Semistructured Data & XML (Winter Term 2017/18)

XML Software Lab (Winter Term 2016/17)

(c) Prof Dr. Wolfgang May
Universität Göttingen, Germany

`may@informatik.uni-goettingen.de`

SSD&XML: Advanced Course in Informatics; 3+1 hrs/week, 6 ECTS Credit Points
XML Lab: Advanced Lab Course in Informatics; 2+2 hrs/week, 6 ECTS Credit Points

A comprehensive German-English dictionary can e.g. be found at

`http://dict.leo.org/`

TASKS IN INFORMATICS

1. Implementing a proposed solution: a job
   requires: good knowledge of common tools
2. Designing solutions: an interesting task
   requires: solid knowledge of up-to-date concepts
3. Development of concepts: a fascination
   requires: deep understanding and analysis of existing concepts

XML is a good example for all of them.

Aims of the Course

- knowledge of the concepts of the XML-World, practical experiences
  $\Rightarrow$ application-oriented
  (requires also to work on your own)
- backgrounds why XML developed, and why it is as it is
  $\Rightarrow$ understanding of concepts und developments
- underlying meta-concepts
  $\Rightarrow$ as an example of “Informatics” as a whole

Overview

- other talk “Introduction to XML” ...

Chapter 1
Introduction

Context and Overview

- Databases are used in many areas ... economics, administration, research ...
- originally: storage of information
  late 60s: Network Data Model, Hierarchical Model
  70s: Relational model, SQL – Lecture “Introduction to Databases”
- evolution: information systems, combination of databases and applications, distributed databases, federated databases, interoperability, data integration
- today: Web-based information systems, electronic data interchange
  $\rightarrow$ new challenges, semistructured data, XML
- tomorrow: Semantic Web etc.

1.1 Data Models

A *data model* defines the modeling constructs that can be used for modeling the application domain.

- *conceptual modeling:* application-oriented model
  - Entity-Relationship-Model (1976, only static concepts: entities and relationships, graphical)
    Lecture: Introduction to Databases
  - Unified Modeling Language (UML 1.0: 1996)
    comprehensive graphical formalism for modeling of processes, based on the object-oriented idea: classes and objects, properties, behavior (states and actions).
    Lecture: Software Engineering
- *logical data models:* (e.g. relational model)
  serve as *abstract data types* for implementations:
  - definitions of operations and their semantics, e.g. relational algebra
  - corresponding languages (as *application programming interfaces*): e.g. SQL
- *physical data models:* the implemented structures.

Data Model: Database Schema and Database State

Usually, for a database (for both, conceptual and logical models), its *schema* and its *state* are considered:

Database schema: the schema contains the *metadata* about the database, i.e., it describes the structure (in terms of the concepts of the data model).

The set of legal states is also described in metadata (e.g., by integrity constraints).

Database state: the state of a database is given as the currently stored information. It describes all objects and relationships that exist in the application at a given *timepoint*.

The database state changes over the time (representing changes in the real word), whereas the database schema is in general unchanged.

Logically spoken, the database state is an *interpretation* of the structure that is determined by the metadata.

Languages for Logical Data Models: In general, a language for operating on a data model consists of

- **D**ata **D**efinition **L**anguage (DDL) for schema definitions,
- **D**ata **M**anipulation **L**anguage (DML) for manipulating and querying database states.

Logical/Implementation Data Models

... there are many different data models.

Basically, all database approaches are grounded on the concept of a “data item” (german: “Datensatz”).

- logical data models and implementation models
  - network data model (IDS (General Electric) 1964; CODASYL Standard 1971), hierarchical data model (IMS (IBM) 1965); data records,
  - relational model (Codd 1970), SQL (IBM System R 1973; products since 1979 (Oracle), ISO SQL Standard 1986); tuples
  - object-oriented model (ODMG 1993; OQL); objects
- document-data model (SGML)
- semistructured data models, XML; nodes: elements, attributes, text
  - why?
  - evolution and current situation

1.2 Relational Model

- *relational model* by E.F. Codd (1970, IBM San Jose): mathematical foundation: set theory
- only a single structural concept: *relation* for entities/objects and relationship types (note that the notions "entity" and "relationship" from the ER model [1976] were not yet defined!)
- properties of entities/objects and relationship types are represented by *attributes*
- a relation schema consists of a name and a set of attributes
  Continent: Name, Area
- each attribute is associated with a *domain* that contains all legal values of the attribute. Attributes can also have *null values*:
  Continent: Name: VARCHAR(25), Area: NUMBER
- a **(relational) database schema** is given by a (finite) set of (relation)schemata:
  Continent: . . . ; Country: . . . ; City: . . . ; encompasses: . . .

RELATIONS

- a *(database) state* associates a *relation* with each *relation schema*.
- the elements of a relation are called *tuples*.
  Each tuple represents an object or a relationship:
  (Name: Asia, area: 4.5E7)

Example:

Continent	
Name	**Area**
VARCHAR(20)	NUMBER
Europe	9562489.6
Africa	3.02547e+07
Asia	4.50953e+07
America	3.9872e+07
Australia	8503474.56

Relations: Example

Continent

Name	**Area**
Europe	9562489.6
Africa	3.02547e+07
Asia	4.50953e+07
America	3.9872e+07
Australia	8503474.56

Country

Name	**code**	**Population**	**Capital**	**...**
Germany	D	83536115	Berlin	
Sweden	S	8900954	Stockholm	
Canada	CDN	28820671	Ottawa	
Poland	PL	38642565	Warsaw	
Bolivia	BOL	7165257	La Paz	
..	..	..	..	

encompasses

Country	**Continent**	**Percent**
VARCHAR(4)	VARCHAR(20)	NUMBER
R	Europe	20
R	Asia	80
D	Europe	100
. . .	. . .	. . .

