Types of Databases

- Relational databases contain stuctured data
  - tables, columns, fixed datatype for each column
- **Text databases** are available for storing non-structured data
  - typically text databases store collections of non-connected documents
  - search of data is speeded up with search indexes
    - like indexes on books - cover the whole text
    - index item = (keyword, document,.., *{weight, position}*)
    - there may be many index items pointing to the same document
  - documents are usually fairly static -> index items related to a document do not usually change.

Types of Databases

- Relational databases may also have indexes
  - an index is made over a single column or a collection of columns
    - index item = (value, internal_rowID)
    - column values are typically dynamic -> index items deleted and added (often).
    - Within one index there is only one index item pointing to a certain row.

- A typical query in a relational databases
  - find rows that have specific values in certain colums
  - find books where author='Smith'
- A typical query in text databases
  - find documents that contain specific keywords
  - find documents that contain keyword Smith and author
  - The relational database query is semantically exact the name Smith has to be in column author - you get what you ask for

Types of Databases

- Text database queries are vague
  - you might miss the documents and get documents you do not want
  - 'Story of Smith, Author: Jones' (retrieved - not wanted)
  - 'Story of Jones, written by Smith' (missed)

Databases and Web

- Information available in Web may currently be considered mostly as a large distributed text database
  - limited indexes (search engines)
  - links connect documents

Databases and Web

- Web material may be viewed with a browser.
- Most of the 'text' in HTML-format, but also other formats are available (e.g. pdf)
  - HTML -language defines the technical structure for the document (titles, paragraps, tables, lists, etc.)
    - abstract layout - browser determines the actual layout
- Documents stored using standard file management systems.
- Local Web servers are configured to find the documents based on universal resource identifiers (URI)

Databases and Web

- HTML is an application of SGML-standard (general markup language)
- SGML- defines how to mark elements
  - Elements are represented between begin and end marks
    - <element_type> element </element_type>
- in HTML:
- <h1>Level 1 header</h1>
  <table>
  <tr><td>table_element</td> …</tr>
  </table>

Databases and Web

- HTML-document may currently contain text, images, audio, video, executables
- HTML-document is
  - readable anywhere - only browser is needed
  - storable anywhere
    - Internet/intranet server needed to provide access

Databases and Web

- The problem with HTML-pages is to locate the information needed
- We are not able to use SQL-style queries

  select information from document_base

  where topic ='needed_topic'
  - HTML does not mark topics but technical structures

Databases and Web

- The queries on HTML-pages correspond to SQL-query

  select URI from document_base

  where document_content like '%keyword%';
- We usually retrieve huge number of documents most out of our interest
- Various search engines sort the result differently
  - no order,
  - newest ones first,
  - most referred ones first

Databases and Web

- XML is a meta-level standard for topic based markup. It is a 'simplification' of SGML
  - elements marked like in SGML
- We may define XML-based presentation languages for document types.
- Construction plan XML (CPML) might have elements like <general_plan>, <electricity_plan>,<building>,<room>, <ceiling>, <wall>, <window>, etc.

Databases and Web

- XML does not define how to present the document (not even abstractly as HTML)
- The presentation may be defined with a separate definition (style sheet). There may be many presentations for the same document type.
- XML-capable browser should be able to show the document according to the style sheet.

Databases and Web

- XML makes it possible to use topics in searches

```
select URL from CPML_document_base
where 'tile' in constructionplan.roof.materal
```

- Searches like the above assume new types of search engines and indexes

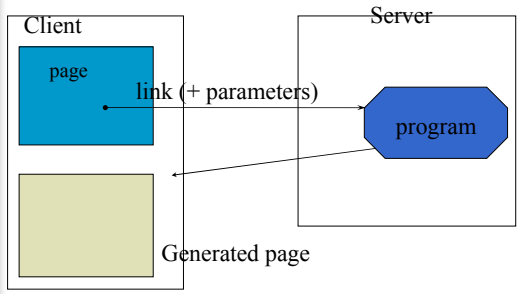
Databases and Web

- WWW-pages
  - hypertext in HTML-language
  - text, images, hypelinks
- **Static pages**
  - written once, changes must be made manually
  - keeping pages up to date is a problem
  - maintaining pages is hard if the same information is provided in many pages
  - Its hard to find the information if the user does not share 'stuctural model' of the author. Structuring the information in many ways causes maintenance problems

WWW ja tietokannat

- Use generated pages for easy maintenance:
  - Pages are generated with programs using data stored in the (relational) database
    - according to a schedule
      - *always when data is changed*
      - *once a week, ...*
    - When requested (=when somebody asks for the data)
      - *When pages are generated for each request, users actually activate a program in the server*
      - *CGI (common gateway interface), Java Servlet, ASP activation techniques*

Generation of a web-page



Sever side programs

- The program to be activated may receive parameters
  - as written in URI
    - book.search?isbn=0-123456-67-1&lang=FIN
  - as entered in HTML-form
- The program to be activated may do anything (register an order, generate the answer for a request, charge 1M euros from users bank account,...)

