UNIVERSIDAD
SAN SEBASTIAN

Data Management and Databases

Chapter 2: Database Design

Leopoldo Bertossi

Universidad San Sebastián

Facultad de Ingeniería y Ciencias

General and Historical Observations

We start this chapter with some relevant, general considerations about Relational Databases (RDBs) (for lack of a better place)

- DBs can be seen as sets of finite relations
- Each relation containing tuples (atoms, raws, ...)
- Tuples are finite sequences of mutually related data items, e.g. $\langle peter, \$23, cd, CDWarehouse \rangle$
- DBs as stored in DBMSs, accessed and manipulated by them
- Three levels of a DB:
  - Internal: Physical structure of DB; physical model; storage; data structures; and access mechanisms
  - Conceptual: Logical structure of DB; hiding storage details; description of classes of data; interrelationships, etc.
  - External: DB from point of view of final users; external applications

Some critical features of RDBMSs:

- Storage: Persistent, Massive, Efficient, Reliable; and Secure
- Access: Fast, Accurate, Sophisticated (complex queries), Simple (query languages), Concurrent
- Update: Reliable, Concurrent, Efficient, Consistent, Simple (update languages)
- Applications: Interconnectivity with other software systems and programming environments

From DBMSs we expect:

- Logical separation between data and programs

  Before the inception of RDBs, the structure of data was embedded in the data manipulation programs

  Any change in the structure of data implies change of the programs, and viceversa

- Security and Integrity

  Access control (what data is available for what users)

  Consistency of information

  Recovery from failures

- Concurrent access to information

  Different transactions can simultaneously access and modify the DB

- Possibility of building applications on top

  Writing and running programs that interact with the DB

Example: Banking System

- Data: Information about accounts, branches, clients, balances, loans, interest rates, transactions, etc.
- Persistent: Data must persist beyond programs, transactions and application programs
- Multi-User: Many people and systems access and change data, possibly the same and simultaneously
- It should be possible to execute these two concurrent transactions

  If initial balance = 400, how does the DB evolve internally?

  How if initially the balance is only 120?

```
Jill@ATM1: withdraws $100 from account #55
  Get balance from the DB;
  If balance > 100 then
      balance:= balance - 100;
      dispense cash;
      write new balance in DB;

Ken@ATM2: withdraws $50 from account #55
  Get balance from DB;
  If balance > 50 then
      balance:= balance - 50;
      dispense cash;
      write new balance in DB;
```

- A whole theory of DB transactions was developed, and eventually implemented
- Main issues related to supporting (synchronizing) concurrent transactions from multiple users (or applications)

  Transactions possibly affecting the same data in secondary memory
- Also similar to problems of concurrent access to file systems, but control at different levels of granularity is required
  - Reliable and secure: The DBMS must be reliable wrt system failures and malicious users
  - Convenient: Simple instructions to withdraw money, get balance, transfer money, etc.
  - Efficient: Efficient search for data, its processing, update, and response to instructions
  - Consistency: At "commit points" of transactions, data integrity (consistency) must be guaranteed (and usually for multiple users)

Some Historical Landmarks:

- 1961: First DBMS: *Integrated Data Store* of GE
- 1962: IBM and American Airlines develop SABRE
- 1966-: IBM develops *Information Management System* (IMS) (hierarchical model)
- 1970: Edgar Codd (IBM) proposes the relational model of data, and specifies how a relational DBMS could be built accordingly
- 1975: First international conferences ACM SIGMOD and VLDB
- 1976: Peter Chen introduces the ER model
- 70s: Development of first RDBMS: *System R* (IBM), *INGRES* (U.C. Berkeley), *System 2000* (U.Texas), *ADABAS* (T. U. Darmstadt)
- 70s: Query languages are proposed and implemented: SQUARE, SEQUEL, **SQL**, QBE, QUEL
- 80s: DBMSs for PCs (DBASE, Paradox, etc.).

