Querying and Updating Structured Files

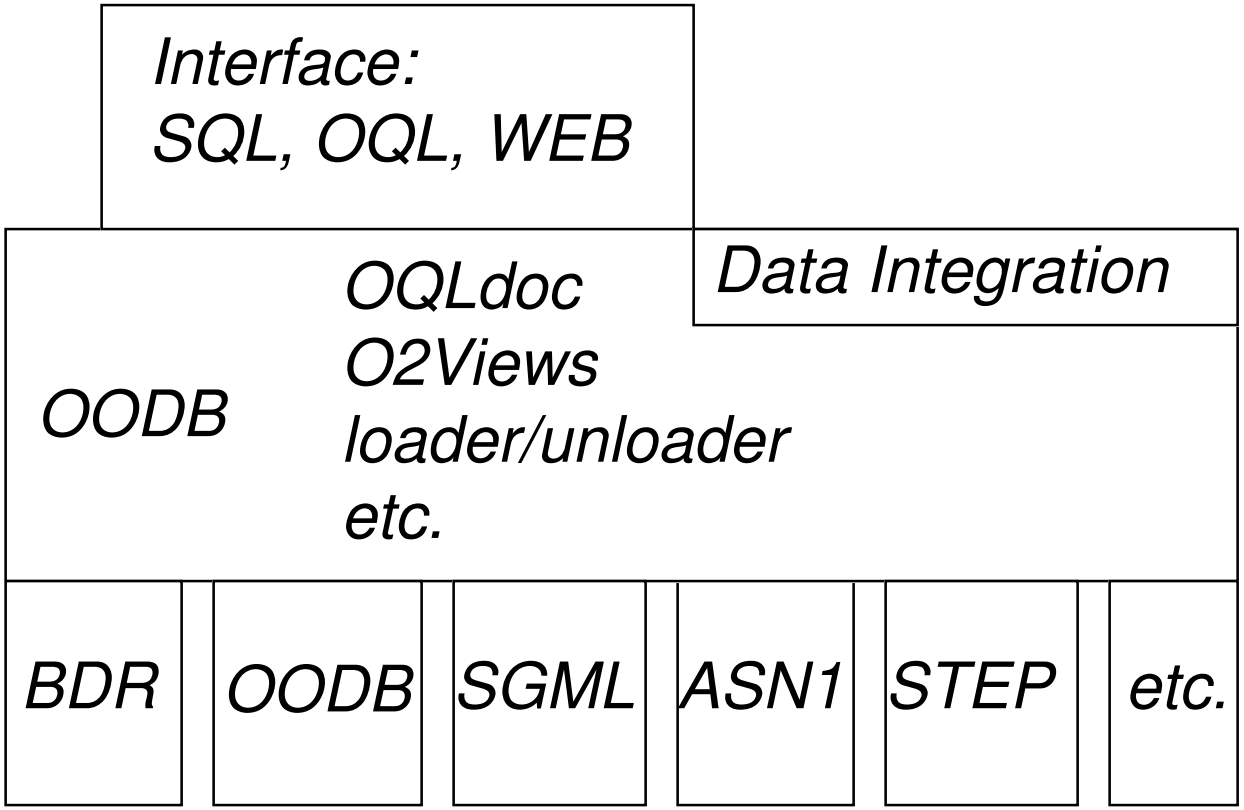
Serge Abiteboul
Stanford U. and INRIA

Joint work with
Cluet-Milo: VLDB93, Sigmod95
Christophides-Cluet: Sigmod94

Organization

- *I. Introduction*
- *II. Structuring Schema*
- *III. Query Optimization*
- *IV. Query Language*
- *V. Updates*
- *VI. Conclusion*

I. Introduction: Context



Motivations: structured data, where?

