# Broadening Enchilada: Generalizing the Particle Abstractions and Importation Devices

Anna Ritz ‘06, Thomas Smith ‘07, Leah Steinberg ’07
With Professor Dave Musicant

## EnChIlADA

For the last two years, we have been working on a piece of software that analyzes aerosols that the our fellow Chemistry research team collects using a TSI Aerosol Time-of-Flight Mass Spectrometer. It is called EnChIlADA, standing for “Environmental Chemistry through Intelligent Atmospheric Data Analysis.” The format of the particles that the ATOFMS records is specific to the instrument, so originally we built the software around the ATOFMS particle characteristics.

But demand has grown! In collaboration with UW-Madison, we spent the summer generalizing EnChIlADA to accomodate any particle type. It now can import and analyze any kind of aeorosol information: ATOFMS particles, AMS particles, mercury levels, wind speed. Even better, it can import any type of time series, meaning any type of information associated with a timeline. Ultimately, this software will be free and useful for any scientist analyzing aerosols, and it will allow different types of data to be imported and analyze synchronously.

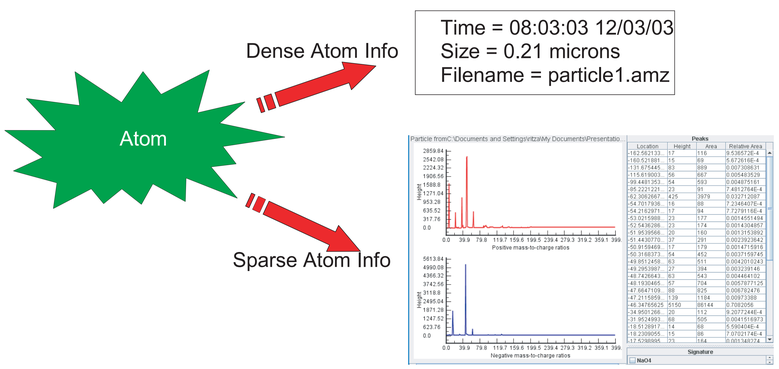
## Building Blocks of a Dataset

In order to generalize the database, we first needed to decided what every single piece of data share. This formed a nested structure that became the basis for the entire generalization process

### Particle Representation in EnChIlADA

The particles contain two different types of information. Dense information is unique to each particle. This information includes particle filename, particle size, or time the particle was collected. Sparse Atom Info contains a list of keys and values; it usually contains some sort of spectrum for the particle. The example below uses ATOFMS particles to portray the information types.

Note to the science-oriented: the computer scientist’s version of an “Atom” here simply means the smallest single unit of data. Oh, the danger of mixing terms in different fields...

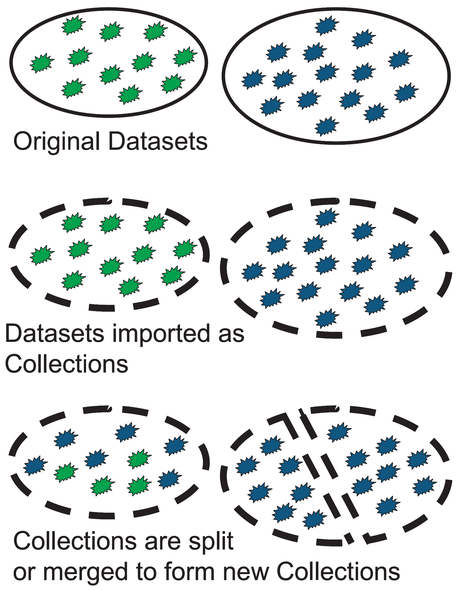



### Datasets in EnChILADA

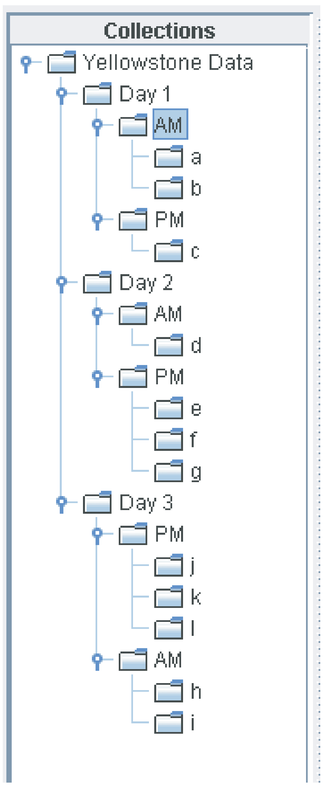
Usually, particles are taken from an instrument in groups, called datasets. These are often determined by when the instrument turns on and when the instrument turns off. Each atom (in computer science terms) belongs to one and only one dataset.

