12 Objectivity/DB Enhancement Requests

Based upon our experience with Objectivity/DB, a number of important enhancement requests have been identified. We list below the main requests and their current status. Smaller issues – such as minor bug fixes – are not covered in this report, but are closely followed with Objectivity.

12.1 Support for STL-based Collection Classes

As previously reported, Objectivity have provided STL-based collection classes, using the ObjectSpace implementation, since V5.0 of the product. The issue of providing ODMG-compliant, STL-based persistent-capable collection classes is therefore considered closed. Questions related to the interoperability of the ObjectSpace and vendor-supplied standard libraries are being followed with ObjectSpace directly.

12.2 Support for the Linux Operating System

The use of PC hardware has increased dramatically in HEP over the past few years. Indeed, there are proposals to focus on such hardware, running either Windows or Linux operating systems, for the medium-term future. Commercial software, such as Objectivity/DB, has been available for some time for the PC platform, but for Windows NT/95 only. Since around the time of CHEP '97, the need for Linux versions of the various commercial packages that are part of the overall LHC++ environment became evident. A initial port of Objectivity/DB to the Linux operating system was made available in mid-1998. About this time, the preferred compiler for Linux changed from g++ to egcs, and a port for this compiler was requested. This version was delayed due to limitations in the compiler that prevented code generated by Objectivity's DDL processor from compiling. These problems were isolated by CERN in early 1999, and a fix to the compiler was obtained. A full release of Objectivity/DB – V5.1 – was made available in April 1999 and the CERN license agreement extended to cover also this platform (for both C++ and Java bindings) in May 1999. This request has thus been satisfied.

12.3 ODBMS to MSS Coupling

An interface between Objectivity/DB and HPSS was formally requested in May 1997. A first "proof-of-concept" prototype was delivered in November of that year and tested at CERN and SLAC, as well as other laboratories, from early 1998. The interface provided is not tied to HPSS – its use with other Mass Storage Systems has been successfully demonstrated. The current interface has been used in production at both CERN and SLAC, with data volumes of 1TB at CERN and many TB at SLAC. Due to a limitation in the most recent production release of Objectivity/DB, V5.1.2, an attempt to access a tape-resident database on a given dataserver will block all other requests to that node until the initial

request is satisfied. This limitation will be removed in the forthcoming V5.2 release, but the issue cannot be considered closed until this release has been received and fully tested.

12.4 SLAC AMS Enhancements

As part of the work to extend Objectivity/DB to interface to MSS, a number of additional enhancements were proposed by SLAC. These features, which are fully described in a SLAC document[23], are all scheduled for the V5.2 release of Objectivity/DB. Included in these enhancements are:

- Defer protocol – the server instructs the client to retry later, for example if busy with another request.
- Redirect protocol – the server can redirect the client to another server.
- Security hooks – permits site-specific client-based authentification.
- Opaque protocol – messages are passed through the server to the underlying system.

12.5 Architectural Changes to Support VLDBs

Although the 64bit OID of Objectivity/DB is theoretically large enough to cope with all LHC event data, practical constraints – in particular the filesize – limits federations to around 500TB – 1TB. In the current architecture, 2^{16} databases (files) are permitted per federation. By using databases in the 10-20GB range, federations up to 1PB would be possible. To create federations up to 100PB, changes to the architecture are required. One possible change would be to allow containers, rather than databases, to be mapped to files. This would effectively permit 2^{32} files. Other possible implementations including a longer OID, allowing multiple federations to be operated on as part of a single transaction, and so forth. No firm commitment regarding the implementation schedule or technical details of the eventual VLDB solution has yet been obtained from Objectivity. However, they are now receiving requests for such a capability from customers outside HEP and hence expect to provide such a facility in the medium term. Meanwhile, it is clear that both BaBar and COMPASS will hit the current limits in the 2000 – 2001 timeframe. Given the lead times in Objectivity releases, it is not unlikely that a stop-gap solution, such as the use of multiple federations, will be required.

12.6 Schema Handling Enhancements

Enhancements to the way in which the schema for persistent C++ classes are handled are required such that it is easy and transparent to develop applications across multiple federations. In other words, the developer should be able to build an application using a test federation, and not the production federation of a given experiment. This would require, for example, that no type numbers are hard-coded into the header files produced by the DDL processor. An acceptable solution would be to adopt a similar mechanism to that employed in the Java binding, where the type number of determined at run-time. Such changes should

[23] See http://wwwinfo.cern.ch/asd/rd45/reports/ooams/oofs_AMS_5.0.doc. A description of these features may also be found in http://wwwinfo.cern.ch/asd/rd45/workshops/july99/babar-ams/index.htm.

be compatible with currently supported features, such as support for named schema, classes of the same name, but in different named schema and for schema evolution.

12.7 Access Control Support

In the current version of Objectivity/DB, access control, based on client credentials, is not supported. It is a requirement that such support be added to a future version of Objectivity/DB. Such access control must work consistently across the entire federation, be supported by both language bindings and tools and support both rôle-based (e.g. DBA) and user-based activities. Given the difficulty of implementing a consistent authentification scheme on all relevant nodes in a federation, exits, e.g. at database open time, where site-specific code may be called would be a valid, if not preferred, solution.

