Introduction to Database Systems

Fundamental Concepts

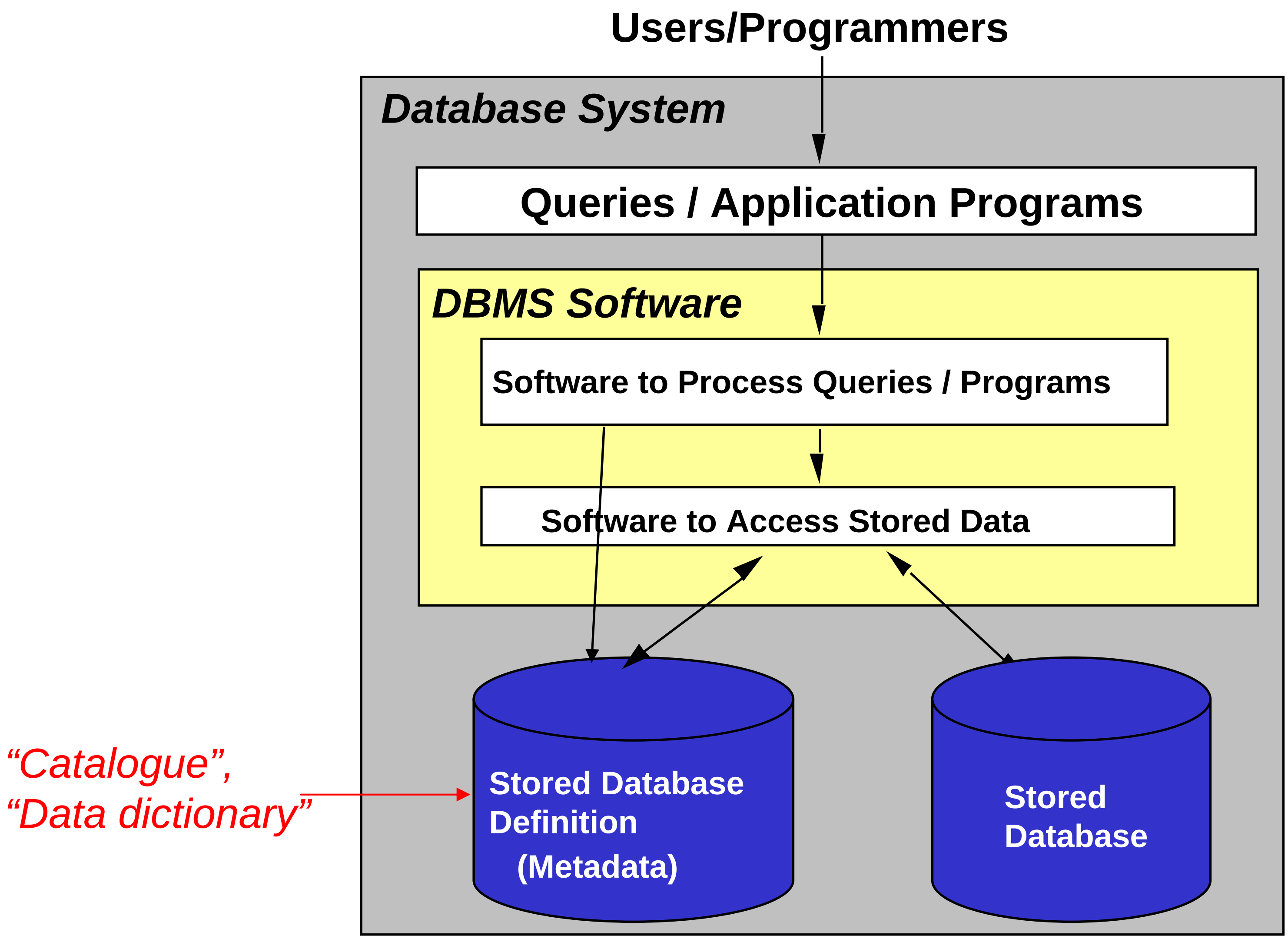
Werner Nutt

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



Data Model, Schema and Instance

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*

Schema
(in relational data model)

```
Student(studno,name,address)
Course(courseno,lecturer)
```

Instance

```
Student(123,Egger,Bozen)
Course(CS321,Nutt)
```

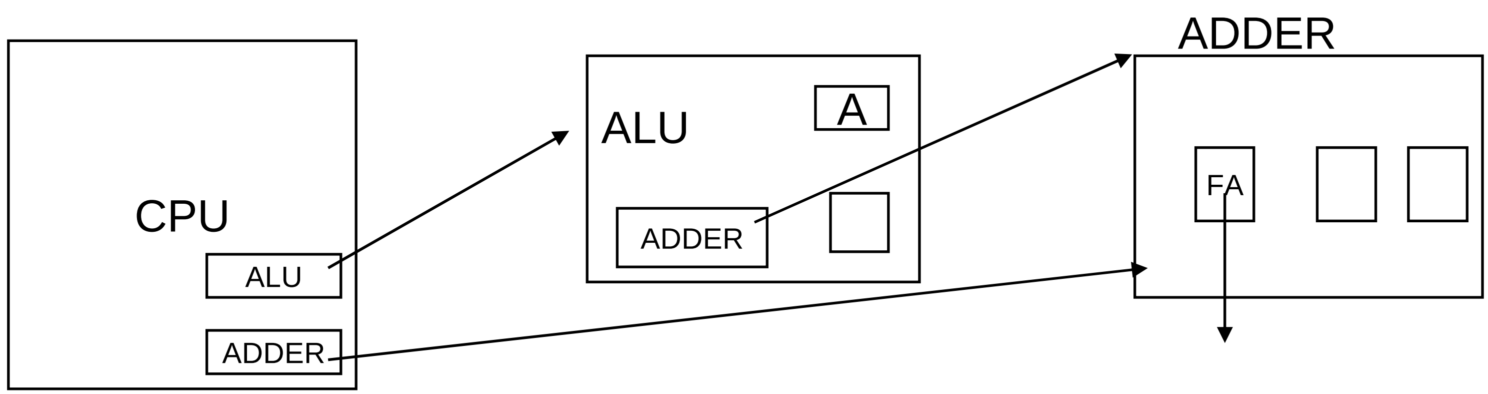
Other Data Models

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.

XML Documents

```
<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

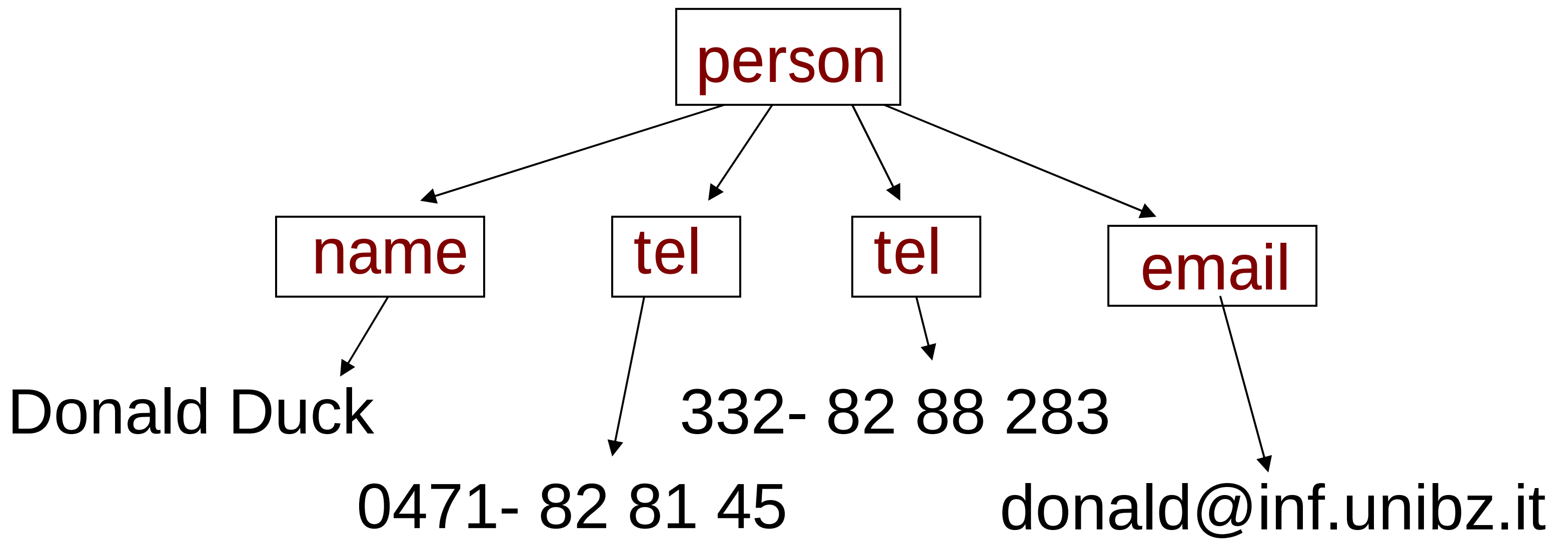
element

element,
a sub-element of

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

not an element

XML Documents are Trees



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

