Beyond the Relational Model

Serge Abiteboul
INRIA Saclay & ENS Cachan

Recall for first lecture: Always question everything

In industry: to challenge the well established guys

In academia: to discover new problems

Revisit the models, languages, principles

Main motivations

- To facilitate application development
- Performance to scale to always more data and queries
- To offer more in terms of reliability, security, etc..

We study here some of the main attempts to go beyond the relational model

Organization

Trees and XML

Graphs and object databases

NoSQL

OLAP (On-line analytical processing)

Conditional tables

Next class: Semantic Web

Trees and XML

Introduction

Trees are useless

***A tree is a tree**. How many more do you have to look at?*

Ronald Reagan, governor of California, opposing the expansion of Redwood National Park (1966)

We don't need anything beyond relations. These things are useless. Reject!

***Anonymous referee** (circa 1990)*

Knowledge lives in trees

*But of **the tree of the knowledge** of good and evil, thou shalt not eat of it: for in the day that thou eatest thereof thou shalt surely die.*

Genesis, 2. 17

The Bible does not say
"But of the two dimensional table of knowledge of good and evil … "

Using trees to represent data: an old idea

From the 60s and IMS (Hierarchical database model)

- But fully procedural languages and records at a time

All really started in the 80s and Non-first-normal-form

- François Bancilhon in France et Hans Schek in Germany
- PhD thesis of Nicole Bidoit

Non-First-Normal-Form N1NF

Name	Child	Car
Alice	Toto Lulu	Jaguar 2CV
Bob	Mimi Zaza	Mustang Prius

Data live in 1NF relations:
Entries of tables should be atomic

Data would prefer to live in infamous nested relations
aka V-relations
aka N1NF relations
aka NF2 relations

The devil is in the details

V-relations

A	B	C
1	1 2	1
2	2 3	
3	1 3	3 4

A	B
1	1
1	2
2	2
2	3
3	1
3	3

A	C
1	1
3	3
3	4

A
1
2
3

A is a key

No new power

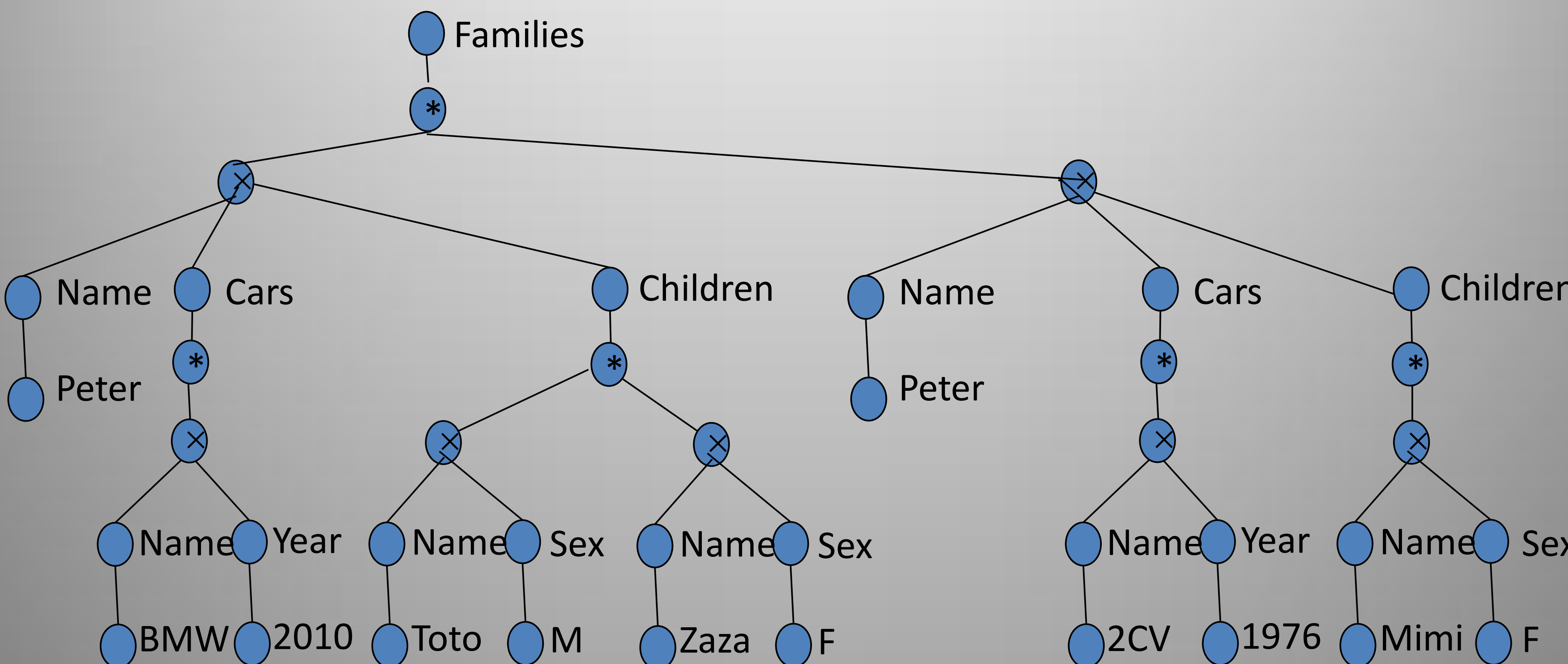
N1NF-relations

A	B
1	
1	1
1	2
1	3
1	1 2
1	1 3
1	2 3
1	1 2 3

A is not a key

The size is now possibly exponential in the size of the domain

Complex object model: set and tuple constructors



Logic for complex objects

Logic: main novelty – variables denoting sets

Example: AbouBanat query

$\{ T.Father \mid Families(T) \wedge \forall X,x \; (T.Children = X \wedge x \in X \Rightarrow x.Sex = F) \}$

