3. Databases and Geographical Information Systems (GIS)

Databases

Motivation:

Computers are often used

- for dealing with **large amounts of data**
- and in situations where **data integrity** is important for the survival of an organization.

Examples:

- Banking
- e-commerce (commercial transactions via WWW – e.g., **`amazon.com`** or **`ebay.com`**)
- meteorological measurements
- booking systems (trains, airlines...)
- telecommunication (phone numbers, fax numbers, mobile phone data...)

Main problems:

- How can large amounts of data be organized so that they can be accessed quickly?
- How can data be organized so that hardware and software failures do not lead to a desaster?
- How can data be changed by several agents in parallel without interference?

Today these problems are being dealt with on the conceptual basis of **relational database management systems** (RDBMS), typically using some dialect of **SQL** (structured query language) as notation for definition and manipulation of data.

In these slides: Only very basic concepts are discussed.

Introduction using an example

Simplistic example: public library. Data organized in **tables**.

- table "Users" with columns `UserID, Name, Address, BirthDate`
- table "Books" with columns `BookID, Title, Author, Keywords`
- table "BorrowedBooks" with columns `UserID, BookID, BorrowedSince, BorrowedUntil`

Principles of database tables

- Relational databases hold the data in (typically several) **tables**.
- Each row represents one **record**.
- The number and meanings of the **columns** of a table is (more or less) **fixed**.
- The number of **rows** of a table is **variable**.

"Entity relationship model":

- Each table describes one kind of **entities** or a **relation** (typically between several entities)
- a model of a certain part of reality based on the concepts of entities and their relationships is called an entity-relationship model.

In our example:
tables "Books", "Users" represent entities,
table "BorrowedBooks" represents a relation between these entities.

Attributes, key candidates and keys

Columns in a table are called **attributes**. Some attributes or attribute combinations **characterize** entities. Such attributes or attribute combinations are **key candidates**. One of the key candidates is designated as **primary key**. The primary key of an entity is used in order to refer to it from other entities or from relations.

In our example, `UserID` is used as primary key in the "Users" table, and `BookID` is used as primary key in the "Books" table. These attributes are used in "BorrowedBooks" in order to refer to the related entities.

Data definition and data manipulation with SQL

Two kinds of languages for working with relational data bases are distinguished:

data definition language (DDL)

data manipulation language (DML)

DDL and DML are today typically combined in dialects of SQL (structured query language) and supported by producers of database management systems. The different dialects are based on similar principles. We will give examples.

Data definition consists in the definition of the structure or tables and their interrelations.

During data definition, it must be defined for each table:

- which **attributes** it contains,
- how each attribute is to be **represented** (a data type must be chosen),
- which attributes form the **primary key** of the table, and
- which attributes refer as **keys** to other tables.

A notation which allows to define tables in this way is called a **data definition language** (DDL).

Data manipulation consists in adding, changing and deleting table rows and in the selection of data from the data base.

A DDL only alows to describe the structure of a data base, not to change its content in any way.

A notation which allows to manipulate tables is called a **data manipulation language** (DML).

Data definition

The “Users” table from the public library example could be defined like this:

```
CREATE TABLE Users (
  UserID     INT(10) NOT NULL,
  Name       CHAR(100),
  Address    CHAR(100),
  Birthdate  DATE,
  PRIMARY KEY (UserID)
)
```

This instruction creates a table names “Users” with the four already described columns. `UserID` is represented a ten-digit decimal number, `Name` and `Address` are represented as 100 characters, `Birthdate` as a date, and `UserID` is the primary key of the table.

For `UserID`, a value must be given for each row in the table – for the other three columns, a standard value (`NULL`) might be used in order to designate that the value of the attribute is not known.

The table “Books” might be defined similarly, only the attribute `Keywords` presents problems. Which amount of memory should we reserve for the keywords of a book if we do not want to restrict the number of keywords beforehand?

One solution consists in the definition of an extra table “Keywords”:

```
CREATE TABLE Keywords (
  BookID  INT(10),
  Keyword CHAR(100)
)
```

Key words have a maximal length of 100 characters, but the number of key words which can be given for a book is not restricted, since the same book can occur any number of times in the table.

The “Books” table could be declared like this:

```
CREATE TABLE Books (
  BookID INT(10) NOT NULL,
  Title  CHAR(100),
  Author CHAR(100),
  PRIMARY KEY (BookID)
)
```

The table representing currently borrowed books might be declared like this:

```
CREATE TABLE BorrowedBooks (
  UserID      INT(10),
  BookID      INT(10),
  BorrowedSince DATE,
  BorrowedUntil DATE
)
```

Data manipulation

The following operations can be used to manipulate the data in the tables:

- The **SELECT** command selects information from the data base.
- The **INSERT** command inserts rows into a table.
- The **UPDATE** command changes the content of existing rows in a table.
- The **DELETE** command removes rows from a table.

SELECT

The list of overdue books can be determined as follows:

```
SELECT b.BookID, b.Author, b.Title, l.BorrowedSince
FROM Books AS b, BorrowedBooks AS l
WHERE     b.BookID = l.BookID
      AND l.BorrowedUntil < TODAY
```

This statement is also called a **query** (the data base system is queried for some data).

This query returns a **table with four columns**. Each row represents an overdue book; the first column contains the book id, the second the author, the third the book title, and the last column the date when the book was borrowed.

A query has the following form:

- After the keyword **FROM**, the tables are listed from which data is to be collected. We use all combinations of rows from “Books” and “BorrowedBooks”, and we abbreviate “Books” as “b” and “BorrowedBooks” as “l” elsewhere in the query.

- The **WHERE** keyword defines a filter: only those combination of rows from the **FROM** clause are kept which fulfill the condition given behind the **WHERE**: The book ids of the two entries must match, and the date until which the book must be given back must lie in the past.

- The **SELECT** keyword introduces a list of expressions which are evaluated for each row combination filtered out by the **WHERE**. In the example, these are simply some of the attributes.

INSERT

When a book is borrowed, a row has to be added to table `BorrowedBooks`. The following instruction adds a row with `UserID` `1053465`, `43565` as `BookID`, `TODAY` as `BorrowedSince` and `TODAY+14` as `BorrowedUntil`. The order of the arguments is the same as the order of the columns in the table declaration.

```
INSERT INTO BorrowedBooks
VALUES (1053465, 43565, TODAY, TODAY+14)
```

The general form is the following: After the keywords `INSERT INTO` and the name of the table, the keyword `VALUES` starts a list of values representing the row to be inserted.

UPDATE

In order to lengthen the borrowing time of the book with id 43565 by a week, the following command could be executed:

```
UPDATE BorrowedBooks
SET BorrowedUntil = BorrowedUntil + 7
WHERE BookID = 43565
```

After `UPDATE`, the name of the table to be changed is given. The `WHERE` predicate defines which rows are affected by the change, and after `SET` it is defined which columns in the rows to be changed are updated, and to which value.

