Logical Approach to Physical Data Independence and Query Compilation

Introduction, Background, and Goals

David Toman

D.R. Cheriton School of Computer Science
University of Waterloo

ORGANIZATION

Lectures/Exercises

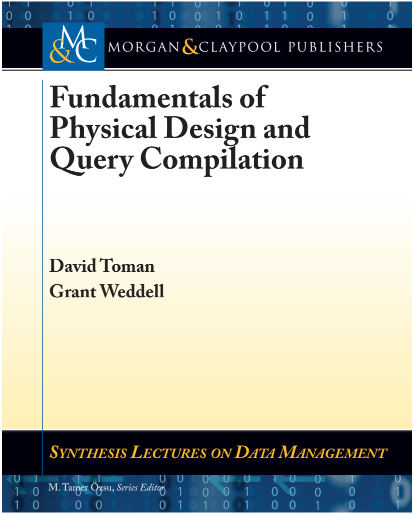
- web page:

  `http://cs.uwaterloo.ca/~david/cs848s14/`

- schedule:

  Thursday 9:30-12:19

Textbook (aka Shameless plug)



D. Toman and G. Weddell.
Fundamentals of Physical Design and Query Compilation.
Morgan and Claypool Data Management Series, 2011.

Projects

Preferred projects will be related to applying (some) of the ideas presented in this class to your own area of research: this can further your own research and may help you to consider alternative views/approaches to what you have been thinking about already.

1. project proposal: one page due Lecture 6;
2. project presentation: 10-20 minutes (depending on the number of projects) in Lecture 10;
3. report (in pdf, up to 10 pages), source code (if applicable) within a week of last Lecture.

Assessment

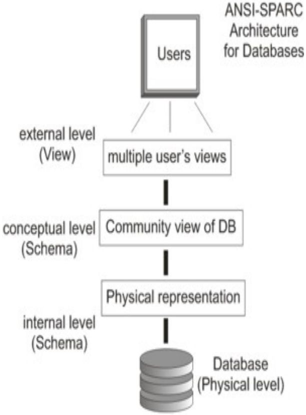
1. class participation, including assignments (15%)
2. in class presentation either of your project or of a paper from the reading list (25%)
3. project (60%)

USE SCENARIOS AND GOALS

Phyical Data Independence

IDEA:

Separate the users' view(s) of the data from the way it is physically represented.



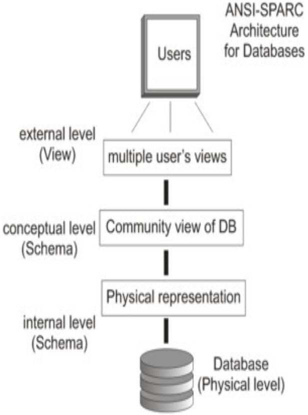
[ANSI/X3/SPARC Standards Planning and Requirements Committee, Bachman, 1975]

Phyical Data Independence

IDEA:

Separate the users' view(s) of the data from the way it is physically represented.

- independent customized user views,
- changes to conceptual structure without affecting users.



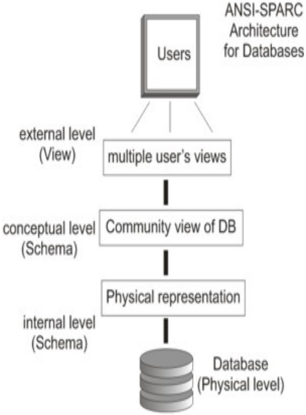
[ANSI/X3/SPARC Standards Planning and Requirements Committee, Bachman, 1975]

Phyical Data Independence

IDEA:

Separate the users' view(s) of the data from the way it is physically represented.

- independent customized user views,
- changes to conceptual structure without affecting users.
- physical storage details hidden from users,
- changes to physical storage without affecting conceptual view,

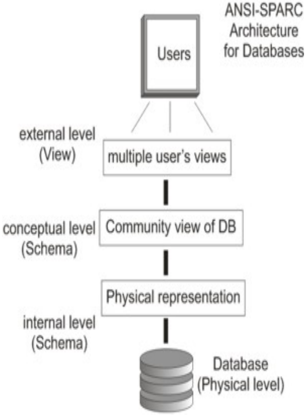


[ANSI/X3/SPARC Standards Planning and Requirements Committee, Bachman, 1975]

Phyical Data Independence

IDEA:

Separate the users' view(s) of the data from the way it is physically represented.



