Querying Big Data with Bounded Data Access

Yang Cao

School of Informatics
University of Edinburgh

PhD thesis defense

March 31, 2016

THE UNIVERSITY
of EDINBURGH

Challenges introduced by Big Data

- The old: query evaluation is cost-prohibitive
  - CQ: NP-complete
  - FO (RA): PSPACE-complete.

Challenges introduced by Big Data

- The old: query evaluation is cost-prohibitive
  - CQ: NP-complete
  - FO (RA): PSPACE-complete.
- The new: efficient algorithms are not fast engough

  Using SSD of 6G/s, a linear scan of a dataset D would take
  - 1.9 days when D is of 1PB (10^{15}B)
  - 5.28 years when D is of 1EB (10^{18}B)

Challenges introduced by Big Data

- The old: query evaluation is cost-prohibitive
  - CQ: NP-complete
  - FO (RA): PSPACE-complete.
- The new: efficient algorithms are not fast engough

  Using SSD of 6G/s, a linear scan of a dataset D would take
  - 1.9 days when D is of 1PB (10^{15}B)
  - 5.28 years when D is of 1EB (10^{18}B)

$O(n)$ time is already beyond reach on big data in practice!

Challenges introduced by Big Data

- The old: query evaluation is cost-prohibitive
  - CQ: NP-complete
  - FO (RA): PSPACE-complete.
- The new: efficient algorithms are not fast engough

  Using SSD of 6G/s, a linear scan of a dataset D would take
  - 1.9 days when D is of 1PB (10^{15}B)
  - 5.28 years when D is of 1EB (10^{18}B)

$O(n)$ time is already beyond reach on big data in practice!

Can we still answer queries on big data with limited resources?

Bounded Evaluability

Answering queries with bounded data access

- Input: a class $\mathcal{L}$ of queries
- Question: Can we find, for any query $Q \in \mathcal{L}$, a plan ξ_Q for Q such that, for any big dataset D,

Bounded Evaluability

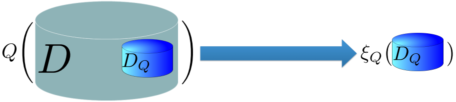
Answering queries with bounded data access

- Input: a class $\mathcal{L}$ of queries
- Question: Can we find, for any query $Q \in \mathcal{L}$, a plan ξ_Q for Q such that, for any big dataset D,
  - ξ_Q *accesses* only a fraction D_Q of D such that $\xi_Q(D) = Q(D)$; and
  - the size $|D_Q|$ of D_Q is *independent* of the size $|D|$ of D.

Bounded Evaluability

Answering queries with bounded data access

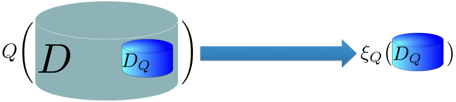
- Input: a class $\mathcal{L}$ of queries
- Question: Can we find, for any query $Q \in \mathcal{L}$, a plan ξ_Q for Q such that, for any big dataset D,
  - ξ_Q *accesses* only a fraction D_Q of D such that $\xi_Q(D) = Q(D)$; and
  - the size $|D_Q|$ of D_Q is *independent* of the size $|D|$ of D.



Bounded Evaluability

Answering queries with bounded data access

- Input: a class $\mathcal{L}$ of queries
- Question: Can we find, for any query $Q \in \mathcal{L}$, a plan ξ_Q for Q such that, for any big dataset D,
  - ξ_Q *accesses* only a fraction D_Q of D such that $\xi_Q(D) = Q(D)$; and
  - the size $|D_Q|$ of D_Q is *independent* of the size $|D|$ of D.

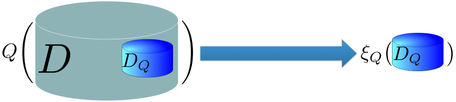


The execution cost of ξ_Q is independent of $|D|$.

Bounded Evaluability

Answering queries with bounded data access

- Input: a class $\mathcal{L}$ of queries
- Question: Can we find, for any query $Q \in \mathcal{L}$, a plan ξ_Q for Q such that, for any big dataset D,
  - ξ_Q *accesses* only a fraction D_Q of D such that $\xi_Q(D) = Q(D)$; and
  - the size $|D_Q|$ of D_Q is *independent* of the size $|D|$ of D.



