Deduktive und objektorientierte Datenbanken

Gerd Stumme
Christoph Schmitz

Wintersemester 2004/05

ENDOWED CHAIR OF THE HERTIE FOUNDATION
Knowledge and Data Engineering
DEPARTMENT OF MATHEMATICS & COMPUTER SCIENCE

Organisatorisches

Vorlesung

- Beginn: 20. Oktober 2004
- mittwochs 8.30-10.00, Raum 0443
- Die Folien sind teilweise in Englisch, der Tafelanschrieb auf Deutsch

Übungen

- nach Absprache
- Beginn: 1. November 2004
- wird als Präsenzübung abgehalten (s. nächste Folie)

Unterlagen

- Ein Reader mit relevanter Literatur ist bei Christoph Schmitz erhältlich
- Die Bibliothek hält Kopien des Buches von Abiteboul el al bereit.

Prüfung

- Die Prüfung wird je nach Teilnehmerzahl mündlich oder schriftlich abgehalten.

Organisatorisches

Präsenzübung bedeutet

- **selbständiges Bearbeiten** des Übungsblattes in Kleingruppen à 3-4 Personen

  unter Betreuung des Assistenten

- **kein prinzipielles Wiederholen** des Vorlesungsstoffs
- **kein Vorrechnen** der Musterlösung etc. (Diese wird später zur Verfügung gestellt.)
- **Nötig dafür:**
  - selbständige Vorlesungsnachbereitung **vor** der Übung
  - Mitbringen des Skriptes
  - eigene Aktivität entfalten

Organisatorisches

Warum ein neues Übungskonzept?

•aktives Erarbeiten des Vorlesungsstoffes bringt mehr

•Zusammenhänge im Stoff erkennen

•strukturiertes Denken und selbständiges Arbeiten lernen

•Teamarbeit lernen

•Erklären lernen (als Tutor und als Teilnehmer)

•Klausurtraining ;-)

•*Ihr Studium der ... haben Sie abgeschlossen. Zu Ihren persönlichen Stärken zählen Sie Eigeninitiave, Kommunikations- und Kooperationsbereitschaft, Teamarbeit.*

(Typischer Anzeigentext)

Organisatorisches

Sprechstunden nach Absprache:

Prof. Dr. Gerd Stumme (Vorlesung):	stumme@cs.uni-kassel.de	0561/804-6251
Dipl.-Inform. Christoph Schmitz (Übungen):	schmitz@cs.uni-kassel.de	0561/804-6254

FG Wissensverarbeitung, FB Mathematik/Informatik

Raum 0439, Wilhelmshöher Allee 73

Informationen im Internet: http://www.kde.cs.uni-kassel.de

Hier ist u.a. folgendes zu finden:

- aktuelle Ankündigungen
- Folienkopien
- Übungsblätter
- Literaturempfehlungen
- Termine

ENDOWED CHAIR OF THE HERTIE FOUNDATION
Knowledge and Data Engineering
DEPARTMENT OF MATHEMATICS & COMPUTER SCIENCE

Bibliography

Main Sources

[AHV 95] S. Abiteboul, R. Hull, V. Vianu: Foundations of Databases. Addison Wesley, Reading, MA, 1995

[Co 98] S. Conrad: A Logic Primer. In [CS 98], 5-30

[KLW] M. Kifer, G. Lausen, J. Wu: Logical foundations of object-oriented and frame-based languages. Journal of the ACM, 42:741-843, 1995.

