# A Faceted Query Engine Applied to Archaeology

Kenneth A. Ross* Angel Janevski Julia Stoyanovich

Columbia University
1214 Amsterdam Ave,
New York, NY 10027, USA
{kar, aj311, jds1}@cs.columbia.edu

## Abstract

In this demonstration, we describe a system for storing and querying faceted hierarchies. We have developed a general faceted domain model and a query language for hierarchically classified data. We present here the use of our system on two real archaeological datasets containing thousands of artifacts. Our system is a sharable, evolvable resource that can provide global access to sizeable datasets in queriable format, and can serve as a valuable tool for data analysis and research in many application domains.

## 1 Introduction

A number of application domains require the modeling of complex entities within classification hierarchies. For many of these domains, the hierarchy and the entity structure are the main sources of complexity, while other features of the domain, such as relationships between entities, are relatively simple. In such domains it is natural to make the set of entities the basic conceptual structure.

*Faceted classification* treats entities or groups of entities as collections of clearly defined, mutually exclusive, and collectively exhaustive aspects, properties or characteristics. Such aspects, properties, or characteristics are called *facets* [8]. For example, in an archaeology domain, we may classify artifacts along multiple orthogonal dimensions: by time period, by culture, by material, by geographic location, etc. Each of these dimensions is a facet. Recent work [4, 3] describes search facilities for hierarchical data using faceted hierarchies. We extend this idea in [7], where we develop a data model and a query language appropriate for such domains. Our primary goal was to create a data model and a query language that is understandable to users, while still allowing one to compose complex queries from simple pieces. A typical user has expert knowledge of the underlying domain, but does not know SQL.

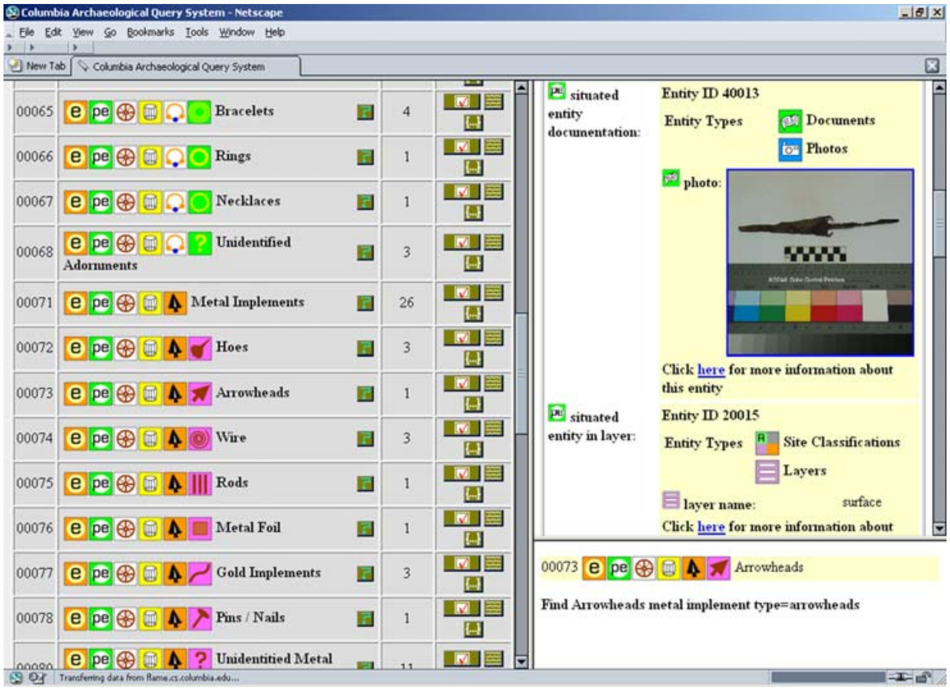
In this demonstration, we present the Faceted Query Engine: a system founded on theory developed in [7]. Our work is part of an inter-disciplinary effort that aims to apply computational tools developed in the areas of Robotics, Virtual Environments, and Databases to modeling, visualizing, and analyzing historical and archaeological sites [1].

We collaborated with archaeologists to create a collection of orthogonal hierarchical classifications of finds from two archaeological excavations: Memphis in Egypt [5] and Thulamela in South Africa [1]. The archaeologists on our team were also responsible for providing input data and helped us with the data cleaning effort. Our system provides a web-based user interface to query thousands of objects from the two locations, and is currently used by archaeologists for data analysis and research. Users of our system can query archaeological finds with respect to any of the facets and have access to text, images, and multimedia data.

