Intelligent Information Systems

WS 2016/17

Deduction in Datalog and SQL

Chapter 2
The essential **topic** of this lecture is to understand the importance of the idea of using the concept of derived data (and of the corresponding technology).

We already learnt last week, that databases „containing“ stored as well as derived data are called **deductive databases** in scientific terminology.

In order to manage derived data, we need special abilities of our DBMS that turn it into a **deductive DBMS**.

**Managing** derived data means
- **designing** a deductive DB schema
- incl. **specifying** derivation rules
- **evaluating** queries over derived data
- **controlling** consequences of changes of derived data
• **Storing** data is the „normal“ way of keeping data (in some storage device).

• Every data element can be stored (in principle).

• But there are data elements that do not (necessarily) have to be stored, but (possibly) can be computed from stored (or other) derived data elements instead. We call such pieces of data „derived“ data (elements).

• It would be more precise to speak about „derivable“ data, because may be a derivable piece of data is actually never derived later on (or has never been derived up till now). We will use the term „derived“ data nevertheless.

• Part of the derivable data can redundantly be stored (after having been derived first) in order to avoid costly rederivation. We call such data materialized (derived) data.
Another Topic of the Lecture: How to Derive Data?

How do inference (derivation) engines look like? Should they be part of a DBMS or external tools?

We follow a specific approach in IIS!
2. Datalog and SQL

2.1 Datalog and SQL: Learning by Doing
2.2 Datalog: Syntax and (Basic) Semantics
Case Study 1: The British Royal Family

A 1st case study on using derived data will give us a concrete start into using the techniques of Deductive Database technology: A Genealogical Database
The Royal Family Tree: Our Example Application Domain

The current family tree of the British royal family (starting from the Queen and her husband)
How to „put a family tree into a (relational) database“?

(i.e., how to design a basic genealogical database)
Conceptual Modeling of the Family Tree DB

- Traditional way of designing a relational DB:
  Start by **conceptually modelling** the resp. application domain

- Corresponding **ER diagram** (Entity-Relationship approach):

![ER Diagram](image)

- Basic technique for **logical design**: deriving a **relational DB schema** from an ER diagram
  - per **entity type** one table: Each attribute of the E-type turned into attribute of table
  - per **relationship type** on table, too: Each attribute of the R-type into attribute of corr. table, plus key attributes of of participating E-types (as foreign keys)
The **basic relational schema** for this ER diagram looks as follows:

- person (title, name, birth, death, sex)
- parents (father, mother, child)
- marriage (husband, wife, married, divorced)

Roles in R-types are turned into attributes of the corr. R-table.
The initial design can be improved by integrating certain relationship tables into entity tables in case of 1:N functionalities:

- **person** (title, **name**, birth, death, sex)
- **parents** (father, mother, child)
- **marriage** (husband, wife, married, divorced)

New entity table: **person** (title, **name**, birth, death, sex, father, mother)

New relationship table: **marriage** (husband, wife, married, divorced)
One possible (initial) relational format for family trees: just two tables!

The entire family tree is „in“ these two tables.

Disadvantage(?) : Many empty cells („nulls“)
Data about „existence“ (birth, death, sex, name etc.) cannot be derived, but have to be entered manually and stored in tables. The same applies to marriage and divorce.

All other data about family relationships, however, are derivable from these stored data, provided we can express them using suitable derivation rules.

Blood relation (or: consanguinity) only!

Numbers indicate percentage of „common blood“

In this graph, relatives from the maternal side are given only – paternal relatives analogously.
Example: Specification of Being an Uncle of Someone

How to derive data about the uncles of a person?

Uncle (from Latin: avunculus "little grandfather", the diminutive of avus "grandfather") is a family relationship or kinship, between a person and his or her parent's brother, parent's brother-in-law or parent's cousin. (from: en/wikipedia)

(Here we restrict ourselves to uncle in the narrow sense)
"Uncle" as a Derived Table

Basic principle of relational DBs: Every query returns a table!
How to Derive a Particular Uncle Row?

A derivation strategy a human would use, might serve as a suitable strategy of a deduction "engine".

(Is this some kind of "artificial intelligence"?)
If derived tables are to be generated as query results, we have to express the deduction „strategy“ as a relational query (and store the query, e.g., as a view in SQL).

How does the SQL view look like that can do this kind of deduction?