The execution cost of ξ_Q is independent of $|D|$.

Making the cost of computing $Q(D)$ independent of $|D|$!

An Example:
Graph search (Facebook)

- Find me restaurants in New York my friends have been to in 2015

select rid
from friend(pid_1, pid_2), person(pid, name, city),
dine(pid, rid, dd, mm, yy)
where $\text{pid}_1 = p_0$ **and** $\text{pid}_2 =$ person.pid **and**
$\text{pid}_2 =$ dine.pid **and** city = $\textsc{NYC}$ **and** yy = 2015

An Example:

Graph search (Facebook)

- Find me restaurants in New York my friends have been to in 2015

$$\begin{array}{ll}\textbf{select} & \text{rid}\\ \textbf{from} & \text{friend}(\text{pid}_1,\ \text{pid}_2),\ \text{person}(\text{pid}, \text{name}, \text{city}),\\ & \text{dine}(\text{pid}, \text{rid}, \text{dd}, \text{mm}, \text{yy})\\ \textbf{where} & \text{pid}_1 = p_0\ \textbf{and}\ \text{pid}_2 = \text{person.pid}\ \textbf{and}\\ & \text{pid}_2 = \text{dine.pid}\ \textbf{and}\ \text{city} = \mathrm{NYC}\ \textbf{and}\ \text{yy} = 2015\end{array}$$

Access constraints (cardinality + indices) from data semantics

- Facebook: 5000 friends per person
- Each year has at most 366 days
- Each person dines at most once per day
- pid is a key for relation person

An Example:

Graph search (Facebook)

- Find me restaurants in New York my friends have been to in 2015

select rid
from friend(pid_1, pid_2), person(pid, name, city),
dine(pid, rid, dd, mm, yy)
where $\text{pid}_1 = p_0$ **and** $\text{pid}_2 = \text{person.pid}$ **and**
$\text{pid}_2 = \text{dine.pid}$ **and** city $= \text{NYC}$ **and** yy $= 2015$

Access constraints (cardinality + indices) from data semantics

- Facebook: 5000 friends per person
- Each year has at most 366 days
- Each person dines at most once per day
- pid is a key for relation person

Boundedly evaluable?

An Example:
Graph search (Facebook)

- Find me restaurants in New York my friends have been to in 2015

$$Q(\mathsf{rid}) = \exists p, p_1, n, c, d, m, y\,(\mathsf{friend}(p_0, p) \wedge \mathsf{person}(p, n, \mathrm{NYC}) \wedge \mathsf{dine}(p, \mathsf{rid}, d, m, 2015)$$

An Example:
Graph search (Facebook)

- Find me restaurants in New York my friends have been to in 2015

  $$Q(\text{rid}) = \exists p, p_1, n, c, d, m, y\,(\text{friend}(p_0, p) \wedge \text{person}(p, n, \text{NYC}) \wedge \text{dine}(p, \text{rid}, d, m, 2015))$$

  A query plan

  - Fetch 5000 pid's (p) for friends of p_0 – 5000 friends per person
  - For each p, check whether she lives in NYC – 5000 person tuples
  - For each p living in NYC, find restaurants (rid) where she dined in 2015 – 5000×366 tuples at most

An Example:
Graph search (Facebook)

- Find me restaurants in New York my friends have been to in 2015

  $Q(\text{rid}) = \exists p, p_1, n, c, d, m, y\, (\text{friend}(p_0, p) \wedge \text{person}(p, n, \text{NYC}) \wedge$
  $\text{dine}(p, \text{rid}, d, m, 2015))$

A query plan

- Fetch 5000 pid's (p) for friends of p_0 – 5000 friends per person
- For each p, check whether she lives in NYC – 5000 person tuples
- For each p living in NYC, find restaurants (rid) where she dined in 2015 – 5000×366 tuples at most

Accessing $5000 + 5000 + 5000 \times 366$ tuples in total

An Example:

Graph search (Facebook)

