Querying Integrated Scientific Observation and Measurement Data

Michael Shell
School of Electrical and
Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332–0250
Email: http://www.michaelshell.org/contact.html

Homer Simpson
Twentieth Century Fox
Springfield, USA
Email: homer@thesimpsons.com

James Kirk
and Montgomery Scott
Starfleet Academy
San Francisco, California 96678-2391
Telephone: (800) 555–1212
Fax: (888) 555–1212

***Abstract*—The abstract goes here.**

I. Introduction

In this work, we study the query processing over scientific observation and measurement data using OBOE model[**?**]. OBOE model is a conceptual model used to interpret observation and measurement data.

A. Background

In many scientific domains (e.g., ecology, hydrology, earth science, geology), people collect observational data. Such data record the observed value of some real world entity at some specific place and time. E.g., ecologists studying relationship between the growth pattern and the treatments often need to record the tree heights. The collected data reflect the fact the tree height of a specific tree is 30.1in on May 1, 2009 and 30.3in on May 1, 2010.

Almost all such scientific data do not follow database higher normal forms. Generally, scientists have their way in interpreting their data, but they are not ready for any normalization process. For example, Table I is a simplified but typical dataset collected by a scientist who study the growth pattern of trees. Obviously, the "plt" of the first two rows is the same place, and the "plt" of the last two rows is the same. Here, their area information is redundant. In the real application, many columns have redundant information.

code	no	ht	plt	area
piru	1	35.8	California	4.0
piru	2	36.2	California	4.0
piru	1	25.7	Oregon	3.0
Oriental poppy	1	7.8	Oregon	3.0

TABLE I
Dataset

In scientific domains where people collect observation and measurement data, there are several commonly used and widely recognized canonical concepts([**?**], [**?**]). In this paper, we refer these concepts as OM concepts. The canonical concepts include *observation*, *measurement*, *characteristic*, *standard*, *protocol* or (*procedure*). For example, **add an example to illustrate these concepts.**

Generally, every scientific dataset goes into the data repository with some metadata, e.g., Dublin core metadata[**?**], Darwin core metadata[**?**]. However, all these metadata are at the dataset level, they did not provide enough information for the data *content* inside the dataset. To better make use of the data content in the repository, more systems are embracing the ideas of having metadata on the data content. E.g., Some systems [**?**] provides a mechanism to collect the column/attribute level metadata, some system [**?**] uses *annotation* to add more semantic information to the data content. In this work, we follow the terminology in [**?**] and use the term *annotation* to distinguish the metadata on the data content from those on data objects at a coarser granularity. We propose tools for scientists to provide annotations to scientific observational data. **[**FROM HP: **Put a screen-dump of annotation.]**

[FROM HP: **Queries to such data and challenges to answer such queries.]** When a user formulates a query, it is unrealistic for a him/her to know the underlying data structure of the data. But they all know the well recognized OM concepts as discussed above. So, naturally, when searching such scientific data, people are more interested in finding datasets related to such OM terminologies as observation measurement, characteristic, etc. Given the dataset in Table I. People may ask the following queries.

Example 1.1: Simple and summarization queries:

- Q_1: Give me the data sets that contain species "Picea rubens" observations.
- Q_2: Give me the data sets that contain species "Picea rubens" observations in "California".
- Q_3: Give me the data sets that contain at least five distinct "Picea rubens" observations.
- Q_4: Give me the data sets that have trees with average "height" than 20.0 in "California".

[FROM HP: **Current data integration effort]**

[FROM HP: **Our effort in querying observational data: introduce OBOE, query]**

[FROM HP: **Need re-organization. The description in the following several sections may be moved to here later.]**

B. Contribution and paper organization

Contributions of this work:

- Formalize the queries over observation and measurement query.
- Propose three methods to evaluate queries.

This paper is organized as follows. Section II reviews works that are related to this research. Section III formalize the data model and the queries that people are interested to ask.

II. Related work

[FROM HP: **This will come after the real problem definition and the solution. The description in the following several sections may be moved to here later.]**

III. Data model, annotation and query

In this section, we first illustrate the data model. Then, we formalize the queries that scientists in this domain tend to ask.