```xml
<?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>
```

An XML document

Document Type Definition (DTD)

DTDs specify the format of documents

```
<!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

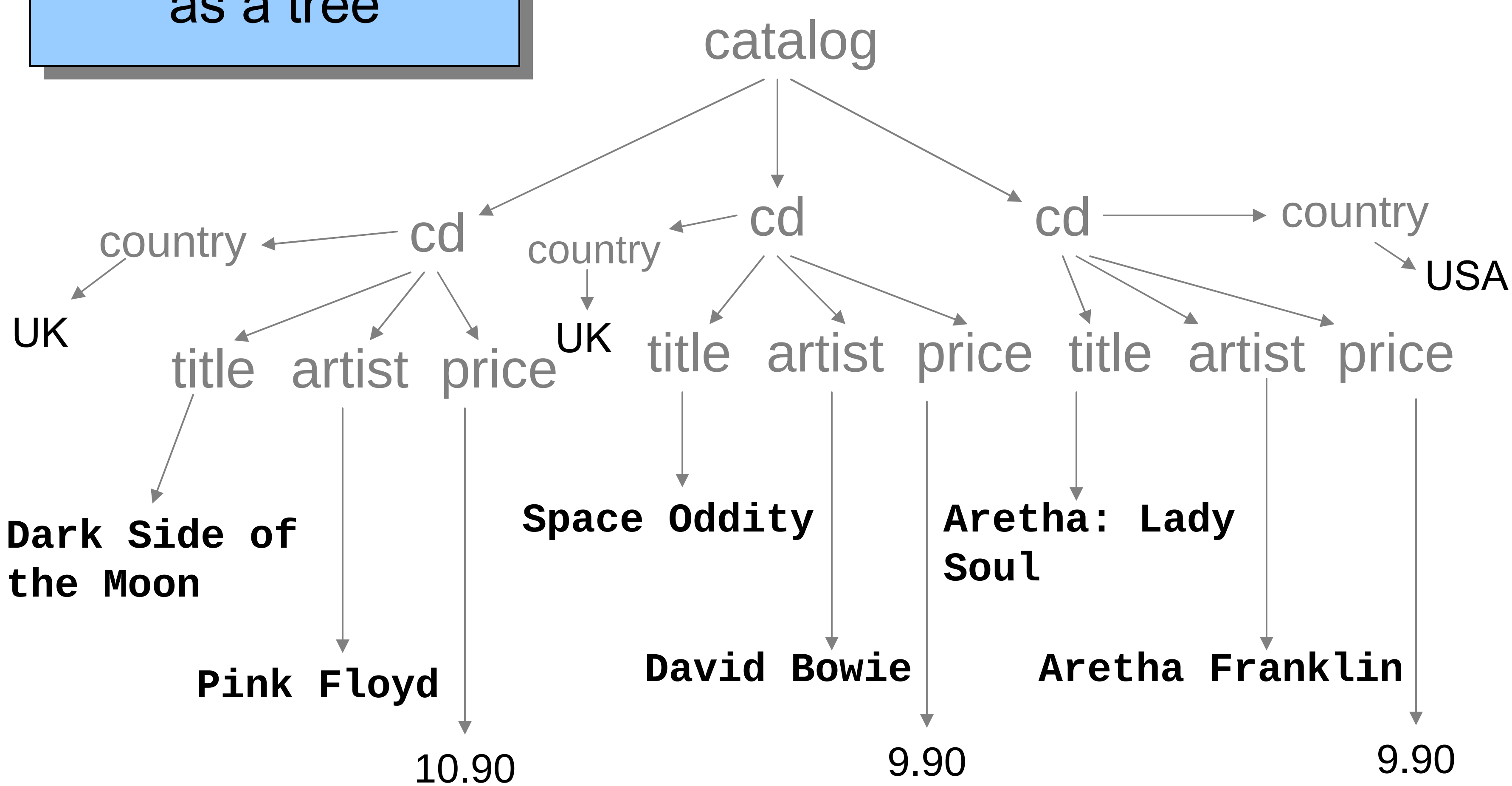
a title, followed by an artist, followed by a price

