Query Combinators for Medical Research and Decision Support

an algebraic theory of database queries applied to medicine

Clark C. Evans ⟨cce@clarkevans.com⟩,
Kyrylo Simonov ⟨xi@resolvent.net⟩

Tuesday, May 28, 2019
OHSDI Community Meeting

Prometheus Research, LLC
1. Introduction

2. Example: Complex Query

3. Thinking in Query Combinators

4. Example: Feasibility Assessment

5. Conclusion
Introduction
Figure 1: Clinical Research Workflow as inspired from Hruby’s observations at Columbia University [2]
Which anti-hypertensive medications are effective in improving blood pressure?

```
SELECT *
FROM patient
JOIN observation
ON (...)  
WHERE ...
```

```
d=read.csv(...)  
...  
a.lm=lm(...,data=d)  
summary(a.lm)
$ r.squared
```
Is Shared Query Infrastructure Is Possible?

Principal Investigator

Informatician

Statistician

Shared & intelligible executable specification
Example: Complex Query
Consider the inquiry, "Which anti-hypertensive medications are effective in improving blood pressure?". This inquiry could be operationalized as:

*Within 6 months of a hypertension diagnosis, when an anti-hypertensive medication was added or intensified, was there a blood pressure decrease of 5 mmHg or more within 5 days after the medication adjustment?*
Consider the inquiry, "Which anti-hypertensive medications are effective in improving blood pressure?". This inquiry could be operationalized as:

*Within 6 months of a hypertension diagnosis, when an anti-hypertensive medication was added or intensified, was there a blood pressure decrease of 5 mmHg or more within 5 days after the medication adjustment?*
What are the query components?

The first thing to do is convert specialized vocabulary in this inquiry into query component definitions in a *query mediation* session.

<table>
<thead>
<tr>
<th>Component</th>
<th>Mediation Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>hypertension_diagnosis</td>
<td>exclude pregnancy &amp; kidney failure</td>
</tr>
<tr>
<td>antihypertensive_medication</td>
<td>a product list is provided</td>
</tr>
<tr>
<td>added_or_intensified</td>
<td>new therapy or larger dose</td>
</tr>
<tr>
<td>blood_pressure_decrease</td>
<td>of both systolic &amp; diastolic</td>
</tr>
<tr>
<td>medication_adjustment</td>
<td>change of daily medication</td>
</tr>
<tr>
<td>active_ingredient</td>
<td>normalize dosage records</td>
</tr>
<tr>
<td></td>
<td>across compound products</td>
</tr>
</tbody>
</table>

**Table 1:** Anti-hypertensive Query Components
patient.keep(it)
antihypertensive_medication
active_ingredient
medication_adjustment
filter(added_or_intensified &
  during(previous(6months), patient.hypertension_diagnosis)
collect(is_effective =>
  during(subsequent(5days),
    patient.blood_pressure_decrease(5mmHg)))
group(active_ingredient)
{  active_ingredient,
   count(medication_adjustment.filter(is_effective)),
   count(medication_adjustment.filter(not(is_effective))) }
This query brings together many things, including:

- query composition algebra;
- built-in combinators, such as filter, collect, group, keep, count, etc.;
- data source queries, including patient and medication;
- domain specific queries, such as medication_adjustment, active_ingredient, and blood_pressure_decrease; and
- domain specific combinators, such as during and subsequent;

The domain specific queries and combinators are then independently defined, constructed, documented, and tested. They can be reused across questions and reflect the shared vocabulary for the research team.
Thinking in Query Combinators
Tabular Model of Clinical Research Data Repository

**Figure 2:** Tabular Model for CRDR

### Patient
- **identifier**: PK, Integer
- **birthdate**: NN, DateTime

### Condition
- **patient_id**: FK, Integer
- **category**: NN, Text
- **onset**: NN, DateTime
- **abatement**: NN, DateTime
Figure 3: Hierarchical Model for CRDR
Example Queries

- patient
- count(patient)
- patient.condition
- patient.count(condition)
- mean(patient.count(condition))
Query Combinators are an algebra of query functions.

- This algebra’s elements, or *queries*, represent relationships among class entities and datatypes.
- This algebra’s operations, or *combinators*, are applied to construct query expressions.