According to one of the users of our system, "The faceted query engine has been successful on several different levels. In terms of flexibility, we were able to transition smoothly between two distinct sites and groups of specialists. The system has also proved to be a powerful research tool. The ability to combine and save queries allows for deep investigation of objects and their contexts over time. In this way one may interrogate a group of objects in terms of their distinct properties while also charting their relationship to other entities – excavated finds, ecological remains, contexts, site features, and so on – within the system. The faceted query engine then has the potential to provide broader contextual information to what

*This research was supported by NSF grants IIS-0120939 and IIS-0121239.

**Proceedings of the 31st VLDB Conference,
Trondheim, Norway, 2005**


Columbia Archaeological Query System - Netscape
File Edit View Go Bookmarks Tools Window Help
New Tab
Columbia Archaeological Query System

| | | |
|---|---|---|
| 00065 | e pe Bracelets | 4 |
| 00066 | e pe Rings | 1 |
| 00067 | e pe Necklaces | 1 |
| 00068 | e pe ? Unidentified Adornments | 3 |
| 00071 | e pe Metal Implements | 26 |
| 00072 | e pe Hoes | 3 |
| 00073 | e pe Arrowheads | 1 |
| 00074 | e pe Wire | 3 |
| 00075 | e pe Rods | 1 |
| 00076 | e pe Metal Foil | 1 |
| 00077 | e pe Gold Implements | 3 |
| 00078 | e pe Pins / Nails | 1 |
| 00080 | e pe ? Unidentitied Metal | 11 |

situated entity documentation: Entity ID 40013
Entity Types: Documents, Photos
photo:
Click here for more information about this entity

situated entity in layer: Entity ID 20015
Entity Types: Site Classifications, Layers
layer name: surface
Click here for more information about

00073 e pe Arrowheads

Find Arrowheads metal implement type=arrowheads

Transferring data from flame.cs.columbia.edu...

Figure 1: View of a Portion of the Thulamela Hierarchy

has been in the past the primarily quantitative side of archaeological research."

## 2 Overview of the Demo

During the demonstration, we go over typical user interactions with two real archaeological datasets. The first dataset is a collection of iron-age finds from the Thulamela site in South Africa. Tools, weapons, ceremonial items, personal adornments, pottery, faunal remains, metallurgical products were excavated at the site and have all been classified and made accessible through our system. Our data collection and classification was the first effort made to systematically record and classify this dataset.

Please refer to the left side of Figure 1 for a view of a portion of the Thulamela hierarchy. Consider the entity set "Arrowheads" that appears among the classes in the hierarchy. The user can display detailed information about objects in this entity set by simply clicking on the "list" button next to the "Arrowheads" entry. The result appears on the right side of the browser window. The bottom right corner of the window gives a technical explanation of how the entity set was formed. In this case, the entity set "Arrowheads" is the result of a selection over the set "Metal Implements" with a condition on the value of the "metal implement type" attribute.

The second dataset is a vast collection of ancient Egyptian artifacts from a six-year-long excavation in Memphis. Finds in this dataset range from status objects such as scarabs and personal adornments, to musical instruments and pottery, to a variety of tools, cosmetic vessels, statues, and architectural elements.

During the demonstration we browse the data hierarchies, provide detailed information about individual entities, and demonstrate how objects can be located in the hierarchy by different facets. For example, a shell in the Thulamela data set can be accessed through the "Faunal Remains" facet as well as through the "Status Objects" facet (since shells were used as currency in many cultures). Alternatively, the same object can be seen in the result of a query that selects objects made of the material "shell" from the set of all objects. The user has the option to save the query, thus adding a new entity set to the hierarchy as a view.

The hierarchies of archaeological finds incorporate spatial and temporal relationships among objects, which allows users to perform more sophisticated data analysis. We show how one can formulate a spatial analysis query, and then demonstrate the use of aggregation to perform further data analysis. For example, one may ask the system to retrieve all cosmetic vessels that were located within 10 meters of personal adornments. The resulting entity set of cosmetic vessels can then be semi-joined to the entity set of archaeological contexts (a type of spatial classifier), and the count of cosmetic vessels per context can be returned.