[CLN 98] D. Calvanese, M. Lenzerini, D. Nardi: Description Logics for Conceptual Data Modeling. In [CS 98], 229-264

Bibliography

Further Material

[Br 96] S. Brass: Bottom-up Query Evaluation in Extended Deductive Databases, Habilitation, Universität Hannover 1996

[CS 98] J. Chomicki, G. Saake (eds.): Logics for Databases and Information Systems, Kluwer, Boston 1998

[GL 91] M. Gelfond, V. Lifschitz: Classical Negation in Logic Programs and Disjunctive Databases. J. New Generation Computing 9(3/4), 1991, 365-386

[Gr 01] M. Grohe: Generalized Model-Checking Problems for First-Order Logic. Proc. 18th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2001), LNCS, Springer, Heidelberg 2001

[KL 97] M. Kifer, G. Lausen: F-Logic: A Higher-Order Language for Reasoning about Objects, Inheritance, and Scheme

[Vi 97] V. Vianu: Databases and Finite-Model Theory. In N. Immerman and P. Kolaitis, editors, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 31, pages 97-- 148. American Mathematical Society, 1997.

Overview

A Introduction and Basics

A.1 Motivation	[Br 96, Intro, especially pp. 5-10], [KL 97, Intro], [CLN 98, pp. 230-231]
A.2 Logic Basics	[Co 98, Sect. 2.2, pp. 6-13+14] + [AHV 2.3]
A.3 The Relational Model	[AHV 3]
A.4 Conjunctive Queries	[AHV 4]
A.5 Adding Negation	[AHV 5]

[AHV n] = Chapter n from [AHV95]

[CS n] = Chapter n from [CS98]

Overview

B Datalog and Recursion

B.1 Datalog	[AHV 12]
B.2 Evaluation of Datalog	[AHV 13]
B.3 Recursion and Negation	[AHV 14]
B.4 Negation in Datalog	[AHV 15]
	Further literature: [Br 96]

C Object-Oriented Databases

C.0 Introduction	
C.1 F-Logic	[KL 97], [KLW]

A Introduction and Basics

A.1 Motivation

A.1.1 Deductive Databases

based on [Brass 1996]

Motivation

1. Wie würden Sie den Stammbaum einer Familie als relationale Datenbank modellieren? Geben Sie das Schema an.
2. Geben Sie eine SQL-Anfrage an, mit der Sie alle Vorfahren einer gegebenen Person feststellen können.
3. Geben Sie die gleiche Anfrage in der Sprache der relationalen Algebra an. (Warum ist es wichtig, dass man dies kann?)

Motivation

1. Eine Tabelle (um es einfach zu machen):

   elternteil: {[Elternteil:string, Kind:string]}

2. „vorfahre" als Sichtdefinition (in DB2):

```
create view vorfahre(V,X) as
  (select Elternteil, Kind
   from elternteilvon
      union all
   select v.Elternteil, e.Kind
   from vorfahre v, elternteilvon e
   where v.Kind = e.Elternteil)
```

```
select V
from vorfahre
where X=„Max"
```

3. Diese Anfrage lässt sich in der relationalen Algebra nicht ausdrücken. Dabei wäre dies wichtig, um die Semantik festzunageln – d.h. damit alle Datenbankmanagementsysteme auf die gleiche Frage bei dem gleichen Datenbestand auch wirklich die gleiche Antwort liefern!

