Introduction:
Distributed Information Systems – An Overview

Today's Questions

1. What is an Information System?

3. What is a Distributed Information System?

What do you think?

- What is an information system ?

- What is an information system doing?

Identify which computer applications that you are encountering in your professional and private life you would consider as information systems? What does them make different from other kinds of computer applications?

Course catalogue

Electronic library

CRM SYSTEM

Book?

GIS

Banking system

Search Engine, Google

Mobile services

The Web

Facebook

Email? Webmail

Operating system?

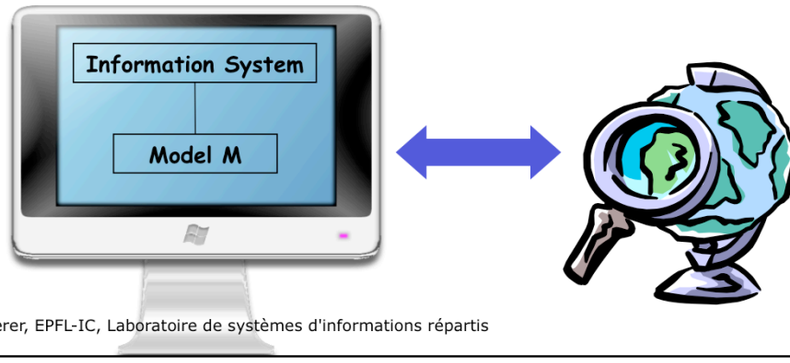
1. Information Systems

- The classical examples
  - Organizational databases
  - Business process management systems
  - Geographic information systems
  - Text retrieval systems
- More recent types of information systems
  - 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.

Classical information systems have been largely developed for the needs of large organizations and businesses. 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 evolving. Most recently the generation of huge datasets (e.g. from clickstreams or sensors) is giving rise to a fundamental change of the role of information systems, as the no longer only are interactively used by humans, but many processes (in business and science) are becoming increasingly data-driven, by applying machine learning and statistics methods.

Definition of Information Systems

- *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**).
- Remark: This is different from other software, e.g., a communication system, which enables the exchange of data



There exists no generally accepted, formal definition of what an information system is. For sure, information systems are pieces of software. However, one can also quite safely say 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).

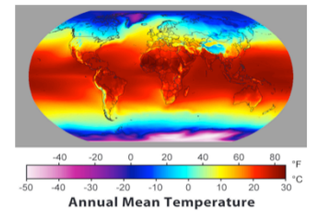
purpose: every information system has an entity (human, computer) that makes use of it. It does so, in order to perform a certain task related to some aspect of the real world (e.g. making a decision, performing a computation etc.).

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.

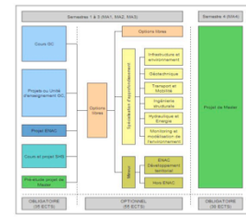
model: what a model is and what is its role we will explore in more detail in the following.

Examples of Real World Aspects

- A physical phenomenon: temperature map



- An organizational structure: university



- A human artifact: documents

What do models in information systems represent? About anything we (humans) can think about. It can represent phyiscally phenomena, organizational structures, but also artifacts produced by humans such as documents. Try to find more examples of what information systems are used for.

Models

- *What is model?*
- A model is a mathematical structure consisting of **constants** and **functions** (or relations) and **axioms** that have to hold
- Remarks
  - we can interchangeable use "functions" or "relations"
  - the constants are often called identifiers, we can think about them as names, or values
  - a central task of information systems is to give names to things (identification)
  - the axioms are also called constraints
  - some information systems support only very limited or specific kinds of models, others very generic models

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), the functions to provide properties of these things, and the axioms provide rules or constraints that state which properties are possible and not.

Examples of Models

- Temperature map
  - Constants: coordinate values, temperature values
  - Function: T[x, y]
  - Axiom: -60 < T[x, y] < 60
- Organisational structure
  - Constants: names of people and units
  - Function: memberOfUnit(name)
  - Axiom: each person belongs to at least one unit
- Documents
  - Constants: document identifies, text (bytestring)
  - Function: similar(doc_1, doc_2)
  - Axiom: 0 <= similar <= 1

Fo 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 realtionships among entities (basically first order logic), whereas documents can be modeled by their degree of similarity.

Models used in Information Systems

- Entity-Relationship Model (also: UML, RDF)
  - Used in most business information systems (conceptual model)
  - Based on first order logics
- OWL
  - Used in Semantic Web
  - Based on description logics (fragment of first order logics)
- Vector Space Model
  - Used in information retrieval
  - Based on linear algebra
- Probabilistic Models
  - Used in scientific and engineering applications, as well as text retrieval
  - Based on probability theory
- Process Models
  - Used in business information systems
  - Based on abstract automata or Petri-Nets

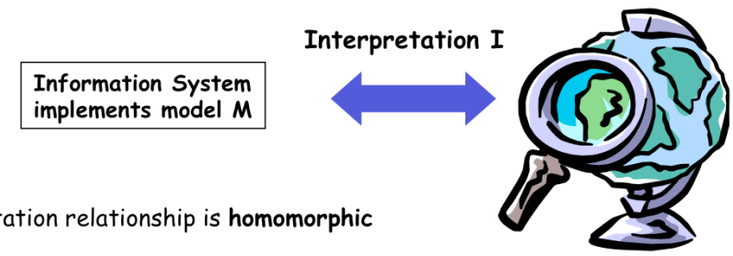
Here we list some examples of formal models used in information systems:

- 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.

Models

- *How do we know that a model is a model?*
- The model is linked by an interpretation (relationship) to the real world



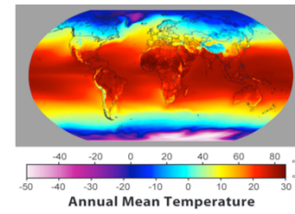
- The interpretation relationship is **homomorphic**

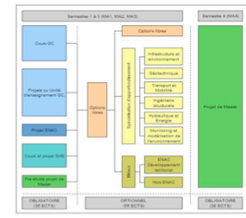
Having a model is fine, but how do we know that a model is a model of the part of the real world we want to represent. From an abstract perspective the answer is simple: there exists a relationship (called interpretation) that maps (or connects) « things » from the real world to elements of the model (e.g. the constants), and properly preserves all relationships or functions. The property to preserve relationships or functions expresses that the interpretation relationship should be homomorphic.

Formally: an interpretation relationship I that maps constants of a (real-world) domain D to a model domain M (I: D->M) is homomorphic if it is true that if f(x) = y then f_I(I(x)) = I(y), where f_I is the function in the model that represents the real-world function f.

Difficulty of Correct Interpretation

- Problem: very hard determine or even formalize I
- Environmental science is trying very hard to determine a temperature model for earth
- Whole armies of business analysts deal with the problem to correctly model an enterprise
- The whole research field of information retrieval addresses the « correct » intepretation of documents





Questions

- An interpretation relationship
  1. provides users of information system the purpose of the system
  2. allows to connect constants in a model to real-world entities
  3. is part of a model managed by an information system

- Functions in models
  1. Are always computable
  2. Can always be represented equivalently as relations
  3. Can be constrained by first order logic axioms

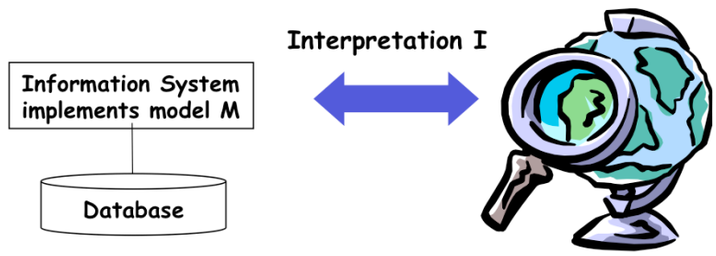
Functions

- *How are functions represented in information systems?*
- Functions can be represented by giving a specification or algorithm (implicitly) or by enumeration (explicitly).
- Example: $f(x) = x^2$ vs. $f(1) = 1$, $f(2) = 4$, $f(3) = 9$ etc.
- Information systems strongly emphasize the explicit representation (since many aspects of the world are not algorithmically defined, e.g., how to compute the birthdate of a person?)
- Computed functions play nevertheless an important role, they are called e.g. queries, views, user-defined functions etc.

For now let's assume that the model we have chosen is adequate. A first question we might ask, is how to represent functions. Actually, there exists principally two ways to do this, by explicitely enumerating all function values for each possible argument, or by providing a specification or algorithm to compute the function value from a given argument. Both ways are actually used in information systems, however, a lot if not the primary significance is given to functions that require explicit enumeration. Many facts about our real world, though they are clearly functions (e.g. the birthday of a person) cannot be computed (e.g. from the name of the person). Nevertheless also computed functions play an important role in information systms, e.g. computing the age of a person from its birthday. Such functions appear under many different names, such as queries, views, user-defined functions etc.

Databases

- The elements of an enumeration of function values are called **facts** (or data) and the set of facts is called a **database**.
- Remarks
  - this is a key aspect that distinguishes and information system from other computer programs/applications; however, the boundary is fuzzy



Since explicit enumeration of function values plays such an important role, such enumerations have a name: databases. Databases are collections of facts, where facts (or data) are definitions of function values. Many computer programs actually use databases, without really calling them that way. So the boundary among systems that use databases such as information systems and other programs is somewhat fuzzy. Ot from another perspective, almost every practical application uses at least some small database.

Data Models

- *How is a model represented in a computer system?*
- A model is represented using a **data model**. A data model supports the representation of constants, facts and functions.
- *What is a data model?*
- The term data model is ambiguous, it may mean **data model theory** and **data model instance**.

Now that we know WHAT a model is, we still have not answered the question of how the model is represented in the computer system. For that purpose we require models that can be « understood » and « processed » by a computer. These models are called data models. In this context we have to be very careful about terminology, as data mode is interchangeable use to designate two different things: on the one hand what we can call a data model theory, respectively a framework or language to express data models, and on the other hand what we can call a data model instance, respectively one specific instance of a model expressed in a data model theory.

Data Models

- A data model theory (logical data model) is a formal description of how data may be structured and accessed in a computer system
- A data model theory has three main components:
  - **Structure**: a collection of data structures, which are used to create databases representing the facts of the database.
  - **Integrity constraints**: a collection of rules governing the constraints placed on these data structures to ensure integrity.
  - **Manipulation**: a collection of operators, which can be applied to the data structures, to update and query the data, contained in the database.
- A data model instance, uses a data model theory to specify a data model instance for some particular application, i.e. a specific model. The specification is also called a (database) **schema**.