- Find me restaurants in New York my friends have been to in 2015

$$Q(\text{rid}) = \exists p, p_1, n, c, d, m, y\, (\text{friend}(p_0, p) \wedge \text{person}(p, n, \text{NYC}) \wedge \text{dine}(p, \text{rid}, d, m, 2015))$$

A query plan

- Fetch 5000 pid's (p) for friends of p_0 – 5000 friends per person
- For each p, check whether she lives in NYC – 5000 person tuples
- For each p living in NYC, find restaurants (rid) where she dined in 2015 – 5000×366 tuples at most

Accessing $5000 + 5000 + 5000 \times 366$ tuples in total

Boundedly evaluable under semantics constraints and indices

Questions Raised

Q1 How to define, decide and evaluate boundedly evaluable relational queries under access constraints?

Q2 Can we make use of the idea for unbounded queries?

Q3 Does the idea work for queries over data of other types?

Questions Raised

Q1 How to define, decide and evaluate boundedly evaluable relational queries under access constraints?

- Characterizations for fragments of boundedly evaluable FO queries;
- An effective syntax for boundedly evaluable RA queries

Q2 Can we make use of the idea for unbounded queries?

Q3 Does the idea work for queries over data of other types?

Questions Raised

Q1 How to define, decide and evaluate boundedly evaluable relational queries under access constraints?

- Characterizations for fragments of boundedly evaluable FO queries;
- An effective syntax for boundedly evaluable RA queries

Q2 Can we make use of the idea for unbounded queries?

- Extending bounded evaluability to bounded approximation
- Bounded evaluability with views

Q3 Does the idea work for queries over data of other types?

Questions Raised

Q1 How to define, decide and evaluate boundedly evaluable relational queries under access constraints?

- Characterizations for fragments of boundedly evaluable FO queries;
- An effective syntax for boundedly evaluable RA queries

Q2 Can we make use of the idea for unbounded queries?

- Extending bounded evaluability to bounded approximation
- Bounded evaluability with views

Q3 Does the idea work for queries over data of other types?

- Boundedly evaluable graph queries (isomorphism; simulation)

Outline

Part I: Boundedly Evaluable Relational Queries (Q1)
- Deciding Bounded Evaluability
- An Effective Syntax for Boundedly Evaluable RA Queries

Part II: Beyond Boundedly Evaluable Queries (Q2)
- From Bounded Evaluability to Bounded Approximation
- Bounded Evaluability with Views

Part III: Boundedly Evaluable Graph Queries (Q3)
- Bounded Evaluability on Graphs

Outline

Part I: Boundedly Evaluable Relational Queries (Q1)

- Deciding Bounded Evaluability
- An Effective Syntax for Boundedly Evaluable RA Queries

Part II: Beyond Boundedly Evaluable Queries (Q2)

- From Bounded Evaluability to Bounded Approximation
- Bounded Evaluability with Views

Part III: Boundedly Evaluable Graph Queries (Q3)

- Bounded Evaluability on Graphs

Bounded Evaluability under Access Constraints

Access constraints: on a relation schema R: $X \to (Y, N)$

- X, Y: sets of attributes of R
- for any X-value, there exist at most N distinct Y-values
- index on X for Y: given an X-value, find relavant Y-values.

Bounded Evaluability under Access Constraints

Access constraints: on a relation schema R: $X \to (Y, N)$

- X, Y: sets of attributes of R
- for any X-value, there exist at most N distinct Y-values
- index on X for Y: given an X-value, find relavant Y-values.

A combination of cardinality constraints and index

Bounded Evaluability under Access Constraints

Access constraints: on a relation schema R: $X \to (Y, N)$

- X, Y: sets of attributes of R
- for any X-value, there exist at most N distinct Y-values
- index on X for Y: given an X-value, find relavant Y-values.

A combination of cardinality constraints and index

For example:

- $\text{friend}(\text{pid}_1, \text{pid}_2)$: $\text{pid}_1 \to (\text{pid}_2, 5000)$ 5000 friends per person
- $\text{dine}(\text{pid}, \text{rid}, \text{dd}, \text{mm}, \text{yy})$: $\text{pid}, \text{yy} \to (\text{rid}, 366)$ each year has at most 366 days and each person dines at most once per day
- $\text{person}(\text{pid}, \text{name}, \text{city})$: $\text{pid} \to (\text{city}, 1)$ pid is a key for person

Bounded Evaluability under Access Constraints

Access constraints: on a relation schema R: $X \to (Y, N)$

- X, Y: sets of attributes of R
- for any X-value, there exist at most N distinct Y-values
- index on X for Y: given an X-value, find relavant Y-values.

