



# The MITRE Corporation

Andrew Gregorowicz and Marc Hadley

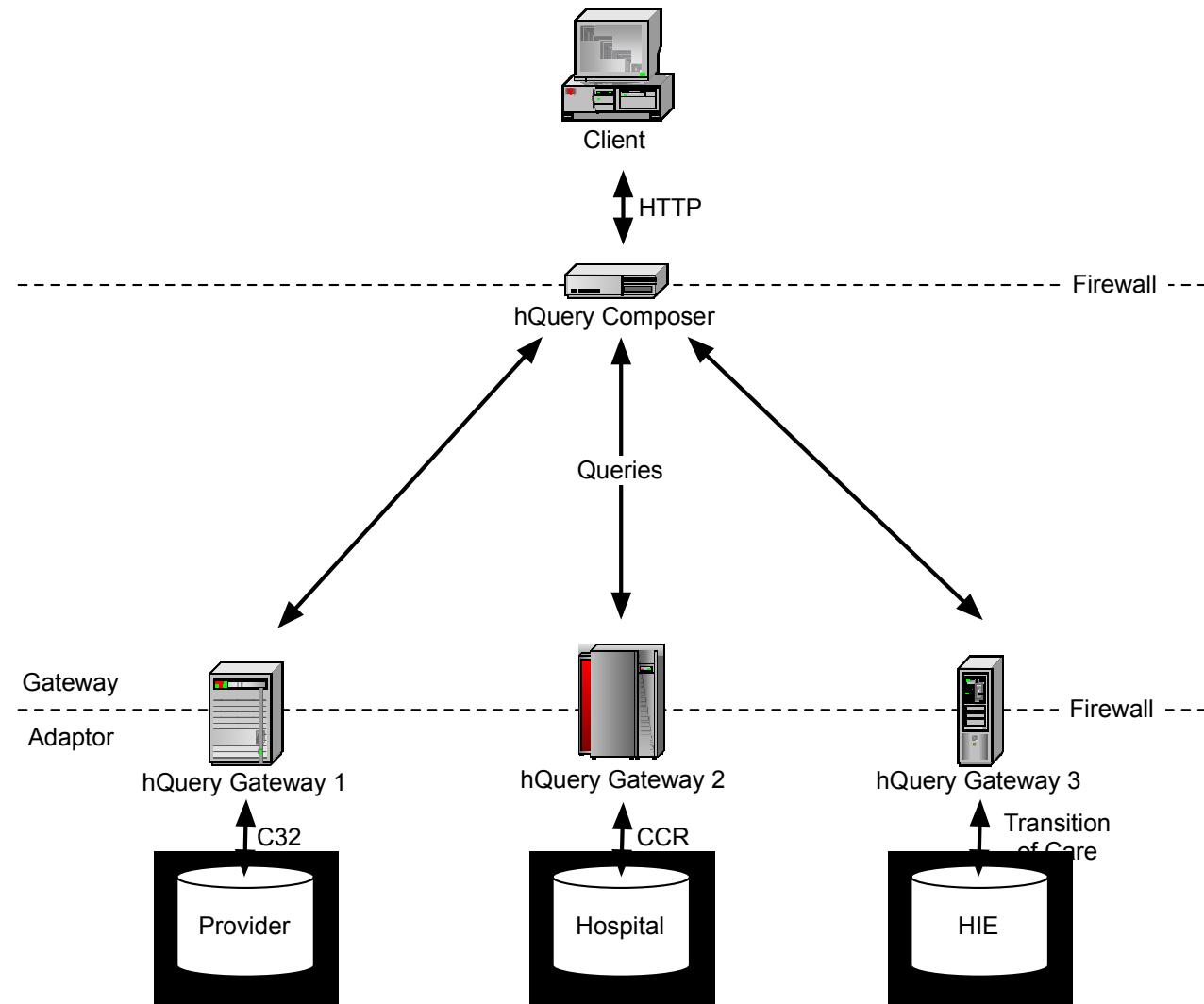
August 8th, 2011



# Background

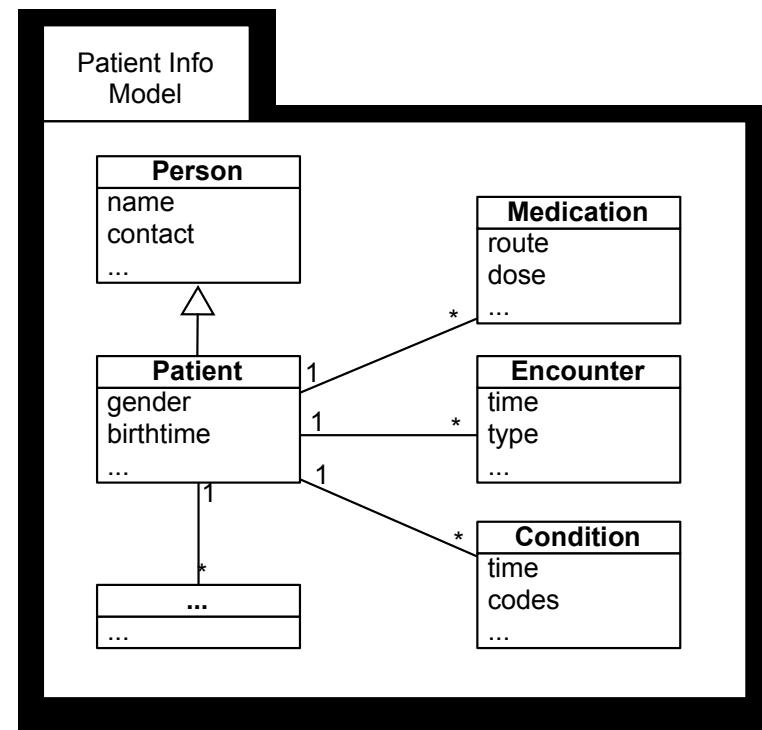
- Sponsored by Office of the National Coordinator for Health IT (ONC)
- Building on experience in healthcare data standards and population health reporting
  - Laika: <http://laika.sourceforge.net/>
  - hData: <http://www.projecthdata.org/>
  - popHealth: <http://projectpophealth.org/>
- Initial proof-of-concept to generalize popHealth architecture for ad-hoc distributed queries
  - Started May 20th 2011
  - Finishing end of September, 2011
- Agile PM, fifth sprint currently underway

# High Level Architecture

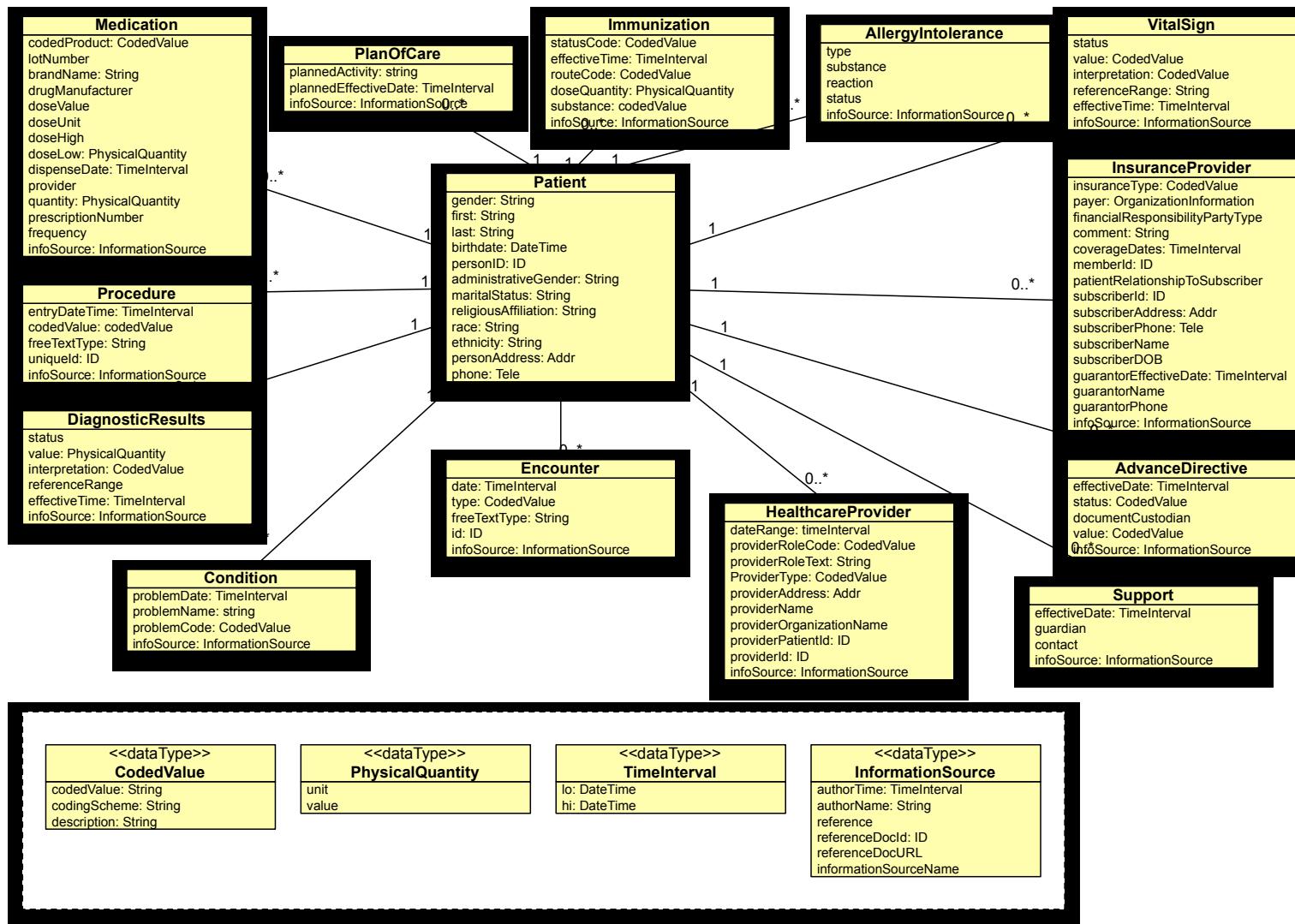


# Patient Information Model

- Standard information model exposed to queries
- Independent of backing data source (C32, CCR, RDBMS, ...)
- Enables generic queries regardless of backing data source
- Tuned for ease of querying