The "SLAC extensions", originally foreseen for Objectivity/DB V5.2, are expected to provide sufficient hooks as to permit a site-specific implementation of an access control mechanism. However, this item clearly cannot be closed until the V5.2 release has been made and the functionality has been evaluated.

12.8 ODMG Compliance

The C++ binding of Objectivity/DB is not fully ODMG-compliant in a number of areas. For example, the ODMG specifies methods *d_activate()* and *d_deactivate()*, which are called when an object enters or leaves scope. It is a requirement that fully ODMG-compliant bindings be provided for all of the languages of interest to HEP (C++, Java), although vendor extensions, for the purpose of performance, are acceptable if clearly marked as such. The Objectivity/DB documentation and training material should be based on the corresponding ODMG language binding.

Unfortunately, improvements in the ODMG compliance of Objectivity/DB are not foreseen for at least the next two releases. Thus, we should assume the current binding and develop whatever work-arounds are required.

12.9 Conclusions

A number of important new features, such as STL-based collections and supported for the Linux operating system, have already been provided. Release V5.2 of Objectivity/DB – currently targetted for end-September 1999 – should satisfy the most time-critical of the remaining enhancement requests, with the exception of VLDB support. Obtaining a satisfactory solution to these issues remains high on the priority list of RD45.

Beyond V5.2 and VLDB, the main enhancements expected in the medium term concern production-level reliability and performance tuning – primarily resulting from the experience of experiments such as BaBar. Objectivity have understood the importance of ensuring the success of BaBar and are reacting accordingly.

13 General Database Activities

In addition to the work-items related to the LCB milestones and recommendations, the following activities are worthy of note.

13.1 RD45 Workshops

As in previous years, a number of RD45 workshops have been held at CERN. These workshops have been attended by approximately 50 people, typically including representatives from Brookhaven, CERN, DESY, FNAL, IN2P3, KEK, LBL, SLAC as well as experiments based at these sites. In addition, people working on other large ODBMS projects – not necessarily based on Objectivity/DB – have also attended. The most recent workshops were held in July 1999[24] and October 1998[25], the latter being transmitted live using "Web-casting"[26]. This technique proved successful and it is planned that future workshops be videoed in this manner. The most recent workshop spanned four full days, covering topics such as Mass Storage Systems, their integration with Objectivity/DB, site and experiment reports, status of enhancement requests, Objectivity/DB futures, as well as progress on the LCB milestones.

The next RD45 workshop is scheduled for the week prior to CHEP 2000, i.e. January 31 – February 4, B160 1-009 at CERN.

13.2 Collaboration with MONARC

There are clearly areas of overlap between the RD45 and MONARC[27] projects, Members of each project also participate in the other and a series of joint meetings have been held. The initial such meetings were devoted to an overview of RD45 activities and ODBMS features. More recently, there have been presentations on MONARC activities at RD45 workshops, showing a bi-directional flow of information.

13.3 Objectivity/DB User Meeting

As in previous years, an Objectivity/DB Developers' Conference was held in Santa Clara in May 1999. Although a rather brief meeting – one and a half days of presentations from Objectivity users and Objectivity themselves – it provides an excellent opportunity to meet Objectivity developers, as well as other users – including those from HEP. At this year's meeting, given the continued delays in the release of Objectivity V5.2, little of what was presented was new. However, Objectivity continue to state that their market is that of

[24] See http://wwwinfo.cern.ch/asd/rd45/workshops/july99/index.html.
[25] See http://wwwinfo.cern.ch/asd/rd45/workshops/oct98/presentations/AGENDA.html.
[26] See http://webcast.cern.ch/.
[27] See http://www.cern.ch/MONARC/.

scalable, distributed systems, including areas such as Telecoms and Science and that they continue to be profitable.

13.4 Objectivity/DB European Technical Forum

Objectivity initiated a user meeting devoted to their European customers in October 1998, the first meeting being held at CERN. At this meeting, each site was asked to provide a short list of enhancements to Objectivity/DB. The highest priority items, listed in descending order, are given below. Only items voted for by more than one customer are listed.

- Support for multiple FDs.
- Enhanced schema handling (e.g. dynamic type number allocation).
- Exclusion list for backup.
- Visualisation tool.
- Documentation of all error codes.
- Object versioning in Java.
- A –host flag on oocleanup.
- Support for the GNU compiler on all platforms.
- Unicode support.

With the possible exception of Unicode support, all of these enhancements are of importance to the HEP community. As these items have been requested by multiple customers, typically from different fields, there is a better chance that they will be implemented than if they were of specialist interest.

The next European Technical Forum will be held on October 28-29 1999 at the European Southern Observatory[28].

13.5 SIGMOD

The 1999 SIGMOD conference on Management of Data was held in June in Philadelphia[29]. An invited tutorial on the management of PB databases was given at this conference[30], helping to spread awareness of the work of RD45. However, with the exception of some follow-on work to the SHORE project, namely PREDATOR, there appears to be no public-domain ODBMS under active development. Similarly, the main interest appears to lie with RDBMSs and "object extensions".