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



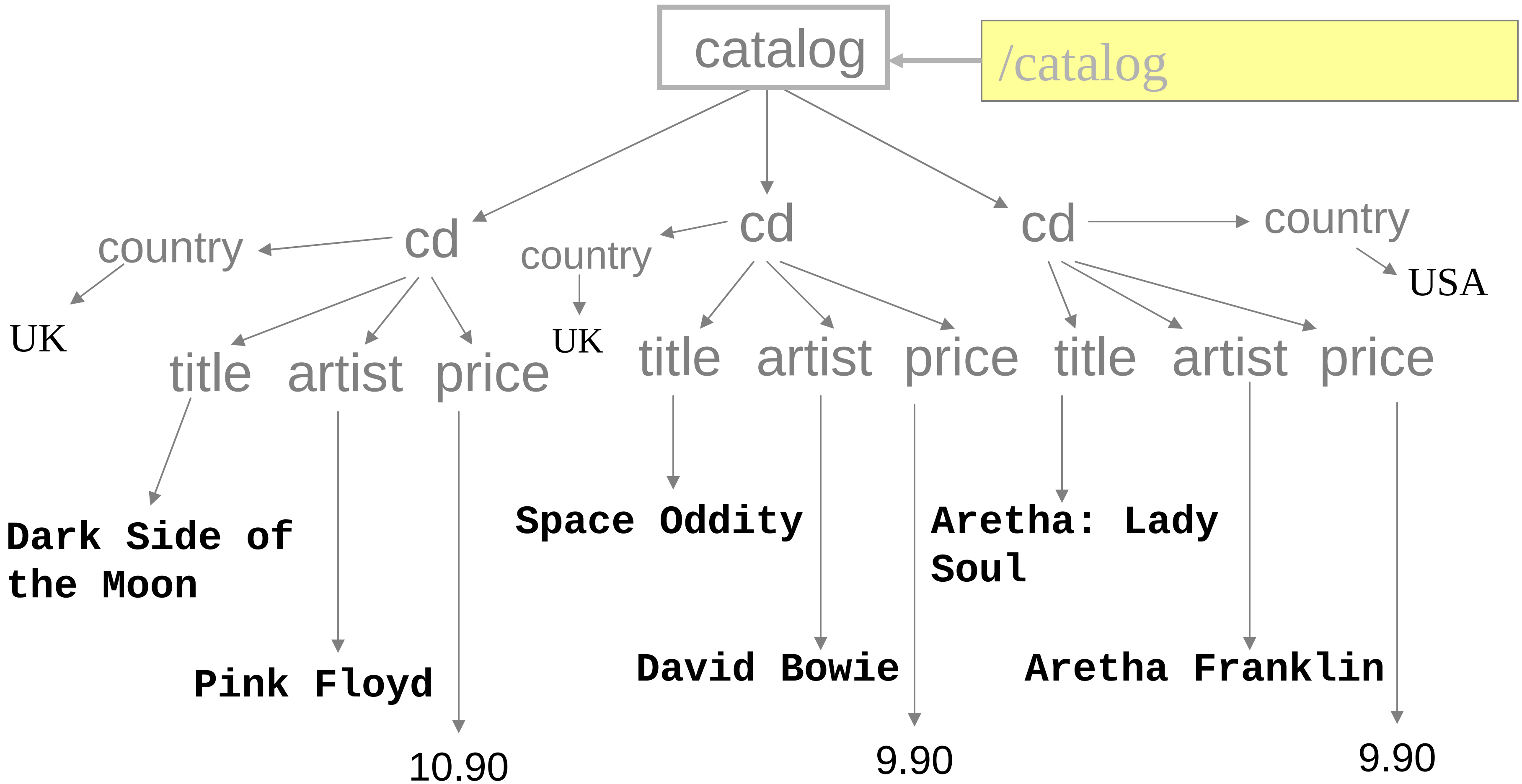
XPath: an XML Query Language

- 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

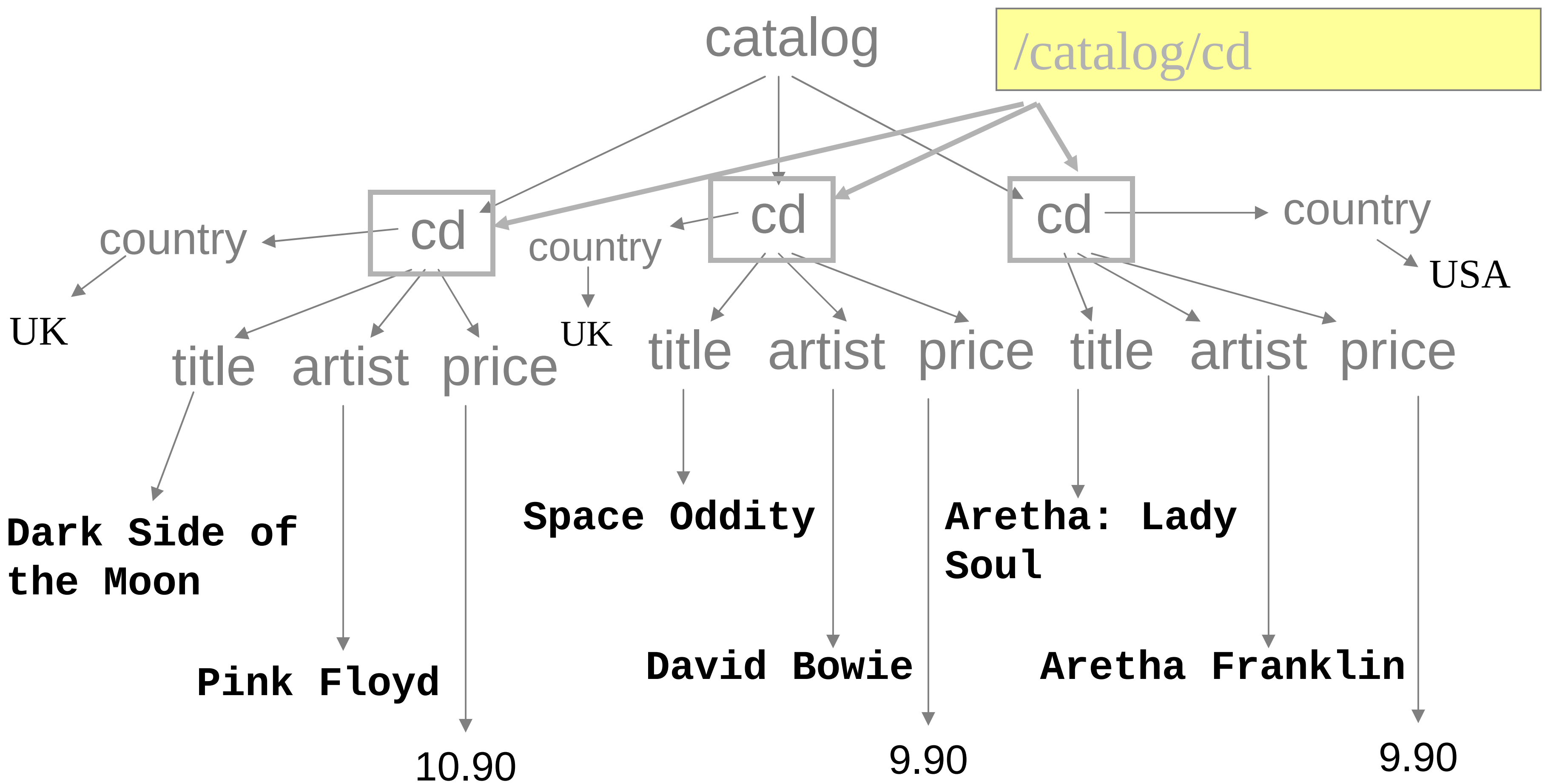
- / at the beginning of an XPath expression represents the root of the document
- / between element names represents a parent-child relationship
- // represents an ancestor-descendent relationship
- @ marks an attribute
- [condition] specifies a condition

catalog.xml



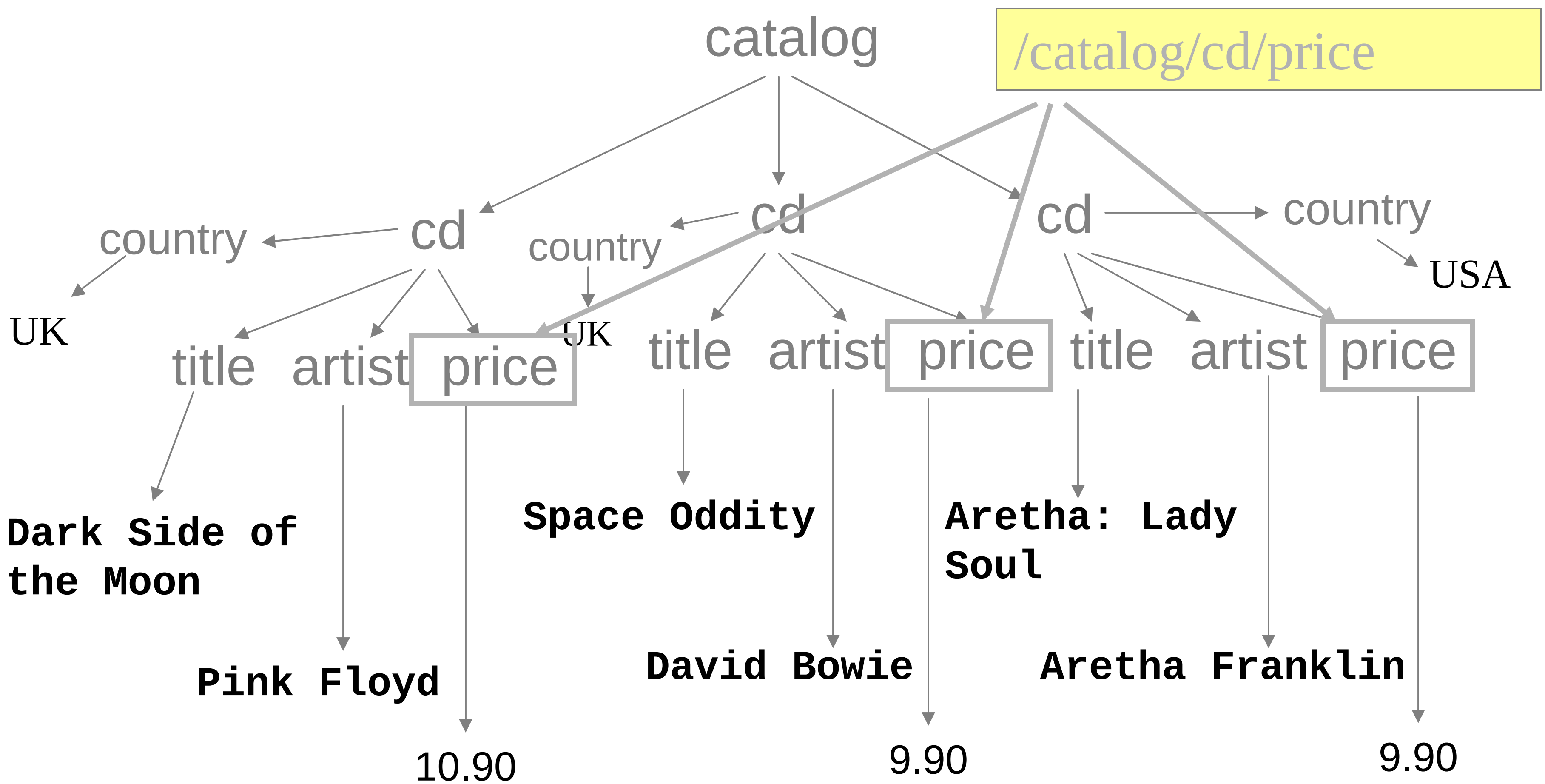
Getting the root element of the document