The father of only girls

Algebra for complex objects

Name	Child
Alice	Toto
Bob	Mimi

Set of sets

Name	Child
Alice	Toto
Bob	Mimi

Name	Child
Alice	Toto

Name	Child

Name	Child
Bob	Mimi

Unnest **Nest** **Unnest**

Name	Child	Car
Alice	Toto	
Bob	Mimi Zaza	Mustang
Bob	Lulu	Prius

Name	Child	Car
Bob	Mimi	Mustang
Bob	Zaza	Mustang
Bob	Lulu	Prius

Name	Child	Car
Bob	Mimi Zaza Lulu	Mustang Prius

Name	Child	Car
Bob	Mimi	Mustang
Bob	Zaza	Mustang
Bob	Lulu	Prius

Identity

Results

Equivalence theorem: algebra and logic have same expressive power

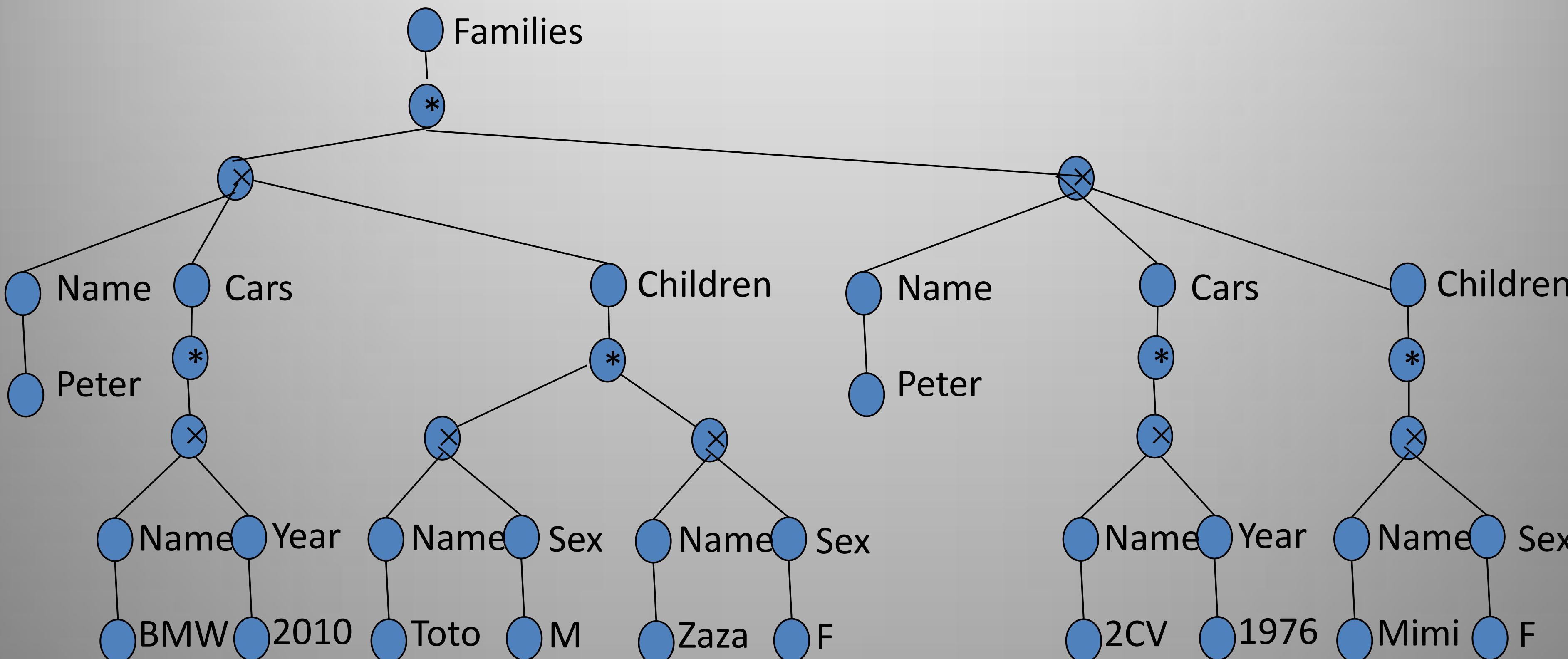
Remark: one can compute transitive closure using algebra/logic (Cool!)

Each new level of nesting introduces one more exponential

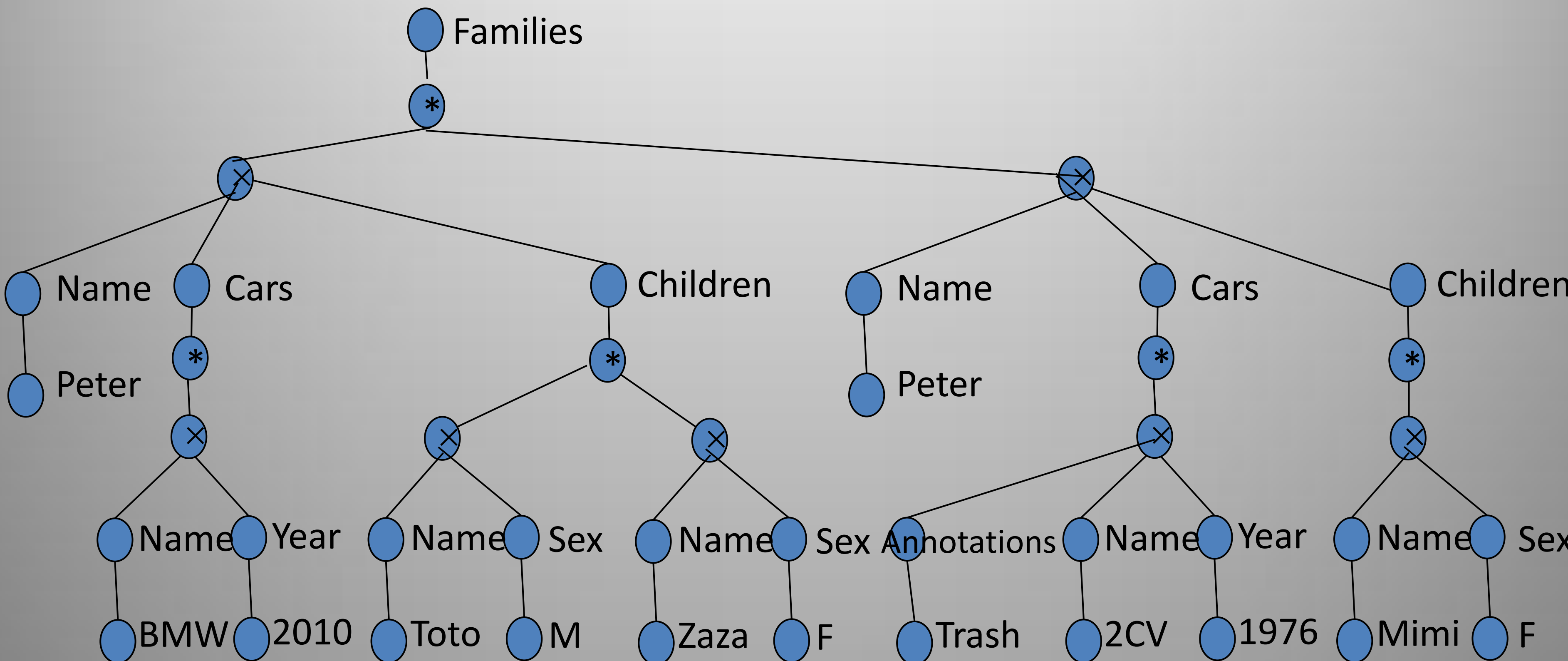
- A query is in the algebra/calculus iff it has elementary time complexity (similarly space complexity)