4. Wir müssen also die Semantik anders festlegen ...

History

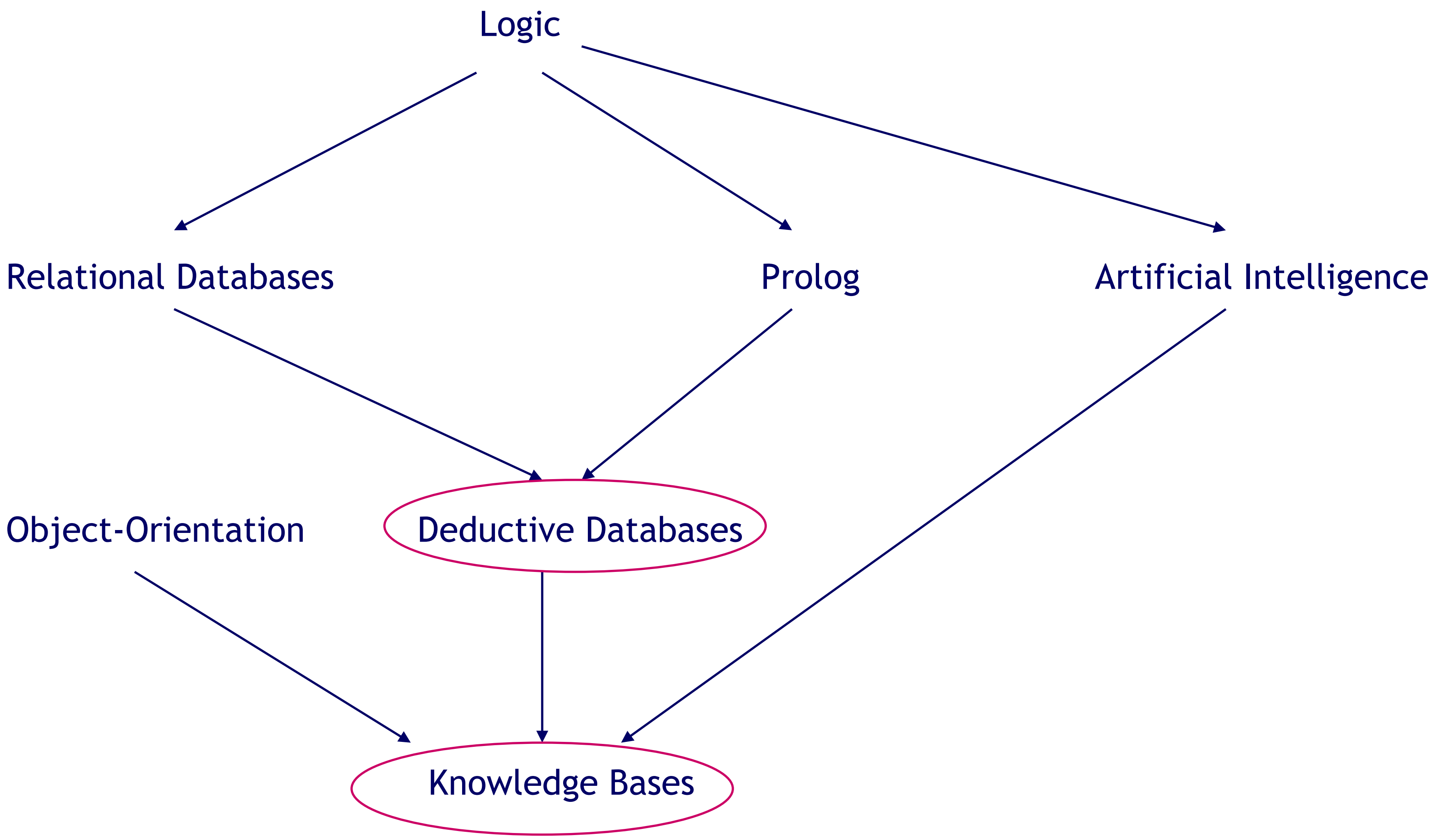
Databases

- At the beginning, not much theory
- Codd, 1970: Relational Data Model
    - first only theoretically defined
    - critized for being un-implementable
- one decade later: first commercial relational database management systems
- today: state of the art

Prolog

- early 70ies: Predicate Logic as programming language

History


Logic
Relational Databases
Prolog
Artificial Intelligence
Object-Orientation
Deductive Databases
Knowledge Bases

Deductive Databases

- A deductive database consists of
  - a relational database: it defines a number of relations (predicates) „extensionally“ by enumerating all tuples
  - a logic program: it defines a number of relations „intensionally“, by given defining rules (logical formulas)

- Alternative ways to understand deductive databases:
  - a relational database with a new query language (Datalog instead of SQL)
  - a logic program with a large number of facts
  - an automated theorem prover for many formulas of a special type

- It is difficult to state when (and by whom) deductive databases and Datalog arose. It is a development with many contributors. See [Brass 96] for a discussion.