- physical storage details hidden from users,
- changes to physical storage without affecting conceptual view,

Originally just two levels: physical and conceptual/logical [Codd1970].

[ANSI/X3/SPARC Standards Planning and Requirements Committee, Bachman, 1975]

Example: PAYROLL

A *Conceptual (user) view* of PAYROLL data:

Example of PAYROLL data:

1. Mary is an employee.
2. Mary's employee number is 3412.
3. Mary's salary is 72000.

Example of PAYROLL:

4. There is a kind of entity called an `employee`.
5. There are attributes called `enumber`, `name` and `salary`.
6. Each employee entity has attributes `enumber`, `name` and `salary`.
7. Employees are identified by their `enumber`.

Example: PAYROLL

A *physical design* for PAYROLL:

8. There is a file of records called `emp-file`.
9. There are record fields `emp-num`, `emp-name` and `emp-salary`.
10. Each `emp-file` record has the fields `emp-num`, `emp-name` and `emp-salary`.
11. File `emp-file` is organized as a B-tree data structure that supports an `emp-lookup` operation, given a value for attribute `enumber`.

12. Records in file `emp-file` correspond one-to-one to `employee` entities.
13. Record fields in file `emp-file` encode the corresponding attribute values for `employee` entities, for example, `emp-num` encodes an `enumber`.

Ontology-based Data Access

IDEA:

Queries are answered not only w.r.t. *explicit data* but also w.r.t. *background knowledge*

$\Rightarrow$ Ontology-based Data Access (OBDA)

Ontology-based Data Access

IDEA:

Queries are answered not only w.r.t. *explicit data* but also w.r.t. *background knowledge*

$\Rightarrow$ Ontology-based Data Access (OBDA)

Example

- Socrates is a MAN (explicit data)
- Every MAN is MORTAL (background)

List all MORTALs $\Rightarrow$ {Socrates} (query)

Ontology-based Data Access

IDEA:

Queries are answered not only w.r.t. *explicit data* but also w.r.t. *background knowledge*

⇒ Ontology-based Data Access (OBDA)

Example

- Socrates is a MAN (explicit data)
- Every MAN is MORTAL (background)

List all MORTALs ⇒ {Socrates} (query)

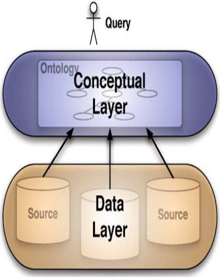


Fig. 1. Ontology-based data access.

[Calvanese et al.: Mastro]

Ontology-based Data Access

IDEA:

Queries are answered not only w.r.t. *explicit data* but also w.r.t. *background knowledge*

$\Rightarrow$ Ontology-based Data Access (OBDA)

Example

- Socrates is a MAN (explicit data)
- Every MAN is MORTAL (background)

List all MORTALs $\Rightarrow$ {Socrates} (query)

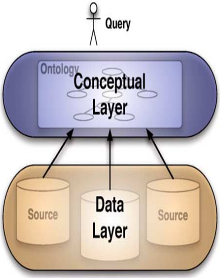


Fig. 1. Ontology-based data access.

[Calvanese et al.: Mastro]

Question:

Is Aristoteles a MORTAL?

Ontology-based Data Access

IDEA:

Queries are answered not only w.r.t. *explicit data* but also w.r.t. *background knowledge*

⇒ Ontology-based Data Access (OBDA)

Example

- Socrates is a MAN (explicit data)
- Every MAN is MORTAL (background)

List all MORTALs ⇒ {Socrates} (query)

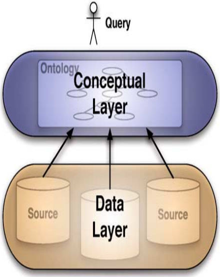


Fig. 1. Ontology-based data access.

[Calvanese et al.: Mastro]

Question:

Is Aristoteles a MORTAL? . . . can we *really* say "NO"?

