IPRs and their data analysis

Module 4 – Relational databases

Jacopo Staccioli, PhD[†‡]

[†] *Università Cattolica del Sacro Cuore, Milan*
[‡] *Scuola Superiore Sant'Anna, Pisa*

a.y. 2023-24

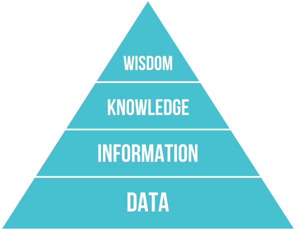
Outline

1. **Relational databases**
   - What is data
   - What is a database
   - Flat-file
   - Relational model
   - Non-relational databases

What is data

data is a *representation* of “something” that captures some *features* and ignores others

- a set of values of *qualitative* or *quantitative* variables about one or more “objects”
- data is the result of *observation* and *collection* (or deliberate *generation*)
- data only becomes useful *information* once it has been analysed in some fashion
- data is often assumed to be the least *abstract* concept



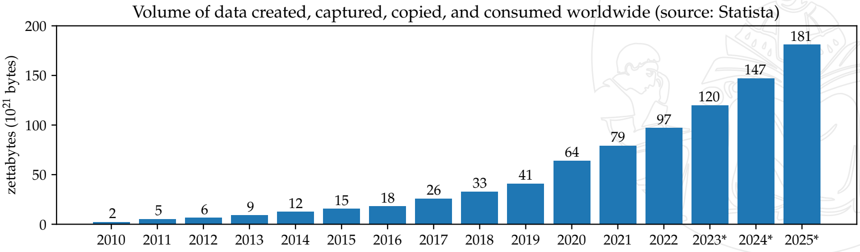
What is data (cont'd)

analog data : embeds *continuously* changeable aspects of the underlying phenomenon

- e.g. sound on a magnetic tape, image on a photographic film

digital data : uses *discrete* sampling (*quantisation*) to encode what is being measured

- a sequence of 1s and 0s (*bits*), typically grouped as *bytes* (8 bits)



What is data (cont'd)

- suppose you have an electronic copy of every patent ever published
    - e.g. a directory or archive containing tens of millions of PDFs
    - *collection* of data or data *store*

Q how many patents does a certain company own?

Q which patents belong to a certain technological class?

Q who is the most prolific inventor?

- straight away, you can only say how many patents are there, at best

unstructured data does not have a pre-defined *model* or is not organised in a pre-defined manner

data *model* organises elements of data and standardises how they *relate* to one another and to the properties of real-world entities

Outline

1. **Relational databases**
   - What is data
   - What is a database
   - Flat-file
   - Relational model
   - Non-relational databases

What is a database

- a *database* is an *organised* collection of data
- a *database management system* (DBMS) is a piece of *software* that enables users to define, create, maintain, and control access to the database
- regardless of the underlying amount of data (*scalability*)
  - the storage medium must be *reliable*
    - should guarantee data *integrity* and be resilient to *corruption*
  - storing data must be *efficient*
    - should be *fast* and avoid *duplication*
  - retrieving data (*querying*) must be *efficient*
    - should be fast
    - separating what should and should not be retrieved must be easy

What is a database (cont'd)

- a well designed database should be a *self-describing* collection of integrated *records*

record : a representation of some physical or conceptual object

- e.g. a patent, an inventor, a firm

metadata : data that provides information about other data

- e.g. *description* of data structures within the database

What is a database (cont'd)

database model : determines the logical structure of a database and fundamentally determines in which manner data can be stored, organised and manipulated

- there exist many different models

  flat-file : e.g. a single table stored as a csv or tsv file

  relational : most popular database model

  document : useful for full-text analysis

  ...

- the optimal structure ultimately depends on
  1. the natural organisation of the underlying data
  2. the requirements of the specific application

Outline

1 **Relational databases**
- What is data
- What is a database
- Flat-file
- Relational model
- Non-relational databases

Flat-file database

- suppose you want to store data about patent numbers and their inventors in a flat file

auth	num	kind	year	name	surname	sex	address	city	country
EP	0000001	A1	1990	John	White	M	22 Deer Rd	London	UK
EP	0000001	A1	1990	Ann	Beech	F	16 Argyll St	New York	US
EP	0001234	B1	2000	David	Ford	M	163 Main St	Sydney	AU
EP	1234567	B2	2010	Mary	Howe	F	32 Manse Rd	London	UK
EP	1234567	B2	2010	Susan	Brand	F	56 Clover Dr	Auckland	NZ

- there are 3 patents and 5 inventors
- there is significant *duplication*
  - $\#rows = \max(\#patents, \#inventors)$
  - even more if other columns are included (e.g. firms, technological codes, references, …)

Flat-file database (cont'd)