DELETE

When a book is brough back by a used, its entry has to be taken out of the “BorrowedBooks” table:

```
DELETE FROM BorrowedBooks
WHERE BookID = 43565
```

Further elements of the SQL language

Above we have only seen the most elementary SQL language elements. Many SQL dialects present many more features.

Examples:

- **Integrity constraints** can be used in order to define *conditions on the content* of a database which shall never be violated during manipulations.
- **Foreign key relations** are used in order to make explicit that values in a column are keys of some other table. They are a special case of integrity constraints.
- **Index declarations** are used in order to accelerate searching in tables.
- **Stored procedures** are used in order to store instructions which are to be executed by the database.
- Further **table operations**: *set union*, *set difference*, *set intersection*, *grouping* of results, *sorting* of results.
- **Views** allow to shield the users of a database from the internal representation of the data.

- **Database administration** consists in deciding how tables etc. are represented and which users get which kind of access to the database.
- **Invariants** and **triggers** are language elements which ensure the fulfillment of integrity constraints independently of the application programme.
- **Transactions** are language elements which ensure that a sequence of changes is either executed *completely* or *not at all*, even in the case of hardware or software failures.

Conceptual database design

The **conceptual design** of a relational database often proceeds according to the following steps:

- First the **entities** relevant in the application area are collected and their types are determined. (Example: books, users)
- Then the relevant **relationships** between entities are determined. (Example: BorrowedBooks)
- For each entity type and each relationship type, the **attributes** and their data types are determined.
- Finally, **integrity conditions** for the database are specified. (Example: BorrowedUntil must not be earlier than BorrowedSince)

On the basis of this design it is decided how entities, relationships, attributes are represented in a specific database management system.

Normalization:

Redundant data in a data base might lead to **inefficiencies** and **inconsistencies**: Updates of redundantly held information have to be performed at several locations instead of at only one, and if this is forgotten, an inconsistency results.

Normalisation of a database consists in the reduction of redundancies, typically via splitting tables.

Architecture of database applications

Database applications often use a *three-layer* architecture:

- A **DBMS** operates as the kernel of the system. It ensures data persistency, data integrity etc.

- An **application layer** provides application-specific functionality. In our example, it would provide the functions “borrow a book”, “lengthen borrowing time”, “register new user” etc.

- A **presentation layer** defines the user interface, which today is often graphical, and not seldom with an alternative using the WWW.

These three components might run as **three different programs** on different computers: A **web-browser** runs the presentation layer, the web-server dispatches the user input to an **application program**, and the application program accesses a **relational database** on a dedicated database server.

Geographical Information Systems

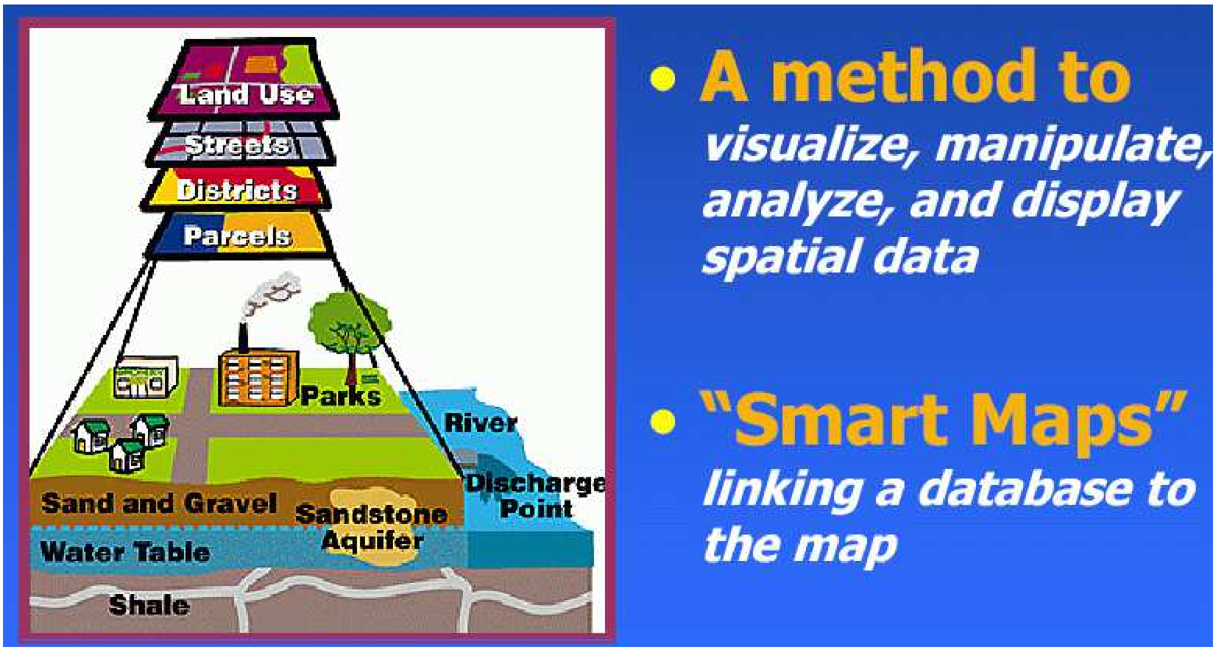
What is a Geographical Information System (GIS)?

- Software, hardware and data to help manipulate, analyse and present information that is tied to *spatial locations* (usually geographical locations).

Estimates are that 80 % of all data stored worldwide has a *spatial* component (Source: www.gis.com).