$$2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}$$

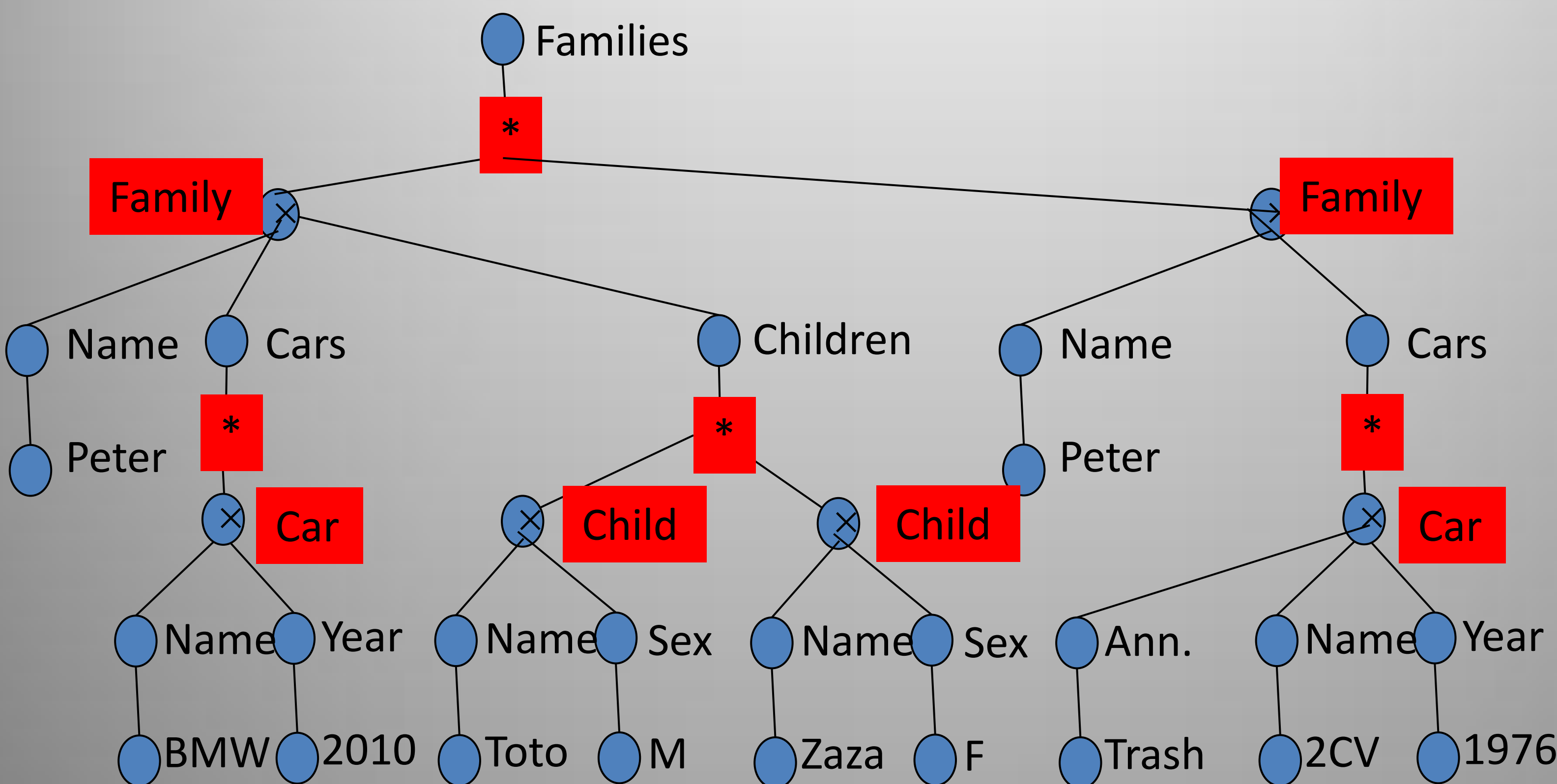
From complex objects to semistructured data