Query expressions, such as `count(condition)` are constructed by applying combinators, such as `count` to queries, such as `condition`. 
Figure 4: Functional Model for CRDR
Table 2: Query Primitives for CRDR

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>patient</td>
<td>Database → Patient*</td>
</tr>
<tr>
<td>identifier</td>
<td>Patient → Integer</td>
</tr>
<tr>
<td>birthdate</td>
<td>Patient → DateTime</td>
</tr>
<tr>
<td>condition</td>
<td>Patient → Condition*</td>
</tr>
<tr>
<td>category</td>
<td>Condition → Text</td>
</tr>
<tr>
<td>onset</td>
<td>Condition → DateTime</td>
</tr>
<tr>
<td>abatement</td>
<td>Condition → DateTime?</td>
</tr>
</tbody>
</table>
The Count Combinator

\[
\begin{array}{ccc}
  f & A \rightarrow B^* \\
  \text{count}(f) & A \rightarrow \text{Integer} \\
\end{array}
\]

\[
\begin{array}{ccc}
  \text{patient} & \text{Database} \rightarrow \text{Patient}^* \\
  \text{count}(\text{patient}) & \text{Database} \rightarrow \text{Integer} \\
\end{array}
\]

\[
\begin{array}{ccc}
  \text{condition} & \text{Patient} \rightarrow \text{Condition}^* \\
  \text{count}(\text{condition}) & \text{Patient} \rightarrow \text{Integer} \\
\end{array}
\]
The Composition Combinator

\[
\begin{array}{c}
f \quad A \rightarrow B^* \\
g \quad B \rightarrow C^* \\
f \cdot g \quad A \rightarrow C^*
\end{array}
\]

\[
\begin{array}{ll}
\text{patient} & \text{Database} \rightarrow \text{Patient}^* \\
\text{condition} & \text{Patient} \rightarrow \text{Condition}^* \\
\text{patient.condition} & \text{Database} \rightarrow \text{Condition}^* \\
\text{condition} & \text{Patient} \rightarrow \text{Condition}^* \\
\text{category} & \text{Condition} \rightarrow \text{Text}^* \\
\text{condition.category} & \text{Patient} \rightarrow \text{Text}^*
\end{array}
\]
Example: Feasibility Assessment
Suppose that an informatician would like to conduct a feasibility assessment to see if the CRDR database has at least some candidate patients relevant to this hypertension effectiveness inquiry.

How many patients, ages 18 or older, have an active diagnosis of Essential Hypertension?
Components of Feasibility Assessment

How many patients, ages 18 or older, have an active diagnosis of *Essential Hypertension*?

<table>
<thead>
<tr>
<th>Component</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>essential_hypertension</td>
<td>’59621000’</td>
</tr>
<tr>
<td>age</td>
<td>years(now() − birthdate)</td>
</tr>
<tr>
<td>has_active_diagnosis(x)</td>
<td>exists(condition.filter( category = x &amp; is_null(abatement)))</td>
</tr>
</tbody>
</table>

**Table 3:** Component Definitions for Feasibility Assessment
How many patients, ages 18 or older, have an active diagnosis of Essential Hypertension?

patient
filter  (age >= 18
       &  has_active_diagnosis(
           essential_hypertension))
count()
Conclusion
There is an implementation of Query Combinators for the Julia Language, called DataKnots.jl.

- this implementation is MIT/Apache licensed
- it includes an in-memory, column-oriented database
- it has adapters to CSV (and soon XML, JSON)
- essential query operators are implemented
- Julia statistics can be lifted to a combinator
- an adapter to SQL datasources is in progress!

https://github.com/rbt-lang/DataKnots.jl
• Thanks to James Shalaby, Pharm.D. for his hypertension research question and for query mediation discussion.
• Thanks to Simons Foundation for years of funding for earlier variants of this initiative, called HTSQL.
C. C. Evans and K. Simonov.  
**Query combinators.**  
2017.

G. Hruby, J. McKiernan, S. Bakken, and C. Weng.  
**A centralized research data repository enhances retrospective outcomes research capacity: A case report.**  
20, 01 2013.