A. Data model

When a scientist contributes data into an integrated data repository, a widely accepted way is to convert each dataset to a data table[**?**] or treated as a separate object entity[**?**]. **(HP: is it really widely used? any other system uses this way?. Add more citations here.)** In using this method, the database contains metadata of each dataset and the definition of the data table (e.g., attribute/column name, attribute type, etc).

In our work, we focus on querying this kind of data repository (or databases). To formalize the scenario, we use D to denote the set of data tables in the data repository and d to refer to a specific data table. Each data table d contains metadata about the attribute definition $Attr_d$. Sometimes, we also use *column* to refer to an attribute. Given one data table d, an attribute $attr_i \in Attr_d$ or column index i, $d[attr_i]$ or $d[i]$ represents the set of values for the attribute $attr_i$ or for the i-th column.

B. Annotation

We use A to denote the annotation of one dataset. Internally, we keep the following information for the annotation.

For data table d with annotation A, the system keeps the information $AD = \{(d_{id}, A_{id}, d_{meta})\}$. As we described before, we have four main objects to describe: observation type (OT), measurement type (MT), and context type (CT), and the mapping (Map) from the measurement type to resource attributes. So, the annotation contains the following information.

- $A.OT = \{(\underline{A_{id}, ot_{id}}, et, isDistinct)\}$ to describe an observation type, which denotes on which entity type (object in the real world) the observation is made. Very often, more than one observation can be made on one entity. Thus,*isDistinct* is used to denote whether the same value of key meaasurements of an obbservation types can uniquely identify one observation or not.
- $A.MT = \{(\underline{A_{id}, mt_{id}}, ot_{id}, isKey, Cha, \cdots)\}$. Generally, MT contains information about characteristic (Cha), Standard, Protocol and Precisio. We do not include them here to make the description clearer. Here, *isKey* is used to denote whether one measurement type is the key measurement for the observation type OT_{id} or not.
- $A.CT = \{(\underline{A_{id}, ot_{id}, cot_{id}, Rel}, isIdentify)\}$ where cot represents the context observation type.
- $A.Map = \{(\underline{A_{id}, mt_{id}, attr_i, Cond}, Val\}$

We use the following example to illustrate the concepts in the annotation.

Example 3.1: $A.OT = \{(A_1, OT_1,$ tree, false),$(A_1, OT_2,$ GeoSpot, true) $\}$ denotes that annotation A_1 are for two observation types. One type OT_1 is on real world entity "tree" and the values of its key measurements do not uniquely identify a distinct observation. The other type OT_2 is for real world entity "GeoSpot" (i.e., geospatial location) and its key measurement types can identify it's unique observations. **[**FROM HP: **Introduce key measurements before this.]**

$A.MT$ =$\{(A_1, MT_1, OT_1,$ Species, true), $(A_1, MT_2, OT_1,$ SpecNo, true), $(A_1, MT_3, OT_1,$ Height, false), $(A_1, MT_4, OT_2,$ Plot_State, true), $(A_1, MT_5, OT_2,$ Plot_Area, false)$\}$ represents that OT_1 (for "tree") has three measurement types ($MT_1, MT_2,$ MT_3) for characteristics "Species", "SpecNo", and "Height". The fist two measurement types together form the key measurement type for OT_1. OT_2 (for "GeoSpot") has one key measurement type MT_4 for characteristic "Plot_State" and another measurement type MT_5 for characteristic "Plot_Area".

$A.CT$ = $\{(A_1$,OT_1, OT_2, within, true)$\}$. It shows that observation of a *tree* is within the context of a geo-spatial location. And this context is used to identify the uniqueness of the observation. E.g., a tree with species name "piru" and species no 1 in California is different from a tree with the same species name "piru" at Oregon.

$A.Map$ = $\{(A_1$, MT_1, "code", "eq 'piru"', "Picea rubens"), (A_1, MT_2,"no", null,), (A_1, MT_3, "plt", null,), (A_1, MT_4,"area", null,$\}$. The first mapping rules maps "code" attribute to the measurement type MT_1 (for "Species" characteristic) and change the value to "Picea rubens" if the code is "piru". The meaning of the other mapping rules are very obvious here, so we skip the details.

