QUERYING LOGICAL DATABASES

Moshe Y. Vardi

IBM Almaden Research Center

PHYSICAL VS. LOGICAL DATABASES

Physical Database: a complete description of the world.

Logical Database: a complete description of our knowledge of the world, a partial description of the world.

ADVANTAGES OF LOGICAL DATABASES

- Uniformity in treating data, integrity constraints, and derivation rules.
- Ability to model incomplete information.

EXAMPLES

- ***Data***:

  Teaches(Plato, Aristotle)

- ***Integrity constraints***:

  $R(x, y) \wedge R(x, z) \rightarrow y = z$

- ***Derivation rules***:

  $R(x, y) \wedge R(y, z) \rightarrow R(x, z)$

- ***Incomplete information***:

Teaches(Hopcroft, Aho)$\vee$Teaches(Aho, Hopcroft)

Relational Vocabulary L: a finite set C of constant symbols and a finite set R of relation symbols.

Interpretation of L: a nonempty finite domain D, assigment of elements of D to constant symbols in C, assigmment of relations over D to relation symbols in R.

Theory T *in* L: a set of sentences in the vocabulary L.

Physical Database: (L,I) - L vocabulary, I interpretation of L.

Logical Database: (L,T) - L vocabulary, T theory in L.

(L,I) *model* of (L,T) if I satisfies T.

Queries

Query in L: $\mathbf{x}.\phi(\mathbf{x})$, where ϕ - formula in L, $\mathbf{x}$ - sequence of distinct variables containing all free variables of ϕ.

Let $PB=\{L,I\}$ and $Q=\mathbf{x}.\phi(\mathbf{x})$, $|\mathbf{x}|=k$.

$$Q(PB)=\{\mathbf{d}\in D^k : I \text{ satisfies } \phi(\mathbf{d})\}.$$

Let $LB=\{L,T\}$ and $Q=\mathbf{x}.\phi(\mathbf{x})$, $|\mathbf{x}|=k$.

$$Q(LB)=\{\mathbf{c}\in C^k : T\models_f \phi(\mathbf{c})\}.$$

Closed World Assumption

Motivation:

$$|Negative\ Facts.| >>>> |PositiveFacts|$$

Default Assumption: Everything is **false** unless explcitly stated otherwise.

Example: In conventional databases only positive facts are mentioned.

Closed-World Databases

First-order theories with 4 components:

(1) *Atomic Facts* model physical tuples, e.g.,

$Teaches(Socrates,Plato)$.

(2) *Uniqueness Axioms* model our knowledge about identity of elements, e.g.,

$\neg(Socrates{=}Plato)$, but not

$\neg(Jack\ the\ Ripper{=}Benjamin\ D'\ Israeli)$.

(3) *Domain Closure Axiom*:

$$(\forall x)(x = c_1 \vee, \ldots, \vee c_m),$$

where $C = \{c_1, \ldots, c_m\}$.

(4) *Completion Axioms*: If $P(\mathbf{c}_1), \ldots, P(\mathbf{c}_n)$ are all the atomic facts about P, then we have

$$(\forall \mathbf{x})(P(\mathbf{x}) \rightarrow \mathbf{x} = \mathbf{c}_1 \vee, \ldots, \vee \mathbf{x} = \mathbf{c}_n).$$

AN EXAMPLE

Let the database scheme consist of a relation R(EMP, SAL). Let the physical database be

EMP	SAL
Gauss	60K
Turing	ϵ

THE LOGICAL DATABASE

1. ***Fact axioms***:

   $R(\text{Gauss}, 60K)$,

   $R(\text{Turing}, \epsilon)$,

2. ***Uniqueness axioms***:
   $\text{Gauss} \neq \text{Turing}$, $\epsilon \neq \text{Gauss}$, $\epsilon \neq \text{Turing}$.

