An Effective Syntax for Bounded Relational Queries

Yang Cao       Wenfei Fan

University of Edinburgh
Beihang University

SIGMOD 2016
June 29, 2016
Background and Motivation

- Motivation: querying big data is cost-prohibitive
  - old challenge: complexity of query evaluation
  - new challenge: big data as the input
Background and Motivation

- Motivation: querying big data is cost-prohibitive
  - old challenge: complexity of query evaluation
  - new challenge: big data as the input
- A recent approach: **bounded evaluatibility**

querying big data by accessing small data of bounded size
Background and Motivation

- **Motivation:** querying big data is cost-prohibitive
  - **old** challenge: complexity of query evaluation
  - **new** challenge: big data as the input
- **A recent approach:** **bounded evaluatibility**

  *querying big data by accessing small data of bounded size*

- **Previous works**
  - **Formalization**
    2. *Querying big data by accessing small data*, W. Fan, F. Geerts, Y. Cao, T. Deng, P. Lu, PODS’15
  - **Incorporating views**
  - **Extending to graph data**
    4. *Making Pattern Queries Bounded in Big Graphs*, Y. Cao, W. Fan and R. Huang, ICDE’15
  - **Restrictions and validation**
    5. *Bounded Conjunctive Queries*, Y. Cao, W. Fan, T. Wo and W. Yu, VLDB’14

The method was recently tested by a giant company: 3 to 1000+ times faster than conventional approach. Bounded evaluatibility is effective for querying big data.
Background and Motivation

- **Motivation:** querying big data is cost-prohibitive
  - **old** challenge: complexity of query evaluation
  - **new** challenge: big data as the input
- **A recent approach:** bounded evaluatbility

  **querying big data by accessing small data of bounded size**

- **Previous works**
  - **Formalization**
    2. *Querying big data by accessing small data*, W. Fan, F. Geerts, Y. Cao, T. Deng, P. Lu, PODS’15
  - **Incorporating views**
  - **Extending to graph data**
    4. *Making Pattern Queries Bounded in Big Graphs*, Y. Cao, W. Fan and R. Huang, ICDE’15
  - **Restrictions and validation**
    5. *Bounded Conjunctive Queries*, Y. Cao, W. Fan, T. Wo and W. Yu, VLDB’14
- **The method was recently tested by a giant company:** 3 to 1000+ times faster than conventional approach.
Background and Motivation

- **Motivation**: querying big data is cost-prohibitive
  - old challenge: complexity of query evaluation
  - new challenge: big data as the input
- **A recent approach**: bounded evaluability
  
  *querying big data by accessing small data of bounded size*

- **Previous works**
  - **Formalization**
    1. On Scale Independence for Querying Big Data, W. Fan, F. Geerts and L. Libkin, PODS’14
    2. Querying big data by accessing small data, W. Fan, F. Geerts, Y. Cao, T. Deng, P. Lu, PODS’15
  - **Incorporating views**
  - **Extending to graph data**
    4. Making Pattern Queries Bounded in Big Graphs, Y. Cao, W. Fan and R. Huang, ICDE’15
  - **Restrictions and validation**
- The method was recently tested by a giant company: 3 to 1000+ times faster than conventional approach.

  *Bounded evaluability is effective for querying big data*
**Basic idea**: compute $Q(D)$ via query plans that access only a small subset $D_Q$ of $D$, via indices built w.r.t. a set $A$ of **access constraints**.

\[ Q(D) \rightarrow \xi_Q(D_Q) \]
**Bounded Evaluability: Review**

**Answering queries with bounded data access**

**Basic idea**: compute \( Q(D) \) via query plans that access only a small subset \( D_Q \) of \( D \), via indices built w.r.t. a set \( A \) of *access constraints*.

![Diagram showing query execution](image)

- **Access constraints**: a combination of cardinality constraints \( R(X \rightarrow Y, N) \) with indices;
Bounded Evaluability: Review
Answering queries with bounded data access

**Basic idea**: compute $Q(D)$ via query plans that access only a small subset $D_Q$ of $D$, via indices built w.r.t. a set $A$ of *access constraints*.