Bounded plans under access constraints $\mathcal{A}$:

$\xi(Q, \mathcal{R}) : T_1 = \delta_1, \ldots, T_n = \delta_n$ where δ_i is

- $\{\mathrm{a}\}$: a constant in Q
- fetch$(X \in T_j, R, Y)$ via access constraint $R : X \to (Y, N)$, $j < i$
- a relational operator occurred in Q
- the length of $\xi(Q, \mathcal{R})$ is determined by $\mathcal{R}$, Q and $\mathcal{A}$ only, and is *independent of* $|D|$.

Bounded Evaluability under Access Constraints

Access constraints: on a relation schema R: $X \rightarrow (Y, N)$

- X, Y: sets of attributes of R
- for any X-value, there exist at most N distinct Y-values
- index on X for Y: given an X-value, find relavant Y-values.

Bounded plans under access constraints $\mathcal{A}$:

$\xi(Q, \mathcal{R}) : T_1 = \delta_1, \ldots, T_n = \delta_n$ where δ_i is

- $\{$a$\}$: a constant in Q
- fetch$(X \in T_j, R, Y)$ via access constraint $R : X \rightarrow (Y, N)$, $j < i$
- a relational operator occurred in Q
- the length of $\xi(Q, \mathcal{R})$ is determined by $\mathcal{R}$, Q and $\mathcal{A}$ only, and is *independent of* $|D|$.

Coping with big data scale independently *via* **boundedly evaluable queries**

Bounded evaluability problem BEP($\mathcal{L}$)

- Input: $\mathcal{R}$, $\mathcal{A}$, a query Q in language $\mathcal{L}$
- Question: Is Q boundedly evaluable under $\mathcal{A}$?

Bounded evaluability problem BEP($\mathcal{L}$)

- Input: $\mathcal{R}$, $\mathcal{A}$, a query Q in language $\mathcal{L}$
- Question: Is Q boundedly evaluable under $\mathcal{A}$?

It is known that BEP(FO) is undecidable in the absence of $\mathcal{A}$.
Is BEP decidable for CQ? UCQ? $\exists$FO$^+$?

Bounded evaluability problem BEP($\mathcal{L}$)

- Input: $\mathcal{R}$, $\mathcal{A}$, a query Q in language $\mathcal{L}$
- Question: Is Q boundedly evaluable under $\mathcal{A}$?

It is known that BEP(FO) is undecidable in the absence of $\mathcal{A}$.
Is BEP decidable for CQ? UCQ? $\exists$FO$^+$?

Challenges

- Only consider database instances $D \models \mathcal{A}$
- Access values (partial tuples) instead of tuples

Deciding Bounded Evaluability

The bounded evaluability problem BEP($\mathcal{L}$)

Bounded evaluability problem BEP($\mathcal{L}$)

- Input: $\mathcal{R}$, $\mathcal{A}$, a query Q in language $\mathcal{L}$
- Question: Is Q boundedly evaluable under $\mathcal{A}$?

It is known that BEP(FO) is undecidable in the absence of $\mathcal{A}$.
Is BEP decidable for CQ? UCQ? $\exists$FO$^+$?

Challenges

- Only consider database instances $D \models \mathcal{A}$
- Access values (partial tuples) instead of tuples

Complexity of BEP($\mathcal{L}$)

BEP(CQ), BEP(UCQ), BEP($\exists$FO$^+$) are all

- decidable in 2EXPSPACE; but
- EXPSPACE-hard

Deciding Bounded Evaluability

The bounded evaluability problem BEP($\mathcal{L}$)

Bounded evaluability problem BEP($\mathcal{L}$)

- Input: $\mathcal{R}$, $\mathcal{A}$, a query Q in language $\mathcal{L}$
- Question: Is Q boundedly evaluable under $\mathcal{A}$?

