

# Support in the BaBar Objectivity Database for Multiple Federations.

Simon Patton<sup>1</sup>, Akbar Mokhtarani<sup>1</sup>

<sup>1</sup>Lawrence Berkeley National Laboratory (LBNL), Berkeley, USA

## Abstract

The BaBar database was built using the Objectivity OO database management system. When the system was original created, any job only had access to a single federation of databases and this federation was limited to a maximum of 64K databases, i.e. files. It became clear early on that given the expected amount of data and the anticipated access patterns that the 64K restriction would have to be removed by some means. By collaborating closely with Objectivity a number of candidate schemes were contemplated. The final choice was to allow a job to be able to access more than one federation while it was executing. This decision lead to a re-engineering of some of the core database infrastructure while leaving most of the higher level database code (and all of the physics code) untouched. This paper presents the overall philosophies adopted so as to allow a smooth migration from a single to multiple federation system. It also reviews the changes to the core infrastructure that were needed to enable multiple federation support and the new structures that were introduced to exploit this new mode of operation. Throughout the discussions it examines how the original design choices helped or hindered this transition.

Keywords: OO Databases, ODBMS, Objectivity, Multiple Federations

## 1 Introduction

BaBar has been filling its single production Objectivity federation since it first turned on in 1999. However, during this time it has become apparent that this federation does not have the capacity to hold all of the data the experiment is expected to collect over its lifetime. A number of possible solutions have been considered, with the final choice being that of allowing a single job access to more than one federation during its execution.

This paper will discuss how this new solution was implemented within the confines of the existing data and the current state of progress on the transition to this new mode of operation.

## 2 Original Design

In the original design of the BaBar Objectivity federation, data was divided onto six separate and distinct *Domains*. Each Domain was characterized by a different access pattern and thus a different interface. The Domains, along with a user's identity, were also the basis for access authorizations. The following is a summary of each Domain:

- Event Store – contains event data and collections of events.
- Conditions – contains calibration data, either specified, derived or as the results of *rolling calibrations*.
- Configuration – contains the various configurations of the detector.
- Ambient – contains recorded data about ambient conditions during data taking.
- Spatial – contains the results of the rolling calibrations for a single processing node.
- Temporal – contains the combined results of rolling calibrations for all node in a single run.

One feature about the Domains that was to be crucial for the transition to multiple federations was that there where no explicit references between objects in different Domains. Access to “matching” objects in other Domains was achieved by keys.

While there are no explicit references between Domains, as only one Objectivity federation can be accessed by a job while it is executing, all Domains existed in a single federation. This placed limitations on that federation. For example, each federation is limited to 64K databases (each database maps onto one file) and thus to accommodate our large data volume databases had to be large, 2-20 GB. Also, each development federation had to contain complete copies of the conditions and configuration Domains if reconstruction code was going to be used in these federations.

### 3 Possible Solutions

BaBar consulted with Objectivity about a series of possible solutions. The following were the main three options:

- Extending the database range – this would increase the 64K database limit of the existing federation. However this would require a substantial amount of re-engineering of the ODBMS.
- Multiple file databases – this would allow a database to span more than one file. However, as the database is the basic unit of management, this may not be too useful.
- Multiple Federations – this would allow a single job to access more than one federation while it is executing.

The first two options would require significant work to modify the ODBMS while it turned out that, allowing for a number of restrictions (see below), the third option was already viable with version 6.0 of the product. It was therefore decided to go with this option.

#### 3.1 Restriction on Multiple federations

While it was already possible to access more than one federation in a single job with version 6.0 of Objectivity, it was discovered that the schemas for each of these federations must be *physically* identical or certain operations would fail. The direct effects of this are that federations created with different versions of Objectivity may not be accessed in the same job and it is not possible to mix federations with different page sizes as both of these situations generate different physical schema.

Another problem, though not really a restriction, with using multiple federation is that the implementation of the `ooHandle` class does not check that it is being accessed in conjunction with the right `ooContext`, and each federation requires a different `ooContext`. This means that great care needs to be taken to make sure that these match at all times in a job. (This is true for any job using more than one `ooContext`, not just those accessing multiple federations.)

Both the restriction on the schema and the problem seen in `ooHandle` access are scheduled to be overcome in version 7.0 of the product.

## 4 Migration to Multiple federations

The migration from BaBar's original design of a single Objectivity federation to multiple federations broke down into two different phases. The first phase was to allow each Domain to exist in its own federation. This is known as *Domain Divestiture*. The second phase is to be able to expand the Event Store so that it could span more than one federation. (The other Domains are not expected to grow to a size where they need to use more than a single federation.)

### 4.1 Domain Divestiture

As has already been noted, Domains do not contain any explicit references to other Domains. This makes their divestiture relatively straight forward. The main challenge at of this phase



from the original event while those portions that are, have new data owned by the new version of the event.

It can be seen from the diagram that all three styles of access have references to common portions of the event, either all or part of it. This means that all the data related to a single event must be contained within the same federation, as no inter-federation references are possible in the current product. Thus the solution to allow the Event Store to be contained in multiple federations is to create a new type of collection, called a *Bridge Collection*, which contains a list of other collections and the federations in which they exist. Then, by iteration over the contents of the Bridge Collection, a job can access each event collection in turn, switching federations when necessary. As access to events is already facilitated by using iterators over the content of collections this means that by providing a suitable iterator, which manages the federation switching, to the client code it can use these new collections transparently and no modification in their code was required.

## 5 Current Status

As of the middle of 2001, the new Objectivity management class has been implemented, including the necessary changes so that jobs can be *inhibited* (to allow for administrative duties to be performed) and to make sure authorizations are matched to the correct federation. A read-only version of a Bridge Collection is currently being tested and no limitations with respect to version 6.1 of Objectivity have yet been discovered. Work is in progress on a writable Bridge Collection.

## 6 Summary

After having collected data in a single Objectivity federation during the first year and a half of running, it became clear that, as it stood, the existing federation would not be able to hold all the data produced by BaBar. After having reviewed a number of possible solutions it was decided to allow a job to access more than one federation while it was executing. The existing methods of use of Objectivity in BaBar meant that the restrictions imposed by version 6.1 of Objectivity did not stop this solution from being practical.

Implementation of the solution required only a small change to the portion of the Domain interface which handled Objectivity management, to include a mechanism to signal which Domain was currently in use. The simplicity of this change was the result of not allowing Domains to reference objects outside the Domain in which they were store.

The expansion of the Event Store, similarly, needed only minor changes to allow the addition of a new Bridge Collection class. This change was transparent to the clients of collection as the federation switching was encapsulated in the implementation of a suitable iterator.

The use of multiple federations in BaBar is now almost complete and thus there should be plenty of capacity into which to store its data over the forthcoming years.