13.6 ECOOP Workshop on ODBMS

At the June 1999 European Conference on Object Oriented Programming[31], a specialist workshop on ODBMSs was held[32]. The aim of the workshop was to bring together

[28] See http://www.eso.org/index.html.
[29] See http://www.research.att.com/conf/sigmod99/.
[30] See http://wwwinfo.cern.ch/asd/rd45/presentations/Sigmod99/index.htm.
[31] See http://www.di.fc.ul.pt/ecoop99/.

researchers working in the field of object oriented databases and to discuss the work that is going on. As there are relatively few papers on ODBMS issues presented at conferences these days, a topic for discussion was whether it was felt that all research in this area has been completed, where the future lay and so forth. A presentation on RD45 was made at the workshop, including a summary of the current risk analysis. Unfortunately, the general conclusions were that the overall market is indeed small and not likely to grow significantly in the foreseeable future. However, on the positive side, it seems that the main technical issues are indeed considered solved – one simply has to refer to the appropriate research paper and implement the algorithm in question.

13.7 CERN School of Computing

The 1998 CERN School of Computing [35] included a track on Petabyte Storage. This included lectures on HPSS, Objectivity/DB and their combined use in HEP. The 1999 school will include a track on Software Building using LHC++, which will contain lectures on Data Storage and Access in HEP, based upon HepODBMS and Objectivity/DB. As well as being included in the proceedings of the respective schools, the material from these courses is available through the Web[33], including the associated exercises (with solutions!) and will be incorporated in the IT tutorial series at CERN.

13.8 Licensing Issues

The existing licenses for Objectivity/DB – covering both C++ and Java bindings – have been extended to include the Linux operating system (in addition to DEC, HP, IBM, Sun, SGI and NT). Objectivity have confirmed in writing that the HPSS interface, including enhancements to the AMS to support multiple threads, will be made available at no charge under the current contract.

A number of new products are expected in the late-1999/2000 timeframe. These include so-called "Active Schema" – an option that provides ODMG-compliant run-time schema access[34] – and Rational Rose integration, permitting the generation of DDL from Rose and reverse engineering from existing DDL files. It is likely that licenses for these options will need to be acquired. Discussions with Objectivity are continuing on these issues.

13.9 Objectivity/DB Support

In preparation for production services, negotiations started in early 1998 concerning adequate support and on-site consultancy. This resulted in a number of important changes, including direct access (via e-mail) to Objectivity/DB developers. In addition, an Objectivity consultant for the HEP market was identified. This consultant, who had previously worked at LBL on BaBar, made a number of visits to CERN, organised to coincide with important COMPASS and NA45 test runs.

---

[32] See http://www.disi.unige.it/conferences/oodbws99/.
[33] See http://wwwinfo.cern.ch/asd/lhc++/TUTORIALS/index.html.
[34] The ODMG standard covers only read access. Objectivity Active Schema will provide also write access.

Significant pressure was also brought to bear on Objectivity regarding reliable release dates for their product, together with advance information about the exact contents of a given release. Objectivity has introduced an internal bulletin on product status, which is forwarded to CERN on a weekly basis. Previously, Objectivity/DB was developed on Solaris and NT and released for these platforms before the work on porting to other systems commenced. This inevitably led to delays and incompatibilities. Recently, they have moved to nightly builds on all platforms. Hence, at "code-freeze" time, the system builds on all supported platforms. This move is welcomed as it is likely to result in a significant improvement in the roll-out schedule as well as reducing the probability of cross-platform incompatibilities.

The code-freeze date for V5.2 was end-August 1999, with a target release date of end-September. Given the changes described above, we can be reasonably confident that V5.2 will be available at CERN on the October timeframe. As there are a number of significant changes in this release, it is not unlikely that "point" bug-fix releases (e.g. V5.2.1) will be made. However, it is reasonable to assume that V5.2.x will be used for production in 2000.

It is too early to confirm that these changes will indeed result in more reliable schedules and better support. However, they do indicate that Objectivity is reacting to our concerns, which in itself should be considered positive.

14 Other Database Developments

14.1 Object-Relational Databases

During the past year, the use of relational databases (RDBMS) together with object technology continued to increase. Techniques for mapping object models to relational ones are now well understood, and there are numerous packages that assist in automating this process. In the immediate future, the use of an RDBMS together with a mapping layer does not appear to be a viable solution to the needs of the LHC experiments, as the underlying databases are not yet capable of scaling to the multi-PB region. At a workshop on large-scale data management held at SLAC in October 1998[35], a representative of ORACLE stated that they had no plans to address these data volumes until there was sufficient commercial demand, although object-relational technology is central to their plans, as witnessed by the ORACLE 8*i* release. Thus, although we can be confident that (O)RDBMS products will be capable of handling such data volumes at some stage during the lifetime of the LHC, it is unlikely that this will happen early enough as to affect the choice for the initial production phase. Nevertheless, as it now appears likely that the mainstream DBMS technology will be RDBMS with sufficient object-extensions as to satisfy business needs [39], the use of this technology should be considered in the medium to long term.

14.2 The Object Database Market

It is now clear that previous predictions concerning the growth of the ODBMS market were overly optimistic. There is no evidence of significant growth – the market appears rather constant at around some $100M. This is no doubt at least partly attributable to the fact that all RDBMS vendors offer some sort of "object solution". Indeed, some analysts attribute the perceived down-turn in the ODBMS market to people waiting to see what products such as Oracle 8*i* offer in terms of OO support. The ODBMS vendors claim that the market has picked up again in recent quarters and is now growing at a healthy rate. Whether this indicates a long-term trend or is just a fluctuation is not clear at this time.