# Information Model Based on Green CDA for C32

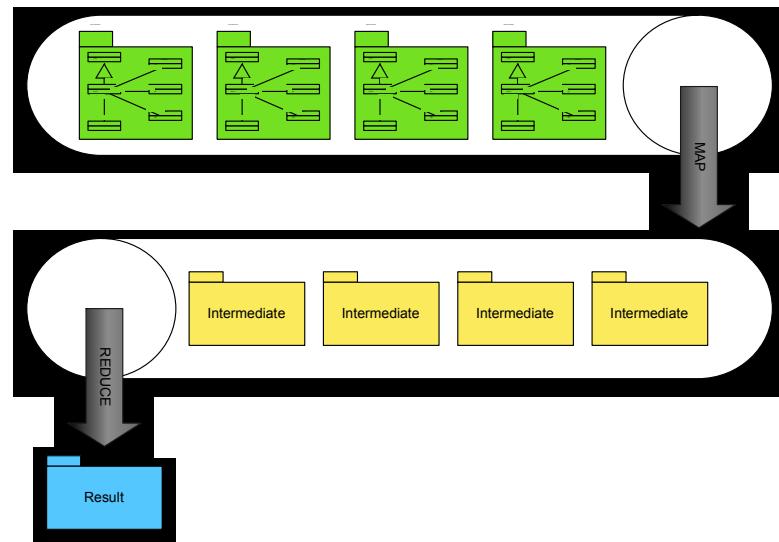


# Information Model Properties

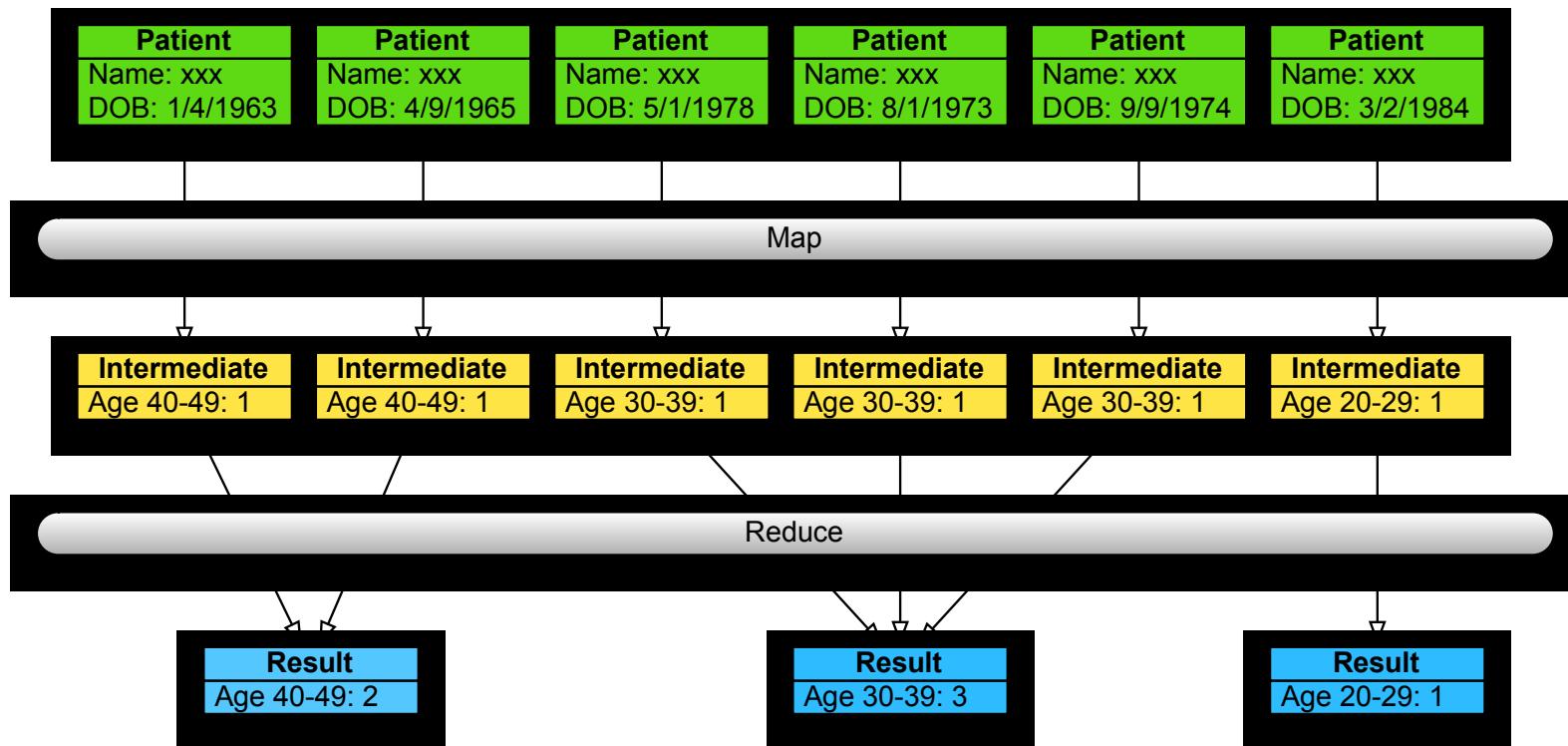
- **Easy to understand and navigate**
  - Balance of specificity and abstraction
- **Extensible via coding, but**
  - Code set and code dependent
  - Still a challenge for semantic interoperability
- **Based on patient summary information model**
- **May not currently include required detail for all queries, e.g.:**
  - Communications
  - Social and family history
  - Devices/medical equipment
  - Functional status
- **Reliance on availability of coded data**

# Map-Reduce Query Engine

- **Two-part queries**
  - Map takes an instance of patient data model and produces an intermediate data structure
  - Reduce takes a set of intermediate data structures and produces a final result
- **Powerful algorithm**
  - Highly concurrent
  - Easily distributed
- **Queries define their own output data structure**
  - Ad-hoc queries fully supported



# Map-Reduce Example: Stratifying Patients by Age



# Query Format

- **Query consists of JavaScript Map and Reduce functions wrapped in a MIME package**
- **Patient information model is presented to Map function as an object-oriented API that includes:**
  - Raw data (e.g., patient date of birth)
  - Convenience functions to simplify queries (e.g., get patient age on specified date)
- **Framework executes map function on each patient and summarizes results using reduce function**
- **Declarative pre-filtering for efficiency**
- **Required libraries included with query**

# Why JavaScript ?

- **ECMA International Standard**
- **Ubiquitous**
  - Implementations for all major hardware and software platforms
  - Already widely used in “big-data” technologies: MongoDB, CouchDB, Riak, ...
- **“Assembly language of the Web”**
  - Fast and getting faster
  - Full programming language with many useful libraries
  - Easy to build concise but arbitrarily complex queries
  - Craft manually or compile from another language
- **Readily sandboxed**
  - Building on 15+ years of Web browser usage

# Simple Example: Stratifying Patients by Age

```
function map(patient) {  
    var when = new Date(2011, 6, 1);  
    var decade = Math.floor(patient.age(when)/10)  
        *10;  
    emit(decade, 1);  
}  
  
function reduce(decade, counts) {  
    var sum = 0;  
    for(var i in counts)  
        sum += counts[i];  
    return sum;  
}
```

# Full Example: NQF 43 - Pneumonia Vaccination Status for Older Adults

```
function map(patient) {  
  
    var outpatientEncounterCodes = {  
        "CPT": ["99201", ...],  
        "ICD-9-CM": ["V70.0", ...]  
    };  
  
    var pneumococcalMedicationCodes = {  
        "RxNorm": ["854931", ...]  
    };  
  
    var pneumococcalProcedureCodes = {  
        "CVX": ["33", ...],  
        "CPT": ["90669", ...]  
    };  
  
    var start = new Date(2010, 1, 1);  
    var end = new Date(2010, 12, 31);
```

# Full Example continued

```
function population(patient) {
    return (patient.age(start)>=64);
}

function denominator(patient) {
    var encounters = patient.encounters().match(
        outpatientEncounterCodes, start, end);
    return (encounters>0);
}

function numerator(patient) {
    var medication = patient.medications().match(
        pneumococcalMedicationCodes, null, end);
    var procedure = patient.procedures().match(
        pneumococcalProcedureCodes, null, end);
    return medication || procedure;
}

function exclusion(patient) {
    return false;
}
```

# Full Example continued

```
if (population(patient)) {  
    emit("population", 1);  
    if (denominator(patient)) {  
        if (numerator(patient)) {  
            emit("denominator", 1);  
            emit("numerator", 1);  
        } else if (exclusion(patient)) {  
            emit("exclusion", 1);  
        } else {  
            emit("denominator", 1);  
        }  
    }  
}
```