Motivation

- A typical database application consists of three parts:
    - user interface: manages the dialogue between user and system
    - code for manipulating the data (format changes, combining and aggregating data) and for checking integrity
    - data base access (searching & sorting, persistence, concurrent access
- Problem I: different languages/compilers
    - special languages (eg Tcl/Tk, HTML) and editors
    - imperative languages (eg C, C++, Java)
    - SQL
- Problem II: Impedance mismatch
    - Eg: the database returns a set of answers, but the programming language needs a loop to fetch every single tuple in it.

Motivation

- Deductive databases solve this problem, as they are both, programming language and a data base. These two aspects are tightly coupled.

- This wish also led to object-oriented databases. While deductive databases are based on the set-oriented paradigm, in OO systems, the tuple-oriented view „won“.

- Even commercial relational databases allow nowadays to store procedures. This is a logical development:
    - in early times, each application had its own data files
    - then the data was shared (in the database), but no controlled way of sharing code
    - The latter is the aim of all of deductive db, OO db, and procedural extensions of relational db.

- Now the functionality of the database (persistency, data dictionary, access rights etc.) is applied to the procedures as well.

- Global query optimization (which requires knowledge of the procedures) is now possible.

Comparison of deductive with relational databases

- Queries which can be formulated within SQL are restricted.
    - No recursion (eg asking for the transitive closure) possible.

- The notion of a view is not fully integrated in the relational model, views are often implemented by query rewriting which not always provides correct SQL. In deductive dbs, queries are „first class citizens", the notion of derived relations is essential.

- The relational model is unsuitable for highly regular data.
    - Eg, a bus schedule is easier to describe by rules than by listing all facts.

Comparison of deductive databases with Prolog

- Deductive dbs are ‚more logical' and have less control (eg, the order of rules does not matter):
    - In Prolog it is clear which predicate is called, the programmer can optimize for it. He knows in advance which variables are bound, and which are free.
    - In deductive dbs, (almost) any query is possible. Hence the system has to optimize. (This is in the database tradition to work on query optimization.)
- Termination of a query would be expected.
    - In modern systems, almost every Prolog query is allowed, thus termination cannot be guaranteed in general.
    - But large subsets of queries have been identified for which termination can be guaranteed.
- Prolog has unsufficient support for external memory.
    - It is tuple-oriented, where deductive dbs are set-oriented. The latter fits better with the block-orientation of eternal memory.

Some typical applications

Expert systems

Expert knowledge is often encoded by rules (not necessarily logic programming rules). Often large sets of facts are also needed.

→ deductive dbs are good support for expert systems

→ more support needed for creating user interfaces, defining a structured user dialog, and better explanation facilities.

Decision support systems

- have the task to aggregate information from large datasets, or to find interesting cases in them.
- Often temporal development should be displayed.
- Flexible and ad-hoc queries should be supported.

→ again, deductive dbs are good support

→ see also [Chomicki, Saake] for temporal logics.

References to examples are given in [Brass 1996].

Some typical applications

Hierarchical design

Computer-aided design of hierarchically structured objects is a good application of deductive dbs, because they support hierarchies (by computing transitive closures).

$\rightarrow$ A problem is performance, therefore OO languages are preferred.

$\rightarrow$ But it is easy to check for inheritance in Datalog, while it is difficult to write a C++ program.

Complex integrity constraints

are usually defined in some logic, so deductive dbs are a natural choice for checking them.

• In relational dbs, one can implement triggers or procedures for that task but it is not easy to verify that they really enforce a given set of constraints.

Some typical applications

Parsing

There is a strong relation between logical rules and grammatical rules. For analyzing natural language, we also have large sets of data (eg, dictionaries).

• Prolog does not allow left-recursive rules, and backtracking involves duplicating syntactical structures.

• Bottom-up evaluation of deductive dbs overcomes these problems.

Graph-structured data

• deductive dbs can compute paths in graphs (again because of the transitive closure).