It is known that BEP(FO) is undecidable in the absence of $\mathcal{A}$.
Is BEP decidable for CQ? UCQ? $\exists$FO$^+$?

Challenges

- Only consider database instances $D \models \mathcal{A}$
- Access values (partial tuples) instead of tuples

Can we make practical use of bounded evaluability?

Covered RA queries

Effective syntax for boundedly evaluable RA queries

Covered RA queries: RA queries whose relation atoms are all "syntactically" covered by access constraints.

Covered RA queries

Effective syntax for boundedly evaluable RA queries

Covered RA queries: RA queries whose relation atoms are all "syntactically" covered by access constraints.

Effective Syntax

Under an access schema $\mathcal{A}$, for any RA query Q,

1. if Q is boundedly evaluable under $\mathcal{A}$, then Q is $\mathcal{A}$-equivalent to an RA query Q' that is covered by $\mathcal{A}$;
2. if Q is covered by $\mathcal{A}$, then Q is boundedly evaluable under $\mathcal{A}$; and
3. it takes PTIME to check whether Q is covered by $\mathcal{A}$.

Covered RA queries

Effective syntax for boundedly evaluable RA queries

Covered RA queries: RA queries whose relation atoms are all "syntactically" covered by access constraints.

Effective Syntax

Under an access schema $\mathcal{A}$, for any RA query Q,

1. if Q is boundedly evaluable under $\mathcal{A}$, then Q is $\mathcal{A}$-equivalent to an RA query Q' that is covered by $\mathcal{A}$;
2. if Q is covered by $\mathcal{A}$, then Q is boundedly evaluable under $\mathcal{A}$; and
3. it takes PTIME to check whether Q is covered by $\mathcal{A}$.

- design a quadratic algorithm for checking coverage;
- design a quadratic plan generation algorithm on top of DBMS;
- optimization: minimizing access constraints in use;

Covered RA queries

Effective syntax for boundedly evaluable RA queries

Covered RA queries: RA queries whose relation atoms are all "syntactically" covered by access constraints.

Effective Syntax

Under an access schema $\mathcal{A}$, for any RA query Q,

1. if Q is boundedly evaluable under $\mathcal{A}$, then Q is $\mathcal{A}$-equivalent to an RA query Q' that is covered by $\mathcal{A}$;
2. if Q is covered by $\mathcal{A}$, then Q is boundedly evaluable under $\mathcal{A}$; and
3. it takes PTIME to check whether Q is covered by $\mathcal{A}$.

A bounded evaluation framework on top of any existing DBMS

Covered RA queries

Effective syntax for boundedly evaluable RA queries

Covered RA queries: RA queries whose relation atoms are all "syntactically" covered by access constraints.

Effective Syntax

Under an access schema $\mathcal{A}$, for any RA query Q,

1. if Q is boundedly evaluable under $\mathcal{A}$, then Q is $\mathcal{A}$-equivalent to an RA query Q' that is covered by $\mathcal{A}$;
2. if Q is covered by $\mathcal{A}$, then Q is boundedly evaluable under $\mathcal{A}$; and
3. it takes PTIME to check whether Q is covered by $\mathcal{A}$.

Efficiency on real-life data: 9 seconds vs 14 hours of MySQL

Outline

Part I: Boundedly Evaluable Relational Queries (Q1)
- Deciding Bounded Evaluability
- An Effective Syntax for Boundedly Evaluable RA Queries

Part II: Beyond Boundedly Evaluable Queries (Q2)
- From Bounded Evaluability to Bounded Approximation
- Bounded Evaluability with Views

Part III: Boundedly Evaluable Graph Queries (Q3)
- Bounded Evaluability on Graphs

A Bounded Approximation Scheme

Bounded approximation scheme. Given any RA query Q and instance D of $\mathcal{R}$ that satisfies $\mathcal{A}$, it is to find a set S of tuples and a deterministic accuracy bound η (via *α-bounded plans*) such that

- it accesses a fraction D_Q of D with $|D_Q| \leq \alpha|D|$, and
- the *accuracy* of S to Q in D is at least η.