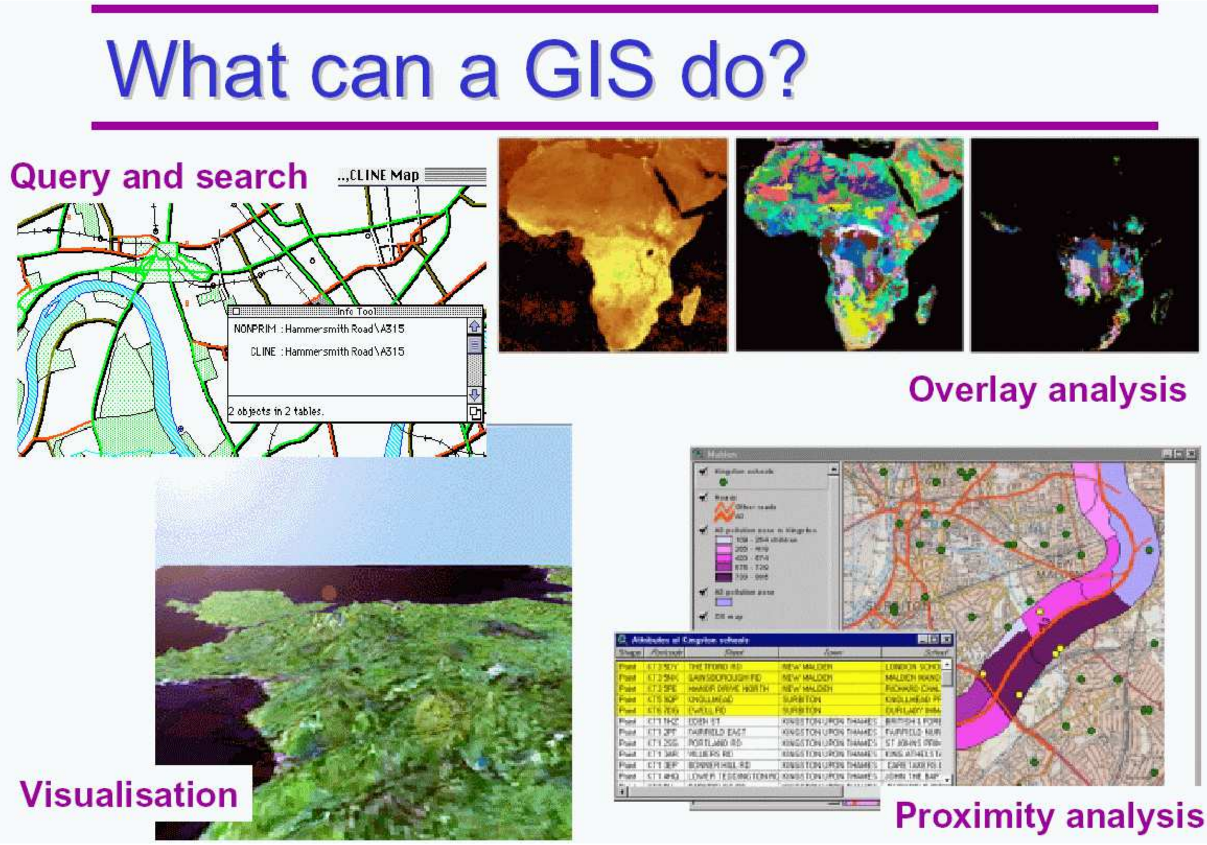
A GIS contains a classical database, but extends its functionality by methods adapted to spatial information.

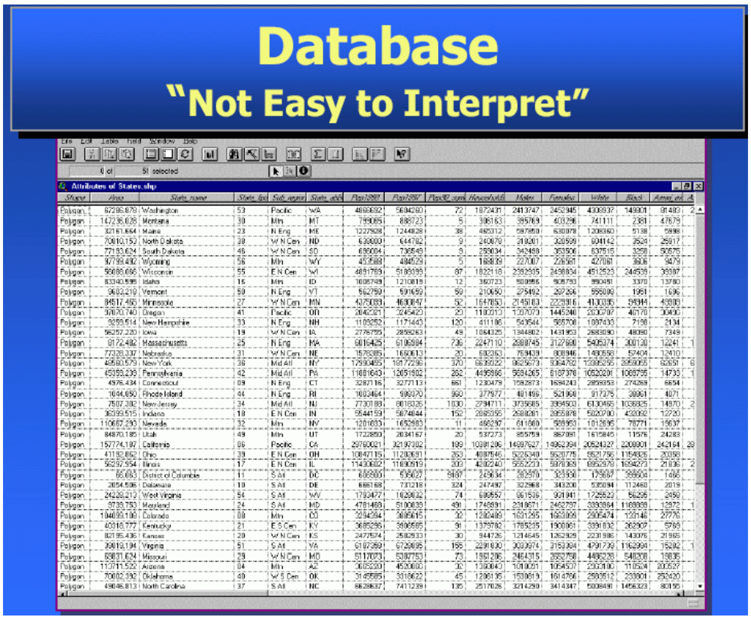
Particularly, a GIS provides...

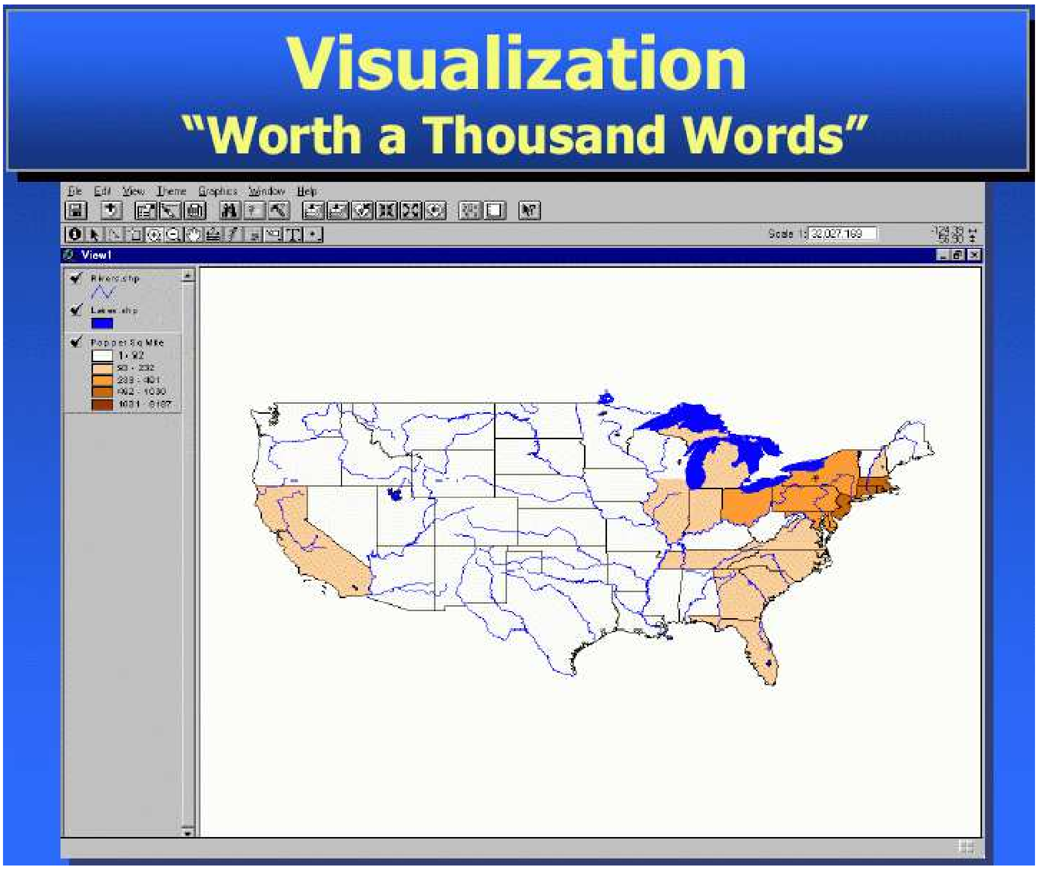


- special forms of *query*, designed to extract information with spatial properties from a data-base (e.g., taking neighbourhoods into account)