The three components of a data model theory are the following: (1) the structural part describes of how constants and functions are represented in data stuctures. This enables the representation of facts of a database. (2) the integrity constraints are the axioms that have to be respected by the facts. (3) the 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.

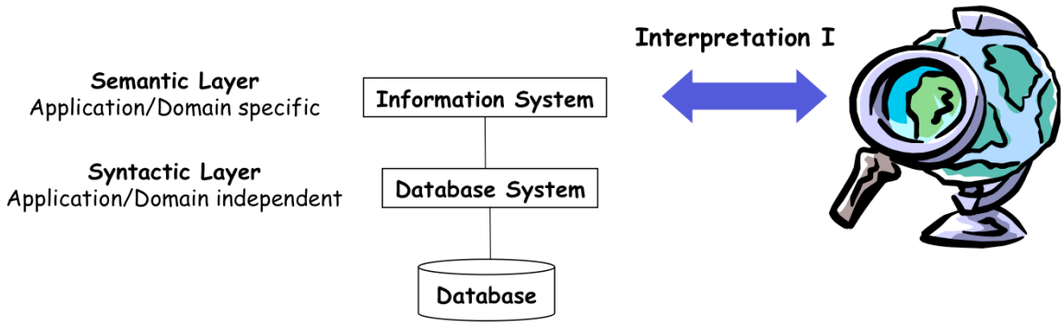
Examples of (popular) Data Models

- Relational data model, Object-relational model
  - Data structure: relational tables
  - Operations: SQL, function extensions
  - Schema: relational schema (mandatory)
- Graph data model
  - Data structure: graphs
  - Operations: navigation and graph manipulation
  - Schema: graph schema
- XML (document model)
  - Data structure: XML trees
  - Operations: XQuery, DOM
  - Schema: XSchema (optional)
- Relational data streams
  - Data structure: tuple streams
  - Operations: SQL plus windowing operators
  - Schema: relational schema
- Associative array
  - Data structure: key-value
  - Operations: get, put
  - Schema: no schema

These are some examples of frequently used data models in database systems. The relational, graph and document models are all similar in the sense that they provide similar ways of expressing relationships and properties of entities and that the support the notion of schemas. Many information systems rely on much simpler, and generic data models respectively data types, such as associative arrays. With the increasing use of temporal data, extensions of models that originally were designed for static data to support also time-dependent data have been introduced.

Database Systems

- A database system is a system that supports a data model
  - Database systems allow to implement many information systems
- Some information systems support only a single data model instance (or a limited set) and do not make use of a database system



Having introduced the notion of data model, we can refine our view of information systems. An information system is (typically) based on a database 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 mamy practical systems this separation is not that explicit or observed at all, but it is very useful to keep in mind for better understanding the architecture of systems and the separation of different concerns (i.e. those related to a specific applications, and those that are domain independent. We can understand the separation among information systems and database systems also as one among syntax and semantics.

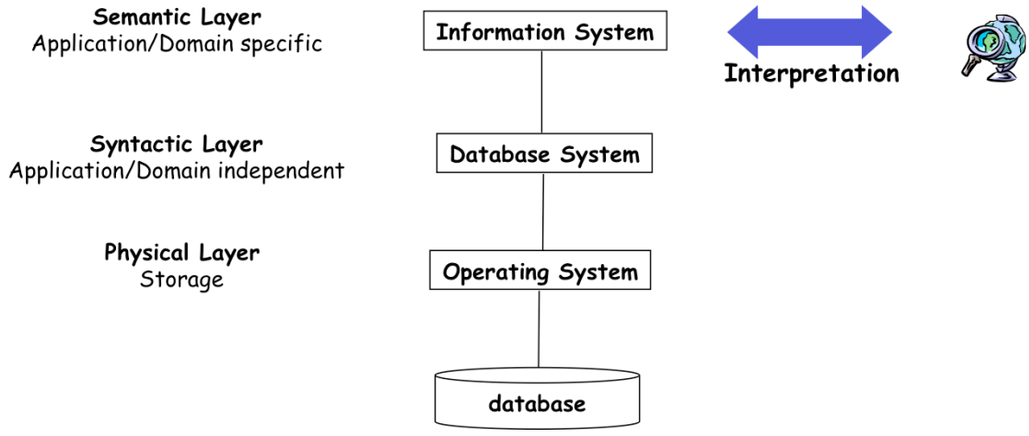
Implementation of Data Models

- *How is a data model represented in a computer system?*
- The data is stored using encodings of constants and data structures exploiting addressing in the storage medium (either main memory or on disk).
- Remark:
  - the field of database management systems (and others) have investigated many methods to make storage and access to stored data efficient, and to maintain consistency of data under various circumstances (failures, parallel access)

Data models are formal models, that can be directly represented and supported within a computer system. This means there exist data structures, that support the representation of constants and facts, programming languages that support the manipulation of databases while observing consistency constraints. Database systems are software that is specifically supports storage and manipulation large databases using a specific data model, such as the relational data model. Many of the models used in information systems, as discussed earlier, can be represented in to mapped to such data models, and thus database systems are often used for the implementation of information systems. However, there many information systems implement their native data management one example being text retrieval or Web retrieval systems.

The implementation of a specific data model leaves still many options of how the model is physically represented within a computer. This concerns for example the binary encoding of fact, the physical layout of those on a storage medium (both memomry and disk) and many mechanisms to ensure the integrity of the data and the efficiency of access.

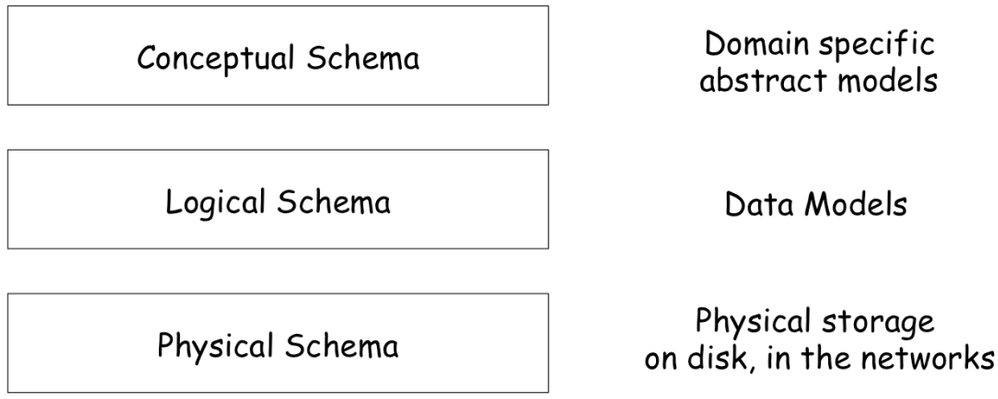
Refined View of an Information System



In view of the design principles of data management systems we can provide now a further refined view on the typical information system architecture. We see that the conceptual layer, or semantic layer as we will call it form now, is concerned with the modeling of the real world. In order to implement the model efficiently, different layers are distinguished. The syntactic layer provides a data model (a syntax) 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 separates the problem of efficient implementation of data management tasks from the logical specification of data models and schemas for specific information systems.

Modelling Architecture

- Defined as standard by ANSI in 1975



The distinction between general formal models and data models is also captured in the ANSI modeling architecture, which distinguishes among the conceptual level and the logical level. In addition, it identifies the physical level, that is concerned with the representation of data on the storage media, at that time in particular disks, nowadays also increasingly the network.

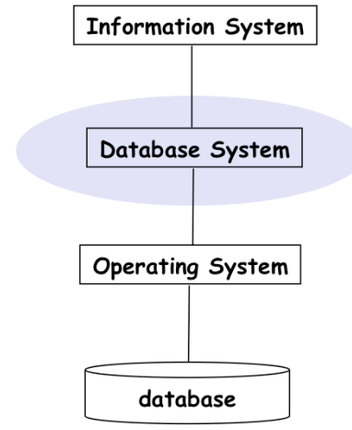
Questions

- What is not specified in a data model theory
  1. The structure of a relational table?
  2. A disk storage layout for the tuples of a relational table?
  3. An query to retrieve entries from a relational table?

- Semantics deals with issues related to
  1. The efficient access to data in a database
  2. The specification of data structures to represent function
  3. The relationship of names to real world entitites

What do you think?

- What are typical problems of *data management*?



In the following we want to provide a short summary of the main tasks of data management. Are aware of some of the functions and services a data management system provides and the techniques that it uses to realize them?

Key Data Management Tasks

- Efficient management of large amounts of data (Scalability)
  - efficient storage schemes (e.g. sparse – dense)
  - efficient access schemes (e.g. indexing)
    - exploiting different media (e.g. memory - disk)
  - Supporting retrieval, restructuring and updates
- Ensuring consistency of data under updates and failures
  - on some persistent medium (e.g. memory, disk, flash, tape)
  - independent of lifetime of programs, access pattern, failures
  - consistent updates

The amount of data that is managed by an information system is usually very large, which obviously implies important challenges for data management to make the system efficient.

One key function of an information systems is to “manage” the facts: managing means to maintain them and to perform operations on them, i.e. to maintain a state and to perform state transitions. This is what also programs do with transient, but in contrast the state in a database system is maintained independently of the life time of any application program. We also say that the data is persistent. To realize this and some medium is required to maintain the state, such as a disk. One major challenge is to maintain the data consistent even in case of failures of the system or system components.

Problem: Scalability for Large Databases

- Store data on available storage medium in an efficient manner
  - Blocks, files, network nodes, …
- Provide efficient access to data for specific addressing methods (Indexing)
  - For different types of queries: predicates on attributes, paths, …
  - Considering typical access patterns
  - Exploiting data access structures (tree, hash table etc.)
- *Example*: B+-Tree
  - tree nodes match block size of storage systems
  - tree is balanced
  - all operations (search, update) logarithmic

Database research has invested a lot of energy to continuously increase the efficiency of database systems.

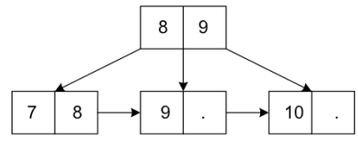
Two key problems in accessing large databases efficiently can be identified:

1. how can one make optimal use of the existing infrastructure, such as disks and networks, by exploiting it's physical characteristics,
2. and how can data be efficiently accessed, located and searched in large databases.

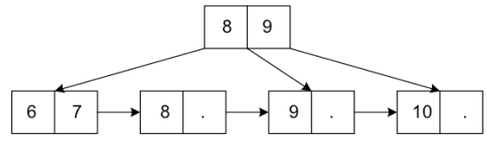
B+-Tree's are one well-known example of an approach that addresses both aspects. The both support efficient search, but further more – and that is a difference to main memory based index structures – they are concerned with the efficient layout of the tree structure on secondary storage.