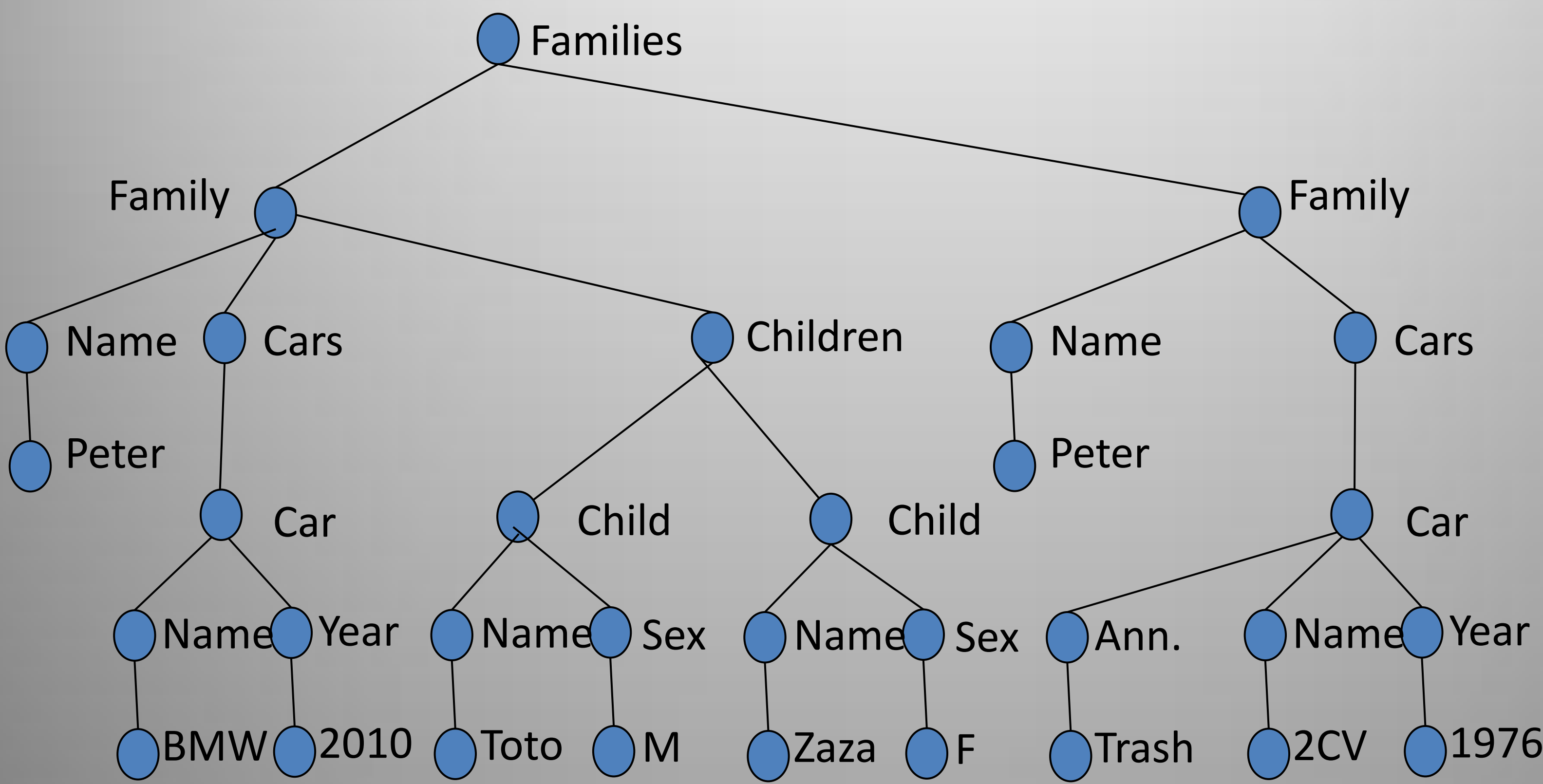
Revolution 1: more flexibility



Revolution 2: get ride of *-nodes and name all nodes



XML = ordered, labeled, unbounded trees



This is better adapted to a Web context

Self describing data: No separation between schema and data

Flexibility

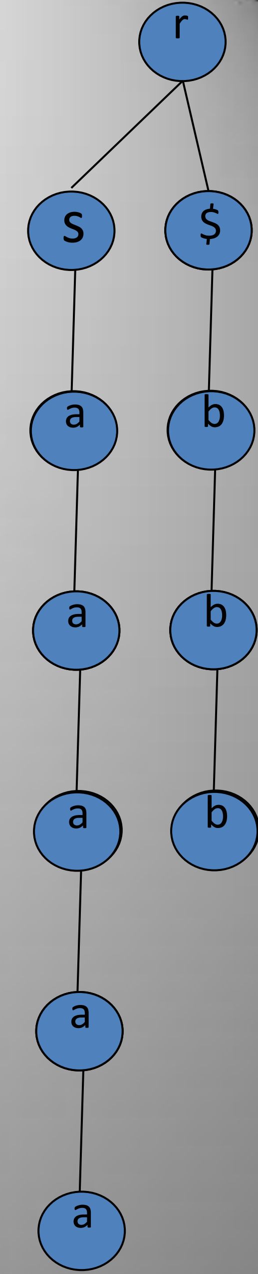
Not such a big deal

A syntax for inlining and exchanging data

<families><family><name>Peter<Name><Cars><Car><Name>BMW</Name><Year>2010</Year></Car></Cars><Children><Child> …

> The more things change,
> the more they stay the same

What else? The trees are unbounded



Like nested relations, trees are **unbounded in width**

Unlike nested relations, they are **unbounded in depth**

One can simulate 2 counter machines with 2 branches

- I am still looking for a real application that simulate 2 counter machines with XML documents?
- XML documents are rarely deep

But even for bounded trees there are fun questions

- Rich study of query languages
- Typing and semantics

What else? the trees are ordered
Unranked labeled *ordered* trees = XML

Ignore order

Classical optimization

Respect order

Totally new ball game

Reconcile?

Order is often painful for optimization

The XML world

Typing

- **Tree automata**, DTD, XML Schema, Relax NG…

Query languages

- XPATH

  article[1]/auteurs/auteur[2]

- Xquery

  ```
  FOR $ p IN document ("bib.xml") / / publisher
  LET $ b: = document ("bib.xml) / / book [publisher = $ p]
  WHERE count ($ b)> 100
  RETURN $ p
  ```

- **Monadic datalog, FO, Pebble automata**…

Transformation language: XSLT

Other standards around XML

- SOAP, DOM
- XML dialects: RSS, WML, SVG, XLink, MathML

Lots of open source software

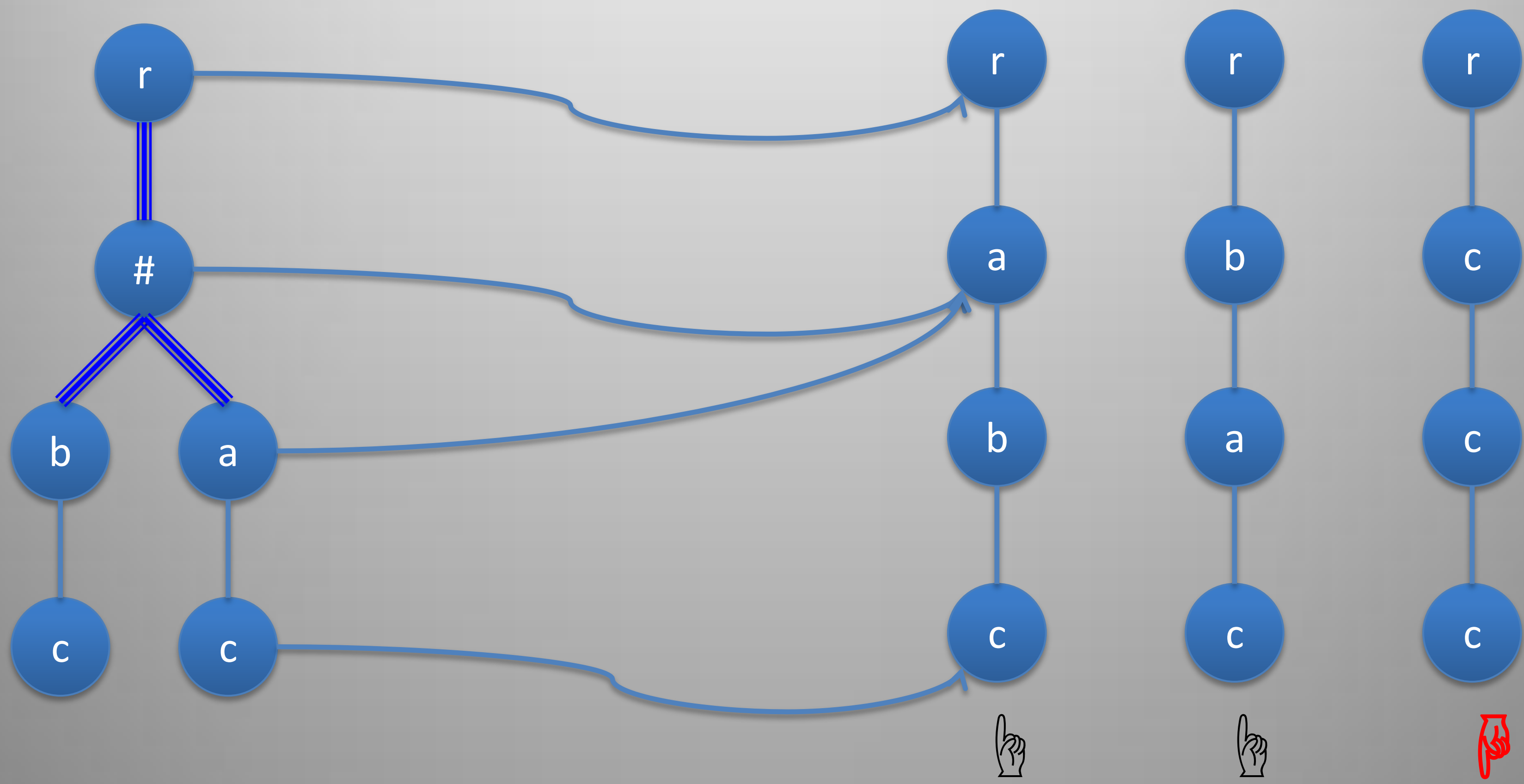
Query containment (continuing jewel of 1st class)

