Towards a theory of search queries

Jan Van den Bussche
(Hasselt University)
Joint work with
George Fletcher, Dirk Van Gucht,
Stijn Vansummeren

ACM TODS (November 2010)

Outline

1. Theory of database queries
2. Relational algebra
3. Semijoin algebra
4. Search queries
5. Dataspaces
6. Structured querying versus searching
7. Research problems

Computational problems

- Classically, any computational problem is a function (mapping) from inputs to outputs
- E.g., route planning:
  - Input: a map (graph), source, target
  - Output: shortest route in graph from source to target
- Deal with nondeterminism

Database queries

- A **query** is a function from databases to databases
- E.g., Employee query
  - Input: history of employee hirings
  - Output: list of all employees who have been hired at least twice
- Also route planning!

Relational algebra

- Language in which queries over relational databases can be expressed
- Every expression denotes a query
  - compare arithmetic: avg(x,y) = (x+y)/2
- Expression is a combination of operators
  - union, intersection, difference
  - cartesian product (join)
  - selection
  - projection
  - renaming

Employee query

relation History(emp_id, hire_date)

$$\pi_{H1.emp\_id}\ \sigma_{H1.emp\_id=H2.emp\_id\ and\ H1.hire\_date\neq H2.hire\_date}\ (\rho_{H1}(History) \times \rho_{H2}(History))$$

equivalently:

$$\pi_{H1.emp\_id}\ (\rho_{H1}(History) \underset{H1.emp\_id=H2.emp\_id \atop H1.hire\_date\neq H2.hire\_date}{\bowtie} \rho_{H2}(History))$$

Another example

- Extreme elements query:
  - Input: a total order relation R(x,y)
  - Output: the minimum and maximum element

$$(\pi_x(R) \setminus \pi_y(R)) \cup (\pi_y(R) \setminus \pi_x(R))$$

Expressibility

- Not all queries are expressible in relational algebra
- E.g., route planning
- Not surprising
  - avg(x,y) versus sin(x)

The first-order queries

- Relational algebra forms an important **core query language**
  - SQL select-statements = rel.alg. + aggregates
  - even XPath 2.0 = relational algebra!
  - also SPARQL = relational algebra
- Queries expressible in relational algebra are called the **first-order queries**
  - relational calculus (first-order logic)

Semijoin

- Recall Employee query:

$$\pi_{H1.emp\_id}\,(\rho_{H1}(History) \bowtie_{\substack{H1.emp\_id=H2.emp\_id \\ H1.hire\_date\neq H2.hire\_date}} \rho_{H2}(History))$$

- We don't need attributes of H2 after join
- Semijoin:

$$\pi_{H1.emp\_id}\,(\rho_{H1}(History) \ltimes_{\substack{H1.emp\_id=H2.emp\_id \\ H1.hire\_date\neq H2.hire\_date}} \rho_{H2}(History))$$

The semijoin algebra (SA)

- Same as relational algebra, except:

  $\times$ and $\bowtie$ are replaced by $\ltimes$

- SA queries…
  - always return subset of the relations (possibly π)
  - can be efficiently processed
    - sorting
    - one-pass query processing
    - linear
- SA with only equalities in join conditions

  = the **linear fragment** of relational algebra

Searching versus Querying

- Users of information systems do not use SQL
  - Google
  - Library catalog
- Programs built over information retrieval (full text) engine cannot call SQL
  - Websites
- They can **search:**
  - ti=databases AND NOT au=ullman
  - pyrrhula OR bullfinch

Pyrrhula pyrrhula (Eurasian Bullfinch)

Abstract Dataspaces

- An abstract **dataspace** is a set of **objects**
- Each **object** is a set of **items**
- E.g., set of webpages
  - each webpage = set of strings
- E.g., classical relation is set of tuples
  - each tuple = set of attribute–value pairs

