Introduction to Database Systems

Fundamental Concepts

Werner Nutt
Characteristics of the DB Approach

• *Insulation* of application programs and data from each other

• Use of a *catalogue* to store the *schema*

• Support of *multiple user views*

⇒ *How can one realise these principles?*
A DBMS Presents Programmers and Users with a Simplified Environment

Users/Programmers

Database System

Queries / Application Programs

DBMS Software

Software to Process Queries / Programs

Software to Access Stored Data

"Catalogue", "Data dictionary"

Stored Database Definition (Metadata)

Stored Database
Data Model
• A set of concepts that can be used to describe the structure of a database: the data types, relationships, constraints, semantics and operational behaviour
• Hides details of data storage

Schema
• A formal definition that fixes all the relevant features of those parts of the real world that are of interest to the users of the database
• The schema of a db is held in the data dictionary

Example:
- Student(studno, name, address)
- Course(courseno, lecturer)
- Student(123, Egger, Bozen)
- Course(CS321, Nutt)
Relational model is good for:

- Large amounts of data and simple operations
- Limited navigation, touching only small numbers of relations/tables

**Difficult** applications for relational model:

- VLSI design (CAD in general)
- CASE
- Graphical data
- Bill of materials, transitive closure
Object Data Models

Where number of “relations” is large, relationships are complex
- Object Data Model
- “Knowledge Data Model” (= Objects + Deductive Rules)

Object Data Model (Principles)

1. Complex Objects – Nested Structure (pointers or references)
2. Encapsulation, set of methods/access functions
3. Object Identity
4. Inheritance – Defining new classes like old classes

Object model: usually, objects are found via explicit navigation. Also query language in some systems.
<addresses>
  <person>
    <name>Donald Duck</name>
    <tel>0471- 82 81 45</tel>
    <tel>332- 82 88 283</tel>
    <email>donald@inf.unibz.it</email>
  </person>
  <person>
    <name>Mickey Mouse</name>
    <tel>0473 – 42 61 14</tel>
  </person>
</addresses>
XML Terminology

The segment of an XML document between an opening and a corresponding closing tag is called an element.

```
<person>
  { <name> Donald Duck </name>
   <tel> 0471- 82 81 45 </tel>
   <tel> 332- 82 88 283 </tel>
   <email> donald@inf.unibz.it </email>
  }
</person>
```

- element
- element, a sub-element of
- not an element
XML Documents are Trees

- XML documents are abstractly modeled as trees, as reflected by their nesting.

- Donald Duck
  - Name: Donald Duck
  - Tel 1: 0471-82 81 45
  - Tel 2: 332-82 88 283
  - Email: donald@inf.unibz.it
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
  <cd country="UK">
    <title>Dark Side of the Moon</title>
    <artist>Pink Floyd</artist>
    <price>10.90</price>
  </cd>
  <cd country="UK">
    <title>Space Oddity</title>
    <artist>David Bowie</artist>
    <price>9.90</price>
  </cd>
  <cd country="USA">
    <title>Aretha: Lady Soul</title>
    <artist>Aretha Franklin</artist>
    <price>9.90</price>
  </cd>
</catalog>
Document Type Definition (DTD)

DTDs specify the format of documents

```xml
<!DOCTYPE catalog [ 
  <!ELEMENT catalog (cd*)>  
  <!ELEMENT cd (title, artist, price)>  
  <!ELEMENT title (#PCDATA)>  
  <!ELEMENT artist (#PCDATA)>  
  <!ELEMENT price (#PCDATA)>  
  <!ATTLIST person       
    country    CDATA ID   #IMPLIED>  
]>  
```

- an arbitrary number of CDs
- a title, followed by an artist, followed by a price
- title, artist, and price contain parsable character data
- A person element can have an (optional) country attribute
The XML document as a tree

catalog.xml

catalog
country
country

UK
UK
USA

country
country
country
country

cd
cd
cd
cd
cd
cd

title
artist
price
title
artist
price
title
artist
price
title
artist
price

dark Side of the Moon
Pink Floyd
Dark Side of the Moon
Pink Floyd
Dark Side of the Moon
Pink Floyd

Space Oddity
UK
Space Oddity
UK
Space Oddity
UK

Aretha: Lady Soul
Aretha Franklin
Aretha: Lady Soul
Aretha Franklin
Aretha: Lady Soul
Aretha Franklin

10.90
9.90
9.90

David Bowie

9.90
XPath: an XML Query Language

• XPath expressions are evaluated over documents

• XPath operates on the *abstract tree document* structure

• Documents are trees with several *types of nodes*, such as
  – element nodes
  – attribute nodes
  – text nodes
XPath: Path Expressions are Main Element of Syntax

- `/` at the beginning of an XPath expression represents the root of the document