One important aspect in making access to large databases efficient is exploiting the statistical properties of the data. This will also be a recurring theme in this lecture where knowledge about frequency of certain data, frequency of accesses and other statistical information is exploited in order to optimize the data access.

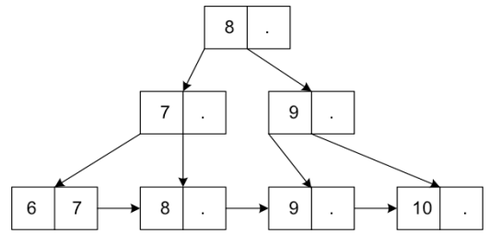
B+-Tree



insertion of 6



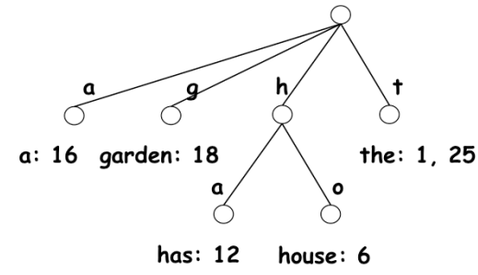
rectification of tree



This example illustrates one interesting aspect in the design of B-Trees: B-trees always remain balanced trees and have a fixed fan-out. When, for example, data is inserted the algorithms for accessing the B-Trees assure that these properties are maintained. In this example this requires that the depth of the tree is increased after the insertion of a data item.

Tries

- Lot of data is represented as strings
  - Obviously: text (origin of name: reTRIEval)
  - But also: navigation paths, time series
- For string data there exist specific index structures, e.g. tries (also called prefix trees)
  - ordered tree data structure
  - used to store an associative array
  - keys are strings
  - one tree node for every common prefix
- Example: "the house has a garden the"
  - Indexed keys (prefixes): a, g, ha, ho, t



Tries are another example of a frequently used index structures in information systems. They are not balanced, as B-trees are, but on the other hand of properties that support more complex searches than B-Trees. In addition to lookup of specific values they also support for example prefix queries, and more generally even regular expression queries. For many application indexing Web information this is very useful, as it can be exploited to support text searches or searches over navigation paths. We will encounter this data structure several times later in the lecture.

Problem: Safe storage and updates

- Maintain consistent state
  - in the presence of multiple users (Concurrency)
  - in the presence of failures (Recovery)
- *Example*
  - acc1 is a data object (tuple, element, object, …)
  - T1 and T2 are different users, what goes wrong ?

Transactions		**State**
T1:	**T2:**	**acc1**
read(acc1)		20
acc1 := acc1 + 10	**read**(acc1)	20
	acc1 := acc1 + 20	
	write(acc1)	40
	commit	
write(acc1)		30
Commit		

The second major technical challenge for database systems is the support of consistent access, even when multiple applications access the system simultaneously. Here we illustrate a potential problem such parallel access 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 program is accessing an account, no other program 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.

Physical Data Independence

- The same logical schema can be physically realized in many ways; operations on the logical schema can be implemented in different ways
- Declarative Querying, Query Optimization
  - abstract from finding optimized access paths

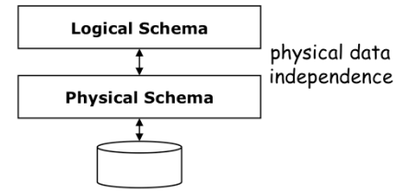
PGA Tour

ID	Name	Earning	Driving Dist
1	T. Woods	6496	294
2	S. Garcia	2319	285
3	E. Els	3180	279
4	D. Toms	2534	271
5	J. Daly	630	305
6	C. Howell	99	290

{ p | Earning(p) > 6000 and Driving Dist(p) > 280}

first select all p with Earning(p) > 6000 or
first select all p with Driving Dist(p) > 280

- Transaction Management
  - abstract from parallel access and system failures



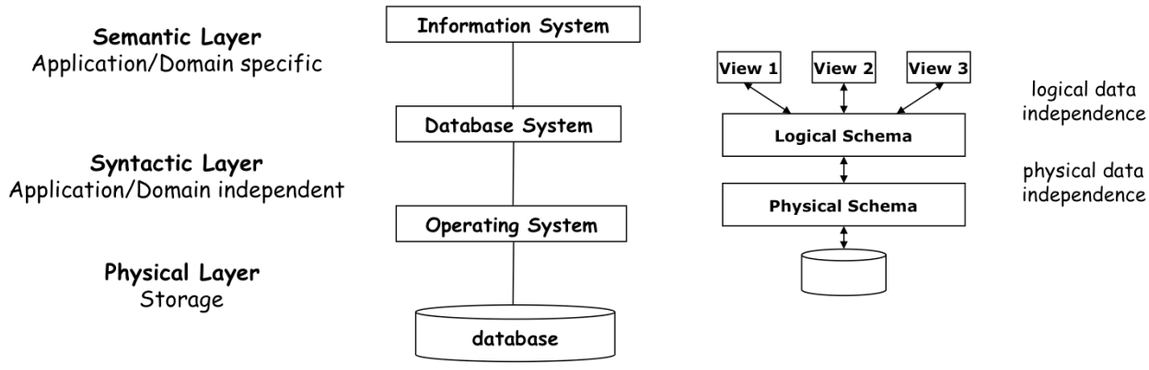
One of the central ideas in the design of database systems (and information systems in general) is data independence. Data independence means that applications and developers are provided with models and interfaces that provide and abstract view of the data that abstracts from many implementation details that are important for the efficient operation of the system but not necessary to specify the application logic. In database systems two forms of data independence are used.

Physical data independence means that database applications are defined in terms of a logical schema, without being concerned with details such as data access structures and transaction management. The database systems performs automatically optimizations to implement the logical functions in an efficient way.

With logical data independence different applications create their own views on the logical schema of a database. The advantage is that changes to the logical schema (e.g. adding an attribute to a table) do not (necessarily) affect the application, and so system evolution can be facilitated.

Logical Data Independence

- The same data can be viewed in different ways
  - Views compute a model on top of the data stored in a database
  - Corresponds to different interpretations of the same data



One of the central ideas in the design of database systems (and information systems in general) is data independence. Data independence means that applications and developers are provided with models and interfaces that provide and abstract view of the data that abstracts from many implementation details that are important for the efficient operation of the system but not necessary to specify the application logic. In database systems two forms of data independence are used.

Physical data independence means that database applications are defined in terms of a logical schema, without being concerned with details such as data access structures and transaction management. The database systems performs automatically optimizations to implement the logical functions in an efficient way.

With logical data independence different applications create their own views on the logical schema of a database. The advantage is that changes to the logical schema (e.g. adding an attribute to a table) do not (necessarily) affect the application, and so system evolution can be facilitated.

Example: Logical Data Independence

- University Rankings: based on same data

Fig. 12: Ranking of ETH Zurich and EPFL according to THE, QS, ARWU and Leiden Ranking

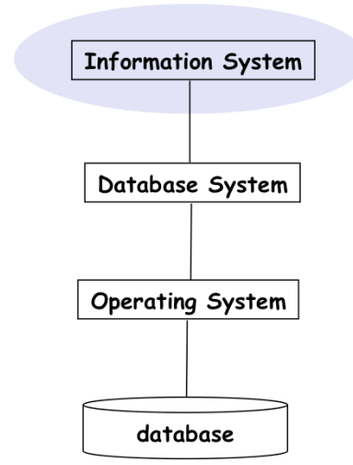
This example nicely illustrates logical data independence. Though raw data for university rankings is often the same (or similar), .e.g the ISI citation database, the rankings computed may vary widely. The different resulting rankings corresponds to different meanings of « quality » that are derived from the same data.

Questions

- Statement 1: Physical data independence expresses that the same logical operation can be executed in different ways on a storage medium
- Statement 2: Logical data independence expresses that the same conceptual view on a database can be represented in different ways in a logical schema
- Which is correct?
  1. Both above statements are correct
  2. Statement 1 only is correct
  3. Statement 2 only is correct

What do you think?

- What are typical problems of *information management*?



We understand now some of the important data management tasks, but are there other tasks that are part of information management systems?

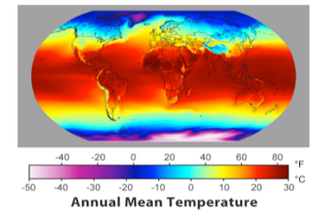
Key Information Management Tasks

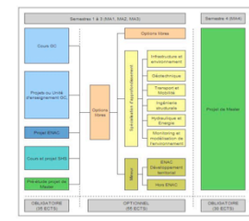
- *Model* the domain of interest
  - Consult users, analyze information
- *Retrieve* information of interest to user
  - Information satisfying conditions, being similar, recent, good quality, popular
- *Analyze* the data for regularities, hidden structure, patterns or outliers
  - Data mining, information extraction, image analysis
- *Monitor* real-world phenomena
  - Environmental monitoring, disaster warning, …
- *Control* processes based on the information
  - Real-time systems, business process management
- *Visualize* information for users
  - Graphs and plots, map-based interfaces, animations, …
- *Disseminate* information
  - To customers, friends, communities, ..

Here we provide a more comprehensive list of typical tasks that are performed in information systems. Solutions to solve some of them we will study later in the lecture.