- ***Databases***
- *relational, OODB*
- *query languages*
- *secondary storage*
- *concurrency, etc.*

- ***Data Exchange Formats***
- *(e.g., sgml, step, asn1, programs, BibTex, etc.)*
- *in file systems*
- *some inner structure*

- *Generic tools (e.g., emacs) ignore structure*
- *Specific tools (e.g., SGML editors, netscape, pat) lack db features: no declarative QL*

Motivations

- *Use database tools to handle structured files -- focus here on query/update*
- *Interoperability between data stored in databases and in structured files*
- *Facilitate the design of applications with data stored in*
  - *Databases*
  - *DX Formats*

Approach

- *Object-oriented database technology*
- *Interface combining*
  - *Declarative query language ala OQL*
  - *Hypermedia technology ala Web*
  - *Information Retrieval*
- *Sophisticate view mechanism (O_2View) for the integration*

Some Issues

- *Efficient loading/unloading of data*
- *Data merging and replication*
- *Object naming in a distributed environment*
- ***Querying and updating virtual data***
- ***Query language for structured files***
- *Merging IR/DB/WEB technologies*
  - *full text index and database access methods*

Structured Files

- *Example: a BibTex file*
- @inproceedings { SW97,
  author = “J. Smith and P. William”,
  title = “Where are the objects?”,
  booktitle = pods,
  year = “1997” }

Issue: Specify mapping file ↔ db

Default Mapping

- *grammar gives inner structure*
- *database = parse tree of the file*
- *problems: lack of flexibility, difficulties to formulate queries (e.g., lists)*
- *Solution:* ***structuring schemas***

  grammar + db schema + annotations
- *[Alternative: default structuring + view lead to similar issues]*

II. Structuring Schema: Grammar

<Reference> ⇒ “@inproceedings {“
identifier “,”
“author =“ <author_set> “,”
“title =“ string “,”
“booktitle =“ string “,” ... “}”

<Ref_Set> ⇒ <Reference> <Ref_set>
| ε

Structuring Schemas: Types

<Reference>

type ref : (class)

[key:string, authors:{authors}, title...]

<Ref_Set>

type { ref }

- *Gives the type of the corresponding value*

Types in this presentation

base_type :- int | real | bool | string

τ :- base_type

| [A1: τ , ... , An: τ] % tuple

| {τ } | < τ > | {{τ }} % set/list/bag

- *Also union of types and classes/objects*

Structuring Schemas: Annotation

<Reference> ⇒ “@inproceedings {“
identifier “,”
“author =“ <author_set> “,”
“title =“ string “,”
“booktitle =“ string “,” … “}”
$$:= new(ref, [key:$1, authors:$2, ...])

<Ref_Set> ⇒
<Reference> <Ref_set> **$$:= {$1} ∪ $2**
| ε **$$:=** ∅

Actions in this presentation

- *each database constant and each* $i
  (for a1,...,an, a, a', a" *actions & correct typing)*
- {a1,...,an}, {{a1,...,an}}, <a1,...,an>, [A1:a1,...,An:an]
- append(a,a') & merge(a',a") *(for set/bag/list)*
- a1 || a2 *(string concatenation)*
- f(a1,...,an) *for some* f *in* F
  - *e.g.:* F *is relational algebra*

III. Querying Structured Files

- *Naive approach: parse/load the file; use standard query evaluation (sometimes the best)*
- *Database is a view of the file*

  load only what is needed and when it is needed

```
select  x.title
from x in references
where x.key = “GGG96”
```

Query Optimization Technique

<Reference> ⇒ “@inproceedings {“
identifier “,”
“author =“ <author_set> “,”
“title =“ string “,”
“booktitle =“ string “,” … “}”

$$:= if $1 = “GGG96”
then [title:$3] else ε

- *Avoid unnecessary construction*

Query Optimization Technique

- *Transform the action in an algebraic expression*
- *Apply rewriting technique (e.g., push selection)*
- *Some operations are not set oriented*
- *Introduce new grammatical symbols*

  e.g. <q(reference)>
- *[Minimize parsing: e.g., full text index]*

Query optimization -- issues

- *Works well for simple projection/ selection queries*
- *Also for some joins that respect the structure of the file*
- *Problems with*
  - *Cyclic data (conditional new)*
  - *Specifications with side effects*

IV. QL for Structured Files

- *Integration of technologies*
  - *queries based on structure (db style)*
  - *queries based on content (ir style)*
  - *queries based on navigation (web style)*
  - *use of object paradigme*
  - *access data without knowing its structure*
- *Advantage of OQL vs. SQL:*
  - *objects for sharing/cycles*
  - *bags, lists, tree structure*

OQL-doc (motivated by SGML appli.)