It is generally accepted that for suitable data models and access patterns, ODBMSs do have a clear and measurable performance advantage over RDBMS products. Unfortunately, this advantage is most critical for large volumes of data. As the amounts of data that can be conveniently handled with RDBMSs continues to grow, coupled to improvements in storage and greater CPU power, the corner of the market available to ODBMSs is perhaps even shrinking. Thus, it now appears most likely that ODBMS technology will continue as a niche market, but will not made serious in-roads in the RDBMS market.

[35] See http://www-user.slac.stanford.edu/rmount/dm-workshop/main.htm

14.3 Versant

The Versant Object Database has been considered the primary commercial alternative to Objectivity/DB for some time. Performance and scalability measurements of this product have been reported in previous RD45 status reports and an evaluation of the issues relating to porting existing applications has been made. Unfortunately, the company has fared rather poorly over the past year or so, although the most recent results show a return to profitability. The number of employees has been cut dramatically and at least one of the key developers has left the company. As a result, Versant is no longer considered to be a viable long-term alternative to Objectivity/DB, although this of course could change with time.

14.4 O_2

The O_2 ODBMS [19] – one of the systems considered in the early days of the RD45 project – is probably closest to the ODMG standard in its OQL and C++ bindings. Certain features of the product – such its limited scalability and support for heterogeneity – meant that it was not considered a viable candidate for handling LHC event data. However, it was a well-known ODBMS product, used particularly widely in the University environment. Recent reports from O_2 customers suggest that the system will no longer be development or maintained. Although O_2 was acquired by UniData – now part of the Ardent software company – it no longer appears to be marketed by this company, and the previous O_2 Website responds with "o2tech.com is no longer hosted on this server."

14.5 Objectivity/DB

Objectivity continues to prepare for a public flotation, focussing strongly on quarterly results. They have been consistently profitable over several quarters, which has allowed the firm to expand some 20% in personnel. Should this trend continue, they could expect to reach the size of a company such as NAG or TGS (some 150-200 people) prior to the startup of the LHC. Such an expansion – including a strong local support team in Europe – would be necessary if adequate resources for product development and support are to be made available.

Much has been made of the Iridium satellite telephone project in the past years, which uses Objectivity/DB as a component of the overall software. As is now well known, Iridium failed to adapt to the dramatic change in market of the mobile phone and hence has not been the financial success that was predicted. However, this is in no part due to Objectivity/DB itself. Objectivity continues to perform well in the telecoms market as a whole, which is perhaps the market most likely to provide long term survivability of the company.

14.6 Conclusions

The ODBMS market has not (yet?) taken off as predicted – only one vendor (Objectivity) stresses its ODBMS product in its Web pages – all others refer to a wider range of applications. A degree of rationalisation has occurred, with existing vendors focussing on

different areas of the market. It is expected that the number of vendors will continue to decrease over the next 1-2 years, resulting in a small number of products that are each dominant in different sectors. A large fraction of the activities of all DBMS vendors – both ODBMS and RDBMS – is related to Java. If this trend continues, it is not unlikely that the C++ side will suffer correspondingly. The emergence of a new ODBMS product – particularly one capable of scaling to multiple PB – is considered rather unlikely on the timescale of the 2001 decision of the ODBMS for the LHC experiments. On the other hand, it now appears likely that ORDBMS technology – basically extensions to existing RDBMS products – dominate and that these products will eventually address the data volumes expected at the LHC. Thus, the potential use of these products should also be considered in the longer term.

15 Objectivity/DB Alternatives

15.1 Introduction

The CMS Computing Technical Proposal [27] states (section 3.2, page 22):

"If the ODBMS industry flourishes it is very likely that by 2005 CMS will be able to obtain products, embodying thousands of man-years of work, that are well matched to its worldwide data management and access needs. The cost of such products to CMS will be equivalent to at most a few man-years. We believe that the ODBMS industry and the corresponding market are likely to flourish. However, if this is not the case, a decision will have to be made in approximately the year 2000 to devote some tens of man-years of effort to the development of a less satisfactory data management system for the LHC experiments."

Based on the conclusions of the risk analysis (see section 11 above) and given the current state of the ODBMS market, an estimate of the effort required to produce an alternative to Objectivity/DB is clearly required. It should be stressed that the intent at this stage is not to implement such an alternative, but to obtain a better understanding of what could be produced if required and the resource implications. A necessary first step in producing such estimates is a list of requirements. Such a list was produced in the early stages of the RD45 project [10], but clearly needs to be revised in the light of the experience gained during the past years. A first attempt at such a revised list is presented below. We then discuss the scope of a prototype system that attempts to meet these requirements, present the main architectural features of the system, then describe the current status and possible future directions.

15.2 Draft Requirements

15.2.1 Introduction

The following draft requirements are based upon the experience gained in using ODBMS technology over the past 5 years. To position these requirements, we first describe a number of use cases typical of the HEP problem domain and then discuss in general terms the overall features that a system needs to provide. Finally, we provide a preliminary list of requirements for the main components of a potential system.

15.2.2 Use Cases

Use cases that need to be catered for include:

- Data acquisition and online reconstruction, populating a large, shared event store.
- A physics working group that produces a large shared event collection for analysis by all members of the group.
- A single physicist performing analysis primarily on "private" data.