A Bounded Approximation Scheme

Bounded approximation scheme. Given any RA query Q and instance D of $\mathcal{R}$ that satisfies $\mathcal{A}$, it is to find a set S of tuples and a deterministic accuracy bound η (via *α-bounded plans*) such that

- it accesses a fraction D_Q of D with $|D_Q| \leq \alpha|D|$, and
- the *accuracy* of S to Q in D is at least η.

RC-measure: for a set S of tuples, RA query Q and database D

- (Relevance η_{rel}) each tuple $s \in S$ is a sensible and relevant answer to Q in D, above η_{rel}; and
- (Coverage η_{cov}) for each $t \in Q(D)$, there exists $s \in S$ that is close enough to t, above η_{cov}.

A Bounded Approximation Scheme

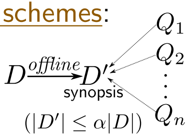
Bounded approximation scheme. Given any RA query Q and instance D of $\mathcal{R}$ that satisfies $\mathcal{A}$, it is to find a set S of tuples and a deterministic accuracy bound η (via *α-bounded plans*) such that

- it accesses a fraction D_Q of D with $|D_Q| \leq \alpha|D|$, and
- the *accuracy* of S to Q in D is at least η.

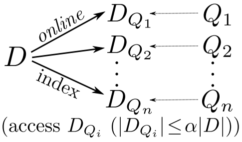
RC-measure: for a set S of tuples, RA query Q and database D

- (Relevance η_{rel}) each tuple $s \in S$ is a sensible and relevant answer to Q in D, above η_{rel}; and
- (Coverage η_{cov}) for each $t \in Q(D)$, there exists $s \in S$ that is close enough to t, above η_{cov}.

Data reduction schemes:



(e) *One-size-fit-all*



(f) *Dynamic reduction*

Approximability theorem

Given any set $\mathcal{A}_c$ of access constraints of $\mathcal{R}$, there exists a set $\mathcal{A}_t$ of *access templates* such that for any $D \models \mathcal{A}_c$,

- $D \models \mathcal{A}_0$, where $\mathcal{A}_0 = \mathcal{A}_c \cup \mathcal{A}_t$;
- $|\mathcal{A}_t| \leq |\mathcal{R}| + ||\mathcal{A}_c||$ and $|\mathcal{I}_{\mathcal{A}_0}|$ is in $O(||\mathcal{A}_0|||D|)$; and
- there exists a bounded approximation scheme with $\mathcal{A}_0$ for any resource ratio $\alpha \in (0, 1]$ and all RA queries.

Bounded Approximation of Relational Queries

The approximability theorem

Approximability theorem

Given any set $\mathcal{A}_c$ of access constraints of $\mathcal{R}$, there exists a set $\mathcal{A}_t$ of *access templates* such that for any $D \models \mathcal{A}_c$,

- $D \models \mathcal{A}_0$, where $\mathcal{A}_0 = \mathcal{A}_c \cup \mathcal{A}_t$;
- $|\mathcal{A}_t| \leq |\mathcal{R}| + ||\mathcal{A}_c||$ and $|\mathcal{I}_{\mathcal{A}_0}|$ is in $O(||\mathcal{A}_0|||D|)$; and
- there exists a bounded approximation scheme with $\mathcal{A}_0$ for any resource ratio $\alpha \in (0, 1]$ and all RA queries.

- Access templates: combine access constraints with approximation;
- α-bounded plans: from bounded query plan to α-execution plan;
- SPC vs. RA: SPC guarantees non-empty accuracy bounds (but finding optimal accuracy bound is already Σ_3^p-hard for SPC).

Extending Bounded Evaluability with Views

- Bounded rewriting using views:
  Query Q has bounded rewriting using views $\mathcal{V}$ under access constraints $\mathcal{A}$ if it has a *bounded (rewriting) plan* using $\mathcal{V}$ under $\mathcal{A}$.

Extending Bounded Evaluability with Views

- Bounded rewriting using views:

  Query Q has bounded rewriting using views $\mathcal{V}$ under access constraints $\mathcal{A}$ if it has a *bounded (rewriting) plan* using $\mathcal{V}$ under $\mathcal{A}$.