![Diagram showing query plan](image)

- **Access constraints**: a combination of cardinality constraints $R(X \rightarrow Y, N)$ with indices;
- **Boundedly evaluable query plan $\xi_Q$**: extended relational algebra query plans with a fetch operation $\text{fetch}(X \in T, R, Y)$ w.r.t. $A$;
Bounded Evaluability: Review
Answering queries with bounded data access

**Basic idea**: compute $Q(D)$ via query plans that access only a small subset $D_Q$ of $D$, via indices built w.r.t. a set $A$ of **access constraints**.

![Diagram](attachment://bounded_evaluability_diagram.png)

- **Access constraints**: a combination of cardinality constraints $R(X \rightarrow Y, N)$ with indices;
- **Boundedly evaluable query plan** $\xi_Q$: extended relational algebra query plans with a fetch operation $\text{fetch}(X \in T, R, Y)$ w.r.t. $A$;
- **Boundedly evaluable queries**: queries with boundedly evaluable plans under $A$. 
Bounded Evaluability: Review
Answering queries with bounded data access

Basic idea: compute $Q(D)$ via query plans that access only a small subset $D_Q$ of $D$, via indices built w.r.t. a set $A$ of access constraints.

- Access constraints: a combination of cardinality constraints $R(X \rightarrow Y, N)$ with indices;
- Boundedly evaluable query plan $\xi_Q$: extended relational algebra query plans with a fetch operation $\text{fetch}(X \in T, R, Y)$ w.r.t. $A$;
- Boundedly evaluable queries: queries with boundedly evaluable plans under $A$.

