Visual Query Combinators for BioMedical Query Mediation

Clark Evans, Charles Tirrell, Oleksiy Golovko, Andrey Popp, Jason Simeone, Kyrylo Simonov, PhD, Leon Rozenblit, JD, PhD
Prometheus Research, LLC, New Haven, CT

INTRODUCTION

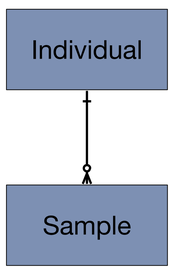
Even as informaticists continue to deliver better and larger central data stores to support research, three important goals have remained out of reach for biomedical research teams: (1) the ability for all team members, including ones who are not database specialists, to proficiently explore data stores for information relevant to their research questions; (2) frictionless sharing of resulting queries to serve as documentation for further refinement and discussion; (3) deep integration of meaningful statistical analysis with the queries. Lacking these capabilities, research teams must involve database specialists to mediate even the most basic database queries and to periodically refresh results. We evaluate whether all three goals can be achieved, and the need for mediation reduced, through an innovative query system.

INNOVATION

Prometheus Research has designed a combinator-based database query model ("QueryCombinators") [1] and released an open source functional prototype [2]. QueryCombinators is a practical, declarative query language designed for domain experts. Unlike SQL and LINQ, the QueryCombinator model lends itself to dynamic, incremental, visual query construction. Moreover, the QueryCombinator model flexibly accommodates all sorts of statistical and domain specific operators. QueryCombinator notation can be understood and shared among all research team members, reducing the need for upstream query mediation. We include a visual query tool based on QueryCombinators in our open source RexDB system [3].

Combinator pattern for query construction: Functional Algebra

Example Entity-Relationship Diagram



Queries are functions having an input domain and output range (which can be plural, indicated here with an asterix):

`individual`: *Database* ⇒ *Individual**
`sample`: *Individual* ⇒ *Sample**
`sample_type`: *Sample* ⇒ *Text*

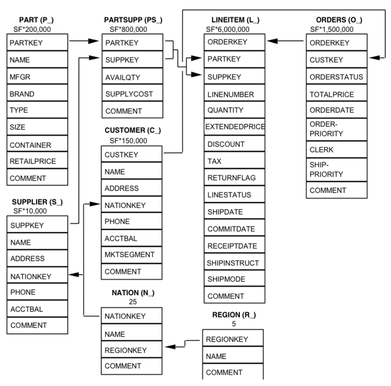
Combinators construct queries from others.

`compose`(*A* ⇒ *B**, *B* ⇒ *C**) : *A* ⇒ *C**
`count`(*A* ⇒ *B**) : *A* ⇒ *Integer*
`filter`(*A* ⇒ *B**, *B* ⇒ *Boolean*) : *A* ⇒ *B**

A combined query listing individuals with at least one sample:

`filter(individual, count(sample)>0)`
: *Database* ⇒ *Individual**

In this functional algebra, data navigation queries are seen as primitive functions and various combinators provide ways to construct new queries from existing ones. This approach supports incremental, interactive construction of complex queries, and enables visual query builder tools such as the RexStudy Visual Query Builder.