Data Exchange

PROBLEM:

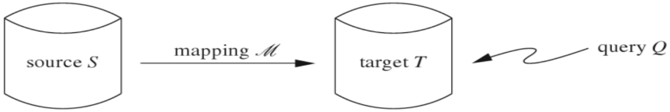
How to transfer (reformat) data conforming to a *source schema* to
data conforming to a *target schema*?

Data Exchange

PROBLEM:

How to transfer (reformat) data conforming to a *source schema* to data conforming to a *target schema*?

The general setting of data exchange is this:



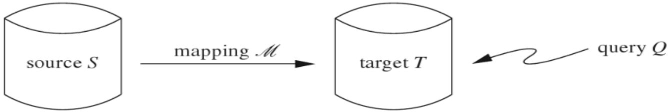
[Arenas et al: Foundations of Data Exchange]

Data Exchange

PROBLEM:

How to transfer (reformat) data conforming to a *source schema* to data conforming to a *target schema*?

The general setting of data exchange is this:



[Arenas et al: Foundations of Data Exchange]

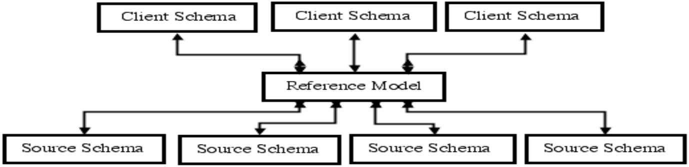
Issues:

- what should happen when the *target* is more complex than the *source*?
- how do we answer queries over the target?

Information Integration

IDEA:

Data integration provides a uniform access to a set of data sources, through a unified representation called global schema. A mapping specifies the relationship between the global schema and the sources.

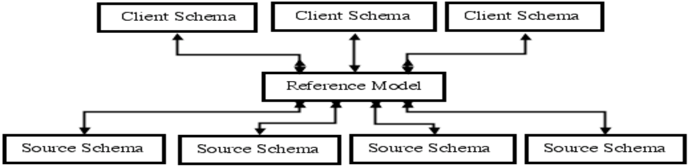


[Genesereth: Data Integration]

Information Integration

IDEA:

Data integration provides a uniform access to a set of data sources, through a unified representation called global schema. A mapping specifies the relationship between the global schema and the sources.



[Genesereth: Data Integration]

Variants "which way do the arrows point" [Lenzerini]

GAV (global as a view), LAV (local as a view), and GLAV ("both ways").

Common Threads and Issues

- In general two *schemas*: Conceptual/Logical and Physical

  ⇒ both endowed with *metadata* (vocabulary, . . .)

  ⇒ mappings connect the schemas

  ⇒ (source) data only “in” the *physical* schema

  ⇒ queries only over the *conceptual/logical* schema

Common Threads and Issues

- In general two *schemas*: Conceptual/Logical and Physical

  ⇒ both endowed with *metadata* (vocabulary, . . .)

  ⇒ mappings connect the schemas

  ⇒ (source) data only “in” the *physical* schema

  ⇒ queries only over the *conceptual/logical* schema

- Issues to be formalized/fixed:
  1. Formal description of the two schemas (same formalism for both?)
  2. Language(s) for metadata and mappings
  3. (user level) Data representation
  4. (user level) Query language (semantics–aka when is an answer an answer?)

Common Threads and Issues

- In general two *schemas*: Conceptual/Logical and Physical

  $\Rightarrow$ both endowed with *metadata* (vocabulary, ...)

  $\Rightarrow$ mappings connect the schemas

  $\Rightarrow$ (source) data only "in" the *physical* schema

  $\Rightarrow$ queries only over the *conceptual/logical* schema

- Issues to be formalized/fixed:
  1. Formal description of the two schemas (same formalism for both?)
  2. Language(s) for metadata and mappings
  3. (user level) Data representation
  4. (user level) Query language (semantics–aka when is an answer an answer?)
  5. Algorithms/Execution model for queries: e.g., does *materialization* matter?