- Recall **Homomorphism Theorem**

q1 ⊆ q2 iff there is a homomorphism from q2 to q1

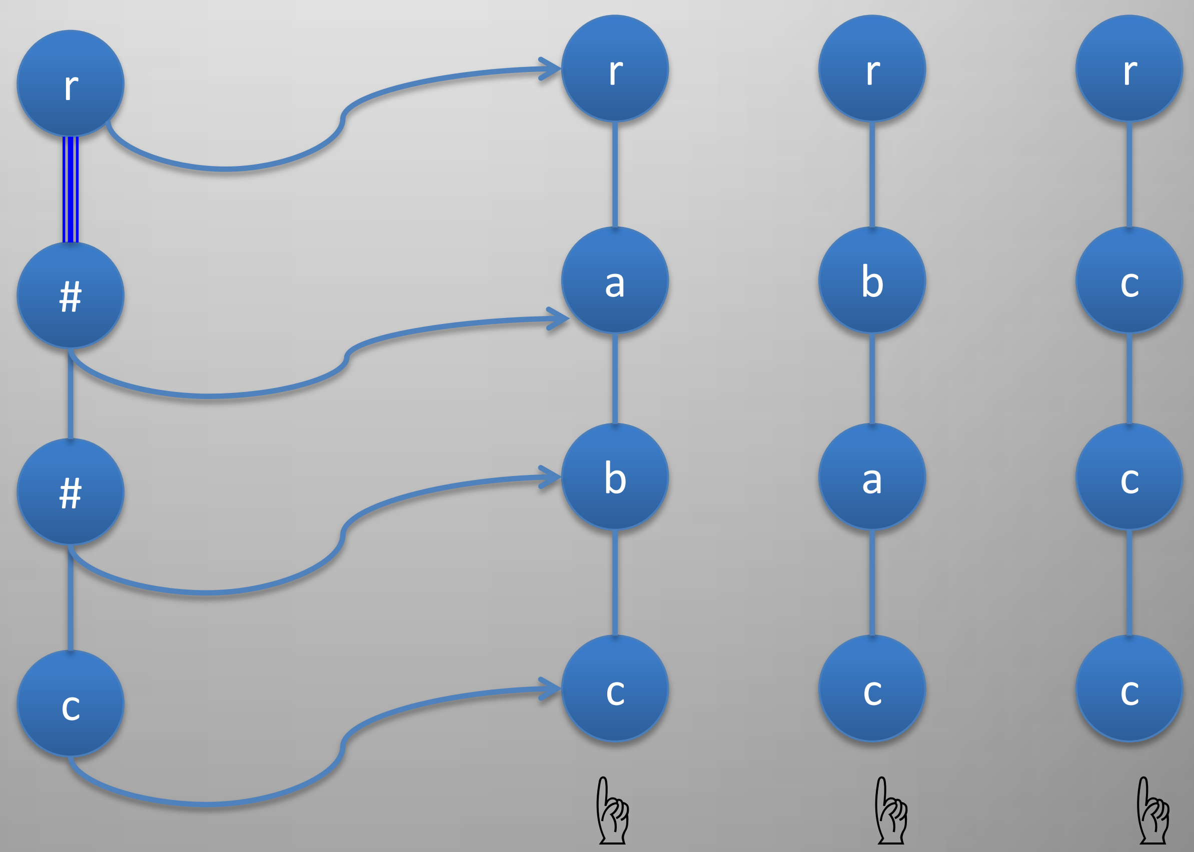
Tree pattern query – semantics

Tree pattern query



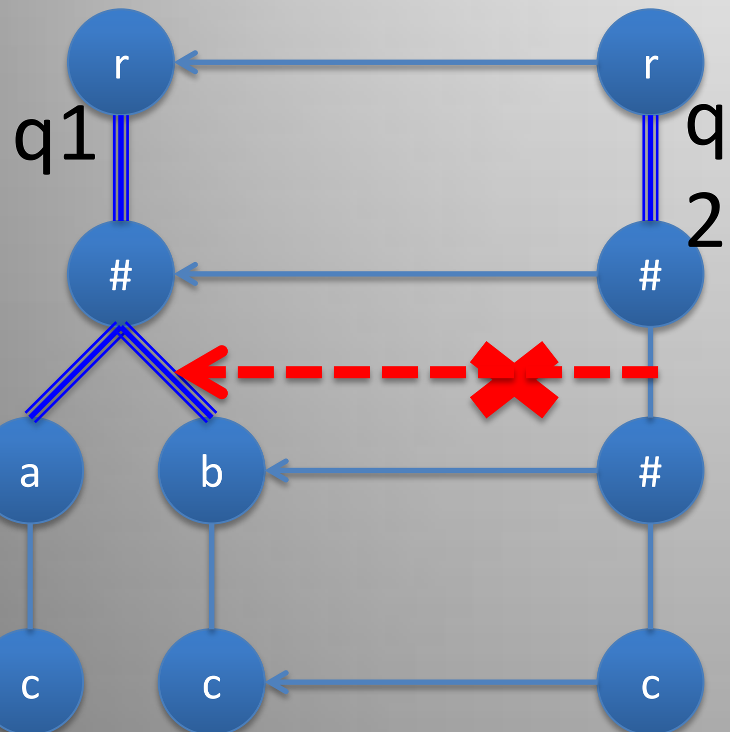
Tree pattern query – semantics

Tree pattern query



Tree pattern query containment

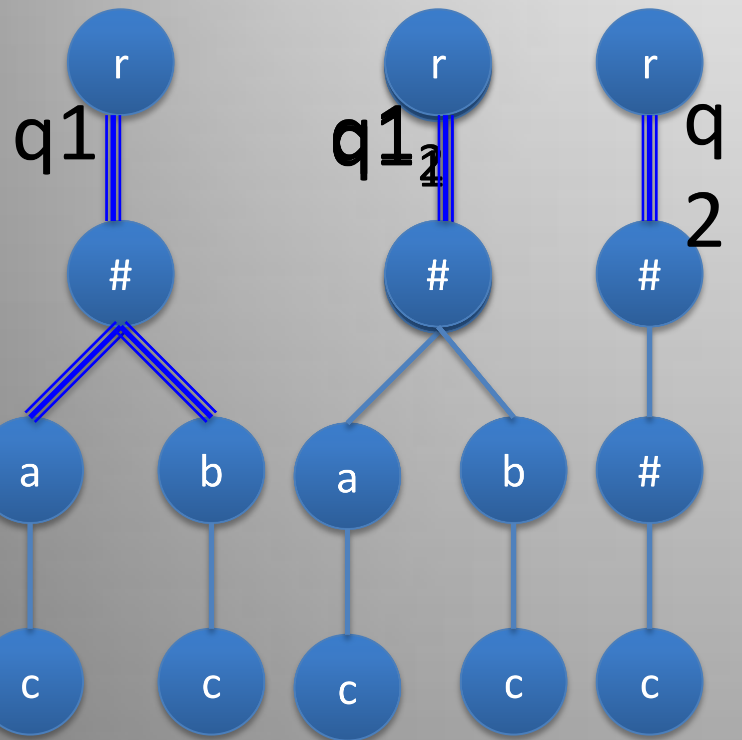
Tree pattern containment



There is no homomorphism from q2 to q1

Tree pattern query containment

Tree pattern containment



- But q1 ⊆ q2

q2 = there is a path of length at least 2 from the root r to a leaf c

q1 & the # is not an ***a***