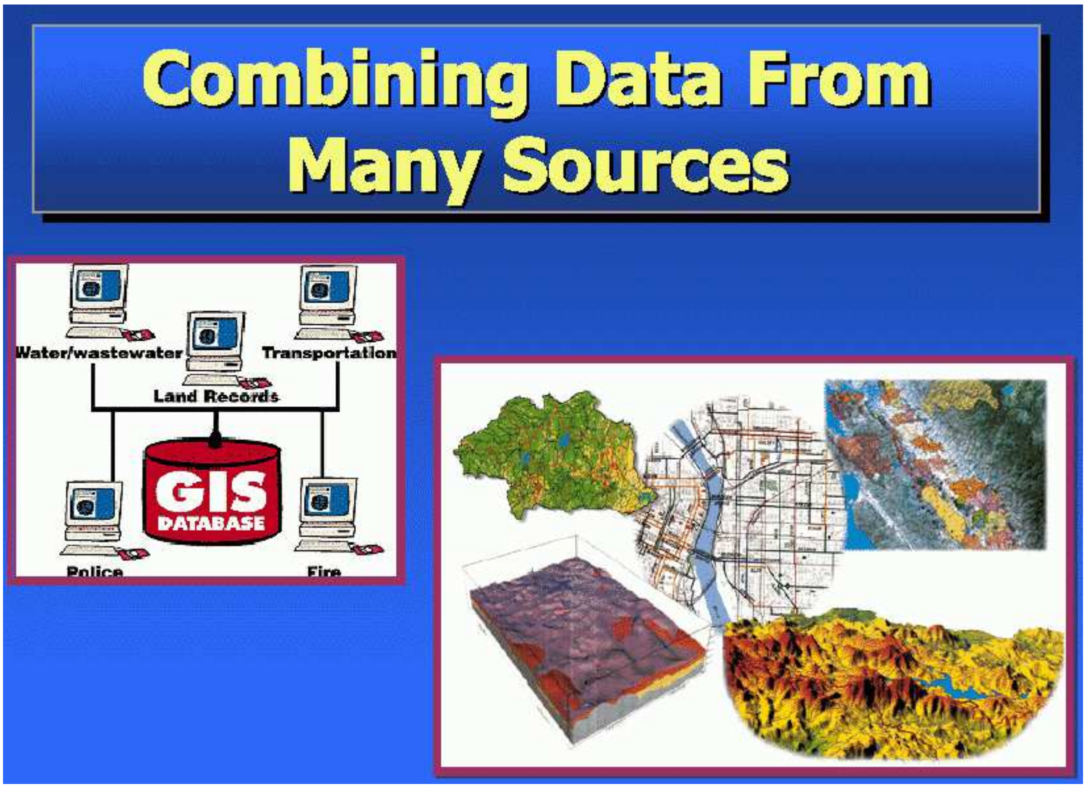
- special forms of *data analysis* (e.g., geostatistics)
- special forms of *integrity checking* adapted to spatial data.