Example: Modeling Methodologies

- The "data mining approach"
  - take an (observational) learning dataset
  - build a model from it
  - take an (observational) test dataset
  - compute how well the model matches
  - Present the model to users for evaluating utility
- The "database approach"
  - Consult the user for requirements
  - Develop a conceptual schema (e.g. entity-relationship)
  - Transform into logical schema (e.g. relational)
  - Normalize the logical schema (e.g. avoid redundancies)
  - Develop the physical schema (e.g. indices)
  - Deploy the application
  - re-consult the user and start over
- The "information retrieval approach"
  - ask human users for the relevance of information for a problem
  - model relevance by a retrieval algorithm
  - compare the results: recall, precision





The first task mentioned on the previous slide is modeling, and apparently it is a central task as it serves the purposes to develop the model that is used by the information system. An important aspect of modeling is to make sure that the model matches the expectations of the potential users of the system. To this end different methodologies have been developed that are typically associated with different types of information system. We mention here three of the most important ones:

1. the database approach: here from an explicitly available structure another explicit structure is generated. To specify the mapping usually first order logic used (though also other functions might be involved, such as manipulation of multidimensional arrays in online-analytical processing). This new structure is associated with an interpretation (e.g. "Pro" is a real-world concept)
2. the data mining approach: here the interpretation is based on the content. Implicit structures that are detected in the content are turned into explicitly available structures, for which presumably an interpretation exists. The example shows an example where the structure is given in form of a rule (effectively a kind of an additional constraint on the data) that has an interpretation.
3. the information retrieval approach: it makes use of implicitly

Information Management Core Task

Study the relationships among models and of models with reality

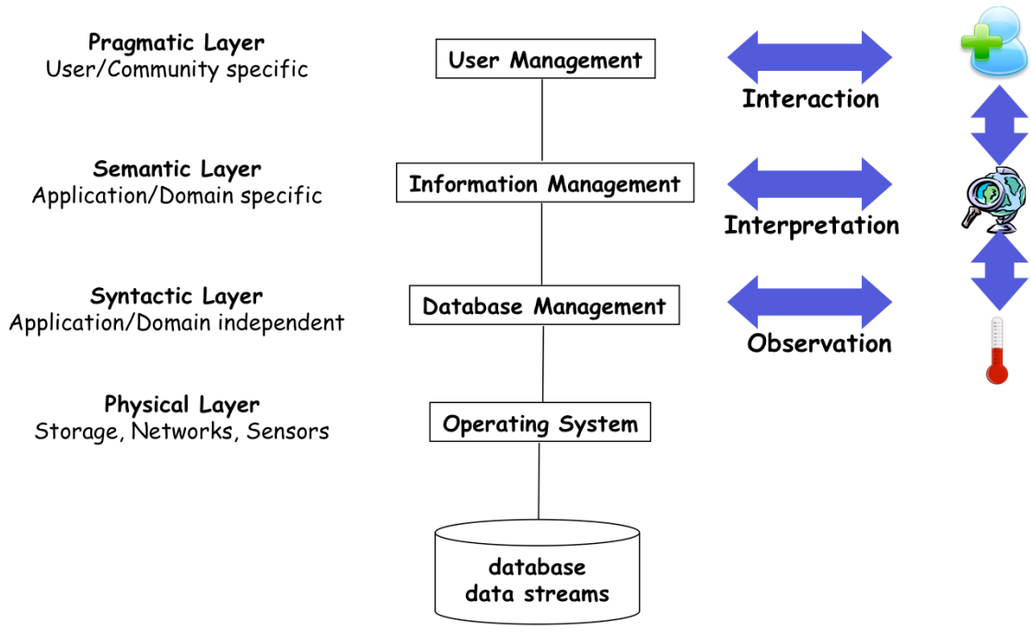
Key Information Management Tasks

- *Model* the domain of interest
  - Consult users, analyze information
- *Retrieve* information of interest to user
  - Information satisfying conditions, being similar, recent, good quality, popular
- *Analyze* the data for regularities, hidden structure, patterns or outliers
  - Data mining, information extraction, image analysis
- *Monitor* real-world phenomena
  - Environmental monitoring, disaster warning, …
- *Control* processes based on the information
  - Real-time systems, business process management
- *Visualize* information for users
  - Graphs and plots, map-based interfaces, animations, …
- *Disseminate* information
  - To customers, friends, communities, ..

We can make two important observations in this list of tasks:

1. Many (if not all) of these tasks involve the user in the one or other form.
2. Some of the tasks connect the information system directly to the physical environment, e.g. through a sensor or a camera.

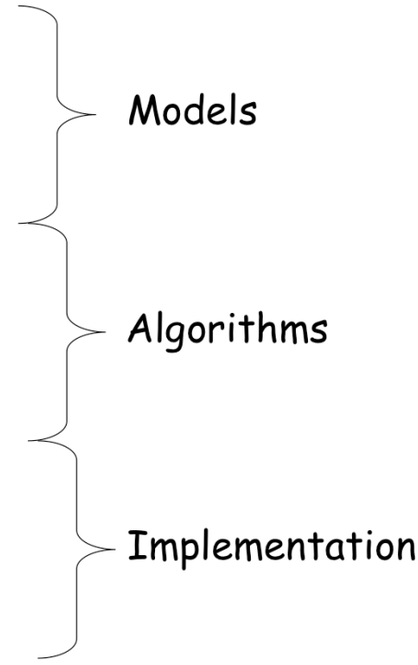
Refined View of an Information System



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 valuation and evaluation of the utility of the information provided. This layer has been mostly ignored historically, but is becoming increasingly technically relevant as more and more users and their behavior are modeled within the information systems, social networks being one of the prominent examples. In addition, information systems are not only processing human-generated data and content, but as illustrated in the previous example, sensor data that is directly captured from the real world. Of course this data needs a semantic interpretation (as illustrated) and therefore we associate the observations (before interpretation) with the syntactic layer of our architecture.

Summary of Problems

1. Data Models (Syntax)
   - Adequate representation and manipulation of data for different requirements: data models
2. Modeling (Semantics)
   - Evaluation: evaluate the suitability of a model for a purpose
   - Provide operations that produce desired interpretations
3. Scalability for large datasets
   - Efficient data access
   - Efficient information processing algorithms
4. Maintain the state safely
   - Storage and processing in presence of errors
   - Malicious behavior
5. Systems – Layered System Architecture
   - Provide different abstractions at different system levels
6. Methodologies
   - Approaching problems in a systematic way



This is a somewhat simplified overview of the main problems in data and information management we have identified so far: we can classify them into issues related to modeling (i.e. what are possible representations for the real world aspects we are dealing with), issues related to algorithms (i.e. how can we perform operations on the models efficiently and reliably), and issues related to the implementation (how can we build and use real software systems for using the models and algorithms). For each of these aspects, we can further distinguish two main categories: in modeling there is a clear distinction whether we are interested only in the syntactic issues (i.e. not considering the interpretation of the model) and semantic issues (taking into account the interpretation). In algorithms we can distinguish between issues of reliability (how can one safeguard against all kinds of adverse effects) and efficiency (how can the systems made scalable). With respect to implementation there are issues of architecting the systems and of using the systems in a systematic way.

Questions

- Which of the following modeling methodologies are data-driven?
  1. Data mining and information retrieval
  2. Database modeling and information retrieval
  3. Database modeling and data mining

- Pragmatics deals with
  1. The structure of symbols
  2. The meaning of symbols
  3. The utility of symbols

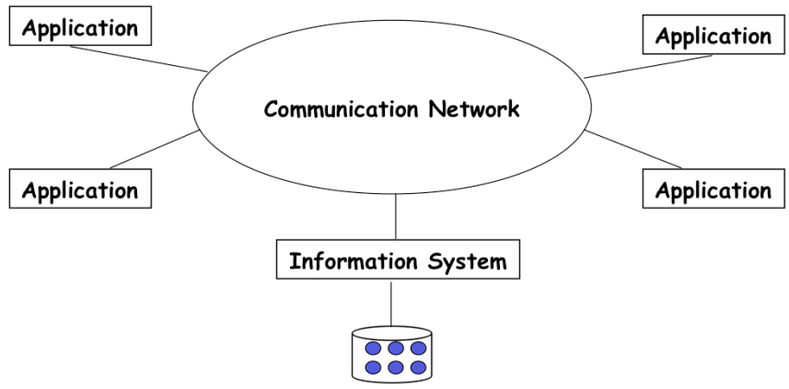
What do you think ?

- What is a distributed information system ?

Understanding the role and issues of information systems, are there any other significant problems we need to consider when we deal with a distributed information system. And, what is after all a distributed information system?

2. Distributed Information Systems

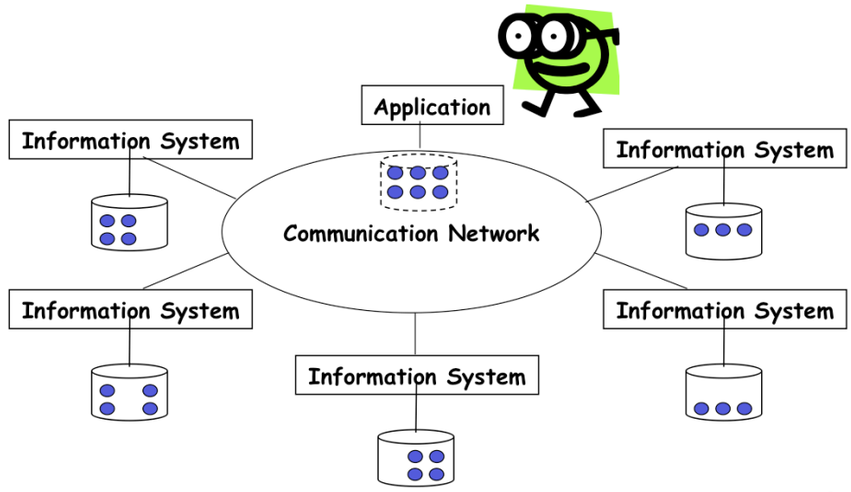
- Central Information System on Computer Network



Except in the very early days, information systems had always been used in computer networks. This does not imply any fundamental problems in addition to those we have discussed up to now as long as the information system is centralized, i.e. running on one physical node under a single authority.

Physical Distribution

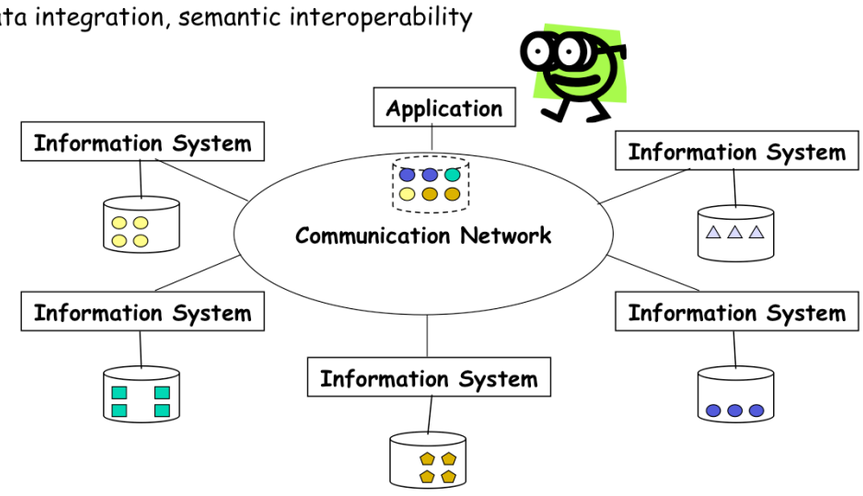
- ***Distribution***
  - support use of distributed physical resources: improve locality of access, scalability, parallelism in the execution
  - Distributed Data Management



There exist however many reasons not to keep all data in a single node on 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 processed, we might take advantage of parallel processing of the data, and we might want to avoid bottlenecks in order to improve scalability of the systems. Thus, we want to distribute the data physically. Nevertheless we don’t want to give up the illusion that we are still accessing a single information system that is running under a single authority.

