
UNIVERSIDAD
SAN SEBASTIAN

# Data Management and Databases

## Chapter 1: A General Overview

**Leopoldo Bertossi**

Universidad San Sebastián

Facultad de Ingeniería y Ciencias

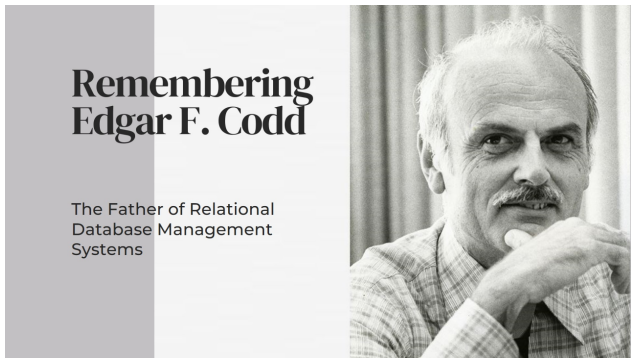
# Relevance of Database Management Systems

- The idea and later, implementation of Relational Databases (RDBs) have been around since 1970

  When they were proposed by Edgar F. Codd (1923, 2003)

  He obtained the Turing Award in 1981 for his contributions to data management

  The most important award for research in Computer Science




- Before, creating, updating, using, querying a DB was painful

  Need to be aware of- and explicitly use the internal aspects:

  Data storage, navigation and access methods, to: locate, connect and retrieve data

  In other words: the internal data structures and algorithms

  Yet in other words: the physical aspects of data

- Main innovation brought about by Codd:

  Separation of the physical aspects of data from the logical view of data

- That is, their logical organization and presentation

  A user could see and interact with the DB without necessarily being aware of (or knowledgeable about) the physical aspects

- This is what you "see", and interact with ...

| Sales | Customer | Price | Article | Store |
|---|---|---|---|---|
| | peter | $23 | cd | cdWarehouse |
| | mary | $30 | book | chapters |

| ParHood | Parent | Child |
|---|---|---|
| | peter | mary |
| | john | stu |

  Most users do not need to peek under the hood ...

- Codd relied on the Predicate Logic

  A symbolic logic that had been investigated and used in Math (since the late 19th Century), and CS (since 1935 approx)

  A RDB is a set-theoretical structure as used in Predicate Logic

- Initially most did not believe the idea would be usable in practice (performance issues)
- Later in the same decade some started to implement Relational Database Management Systems (RDBMSs)

  Main efforts: performance, internal algorithms, optimizations, transactions, ...
- New Turing Awardees for early work on the subject:

  Jim Gray (1944, 2012) TA: 1998
  Michael Stonebreaker (1943) TA: 2014
- RDBs is a success story!
- Not many other technological achievements in Computing with this level of impact on everyday life

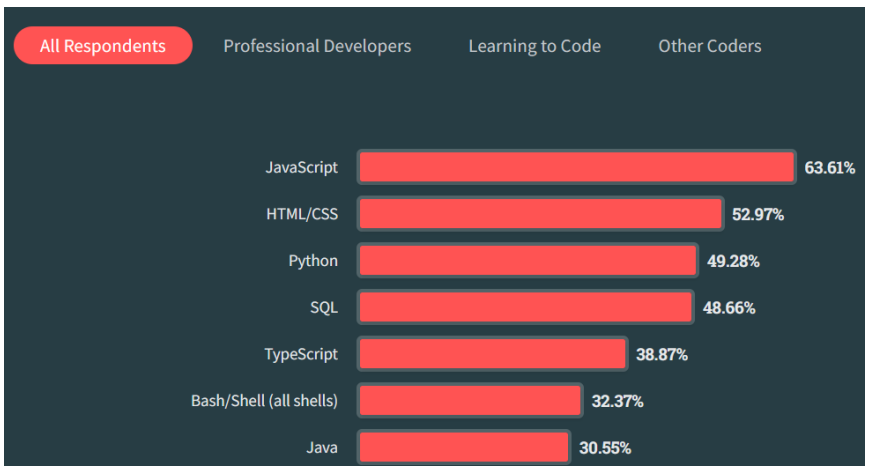
  Any others?