Phyical Data Independence: My Motivation

Goal: Application of the Ideas to Embedded Systems

1. High-level conceptual view of the system
2. High level query (and, eventually, update) language
3. Fine-grained physical schema description
4. Flexible conceptual-physical mappings
5. Queries (updates) *compiled* to operations on physical level

Phyical Data Independence: My Motivation

Goal: Application of the Ideas to Embedded Systems

1. High-level conceptual view of the system [relational]
2. High level query (and, eventually, update) language [SQL]
3. Fine-grained physical schema description [records, pointers, . . .]
4. Flexible conceptual-physical mappings
5. Queries (updates) *compiled* to operations on physical level [pointer navigation, field extraction, conditionals, . . .]

Phyical Data Independence: My Motivation

Goal: Application of the Ideas to Embedded Systems

1. High-level conceptual view of the system [relational]
2. High level query (and, eventually, update) language [SQL]
3. Fine-grained physical schema description [records, pointers, . . .]
4. Flexible conceptual-physical mappings
5. Queries (updates) *compiled* to operations on physical level [pointer navigation, field extraction, conditionals, . . .]

Challenge

The code generated from queries *must be competitive* with hand-written code.

LINUX-INFO System: Conceptual View

Example of LINUX-INFO data:

1. process (called) `gcc` is running;
2. `gcc`'s process number is 1234;
3. the user (id) running `gcc` is 145;
4. `gcc` uses files "`foo.c`" and "`foo.o`".

LINUX-INFO System: Conceptual View

Example of LINUX-INFO data:

1. process (called) `gcc` is running;
2. `gcc`'s process number is 1234;
3. the user (id) running `gcc` is 145;
4. `gcc` uses files "`foo.c`" and "`foo.o`".

Example of LINUX-INFO metadata:

5. There entities called `process` and `file`.
6. There are attributes called `pno`, `pname`, `uname`, and `fname`.
7. Each process entity has attributes `pno`, `pname` and `uname`.
8. Each file entity has attribute `fname`.
9. Processes are identified by their `pno`.
10. Files are identified by their `fname`.
11. There is a relationship `uses` between processes and files.

The LINUX-INFO System: Physical Design

A *physical design* for LINUX (selected by Linus Torvalds).

12. There are process records called `task-struct`.
13. Each `task-struct` record has record fields `pid`, `uid`, `comm`, and `fds`.
14. All `task-struct`s is organized as a tree data structure.
15. The `task-struct` records correspond one-to-one to `process` entities.
16. Record fields in `task-struct` encode the corresponding attribute values for `process` entities, for example, `pid` encodes an `pno`, etc.
17. Similarly, `fs`s correspond appropriately to (open) `file` entities.
18. `fds` field of `task-struct` is an array of `fd`s; a non-null entry in this array indicates that the `process` corresponding to this `task-struct` is using the `file` identified by the `name` field of the `fd` record in the array.

Back to Desiderata

- User Query:

  find all `files` used by `process`es invoked by user 145.

Back to Desiderata

- User Query:

  find all `files` used by `process`es invoked by user 145.

- Query Plan:

  for each `task-struct` *t* in tree of `task-struct`s

  check if *t*'s `uid` field is 145 and, if so

  scan the `fds` array in *t* and

  if the file descriptor (`fd`) is non-NULL

  print out the `name` of file field in `fd`.

LINUX-INFO System: Queries and Query Plans

Back to Desiderata

- User Query:

  find all `files` used by `process`es invoked by user 145.

- Query Plan:

  for each `task-struct` *t* in tree of `task-struct`s
  check if *t*'s `uid` field is 145 and, if so
  scan the `fds` array in *t* and
  if the file descriptor (`fd`) is non-NULL
  print out the `name` of file field in `fd`.

Is the plan correct?

… and how do/can we answer this question?

Unifying Logic-based Approach

Metadata and Signatures

Vocabularies: Relational Model for both Conceptual and Physical Schemata.

Conceptual/Logical (S_L):

predicate symbols $R_1/a_1, \ldots, R_k/a_k$ (a_i is the *arity* of R_i)

(possibly) constants $c_1, \ldots, c_n$

Metadata and Signatures

Vocabularies: Relational Model for both Conceptual and Physical Schemata.