Distribution of Modeling

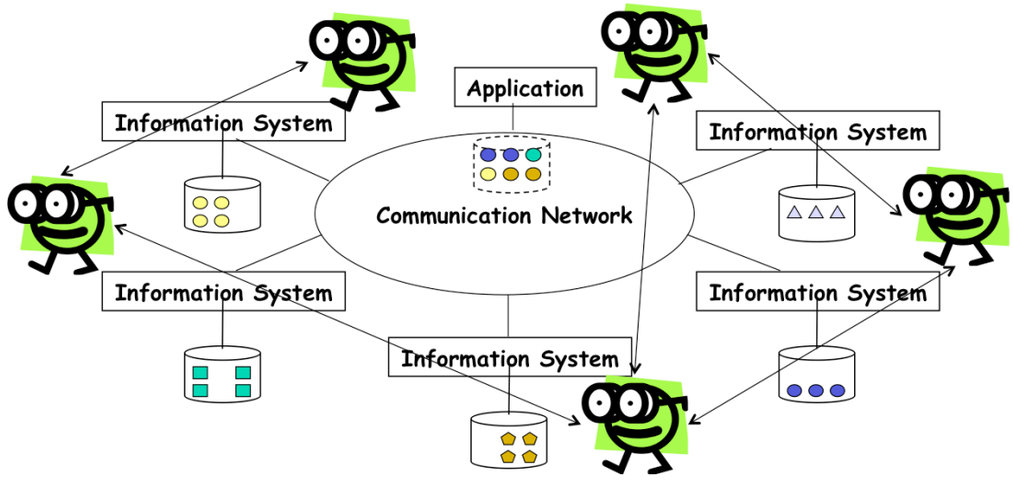
- *Heterogeneity*
  - different data models, schemas, formats
  - enable interoperability of (pre-existing) information systems
  - different interpretations of the same data for different needs
  - Data integration, semantic interoperability



Things can become more complex if we give up the assumption, that a distributed information system should appear like one single information system, even when it is physically distributed over a network. In fact, we might want to make the distribution visible in order to enable multiple authorities to infuse their knowledge into their network, in order to make information systems that where formerly separated working together, and in order to make different interpretations of data available. Thus we are considering distribution of knowledge. New problems occurring in this type of information systems are related to the fact the differences in representation (heterogeneity) needs to be overcome. These are problems that are addressed by methods for data integration and for enabling semantic interoperability.

Distribution of Control

- ***Autonomy***
  - Different participants have to collaborate, coordinate, negotiate, mutually evaluate in order to perform information management tasks
  - Social networks, trust, privacy et.



Finally different, heterogeneous information systems might be under the control of different authorities. We say these systems are autonomous. In this case the different authorities need protocols to coordinate, or to compete in order to jointly perform tasks. In social networks also questions of trustworthiness of information and protection of privacy become an issue. All these problems are related to the pragmatic layer of information systems.

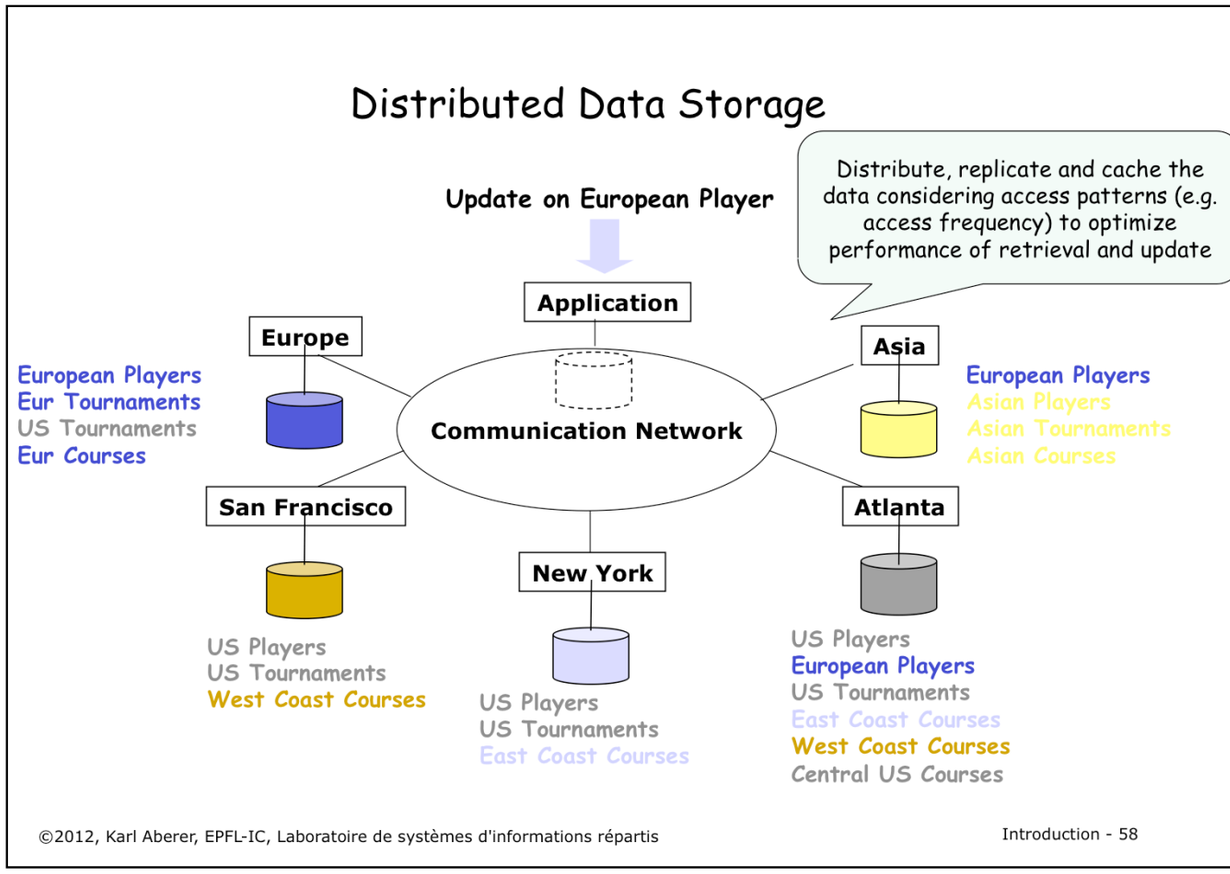
What do you think ?

- What are specific problems to be solved to handle *physical distribution*?

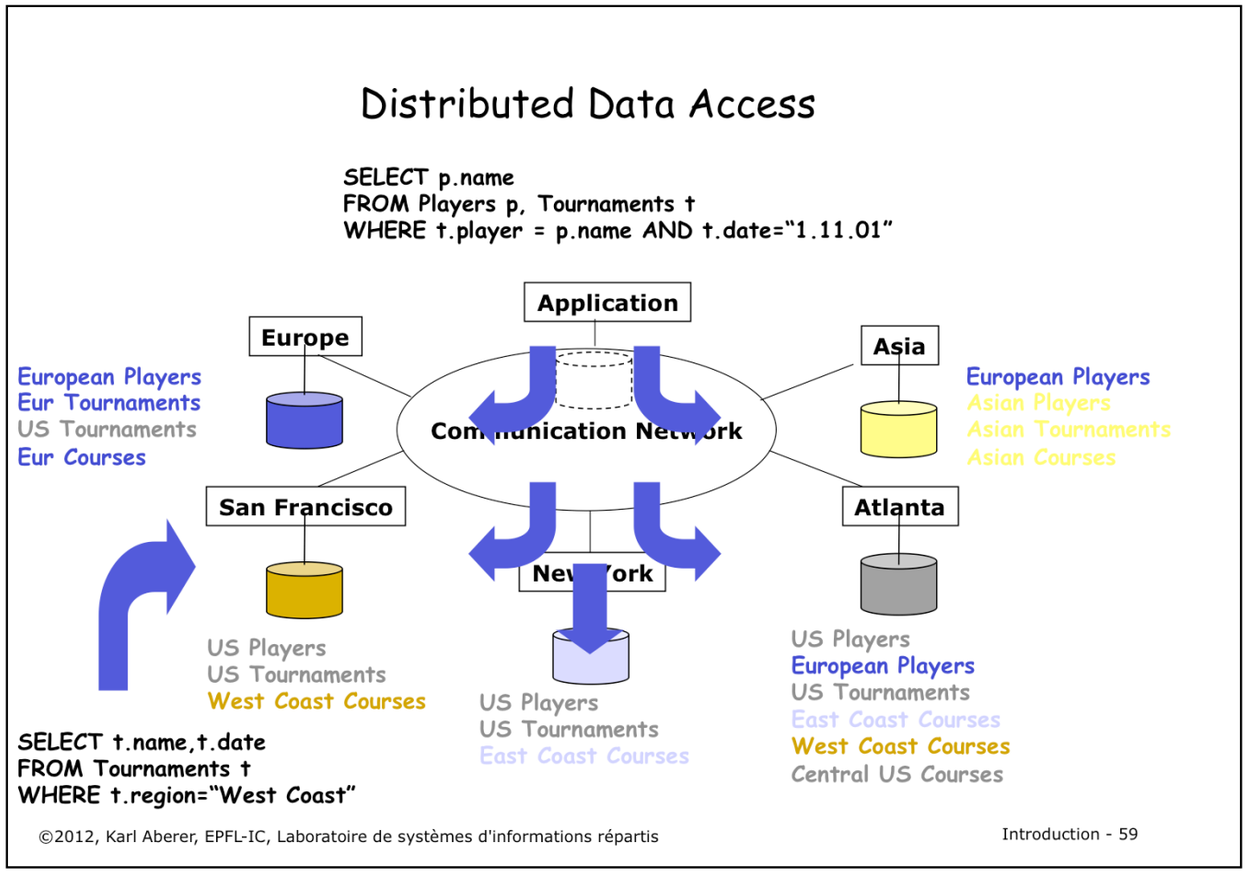
Key Issues in Distributed Data Management

- Physical Distribution
  - requires control of many autonomous sites and efficient usage of existing resources
  - Abstraction: hide as many unnecessary details as possible
- Where to place data in the network?
  - partitioning and distribution of data, replication, caching
  - Considering typical access patterns
  - Consistency of replicated data and of distributed dependent data
  - How to index data in the network?
- How to retrieve data from the network?
  - decomposition of queries and filters
  - information dissemination models
- How to maintain data consistent?
  - atomicity and durability (distributed commit protocols), isolation
  - Distributed transaction management

Physical distribution deals essentially with the distributed counterparts of the standard problems in centralized data management. Physical distribution introduces however one aspect that is essentially different from a centralized setting. Whereas in a centralized setting data is maintained (communicated) only over time, in a distributed setting it is additionally communicated over networks/space(between the nodes). Thus in particular cost of network transmission becomes an important performance factor.