- ... with referential integrity constraints
- abstract datatype for this model: relational algebra
- application interface: SQL

Query Language: SQL

- Since 1973 “SEQUEL – Structured English Query Language” in IBM System R (E.F. Codd (Turing Award 1981), D. Chamberlin (2001: co-designer of XQuery)) etc.; Research-only (IBM continued to sell only IMS until SQL/DS (1980), DB2 (1983))
  Stories: `http://www.mcjones.org/System_R/SQL_Reunion_95/`
  `http://www.nap.edu/readingroom/books/far/ch6.html`
- 1974 INGRES (UC Berkeley, M. Stonebraker; NSF funding), QUEL language, open-source.
  Led to the products INGRES (“Relational Technology Inc.” 1980, QUEL; since 1986 with SQL), INFORMIX (1981; since 1984 with SQL), SYBASE (1984, since 1987 with SQL)
- Oracle: founded in 1977 as “Relational Software” (L. Ellison worked before on a consultant project for CIA that wanted to use SEQUEL), 1983 renamed to “Oracle”.
  Product: 1979 Oracle V2 (SQL), first commercial relational DB system.
- Standard SQL: 1986 ANSI/ISO (least common denominator of existing products); SQL-1 1989 (Foreign Keys, ...); SQL-2 1992 (multiple result tuples in subqueries, SFW in FROM, JOIN syntaxes, ...); SQL-3 1999 (PL/SQL etc) ...
- 1995: 80% of all databases use the relational model and SQL

Query Language: SQL

```
SELECT name, percent
FROM country, encompasses
WHERE country.code = encompasses.country
   AND encompasses.continent = ’Europe’;
```

- intuitive to understand,
- *clause-based*, *declarative* language,
- *set-oriented*, *closed*: result of (nearly) each expression is again a relation,
- *orthogonal constructs*, can be nested (nearly) arbitrarily,
- *functional programming paradigm*: each SFW query is a function that maps relations to another relation. Such functions can be nested.

... so far the things you have learnt in “Databases” about the relational model and SQL.

1.3 Concepts and Notions

- the relational model is a *data model*.
- (relational) databases follow a 3-level architecture:
  - *physical level/schema:* actual storage of tables in files, as sequenced records, with length indicators etc; additional index files, and allocation tables.
  - *logical level/schema: user level.*
    Relational model (*logical data model*) with given database schema (table names, attributes, keys, foreign keys etc), relational algebra, SQL (*database language*).
    Abstract, *declarative*, *set-oriented* language, distinguished notions of schema and state.
    Internal: mapping to physical schema. Admin can change the physical schema and adapt the mapping without effecting the logical schema.
  - *external level (optional):* possible views, given by SQL queries.
    A *view* is (any kind of) a mapping from underlying “base” data to derived information.
- note: SQL is the only language with which users work on relational data. Relational data exists only inside databases.

Concepts: Preview

- network data model: mainly a physical data model; "logical" model on a very low level of abstraction.
  No database language, only some data-management-oriented operations extending a common programming language.
- relational model: abstract/logical data model, relational algebra, declarative, set-oriented query+update language.
- early semistructured data models (OEM, F-Logic etc.): not comparable, separate experiments how to extend functionality without losing the advantages from relational databases and SQL.
- for XML there are several languages (“views” can also be defined in several ways), and XML exists also as a data structure used in non-database tools.

1.4 Aside: Really Declarative Languages ...

SQL is already called “declarative”: express what, not how.

But there is an even more declarative language family: *logic-based* languages.

Relational Calculus, Datalog

- Facts (tuples) are called “atoms”:
  country(“Germany”, “D”, 83536115, 356910, “Berlin”, “Berlin”),
  city(“Berlin”, “Berlin”, “D”, 3472009), etc.
- queries are given as “patterns” with free variables:
  ?- country(N,C,Pop,Area,CapProv,Capital).
  yields a set of *answer bindings* for the variables N,C,Pop,Area,CapProv,Capital.
- Projection via *don’t care* variables:
  ?- country(N,_C,Pop,_Area,_,_).
  yields a set of *answer bindings* for the variables N and Pop.
- Selection: ?- country(“Germany”, “D”, Pop, Area,_,_). binds only Pop and Area.

Relational Calculus (cont'd)

- Selection as Conjunction:
  ?- country(N, C, Pop, _,_,_), Pop $>$ 1.000.000. binds N, C, Pop
  ?- country(N, _, _Pop, _,_,_), _Pop $>$ 1.000.000. returns only the set of names of countries with more than 1000000 inhabitants.
- Joins as conjunctions:
  ?- country(N,_C,_,_,Area,_,_), encompassed(_C,Cont,Perc), continent(Cont, ContArea).
  ?- country(_, "D",_,_,_,CapProv,Capital), city(Capital, CapProv, "D", Pop, _, _)

Datalog

- Views as "derived/virtual relations":
  ancestor(X,Y) :- parent(X,Y).
  ancestor(X,Y) :- ancestor(X,Z), parent(Z,Y).
  ?- ancestor(X,Y).
  can express e.g. transitive closure (recursive rules and fixpoint semantics).
- flows_transitive(Name, Sea) :- river(Name, _, Sea, _).
  flows_transitive(Name, Sea) :- river(Name, River, _ , _), flows_transitive(River, Sea).
  ?- flows_transitive(River, Sea).

[see Datalog/tcRivers.P]

REMARK

The term "Database" does not only mean the relational model & SQL, but is a general notion:

- persistent storage
- mass data
- multiuser concepts
  - access control/safety
- transaction concepts
  - correctness/consistency
  - safety: rollback, recovery

... all these are *general concepts* that apply for the network data model, the relational model, the object-oriented model, and also for XML databases.