- *Union of types: same element has alternative possible structures*
- *Less important extensions (ordered tuples/&)*
- *[exceptions: e.g., footnotes]*
- *String primitives: pattern matching*
- ***Path and Attribute Variables***

OQL-doc -- continued

select t from my_article Path_P . title (t)

select tuple(A:ATT_A, B:V)
from my_article Path_P . Att_A (V)
where V contains “SGML”

- *union of types: some titles may be strings, some may be pairs, triples...*
- *the where clause restricts possible paths*

OQL-doc -- issues

- *Cyclic paths - paths that lead to "Adam"*
  - *adam's name, the name of the spouse of adam's spouse, etc.*
- *Type constraints in queries and optimization*
- *Other forms of optimization (e.g., use of full text index)*

V. Updating Structured Files

- *Two data representations*

  real one (file) -- virtual one (db)
- *The update is specified on the db*
  - *inverse important as well (e.g., genome db)*
- *The update is propagated to the file*
- *Is this the view update problem?*

  yes, but in a special setting

Updates: Naive approach

- ***1. Parse/load 2. Update 3. Unparse***
- *Unparsing: translation from a db value to a string using the SS*

  string: “John, Sally %comment{just married}”

  parsed: { oid34, oid54 }, oid34.name=John...

  unparsed: “John, Sally” *or* “Sally, John”

  or “John, Sally %comment{gfjhgfjghf}”
- *Loss of information*

Updates: This is not easy

- *Unparsing is often undecidable*
  - *E.g., for* F *containing join and projection*
  - *algo for* F= ∅ *based on pattern matching*
- *The problem of testing if a SS is lossless is undecidable (even for* F = ∅*)*
  - *One can find some large classes of lossless schemas*
  - *In most applications, the schema is lossy: sets, bags, comments, etc.*
- *Some updates cannot be propagated*