- `/` between element names represents a parent-child relationship

- `//` represents an ancestor-descendant relationship

- `@` marks an attribute

- `[condition]` specifies a condition
Getting the root element of the document
Finding descendant nodes
catalog.xml

/catalog/cd[price<10]

country

cd

title, artist, price

UK

Dark Side of the Moon
Pink Floyd
10.90

country

country

UK

US

Space Oddity
David Bowie
9.90

Aretha: Lady Soul
Aretha Franklin
9.90

Condition on elements
// represents any directed path in the document
catalog.xml

/catalog/cd/*

country

cd

title
artist
price

UK

Dark Side of the Moon
Pink Floyd
10.90

UK

Space Oddity
David Bowie
9.90

USA

Aretha: Lady Soul
Aretha Franklin
9.90

* represents any element name in the document
What will the following expressions return?

```xml
catalog.xml
  catalog
    cd
growth
    title
      artist
        price
          country
            title
              artist
                  price
                    country
                        title
                            artist
                                price
                                    country
                                        USA
```
How would you write:

The price of the CDs whose artist is David Bowie?
Data Models

60's
- Hierarchical
- Network

70's
- Relational

80's
- Object Models

90's
- Knowledge Bases, Rules

00's
- Semistructured Data, XML
- Semantic Web, RDF

Choice for most applications today
Sharing—Multiple views of data
Three Levels of Abstraction

ANSI/SPARC architecture for DBMSs (1978):

- Many *external views*
- One *conceptual* (= logical) *schema*
- One *physical* (= internal) *schema*
  - Views describe how users see the data
  - Conceptual schema defines logical structure
  - Physical schema describes the files and indexes used
Data Independence

• **Logical** data independence
  – change the logical schema without having to change the external schemas

• **Physical** data independence
  – change the internal schema without having to change the logical schema

*Change the mapping, not the schema!*
Database Languages

• **Data Definition Language (DDL)**
  – Commands for setting up the schema of a database
  – The process of designing a schema can be complex, may use a design methodology and/or tool

• **Data Manipulation Language (DML)**
  – Commands to manipulate data in database:
    
    ```
    RETRIEVE, INSERT, DELETE, MODIFY
    ```
  – Also called “query language”
Host Languages

C, C++, Fortran, Lisp, Java, Perl, Python, ...

**Host language** is completely general (Turing complete) but gives no support for data manipulation

**Query language**—less general, “non procedural” and optimizable
Building an Application with a DBMS

• Requirements gathering (natural language, pictures)

• Requirements modeling (conceptual data model, ER)
  – Decide what entities should be part of the application and how they should be related

• Schema design and implementation
  – Decide on a set of tables, attributes
  – Create the tables in the database system
  – Populate database (insert records/tuples)

• Write application programs using the DBMS
  – … a lot easier now that the data management is taken care of
### Schema Design and Implementation

- **Tables:**

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-45-6789</td>
<td>Charles</td>
<td>undergrad</td>
</tr>
<tr>
<td>234-56-7890</td>
<td>Dan</td>
<td>grad</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>CID</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-45-6789</td>
<td>CSE444</td>
</tr>
<tr>
<td>123-45-6789</td>
<td>CSE444</td>
</tr>
<tr>
<td>234-56-7890</td>
<td>CSE142</td>
</tr>
</tbody>
</table>

- **Course:**

<table>
<thead>
<tr>
<th>CID</th>
<th>Name</th>
<th>Quarter</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSE444</td>
<td>Databases</td>
<td>fall</td>
</tr>
<tr>
<td>CSE541</td>
<td>Operating systems</td>
<td>winter</td>
</tr>
</tbody>
</table>

- The **logical schema** separates the logical view from the physical view of the data.
Querying a Database

- “Find all courses that Mary takes”
- S(structured) Q(uestion) L(anguage)

```sql
select c.name
from Student s, Takes t, Course c
where s.name = 'Mary' and
  s.ssn  = t.ssn and
  t.cid   = c.cid
```

- The query processor figures out how to answer the query efficiently
Query Optimization

**Goal:** Declarative SQL query $\Rightarrow$ Query execution plan

```sql
select c.name
from   Student s, Takes t,
       Course c
where  s.name = 'Mary' and
       s.ssn = t.ssn and
       t.cid = c.cid
```

**Plan:** Tree of relational algebra operators, choice of algorithm for each operator

**Ideally:** Find best plan  
**Practically:** Avoid worst plans!
Traditional and Novel Data Management Issues

• Traditional Data Management:
  – Relational data for enterprise applications
  – Storage
  – Query processing/optimization
  – Transaction processing

• Novel Data Management:
  – Integration of data from multiple databases, warehousing
  – Data management for decision support, data mining
  – Managing documents, audio, and visual data
  – Exchange of data on the web: XML
  – Data Streams
  – Querying data on the Web: RDF