# JavaScript vs eMeasure: NQF 43 Population

```
function population(patient) {  
    return (patient.age(start)>=64);  
}
```

```
<entry typeCode="DRIV">  
    <observation classCode="OBS" moodCode="EVN.CRT"  
        isCriterionInd="true">  
        <id root="4AAEF95D-DCC6-459C-839C-C820DF310D60"/>  
        <code code="ASSERTION" codeSystem="2.16.840.1.113883.5.4"/>  
        <value xsi:type="CD" code="IPP"  
            codeSystem="2.16.840.1.113883.5.1063"  
            codeSystemName="HL7 Observation Value"  
            displayName="Initial Patient Population"/>  
        <sourceOf typeCode="PRCN">  
            <conjunctionCode code="AND"/>  
            <act classCode="ACT" moodCode="EVN" isCriterionInd="true">  
                <templateId root="2.16.840.1.113883.3.560.1.25"/>  
                <id root="52A541D7-9C22-4633-8AEC-389611894672"/>  
                <code code="45970-1" displayName="Demographics"  
                    codeSystem="2.16.840.1.113883.6.1"/>  
                <sourceOf typeCode="COMP">  
                    <observation classCode="OBS" moodCode="EVN"  
                        isCriterionInd="true">  
                        <code code="2.16.840.1.113883.3.464.0001.14"  
                            displayName="birth date HL7 Code List"/>  
                        <title>Patient characteristic: birth date</title>  
                        <sourceOf typeCode="SBS">  
                            <pauseQuantity xsi:type="IVL_PQ">  
                                <low value="64" unit="a" inclusive="true"/>  
                            </pauseQuantity>  
                            <observation classCode="OBS" moodCode="EVN">  
                                <id root="F8D5AD22-F49E-4181-B886-E5B12BEA8966"/>  
                                <title>Measurement period</title>  
                                </observation>  
                            </sourceOf>  
                            </observation>  
                        </sourceOf>  
                    </act>  
                    </sourceOf>  
                </observation>  
            </entry>
```

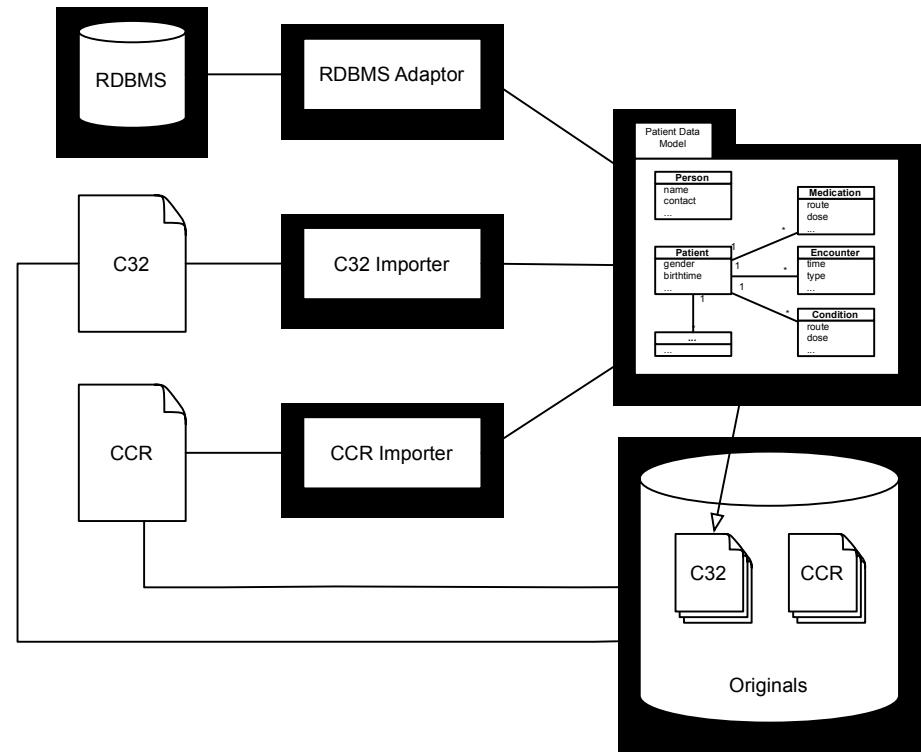
# JavaScript vs eMeasure: NQF 43 Numerator

```
function numerator(patient) {
    var medication = patient.medications().match(
        pneumococcalMedicationCodes,
        null, end);
    var procedure = patient.procedures().match(
        pneumococcalProcedureCodes,
        null, end);
    return medication || procedure;
}
```

```
<entry typeCode="DRIV">
<observation classCode="OBS" moodCode="EVN.CRT" isCriterionInd="true">
<id root="4616A371-29FB-46CA-A9E4-8F7FF1DBEDDF"/>
<code code="ASSERTION" codeSystem="2.16.840.1.113883.5.4"/>
<value xsi:type="CD" code="NUMER" codeSystem="2.16.840.1.113883.5.1063"
codeSystemName="HL7 Observation Value" displayName="Numerator"/>
<sourceOf typeCode="PRCN">
<conjunctionCode code="AND"/>
<act classCode="ACT" moodCode="EVN" isCriterionInd="true">
<sourceOf typeCode="PRCN">
<conjunctionCode code="OR"/>
<act classCode="ACT" moodCode="EVN" isCriterionInd="true">
<templateId root="2.16.840.1.113883.3.560.1.14"/>
<id root="10165EC8-53EE-4242-A20D-B1D21CE0DC73"/>
<code code="18610-6" displayName="Medication administered"
codeSystem="2.16.840.1.113883.6.1"/>
<sourceOf typeCode="COMP">
<substanceAdministration classCode="SBADM" moodCode="EVN" isCriterionInd="true">
<title>Medication administered: Pneumococcal Vaccine all ages</title>
<participant typeCode="CSM">
<roleParticipant classCode="MANU">
<playingMaterial classCode="MMAT" determinerCode="KIND">
<code code="2.16.840.1.113883.3.464.0001.430"
displayName="Pneumococcal Vaccine all ages Code List GROUPING"/>
</playingMaterial>
</roleParticipant>
</participant>
</substanceAdministration>
</sourceOf>
</act>
</sourceOf>
<sourceOf typeCode="PRCN">
<conjunctionCode code="OR"/>
<act classCode="ACT" moodCode="EVN" isCriterionInd="true">
<templateId root="2.16.840.1.113883.3.560.1.6"/>
<id root="482902EC-E214-4FB4-8C5A-85A41250573C"/>
<code code="47519-4" displayName="Procedures" codeSystem="2.16.840.1.113883.6.1"/>
<sourceOf typeCode="COMP">
<procedure classCode="PROC" moodCode="EVN" isCriterionInd="true">
<code code="2.16.840.1.113883.3.464.0001.143"
displayName="Pneumococcal Vaccination all ages Code List GROUPING"/>
<title>Procedure performed: Pneumococcal Vaccination all ages</title>
</procedure>
</sourceOf>
</act>
</sourceOf>
<sourceOf typeCode="DURING">
<observation classCode="OBS" moodCode="EVN" isCriterionInd="true">
<id root="F8D5AD22-F49E-4181-B886-E5B12BEA8966"/>
<title>Measurement period</title>
</observation>
</sourceOf>
</act>
</sourceOf>
</observation>
</entry>
```

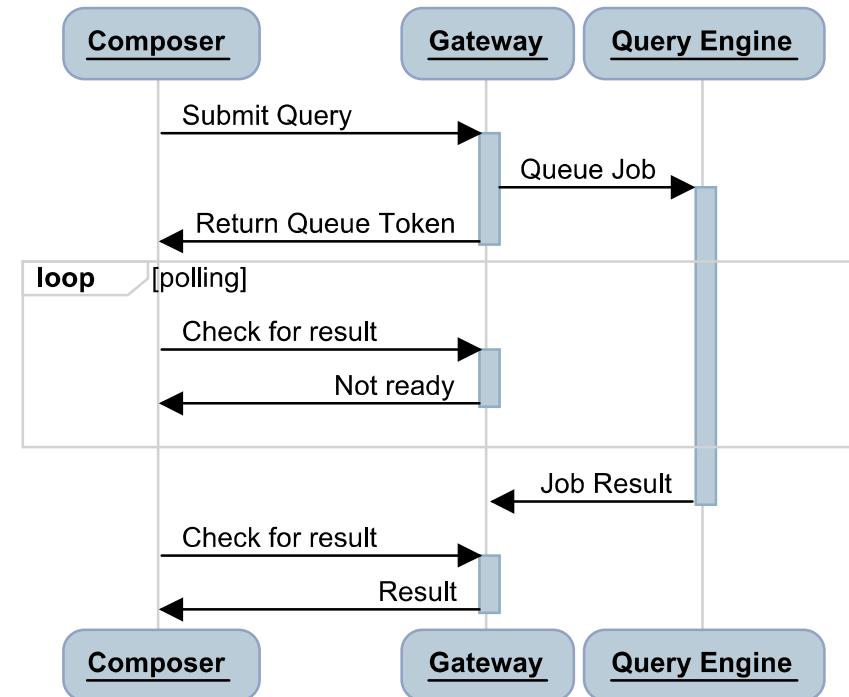
# Gateway Data Import and Transformation

- Data is transformed into patient information model
  - Statically on import
  - Dynamically during query execution
- Static Import
  - Transform C32 or CCR XML into JSON model
  - Original data preserved for provenance and linked to patient information model instance
- Dynamic Import
  - Requires adaptor to native data format