## Abstracting the Dataset

In EnChIlADA, we have implemented the idea of Collections instead of Datasets. This allows the user to combine and split datasets in addition to analyzing them. However, the original dataset that the atom came from is recorded in the database. A collection is essentially a virtual dataset. When first imported, a collection looks identical to the original dataset. However, if the collection is split, more collections are created with the appropriate atoms in them. The figure to the right shows two original datasets that become collections and are then split and merged to become even more collections.




## The Collection Hierarchy




The collections in our abstraction are organized in a specific way: a tree. In computer science, a “tree” is a structure that has a root node (oddly enough, at the top) and branches that contain more nodes. In our case, nodes are collections, and each branch contains atoms from the collections above it. The nested collections are called subcollections. This provides an excellent hierarchical organization.

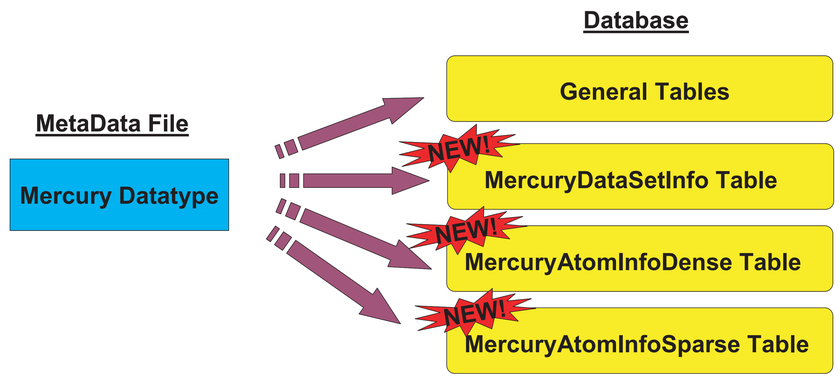
Take the example to the left. The Yellowstone Data collection is actually comprised of multiple datasets (here, they are a, b, c, etc). Further, the hierarchy allows the datasets to be divided into mornings and afternoons and days. By clicking on the AM on Day 1 (the selected collection), you will get the atoms from datasets a and b. Clicking on Day 1 will get you the atoms from datasets a,b, and c. Finally, clicking on Yellowstone Data will give you all the datasets. In this way, users can specify which parts of a dataset is going to be analyzed.

In addition to selecting collections, you also have the ability to cut, copy, and past them. You can perform these tasks using existing collections or by creating a new, empty collection.

## Creating New Datatypes

Suppose we want to use EnChIlADA to analyze a type of data it hasn’t seen yet, and imagine it’s called Mercury. It could have any type of information, including dataset information and both sparse and dense atom information. Mercury will become a new *datatype* in EnChIlADA, or a type of data.

Before we can import Mercury information, we first must let the database know that there is such a thing as a Mercury datatype. This is done through XML files called MetaData files. They are used to create datatype-specific tables in the database. They are only imported once; after the database knows about Mercury, it will be ready to import as much Mercury data as the user wants.




## Importing New Data

So we finally have a Mercury datatype that EnChIlADA recognizes. How do we get Mercury atoms into the program? Again, we use a nested structure to represent the data. Using XML (Extensible Markup Language), an industry standard for exchanging data, you can write plain text files that EnChIlADA accepts. The following is a snippet of XML that the program recognizes:

```
<atominfodense>
    <field>ExampleParticle</field>
    <field>2003-12-03 08:03:03</field>
    <atominfosparse table=“Peaks”>
      <field>12</field>
      <field>.02568247</field>
    </atominfosparse>
</atominfodense>
```

## Acknowledgements

We would like to thank the Chemistry research team at Carleton: Melanie Yuen, John Choiniere, and Katie Barton under the supervision of Professor Deborah Gross. We would also like to thank the Computer Science Research Group at UW-Madison , especially Greg Cipriano.