All these use cases are by no means exhaustive, they do demonstrate the wide range of needs that must be catered for. In each of these three cases – which are strongly coupled, if the end user is to be provided the possibility to access any part of the event store – there are different priorities. For example, in the case of data acquisition and online reconstruction, priorities include scalability to sufficiently large data volumes and excellent throughput (close to that of the underlying filesystems). The main priority of a physics working group is somewhat different – probably reliability and consistency are of greatest concern. On the other hand, an individual physicists requires features such as ease of use, facilities for exporting subsets of the data, the ability to create private classes and instances thereof and so forth. These rather different needs must all be carefully considered if a single system is to be used efficiently for all such purposes.

15.2.3 Data and Resource Sharing

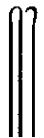
Sharing data and attribute definitions is particularly important if one is to ensure consistent results across multiple users or analysis groups. In addition, the use of shared data offers the potential of better resource usage: there is less overhead due to multiple data copies and a higher chance of finding useful data in the staging pool or memory cache. This in turn translates to less data transfers to and from tape and over the network. Enabling such shared access is an important design decision that needs to be made as early as possible: it is far from easy to convert single user systems into multi-user ones at a late stage.

15.2.4 Scope of the Data Store

An early goal of the RD45 project was to find a solution capable of handling persistent data of all types. In other words, the system should be capable of handling the production data of an experiment (raw data, reconstructed data, analysis data, event tags etc.), statistical data (histograms etc.), detector-oriented data (calibration data etc.), as well as associated meta-data. It is assumed that the data store should meet the requirements of all of these domains.

15.2.5 Data and Meta-data

The bulk of the data in an HEP experiment is composed of data that is write-once and then read-only. These data – the raw data – are never modified. The reconstructed data, which also represents a significant data volume, is gradually rendered obsolete by more recent versions – perhaps produced with more recent calibrations, better algorithms or simply as the result of a bug fix. On the other hand, there remains an important fraction of the data that is frequently updated and for which concurrent access must be supported. Much of this data can be classified as meta-data: database schema, calibration data, tags and so forth. In the past, experiments have developed solutions for handling these updates without resorting to database technology. Examples of such systems include DBL3, FATMEN, HEPDB [28]



and OPCAL. However, all of these packages serialised updates through a single server – a highly non-scalable solution. Alternative approaches include separate systems for data and meta-data. However, both as the boundary between the two is somewhat arbitrary and as "navigation" between "data" and "meta-data" objects is essential (for example, raw data to calibration), the advantages of a unified approach are clear.

15.2.6 Consistency

In a large, distributed storage system, the problems of data consistency are significant. Problems that must be handled include:

- Partial writes – for example as the result of a program crash.
- Dirty writes – multiple jobs writing to the same set of files.
- Dirty reads – jobs accessing data that has only been partly updated.
- Unrepeatable reads – such as when the data is changed between analysis runs.

Handling such situations is often performed "by hand" in today's experiments – a highly labour intensive and unrewarding task. It is clearly highly desirable that the data store handle such cases consistently and, where possible, transparently.

15.2.7 Storage Manager Requirements

The principle requirements that must be met by the storage manager are as follows:

- **Volume** – the address space must be sufficient to cater for the needs of the LHC experiments.
  - The store must scale to at least 100PB, allowing a reasonable degree of flexibility in object placement (that is, it should not be necessary to fill each page / database to the absolute limit to satisfy the overall needs in terms of storage capacity.
  - Practical limitations on database / filesizes must be allowed for in the overall design of the store. Thus, a system that required 1TB files to satisfy the overall needs for capacity would not be acceptable.
- **Location transparency** – Support for heterogeneous and distributed systems.
  - Access to data must be transparent and independent of client or server hardware, operating system or compiler.
  - "Transportable" data formats are highly preferred – i.e. those that can be transferred using standard tools. This is important, not only to permit movement of data between different hosts (for example to better balance the load, or to move the data closer to the users), but also to cater for data that will be offline for long periods. It may well happen that the architecture used when a given database was written is no longer available when the data is retrieved, some years later, from the archive!
  - Users should not be required to know explicit file, host or even tape volume names in order to access their data.
- **Navigation** – Transparent and efficient navigation between any objects in the store must be provided.

  - It must be possible to directly access any object in the entire store. Access to such objects must be scalable – that is, the lookup time should depend no more and preferably less than linearly on the size of the store.
  - To support multiple streams – both per event and by event (for example different parts of each event versus different event classifications), both cross-file and cross-host navigation must be supported. Clearly, such navigation cannot use file and or host names – which are likely to change during the lifetime of the data and would imply a large overhead.
- **Consistency and Robustness** – the store must be highly reliable and consistent.
  - Consistent, concurrent access to shared data – including the possibility of updating – must be supported. Users performing analysis should not see the data change during the course of an analysis – leading to unreproducible results. However, it must still be possible to update the event store even whilst users are accessing it.
  - The system must be resilient to hardware and software problems, including crashing or badly behaved applications and hardware failure. It must be possible to recover from such problems – preferably automatically – with little or no impact on users.

15.2.8 Meta-data Requirements

Meta-data is usually defined as being "data describing data" – that is, data that makes other data usable. Examples of meta-data include the database schema, detector calibration data, catalogue and naming information, descriptions of event collections and so forth.