# Asynchronous Query Submission

- **Queries can take a long time to execute**
  - Large data sets
  - Server loading
- **Asynchronous query submission and result retrieval required**
  - Polling vs notification
- **RESTful Web service for submitting queries and obtaining results**



# **hQuery Composer Web Application**

- **Configure hQuery gateways**
- **Build and execute queries**
  - Basic query syntax checking
- **Monitor in-progress queries, maintain audit history**
- **Combine query results from multiple endpoints**
- **Query test harness**
  - Simulated in-browser gateway with sample patient data
  - Query debugging

# Developer UI: Dashboard

The screenshot shows a web-based developer interface titled "QueryComposer". The URL in the browser bar is <http://127.0.0.1:3000/>. The top right corner displays a welcome message: "Welcome, Marc | help | account | admin | logout". On the left, there is a vertical navigation menu with four items: "Queries" (selected), "Endpoints", "Query Admin", and "Library Functions". The main content area is titled "Queries" and lists four entries:

Title	Description	Status
<a href="#">Patients by Surname</a>	Count of patients by first letter of surname	Complete (1 of 1) <input checked="" type="button"/>
<a href="#">Patients by Gender</a>	Count of patients of each gender	Complete (1 of 1) <input checked="" type="button"/>
<a href="#">NQF 43</a>	Pneumonia Immunization for Patients >= 65 Years Old	Complete (1 of 1) <input checked="" type="button"/>
<a href="#">Patients by decade of birth</a>	Count of patients by decade of birth	Complete (1 of 1) <input checked="" type="button"/>

A blue "ADD QUERY" button is located at the bottom left of the main content area.

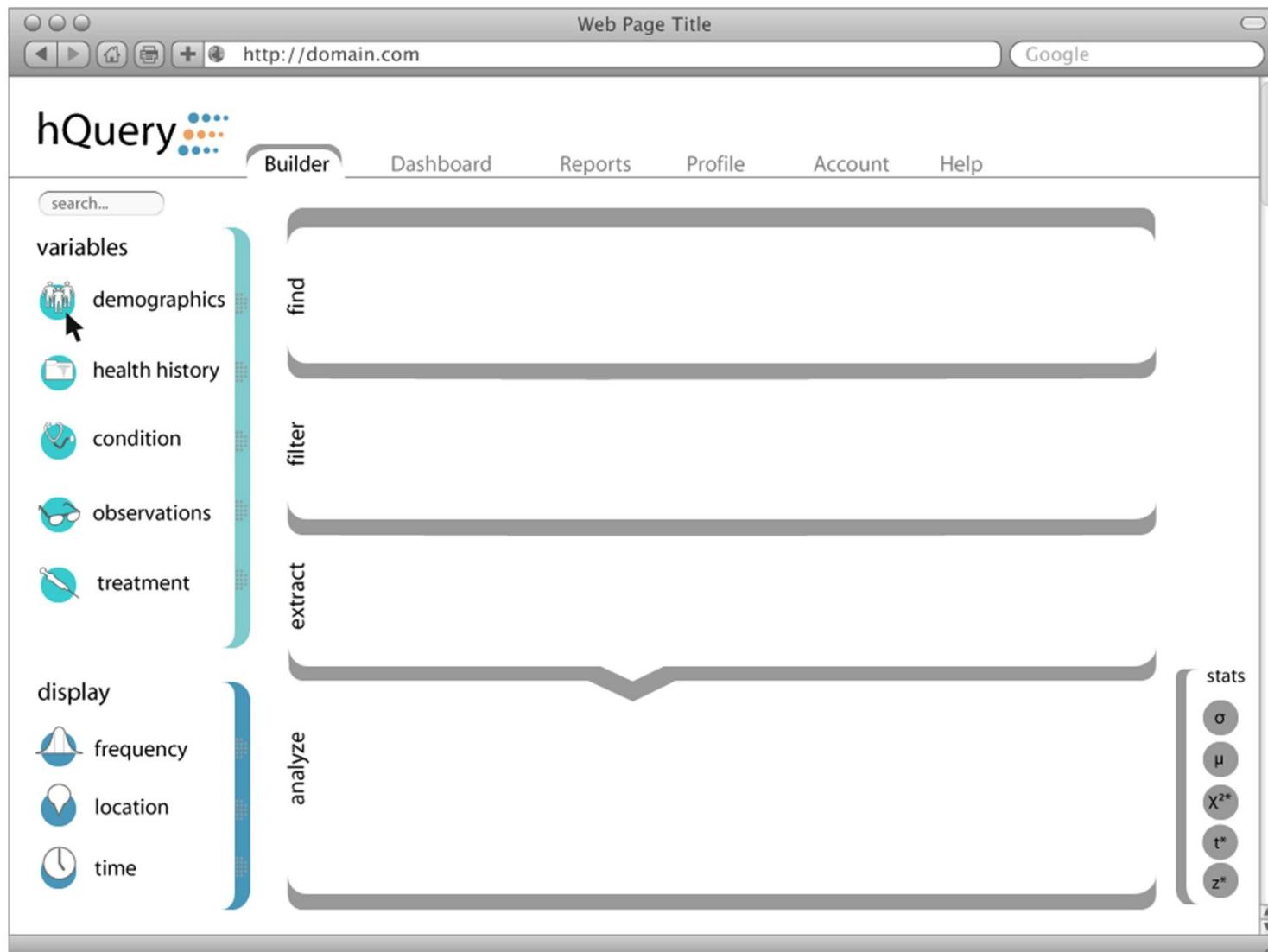
# Developer UI: Query Execution

The screenshot shows a web-based developer interface for executing queries. The title bar reads "QueryComposer" and the address bar shows the URL <http://127.0.0.1:3000/queries/4df775bd4f85cf6558000002>. The top navigation bar includes links for "Home", "Queries", "Patients by Gender", "Welcome, Marc", "help", "account", "admin", and "logout". On the left, a sidebar menu has "Queries" selected, with other options like "Endpoints", "Query Admin", and "Library Functions". The main content area is titled "Query Definition" and displays the following details:

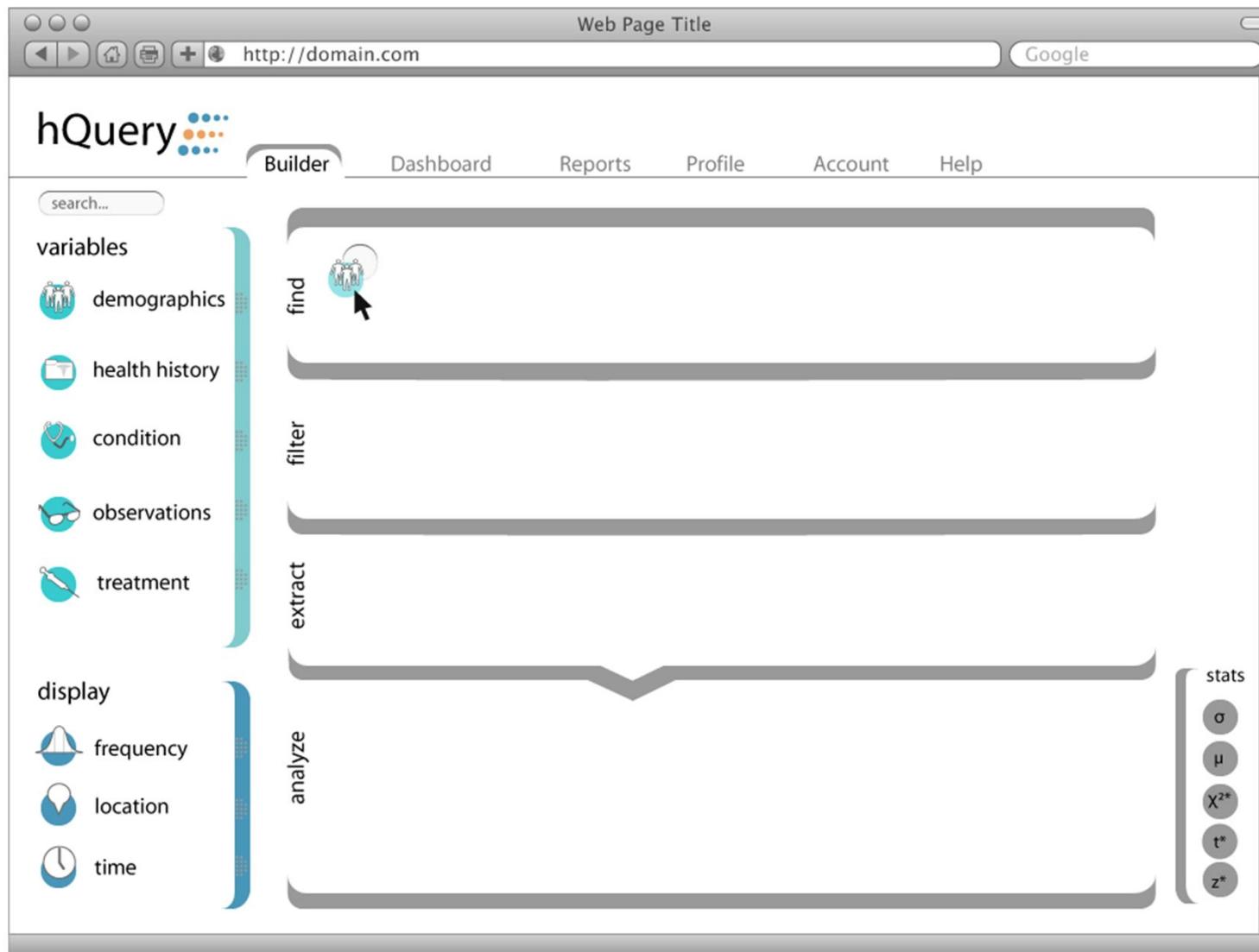
Title	Patients by Gender
Description	Count of patients of each gender
Filter	(None)
Map Function	<pre>function map(patient) {     emit(patient.gender(), 1); };</pre>
Reduce Function	<pre>function(gender, counts) {     var sum = 0;     for(var i in counts) sum += counts[i];     return sum; };</pre>
Endpoints	Query URL: http://127.0.0.1:3001 Status: Complete Result: F: 275.0 M: 231.0
Aggregate Result	F: 275.0 M: 231.0

At the bottom of the main content area are three buttons: "Edit", "Execute", and "Execute and Notify". Below the content area are links for "Query List", "Event Log", and "Execution History".