Attribute–value pairs

- Tuple

emp_id	hire_date	job
1234	20091021	programmer

- Set of attribute–value pairs

att	val
emp_id	1234
hire_date	20091021
job	programmer

Attribute–value dataspaces

- Objects are arbitrary sets of AV-pairs

name	John
paper	p1
paper	p2
location	Namur
likes	Orval

name	Anne
paper	p1
location	Brussels
phone	022222785

name	Mary
paper	p2
paper	p3
location	Brussels
location	Antwerp
hobby	birdwatching

paper_id	p1
title	SQL
proceedings	VLDB

paper_id	p2
title	XQuery
proceedings	VLDB
citations	55

paper_id	p3
title	Pyrrhula song
journal	Ornithology

drink_type	beer
name	Orval
kind	Trappist

Orval


ORVAL
TRAPPIST ALE
IMPORTERS MERCHANT DU VIN CORP. TUKWILA, WA
ORVAL
TRAPPIST ALE

“Database of everything”

- Alon Halevy
- Very similar to Semantic Web
  - RDF
  - Linked Data
- Personal Information Management
- NoSQL databases

A–V dataspace as RDF store

- RDF store: set of triples
  - (subject, predicate, object)
- view A–V dataspace *D* as set of triples:
  - {(oid,att,val) : oid ∈ *D* & (att,val) ∈ *D*}

1	name	John
1	paper	p1
1	paper	p2
1	location	Namur
1	likes	Orval
2	name	Anne
2	paper	p1
2	location	Brussels
2	phone	022222785
3	name	Mary
3	paper	p2
3	paper	p3
3	location	Brussels
3	location	Antwerp
3	hobby	birdwatching
4	paper_id	p1
4	title	SQL
4	proceedings	VLDB
5	paper_id	p2
5	title	XQuery
5	proceedings	VLDB
5	citations	55
6	paper_id	p3
6	title	Pyrrhula song
6	journal	Ornithology
7	drink_type	beer
7	name	Orval
7	kind	Trappist

RDF triple store as A–V dataspace

- Use 3 special attributes
  - **subj**ect
  - **pred**icate
  - **obj**ect
- RDF triple store is just a relation over the scheme {subj,pred,obj}
- Already know a relation is a dataspace!
- No RDFS

Searching Dataspaces

- Abstract Dataspace
  - set of **objects**
  - **object**: set of **items**
- Abstract **keyword**
  - predicate on items
- E.g., when items are strings:
  - string contains “Brussel”

Boolean Search Language (BSL)

- Every keyword *k* is an expression
- Meaning:
  - Retrieve all objects containing some item satisfying *k*
- If e1 and e2 are expressions then so are:
  - e1 OR e2
  - e1 AND e2
  - e1 AND NOT e2
- Meaning: union, intersection, set difference
- Bruxelles AND NOT (Orval OR Chimay)

Dataspace search queries

- Database query:
  - mapping from databases to databases
- Dataspace query:
  - mapping *q* from dataspaces to dataspaces
- Dataspace **search** query:
  - such that $q(D) \subset D$ for each D
- Bit like semijoin queries…

Which dataspace search queries…

- …are expressible in BSL?
- BSL queries are **safe**
  - Only returns objects containing some item satisfying some keyword that we used
- BSL queries are **additive**

$$q(D) = \text{union of all } q(\{o\}) \text{ for } o \in D$$

BSL queries are **finitely distinguishing**

- Only distinguish objects using some finite set *K* of keywords
- *o1* and *o2* are "*K*-equivalent" if for each *k* in *K*, *o1* matches *k* $\Leftrightarrow$ *o2* matches *k*
- when *o1* and *o2* from *D* are *K*-equivalent then $o1 \in q(D) \Leftrightarrow o2 \in q(D)$

Characterisation of BSL

- A dataspace query *q* is expressible in BSL if (and only if) *q* is additive, and for some finite set *K* of keywords,
  - *q* is *K*-safe and
  - *q* is *K*-distinguishing

Application to relational selection queries

- Recall: relation = set of tuples = set of objects
- Object = set of attribute–value pairs
- Keywords: *A=c*
  - *A*: attribute from the given relation scheme
  - *c*: arbitrary constant
- Also wildcard keyword: *
- Example BSL query:

* AND NOT (job=programmer OR emp_id=1234)