- 1981: Edgar F. Codd receives the ACM Turing Award

  For his fundamental and continuing contributions to the theory and practice of database management systems. He originated the relational approach to database management in a series of research papers published commencing in 1970. His paper "A Relational Model of Data for Large Shared Data Banks" was a seminal paper, in a continuing and carefully developed series of papers. Dr. Codd built upon this space and in doing so has provided the impetus for widespread research into numerous related areas, including database languages, query subsystems, database semantics, locking and recovery, and inferential subsystems.

- 1985: Preliminary publication of standard for SQL

  Object Oriented DBMSs, Client/Server Architectures, Distributed DBs

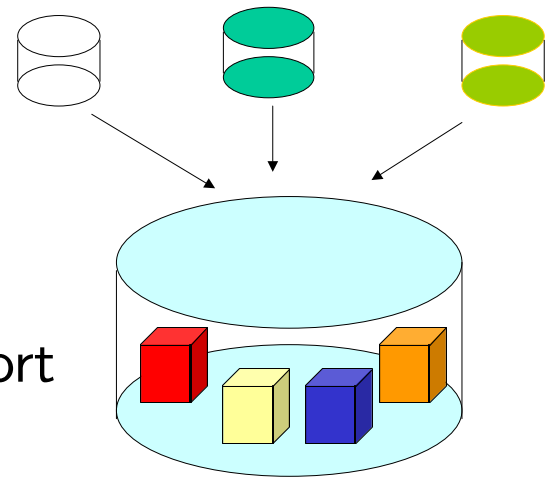
- 80s: Datalog (deductive extension of Relational Calculus with recursion), Deductive DBs

- 90s: Active DBs

  They store *active rules* (or triggers) that automatically react, executing actions when certain events happen inside the DB and under certain conditions

- 90s: Data Warehouses (DWHs)

  Large repositories of physically integrated data from different sources

  Designed for data analysis, business undertanding and decision making support

  DWHs handle complex queries involving massive data

  On-Line Analytical Processing (OLAP)

  Not for operational/transactional purposes (or OLTP, "on-line transaction processing")

  Usually implemented separately from operational DBs

  DWHs can be created inside (as) RDBs (ROLAP)

- 1998: Jim Gray receives the ACM Turing Award

  For seminal contributions to database and transaction processing research and technical leadership in system implementation.

- 1999: SQL3: New release of the Standard: Triggers, Recursive Queries, ... (a few other releases after that)
- 2014: Michael Stonebraker receives the ACM Turing Award

  For fundamental contributions to the concepts and practices underlying modern database systems.

- Many developments and challenges in between ...

  Many due to emergence of Data Science, AI, and ML, in particular

- E.g. In-DB ML?

  Push data-related ML operations inside the DB, where the data are located

  Try to take advantage of the highly optimized data handling mechanisms of the RDBMS

The ER Model Revisited

- In this chapter, we have mentioned some functional issues of DBMSs

  They will be retaken in the next chapter

  However, before they affect us, we need a DB ...

- We retake the subject of DB design

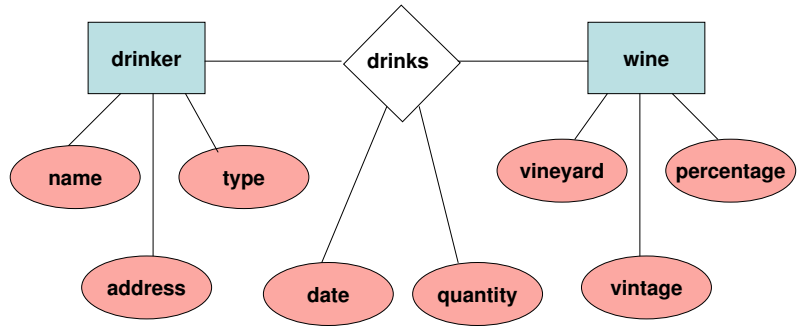
- As we saw before, we usually start with a conceptual model of data