• ‚Locality of access' becomes an issue, because graphs are often too large to fit completely into main memory.

Some typical applications

Integration of heterogeneous datasources

is still a big issue, both in research and in industry.

- The powerful view concept of deductive dbs is helpful.

Semantic Web

For establishing this second generation WWW, enhanced by ontologies and metadata for better machine support, all of the above issues are important.

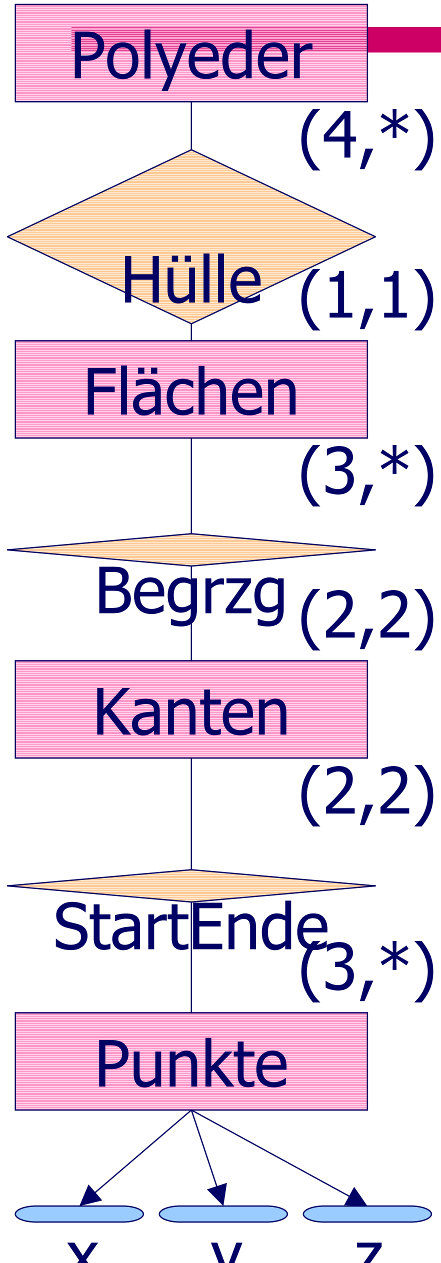
→ After some rather silent years, „Deductive Databases“ becomes an interesting topic again!

A Introduction and Basics

A.1 Motivation

A.1.2 Object-Oriented Databases

Nachteile relationaler Modellierung



Polyeder

PolyID	Gewicht	Material	. . .
cubo#5	25.765	Eisen	. . .
tetra#7	37.985	Glas	. . .
.			. .

Flächen

FlächenID	PolyID	Oberfläche
f1	cubo#5	. . .
f2	cubo#2	. . .
. . .	. . .	. . .
f6	cubo#5	. . .
f7	tetra#7	. . .

Kanten

KantenID	F1	F2	P1	P2
k1	f1	f4	p1	p4
k2	f1	f2	p2	p3
. . .	. . .	. . .		