catalog.xml



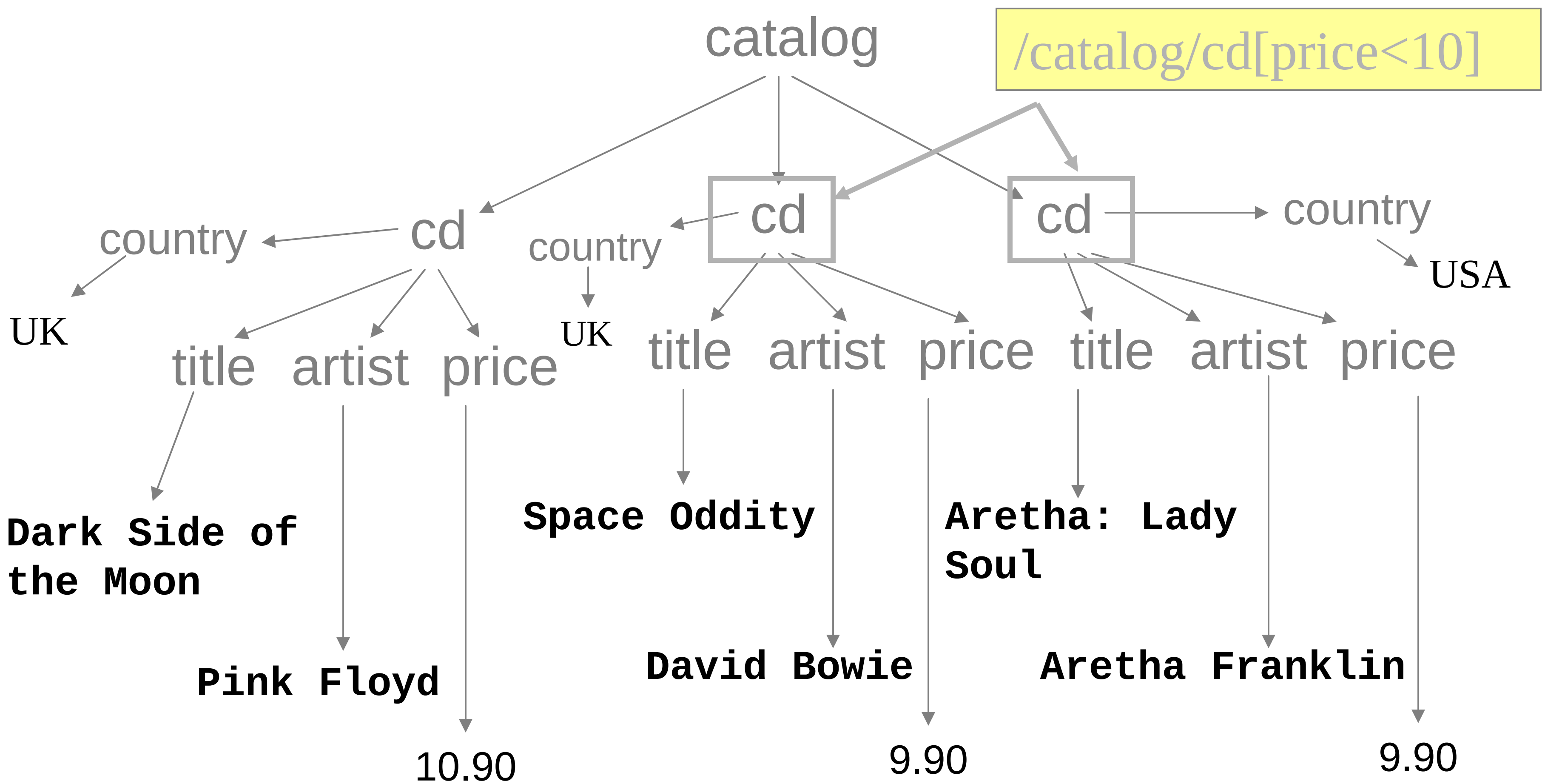
Finding child nodes

catalog.xml

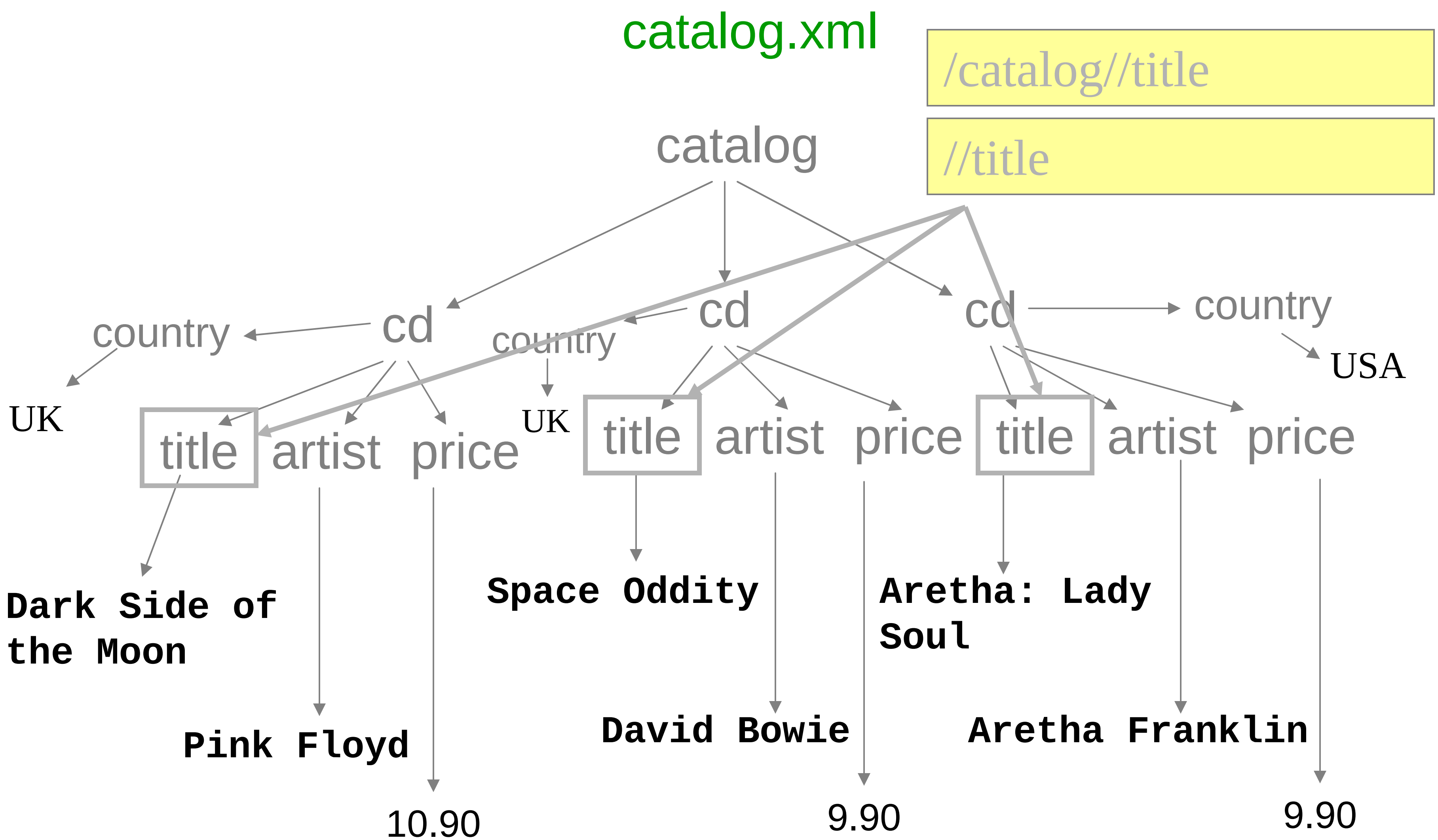


Finding descendant nodes

catalog.xml

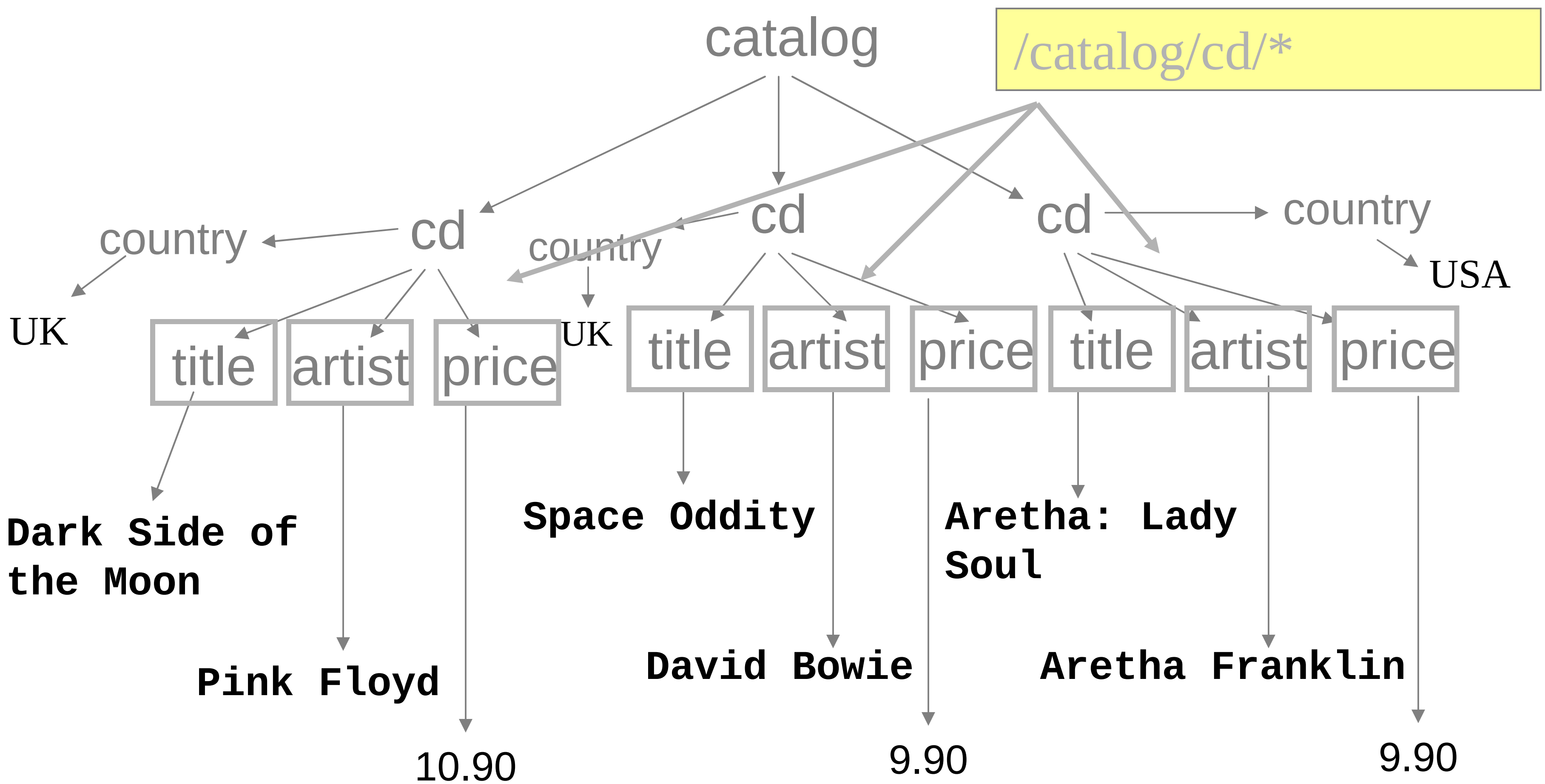


