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:

```json
{ "_id": 4, "awards": [ { "award": "Rosing Prize", "year": 1999, "by": "Norwegian Data Association" }, { "award": "IEEE John von Neumann Medal", "year": 2001, "by": "IEEE" } ], "birth": "1926-08-27", "name": { "first": "Kristen", "last": "Nygaard" } }
```

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

**Aggregation Framework: MQuery**

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 M^{\text{query}}. It includes five stages:

1. **Match**
   ```json
   db.bios.aggregate(
     { $match: { $and: [ { award: "Rosing Prize" }, { year: 1999 }, { by: "Norwegian Data Association" } ] } }
   )
   ```

2. **Unwind**
   ```json
   db.bios.aggregate(
     { $unwind: "$awards" },
     { $lookup: { from: "events", localField: "_id", foreignField: "year", as: "joinedDocs" } }
   )
   ```

3. **Project**
   ```json
   db.bios.aggregate(
     { $project: { firstName: true, first_name: "$name.first", calledJohn: { $cond: { if: { $eq: [ "$name.first", "John" ] } } } } }
   )
   ```

4. **Group**
   ```json
   db.bios.aggregate(
     { $group: { _id: "$year", awards: "$awards.year" } }
   )
   ```

5. **Lookup**
   ```json
   db.bios.aggregate(
     { $lookup: { from: "events", localField: "_id.year", foreignField: "year", as: "joinedDocs" } }
   )
   ```

**Expressivity of MQuery**

We have shown that well-typed M^{\text{query}} is equivalent to nested relational algebra (NRA):

- well-typed M^{\text{query}} = NRA over a single collection.
  Hence it is possible to express joins without lookup.
- well-typed M^{\text{query}} = NRA

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

**Complexity of MQuery (and NRA)**

<table>
<thead>
<tr>
<th>Fragment</th>
<th>Query complexity</th>
<th>Combined complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>M^{\text{query}}</td>
<td>LogSpace-complete</td>
<td>( \text{PTime-complete} )</td>
</tr>
<tr>
<td>M^{\text{query}}</td>
<td>LogSpace-complete</td>
<td>( \text{NP-complete} )</td>
</tr>
<tr>
<td>M^{\text{query}}</td>
<td>LogSpace-complete</td>
<td>( \text{PSpace-hard} )</td>
</tr>
<tr>
<td>NRA</td>
<td>TA(2^{\text{O(n)}}, \text{poly})-complete</td>
<td>TA(2^{\text{O(n)}}, \text{poly})-complete</td>
</tr>
</tbody>
</table>

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