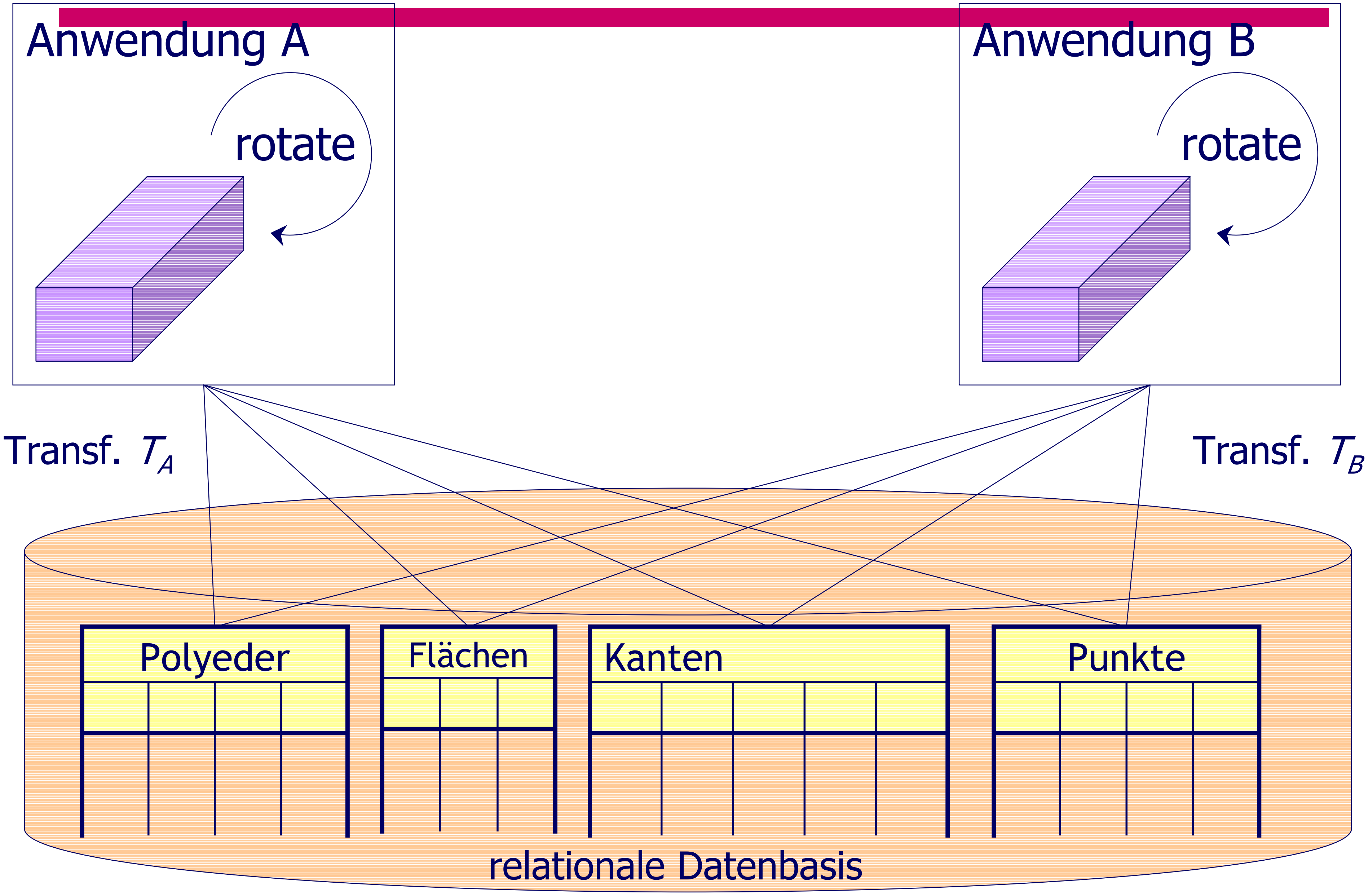
Kanten

PunktID	X	Y	Z
p1	0.0	0.0	0.0
p2	1.0	0.0	0.0
. . .	. . .	. . .	