In distributed storage management the key question is of how to distribute the data in the network most effectively.



If data is distributed in a network, in order to retrieve it, the information system has to solve the problem of how to efficiently locate the nodes that contain the desired data and to retrieve the data from there.

New Models of Information Dissemination

- Classical Model of Information Access: Client-Server
- Today the world is changing
  - Mobile communication: wireless data access, broadcasting, embedded systems
  - Information push: RSS, publish-subscribe, real-time data publishing
  - Social networks: gossiping, peer-to-peer dissemination, decentralized systems, Twitter
- Changes how information is stored, indexed, and retrieved
- Information dissemination in distributed databases, social networks

A particular and interesting aspect that cuts across issues of distributed data management, semantic heterogeneity and autonomy is the issue of disseminating information in distributed systems and in particular in networks. Today, for example, much of the information that is accessed is no longer proactively requested by the consumers but pushed within networks to potential consumers. Thus an important aspect in distributed information management systems is to understand the different models of information dissemination.

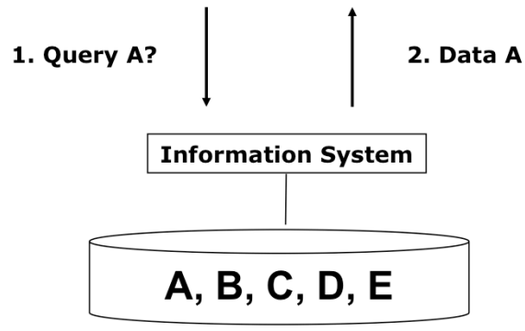
Information Dissemination

- Control
  - push: initiated by source
  - pull: initiated by consumer
- Event
  - periodic
  - conditional (e.g. on change)
  - ad-hoc
- Communication model
  - unicast (point-2-point)
  - broadcast
  - multicast

In the following we will use a classification of information dissemination that follows the work of [Ozsu, Liu, Yang 1998]. It identifies three main dimensions:
control of the data exchange, event leading to data exchange, and communication model used.

Example: Database Server

Control	Event	Communication



Control
1. push: initiated by source
2. pull: initiated by consumer

Event
1. periodic
2. conditional (e.g. on change)
3. ad-hoc

Communication model
1. unicast (point-2-point)
2. broadcast
3. multicast

In the classical Client-Server setting a client sends a request to the information system, and receives as a result a response.

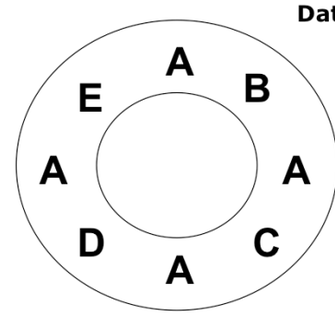
Example: Mobile Broadcast

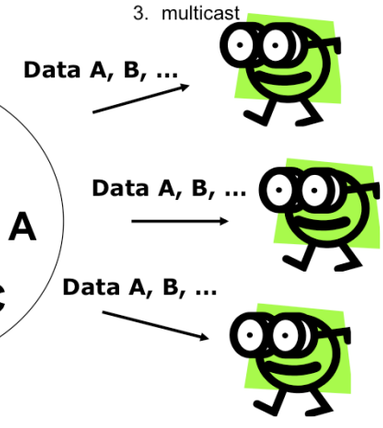
Control	Event	Communication

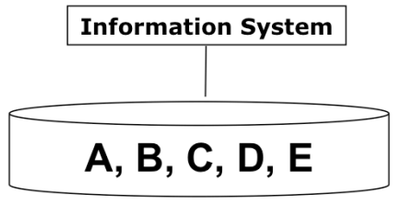
Control
1. push: initiated by source
2. pull: initiated by consumer

Event
1. periodic
2. conditional (e.g. on change)
3. ad-hoc

Communication model
1. unicast (point-2-point)
2. broadcast
3. multicast







In mobile broadcast a server broadcasts periodically all the available information to all clients (within it's range). The clients select from the incoming data stream the data they are interested in.

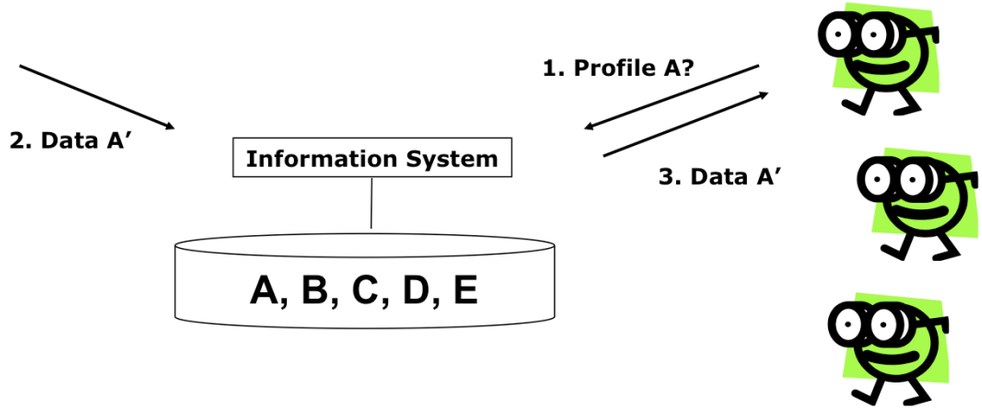
Publish-Subscribe

Control	Event	Communication

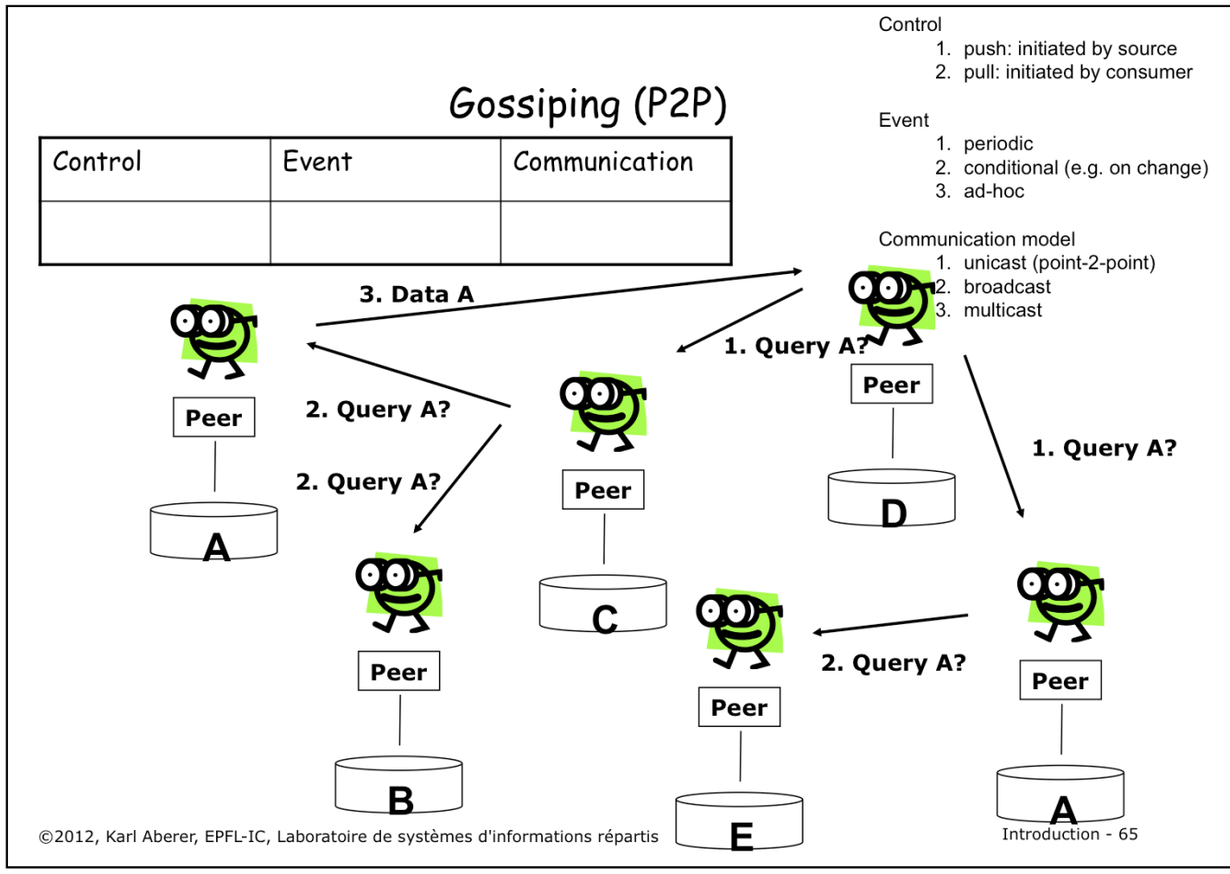
Control
1. push: initiated by source
2. pull: initiated by consumer

Event
1. periodic
2. conditional (e.g. on change)
3. ad-hoc

Communication model
1. unicast (point-2-point)
2. broadcast
3. multicast



In a publish-subscribe system the clients first deposit what information they are interested in (profile), and whenever new data arrives that is matching the profile the server sends the data to the client.



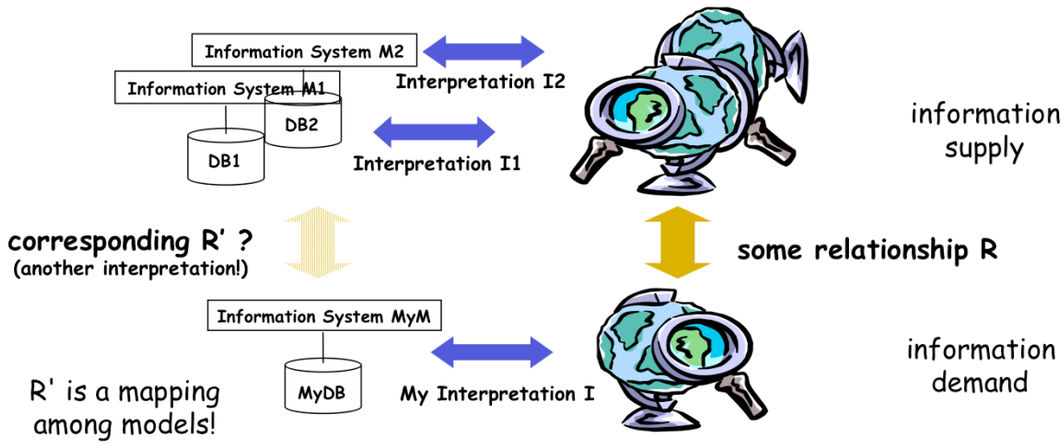
In P2P information systems no server exists. Every client propagates search request to local neighborhood, which further spreads the rumor till eventually some node can provide the requested data.