C. Observation and measurement (OM) query

Above we defined the data model and the annotation, in this sub-section, we define the queries that scientists are interested in We denote the queries on the observational data as *Observation and measurement (OM) query*.

In using the observational data, people are generally interested in the following concepts: *observation, measurement,characteristic or standard.*

Definition 3.1 (Basic OM query): A basic observation and measurement query is defined as Q::=`concept:cond`. Here, `concept` is the main term in an observation model. `cond` is in the form of $f([distinct]\ attr)\ op\ value$ where op is basic operator from $\{=, \neq, >, <\}$ and f is an aggregation function from $\{sum, avg, count, min, max\}$.

Definition 3.2: The result of a query Q is a set of data objects (e.g., data tables) $\{d | d \in D \wedge d\ satisfies$ `cond`$\}$. For each of such result d, we use $d\ s.t.\ Q$ to denote that data object d satisfies the query.

Based on the different `cond` definition, "$d\ satisfies$ `cond`" is translated into different formulas. E.g., if `cond` is defined that the value of attribute "area" need to be smaller than 3.0, then, "$d\ satisfies$ `cond`" is translated to "$dv < 3.0 | dv \in d[area]$".

Definition 3.1 can formulate queries on the one specific observation and measurement. In real application, we need to consider the context relationship. So, we generalize the basic OM query to contextualized OM query as follows.

Definition 3.3 (Contextualized OM query): CQ::=Q_1 `context`$(Q_1.concept, Q_2.concept)$ Q_2. Here Q_i is a basic query in the form of `concept:cond`.

For example, the queries in Example 1.1 can be formalized as the following formal OM queries.

- $Observation : \langle Species =' Picea\ rubens' \rangle$
- $Observation_1 : \langle Species =' Picea\ rubens' \rangle$ $IN(Observation_1, Observation_2)$ $Observation_2 : \langle Meas_{unknown} = `California' \rangle$
- $Observation : \langle Species =' Picea\ rubens'$ $\wedge count(distinct\ Species) \geq 5 \rangle$
- ??

IV. Querying annotated and integrated OM data

In this section, we discuss several methods that we can use to answer OM queries over scientific observational data with annotations.

A very direct way search such data is to extract all the content in the data cells and index them, this way, the system may help answer very simple queries with the help of some thesaurus. For example, for Q_1 in Example 1.1, as long as we can translate the abbreviation "piru" in the dataset to "Picea rubens" based on some predefined thesaurus. We can find the results. Even for this kind of simple query, however, if we want to find observation of "California poppy", an approximate match may return the dataset in Table I as a result because it contains both *California* and *Poppy*. If such compound word search can be alleviated using some existing techniques [**?**], it is almost impossible to directly use such IR-style system to answer a query in the form of Q_3 and Q_4. I.e., answer queries with context.

This naive method cannot answer the desired query. It is because it fails to catch the semantics in the data. In what follows, we would use the annotation during our search to get more semantic support.

As we analyzed in Section II, there are two extrems in querying integrated data. One is to rewrite the original query to a series of new queries over the data; the other is to materialize the data to a consistent data model and then answer the query over the materialized database. We what follows we work from these two directions to leverage the semantic information in answering such queries.

A. Query rewriting

As described in Section III, an OM query is represented using OM concept terminologies such as observation measurement, characteristic, etc. To answer such query from the original data model (i.e., dataset metadata and data tables), we need to *rewrite* the quey over the real data table. In what follows, we first sketch the re-writing process. Then, we detailed the procedure in using the data model and structures. Finally, we include the process for the more complicated cases (e.g., with distinct, aggregate).

Roughly speaking, the query rewriting consists of two steps:

- From the given query, find the relevant data tables and attributes that need to be used answer the given query.
- Translate the given query to queries over the relevant data tables.

TODO: put a figure here to illustrate the steps.

The first step is to map the OM query to the real data structure. We can utilize the annotation structure $A.Map$ which keeps the correspondences between measurement type and table attributes. So, when the given concept is *measurement*, we can directly get its related attribute names and annotation id A_{id}. In case that the given OM concept is a non-measurement concept, we can use other Annotation structures $A.*$ to figure out the measurementt types. E.g., if the OM concept is Characteristic "Species", we can use $A.MeasType$ to get the measurement type id.