- Same as rel.alg. using only $\cup$, $\setminus$, $\sigma_{A=c}$

Characterising relational selection queries

- A relational selection query is expressible in the relational algebra using only $\cup$, $\setminus$, $\sigma_{A=c}$ if and only if it is additive and commutes with any C-epimorphism, for some finite set C of constants.
- C-epimorphism: function f from values to values such that f and f^{-1} are the identity on C.
- q commutes with f:

$$q(f(D)) = f(q(D))$$

- In line with known "genericity" properties [Aho&Ullman, Chandra&Harel, Hull&Yap, Abiteboul&Vianu]

Characterisation of BSL (repeated)

- A dataspace query *q* is expressible in BSL if (and only if) *q* is additive, and for some finite set *K* of keywords,
  - *q* is *K*-safe and
  - *q* is *K*-distinguishing

Not expressible in BSL

- Negated keywords (if you don't have them)
  - retrieve all objects containing an item **not** matching "Brussel"
  - not finitely distinguishing over positive keywords
- Normally will use boolean-closed repertoire of keywords

Neither expressible in BSL

- Retrieve all objects sharing an item with an object matching “Brussel”
- Retrieve all co-authors of Mary
- Not additive
- We cannot do joins or even semijoins
- Want to do such “associative search”

Similarity relations (simrels)

- How to link two objects?
  - hardwire links between objects in the dataspace
  - not necessary
  - not flexible
- Better: use **simrels** between items
  - a simrel is a binary predicate on items

Examples of simrels

- Equality
- Translation on city names:
  - Bruxelles trans Brussel
  - Anvers trans Antwerpen
  - Namur trans Namen
- Equal-value on A–V pairs:
  - (likes, Orval) eqval (name, Orval)
- Equal-attribute on A–V pairs:
  - (name, John) eqatt (name, Orval)

Simlinks

- If *k* and *k'* are keywords, and ≈ is a simrel, then *k* ≈ *k'* is a **simlink.**
- Meaning: binary predicate on **items**
  - will be used to link **objects**
- *i1* [*k* ≈ *k'*] *i2* if
  - *i1* satisfies *k*
  - *i2* satisfies *k'*
  - *i1* ≈ *i2*
- Example on string items, with substring and wildcard keywords and translation simrel:

  "Grand Place" [Grand trans *] "Grote Markt"

Linking objects using simlinks

- For objects *o1* and *o2,*
  o1 [*k* ≈ *k'*] *o2* if
  - *o1* contains some item *i1*
  - *o2* contains some item *i2*
  - *i1* [*k* ≈ *k'*] *i2*
- New associative search operator on dataspaces:
  LINK [*k* ≈ *k'*] (*S*)
  - retrieves all objects in the dataspace that are linked by [*k* ≈ *k'*] to some object in *S*

LINK [Grand trans *] (Markt)

Associative Search Language (ASL)

- BSL extended with link operator
- Parameterised by choice of:
  - keywords (already for BSL)
  - simrels (for link operator)
- What is the expressiveness of ASL?
- Link operator is like semijoin…

e1 AND LINK [θ] (e2)

$$e1 \ltimes_{\theta} e2$$

ASL on A–V dataspaces

- Keywords:
  - literals & wildcards

    (name: John) (name: *) (*: John)
  - negation on values

    (likes: ¬(Heineken,Budweiser))
  - negation on attributes

    (¬(paper_id,title): Orval)
  - negation on both values and attributes

    (¬(paper_id,title): ¬(Heineken,Budweiser))
- Simrels:
  - eq, eq_val, eq_att

Example query

- Retrieve all people located in Antwerp who have published a paper in *Ornithology:*

(location: Antwerp) AND

LINK [(paper: *) eq_val (paper_id: *)]

(journal: Ornithology)

- Which queries can we express?