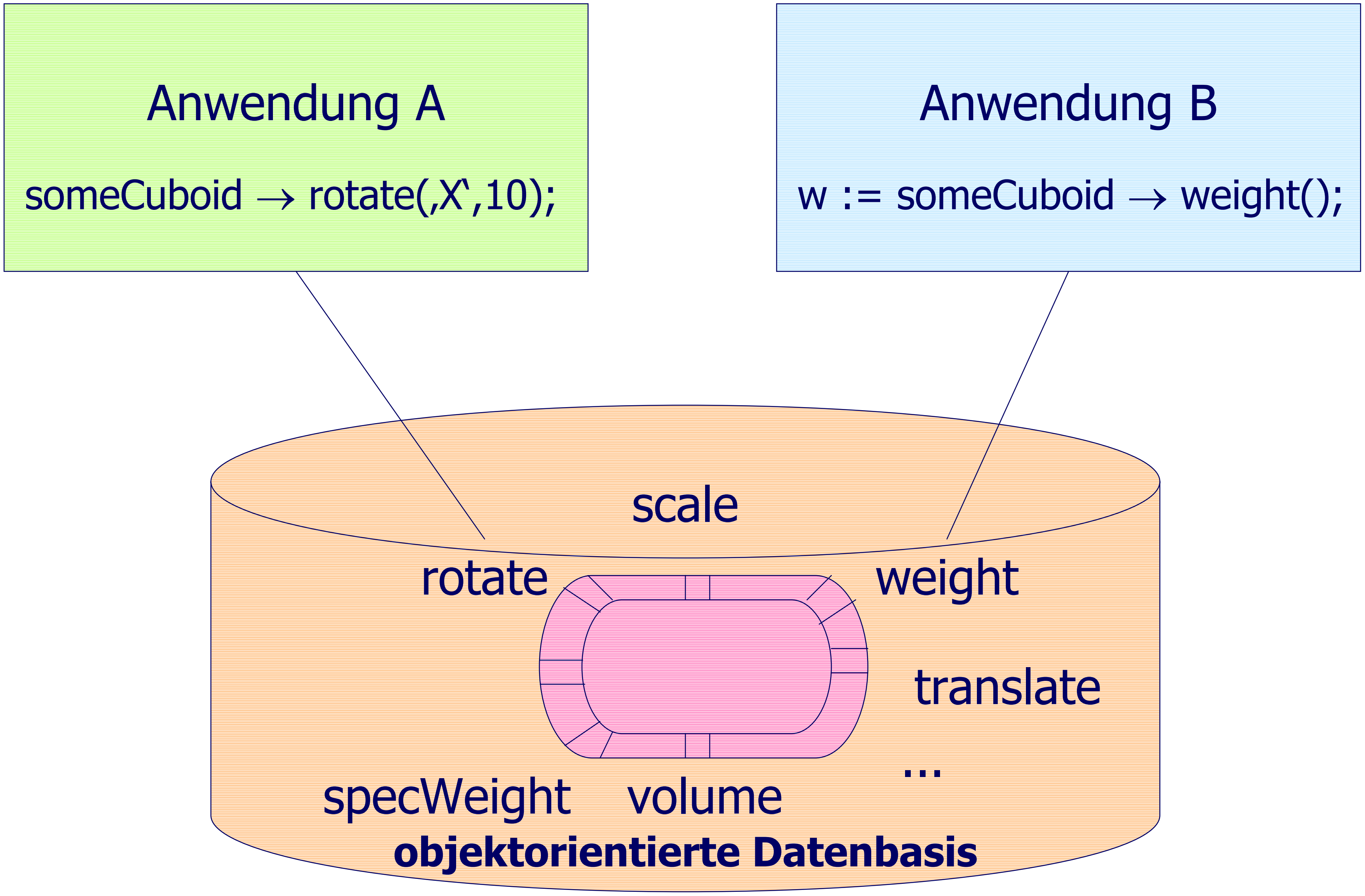
Nachteile relationaler Modellierung

- **Segmentierung**
- **Künstliche Schlüsselattribute**
- **Fehlendes Verhalten**
- **Externe Programmierschnittstelle**

Visualisierung des „Impedance Mismatch“



Vorteile objektorientierter Datenmodellierung


Anwendung A
someCuboid → rotate(‚X‛,10);
Anwendung B
w := someCuboid → weight();
scale
rotate
weight
translate
...
specWeight
volume
objektorientierte Datenbasis

Vorteile objektorientierter Datenmodellierung

- **„information hiding“/Objektkapselung**
- **Wiederverwendbarkeit**
- **Operationen direkt in Sprache des Objektmodells realisiert (kein Impedance Mismatch)**