  Say, an ER model

- We will show some elements of the model that we did not introduced in the previous chapter

  Mostly through examples

Example: Elements:



- Entities: Represent classes of objects of the world being modelled

  Also called concepts, entity sets, e.g. "drinker"

- Relationships: Associations between entities; e.g. "drinks"

- Attributes: Properties of entities or relationships

  E.g. the "address" of a drinker

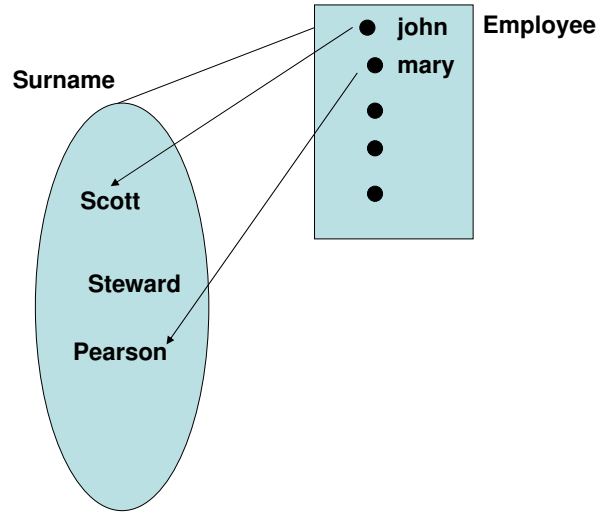
  Or the "quantity" of wine that a drinker drinks

- Links: Used to represent connections between the previous elements

  Most of the time, disregarded, and without a name

  Having names for them is useful for an "ontological extension" of an ER model

- Intuitively, there are "data items" underneath



- Confusingly enough, an element of an entity (as a class) is sometimes called "an entity"

  Hence, using "entity set" instead, and entities can be elements of an entity set

  E.g. "john" could be an entity of the entity set ' "Employee"

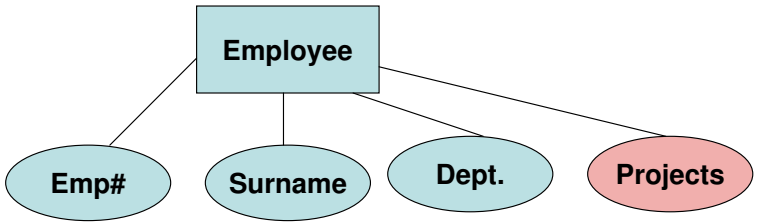
- However, and contrary to the figure above, in the ER itself, we do not see the data items

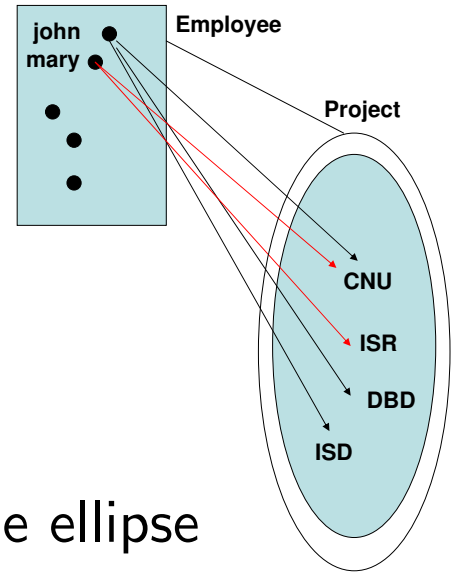
- Usually, an attribute gives a single value to an element of an entity (or to a pair of elements of entities in relationship)