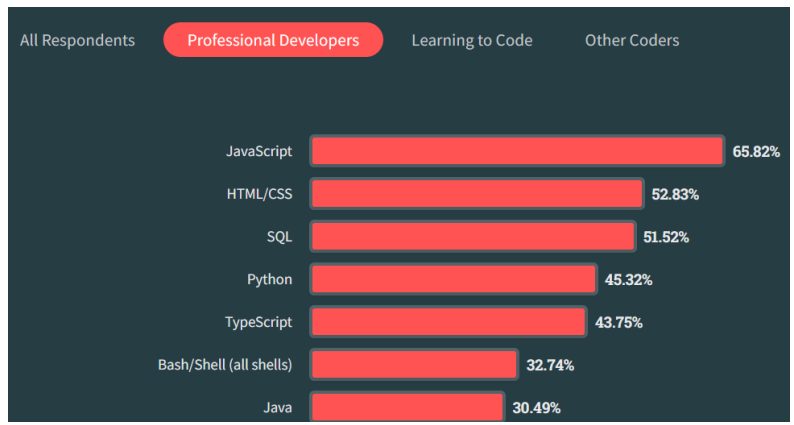
- RDBs are used for Data Storage, Processing, Analytics, Business Intelligence, Business Support

  Machine Learning (much about data), AI in general

  Enterprise Resource Planning, Inventory Management, Logistics, etc., Transaction Processing (banks, ...), Web Services, ...
- The relevance of SQL, the standard query (and interaction) language for (with) RDBs, cannot be emphasized enough

  A must for almost every computer professional (and others)




Survey (and much more): https://survey.stackoverflow.co/2023/#overview

# What Is a Database?

- This course is about SQL-based Relational DBs

  And some extensions, diversions, ...

- But: What is a (Relational) Database?

- In principle, different things for different people/communities

  - A repository of data
  - A collection of tables
  - A collection of relations  And then, based on set-theory

    Actually, finitely many finite relations
  - More interestingly: A set-theoretic structure, with:

    - A data domain (possibly infinite)

    - Finitely many finite relations; each with a fixed, finite "arity" (number of arguments/"columns")
  - Expanding previous item: An interpretation structure for a symbolic language of Predicate Logic

- A Knowledge Base (KB)

  And then, a simple form of knowledge representation

  A finite set of "atomic symbolic sentences (statements)":

  $Sales(peter, \$23, cd, cdWarehouse), \ldots$

  $ParHood(peter, mary), \ldots$

- Elaborating on previous item:

  A RDB as a KB can be extended into more sophisticated forms of knowledge representation

  As those found in AI

- Particularly interesting: A model of an external (data-related) reality (more on this next)

- A RDB can be any of these

  And different views can be reconciled ...

- Particularly relevant: One can (computationally) interact with the RDB

# Data Models, Models in General

- To understand/manipulate/transform/update/extract/process data, we need to create the right models of data
- What is a model?
- An abstraction, a simplified description or representation of an external reality, phenomenon

  A physical phenomenon? A company? etc.
- The model explicitly captures some salient, relevant aspects of the external reality; others are left implicit
- We may be interested in extracting from the model both explicit and implicit information
- There is nothing like a universal modeling language
- If we want to represent mathematical relationships between numerical variables, we may use mathematical equations (they come in different forms)

- If we want to model data, we may (and usually) create mathematical models of data
- In data management those models are directly based on set theory and symbolic logic
- Different ways of modeling data depending on the application and the (mathematical) elements we use to represent data

Example of Physical Model:

- In Physics, a model may given as motion equation, for the position as a function of time:

$$p(t) = 0.5 \times t \; + \; p(0)$$

  Plus the initial observation: $p(0) = 2$

- This is the model

  We do not explicitly include all possible pairs (position, time)

  That is implicit

  We can obtain from the model: $p(6) = 5$

Example from Logic: Aristotelian argument:

*"If Socrates is a man, then Socrates is mortal.*
*Socrates is a man. Then, Socrates is mortal."*

- It is expected to be a *valid* argument, i.e. always true
- Actually more due to its logical structure than to the particular properties, e.g. being mortal, and individuals involved, e.g. Socrates
- All this could be expressed in propositional logic, as follows:
  1. Denote the most basic and atomic propositions involved, i.e. not decomposable in sub-propositions, by means of propositional variables:

  $P$: "Socrates is a man" $Q$: "Socrates is mortal"

  2. Use symbolic logical connectives to combine the propositional variables, to symbolically express the first statement above (among many other possible statements): $(P \rightarrow Q)$

- The model in this case becomes the following formula of propositional logic:

$$((P \underbrace{\longrightarrow}_{\text{implies}} Q) \underbrace{\wedge}_{\text{and}} P)$$

- Equivalently, as a knowledge base expressed as a set of formulas: $KB = \{ (P \longrightarrow Q),\ P \}$
- The implicit knowledge should be $Q$, in the sense that

$$((P \longrightarrow Q) \wedge P) \longrightarrow Q) \qquad (*)$$

  is an always true formula, i.e. a tautology
- We say that $Q$ is a logical consequence of $KB$
- Formula (*) is always true for every truth values (0 or 1) assigned to $P$ and $Q$

  This can be easily checked with a truth table (Do it!)

- Alternatively, expressing it in an logically equivalent formula

  By means of logico-algebraic operations:

$$((P \to Q) \ \wedge P) \ \to \ Q) \ \equiv$$

$$\underbrace{\neg}_{\text{not}}((P \to Q) \wedge P) \ \underbrace{\vee}_{\text{or}} \ Q \equiv (\neg(P \to Q)) \vee \neg P) \vee Q$$

$$\equiv (\neg(\neg P \vee Q)) \vee \neg P) \vee Q \equiv \neg(\neg P \vee Q) \vee (\neg P \vee Q)$$

- The last formula is of the form $\neg p \vee p$, which is always true

  The most common form of tautology!

- Another Aristotelian argument:

  *"All men are mortal. Socrates is a man. Then, Socrates is mortal."*

- It should be again a valid argument

  Can it be expressed as such in propositional logic?

  Let's try as above:

  1. Propositional variables: $P$: "All men are mortal"
     $Q$: "Socrates is a man" $R$: "Socrates is mortal"
  2. The argument in propositional logic?

     $$(P \wedge Q) \longrightarrow R$$

- Not a tautology: It is logically equivalent (denoted $\equiv$) to

  $$\neg(P \wedge Q) \vee R \equiv \neg P \vee \neg Q \vee R$$

  If $R$ is false and $P, Q$ are true, the formula becomes false

- We need a more expressive logic

  With different levels of granularity: individual and conceptual levels! $\rightsquigarrow$ Predicate Logic (the logic at the basis of relational DBs)

- The argument involves general, non-instantiated concepts (properties, predicates, attributes), e.g. "being mortal"

  1. Introduce symbolic predicates: $Man(\underbrace{\cdot}_{\text{for 1 argument}})$, $Mor(\cdot)$

     A constant to denote (name) Socrates: $s$

     Variables: $x, y, \ldots$ (implicitly ranging over the underlying domain of discourse)

     A universal quantifier: $\forall$ (intended meaning "for all")

  2. Now the argument as a formula of predicate logic:

     $$(\forall x(Man(x) \longrightarrow Mor(x)) \wedge Man(s)) \longrightarrow Mor(s))$$

  3. This formula is "universally valid": It is true under all possible interpretation (intuitively for now)

- Predicate Logic is more expressive than propositional logic

  The one we use in RDBs and many areas of data management!

- When Edgar F. Codd proposed relational DBs in 1970 he explicitly based his model on Predicate Logic

  His seminal paper: "A Relational Model of Data for Large Shared Data Banks"

  He won the Turing Award in 1981

- Predicate Logic has multiple applications in Computer Science

  Prominently in AI!

