In many scientific disciplines, the nature of research and publication is changing. Rather than individual scientists collecting and working with their own, small datasets, the trend has been to group all knowledge into much larger, publicly accessible datasets. Moreover, each field is increasingly finding themselves inundated with a mountainous volume of growing data. There is usually no clear way to organize, store, search and process it in a reliable and timely fashion. This is the junction at which computer science can help.

The authors, in helping to manage a database of astronomical findings, developed a number of important criteria that helped them design the system. The first was recognizing that most scientists find databases hard to use and understand, thus shunning them for time tested techniques (ftp-grep). Building a system then has to focus on the ease of data loading, combined with a simple model for posing queries and visualizing the data. Next, in order to establish what could be expected, they defined a set of twenty questions from the field that scientists would like to see solved quickly. Using these criteria, a web service portal was created that allowed scientists to run jobs, maintain private data, and view results.

A crucial piece of experiential knowledge dealt with managing the hardware and understanding the requirements that would be needed over time. For example, the secondary, processed data generated from the initial collection will grow fast. The amount of processing power needed is much more than can initially be expected. Finally, it is important to archive problems, discussions and decisions that occurred during development.
Lessons learned from Managing a Petabyte

The BaBar project is the aim of a long running study in High Energy physics which has amassed over a petabyte of data. This is the largest database in the world and building, managing and improving it has provided a wealth of experience.

In the first iteration, they deployed an object oriented commercial database with modifications that allowed for federated databases to be centrally organized. Though this allowed for better organization, the issue of lock contention came up. This was solved through dedicating separate containers for data and only sharing metadata. Another important consideration was dealing with hardware and reliability. With the large number of disks and servers, they could expect some equipment to fail regularly.

A lesson learned early on was that using the commercial ODBMS locked users into working with data inside a framework, something that scientists did not particularly enjoy. This, among other reasons, led to a refactoring of the system.

In the second version, the main goal was to move away from closed source models which often didn't fit with internal goals. An open source choice was made that allowed for data persistence. They further realized that the system needed to be easier to use and the data had to be more accessible for scientists. In this sense, they utilized ROOT with a C++ interpreter that helped users develop code faster.

The lessons to be taken away boil down to: 1) make sure the system is easy to use and the data is easily accessible. 2) issues of scalability should never be underestimated. A little foresight goes a long way.