A–V dataspace as relation

- We saw this already: set of triples (oid, att, val)

1	name	John
1	paper	p1
1	paper	p2
1	location	Namur
1	likes	Orval
2	name	Anne
2	paper	p1
2	location	Brussels
2	phone	022222785
3	name	Mary
3	paper	p2
3	paper	p3
3	location	Brussels
3	location	Antwerp
3	hobby	birdwatching
4	paper_id	p1
4	title	SQL
4	proceedings	VLDB
5	paper_id	p2
5	title	XQuery
5	proceedings	VLDB
5	citations	55
6	paper_id	p3
6	title	Pyrrhula song
6	journal	Ornithology
7	drink_type	beer
7	name	Orval
7	kind	Trappist

- How does ASL compare to querying this relation using relational algebra?

ASL translated into semijoin algebra

(location: Antwerp) AND

LINK [(paper: *) eq_val (paper_id: *)]

(journal: Ornithology)

$$\pi_{oid}\, \sigma_{\substack{att='location' \\ val='Antwerp'}}(T) \ltimes$$

$$\pi_{oid}\,(\sigma_{att='paper'}\,(T) \ltimes$$

$$\pi_{val}\, \sigma_{att='paper\_id'}\,(T \ltimes\ \pi_{oid}\, \sigma_{\substack{att='journal' \\ val='Ornithology'}}\,(T)))$$

- Only natural semijoins are used

SA queries not expressible in ASL

- “Retrieve all people who have the same value for a boss and a friend attribute”
- “Retrieve all people who like some beer that nobody else likes”
- Can prove that these are not expressible using **invariance under bisimulations**

Bisimilarity of Dataspaces

- Dataspace D and object o in D, also D’ and o’
- Natural number n
- We say that $(D,o) \rightleftarrows_n (D',o')$ if
  - o and o’ match precisely the same keywords
  - moreover for n>0:
  - for each simrel ≈ and for each object p in D such that o ≈ p, there exists p’ in D’ such that o’ ≈ p’ and $(D,p) \rightleftarrows_{n-1} (D',p')$
  - vice versa (from D’ to D)

Invariance under bisimilarity

- Let q be an ASL query using at most n nested link operators
- Let $(D,o) \rightleftarrows_n (D',o')$
- Then (D,o) is **indistinguishable** from (D',o'):
  - o in q(D) if and only if o' in q(D')
- (Converse holds as well: if indistinguishable, then bisimilar)

SA queries not expressible in ASL (repeated)

- “Retrieve all people who have the same value for a boss and a friend attribute”
- “Retrieve all people who like some beer that nobody else likes”
- Can prove that these are not expressible using **invariance under bisimulations**

The “search” fragment of SA

E ::= T
- | $\sigma_{att=c}(E)$
- | $\sigma_{val=c}(E)$
- | $E \cup E$
- | $E \setminus E$
- | $\pi_{\alpha}(E)$
- | $E \ltimes \pi_{oid}(E)$
- | $\pi_{oid}(E \ltimes \pi_{\beta}(E))$

- *c*: constant
- α: {oid}, {oid,att}, or {oid,val}
- β: {att}, {val}, or {att,val}

What have we learned?

- Searching unstructured information motivates to investigate new query languages
  - but the classical theory is still very useful:
    - relational databases
    - relational algebra
    - genericity
    - semijoin algebra
    - bisimilarity
- Querying RDF triple stores

Open research problems

- Algorithms, data structures for query processing
- Are BSL and ASL sufficient? Other search primitives?
- User interface: search should be easier than full querying in SQL
- How to represent relational databases as dataspaces (or RDF) such that querying can be done by searching?
  - Querying the Deep Web [Halevy]

Orval


ORVAL
TRAPPIST ALE
IMPORTERS MERCHANT DU VIN CORP. TUKWILA, WA
ORVAL
TRAPPIST ALE

Computability

- Of course a query q must be computable
- So, there must exist:
  - representation of databases into strings
  - algorithm A

Genericity: motivation

- Not just any crazy function is a “reasonable” database query
- E.g., random choice:
  - input: a list of names
  - output: one name from the list
- Better: minimum element query:
  - input: a list of names, and a total order over it
  - output: the minimum according to given order

Genericity: definition

- A query q is **generic** if it is invariant under isomorphisms
  - formally, for any permutation f of data values,

$$q(f(\mathrm{D})) = f(q(\mathrm{D}))$$

Not generic

- Random