# Data Models, ER, Relational Model

A model of data -for a particular domain- has to capture the characteristics of data:

- Kinds of data items
- Associations/relationships between data items
- Associations between classes of data items
- Dependencies between data items
- Restrictions of the kinds of data
- Natural organization of data items (if any)
- Etc. Etc.

The domain or external reality (whose data is) being modeled could be about a university, sales of articles, a medical domain, etc.

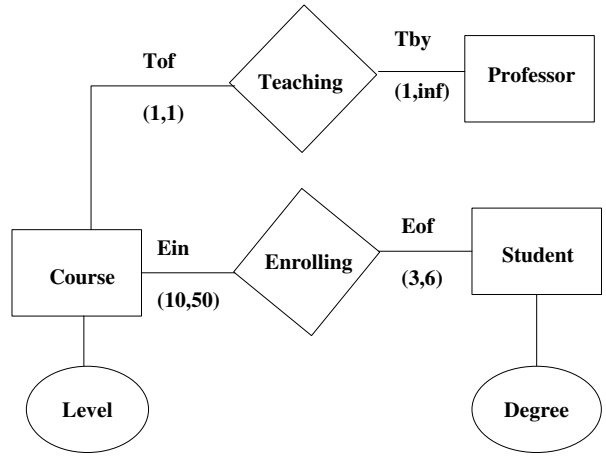
Example:

- "*Peter spends $23 on a CD at CDWarehouse*"

  "*Mary spends $30 on a book at Chapters*"
- Here "Peter" is a data item, the same applies to "$23", "$30", "book", "Chapters", ...
- Not only that: "Peter" is of the kind, say "Customer", "CD" of the kind "Article", ...
- We have categories (or concepts, classes, entities) of data items
- The concept "Customer" is related to the concept "Article"

  The data item "Peter" is related to data item "CD", but not to "book", etc.
- How can we capture all this?

## An Entity/Relationship (ER) Model:




- A graphical "language" that uses diagrams to model data
- Entities: represented by boxes; they correspond to classes of data items of a same kind
- Relationships: by diamonds; they relate (data items from) different entities
- Attributes: by ovals hanging from entities; representing properties (of elements) of an entity

  May also hang from relationships, e.g. "Character" hanging from "Enrolling" (a student may take the course as elective, mandatory, etc.)
- Cardinality constraints: label links between relationships and entities

- Do not take edge (link) names too seriously for now

  They become relevant when ER is translated into a symbolic ontology

- Intuitively, for entity *Course*, and through the relationship *Teaching*, the entity *Professor* becomes an "attribute" of *Course*

  Hence the "TBy" (taught by). Etc.

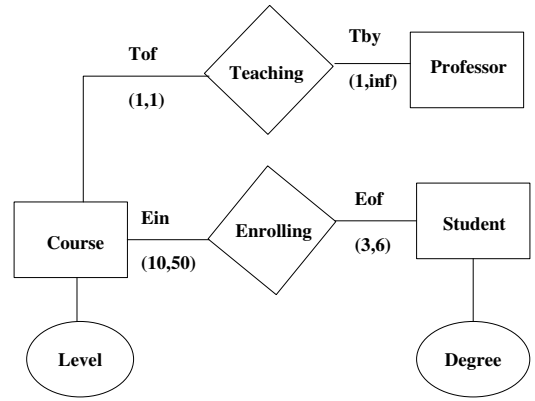
- An ontology is a description of a domain in terms of classes and relationships between them

  There are many ontological languages (for the descriptions)

  ER models can be seen as very basic, graphical ontological languages

  Ontologies are important in Data Management and AI

- Entities: *Course, Professor, Student*
- Relationships: *Teaching, Enrolling*




- *Course* represents courses

  Where students from *Students* are enrolled

  And are taught by professors from *Professors*

- Attributes: *Degree, Level*

  We could also have an attribute *Term* or *Character* hanging from the *Enrolling* relationship