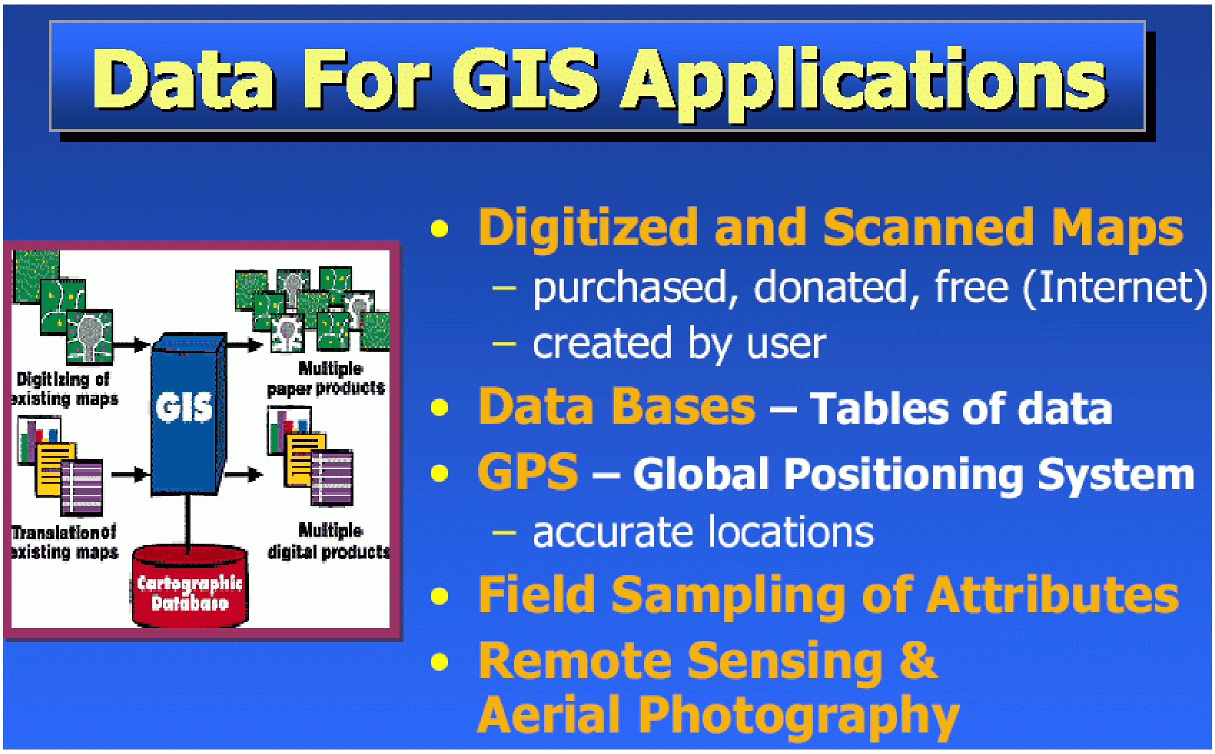




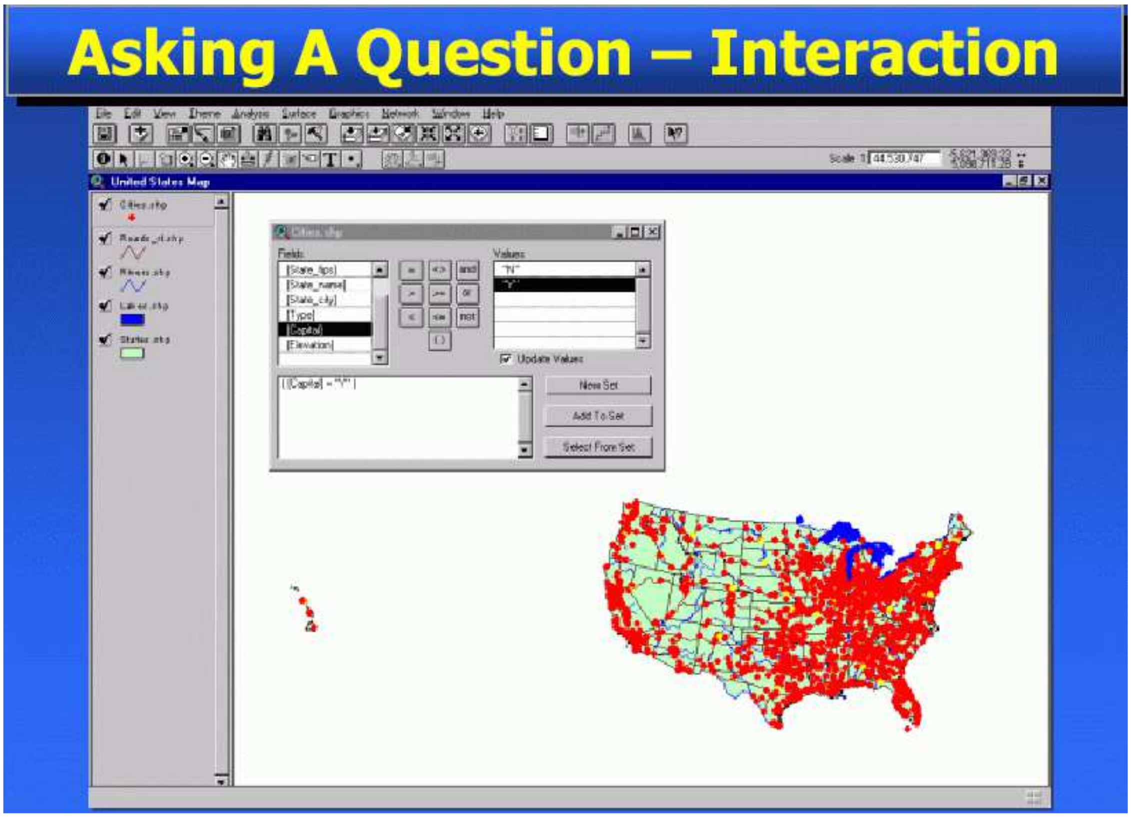


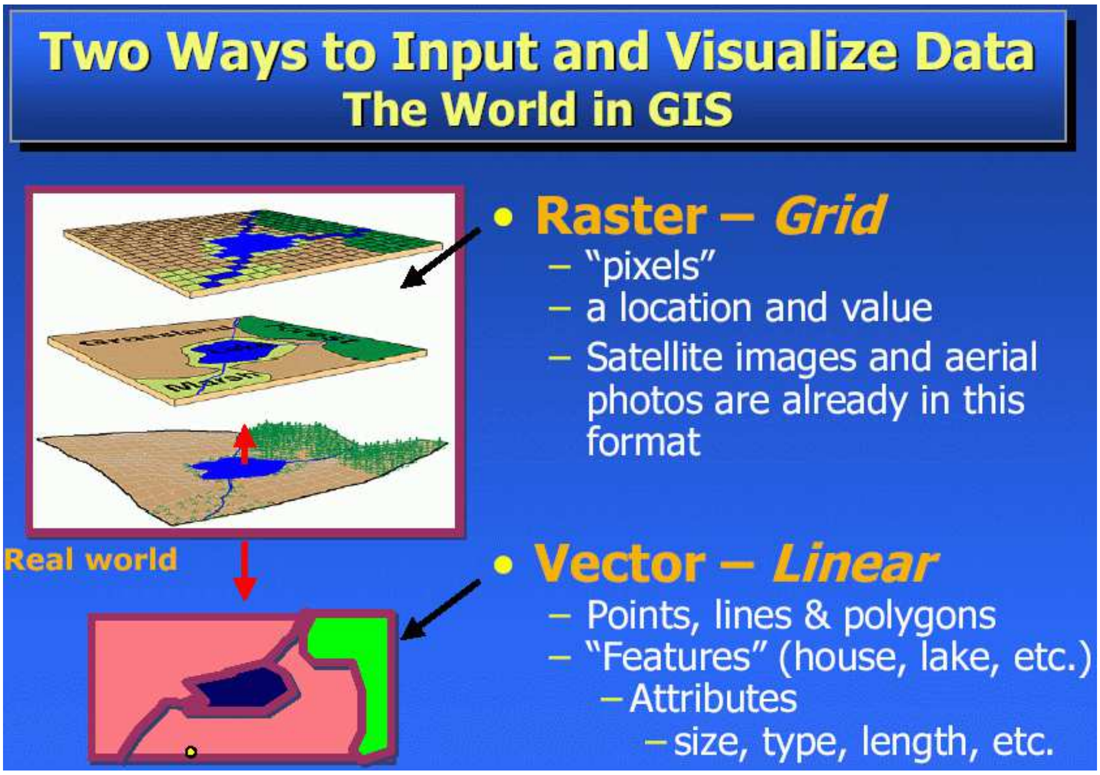
One of the main advantages of GIS over classical geographical maps:





Further advantage: Easy interaction, visualization, manipulation of maps

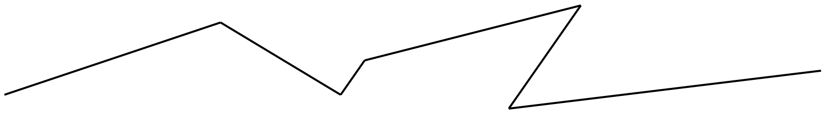




The vector representation is more appropriate for senseful queries (and is more exact)
– basis for relational database representation of geographical data

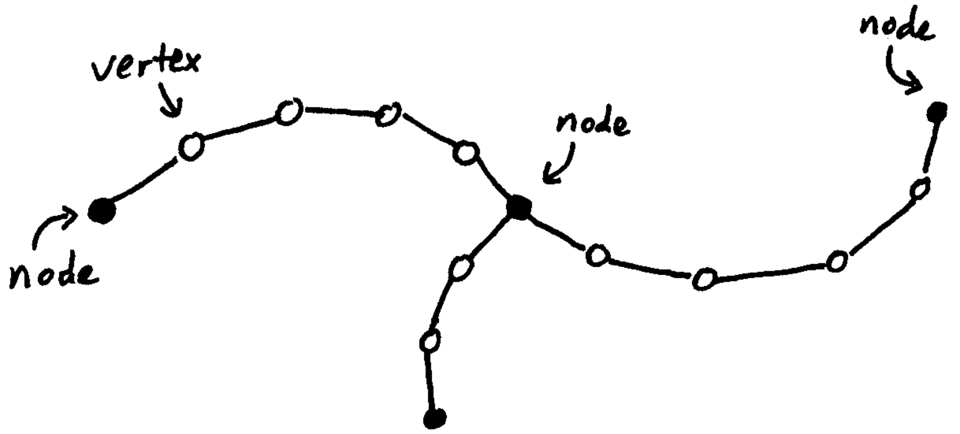
Typical *entities* of a GIS:

- Points
- *Tics* (= special points for which the exact real-world coordinates are known, used to fit a digital map into a global coordinate system)
- *Lines*, also called *arcs* (more precisely: Multilines, i.e. consisting of several linear segments)



- *Polygons* (closed multilines, possibly with additional attributes)
- *Annotations* (text objects associated with points).