Conceptual/Logical (S_L):

predicate symbols $R_1/a_1, \ldots, R_k/a_k$ (a_i is the *arity* of R_i)
(possibly) constants $c_1, \ldots, c_n$

Physical (S_P):

predicate symbols $S_1/b_1, \ldots, S_k/b_k$
a distinguished subset $S_A \subseteq S_P$ of *access paths*

- ⇒ denote *capabilities to retrieve tuples* (i.e., data structures)
- ⇒ (optionally) binding patterns (restrictions on tuple retrieval)
- ⇒ associated with set of *tuples* (closed-world semantics)

Metadata and Signatures

Vocabularies: Relational Model for both Conceptual and Physical Schemata.

Conceptual/Logical (S_L):

predicate symbols $R_1/a_1, \ldots, R_k/a_k$ (a_i is the *arity* of R_i)
(possibly) constants $c_1, \ldots, c_n$

Physical (S_P):

predicate symbols $S_1/b_1, \ldots, S_k/b_k$
a distinguished subset $S_A \subseteq S_P$ of *access paths*

- ⇒ denote *capabilities to retrieve tuples* (i.e., data structures)
- ⇒ (optionally) binding patterns (restrictions on tuple retrieval)
- ⇒ associated with set of *tuples* (closed-world semantics)

... a standard way of defining interpretations

University of Waterloo

Metadata and Constraints

Metadata: First-order sentences Σ over $S_L \cup S_P$.

Conceptual/Logical (Σ_L):

$\Rightarrow$ keys, inclusion dependencies, hierarchies, ...

Metadata and Constraints

Metadata: First-order sentences Σ over $S_L \cup S_P$.

Conceptual/Logical (Σ_L):

$\Rightarrow$ keys, inclusion dependencies, hierarchies, ...

Physical (Σ_P):

$\Rightarrow$ keys, inclusion dependencies, hierarchies, ...

$\Rightarrow$ formulae that link to symbols in S_L (mapping constraints).

Metadata and Constraints

Metadata: First-order sentences Σ over $S_L \cup S_P$.

Conceptual/Logical (Σ_L):

$\Rightarrow$ keys, inclusion dependencies, hierarchies, ...

Physical (Σ_P):

$\Rightarrow$ keys, inclusion dependencies, hierarchies, ...
$\Rightarrow$ formulae that link to symbols in S_L (mapping constraints).

... we resort to fragments of FOL to gain better computational properties

Example: LINUX-INFO

Conceptual/Logical:

$$S_L = \{\, \texttt{process}/3, \texttt{file}/1, \texttt{uses}/2 \,\}$$

$$\Sigma_L = \{\, \texttt{process}(x, y_1, z_1) \wedge \texttt{process}(x, y_2, z_2) \rightarrow y_1 = y_2 \wedge z_1 = z_2, \\ \texttt{uses}(x, y) \rightarrow \exists z, w.\texttt{process}(x, z, w) \wedge \texttt{file}(y), \quad \ldots \quad \}$$

Physical:

$$S_A = \{\texttt{task\_struct}/1/0, \texttt{pid}/2/1, \texttt{uid}/2/1, \texttt{fds}/2/1, \texttt{fname}/2/1\}$$

$$\Sigma_P = \{\, \texttt{task\_struct}(x) \rightarrow \exists y, z, w.\texttt{pid}(x, y) \wedge \texttt{uid}(z) \wedge \texttt{fds}(x, w) \\ \texttt{pid}(x_1, y) \wedge \texttt{pid}(x_2, y) \rightarrow x_1 = x_1 \\ \texttt{process}(x, y, z) \rightarrow \exists t.\texttt{task\_struct}(t) \wedge \texttt{pid}(t, x), \quad \ldots \quad \}$$

Queries and Answers

Queries: First-order formulae (φ) over S_L.

$$\Rightarrow \exists p, n, u.\texttt{process}(p, n, u) \land u = 145 \land \texttt{uses}(p, f) \land \texttt{file}(f)$$

Queries and Answers

Queries: First-order formulae (φ) over S_L.

$$\Rightarrow \exists p, n, u.\texttt{process}(p, n, u) \wedge u = 145 \wedge \texttt{uses}(p, f) \wedge \texttt{file}(f)$$

Data D:

Sets of (ground) tuples that *fix* meaning of every access path.