Based upon our experience with a commercial tool such as Objectivity/DB, we can identify the following additional requirements – that is, those that are not currently satisfied by the above tool. As such, the following list is clearly far from exhaustive.

- **Scalability**

The system should be able to handle schema and catalogue entries from several thousand users, without interfering with the production schema and catalogue.

- **Schema Validation**

The system should check for consistency between the classes used by the application, the schema stored in the system and the persistent objects residing in the store.

- **Support for Continuous Development**

The system should cater not only for "production mode", where production schema are relatively stable, but also for the continuous development mode, where schema changes are very frequent.

15.2.9 Language Binding Requirements

It is assumed that the most important language binding – at least for the immediate future – is C++. However, it is clear that other languages – in particular Java – will also play and important rôle and hence a strategy for adding new language bindings must be defined from the start. The question of interoperability of the different language bindings – in particular at the level of accessing data stored from other supported languages, must also be carefully considered.

For the C++ binding, important requirements include:

- **Full language Support**
  - All features of the language – particularly those that are widely used by existing applications – must clearly be supported. This includes support for templates, including the full STL, as well as polymorphism.
- **Standards compliant API**
  - The ODMG C++ binding should be supported.
  - Features currently not provided by Objectivity/DB include support for the *d_activate* and *d_deactivate* methods, plus a *d_Ref* class capable of pointing to both transient and persistent classes.
- **Support for multiple concurrent transaction contexts**
  - This feature is essential to permit multiple transactions – perhaps from multiple threads – against a single "federation".
  - Concurrent access to multiple federations (with consistent schema) should also be provided.

15.2.10 Summary

The key requirements can thus be summarised as follows:

- **Storage Manager**
  - Volume - Address space sufficient for LHC data stores
  - Location Transparency - Heterogeneous and distributed access
  - Navigation - Between arbitrary data objects
  - Consistency & Robustness
- **DB Meta Data**
  - Scalable DB Meta Data
  - Support for Schema Validation
  - Support for "continuuos development" mode
- **Language Binding**
  - Full C++ Language Support
  - Standards compliant API (ODMG)
  - Support for multiple concurrent transaction contexts
  - Other language bindings: Java, others?

15.3 General Design Issues

15.3.1 Scope of the Prototype

As described above, the scope of the prototype is to directly address the conclusions of the risk analysis requested by the LCB – namely to investigate the manpower requirements should it be decided that an alternative data management solution is required. It should be stressed that there is no intention to produce a product at this stage and that continued production services based upon Objectivity/DB remain high priority.

In addition, it is clear that a small prototype could be used for other purposes – such as prototyping new ideas, such as the effect of read-ahead, the impact of using different page sizes, and so forth.

15.3.2 Feasibility Issues

Although at the start of the RD45 project, it was estimated that a full ODBMS implementation required something like 150 man-years to bring to market, these estimates were based upon 1980's technology. Since that time, numerous ODBMS products have been brought to market – an important existence proof. It is also important to note that ODBMSs are no longer considered a research topic – the important issues are already well researched and documented in the literature. Finally, together with the wealth of experience that has been gained within the RD45 project, it is felt that the CMS estimate (see section 15.1 above) of some tens of man-years is more realistic for an implementation that was started today. This is not inconsistent with the above estimates – in addition to the improvements in technology that have occurred over the past ten years – such as to the C++ language itself – it is assumed that not all features of a full commercial system would be required. For example, it is currently assumed that features such as OQL and ODBC would not be required – at least in an initial version – and that the Smalltalk language binding can be safely ignored. Other features – such as synchronous replication – could perhaps be implemented in a simpler manner, taking into account the features of HEP event data.

To develop a system capable of meeting the draft requirements listed in section 15.2 above in time for the "2001 ODBMS selection" – that is by the end of 2000, would require a significant amount of effort in the intervening period. However, the development a system in time for the ramp up to the LHC, with a fully functional, tested and documented system ready by mid-2003, would appear more feasible. However, it is clear that much more detailed manpower studies – which can only come out of further prototyping – are required before any decision can be taken in this area. It is our intent to write detailed white papers on the key architectural components listed below, to allow extensive discussion of the ideas that are being studied, together with the preparation of realistic man-power estimates, by the end of 1999. These issues would then form a key element of the RD45 workshop that is scheduled for January 31 – February 4 2000.

15.3.3 Scalability Issues

It is clear that any new system should avoid the scalability problems of existing systems. This is somewhat easier for a "clean-sheet" design than for a deployed system, as there are no legacy constraints. Not only does the scalability of the storage hierarchy need to be considered, but also issues related to the handling of meta-data – i.e. the schema and catalogue.

15.3.4 HEP Data Models

Although HEP data models are complex, with many associations, these associations are typically rather local – e.g. within a given event. More generally, the data consists of weakly-coupled groups of tightly-coupled objects. Although a general purpose system cannot afford to make any assumptions about the data models that will be used, it would be possible to exploit this fact to simplify and optimise the system. Similarly, it is well known that the bulk of HEP data is read-only, potentially leading to significant simplifications. Can these features be exploited to result in a reduction of the implementation and support effort, whilst still offering sufficient flexibility? A possible approach would be to mark data as read-only in the catalogue. This would avoid the locking overhead for such data, improving the overall scalability of the system and support for multiple concurrent clients.