- no way of recognising *relationships* between records
  - relationships can be inferred from the data, but flat files do not make relationships *explicit*
- no way of imposing *constraints* and ensure *consistency*
  - e.g. patents must have a human inventor (*inventorship* requirement)
  - every human being must have a name, surname, sex, …
  - $\Rightarrow$ it should not be possible to leave blank the attributes for `name`, `surname`, `sex`, …
- no way of *indexing* attributes (columns)
  - retrieval operations are not efficient
  - $O(n)$: a query needs to check every record (line) in the file (*linear* lookup time)

time complexity and *big O notation*

- asymptotic behaviour of a function when the argument tends towards infinity ($\lim_{n\to\infty}$)
- used to classify algorithms according to how their run time grows as the input size n grows

Outline

1. **Relational databases**
   - What is data
   - What is a database
   - Flat-file
   - Relational model
   - Non-relational databases

Relational model

- a *relational DBMS* is a DBMS based on the *relational model*[1]
  - a *relation* (table) is a set of *tuples*
  - a *tuple* (row) is a set of *attribute* values
  - an *attribute* (column) specifies a data *domain*
  - a *domain* (data type) define admissible attribute values, e.g. `integer`, `float`, `string`
  - *constraints* further restrict admissible data values if required, e.g. `unique`, `not null`
  - the *cardinality* of a relation equals the number of tuples
  - the *degree* of a relation equals the number of attributes
  - the *schema* is the overall logical structure of relations in a database
- in a nutshell, a relational database is a set of connected tables holding consistent data

[1] Codd, E. F. (1970) "A Relational Model of Data for Large Shared Data Banks". *Communications of the ACM* 13(6), pp. 377–387. DOI: 10.1145/362384.362685

Example

attributes

patents

relation

id	auth	num	kind	year
1	EP	0000001	A1	1990
2	EP	0001234	B1	2000
3	EP	1234567	B2	2010

cardinality

degree

primary key

foreign key

inventors

relation

id	name	surname	sex	address	city	country	patent_id
1	John	White	M	22 Deer Rd	London	UK	1
2	Ann	Beech	F	16 Argyll St	New York	US	1
3	David	Ford	M	163 Main St	Sydney	AU	2
4	Mary	Howe	F	32 Manse Rd	London	UK	3
5	Susan	Brand	F	56 Clover Dr	Auckland	NZ	3

Database keys

primary key : *choice* of a minimal set of attributes that *uniquely* specify a tuple in a relation

- e.g. the `id` attribute of the `patents` relation
- could be multiple attributes concatenated together, if their union is unique
- e.g. `auth` ⊕ `num` ⊕ `kind`
- called *surrogate key* if it consists of a database-specific identifier (e.g. the `id` URI)
- called *natural key* if it consists of application-specific data (e.g. whole pub. number)

foreign key : one or more attributes that match the primary key of another relation

- used to *cross-reference* different relations
- implicit constraint: foreign key values must exist as primary key in the referenced relation

Database indices

index : a data structure that improves the speed of data retrieval operations on a relation

- cost entails additional writes and storage space to maintain the index
- can be created on any set of attributes
- avoid searching every tuple when locating specific attribute values
- lookup performance is typically sub-linear $O(\log n) \ll O(n)$ of flat files
  - usually implemented as a self-balancing `B±tree` or a hash `table`
- primary keys implicitly create an index on the underlying attribute

Database reliability

transaction : symbolizes a single unit of work performed within a DBMS

- generally represents any change in a database
- e.g. transfer of funds from one bank account to another
- ACID requirements for *reliable* transactions

A*tomicity* ensures that a transaction is treated as a single unit, which either *succeeds* completely, or *fails* completely

C*onsistency* ensures that a transaction can only bring the database from one *valid* state to another (e.g. constraints are enforced)

I*solation* ensures that *concurrent* execution of transactions leaves the database in the same state as if the transactions were executed *sequentially*

D*urability* ensures that once a transaction has been *committed*, it will remain committed even in the case of a system failure

Popular RDBMS

MySQL

SQLite

MariaDB

IBM DB2

ORACLE

Microsoft SQL Server

teradata.

Outline

1. **Relational databases**
   - What is data
   - What is a database
   - Flat-file
   - Relational model
   - Non-relational databases

Non-relational databases

- designed for data modelled in means other than tabular relations
- also known as *NoSQL* (after Strozzi, 1998)[2] or *"not only SQL"*
- query language is typically SQL-like or lower-level
- many available models for different data architectures

  document : suitable for storing *semi-structured* data

  key–value : a *key* uniquely identifies a *record* whose *value* can be of any *type*

  - suitable for storing *associative arrays* (akin to Python `dict`)
  - the value is *opaque* to the database

  wide column : names and format of columns can vary across rows in the same table

  - akin to a 2-dimensional key–value store

  graph : suitable for storing *network* data (*nodes*, *edges*, *properties*)

[2] http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page

Document-oriented databases

- *document oriented DBs* are so popular that sometimes are used as synonym for NoSQL
- unlike *tuples* in a *relation*, documents can differ in their structure from one another
  - documents are not required to adhere to a standard *schema*
- internally implemented as a "specialised" key–value store, where the value is *not* opaque
- document structure and metadata are used for query optimisation and fast traversal
- typical encodings include `XML`, `YAML`, `JSON`, `BSON`

mongoDB CouchDB relax elasticsearch