Queries and Answers

Queries: First-order formulae (φ) over S_L.

$$\Rightarrow \exists p, n, u.\texttt{process}(p, n, u) \wedge u = 145 \wedge \texttt{uses}(p, f) \wedge \texttt{file}(f)$$

Data D:

Sets of (ground) tuples that *fix* meaning of every access path.

Query Answers:

answers *in common* when evaluating φ over *every* interpretation (database) that is a model of Σ and that extend D.

Queries and Answers

Queries: First-order formulae (φ) over $\mathrm{S_L}$.

$$\Rightarrow \exists p, n, u.\texttt{process}(p, n, u) \wedge u = 145 \wedge \texttt{uses}(p, f) \wedge \texttt{file}(f)$$

Data D:

Sets of (ground) tuples that *fix* meaning of every access path.

Query Answers:

answers *in common* when evaluating φ over *every* interpretation (database) that is a model of Σ and that extend D.

Definition (Certain Answers)

$$\begin{aligned} \mathrm{cert}_{\Sigma,D}(\varphi) &= \{\vec{a} \mid \Sigma \cup D \models \varphi(\vec{a})\} && \text{logical implication} \\ &= \textstyle\bigcap_{I \models \Sigma \cup D} \{\vec{a} \mid I \models \varphi(\vec{a})\} && \text{answer in every model} \end{aligned}$$

The BAD News (and what can be done)

Theorem

"$\vec{a} \in \text{cert}_{\Sigma,D}(\varphi)$? " is undecidable.

$\Rightarrow$ sources of undecidability: both Σ and φ!

The BAD News (and what can be done)

Theorem

"$\vec{a} \in \text{cert}_{\Sigma,D}(\varphi)$?" is undecidable.

$\Rightarrow$ sources of undecidability: both Σ and φ!

Standard solution:

1. restrict Σ to decidable fragments of FOL (e.g., DLs)
2. restrict φ to a decidable fragment of FOL (e.g., UCQ)

The BAD News (and what can be done)

Theorem

"$\vec{a} \in \text{cert}_{\Sigma,D}(\varphi)$?" is undecidable.

$\Rightarrow$ sources of undecidability: both Σ and φ!

Standard solution:

1. restrict Σ to decidable fragments of FOL (e.g., DLs)
2. restrict φ to a decidable fragment of FOL (e.g., UCQ)

	S_L, Σ_L	S_P, Σ_P	queries
OBDA	(lite) TBox	ABox	CQ/UCQ
Data Exchange	target, target deps	source, st-tgds	CQ/UCQ
Information Integration	global view	local view, $\{G\|L\}$AV	CQ/UCQ

What do Relational Systems do??

IDEA: “make it look like a single model”

(severely) restrict what logical schema may look like:

every logical predicate $P(\vec{x})$ must correspond 1-1 to *some* access path.

... conceptual/logical symbols in queries *are* (mere aliases of) access paths.

... completely against the idea of *physical data independence*.

What do Relational Systems do??

IDEA: “make it look like a single model”

(severely) restrict what logical schema may look like:

every logical predicate $P(\vec{x})$ must correspond 1-1 to *some* access path.

... conceptual/logical symbols in queries *are* (mere aliases of) access paths.

... completely against the idea of *physical data independence*.

Is this enough?

What do Relational Systems do??

IDEA: “make it look like a single model”

(severely) restrict what logical schema may look like:

every logical predicate $P(\vec{x})$ must correspond 1-1 to *some* access path.

... conceptual/logical symbols in queries *are* (mere aliases of) access paths.

... completely against the idea of *physical data independence*.

Is this enough? $\neg P(x)$? $\forall x.P(x)$?

What do Relational Systems do??

IDEA: “make it look like a single model”

(severely) restrict what logical schema may look like:

every logical predicate $P(\vec{x})$ must correspond 1-1 to *some* access path.

... conceptual/logical symbols in queries *are* (mere aliases of) access paths.
... completely against the idea of *physical data independence*.

Is this enough? $\neg P(x)$? $\forall x.P(x)$? ... depend on the *domain* of the model

What do Relational Systems do??

IDEA: "make it look like a single model"

(severely) restrict what logical schema may look like:

every logical predicate $P(\vec{x})$ must correspond 1-1 to *some* access path.