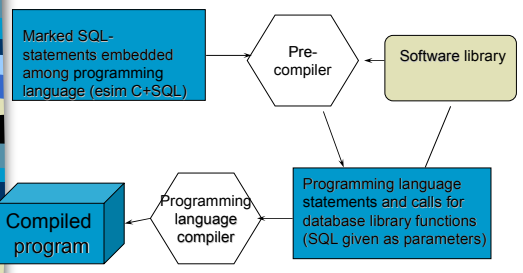
Sever side programs

- Server side programs may be written by any programming language
  - Perl is common is CGI technique is used
  - Java is very popular too (as servlets)
  - There are techniques that combine the HTML-text and program code in the same file
    - JSP (Java Server Pages)
    - ASP (Active server pages, Microsoft)
    - PHP

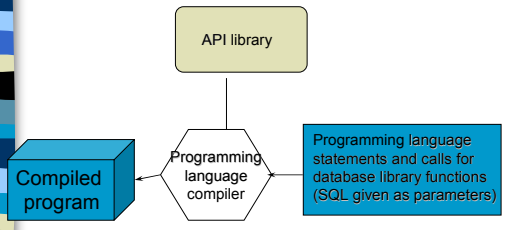
Writing programs to process a database

- Databases may be used in programs written in various programming languages.
- Two approaches
  - embedded SQL - pre-compiler based
  - Use of a connection library - API, application programming Interface
    - native dbms supplier libraries
    - independent interfaces ODBC (C style), JDBC (Java)

Embedded SQL

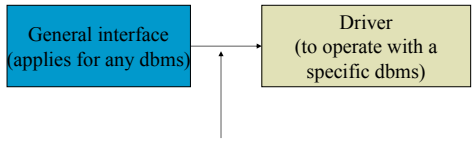


Connection library approach



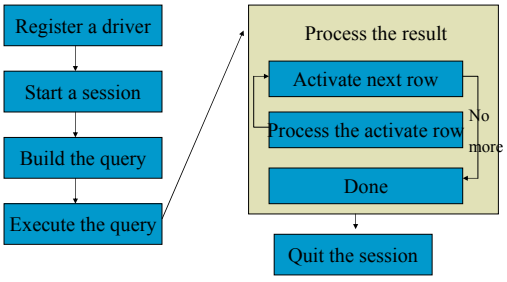
Connection library approach

- A general API must be used with a database magement system specific driver



Program defines which driver to use

Query processing in a program




```
// Example program -  uses solid db
// path for driver must be included in CLASSPATH:

import java.sql.*;
import java.io.*;
public class Esim1  {
  public static void main(String args[]) throws Exception {
  String url = "jdbc:oracle:thin@db.cs.helsinki.fi:1521:test";
  // This url identifies a database in  db.cs.helsinki.fi
  Connection con;
  String query =
            "select code, course.name AS KNI, credits, "+
                  "teacher.name AS OPNI " +
                        "from course, teacher " +
                        "where course.lecturer= teacher.teacherid " +
                        "order by OPNI, KNI";

  Statement stmt;
```

```
try {
    Class.forName(:"oracle.jdbc.OracleDriver");
  } catch(java.lang.ClassNotFoundException e) {
     // Driver not found.
              System.err.print("ClassNotFoundException: ");
              System.err.println(e.getMessage());
}
```

```
try {
          // start a session
     con = DriverManager.getConnection(url, "info", "expert");
        // make execution environment
     stmt = con.createStatement();
        // Variable ope is used for keeping book on previous
        // teacher
     String ope =" ";
        // print header
     System.out.println("Teaching tasks:");
     System.out.println();
       // execute the query
     ResultSet rs = stmt.executeQuery(query);
```

```
// process the result
     while (rs.next()) {
          // next row, getString gets information from active row
       String opn = rs.getString("OPNI");
          // print empty row if teacher changes

       if (ope.equals(opn)) {
          System.out.println("     " +
            rs.getString("KNI")+ " (" + rs.getInt("KOODI")+")");
       }
       else {
         System.out.println();
         ope= opn;
         System.out.println(ope);
         System.out.println("     " +
            rs.getString("KNI")+" ("+rs.getInt("KOODI")+")");
       }
    }
```