15.3.5 Conclusions

There appears to be a great deal of scope for exploiting the features of HEP data and ways of working to simplify the overall system, whilst retaining sufficient flexibility as to meet the overall requirements from the different application domains. The ideas presented above area clearly far from complete, but represent some initial thoughts in this area. Further discussions are clearly required and will be organised through RD45 meetings, mini-workshops and discussions with the experiments.

15.4 Architectural Overview

The overall architecture that is currently being prototyped is based upon the following storage hierarchy:

- **File** – the smallest component of the store, containing ***objects*** clustered on ***pages***. In addition to the data, a file could optionally contain ***schema***. Within the file, fast navigation would be performed using physical OIDs. Only relative OIDs are stored (within the scope of the containing file), allowing the file to be easily moved without requiring an update of all internal OIDs.
- **Domain** – a set of files that are tightly coupled. A domain has its own domain catalogue, data and optionally schema. Again, OIDs are relative within the scope of the domain.
- **Federation** – a set of loosely coupled domains. A catalogue of all domains and the global schema would also appear at this level. In both cases, the frequency of updates is

low. As the domains that make up a federation are loosely coupled, the performance penalty of a logical OID to cross domains is probably acceptable, although more studies need to be made in this area.

In order to satisfy requirements ranging from those of individual end-users to the global requirements of the production event store, the smallest usable database would consist of a single file. Scaling up, one could work with a set of files defined by a file-system directory, with optional schema, as shown in Figure 40.

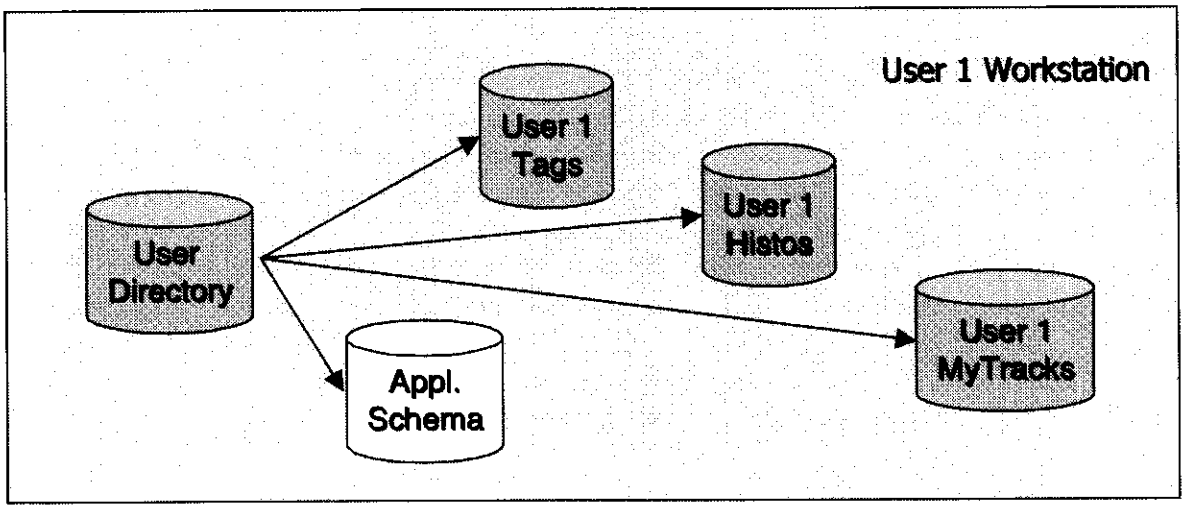


Figure 40 - A Set of User Files

Finally, one could construct a production federation out of multiple domains, as shown in Figure 41. Given the fact that all OIDs are relative, it would be trivial to move files or domains. It would also be possible to attach files to multiple domains or domains to multiple databases in read-only mode.

The above architecture is clearly both flexible and scalable and addresses a number of issues that are seen today. For example, by segmenting the catalogue and schema, the chance of locking the entire federation is greatly reduced. It also simplifies issues such as backup, data exchange and replication, which can work on a single domain, which is by definition logically coherent.