The endpoints of a line (and possible branching points) are called *nodes*.
Intermediate points (without branching) are called *vertices*.



Tables in the underlying relational database:

- Tic table
- boundary table (represents the spatial extent of the map – a surrounding rectangle)
- arc attribute table (AAT)
- polygon attribute table (PAT).

E.g., a *polygon* is represented as a line in the PAT, with attributes:
polygon ID, nodes, arcs, a label point (in the interior), further attributes (e.g., area, slope, population density...).
Details differ between different GIS.

Usually, a GIS does not only contain information for a single map of a region, but *several sorts of information for the same region*:

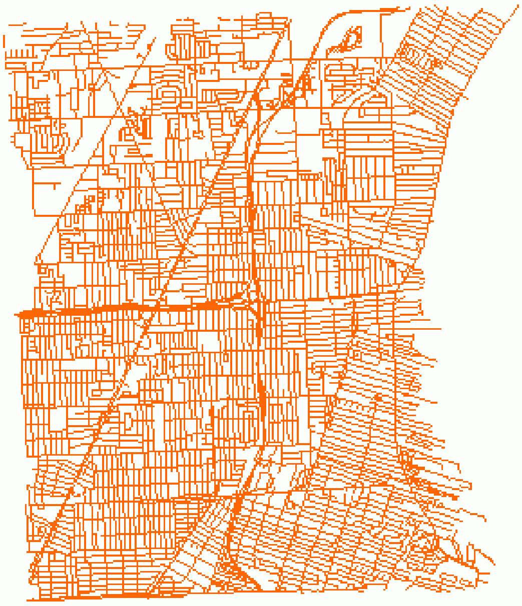
each sort of information is represented in an extra *coverage* (also called *layer*, *cover* or *theme*).

Example: Different coverages of a town area

Road Centerline/Address (Geographic Township)

- source: 2000 Digital Orthophotography

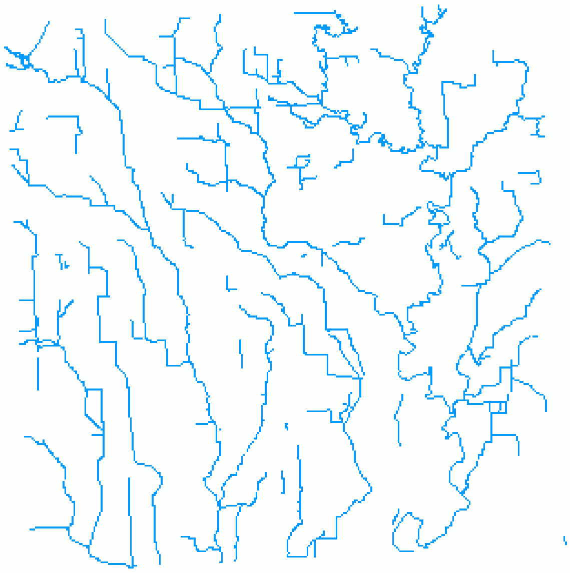
- Road Arcs
- Address Ranges
- Road Names



Hydrology (Geographic Township)

- source: 2000 Digital Orthophotography

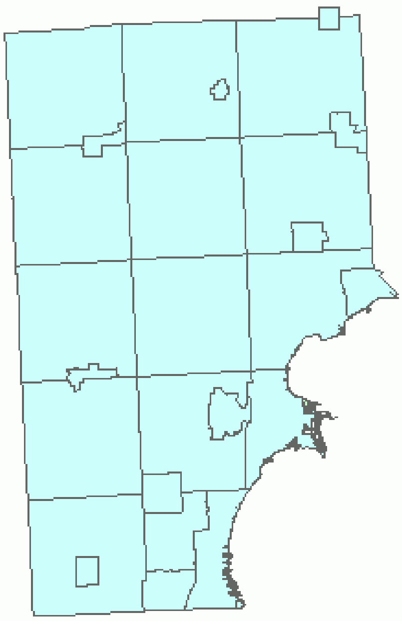
- Water feature arcs (rivers, streams, drains)
- Water feature polygons (lakes, ponds, retention basins)
- Names of County maintained drains



Municipal Boundaries (County)

- source: 2000 Digital Orthophotography

- Community polygons
- Community name

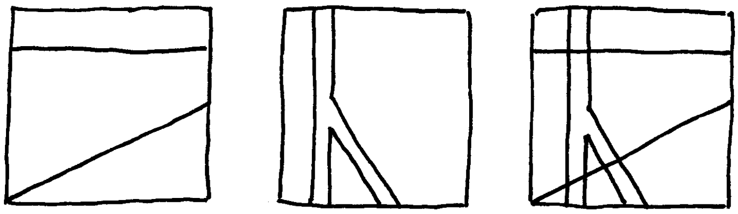


(source: `http://macombcountymi.gov/gis/gis_coverage_samples.htm`)

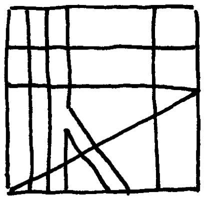
How to combine several coverages?

- *Overlay operation*

From two geometries, the GIS calculates the coarsest common geometry:



a *b* overlay of *a* and *b*

Attention: The following geometry  would also be a common geometry of *a* and *b*, but not the coarsest one!