  An attribute of the relationship between students and courses

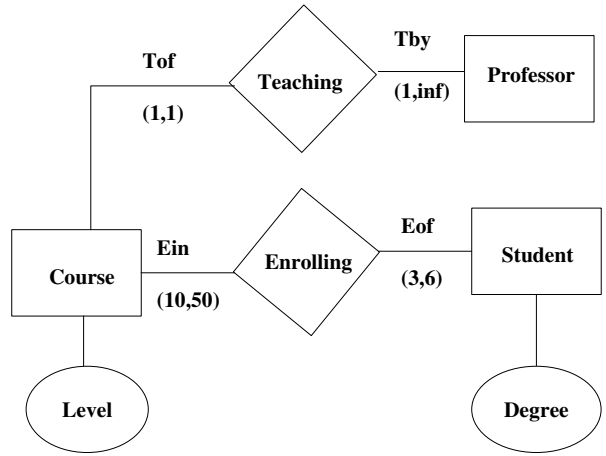
- *Course, Professor, Student* are also called "concepts" and the model above is also called a "conceptual model of data"
- An ER model is close to the outside (data) reality that is being modeled, close to how a user or modeler sees the world
- An ER model does not fully represent how data are represented, organized, structured, handled, ...
- ER is a high-level model that does not show much about the details of data
- An ER model, being a model, is expected to stay close to the outside reality that is being modeled
- That external reality gives a meaning (semantics) to the model

  And the ER model models the external reality

  The more elements (that are semantically correct w.r.t. the external reality) we add to the model, the closer we are to the external reality

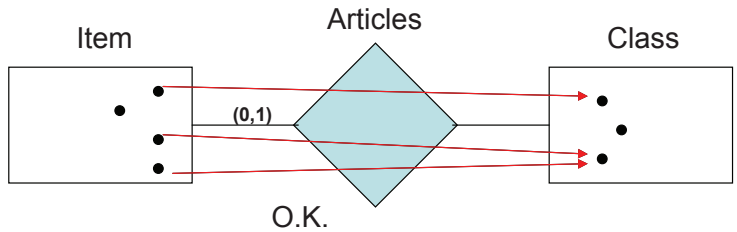
- Cardinality constraints are used to capture more meaning or semantics

  For a better representation of the external reality by means of the ER model




- They are semantic constraints
- In the example, they capture:
  - The (external) limit on the number of students that may register in a course (between 10 and 50)
  - The limit on the number of courses that each student can take (between 3 and 6)
  - That each course is taught by exactly one professor, who has to teach at least one course

- Cardinality constraints can be understood as restrictions on the mappings between entities through the relationship




  - In particular, label (0,1) imposes that "at most one entity in *Class* is associated to each entity in *Item*"

    A very common constraint ...
  - (1,1) indicates "exactly one ..."
  - (1,N) indicates "at least one ..." (with N standing for a generic, arbitrary numerical upper bound)
- ER models can be extended with notation for indicating sub- or super-entities (subclasses, subconcepts, etc.)

  E.g. *GradCourse* can be defined as a subentity of *Course*
- With inheritance of relationships and attributes ...

## Relational Model of Data: (see example on page 17)

- What about this tabular representation?

| Sales | Customer | Price | Article | Store |
|---|---|---|---|---|
| | peter | $23 | cd | cdWarehouse |
| | mary | $30 | book | chapters |

| ParHood | Parent | Child |
|---|---|---|
| | peter | mary |
| | john | stu |

- Notice that this data set is (seems to be) quite structured
- Structured in the sense that each "tuple" or row in a table (or element if seen as a relation) is expected to have four values for each of the attributes (and not more)

  If some values are missing, we usually insert "null values", but they are generally not welcome (more later ...)

  If an external reality is expected to give rise to less structured data, we may consider more flexible alternatives to the relational model (more later ...)

  This could be the case, for example, if we have very sparse data about the stores where articles were bought

- Back to our model on preceding page ...
- It is Indeed a simplified representation
- It captures the relationships between data items through a same row in the table
- And the fact that data items are of different kinds
- Is this a mathematical model?
- It can be the tabular presentation of a mathematical model
- Actually, everything can be formulated using set-theory and predicate logic