– There is such a path

q1 & the # is not a ***b***

– There is such a path

XML storage

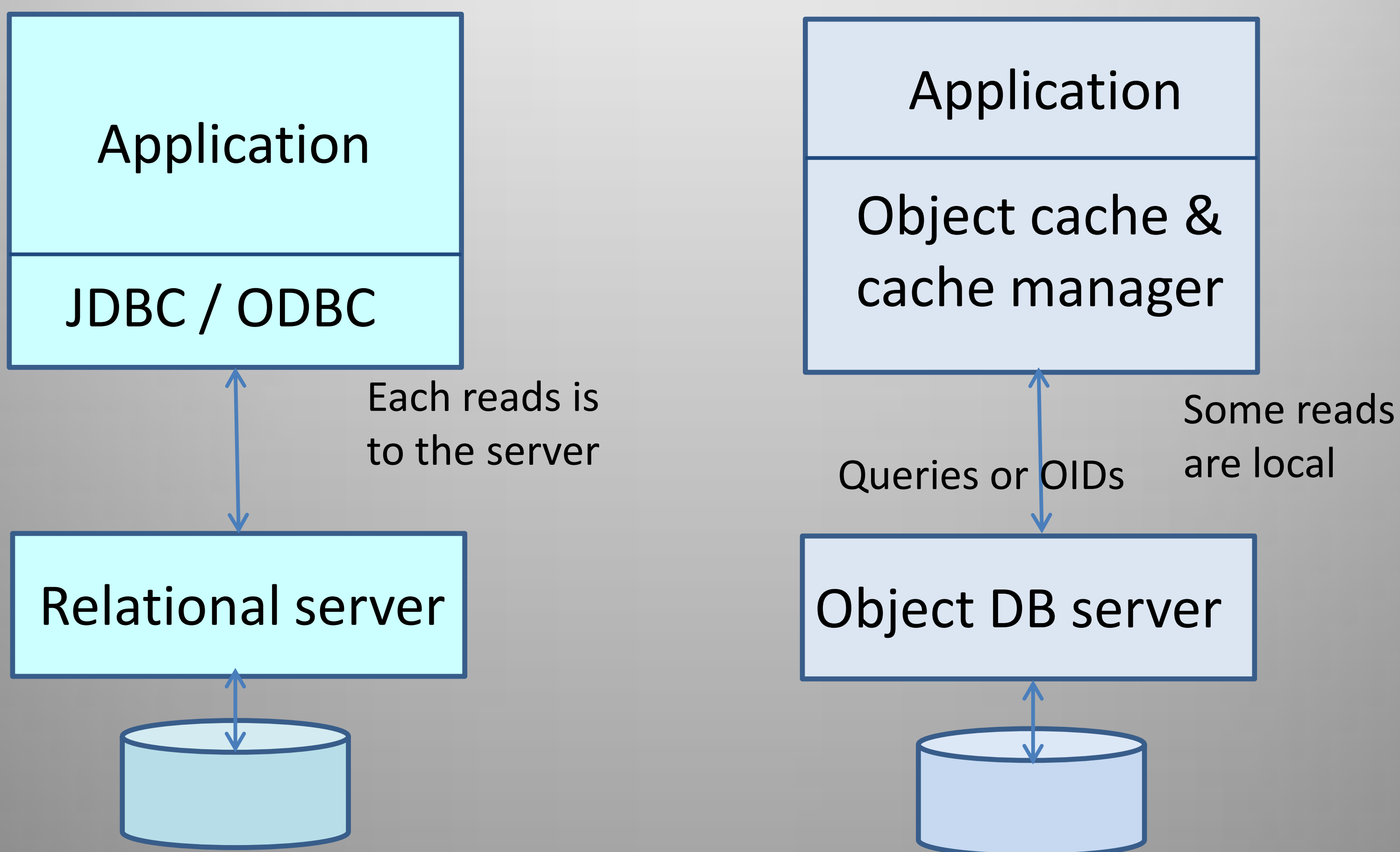
- In a file system
  - A directory is now becoming a searchable database
- In a native XML DBMS
  - eXist: open source
  - MonetDB
- In a relational DBMS
  - Blades for storing XML
- Several types of API
  - XQJ XQuery API for Java specification (XQJ)
  - XML:DB JDBC for XML databases
- Trend: reduce the separation between DBMS and file systems

Graphs and object databases

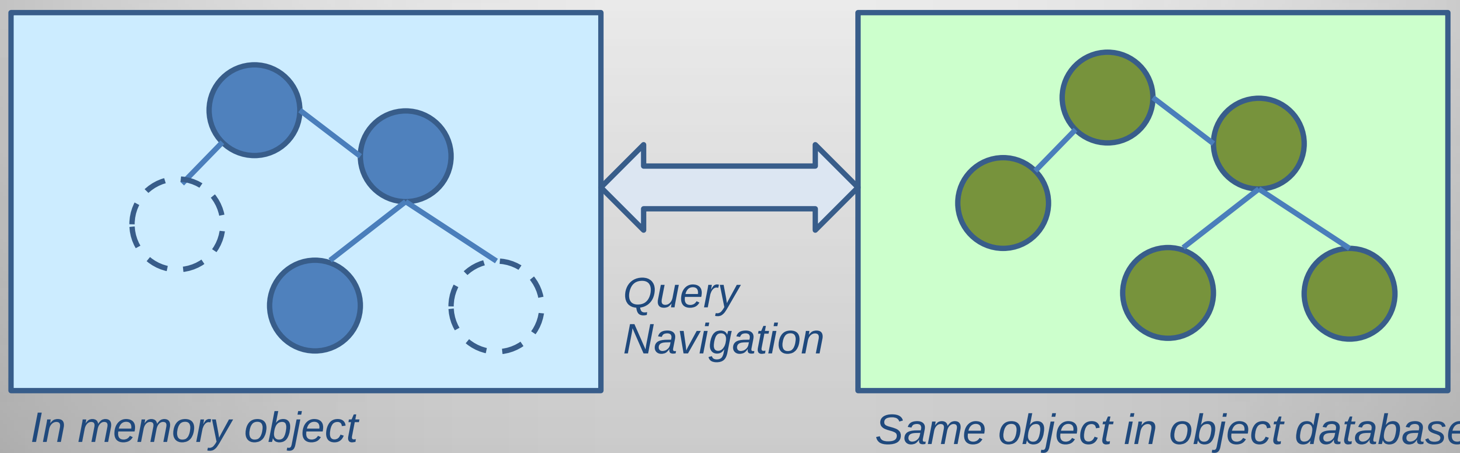
Object databases = Object-oriented languages + Databases