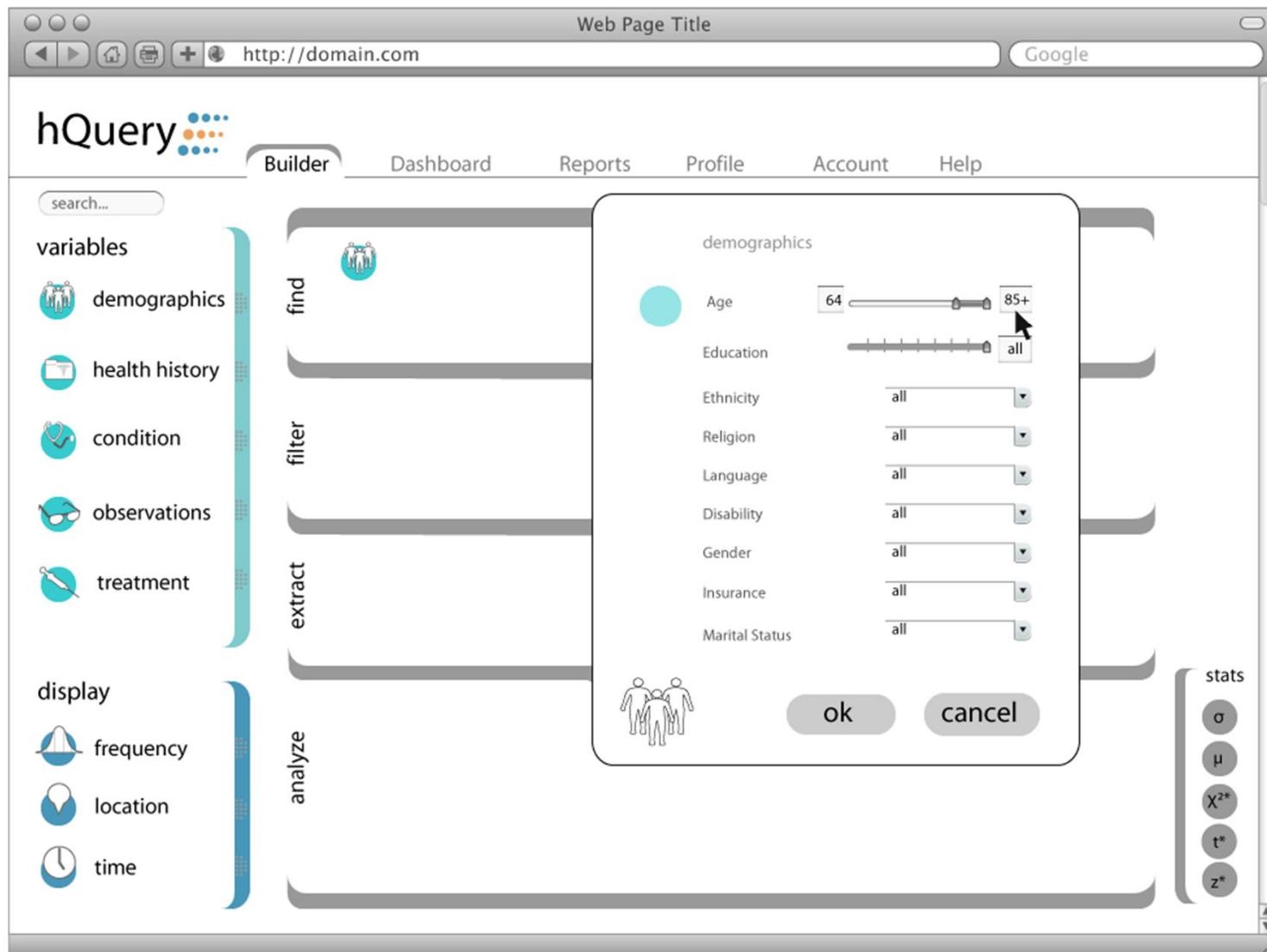
# Graphical UI: Query Builder



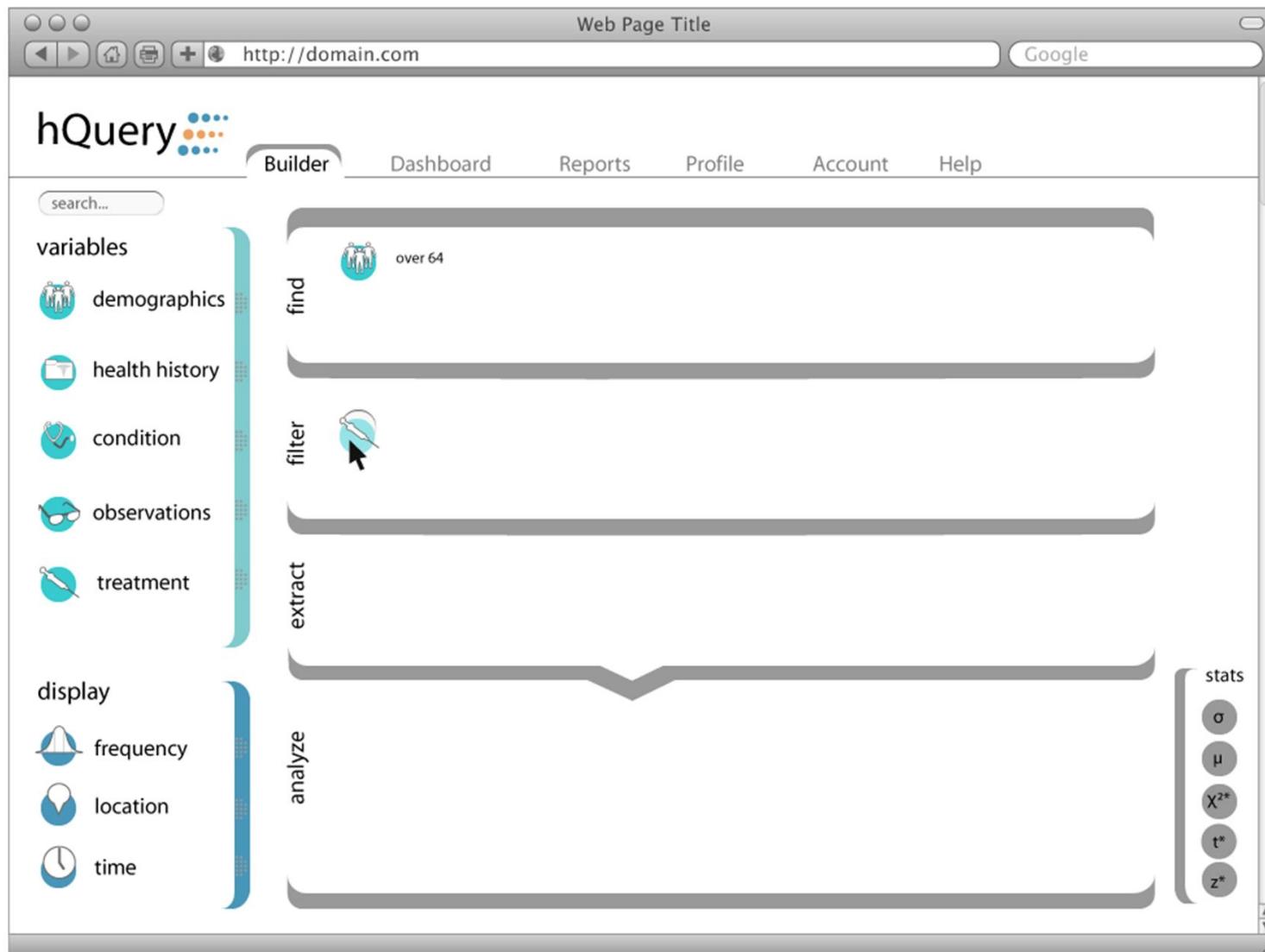
# Graphical UI: Define Population



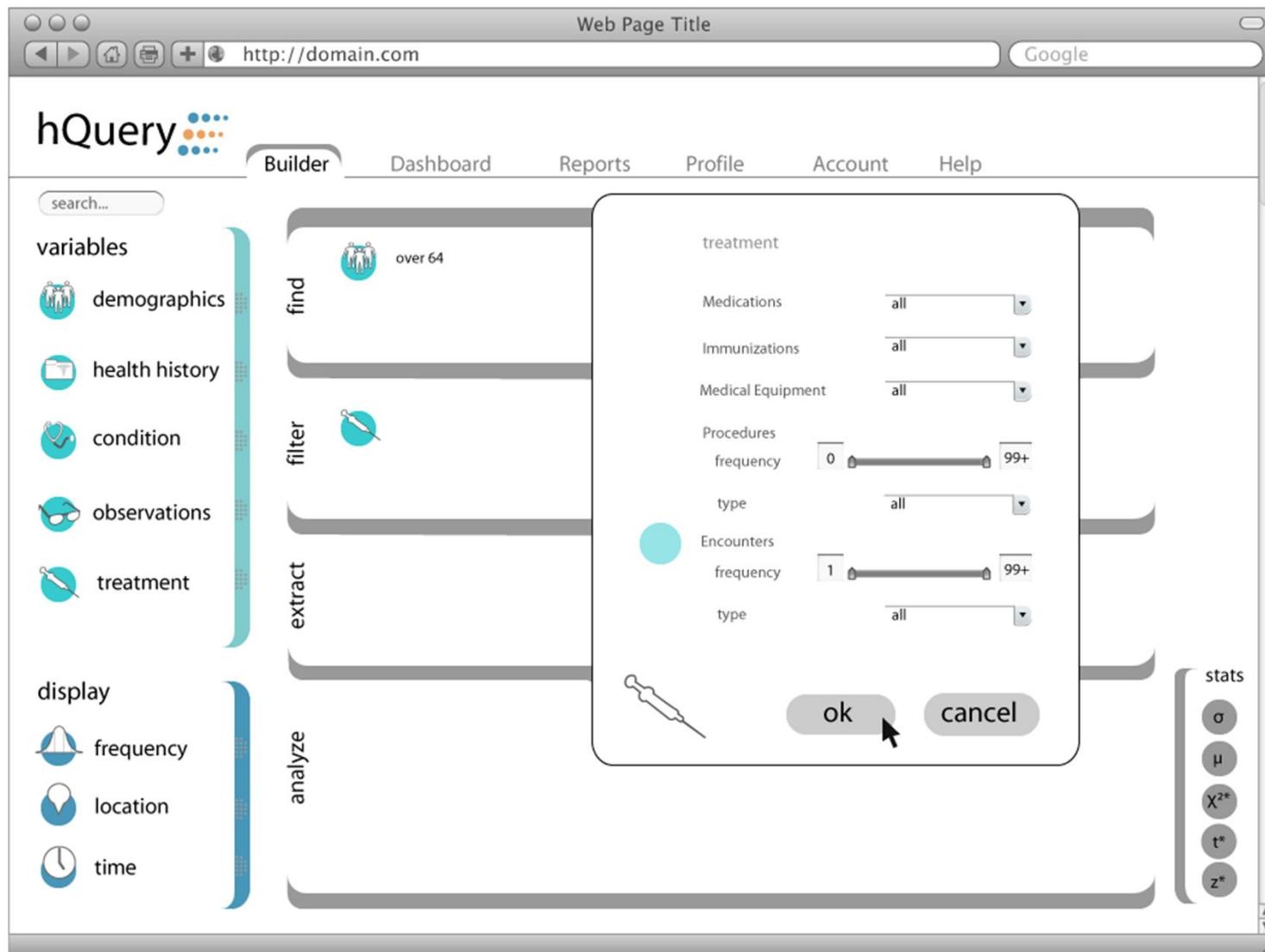
# Graphical UI: Define Population



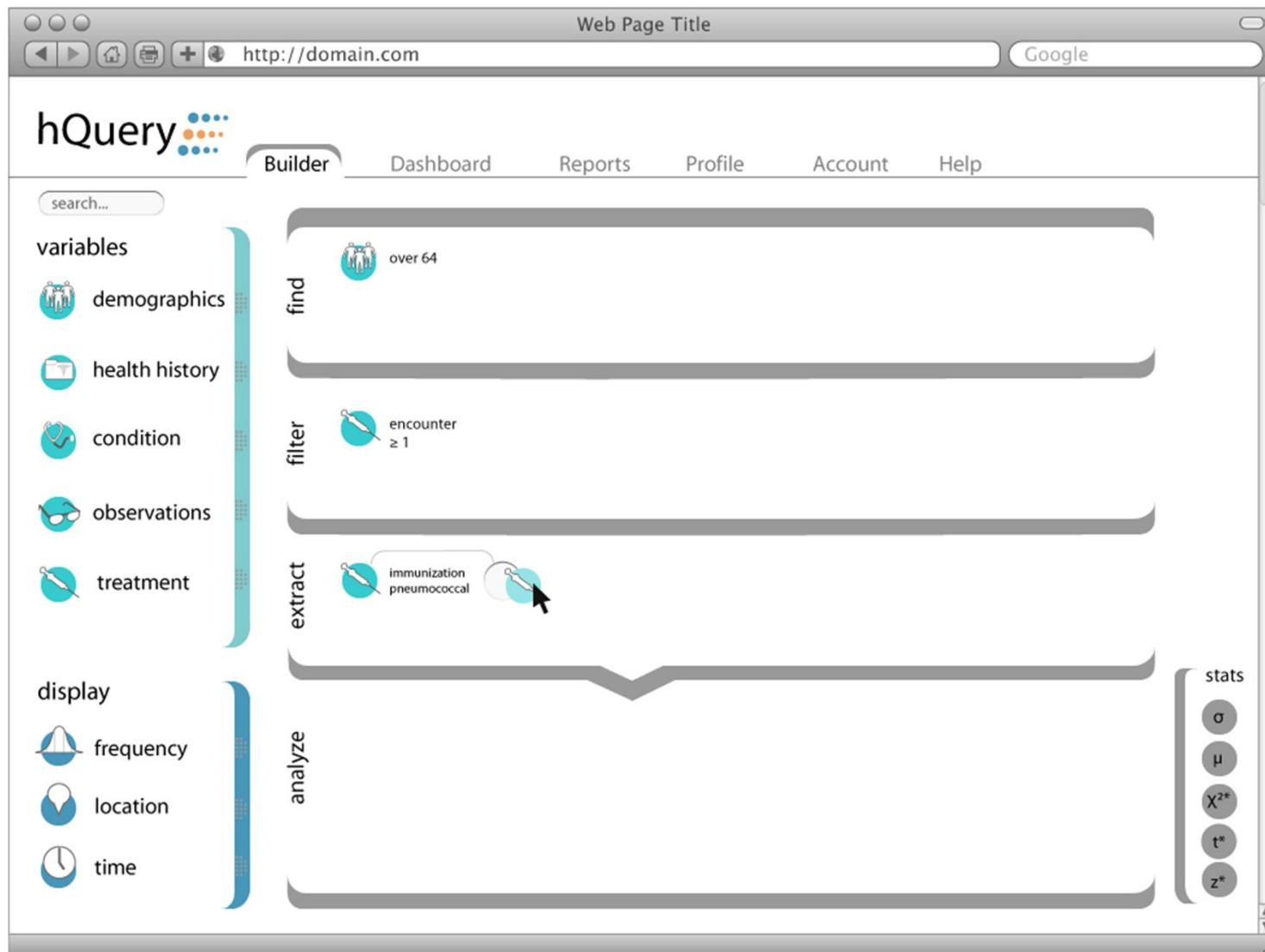
# Graphical UI: Filter



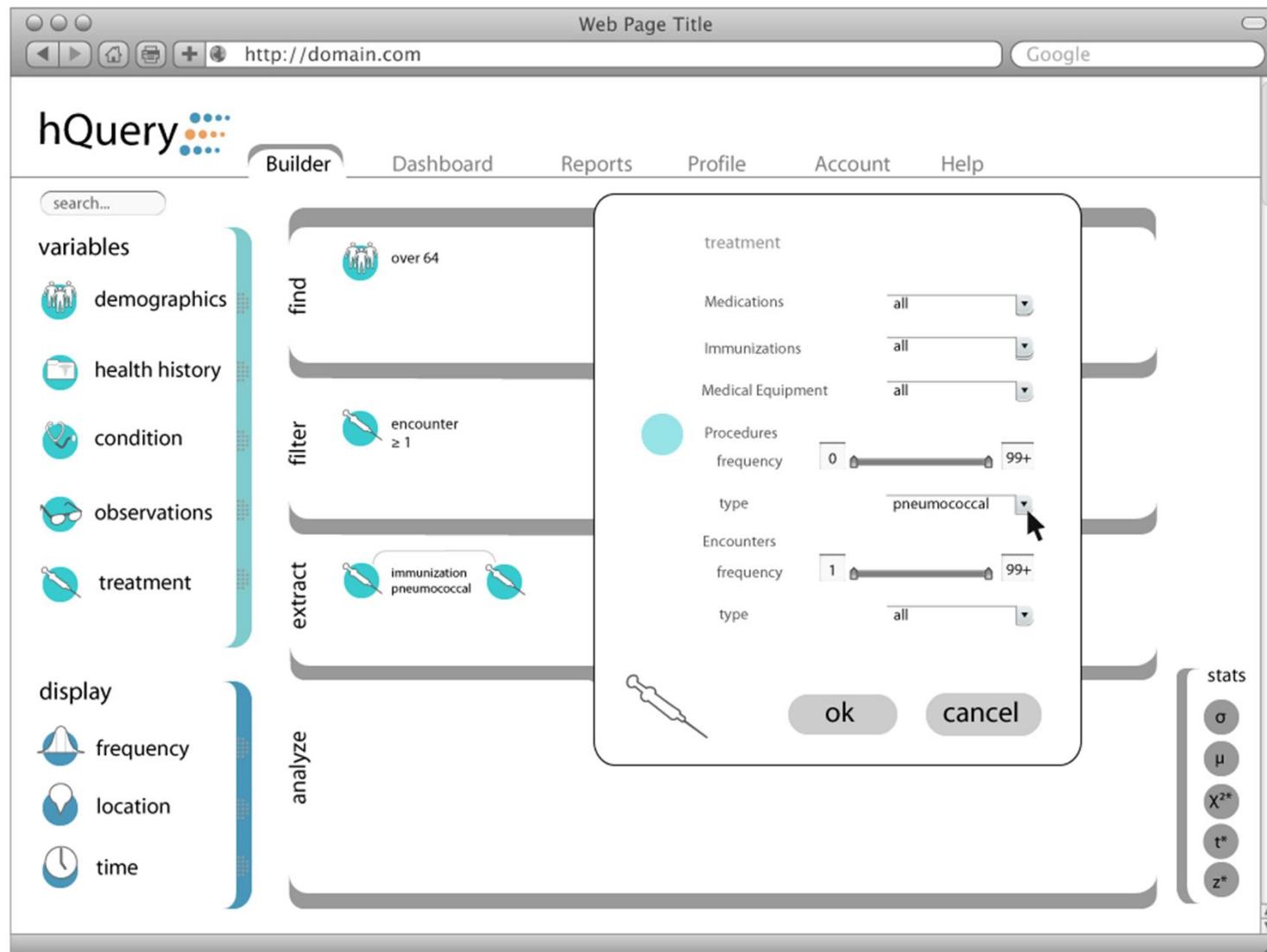