As mentioned in Section III-B, when some annotations are done over data tables, the system keeps the information $\langle A_{id}, DTableId \rangle$. With the A_{id}, we can ge the data table information from the metadata $\langle A_{id}, DTableId \rangle$. Then, we get the relevant data table information and the needed attribute information.

Let RD be the set of relevant data tables and $d_{a_1} \cdots d_{a_m}$ are the related attributes for table $d \in RD$. We can translate the original OM query to SQL queries over the data tables.

[`SELECT DISTINCT d.record_id`
`FROM d`
`WHERE` $f(d_{a_1} \cdots d_{a_m}) = true$
[`GROUP BY`]
[`HAVING`]]

The above describes the process for the most basic OM query. However, in the real application, people ask queries with distinct observation or entity constraints. Also, people ask queries about different observatitons using context.

Many times, a user may want to find *distinct* observations or entities. In this case, the above simple solution cannot work correctly because it treats observations in different rows as the same observation. So, we need to use the annotation to figure out what distinguishes one observation from another. When an observation type is denoted with *distinct yes* and some of its measurement types are denoted with *key yes*, the values of the measurements on the key measurement types can uniquely distinguish one observation from the other. To implement this, in SQL, we can perform "group by" operations on the key measurement types of the observations.

The above description discusses cases without context. However, in real application, the observation has context. When the context is not *identifying*, i.e., the context value does not affect the uniqueness of one observation, this can be processed as the basic way. When the context is denoted with *identifying yes*, it means the identity of one observation also depend on the contextual observation. In this case, we need to figure out the all the key measurement types of one observation, then the "GROUP BY" operation should be on all these key measurement types.

TODO: put an example here.

In this case, there is no need to change the SQL scripts. the only change are the relevant attribute set $d_{a_1} \cdots d_{a_m}$, which corresponds to not only the measurement types that directly related to the observation, but also the measurement types that are in the identifying context chain of the observation.

Example 4.1: **TODO: refine this example** Take Q_1 as example. The `concept` is "observation" and `cond` is *species = "Picea rubens"*;

From the annotation $A.MeasType$, we can find the measurement type m_3 with the characteristic *species*. From the mapping $A.Map$, we further find that the attribute *spp* that the measurement type m_3 is mapped to.

The second step is to find the data file which really have this value. For this one, we find the data table, and do a selection on the table content.

TODO: SQL here.

From the above description, we can tell that the computation cost is to search the different data tables. This way, for each candidate data table, we need to send an SQL query to the server for evaluation purpose. If the number of candidate data tables are small. This method should work well. However, when we have a lot of candidate data tables, the efficiency may be affected. In what follows, we introduce another strategy, which logically merge the data content in different tables and perform queries ober the materialized database.

B. Querying over materialized database

The previous section shows the method to rewrite a given query. However, as we analyzed, the computation cost may be higher when the candidate data tables are many. In this section, we propose another strategy to make use of the existing optimization strategies of current DBMS.

The basic idea is to materialize the data into some centralized concept tables with instances of entities, observations, measurements and the contexts between observations. We call such concept tables core OBOE tables, and denote them as $OBOE.*$. In partibular, we have the following concept instance tables.

- $OBOE.EI$ = $\{(\underline{ei_{id}}, et_{id})\}$ for all the entity instances;
- $OBOE.OI$ = $\{(\underline{oi_{id}}, ot_{id}, ei_{id})\}$ keeps all the observation instances;
- $OBOE.MI$ = $\{(\underline{mi_{id}}, oi_{id}, mt_{id}, mVal)\}$ for all the measurement instances;
- $OBOE.CI$ = $\{(\underline{oi_{id}, coi_{id}, ct_{id}})\}$ for all the context instances;

In what follows, we describe how to materialize the data into such concept instance tables in Section IV-B2 when considering all the distinct and context constratins. Then, in Section IV-B2, we describe our strategy to perform queries over materialized database.