- Object-oriented language
  - Object = data + behavior
  - Objects encapsulate data
- Standard database features
  - Transactions
  - Queries, etc.
- Object data model
  - Object identity
  - Complex structure (typically set & tuple constructors)
  - Classes: type and class hierarchies
  - Inheritance

Architecture: relational vs. object



The same object from disc to memory



Greatly facilitates developing applications

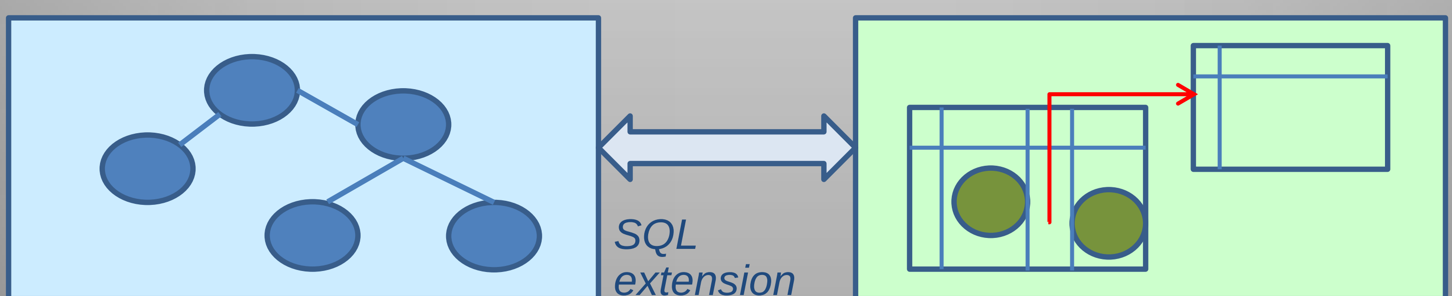
- A single data model (richer)
- Integration with an object programming language,

Performance because of complex objects

- Join between multiple tables replaced by navigation between objects
- Object often in local cache

Moderate industrial success

- Object database systems
  - 1989: Object Database Manifesto (Atkinson, Bancilhon et al)
  - Pioneers: O2, ObjectStore, Objectivity, Versant…
  - ODMG Standard, OQL
- Object-Relational
  - Dirty attempts to use relational back-ends to store objects



In memory objects

Relational data with pointers to objects

But the ideas are spreading

Standard around Java: JDO

Popular open source software such as Db4o

Frameworks for languages with persistence: JPA, DataObjects.NET

NoSQL

Motivations for NoSQL

DBMSs pay a high overhead for their universality

Avoid this overhead for very demanding applications

Major overheads to avoid:

1. Buffer Management: cache disk blocks in memory
2. Locking: for the management of concurrency.. Transactions must wait for the release of locks
3. Latching: Short term locks used for access structures that are shared as B-tree
4. Logging: Every update is written in the log that is forced to disk

Analysis of OLTP applications [Harizopoulos & AL08]:

35% buffer management	21% locking
19% latching	17% logging

Specialized data management systems

Specialized for certain types of queries

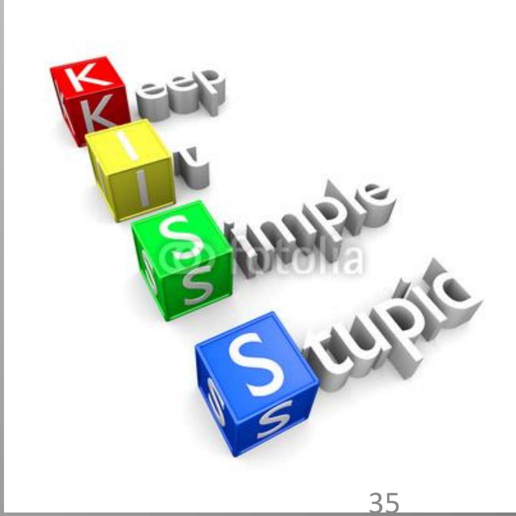
Specialized for certain aspects such as scalability

In return: sacrifice universality

- Sacrifice certain types of queries like the join
- Sacrifice some features, such as concurrency

No SQL

- Non-standard systems for data management
- Typically simpler data models
- (Support sometimes SQL)



Warning: the term NoSQL is also used sometimes for systems based on the contrary, more complex models: Object /XML / RDF – not here

NoSQL : different flavors

Extreme performance

- Massive scalability
- Massive distribution
- Total availability

Specialization

- High transaction rates
- Simple OLAP queries on very large volumes

No universality

Less independence

- No 3 levels

Less abstraction

- Not relational and SQL
- Simple Data: key / value
- Simple queries

Loss of functionality

- No ACID (strict)
- Less typing and integrity
- Simple access structures
- simplistic API - no JDBC

Examples

Key / value store with weak consistency
- Cassandra (Apache), Dynamo (Amazon)

Key / value store on disk
- **Hadoop** Hbase (Apache), BigTable (Google)

Document store with N1NF
- MongoDB (free software)

Main memory database single-threaded for OLTP
- VolTDB

Massively parallel database for analysis
- Greenplum, MySQL Cluster

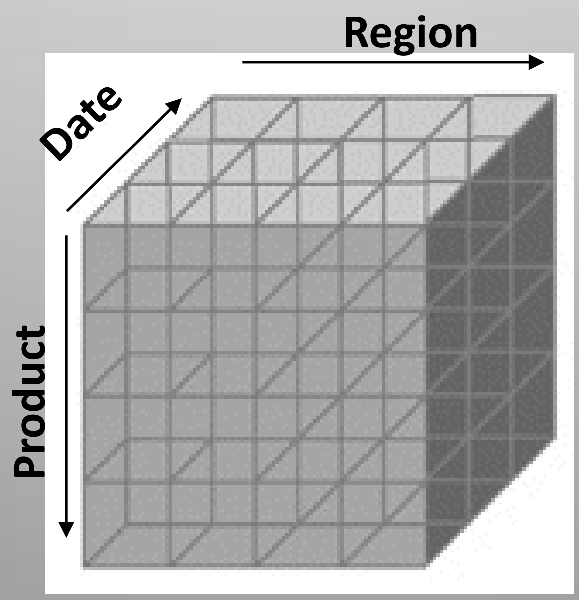
And many more …

The OLAP multidimensional model

Data get organized in cubes

March
February
January

	USA	Canada	France	UK
Pain	12	25	14	86
Fromage	23	68	45	25
Yaourt	12	95	65	42
Chocolat	44	22	33	18



\+ more dimensions:

- Kind of customer
- Kind of sale (web,
- …

Discussion

- Ted Codd 1995
- Evolution from spreadsheet
- Provide multidimensional views for analysis
  - Hierarchical domains – Time: day, week, month, year
  - Aggregation
- Example of queries
  - 5 top demography groups buying videos
  - Products sold in France where rejection rate diminished by more than 5%
- Querying, navigation, reporting

Standard query language: MDX (MSFT, 1997)

SQL

select, from, where, group-by

Yields a table (2-dim)

Select columns from some tables

Filter lines with predicates in *where* clause

Aggregation using *group by*

MDX

with, select, from, where

Yields a cube (N-dim)