What do you think ?

- What are specific problems to be solved to handle *distribution of modeling*?

Key Tasks in Distributed Information Management

- Having access to (many) different information systems is great!
  - More data! … More models!? … More information?



This figure illustrates a key problem in distributed information systems: there exist many information systems supplying data, but having their own view on the world, which does not necessarily match the purpose of a specific consumer. There exists some relationship R between the real world aspects that potential data providers cover and the real world aspects we are interested in. Given this real world relationship (which typically cannot be formalized) the problem is now to provide a corresponding relationship R' between the data we are provided with and data that we are able to use. Since R is normally not well specified, finding R' is in general a difficult problem. From the viewpoint of the data providers introducing R' introduces a new interpretation of their data, and thus creates a new "real world aspect" that they can relate their data to. Thus by creating a proper interpretation of provided data we implicitly also connect the data providers to "new worlds".

Problem in Distributed Information Management

- *Data overload*
  - more databases, more connectivity, disintermediation (more direct access to data sources)
  - more information ?
- *Information starvation*
  - problem: *data supply* does not match *data demand*
  - models used by data provider is different from models used by data consumer
- *Proper Interpretation* of other information resources (usually) requires human intervention
  - Human attention is the scarcest resource !
- **Semantic heterogeneity**
  - The same information can be modelled differently

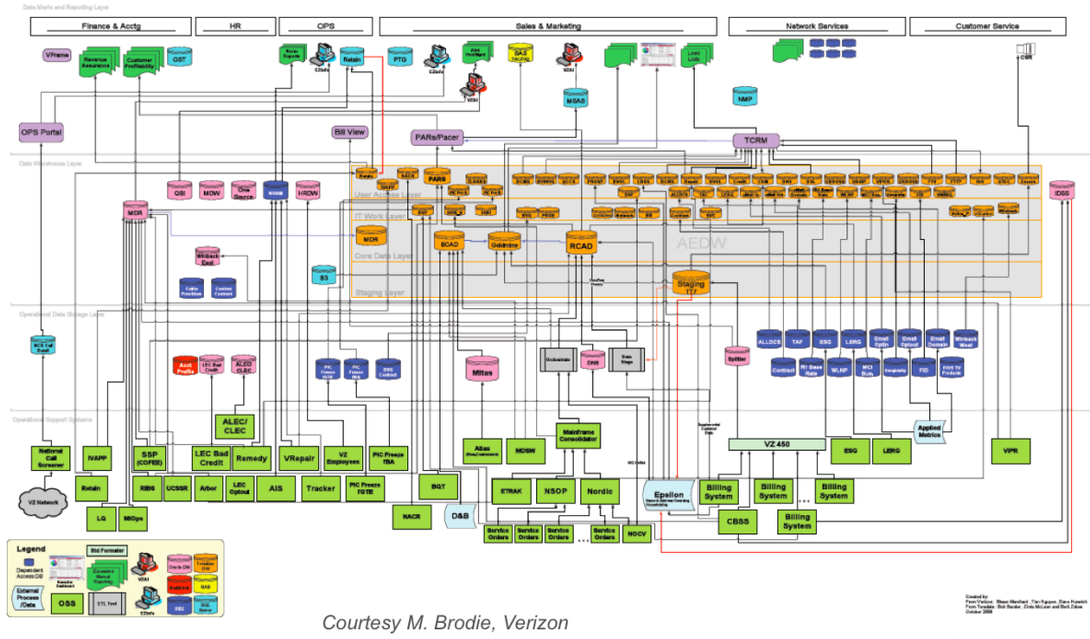
Distribution of modeling is related to the problem of having multiple interpretations. There exists an abundance of data. The problem is whether the user (or the applications) can interpret it properly. I.e., as long as no proper interpretation exists, the data is of no use. This is what is called information starvation. In particular, as human attention is the scarcest resource that exists, and human attention is often required to transform the data into a form that a user is are able to interpret, solving information starvation is a task with a natural resource bottleneck, human time.

Examples

Supply	*Model*	*Demand*	*Mapping (R')*	*Model*	*Purpose*
Bank account	Table	SWIFT document	Converter	Document	Data exchange
Web document	Text	Relevance	Text similarity	Ranked list	Read top document
Shopping transactions	Table	Customer class	Classification (mining)	Tree	Marketing campaign
Astronomical observation	Image	Asteroid	Image analysis	Table	Warning
… find more …					

These examples illustrate of how from existing data (supply denotes the model of the provider), new data (demand denotes the required model of the demander) is created for new purposes. Note that the necessary operations can be everything from simple manipulations of data structures to complex algorithmic procedures such as text or image analysis.

Interoperability - The problem today

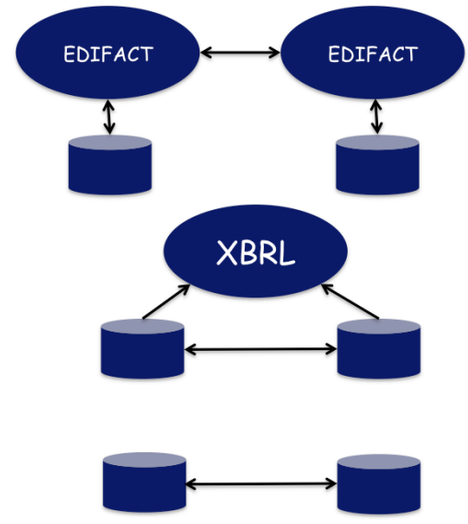


Courtesy M. Brodie, Verizon

This figure gives an impression of the complexity of the structure of information system n a modern enterprise. Many different independently developed systems co-exist, and making them work togeter (to interoperate) is a huge challenge. It is estimate that a large fraction of IT expenses today are due to this problem.

Mapping: Three Approaches

- Standardization
  - Mapping through standards
- Ontologies
  - Mediated mapping
- Mapping
  - Direct mapping

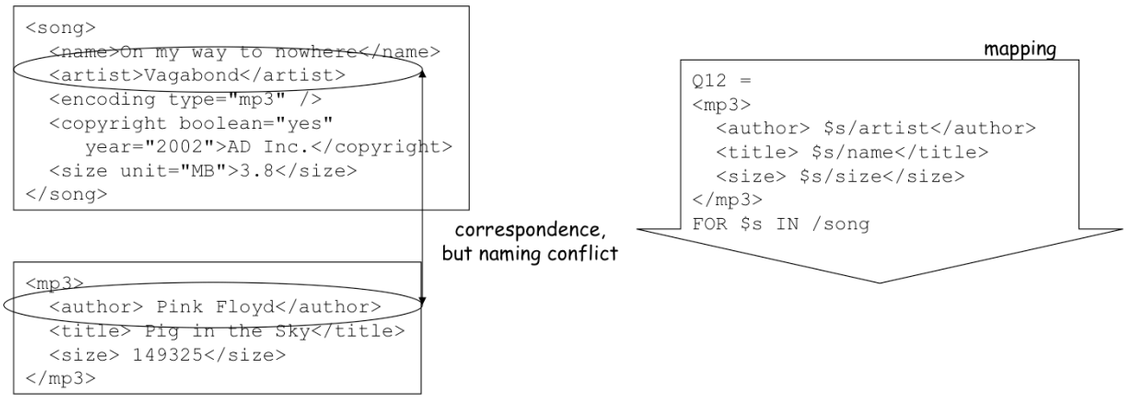


Conceptually there exists three main possibilities of how to overcome semantic heterogeneity. A first is to map everything to a common global model, this approach is taken with standardization. A second, is to relate the model of an information system to a common mode, frequently called ontology, and use this to construct a mappping among the information systems. A third consists of trying to construct directly a mapping among two information systems.

Direct Schema Mapping

- Assume all data represented in canonical data model (e.g. XML)
  - detect correspondences (schema matching)
  - resolve conflicts
  - integrate schemas (schema mapping)

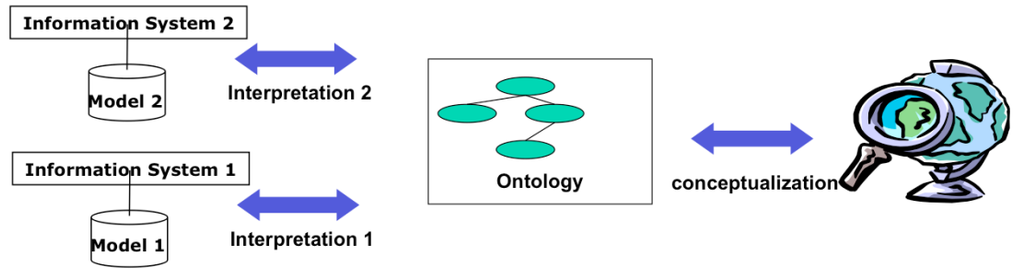
  => a methodology!
- Mappings are frequently expressed as queries (e.g. XML Query)



For constructing a mapping among different models, be it for direct mappings among information systems, or for mapping from an information system to a standard or ontology, a basic methodology has been identified. The standard *methodology* for mapping is to first identify which constituents of a schema correspond to each other (this is also called matching), then to identify whether there exist differences in the way the same information is represented and find ways to overcome them, and then to construct from that a mapping for the data. In databases the mapping is typically expressed using a query language.

Ontologies

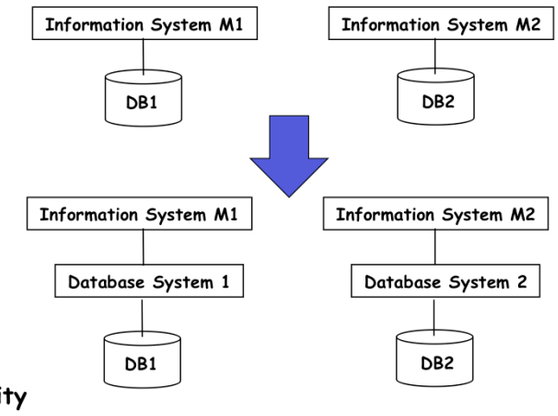
- Ontologies are an explicit specification of a conceptualization of the real world (Gruber, 1993)
- Ideally
  - different information systems agree on the same ontology
  - relate their model/schema/data elements to the ontology
  - mapping can be constructed via the ontology
- Issues
  - ontology languages (e.g. OWL), mappings and engineering
- Problem: requires agreement on the conceptualization of the real world !