3. ***Domain closure axiom***:

   $$\forall y(y = \text{Gauss} \vee y = \text{Turing} \vee y = 60K \vee y = \epsilon)$$

4. ***Completion axiom***:

   $$\forall y_1 y_2 (R(y_1, y_2) \rightarrow (y_1 = \text{Gauss} \wedge y_2 = 60K) \vee (y_1 = \text{Turing} \wedge y_2 = \epsilon)).$$

- Closed-world databases have only **finite** models. The *proof-theoretic approach*: replace $\models_f$ by $\vdash$.

- If $\neg(c_i = c_j)$ for all pairs of constants c_i and c_j in C, then T has a unique model, that is, it is **fully specified**.

Simulating Logical DB by Physical DB

L': add to L a new binary relation symbol NE ("inequality").

$LB=(L,T)$ - cw logical db

$\rightarrow Phys(LB)=(L',I)$ - physical db.

Domain - C (the set of constant symbols),

$I(c)=c$, $I(P)=\{\mathbf{c}:P(\mathbf{c})\in T\}$,

$I(NE)=\{(c_i,c_j):\neg(c_i=c_j)\in T$

or $\neg(c_j=c_i)\in T\}$.

Theorem. For every query Q, there is a query Q' such that if LB is a cw logical db, then $Q(LB) = Q'(Phys(LB))$.

Q' is obtained by adding to Q *second-order universal quantification* (i.e, *"for all relations"*).

Suggested implication: the combination of **nulls** with the **cw assumption** is inherently second order, which makes it less tractable.

Suggestion is correct!

Theorem

There is a first-order query Q such that for every first-order query Q' there exist a cw logical db LB such that

$$Q(LB) \neq Q'(Phys(LB))$$

Computational Complexity

$LOGSPACE \subseteq PTIME \subseteq NP \subseteq PSPACE.$

Complexity can be measured as a function of the *data*, as a function of the *query*, or as a function of *both*.

For simplicity we consider only *Boolean* queries, i.e., the answer is **Yes** or **No**.

How do you say "water" in a language L?

E.g., in Farsi, Thai, Swahili, Basquo,...

Time: O(log ||dictionary||).

How do you say a word w in Hebrew?

E.g., paper, article, reference, referee,...

Time: O(||word||).

Data Complexity

Physical Data Complexity:

$$PAns(Q) = \{PB : \mathbf{Yes} \in Q(PB)\}$$

Logical Data Complexity:

$$LAns(Q) = \{LB : \mathbf{Yes} \in Q(LB)\}$$

Expression Complexity

Physical Expression Complexity:

$$PAns(PB) = \{Q : \mathbf{Yes} \in Q(PB)\}$$

Logical Expression Complexity:

$$LAns(LB) = \{Q : \mathbf{Yes} \in Q(PB)\}$$

Combined Complexity

Physical Combined Complexity:

$$PAns = \{ < Q, PB > \ : \mathbf{Yes} \in Q(PB) \}$$

Logical Combined Complexity:

$$LAns = \{ < Q, LB > \ : \mathbf{Yes} \in Q(LB) \}$$

Physical Complexity of First-Order Queries

Theorem. [Chandra + Merlin]

(1) $PAns(Q)$ is in $LOGSPACE$. – Data complexity

(2) $PAns(PB)$ is in $PSPACE$. – Expression complexity

(3) There is a physical db PB, such that $PAns(PB)$ is $PSPACE$-complete. – "

(4) $PAns$ is $PSPACE$-complete. – Combined complexity

Logical Data Complexity of 1st-Order Queries

Theorem.

(1) $LAns(Q)$ is in co-NP.

(2) There is a query Q such that $LAns(Q)$ is co-NP-complete.

Co-NP is the class of languages whose complement is in NP. Validity of propositional formulas is the most well-known co-NP-complete problem.

Logical Expression Complexity

- A cw logical db has a **finite** number of models.

- Logical query evaluation = Physical query evaluation $\times$ Number of models.

Conclusion: Logical expression complexity= Constant $\times$ Physical expression complexity.

Combined Logical Complexity

Theorem. *LAns* is *PSPACE*-complete.

Complexity not affected because it is already **high**.