Select: select cube dimensions

With: specification on selected dimensions

Where: specification on non selected dimensions

Implicit aggregation

```
with member Measures.profit as Measures.StoreSales - Measures.cost
select
   {Measures.StoreSales, Measures.Profit} on columns,
    non empty filter(Product.ProductDepartment.members,
           (Product.currentMember, Measures.StoreSales) > 20000.0) on rows
from [Sales]
where ([Time].[1997])
```

Conditional tables

Uncertainty

- Lots of uncertain data
- Studied in academia
- Not much in industry
  - Null values in SQL – Trash semantics
  - No clear standard
- We will see here in brief
  - Conditional tables
  - How to turn them probabilistic

Conditional tables & uncertainty

Friend	Location	Condition
Alice	London	E
Bob	London	$E \wedge F$
Alice	Paris	$\neg E$
Lucile	London	F

Friend	Location
Alice	London
Bob	London
Lucile	London

Friend	Location
Alice	London

Friend	Location
Alice	Paris
Lucile	London

Friend	Location
Alice	Paris

4 possible worlds

Conditional tables & probabilities

Friend	Location	Condition
Alice	London	E
Bob	London	E∧F
Alice	Paris	¬E
Lucile	London	F

E is 80%

F is 40%

Friend	Location
Alice	London
Bob	London
Lucile	London

32%

Friend	Location
Alice	London

48%

Friend	Location
Alice	Paris
Lucile	London

8%

Friend	Location
Alice	Paris

12%

A jewel of databases

The worst way I know of computing transitive closure

Calculus for complex objects

The points reachable from *a* in a graph *G*

$$\{ x \mid \forall R \, (\, (R(a) \land \forall y,z \, (R(y) \land G(y,z) \Rightarrow R(z)) \,) \Rightarrow R(x) \,) \}$$

x is reachable from *a* if $x \in R$

for each set *R* containing *a* and “closed under” *G*

Algebra for complex objects

The points reachable from *a* in a graph *G*

$D := \mathbf{\Pi}_1(G) \cup \mathbf{\Pi}_2(G)$: the nodes in *G*

$P := 2^D$: the powerset of *D*

Θ an algebraic query (in classical relational algebra) equivalent to:

$$R(a) \wedge \forall x,y \; (R(x) \wedge G(x,y) \Rightarrow R(y))$$

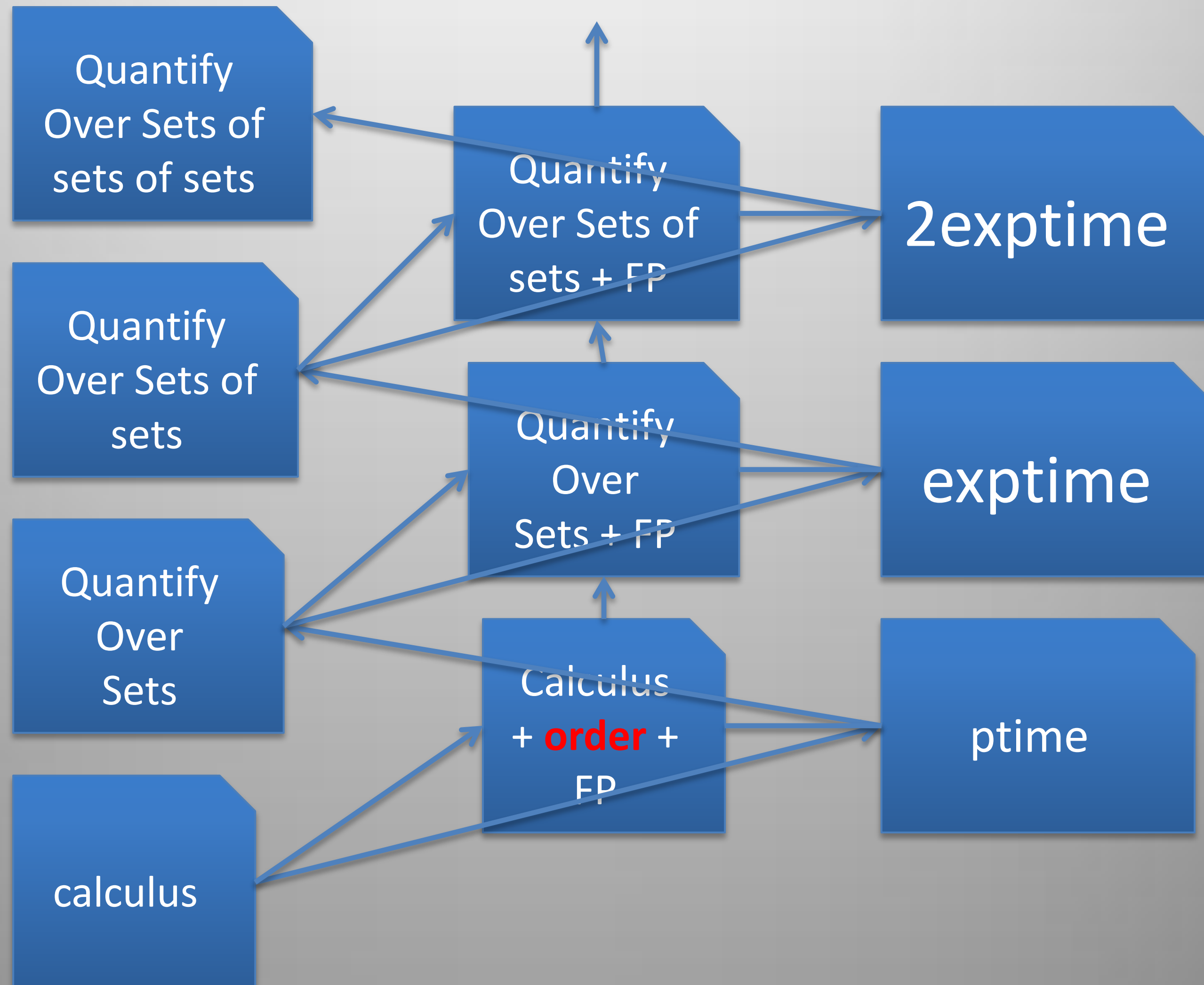
$Q := \sigma_\theta(P)$: the subsets of *D* satisfying *Θ*

$Q' := \mathbf{\Pi}_1(\sigma_{1\supset 2}(Q \times Q))$: the non-minimal elements in *Q*

$Q'' := Q - Q'$: the minimal elements in *Q* (unique)

unnest(Q'') : the points reachable from *a* in *G*

Complexity


Quantify Over Sets of sets of sets
Quantify Over Sets of sets
Quantify Over Sets
calculus
Quantify Over Sets of sets + FP
Quantify Over Sets + FP
Calculus + order + FP
2exptime
exptime
ptime

Conclusion

Conclusion

- Regain the 3 principles
  – Is this desirable?
- Build a unifying theory
  – Is this desirable?
- Develop new systems
- Develop new theories
- Consider richer semantics
  – Semantic Web: next time


Merci !
FRAPAR.