Condition on elements



// represents any directed path in the document

catalog.xml

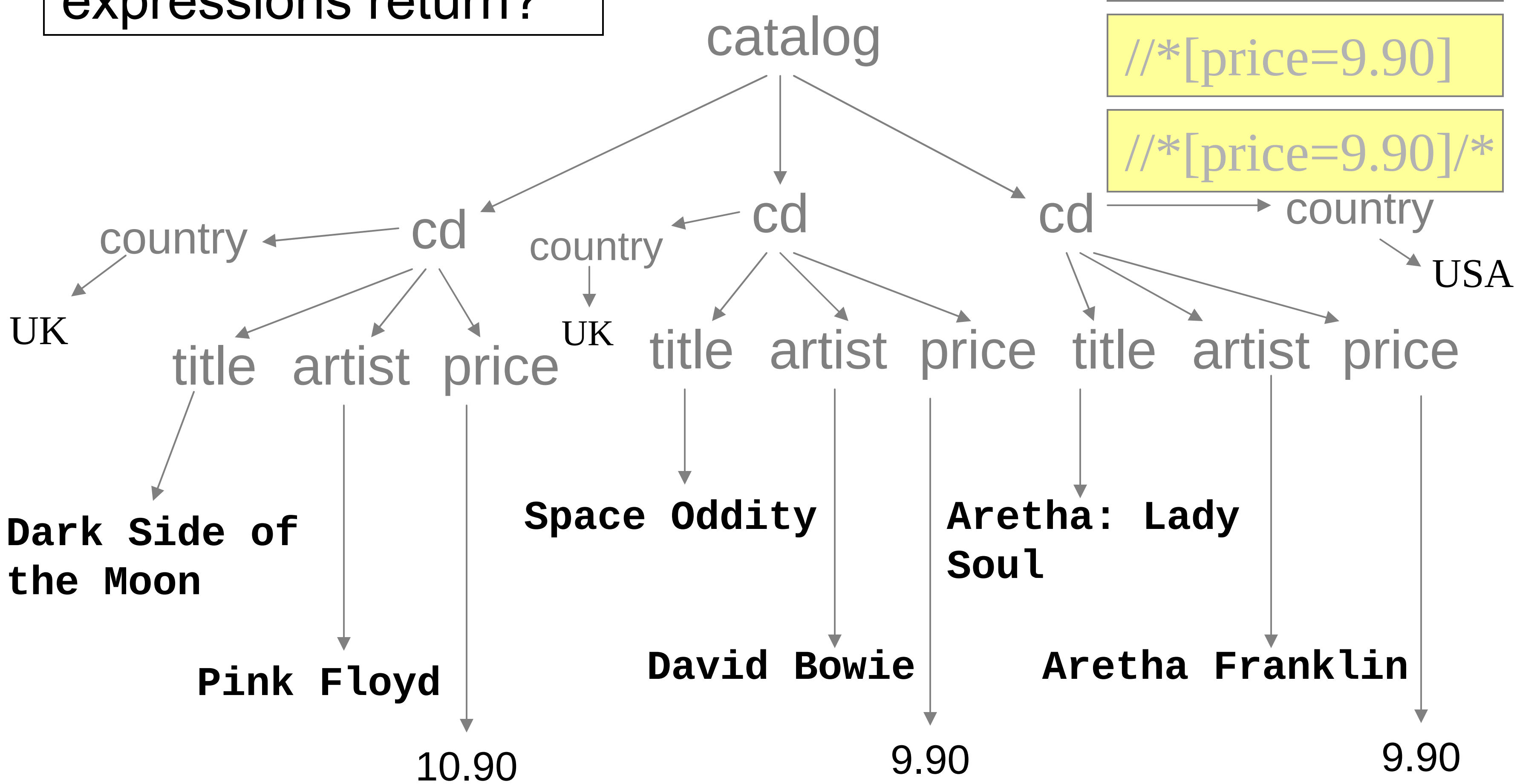


* represents any element name in the document

What will the following expressions return?