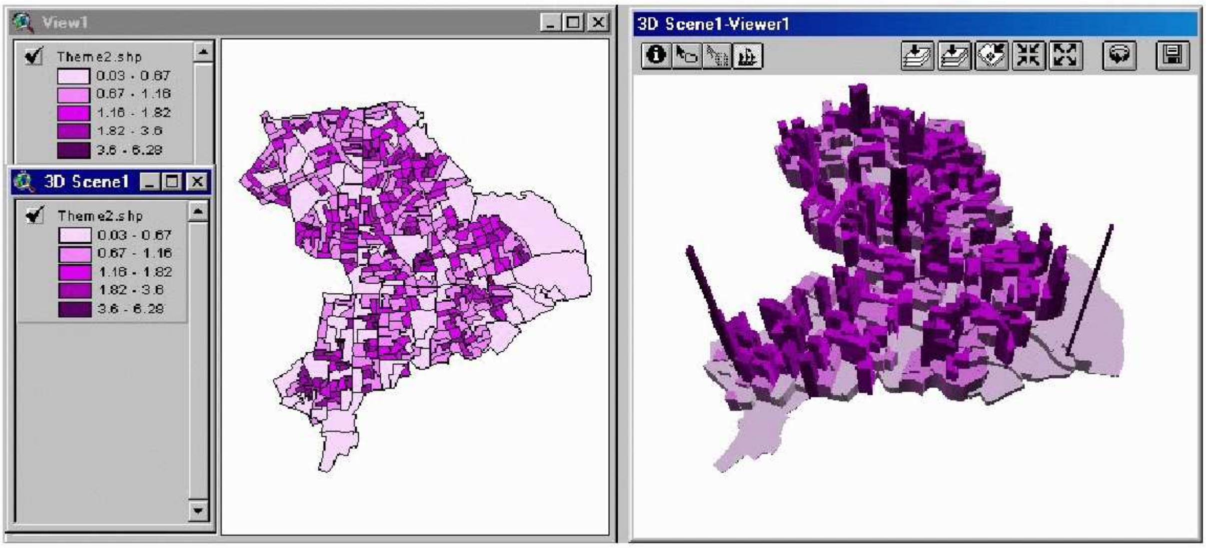
Using overlay, a GIS can give answer to questions like this:
"What forest areas of district *x* are within 100 m distance to a road, are stocked with conifers and have a slope < 5 degrees?" (e.g., for a chalking action)

Layers used for this task:

- landuse map (→ forests)
- political district map (→ district *x*)
- road map (→ 100 m neighbourhood to a road)
- forest type map (→ stocked with conifers)
- digital elevation model (→ slope < 5 degrees)

Selection of polygons of the overlay using an "and" operation

Further functionality of GIS: 3D visualization



[redacted] *Representing Attribute Data in 3-D: Population Density in Small Census Areas in the London Borough of Hackney*

Widely used GIS products:

- ESRI ArcGIS (licenced commercial software)
- QuantumGIS (free and open source, **`http://www.qgis.org`**)

4. The World Wide Web

The World Wide Web (WWW) is
a *hypertext system* which is accessible via *internet*

(WWW is only one system using the internet – others are e-mail, ftp, telnet, internet telephone ...)

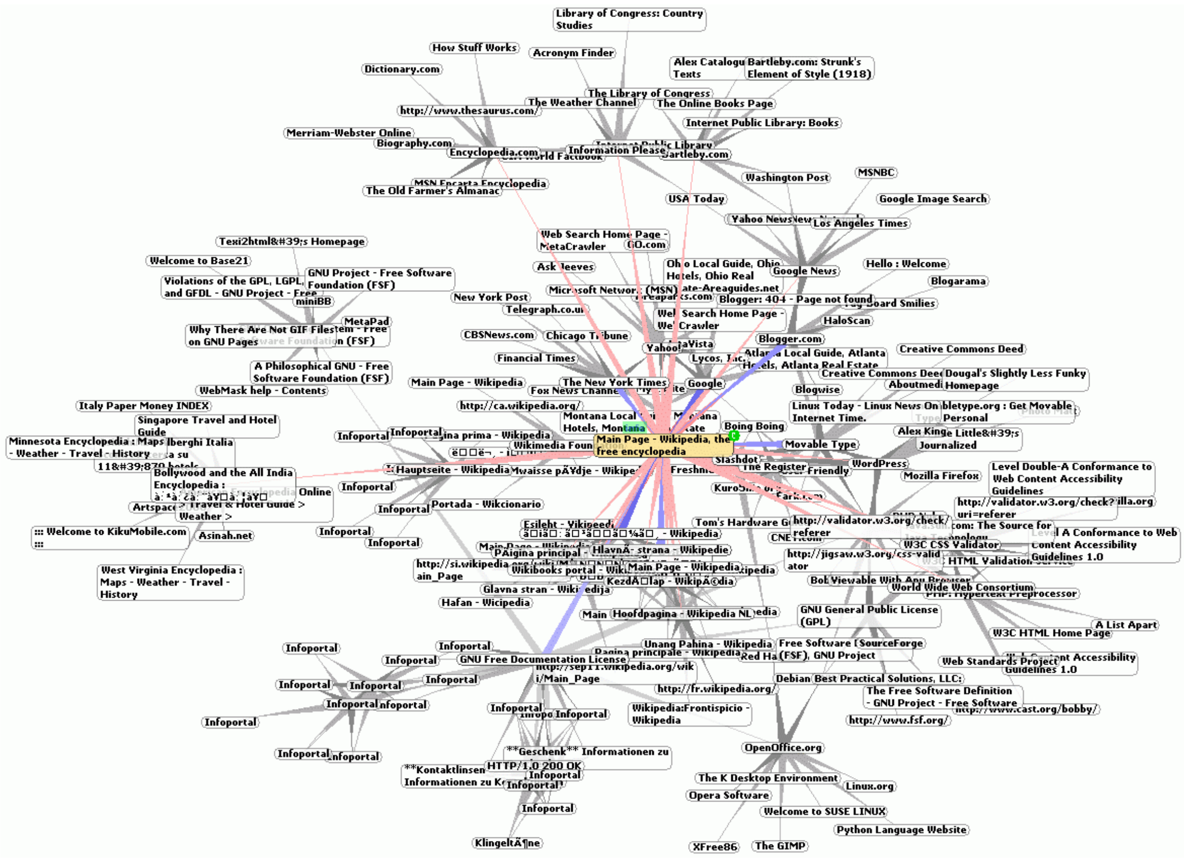
Hypertext: Pages of text containing *hyperlinks* (short: *links*) referring to other pages

Hypertext is viewed using a program called a web browser which retrieves pieces of information, called "documents" or "web pages", from web servers and displays them, typically on a computer monitor. One can then follow hyperlinks on each page to other documents or even send information back to the server to interact with it. The act of following hyperlinks is often called "*surfing*" or "*browsing*" the web. Web pages are often arranged in collections of related material called "web sites."

(from **`www.wikipedia.org`**, the open www encyclopedia)

The link structure of the web forms a very large graph –

the following is a very small subgraph of it:



The Web can be seen as a sort of database – but very different from relational databases:

- highly distributed, decentralized;
- based on the hypertext model instead of the entity-relationship model;
- with only very weak standards to restrict form and content of the pages;
- very large
- without a universal query language.