Making the cost of computing $Q(D)$ independent of $|D|$!
Find me restaurants in San Francisco my Facebook 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   pid_1 = p_0 and pid_2 = person.pid and
        pid_2 = dine.pid and city = SF and yy = 2015
```
Bounded Evaluability: An Example

Graph search (Facebook)

Find me restaurants in San Francisco my Facebook friends have been to in 2015

\[ Q(\text{rid}) = \exists p, p_1, n, c, d, m, y \ (\text{friend}(p_0, p) \land \text{person}(p, n, \text{SF}) \land \text{dine}(p, \text{rid}, d, m, 2015) \]

Access constraints (cardinality + indices) from data semantics
Bounded Evaluability: An Example

Graph search (Facebook)

Find me restaurants in San Francisco my Facebook friends have been to in 2015

\[ Q(rid) = \exists p, p_1, n, c, d, m, y \ (\text{friend}(p_0, p) \land \text{person}(p, n, \text{SF}) \land \text{dine}(p, rid, d, m, 2015)) \]

Access constraints (cardinality + indices) from data semantics

- friend(pid₁, pid₂): pid₁ \rightarrow (pid₂, 5000)  \hspace{1cm} 5000 friends per person
- dine(pid, rid, dd, mm, yy): pid, yy \rightarrow (rid, 366)  \hspace{1cm} each year has at most 366 days and each person dines at most once per day
- person(pid, name, city): pid \rightarrow (city, 1)  \hspace{1cm} pid is a key for person
Bounded Evaluability: An Example

Graph search (Facebook)

Find me restaurants in San Francisco my Facebook friends have been to in 2015

\[ Q(\text{rid}) = \exists p, p_1, n, c, d, m, y \ (\text{friend}(p_0, p) \land \text{person}(p, n, \text{SF}) \land \text{dine}(p, \text{rid}, d, m, 2015) \]

Access constraints (cardinality + indices) from data semantics

A boundedly evaluable query plan:

- Fetch 5000 pid’s (p) for friends of p_0 – 5000 friends per person
- For each p, check whether she lives in SF – 5000 person tuples
- For each p living in SF, find restaurants (rid) where she dined in 2015 – 5000 \times 366 tuples at most
Bounded Evaluability: An Example

Graph search (Facebook)

Find me restaurants in San Francisco my Facebook friends have been to in 2015

\[ Q(\text{rid}) = \exists p, p_1, n, c, d, m, y \ (\text{friend}(p_0, p) \land \text{person}(p, n, \text{SF}) \land \text{dine}(p, \text{rid}, d, m, 2015) \]

Access constraints (cardinality + indices) from data semantics

A boundedly evaluable query plan:

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

Accessing 5000 + 5000 + 5000 \( \times \) 366 tuples in total
Bounded Evaluability: An Example

Graph search (Facebook)

Find me restaurants in San Francisco my Facebook friends have been to in 2015

\[ Q(\text{rid}) = \exists p, p_1, n, c, d, m, y \ (\text{friend}(p_0, p) \land \text{person}(p, n, \text{SF}) \land \\
\text{dine}(p, \text{rid}, d, m, 2015) \]

Access constraints (cardinality + indices) from data semantics

Price to use bounded evaluability:

- **EXPSPACE-hard** to decide whether an SPC query (CQ; basic Select-From-Where clause) is boundedly evaluable
- **undecidable** for RA (FO; SQL) queries
Find me restaurants in San Francisco my Facebook friends have been to in 2015

\[ Q(\text{rid}) = \exists p, p_1, n, c, d, m, y \ (\text{friend}(p_0, p) \land \text{person}(p, n, \text{SF}) \land \text{dine}(p, \text{rid}, d, m, 2015) \]

Access constraints (cardinality + indices) from data semantics

Price to use bounded evaluability:

- **EXPSPACE-hard** to decide whether an SPC query (CQ; basic Select-From-Where clause) is boundedly evaluable
- **undecidable** for RA (FO; SQL) queries

How to make *practical* use of bounded evaluability *without sacrificing* its express power?
Covered RA queries $\mathcal{L}_{RAFT}^c$: RA queries whose relation atoms are all “syntactically” covered by access constraints.
Covered RA queries $\mathcal{L}^c_{\text{RA}}$: RA queries whose relation atoms are all “syntactically” covered by access constraints.

Under a set $\mathcal{A}$ of access constraints,

1. any boundedly evaluable RA query is $\mathcal{A}$-equivalent to a query covered by $\mathcal{A}$;
2. every covered query is also boundedly evaluable;
3. it takes PTIME to check whether $Q$ is covered by $\mathcal{A}$.
Covered queries

Covered RA queries $\mathcal{L}_{RA}^c$: RA queries whose relation atoms are all “syntactically” covered by access constraints.

Effective Syntax

Under a set $\mathcal{A}$ of access constraints,

1. any boundedly evaluable RA query is $\mathcal{A}$-equivalent to a query covered by $\mathcal{A}$;
2. every covered query is also boundedly evaluable;
3. it takes PTIME to check whether $Q$ is covered by $\mathcal{A}$.

$\mathcal{L}_{RA}^c$ identifies the core subclass of boundedly evaluable RA queries, without sacrificing their expressive power.
A Bounded Evaluation Framework
A constructive proof of the effective syntax (2-3)
A Bounded Evaluation Framework
A constructive proof of the effective syntax (2-3)

- **C2**: a proof of property 3
- **C4**: a proof of property 2
- **C3**: optimizing bounded plans
- **C5**: ensure DBMS-independence
Experimental results

We evaluated the bounded evaluation framework:

- **easy-to-use**
  - easy to find 100+ access constraints in real-life data by extending constraints discovery algorithms
  - half of queries over the attributes in the constraints are covered
- **DBMS-independence:**
  - access constraints index is easy to build via DBMS
  - bounded plans can be directly executed on DBMS engines

- **speedup of boundedly evaluable plans vs conventional**
  - 5.9 seconds by accessing at most 0.00017% of the data (60GB)
    vs. 3000+ seconds (3.75GB) for many-join queries
  - guaranteed scale independence once covered
Experimental results

We evaluated the bounded evaluation framework:

- **easy-to-use**
  - easy to find 100+ access constraints in real-life data by extending constraints discovery algorithms
  - half of queries over the attributes in the constraints are covered

- **DBMS-independence**:
  - access constraints index is easy to build via DBMS
  - bounded plans can be directly executed on DBMS engines

- **speedup of boundedly evaluable plans vs conventional**
  - 5.9 seconds by accessing at most 0.00017% of the data (60GB)
    VS. 3000+ seconds (3.75GB) for many-join queries
  - guaranteed scale independence once covered

*With the effective syntax, we can use bounded evaluability to query big data by accessing bounded small data.*
The End

THANK YOU!

Q&A