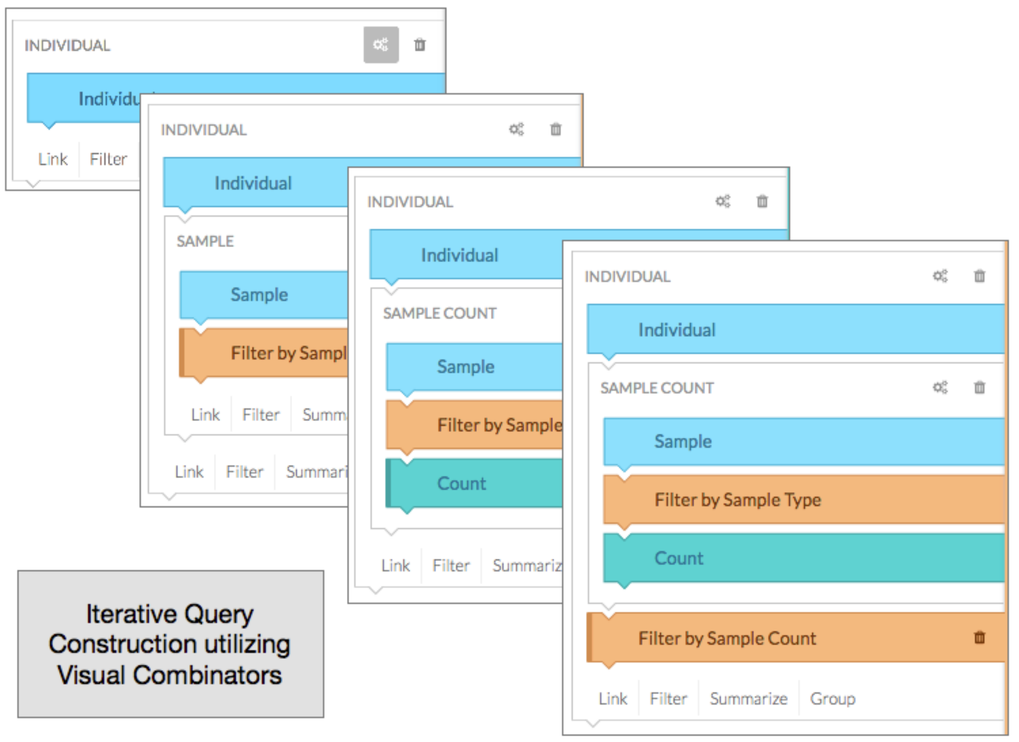


Standard schema used for evaluation: TCP-H benchmark data [4]

METHODS

To validate our claims about the completeness and learnability of this visual query system, we conducted usability tests on a convenience sample of participants that included experienced data analysts and novice users. Each subject was given an unfamiliar data model and 3 minutes to explore the query interface. Subjects then answered a series of challenging timed questions about the data. For example, subjects were asked to select a table, add a column from a related table, perform a maximum value aggregation on the related table, and filter the primary table records based on this aggregate value.

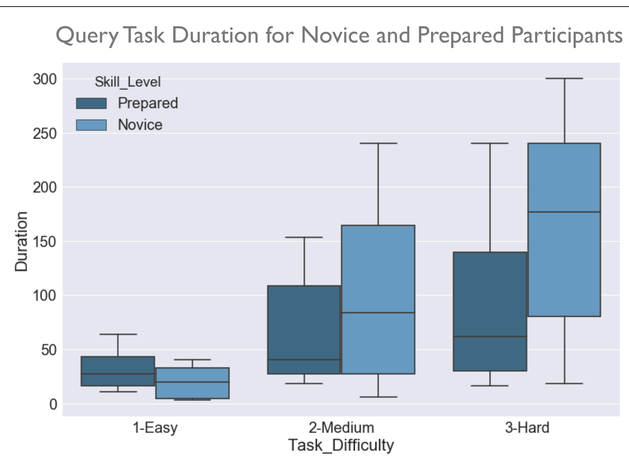
"Scratch-like" UI for Combinator-based Query Construction



Iterative Query Construction utilizing Visual Combinators

RESULTS

Novice query users with no SQL experience were able to generate complex queries using our interface, including multi-table joins with aggregation, advanced filtering, and distinct operations. Novice query users were able to complete an average of 71% of the 15 assigned tasks in under three minutes. We found that prepared query users with minimal SQL experience were able to complete 93.3% of the assigned tasks in under three minutes. Exceeding our expectations, we found that novice users were only 53.5% slower, on average, than more experienced data analysts.



Examples of Query Tasks

Easy
Select a table and filter by a column value. "We would like you to create a report that lists all customers with an account balance over 9000."

Medium
Select a table and return count of all records in it. "We would like you to query the total number of customers."

Hard
Select a table, add a sub table, and aggregate on the sub table. "We would like you to create a report of all nations and the maximum account balance over all customers in that nation."

**Novice users had no experience with querying databases; Prepared users had experience querying databases using various interfaces, but were new to the VQB tool. Neither group was familiar with the data model.*

CONCLUSION

The preliminary results suggest that a combinator-based visual query builder can empower a broader range of users to retrieve complex data from rich data stores. We expect to demonstrate in future research that this approach also facilitates sharing and refinement of queries, as well as the integration of statistical procedures into the queries themselves. Because our implementation uses a statistical processing language, Julia, deep integration of querying with with statistical analysis is technically straightforward.

References

1. Evans CE, Simonov K (2017) "Query Combinators" **arXiv:1702.08409**
2. QueryCombinator Language https://github.com/rbt-lang/
3. RexDB Database Platform http://rexdb.org
4. TPC Benchmark H (Decision Support) Standard Specification Revision 2.14.0, www.tpc.org

PROMETHEUS | RESEARCH