> Need to check whether sub-queries containing views have *bounded output* under access constraints

Extending Bounded Evaluability with Views

- Bounded rewriting using views:
  Query Q has bounded rewriting using views $\mathcal{V}$ under access constraints $\mathcal{A}$ if it has a *bounded (rewriting) plan* using $\mathcal{V}$ under $\mathcal{A}$.

  Need to check whether sub-queries containing views have *bounded output* under access constraints

- Complexity: undecidable for FO; Σ_3^p-hard for CQ.

Extending Bounded Evaluability with Views

- Bounded rewriting using views:
  Query Q has bounded rewriting using views $\mathcal{V}$ under access constraints $\mathcal{A}$ if it has a *bounded (rewriting) plan* using $\mathcal{V}$ under $\mathcal{A}$.

> Need to check whether sub-queries containing views have *bounded output* under access constraints

- Complexity: undecidable for FO; Σ_3^p-hard for CQ.
- Effective syntax to make practical use of bounded rewriting

Topped queries

- each FO query Q with a M-bounded rewriting is $\mathcal{A}$-equivalent to a topped query;
- every topped query has an M-bounded rewriting;
- it is in PTIME to check whether Q is topped (call for checking bounded output)

Size-bounded queries

- each FO query Q with bounded output is $\mathcal{A}$-equivalent to a size-bounded query;
- every size-bounded query has bounded output under $\mathcal{A}$;
- it takes PTIME in $|Q|$ to check whether Q is size-bounded.

Outline

Part I: Boundedly Evaluable Relational Queries (Q1)

Deciding Bounded Evaluability

An Effective Syntax for Boundedly Evaluable RA Queries

Part II: Beyond Boundedly Evaluable Queries (Q2)

From Bounded Evaluability to Bounded Approximation

Bounded Evaluability with Views

Part III: Boundedly Evaluable Graph Queries (Q3)

Bounded Evaluability on Graphs

Boundedly Evaluable Graph Pattern Queries

Access constraints on graphs: $S \to (l, N)$

- S: a set of labels; l: a label.
- For any S-labelled set V_S, there exists at most N l-labelled common neighbors of V_S.
- Index on G: given a V_S, find relevant l-labeled relevant neighbors.

Boundedly Evaluable Graph Pattern Queries

Access constraints on graphs: $S \to (l, N)$

- S: a set of labels; l: a label.
- For any S-labelled set V_S, there exists at most N l-labelled common neighbors of V_S.
- Index on G: given a V_S, find relevant l-labeled relevant neighbors.

Boundedly evaluable graph pattern queries:

- Characterizations via node and edge coverage;
- Decidable in quadratic time for both subgraph (localized) and simulation (non-localized) pattern queries.

Boundedly Evaluable Graph Pattern Queries

Access constraints on graphs: $S \rightarrow (l, N)$

- S: a set of labels; l: a label.
- For any S-labelled set V_S, there exists at most N l-labelled common neighbors of V_S.
- Index on G: given a V_S, find relevant l-labeled relevant neighbors.

Boundedly evaluable graph pattern queries:

- Characterizations via node and edge coverage;
- Decidable in quadratic time for both subgraph (localized) and simulation (non-localized) pattern queries.

Algorithms:

- Quadratic time checking and plan generation algorithms
- Instance-boundedness: making query workload bounded

Boundedly Evaluable Graph Pattern Queries

Access constraints on graphs: $S \to (l, N)$

- S: a set of labels; l: a label.
- For any S-labelled set V_S, there exists at most N l-labelled common neighbors of V_S.
- Index on G: given a V_S, find relevant l-labeled relevant neighbors.

Boundedly evaluable graph pattern queries:

- Characterizations via node and edge coverage;
- Decidable in quadratic time for both subgraph (localized) and simulation (non-localized) pattern queries.