- In a more complex setting we can have a multivalued attribute

  It may assign a set of values to an (element of an) entity

- An Employee may bee assigned to several projects

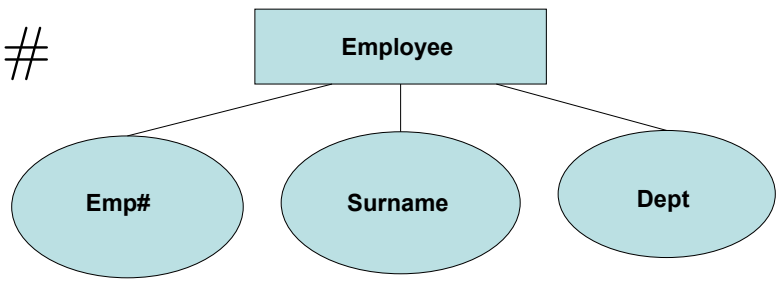




A multivalued attribute

Usually represented with a double ellipse

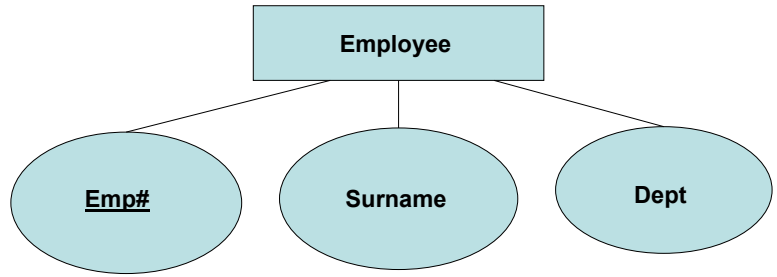
- In this example, attribute Empl# is meant to take a single value for an Employee



Even more, this attribute (value) is expected to functionally determine all the other attributes (attribute values)

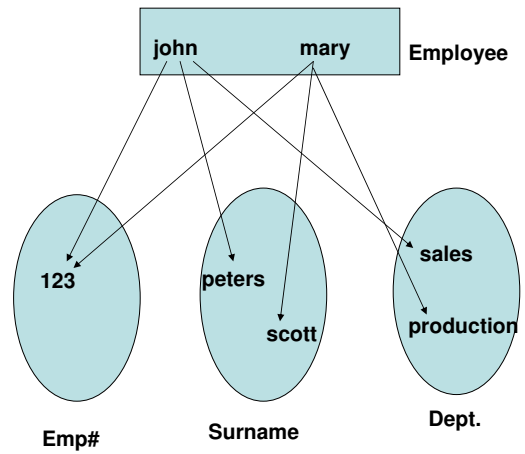
- We say "Empl# becomes a key of the entity Employee

- Represented in the ER model like this (underlined)
- This declares Emp# a key of entity Employee



If any two elements of Employee share the same value for Empl#, then they must share the same values for all the other attributes

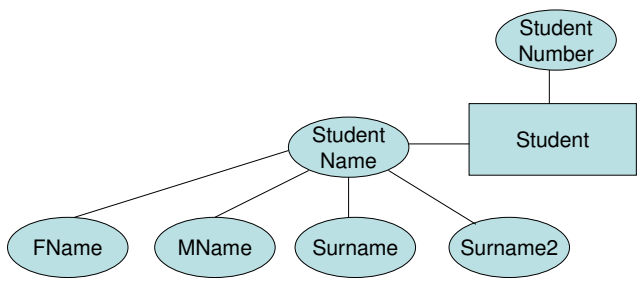
- Something like this cannot happen



- Attributes can themselves have attributes

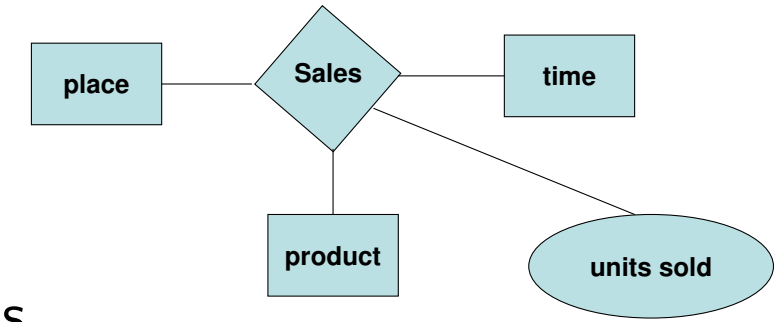
A composite attribute

Student Name is composite



- There may be relationships of arity higher than 2

  Here, a 3-ary relationship



- Particulary interesting!
- Representation of Sales of Units according to three dimensions: Product, Place, and Time