# Graphical UI: Filter



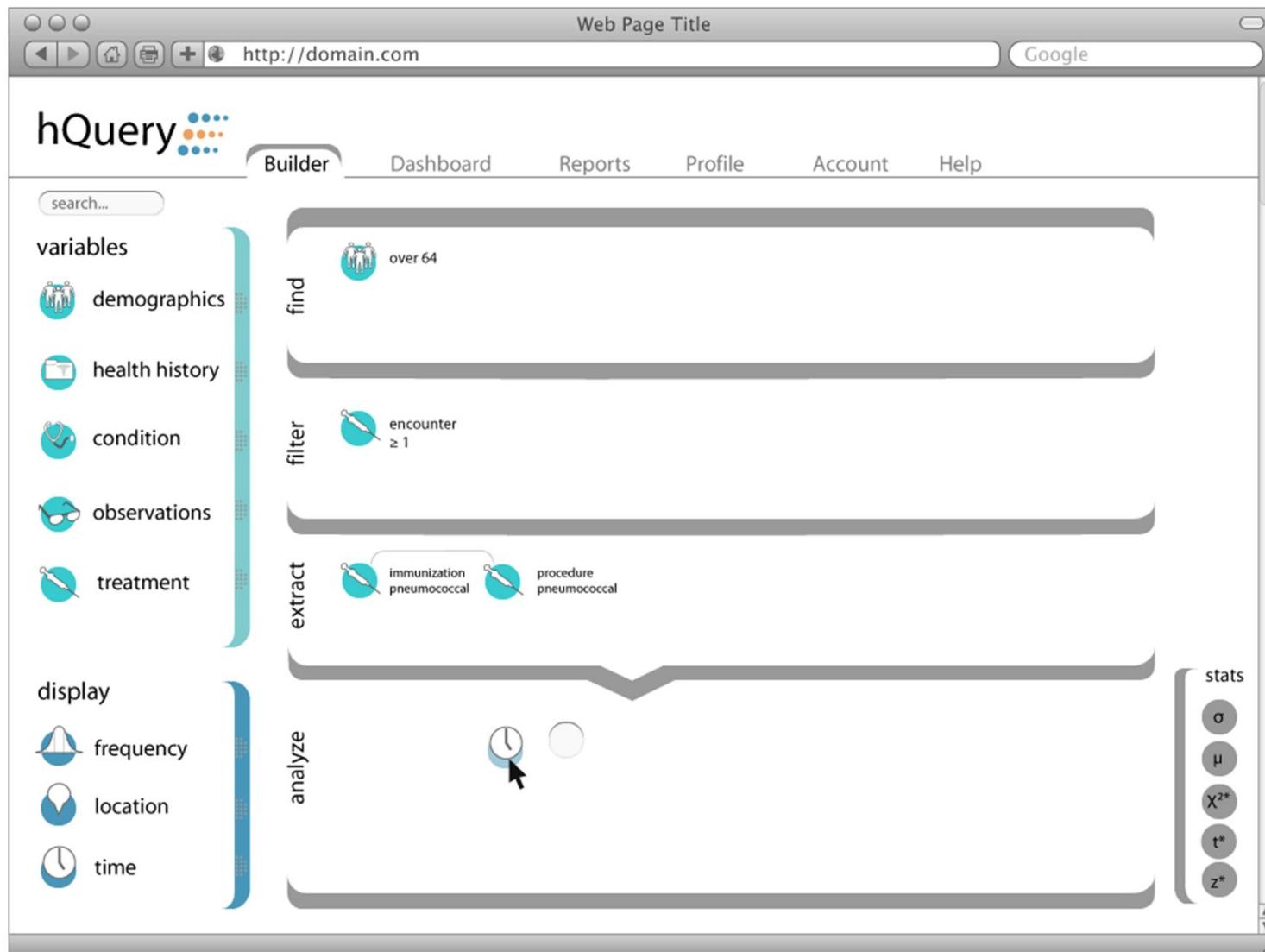
# Graphical UI: Data Extraction



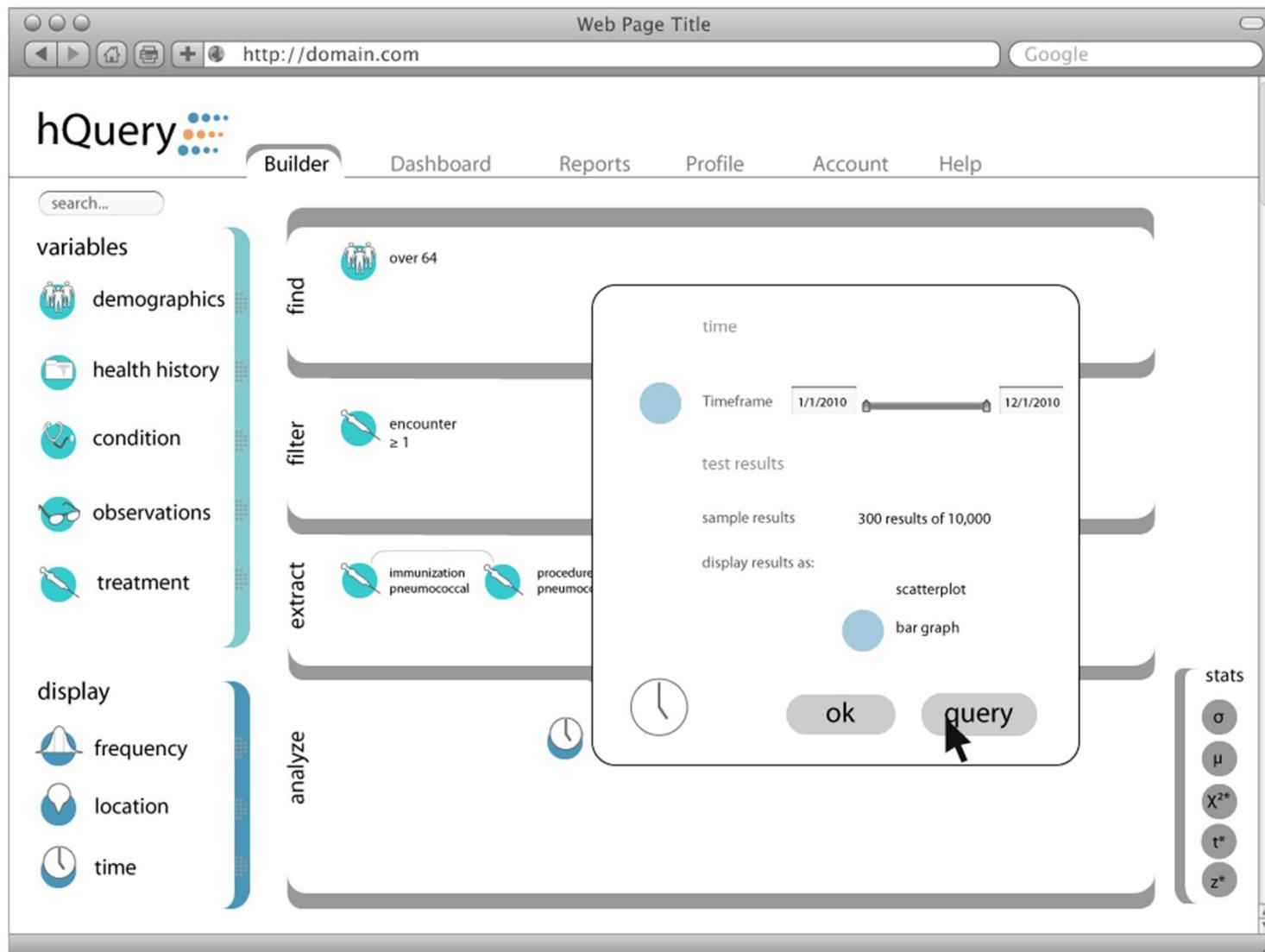
# Graphical UI: Data Extraction



# Graphical UI: Data Analysis



# Graphical UI: Data Analysis



# Graphical UI: Query Dashboard

The screenshot shows a web browser window titled "Web Page Title" with the URL "http://domain.com". The browser toolbar includes Back, Forward, Stop, Refresh, and Home buttons, along with a Google search bar.

The main interface is the "hQuery" application. At the top, there is a navigation menu with tabs: "Builder", "Dashboard" (which is currently selected), "Reports", "Profile", "Account", and "Help". A "Signed in as: User" message is displayed above the navigation.