Updates: Requirements

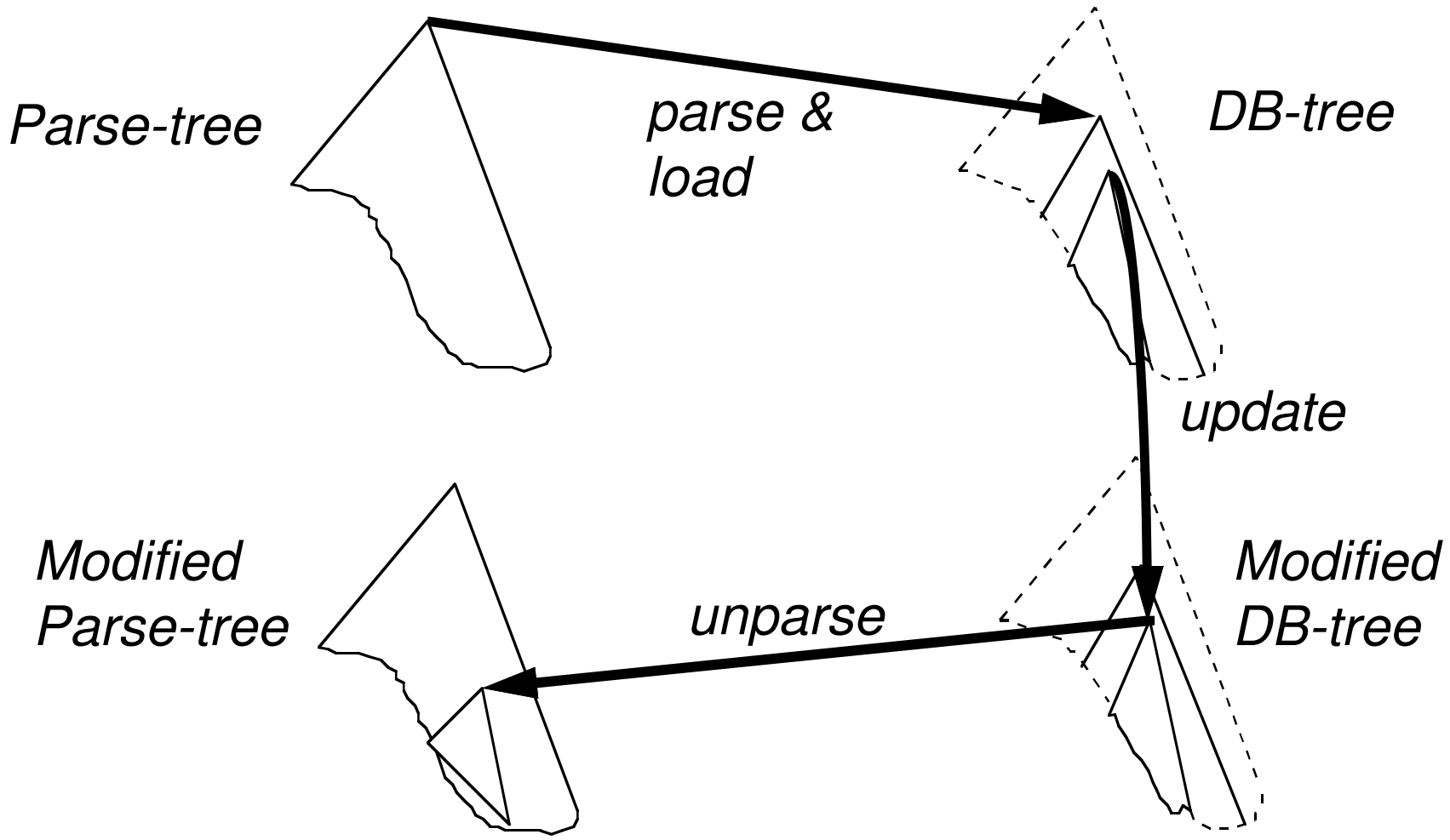
- *E.g.: insert one new author to GGG96*
- *avoid unnecessary data construction*
  - *other references should not be constructed*
- *minimize loss of information*
  - *no loss of information for other references*
  - *e.g., comments or order of author sets*

Updates: Local Update Algorithm

- *Use query optimization techniques to construct only the relevant portion of db*
- *apply the update and unparse only the part of db that changed*
- *[guided unparsing to do even better]*
- *This works only for some particular structuring schemas:*

  local structuring schemas

Updates: Local Update Algorithm



Updates: Local UpdateAlgorithm

- *Key role: a correspondance between the vertices in the parse-tree and db-tree*
- *Locality: if a db-vertex is modified, it suffices to modify the parse-tree rooted in the corresponding vertex*
- ***An efficient algo for local SS***
- *Locality is undecidable but*
- ***A large decidable class that covers the applications we considered***

Updates: A non-local SS

S :-	S1 T	| S2 T	$$:- [A:$1,B:$2]
S1 :-	T		$$:- { $1 }
S2 :-	T T		$$:- {$1, $2 }
T :-	string		$$:- $1

"Joe" "X" $\Rightarrow$ [A:{"Joe"},B:"X"]

insert "Sue" in A *: non-local SS (S2/S1)*

insert "Bob" in A *: no! constrained SS*

V. Conclusion -- I presented:

- *An approach to access data stored in structured files without loading it entirely*
- *The notion of SS*
- *A query optimization technique*
- *A query language for such data*
- *An update optimization technique that works only for particular SS*

Conclusion -- Implementation Status

within GoodStep & Opale EC projects

on top of the O_2 OODBMS

- *O_2 Views*
- *loader/unloader of SGML documents*
- *full text index et Web interface*
- *first prototype of OQL-doc*
- *ongoing work on optimization*

Conclusion -- Ongoing work

- *General framework for multiple data representations*
- *Setting: labelled trees/graphs + correspondance patterns/ graph rewriting*
- *collections: append/merge + algebraic properties (commutativity, idempotence)*
- *Derive correspondances*
- *Introduce new vertices (translation)*