  E.g. 1000 units of "milk" sold in "Toronto" on "August, 12th, 2024"

- An ER model like this is common for data warehouses (DWHs)

  Numerical attributes are given context by certain dimensions

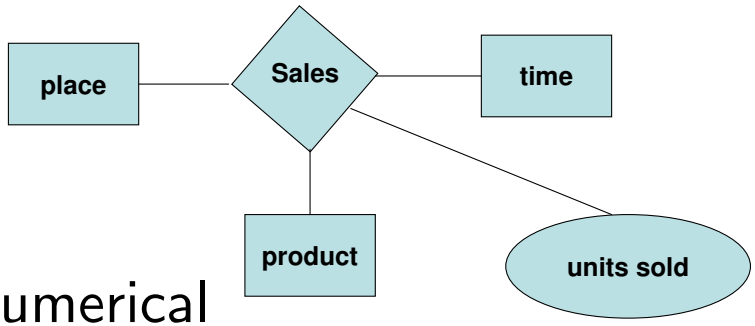
- A multidimensional view of data

  Data to be analyzed according to the given dimensions

- On page 9 a DWH was depicted as a collection of cubes

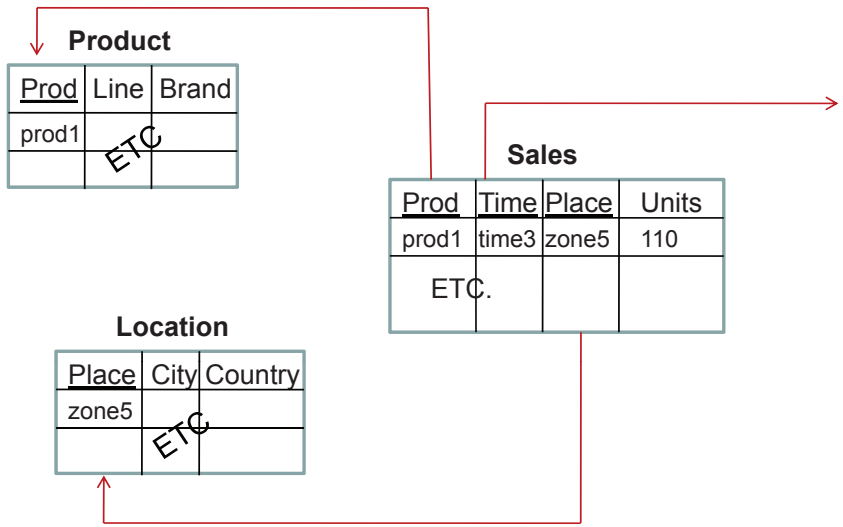
  Their edges corresponding to dimensions

- Going a bit ahead of ourselves: What kind of relational schema could emerge from such an ER model?