One way to deal with semantic heterogeneity is to agree on a common representation on the real world, which are called ontologies. Information systems can then provide a explicit and formally specified “interpretation mapping” with respect to the common ontology, which serves as a “proxy” for the real world.

Different Data Models?

- Is that all?



Data Models might be different!

- **Syntactic heterogeneity**
  - Data represented in different formats, data models
  - « Straightforward » but costly problem

Actually heterogeneity can also occur due to the fact that different information systems use different data models to represent their models in a computer sysetm. In that case we are talking of syntactic heterogeneity.

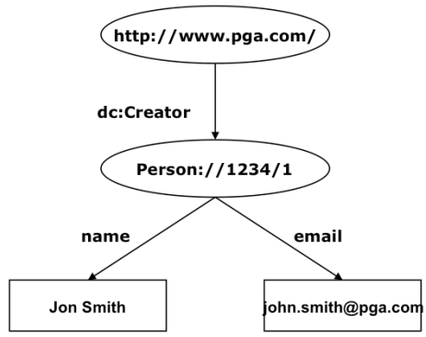
Mappings among Data Models

- Frequently data from one data model needs to be mapped to a different data model
  - Reusing existing implementations of database systems to support other typed of data models
  - Integrating information from different information sources
- Examples
  - Storing Java objects in relational databases (EJB)
  - Storing XML in relational databases
  - Exchanging relational data through XML
  - Representing RDF in XML
  - Etc.

For overcoming syntactic heterogeneity mappings among different data models are needed. This problem has been studied in many contexts, e.g. for mapping data from programming languages models to models of data management systems, or for mapping data from data models of data management systems into document models for data exchange.

Semi-Structured Data Models

- Semi-Structured Data (e.g. XML, RDF)
  - Self-describing: schema information directly encoded into data
  - Generic: handle all kinds of structures, in particular graphs
- Result
  - Everything can be converted to this data model
  - Data is self-contained when exchanged




```
<rdf:RDF>
  <rdf:Description about="Person://1234/1">
     <s:Creator>John Smith</s:Creator>
     <s:createdWith
        rdf:resource="http://www.w3c.org/amaya"/>
  </rdf:Description>
</rdf:RDF>
```


In the Web a new class of data models evolved, so –called semi-structure data models. Different to classical structured models, such as the relational data models, they are designed to deal in particular with issues that are related distribution and communication:

- semistructured data models support serializability (i.e. the possibility to represent the data as a string that can be sent over a communication channel) which facilitates data exchange over networks

- they relax the need to use a static schema or do not require a schema at all, which is helpful to process data from different information systems that are using different models or for evolving or discovering structural regularities in data for deriving schemas a posteriori

- they include explicit mechanisms to represent relationships, thus facilitating the representation of graph and network structures.

Questions

- Mapping a schemas of two relational database to a common XML-based electronic data exchange data format allows to overcome
  1. Semantic heterogeneity
  2. Syntactic heterogeneity
  3. Both

- An ontology
  1. Is a standard data model to enable data exchange among different information system
  2. Is a standard conceptual model to represent a certain aspect of the world in an agreed upon form
  3. Is a self-describing schema language supporting conceptualization of real-world aspects

What do you think ?

- What are specific problems to be solved to handle *autonomy*?

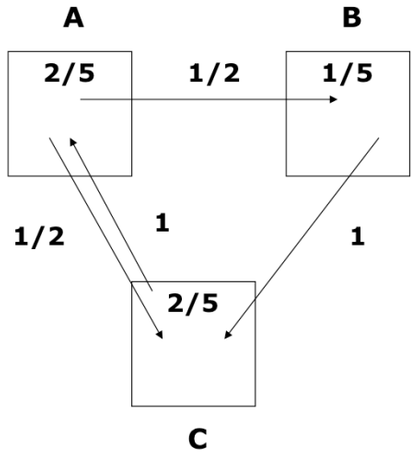
Autonomy-related Issues

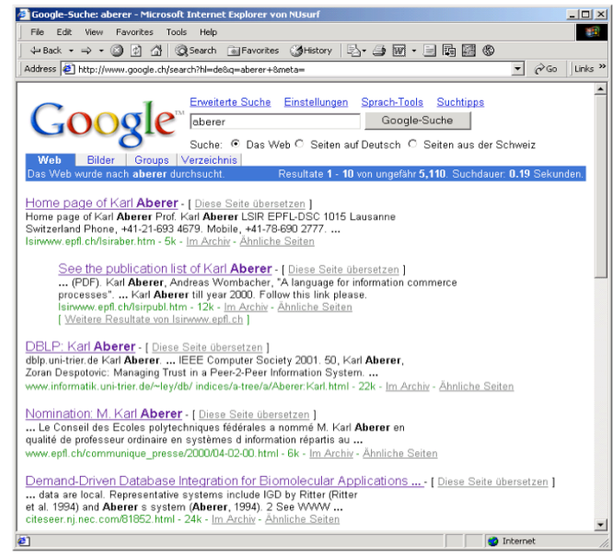
- Problem 1: Quality of Information and Trust
  - Even if information is well understood, it is not always clear whether it is trustworthy, correct, important etc.
  - Solutions
    - Check consistency of information
    - Collect feedback on the information (reputation)
- Problem 2: Privacy
  - Access to information needs to be restricted to selected users in order to avoid undesirable consequences (economic, legal, personal)
  - Solutions
    - Access control mechanisms, encryption
    - Data anonymization, obfuscation etc.
- Problem 3: 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

In order to deal with autonomy mechanisms are required that enable different participants (agents) to interact in meaningful ways, while trying to achieve their objectives. This implies a wide range of problems of which we mention here three which have been receiving recently significant attention.

Ranking of Information

- Example: Web search, which pages are important?
  - Web crawling
  - Link analysis: Page rank

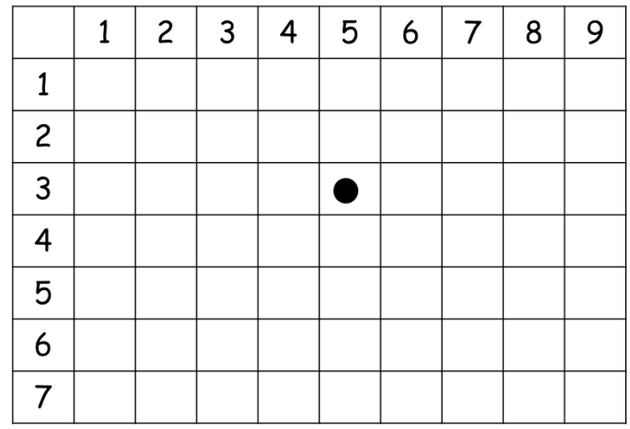




One example, related to the issue of autonomy is how in a large community (such as the Web) distributed knowledge can be exploited in order to asses the quality of information. This is what Google for example does with its link-based ranking mechanism, that tries to establish a “consensus” among many “opinions” (expressed as Web links) resulting in a quality rating.

Privacy

- Example: location privacy – obfuscation methods
  - Current location: (3,5)
  - Perturbation: (3,7)
  - Adding dummy regions: (3,5), (1,4), (6,3)
  - Reducing precision: (2,5), (3,4), (3,5), (3,6), (4,5)

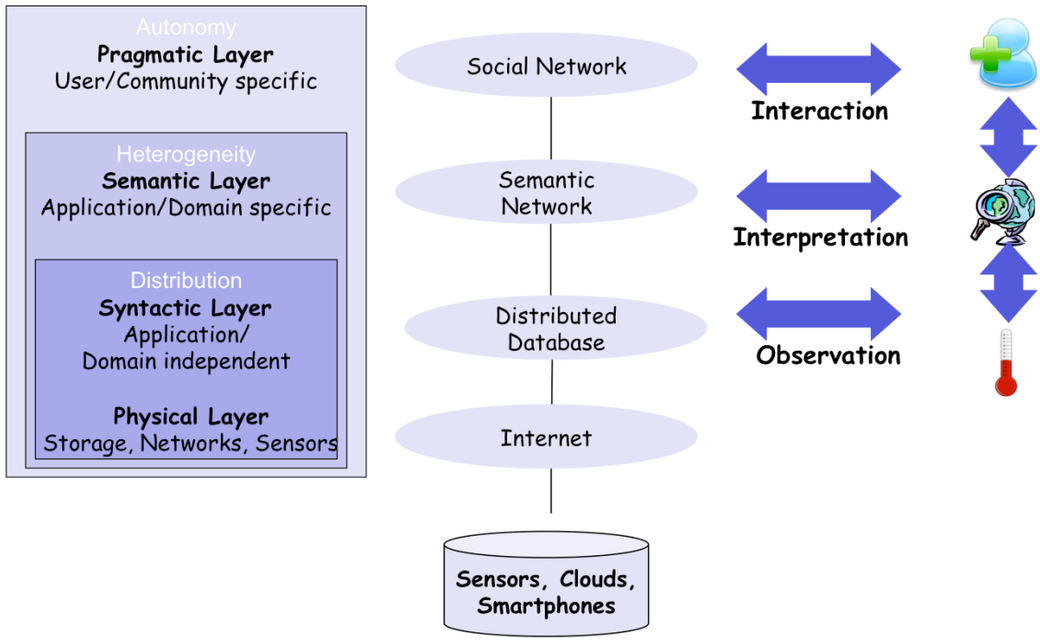


Another example of an autonomy-related issue is privacy. On the one hand participants like to share their own information (such as their location) with other participants in order to obtain useful services, on the other hand they do not like share such information as this might lead to undesired uses by others. In order to deal with such problems, methods such as obfuscation are use to provide sufficient information to obtain service, but not so much that it may be harmful. Like in the previous example, the problem is linked to the “value” or “utility” of information, with respect to the purpose of its usage.

Questions

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

Refined View of a Distributed Information System



If we generalize our overall architectural view of information systems to the distributed case, we see that in particular former system components that could be identified with pieces of software running on servers, are replaced by different types of networks. The analysis and maintenance of such networks has indeed become one of the major foci of attention in distributed information systems, and will also play a role later in this lecture.