Expressivity and Complexity of MongoDB Queries

Elena Botoeva, Diego Calvanese, Benjamin Cogrel, and Guohui Xiao
KRDB, Free University of Bozen-Bolzano, Italy *surname*@inf.unibz.it

MongoDB

- a popular document database
- stores collections of JSON-like documents with the identifier _id.

For instance, documents about prominent computer scientists in a **bios** collection:

```
{
  "_id": 4,
  "awards": [ {"award": "Rosing Prize", "year": 1999, "by": "Norwegian Data Association"},
              {"award": "Turing Award", "by": "ACM", "year": 2001},
              {"award": "IEEE John von Neumann Medal", "year": 2001, "by": "IEEE"} ],
  "birth": "1926-08-27",
  "contribs": ["OOP", "Simula"],
  "death": "2002-08-10",
  "name": {"first": "Kristen", "last": "Nygaard"}
}
```

We formalize JSON-documents as trees and define a relational view over them.

events collection:

```
{
  "_id": 1,
  "year": 1997,
  "event": "Deep Blue defeats Garry Kasparov"
}
{
  "_id": 2,
  "year": 1999,
  "event": "Melissa virus outbreak"
}
{
  "_id": 3,
  "year": 1999,
  "event": "Jeff Bezos is person of the year"
}
```

Aggregation Framework: MQuery

match → unwind → project → group → lookup

MongoDB provides a powerful querying mechanism in the form of the **aggregation framework**, where a query is a multi-stage pipeline evaluated over one collection.
We formalized this query language as MQuery, or $\mathcal{M}^{\text{mupgl}}$. It includes five stages:

```
db.bios.aggregate([
```

Match

$\mu_{criterion}$

```
{$match: {$and: [
    {"awards.year": {$eq: 1999}},
    {"name.first": {$eq: "Kristen"}}
]} },
```

_id	awards			birth	contribs	death	name.first	name.last
	award	year	by		lit			
4	Rosing Prize	1999	Norwegian Data Association	1926-08-27	OOP, Simula	2002-08-10	Kristen	Nygaard
	Turing Award	2001	ACM					
	IEEE John von Neumann Medal	2001	IEEE					

Selection σ

Unwind

ω_{path}

```
{$unwind: "$awards"},
```

_id	awards.award	awards.year	awards.by	birth	contribs	death	name.first	name.last
					lit			
4	Rosing Prize	1999	Norwegian Data Association	1926-08-27	OOP, Simula	2002-08-10	Kristen	Nygaard
4	Turing Award	2001	ACM	1926-08-27	OOP, Simula	2002-08-10	Kristen	Nygaard
4	IEEE John von Neumann Medal	2001	IEEE	1926-08-27	OOP, Simula	2002-08-10	Kristen	Nygaard

Unnest χ

Project

$\rho_{path/definition}$

```
{$project: {
    "awards": true,
    "firstName": "$name.first",
    "calledJohn":
      {$eq: ["$name.first", "John"]},
    "invisible": "$abc"
}},
```

_id	awards.award	awards.year	awards.by	firstName	calledJohn
4	Rosing Prize	1999	Norwegian Data Association	Kristen	false
4	Turing Award	2001	ACM	Kristen	false
4	IEEE John von Neumann Medal	2001	IEEE	Kristen	false

Extended projection π

Group

$\gamma_{grouping:aggregation}$

```
{$group: {
    "_id": {"year": "$awards.year"},
    "awardNames":
        {$addToSet: "$awards.award"}
}},
```

_id.year	awardNames
	lit
1999	Rosing Prize
2001	Turing Award, IEEE John von Neumann Medal

Nest ν

Lookup

$\lambda^{local=coll.foreign}_{result}$

```
{$lookup: {
    from: "events",
    localField: "_id.year",
    foreignField: "year",
    as: "joinedDocs"
}}
```

```
])
```

_id.year	awardNames	joinedDocs		
	lit	_id	year	event
1999	Rosing Prize	2	1999	Melissa virus outbreak
		3	1999	Jeff Bezos is person of the year
2001	Turing Award, IEEE John von Neumann Medal			

Left outer equijoin $⟕$

Expressivity of MQuery

We have shown that **well-typed** MQuery is equivalent to nested relational algebra (NRA):

- well-typed $\mathcal{M}^{\text{mupg}} \equiv$ NRA over a single collection, hence it is possible to express **joins** without lookup
- well-typed $\mathcal{M}^{\text{mupgl}} \equiv$ NRA

Well-typedness is required as arbitrary MQueries may produce forests for which a relational view cannot be defined.

Complexity of MQuery (and NRA)

Fragment	Query complexity	Combined complexity
$\mathcal{M}^{\text{m}}$	LogSpace-complete	
$\mathcal{M}^{\text{mp}}$, $\mathcal{M}^{\text{mpgl}}$	PTime-complete	
$\mathcal{M}^{\text{mu}}$	LogSpace-complete	NP-complete
$\mathcal{M}^{\text{mup}}$, $\mathcal{M}^{\text{mul}}$, $\mathcal{M}^{\text{mupl}}$	NP-complete	
$\mathcal{M}^{\text{mug}}$	PSpace-hard	
$\mathcal{M}^{\text{mupg}}$,$\mathcal{M}^{\text{mupgl}}$	TA$[2^{n^{O(1)}}, n^{O(1)}]$-complete*	
NRA	TA$[2^{n^{O(1)}}, n^{O(1)}]$-complete	

∗ The class of problems solvable by an alternating Turing machine running in exponential time with polynomially many alternations.