Solution: consider complexity for subclasses.

$LAns_{\Psi} = \{ < Q, LB > : Q \in \Psi \text{ and } \mathbf{Yes} \in Q(LB) \}$

Σ_k - 1st-order queries with k alternation of quantifiers, starting with an existential one.

E.g., $(\forall x)(\exists y)(P(x,y))$ is in Σ_2 .

$(\exists x)(\forall y)$

Complexity Shift

Polynomial Hierarchy: $\Sigma_0^p = \Pi_0^p = PTIME$,

$\Sigma_1^p = NP$, $\Pi_1^p = co\text{-}NP$, ...

$$\Sigma_0^p \subseteq \Sigma_1^p \subseteq \cdots \subseteq PSPACE$$
$$\Pi_0^p \subseteq \Pi_1^p \subseteq \cdots \subseteq PSPACE$$

Theorem. $\Psi = \Sigma_k$

(1) [Chandra+Harel] $PAns_\Psi$ is Σ_k^p-complete.

(2) $LAns_\Psi$ is Π_{k+1}^p-complete.

Approximate Query Evaluation - desiderata ~~Desiderata~~

- $A(Q,LB)$ *approximates* $Q(LB)$

- *Soundness* - $A(Q,LB) \subseteq Q(LB)$.

- *Completeness* - $A(Q,LB) = Q(LB)$ for fully specified db LB.

- Approximation Complexity = Physical Complexity

Simulating Logical DB by Physical DB

L': add to L a new binary relation symbol NE ("inequality").

$LB=(L,T)$ - cw logical db

$\rightarrow Phys(LB)=(L',I)$ - physical db.

Domain - C (the set of constant symbols),

$I(c)=c$, $I(P)=\{\mathbf{c}:P(\mathbf{c})\in T\}$,

$I(NE)=\{(c_i,c_j):\neg(c_i=c_j)\in T$

or $\neg(c_j=c_i)\in T\}$.

Theorem. For every query Q, there is a query Q' such that

(1) if LB is a cw logical db, then $Q(LB) \subseteq Q'(Phys(LB))$.

(2) if LB is a fully specified cw db, then $Q(LB) = Q'(Phys(LB))$.

(3) if Q is 1st order, then Q' is 1st order.

Query Modification

---

(1) Push negations in Q down to atomic formulas.

(2) Replace $\neg(x_i = x_j)$ by $NE(x_i, x_j)$.

(3) Replace $\neg P(\mathbf{x})$ by $\alpha_P(\mathbf{x})$.

Corollary: The algorithm is complete for positive queries.

Dealing with Negation

We want $\neg P$ to contain all tuples that cannot *be in* P.

$\mathbf{c} = < c_1 , \ldots , c_k >$, $\mathbf{d} = < d_1 , \ldots , d_k >$.

c and **d** **disagree** if

$$Unique(T) \cup \{ c_i = d_i : 1 \leq i \leq k \}$$

is *unsatisfiable*, where $Unique(T)$ is the set of uniqueness axioms in T.

Theorem. P - k-ary relation symbol. There is a 1st-order formula $\alpha_P(\mathbf{x})$ of length $O(k\log k)$ in the vocabulary $\{P,NE\}$ such that: for every interpretation I with C as its domain and every k-tuple $\mathbf{c}$ of constant symbols, I satisfies $\alpha_P(\mathbf{c})$ iff $\mathbf{c}$ disagrees with any k-tuple $\mathbf{d}$ in $I(P)$.

Intuitively: α_P is the provable complement of P.

Implementation

Logical databases can be approximately implemented on top of standard DBMS.

Problem: NE too big!!

Solution: U - unary relation of unknown values, NE' - binary relation of inequalities of unknown values, NE defined by

$$N(x,y) \equiv NE'(x,y) \backslash / (\neg U(x) / \backslash \neg U(y) / \backslash x \neq y)$$

Conclusions

- Extra expressive power costs money.

- One can have, though, a practical approximation.

- *Question*: How good is the approximation?