doing by introducing Pagerank, a method to assess the quality of a Web page by considering the Web links that point to that Web page. The underlying idea is that the more Web pages refer to one page, the more trustworthy it is and at the same time also to consider the trustworthiness of the recommender itself.

Evaluating Quality of Information

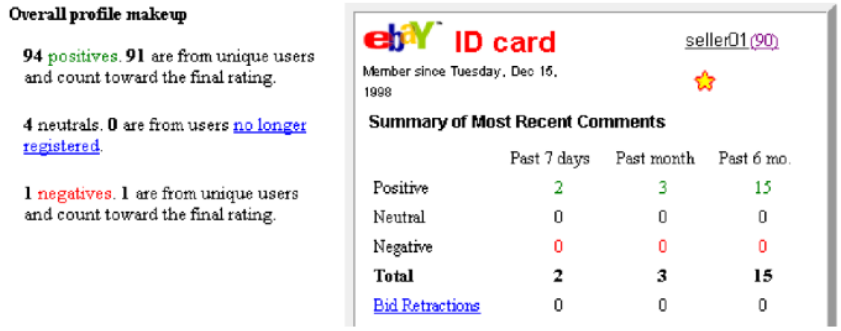
Consistency (e.g. temperature with model)



Another method to evaluate the quality of information, and thus of an information provider is to compare the information provided to a model we trust in. Assume we obtain temperature measurements from different information providers and we have a reliable method to interpolate those measurement values. Then, if we receive a value that is highly implausible as it strongly deviates from the model (a so-called outlier), this would lower the trust into the provider of this information.

Evaluating Trust

Reputation-based trust: if users behaved honestly in previous interactions, they will do so in the future



Overall profile makeup

94 positives. **91** are from unique users and count toward the final rating.

4 neutrals. **0** are from users no longer registered.

1 negatives. **1** are from unique users and count toward the final rating.

ID card — seller01 (90)

Member since Tuesday, Dec 15, 1998

Summary of Most Recent Comments

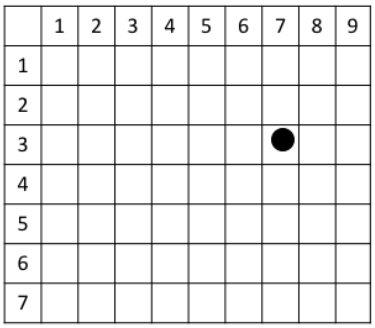
	Past 7 days	Past month	Past 6 mo.
Positive	2	3	15
Neutral	0	0	0
Negative	0	0	0
Total	**2**	**3**	**15**
Bid Retractions	0	0	0

One way to evaluate trust is to analyse earlier behaviours of users. This is, for example, widely used in ecommerce sites, where users can provide ratings for vendors. The underlying assumption is, that if vendors have behaved well in the past they will also do so in the future. From a vendors perspective such ratings of course foster honest behaviour, as negative ratings would affect the future business. Evaluating trust on such histories of behaviours, is called reputation-based trust, where the reputation is based on or corresponds to the data gathered about a user.

Protecting Privacy

Example: location privacy – obfuscation methods

- Perturbation: (3,7)
- Adding dummy regions: (3,5), (1,4), (6,3)
- Reducing precision: (2,5), (3,4), (3,5), (3,6), (4,5)



In order to deal with privacy, methods such as obfuscation are use to provide sufficient information to obtain a useful service, but not so much that sharing the information may be harmful. Here we illustrate different methods that could be used to obfuscate the location that is reported, e.g. when using a mobile service. Other methods to protect privacy include access control and data anonymization.

Distributed Control

Self-organization

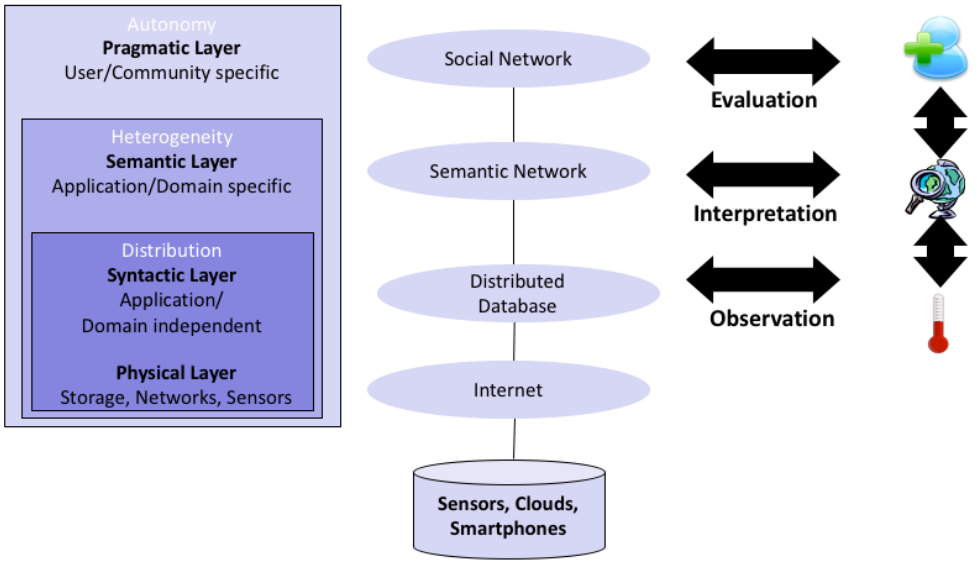
- Coordination in large-scale systems needs to be decentralized for scalability reasons
- Solutions
  - Decentralized optimization, e.g. economic resource allocation
  - Decentralized information dissemination, e.g. gossiping

A big challenge in very large-scale information systems with autonomous participants is of how to realize distributed control. Central coordination mechanisms, with a single node coordinating all the participants to task do not scale with a growing number of participants. Alternatively decentralized coordination mechanisms, or so-called self-organization mechanisms can be employed. We will present in this course such mechanisms in the context of peer-to-peer information systems.

Trust is …

1. a quality of information
2. a quality of a user
3. a quality of the relationship among user and information
4. a quality of the relationship among users

Refined View of a Distributed Information System



Exercise

Big Data is often characterized by the four concepts of Volume, Velocity, Variety and Veracity

1. Inform yourself what is meant by those concepts
2. Identify from this lecture four problems / methods that are related to each of those four concepts