1) Materialize data: **: TODO: refine the writing in MaterializeDB and move them here.**

2) Querying materialized database:

Example 4.2: Example 4.1 shows how to deal with Q_1 using the query rewriting method. Using the materialized database, the first step is the same. After we know the data file, then we search the OBOE.MI with the condition that *mVal="Picea rubens"*.

`SELECT distinct` $AD.d_{id}$
`FROM MI, MT, AD`
`WHERE` $MT.Cha =' Species'$
`AND` $MI.mVal = `Picearubens'$
`AND` $MI.mt_{id} = MT.mt_{id}$ `AND` $MT.A_{id} = AD.A_{id}$`;`

In this example, the computation cost is the selection cost of the measurement instance table. It should be faster than the query rewriting.

If the query has some restriction on the aggregate result, we can easily add the GROUP BY and HAVING clause to deal with it. We take Q_3 as an example, which asks for data sets that contain at least five distinct "Picea rubens" observations. The SQL query can be written as:

`SELECT distinct` d_{id}
`FROM MI, MT`
`WHERE` $MT.Cha =' Species'$
`AND` $MI.mVal = `Picearubens'$
`AND` $MI.mt_{id} = MT.mt_{id}$
`GROUP BY` d_{id}, oi_{id}
`HAVING COUNT(*)`$>$`5;`

Using the query rewriting, First, we find the data tables that contain the needed concepts. In each table, we need to figure out the distinct observations. From m_i $obstype_i$, all the other measurements that are of $obstype_i$. Group by the key measurements of $obstype_i$ if distinct yes. If this observation type does not have any context, then the key measurements are all the measurements directly defined under this observatioin type. Otherwise, i.e., this observation type has context, then the key measurements are all the measurements in the context chain.

When we are given contextualized query, e.g., Q_2. Give me the datasets that contain species "Picea rubens" observations in "California".

Species = "Picea rubens" and observation has context *in* another observation which has measurement of value "California".

`SELECT record_id`
`FROM CI as ci, MI as mi, MI as cmi,MT as mt, MT as cmt,`
`WHERE` $mi.oi_{id} = ci.oi_{id}$ `AND` $cmi.oi_{id} = ci.coi_{id}$
`AND` $ci.coi_{id} = coi.oi_{id}$ `AND oi.otype = 'olabel'`
`AND mi.mvalue='Picea rubens' AND`
`cmi.mvalue='California'`
`AND MT.mt`$_{id} = mi.mt_{id} AND MT.mLabel =' Species'$
$AND CMT.mt_{id} = cmi.mt_{id} AND cmt.mLabel = `State'$
$AND coi.otype IN($

$SELECT context\_olabel$
$FROM context\_relationship$
$WHERE olabel =' olabel'); The sub - query finds the set containing all the observation type labels in the context chain.$

```
Use query rewriting, what I can
do? It's the same till to getting
context_olabel_set.
  Get the asked condition columns and the
context columns,
  select record_id from data_tables where
condition column 1='value1' and (context
column1 = 'context value' or context
column2 = 'context value', etc.);
```

Example 4.3: SQ2: Give me the datasets that have measurements with average "area" bigger than 5.0 square feet.

```
Use materialized database: Analysis:
''Area'' is the context of some
observation. Area --> mlabel, observation
type
  From the context observation lable, find
the observation labels,
select olabel
from context_relationship
where context_olabel = 'olabel';
  Let olabel_set be the set containing
all the observation type labels that use
''Area'' as context.
  SELECT oi.record_id
FROM observation_instance
oi, measurement_instance mi,
observation_instance coi,
measurement_instance cmi,context_instance
ci
WHERE (mi.oid=ci.oid AND ci.oid = oi.oid)
and (cmi.oid=coi.oid AND ci.context_oid =
coi.oid) AND coi.otype ∈ olabel_set AND
cmi.mlabel='Area' GROUP BY cmi.oid HAVING
avg(cmi.mvalue)>5.0in;
```

V. Experiments

```
Synthetic data generator.
  Algorithm to materialize DB. Report the
time and space.
  Synthetic query generator.
  Test algorithm to perform query over
materiazed DB. Test algorithm to perform
query-rewriting over materiazed DB.
  Test algorithm for half-materialized
data.
```

VI. Conclusion