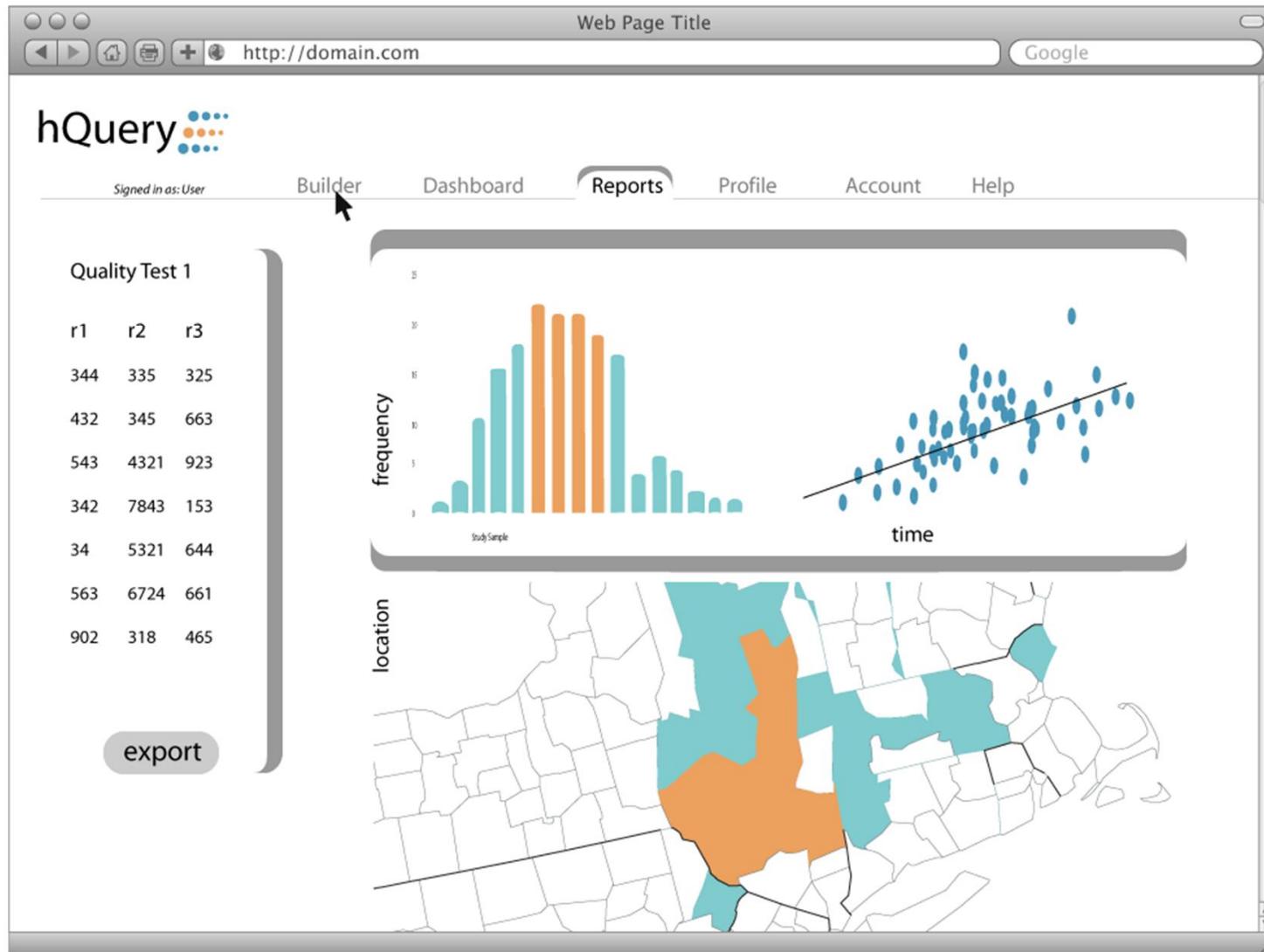
On the left side, there is a sidebar titled "gateways" containing the following information:

- region 1 (blue circle)
- region 2 (blue circle)
- region 3 (orange circle)
  - Speed: 5Mbs
  - Status: Queued
  - Jobs: 200
- add a gateway

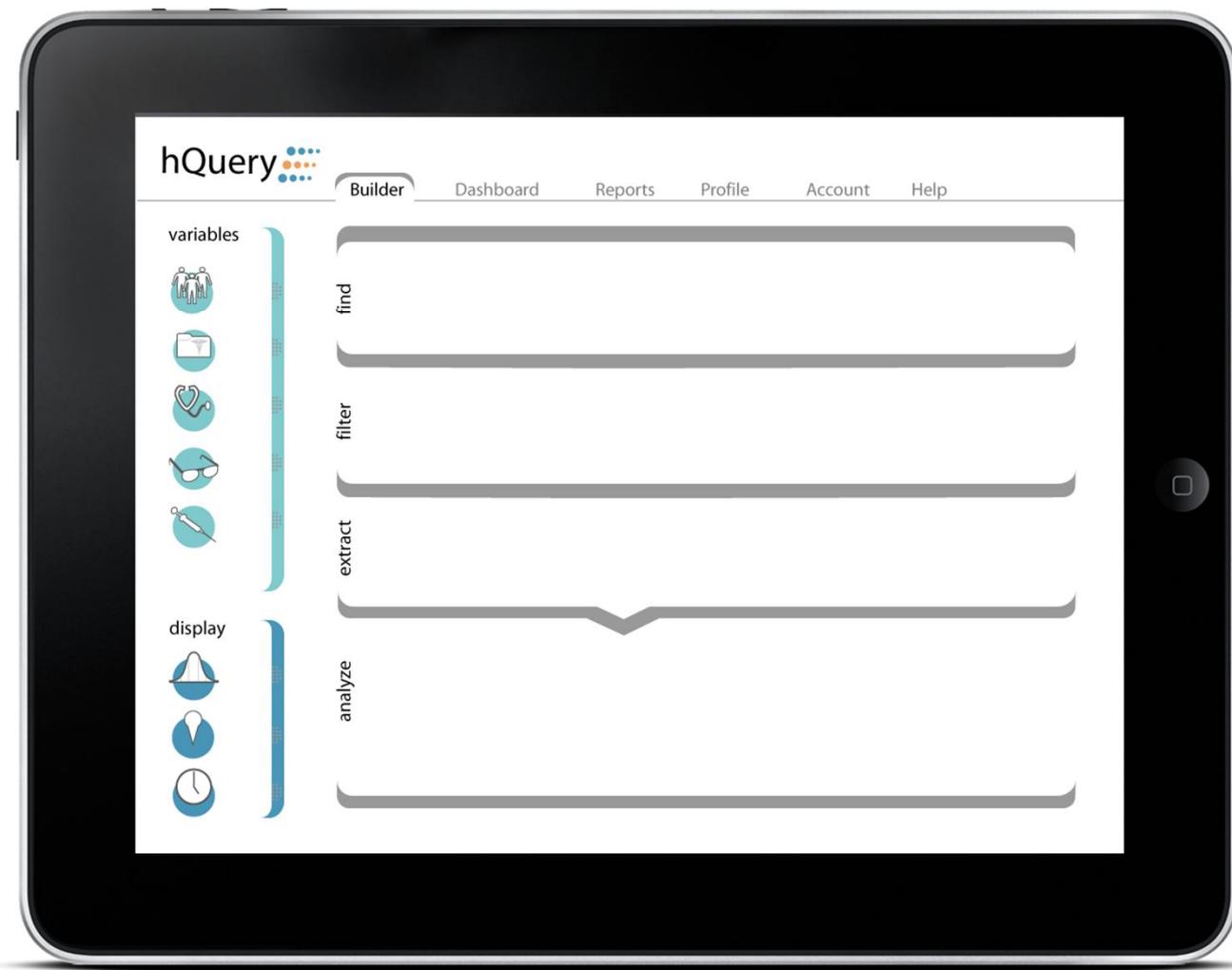
The main content area is divided into three sections by large grey brackets on the right side:

- working**: Contains "Simple Query 1" which has four items, each with a small icon and a progress bar. Below it is a "Status" section with three bars: blue, blue, and orange.
- finished**: Contains "Quality Test 1" which has several items, some with checkmarks and some with error icons. A mouse cursor is hovering over one of the items. To the right of this section is a button labeled "Workgroup Queries".
- samples**: This section is currently empty.

# Graphical UI: Data Visualization



# Graphical UI: Tablet Ready



# Future Work

- **Anonymization**
  - Removal of PHI and PII from patient data model prior to exposing data to queries
    - Privacy-preserving patient identifiers for cross-gateway reconciliation
  - Policy and identity based, different clients may see different parts of data model depending on who they are and what they want to use the data for
- **Vocabulary Management**
  - Query can look for “diabetes” without including corresponding code sets in query
  - Queries can be code-set agnostic
- **Patient data aggregation and reconciliation**
  - Support for incremental updates
  - Support for record merging

# Take-Home Messages

- **Keep it simple**
  - Complexity is the enemy of adoption and interoperability
- **Base Query Health on standard internet technologies**
  - HTTP, SMTP, MIME, JavaScript
- **Queries should be self-contained**
  - Decentralized extensibility
  - Avoids the need for additional standard vocabularies: both functional and clinical
- **Promote serendipitous reuse**
  - Make it easy to ask unanticipated questions
  - Support free-form queries and results
- **More information:**
  - Open source project: <http://github.com/hquery> (available now)
  - Project Web site: <http://projecthquery.org/> (coming soon)