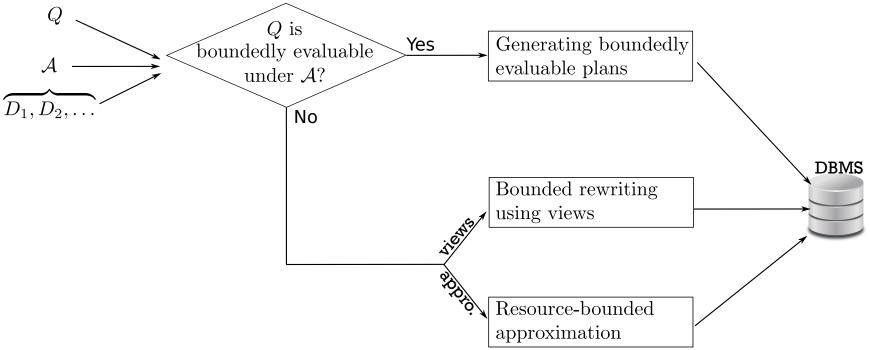
Algorithms:

- Quadratic time checking and plan generation algorithms
- Instance-boundedness: making query workload bounded

> - *Pattern queries are found commonly bounded with small N's.*
> - *Outperform conventional approaches by 3-4 orders of magnitude on medium size graphs.*

Conclusion and Further Research

A package of methods for querying big data with quantified data access



Conclusion and Further Research

Further research: *quantified query answering*

- Bounded evaluability
  - Bounded evaluability with integrity constraints (*e.g.*, INDs)
  - $\mathcal{L}_Q$-to-$\mathcal{L}_{Plan}$ evaluability
  - M-bounded evaluability
  - Bounded approximation with views
  - Access constraints discovery for parameterized query workload
- Computation models
  - bounded communication ($\textsc{Local}$ distributed model);
  - bounded I/O (EM model);
  - bounded incremental (Local Persistent model);
- Data structures and algorithms for sublinear query evaluation with pre-processing
  - Bounded sparsification for approximating Boolean pattern queries

Things Done in the PhD Study

- Querying big data with bounded resources
  - *An Effective Syntax for Bounded Relational Queries*, with Fan, SIGMOD'16
  - *Bounded Query Rewriting Using Views*, with Fan, Geerts & Lu, PODS'16
  - *Making Pattern Queries Bounded in Big Graphs*, with Fan & Huang, ICDE'15
  - *Querying Big Data by Accessing Small Data*, with Fan, Geerts, Deng & Lu, PODS'15
  - *Bounded Conjunctive Queries*, with Fan & Yu, VLDB'14
  - *Answering Relational Queries with Bounded Resources*, with Fan, submitted to VLDB
- Graph pattern matching: semantics, algorithms and applications
  - *Virtual Network Mapping: A Graph Pattern Matching Approach*, with Fan & Ma, BICOD'15
  - *Strong Simulation: Capturing Topology in Graph Pattern Matching*, with Ma, Fan, Huai & Wo, TODS'14
- Data quality: accuracy and information incompleteness
  - *On the Data Complexity of Relative Information Completeness*, with Deng, Fan & Geerts, Information System'14
  - *Determining the Relative Accuracy of Attributes*, with Fan & Yu, SIGMOD'13

Things Done in the PhD Study

- Querying big data with bounded resources

This Thesis

  - *An Effective Syntax for Bounded Relational Queries*, with Fan, SIGMOD'16
  - *Bounded Query Rewriting Using Views*, with Fan, Geerts & Lu, PODS'16
  - *Making Pattern Queries Bounded in Big Graphs*, with Fan & Huang, ICDE'15
  - *Querying Big Data by Accessing Small Data*, with Fan, Geerts, Deng & Lu, PODS'15
  - *Bounded Conjunctive Queries*, with Fan & Yu, VLDB'14
  - *Answering Relational Queries with Bounded Resources*, with Fan, submitted to VLDB

- Graph pattern matching: semantics, algorithms and applications
  - *Virtual Network Mapping: A Graph Pattern Matching Approach*, with Fan & Ma, BICOD'15
  - *Strong Simulation: Capturing Topology in Graph Pattern Matching*, with Ma, Fan, Huai & Wo, TODS'14

- Data quality: accuracy and information incompleteness
  - *On the Data Complexity of Relative Information Completeness*, with Deng, Fan & Geerts, Information System'14
  - *Determining the Relative Accuracy of Attributes*, with Fan & Yu, SIGMOD'13

The End

THANK YOU!

Q and A