An archaeologist on our team described the following scenario that can occur during the analysis and interpretation of finds. It is often difficult to identify and categorize objects when they are taken out of context. For example, a metal implement, such as the arrowhead in Figure 1, could have been used as a weapon or it could have had a purely ceremonial pur-

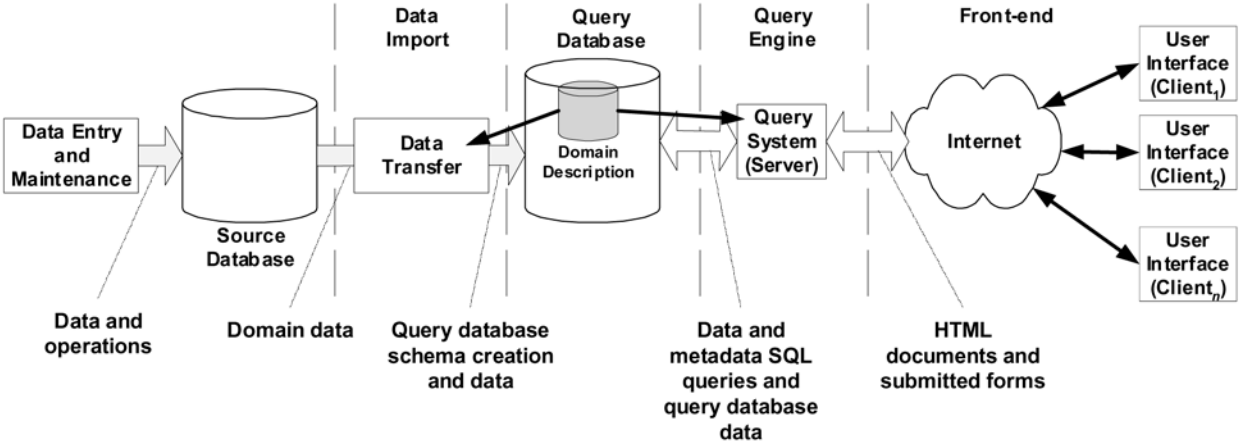

Data Import
Query Database
Query Engine
Front-end
Data Entry and Maintenance
Source Database
Data Transfer
Domain Description
Query System (Server)
Internet
User Interface (Client$_1$)
User Interface (Client$_2$)
User Interface (Client$_n$)
Data and operations
Domain data
Query database schema creation and data
Data and metadata SQL queries and query database data
HTML documents and submitted forms

Figure 2: System Architecture

pose. For another metal implement it may be difficult to tell whether it was used as a tool, a weapon, or a ceremonial item, particularly if the object is not well-preserved. The Faceted Query Engine can help the user put the artifact back into context by means of spatial analysis. In our example, the user may query the system for objects that were found in close geographic proximity to the arrowhead. If other ceremonial items, such as a ceremonial gong, were found in the same geographic context, then it is more likely that the arrowhead also had a ceremonial function.

# 3 Technical Overview

## 3.1 Data Model and Query Language

The basic conceptual structure in our data model is an *Entity Set*. Entity Sets that have entities stored in them are referred to as *Classes*. Classes have attributes and constraints associated with them, and each entity that belongs to a class must specify a value for each of the attributes. A schema defines a finite set of classes organized into a hierarchy, with a single maximal superclass, *Entity*, that adds the attribute *EntityID*. The value of this attribute uniquely determines the entity. An individual entity can belong to one or several classes.

During the execution of a query, operators are applied to one or more Entity Sets to form a new Entity Set. In [7] we develop formal foundations for a query language, *Entity Algebra*, that defines the following operators: selection, union, intersection, set difference, and semi-join. Projections are not allowed by the language. The typing facilities of the language determine the attributes available in the result of a query.

We show in [7] that Entity Algebra has linear space complexity and quadratic time complexity for all expressible queries.

In principle, schemas that are expressible using our model can also be expressed using the Resource Description Framework (RDF) [2], in combination with a schema and constraint language, such as RDF-Schema, and an ontology language like OWL. Our solution is far less expressive, but in return has a much lower conceptual complexity. We believe that our model hits a sweet spot in the trade-off between available features and complexity: it is sufficiently expressive to capture many real-life domains, yet simple enough to be mastered by a non-technical user.

## 3.2 System Architecture

An overview of the system architecture is given in Figure 2 [6]. The *Data Entry and Maintenance* module allows the transfer of domain data into the relational *Source Database*, which captures the aspects of the domain data that will be explored by the query system. The *Data Transfer* module implements a generic procedure that translates domain-specific data into a domain-independent faceted hierarchy in accordance with the *Domain Description*, i.e., the application schema. The resulting hierarchy is stored in the relational *Query Database* using a commercial database system. We describe the schema of this database in more detail in Section 3.3. The *Query Engine* is implemented as a web service and interacts with the Query Database. We elaborate on the functionality of this component in Section 3.4. Finally, the graphical user interface is implemented as a thin browser-based client that accesses the Query Engine over the Internet to submit queries and receive query results. Please refer to Section 3.5 for a description of the graphical user interface.