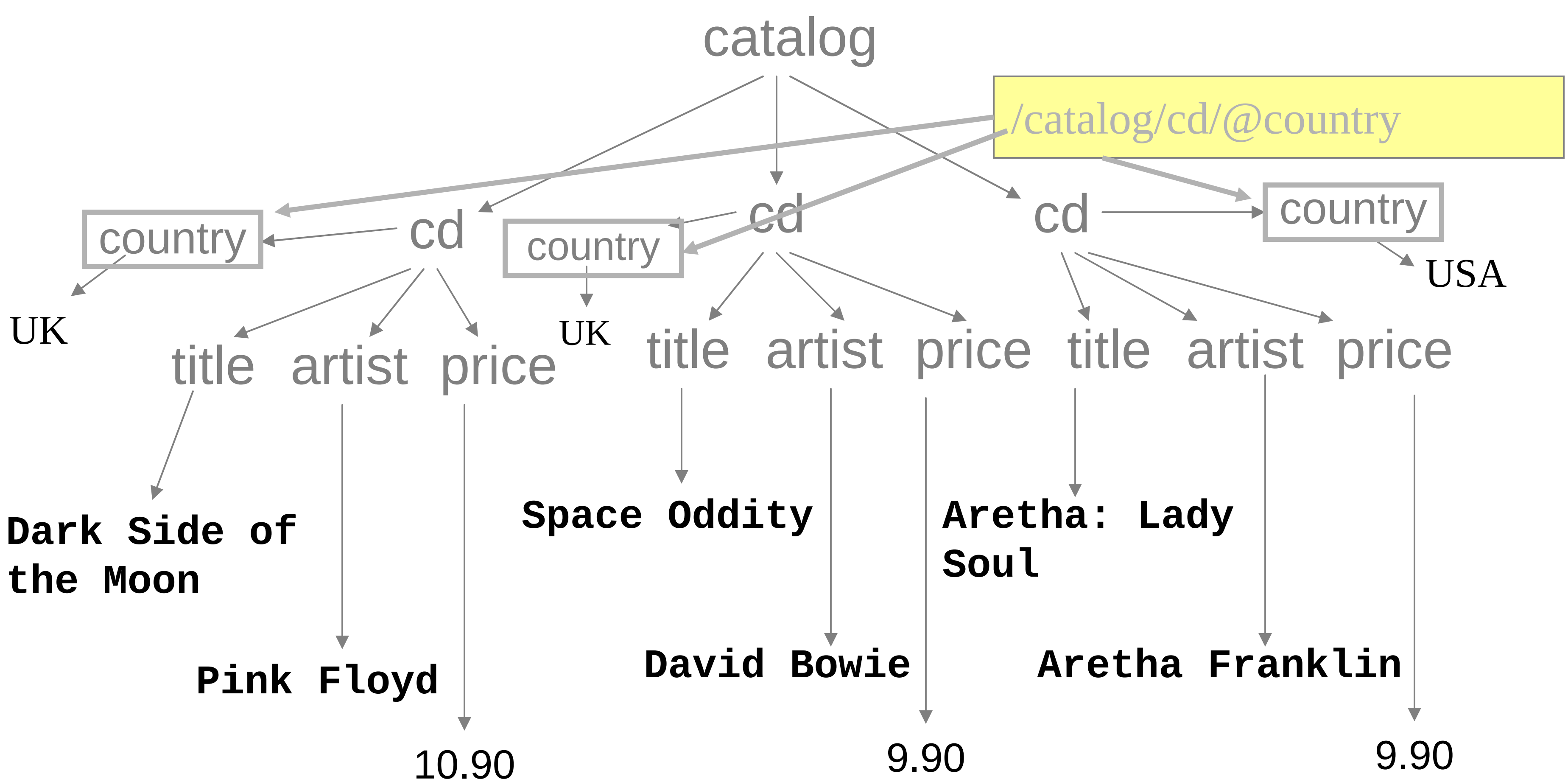
catalog.xml

- /*/*
- //*
- //*[price=9.90]
- //*[price=9.90]/*



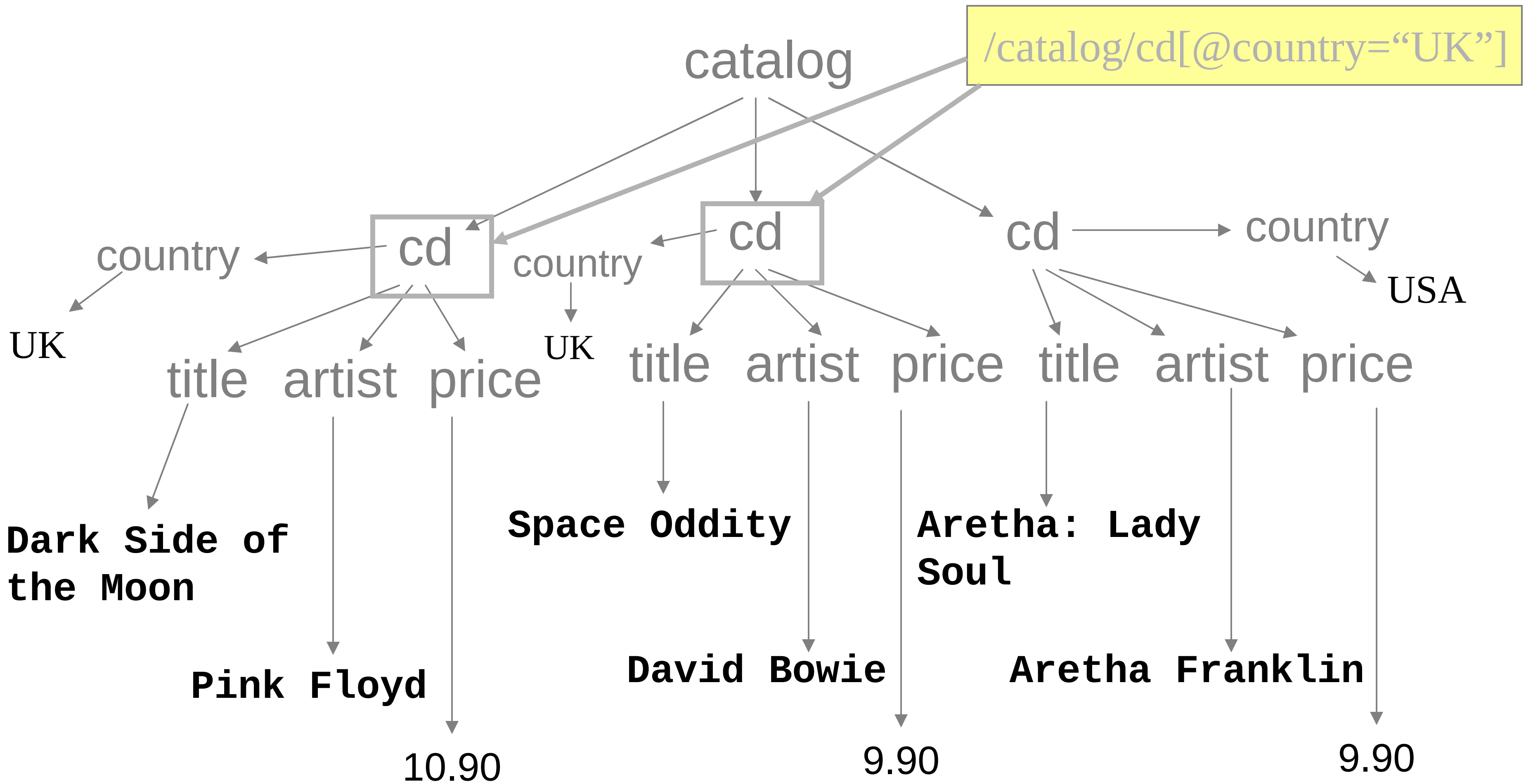
* represents any element name in the document

catalog.xml



@ marks attributes

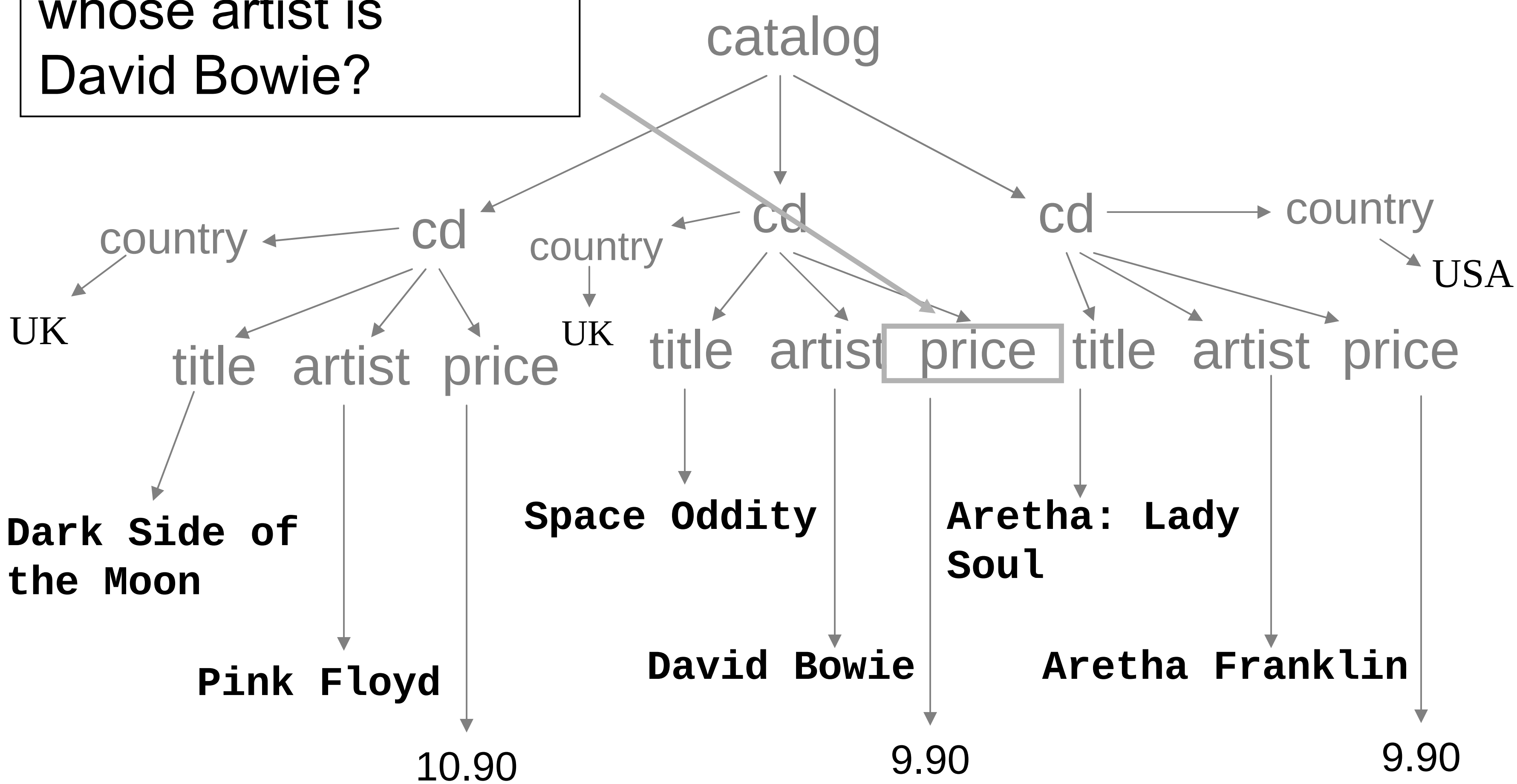
catalog.xml



@ marks attributes

How would you write:
The price of the cds
whose artist is
David Bowie?