- It would give rise to a star schema
- A central table containing the numerical values: the facts table

  In its "periphery", one table for each dimension

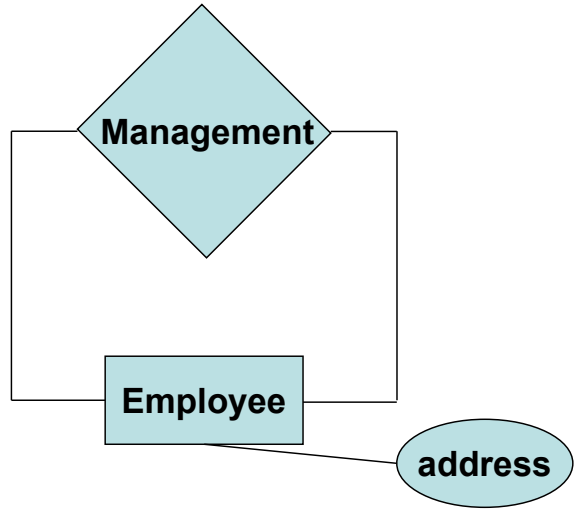


- In "Sales" table, the key is formed by the three underlined attributes
- We also need referential constraints, etc.

  We will come back to translations from ER to relational models

- We can have recursive relationships

  Employees manage other employees

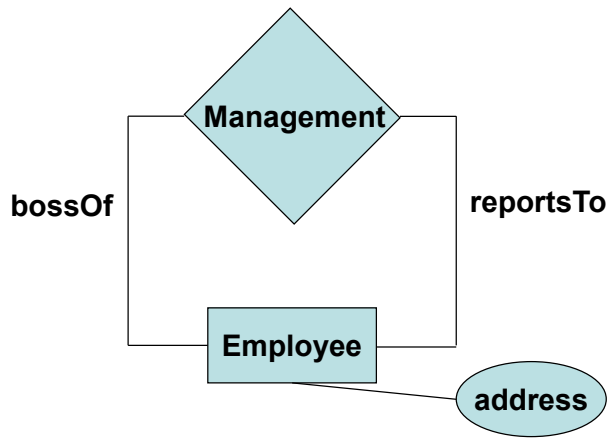


- Who are managers and who are managed?

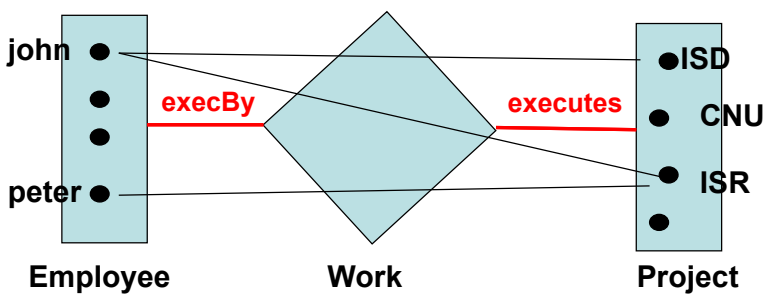
  Not clear ...

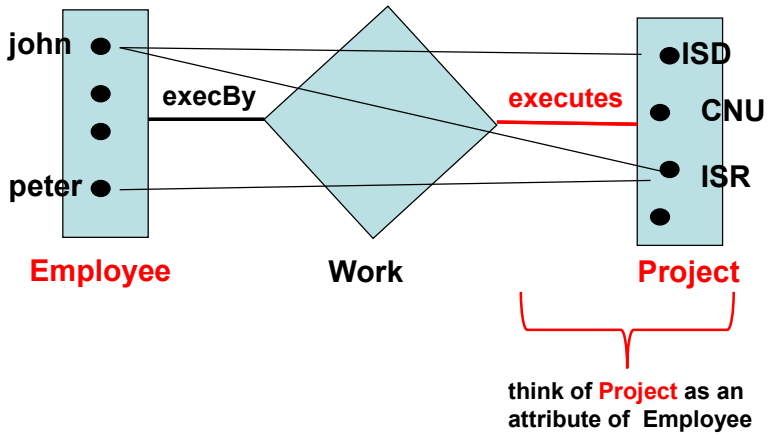
- This is a case when link names could come handy ...

  Also called “roles” in ontological languages



- Another example





- "john executes project ISD", "proyect ISR is executed by peter", etc.
- Roles names can be useful when transforming an ER model into a formal ontology (coming much later)

  They emerge from a relationship of the ER model