## 3.3 Query Database Schema

We use a commercial RDBMS to store the Query Database. While the relational model does not directly support hierarchies, we choose a relational database as our data storage and management platform because of high reliability and standardization of the commercial products. The Query Database stores domain meta-data as well as data. Domain meta-data is generated automatically based on specifications provided by the domain expert, and includes information about appropriate classification of the domain, individual classes, class attributes, and constraints. The Query Database schema is domain-independent and consists of a set of tables that store the faceted hierarchy of the domain along with pre-defined and user-defined queries and domain data. The central table of the schema, `QUERY`, stores the classes, queries, and operators. The table `HIERARCHY` contains `(class, parent class)` pairs for all classes in the hierarchy. A list of valid attributes for each class is stored in the `ATTRIBUTES` table.

Values of enumerated attributes are stored in the table `ER` in the form `(entityID, attributeID, value)`, where `value` is a foreign key reference to a lookup table. The reference maps to a `string` column in the lookup table. Values of other attributes are stored in several tables that differ only by the type of the value column. Which tables will be created for these attributes is determined during data transfer and based on the Domain Description.

The model supports one-to-one, one-to-many, and many-to-many relationships between entities. Many-to-many relationships are presented to the user as set-valued attributes although the underlying implementation still preserves first normal form.

### 3.4 Query Engine

The Query Engine implements the basic Entity Algebra operators by translating an Entity Algebra query into a corresponding SQL query, executing the query against the Query Database, and returning the result to the client. Query specifications can be saved for future reference in the `QUERY` table. When a query is saved, query results are not cached. Rather, information about input Entity Sets and operators is kept in the table, which is sufficient to reproduce query results. This is analogous to defining a view.

In addition to the basic operators, the Query Engine also implements aggregation. Aggregates are built from existing Entity Sets by selecting one of the available attributes and performing a series of semi-joins with other Entity Sets. Finally, the attributes of the resulting Entity Set are augmented by an aggregate (`count`, `count distinct`, `max`, `min`, `sum`, `avg`).

### 3.5 Graphical User Interface

The graphical user interface is implemented as a thin browser-based client that abstracts the Data Model and the Entity Algebra from the user. The user can interact with the client to browse the class hierarchy, request detailed information about an Entity Set or an individual entity, and pose queries to the Query Engine. The user can save results of a query as new Entity Sets in the class hierarchy alongside the predefined Entity Sets and then use them as input to future Queries. When performing selections, semi-joins, or aggregates, the system determines which attributes are available for the operation.

The user can access text, images, and multimedia information such as QTVR (QuickTime Virtual Tour) by browsing the hierarchy and requesting a detailed view of an entity. The types of multimedia data the system supports are limited only by capabilities of the web browser and availability of browser plug-ins.

## 4 Conclusions and Future Work

Our system demonstrates that faceted classifications can be used naturally to model data from domains where the complexity lies primarily in the hierarchy. The system has been well-received by the archaeologists on our team.

The Faceted Query Engine is a sharable, evolvable resource that can provide global access to sizeable datasets in queriable format, and can serve as a valuable tool for data analysis and research in many application areas. We plan to make the system available on the web and believe that the system can facilitate global collaboration among archaeologists. In future work we will invstigate query publishing and annotation functionality to allow users to share their interpretation of the data.

## References

[1] Computational Tools for Modeling, Visualization and Analyzing Historic and Archaeological Sites. http://www.learn.columbia.edu/nsf/.

[2] Resource Description Framework (RDF). http://www.w3.org/RDF/.

[3] The FacetMap Project. http://facetmap.com.

[4] The Flamenco Project. http://bailando.sims.berkeley.edu/flamenco.html.

[5] L. Giddy. *The Survey of Memphis II. Kom Rabi'a: the New Kingdom and Post-New Kingdom Objects.* The Egypt Exploration Society, London, UK, 1999.

[6] A. Janevski. A query system implementing entity algebra., 2003. Professional Degree Thesis. Columbia University, Department of Computer Science.

[7] K. Ross and A. Janevski. Querying faceted databases. In *In Proceedings of the 2004 SWDB Workshop*, Toronto, Canada, August 2004.

[8] B. Wynar. *Introduction to Cataloging and Classification.* Libraries Unlimited, Inc., 8th edition edition, 1992.