catalog.xml



Data Models

60's

Hierarchical

Network

70's

80's

Relational

Choice for most applications today

90's

Object Models

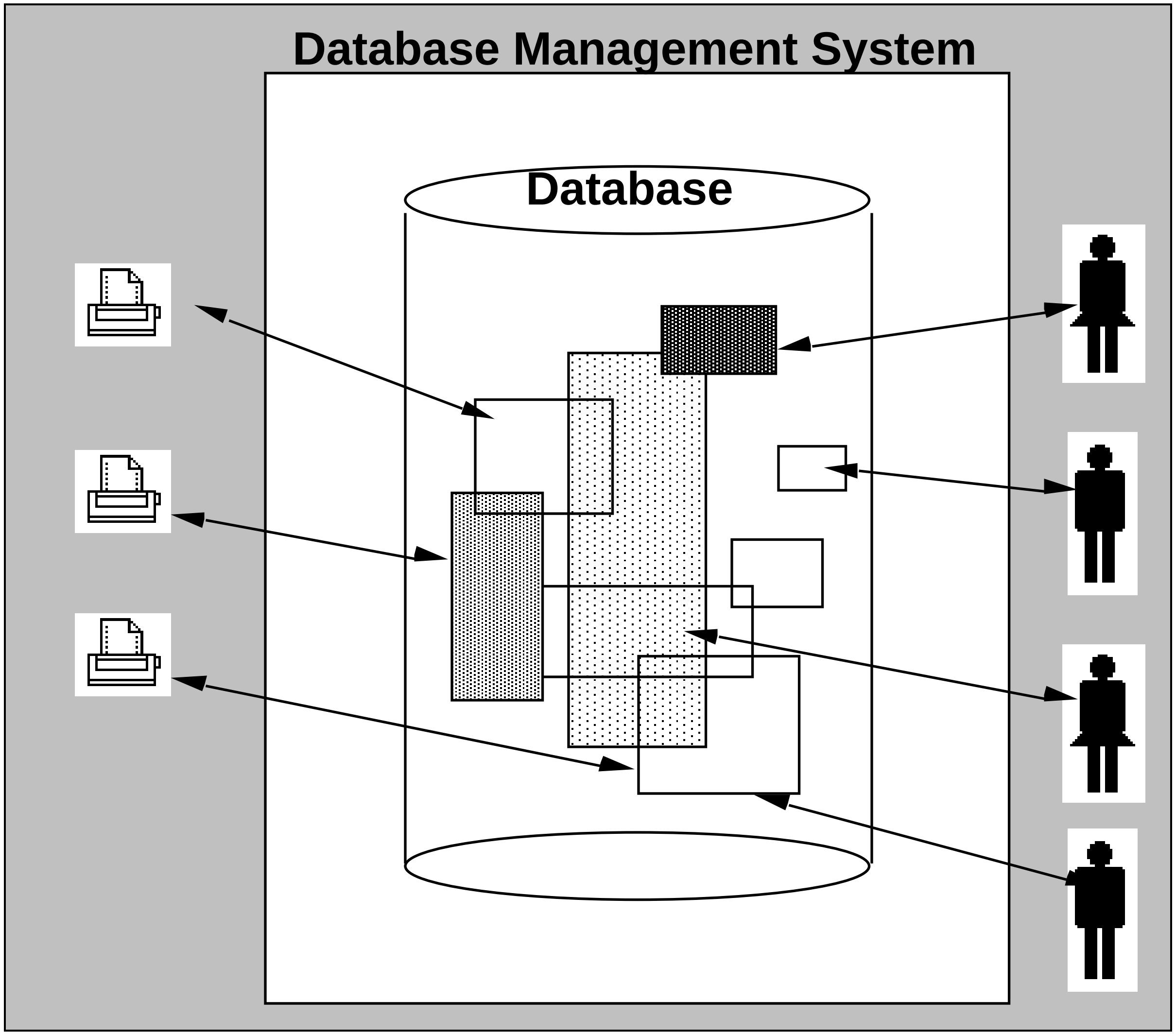
Knowledge Bases, Rules

00's

Semistructured Data, XML

Semantic Web, RDF

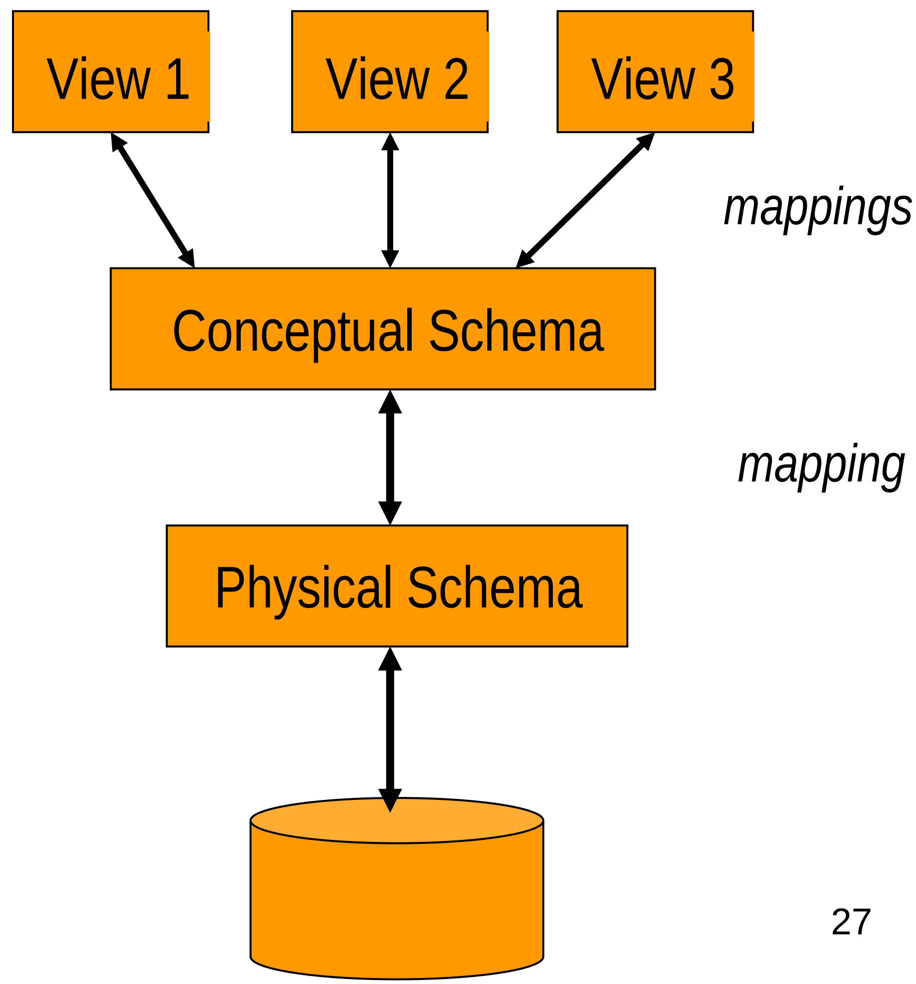
Sharing—Multiple *views* of data



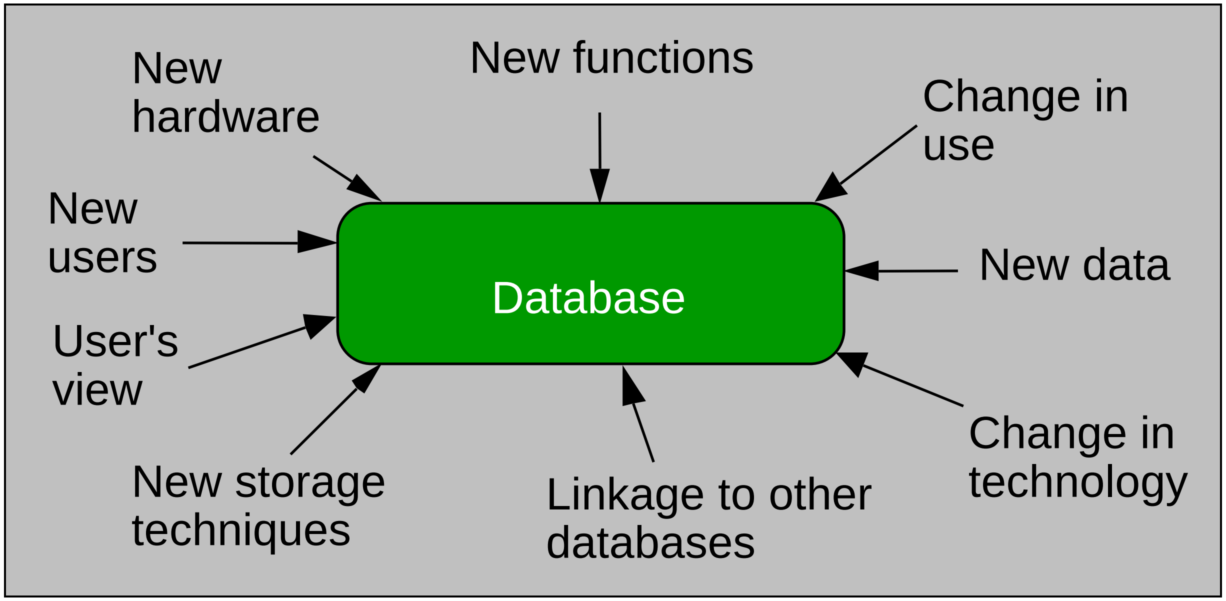
Three Levels of Abstraction

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

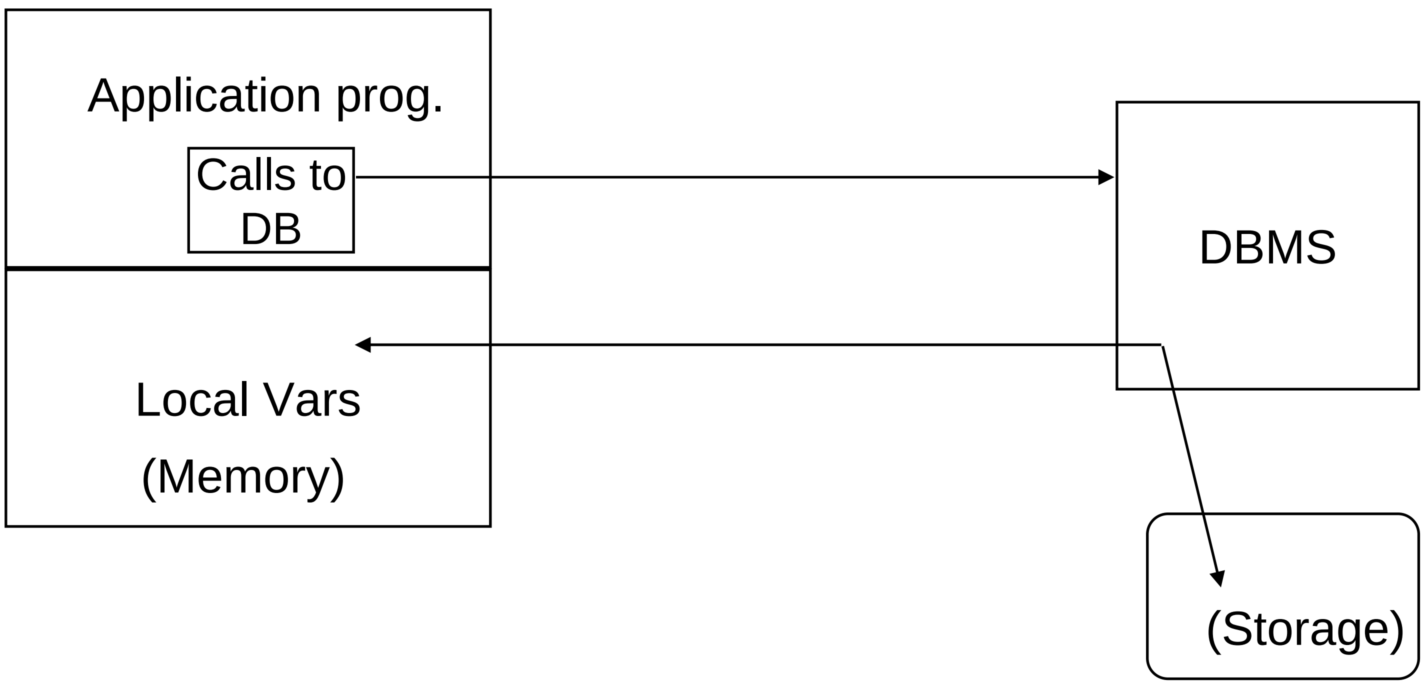
- Data Manipulation Language (DML)
  - Commands to manipulate data in database:

    RETRIEVE, INSERT, DELETE, MODIFY

  - Also called "query language"

Host Languages

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



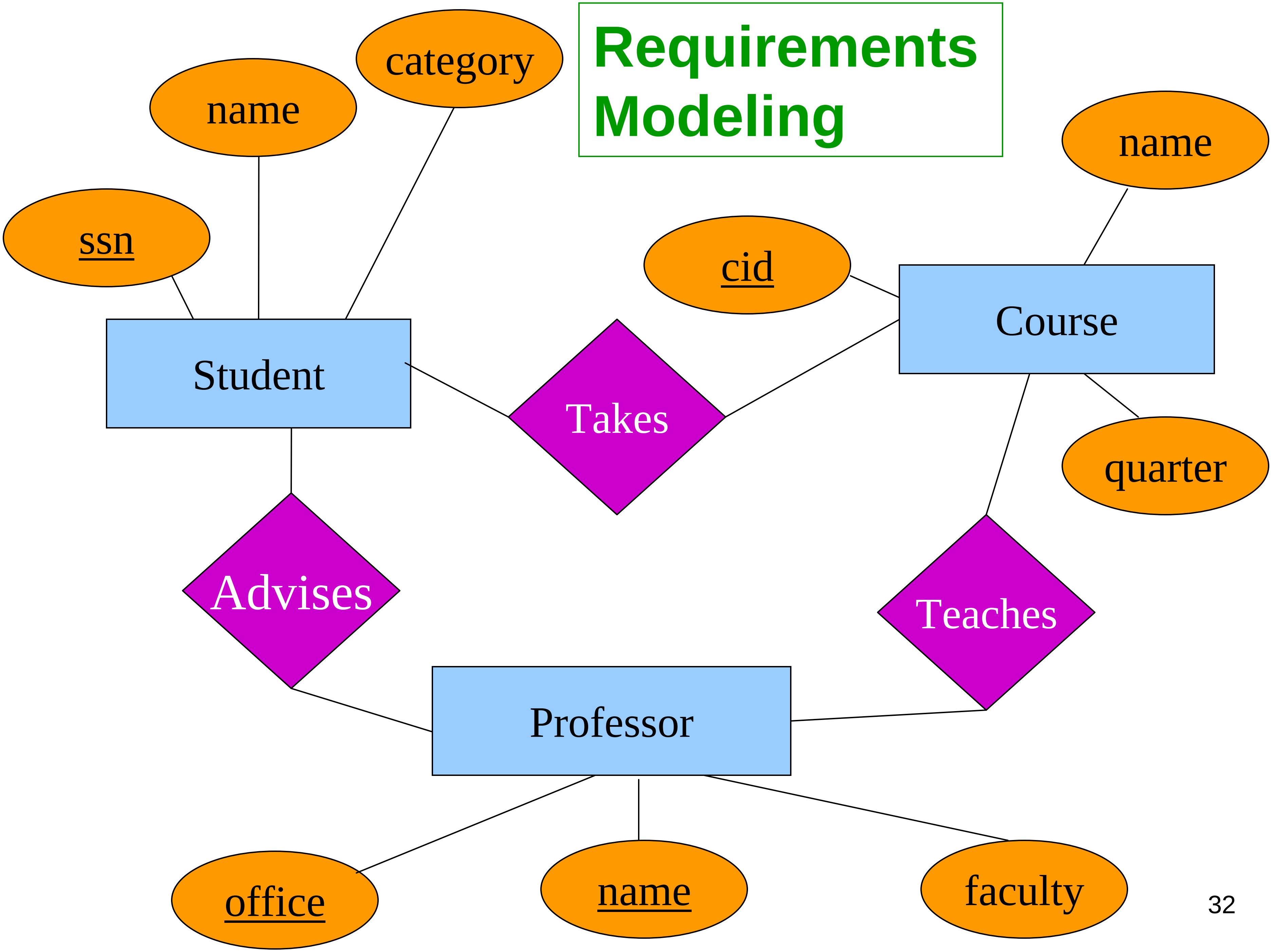
Host language is completely general (Turing complete) but gives no support for data manipulation