(*Search engines* try to compensate the last item; see below.)

History of the WWW:

- Idea of hypertext: Vannevar Bush 1945
- Origin of WWW: a project at CERN (Geneva) in 1989
- *Tim Berners-Lee* and *Robert Cailliau*
- their system: ENQUIRE, realized core ideas of the Web in order to enable access to library information that was scattered on several different computers at CERN
- proposal for the WWW: published by Berners-Lee on November 12, 1990
- *first web page* on November 13 on a NeXT workstation
- Christmas 1990: Berners-Lee built the first web browser and the first web server
- August 6, 1991: summary of the WWW project posted in a newsgroup in the internet
- April 30, 1993: CERN annouced that the WWW would be free to anyone
- 1993: Browser Mosaic (forerunner of Internet Explorer or Firefox) starts to popularize the WWW

The three core standards of the Web:

• Uniform Resource Locator (URL): specifies how each page of information is given a unique address at which it can be found (e.g., **`http://en.wikipedia.org/wiki/World_Wide_Web`**)

• Hypertext Transfer Protocol (HTTP): specifies how the browser and server send the information to each other

• Hypertext Markup Language (HTML): a webpage description language used to encode the information so that it can be displayed on a variety of devices and under different operating systems.

Later extensions:

- Cascading Style Sheets (CSS): define the appearance of elements of a web page, separating appearance and content
- XML: more general language than HTML, designed to enable a better separation of appearance and content; also applicable to other sorts of information
- ECMAScript (also called JavaScript or JScript): a programming language with commands for the browser, enables embedding of programmes (scripts) into web pages. Thus web pages can be changed dynamically.
- Hypertext Transfer Protocol Secure (HTTPS): Extension of HTTP where the protocol SSL is evoked to encrypt the complete data transfer
- Java applets (small programmes) can be embedded in web pages and run on the computer of the Web user

The *World Wide Web Consortium* (*W3C*) develops and maintains some of these standards (HTML, CSS) in order to enable computers to effectively store and communicate different kinds of information.

Problems with the Web:

- *highly decentralized*, no control of the content

→ there is a lot of false and misleading information, hate campaigns, promotion of sexual exploitation, of terrorism and of other crimes...

- *highly dynamic*: *Web pages change all the time!*

Links point to nowhere when the target page was removed...

→ when you give a Web address in the References section of a scientific paper or in your thesis, you should add the *date* when you visited that page!

Archive of (a part of) the Web:

http://archive.org/web/

→ lost Web references can (in some cases) be reconstructed if the date is known

- *highly chaotic*: no global index or table of content is available; search for a certain content is complicated and time consuming

→ development of specialized search engines,
the most well-known one: *Google* (**http://www.google.de**)

How does a search engine work?

- First component: a web crawler, visiting all accessible web pages worldwide, one after the other, following the hyperlinks

but: when you look for a certain keyword, this process would take much too long!
→

- second component: a large database, containing keywords and web addresses where these keywords were already found

the web crawler is working in the background and does only actualize the database

when you invoke Google, you search in Google's database, not in the Web!

→ not all Web pages can be found, because not all are in the database

Usually, you get many, many, many Web pages containing a given keyword (often millions...)
→
first remedy: make more intelligent queries
e.g., combining several keywords by "and", or looking for phrases instead of keywords (use quotation marks)
– Google provides such facilities under "extended search"

still there are often too many results

$\rightarrow$ priorisation of the found web pages necessary

- third component of the search engine (and best capital of the Google company): a *ranking algorithm* for search results

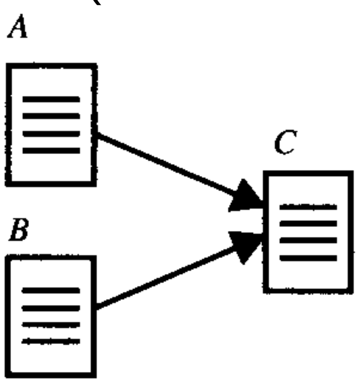
Basic principles of Google ranking of web pages

(Attention: the exact algorithm is changing continuously and is not published)

"Importance" of a web page:
recursively defined, using the hyperlink structure of the Web

The importance of a page is the larger,
the more important pages refer to it!

More precisely:
Let *FLinks*(*A*) be the set of all outgoing links (forward links) of a page *A* and *BLinks*(*A*) the set of all incoming links (backward links) of *A*



$FLinks(A) = \{C\}$
$FLinks(B) = \{C\}$
$BLinks(C) = \{A, B\}$

- *A* has high page rank if the sum of the page ranks of its incoming links is high,
- a page *B* distributes its importance in equal parts to all pages which are referred by it:

$$PageRank(A) = \frac{1}{c} \sum_{B \in BLinks(A)} \frac{PageRank(B)}{|FLinks(B)|} .$$

(c = normalisation factor)

Iterative determination of the page rank:

- initially, an arbitrary mapping of values to all web pages is done (typically, the *constant value* 1 is used),
- *iterate the calculation* using the above formula for all pages, until the values remain stable,
- they *converge* against the Eigenvectors of the adjacency matrix of the graph consisting of the web pages (nodes) and their links (edges). (*Adjacency matrix*: a_{ij} = 1 iff nodes i and j are connected by an edge.)

Additionally, the Google page rank utilizes:

- *proximity* of the given key words to each other (in the text),
- the *anchor texts* of the links: these are the texts which can be clicked upon. A page *A* gets higher importance when the anchor texts of links referring to *A* contain the keywords, too.

The underlying technology of the WWW:
the ***Internet*** (short for "Interconnected Networks")

predecessor (end of the 1960s): ARPANET (U.S. military project)
was later used to connect universities and research labs

Internet today: A worldwide network of computer networks

- Computers in this network communicate using the standardized *TCP/IP protocol* (Transmission Control Protocol / Internet Protocol: Rules governing the communication)
- Transmission of the information in small portions
- For identification, each computer in the net has a unique number, the *IP address*
- to get identifiers which can better be memorized: *Domain Name System* (DNS)
  – system of (textual) names,
  association between names and IP addresses
- hierarchy: Domains, subdomains, sub-subdomains..., e.g.,
  `www.uni-forst.gwdg.de`
  (from right to left!)

- *Top-level domains*: Country abbreviations and some others ("generics"): .de, .fr, .eu, .com, .edu, .gov ...
- Lowest level: host name of a single computer (here: www, Web server of the forestry faculty)
- domain name corresponds to IP address
- transformation of domain names into IP addresses and vice versa: Task of special computers, so-called *nameservers*
- this transformation takes place any time when you click on a hyperlink on a web page!
- each nameserver is responsible for a certain part of the hierarchical name space