... conceptual/logical symbols in queries *are* (mere aliases of) access paths.
... completely against the idea of *physical data independence*.

IDEA-2: "only queries that think there is a single model"

A formula φ is *domain independent* if for all pairs of models I_1, I_2 of D and valuation θ we have

$$I_1, \theta \models \varphi \text{ if and only if } I_2, \theta \models \varphi.$$

... I_1 and I_2 can only differ in their *domains* (hence the name).

A LOGSPACE Algorithm

IDEA

Domain independent formulae can be evaluated in a model based on the *active domain of D* (set of individuals that appear in the access paths).

A LOGSPACE Algorithm

IDEA

Domain independent formulae can be evaluated in a model based on the *active domain of D* (set of individuals that appear in the access paths).

. . . active domain of *D* is a *finite set*.

A LOGSPACE Algorithm

IDEA

Domain independent formulae can be evaluated in a model based on the *active domain of D* (set of individuals that appear in the access paths).

. . . active domain of D is a *finite set*.

A Turing machine T_φ

- read only input tape storing (an encoding of) $\vec{a}$ and D;
- read/write work tape storing a *counter* for each variable in φ ($\log|D|$ bits) and fixed number of auxiliary counters;
- a finite control that *implements* top-down satisfaction check w.r.t. a valuation defined by the current state of the counters

$\Rightarrow$ used as pointers to individuals on the work tape.

A LOGSPACE Algorithm

IDEA

Domain independent formulae can be evaluated in a model based on the *active domain of D* (set of individuals that appear in the access paths).

... active domain of D is a *finite set*.

A Turing machine T_φ

- read only input tape storing (an encoding of) $\vec{a}$ and D;
- read/write work tape storing a *counter* for each variable in φ ($\log |D|$ bits) and fixed number of auxiliary counters;
- a finite control that *implements* top-down satisfaction check w.r.t. a valuation defined by the current state of the counters

$\Rightarrow$ used as pointers to individuals on the work tape.

Theorem

$$\mathrm{cert}_{\Sigma,D}(\varphi) = \{\vec{a} \mid \langle \vec{a}, D \rangle \in \mathcal{L}(T_\varphi)\}.$$

Range-restricted Formulas and Relational Algebra

Nobody uses that algorithm!

Range-restricted Formulas and Relational Algebra

Nobody uses that algorithm! Instead:

Range-restricted Formulae (queries):

$$\varphi ::= R(\vec{x}) \mid \varphi \wedge x = y \mid \varphi \wedge \varphi \mid \exists s.\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \neg\varphi$$

Bottom-up "Algebraic" Query Evaluation:

every production above maps (at least naively) to a algebraic operation on finite relations:

- scan (with renaming),
- selection,
- join,
- projection,
- union, and
- difference.

Range-restricted Formulas and Relational Algebra

Nobody uses that algorithm! Instead:

Range-restricted Formulae (queries):

$$\varphi ::= R(\vec{x}) \mid \varphi \land x = y \mid \varphi \land \varphi \mid \exists s.\varphi \mid \varphi \lor \varphi \mid \varphi \land \neg\varphi$$

Bottom-up "Algebraic" Query Evaluation:

every production above maps (at least naively) to a algebraic operation on finite relations:

- scan (with renaming),
- selection,
- join,
- projection,
- union, and
- difference.

Datalog (limited iteration)

additional predicates defined as a fixpoint positive query allows PTIME-complete problems.

Summary

- comprehensive framework based on certain answers that unifies many database/KR approaches to handling information in presence of background information/theory/ontology;
- too expressive and in turn computationally in-feasible;
- practical (relational) systems: (almost) trivial instance of the framework.

Summary

- comprehensive framework based on certain answers that unifies many database/KR approaches to handling information in presence of background information/theory/ontology;
- too expressive and in turn computationally in-feasible;
- practical (relational) systems: (almost) trivial instance of the framework.

Plan of Lectures:

1. Classical OBDA: another way of gaining tractability (and its limits)
2. Database Approach Extension and Interpolation
3. Modeling Complex Physical Designs
4. Updates of Data and Future Directions