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

Requirements Modeling



Schema Design and Implementation

- Tables:

Student:

SSN	Name	Category
123-45-6789	Charles	undergrad
234-56-7890	Dan	grad
	…	…

Takes:

SSN	CID
123-45-6789	CSE444
123-45-6789	CSE444
234-56-7890	CSE142
	…

Course:

CID	Name	Quarter
CSE444	Databases	fall
CSE541	Operating systems	winter

- The logical schema separates the logical view from the physical view of the data.

Querying a Database

- *“Find all courses that Mary takes”*
- **S**(tructured) **Q**(uery) **L**(anguage)

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

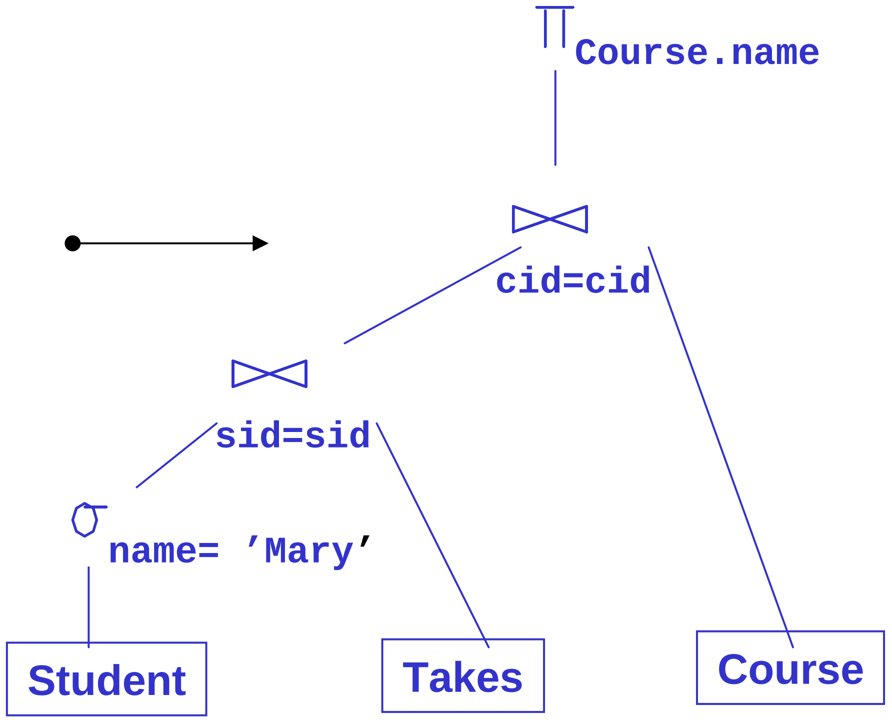
- The query processor figures out how to answer the query efficiently

Query Optimization

Goal: *Declarative SQL query* → *Query execution plan*

```
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