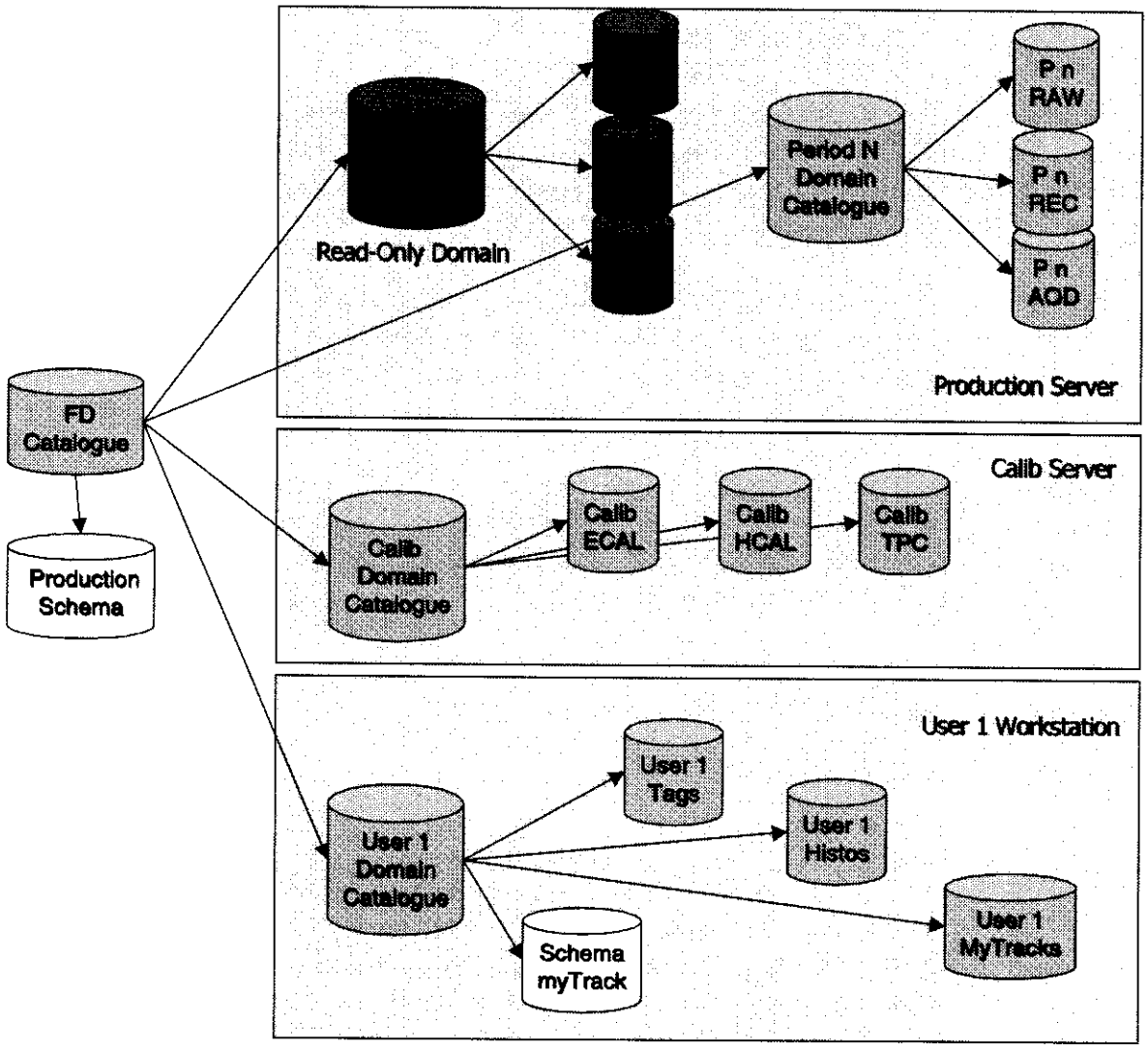


Figure 41 - A Production Federation Consisting of Multiple Domains

15.5 Design of the Prototype

15.5.1 Introduction

The main design choices of the current prototype are listed below, with more detailed descriptions in the following sections.

- The overall system is based on a page-server architecture with client-side page caching.
- Transactions are based on "shadow paging".
- Physical OIDs (96bit) are used, together with a flexible storage hierarchy.
- The C++ binding conforms to the ODMG standard.
- No schema pre-processor is required (and hence no code is generated).

15.5.2 Page Server vs Object Server

Commercial databases are implementing using a wide range of architectures. Some systems – such as Objectivity/DB – use a "fat-client" / "thin-server" model, where the bulk of the intelligence resides on the client side. Other systems – such as Versant – are built upon a more server-centric model. These two approaches can be classified as:

- **Page-server**: the database server simply transfers database pages to / from the client, with no knowledge of their contents. Such servers are rather simple to implement and to port to new platforms. The overhead on the server side is minimised, allowing a smaller number / smaller configuration servers to deal with a larger number of clients. In the case of well-clustered data, network I/O is also optimised.
- **Object-server**: only those objects that are required are transferred to / from the client. Requires considerably more processing than in the case of a page-server. Avoids the network overhead associated with badly clustered data, at the cost of accessing individual objects, rather than complete pages.

The page-server approach, being that adopted by Objectivity/DB, is well understood in the HEP community, and is that used in the current prototype.

15.5.3 Physical vs Logical OID

Another important choice that needs to be made concerns the mapping of the OID. Again, Objectivity/DB and Versant implement different approaches: the former using a *physical*[36] OID, the latter a *logical* OID. A physical OID leads to more predictable access times and better scalability, at the cost of less flexibility when it comes to re-clustering. A logical OID allows objects to be moved without changing their identity, but typically requires a redirection table, which does not scale well to large stores. As described in section 15.4 above, an interesting combination could be the use of physical OIDs within domains, allowing the possibility of logical OIDs across domains. The current prototype, however, implements only physical OIDs.

The OID mapping that is used consists of four fields: the domain, file, page and slot. In the current prototype, the domain and page fields occupy 32bits, whereas the file and slot fields are 16bit. These allocations are parameterised and further study is required to determine the optimum values.

[36] In fact, the OID in Objectivity/DB is not completely physical. For example, the page number refers to a logical page, which may be reallocated as the result of object resizing or other operations.

Federation
- **Set of weakly coupled domains**

Domain (32bit)
- **Set of tightly coupled objects**
- **e.g. a run or run period, an end-user workspace**

File (16bit)
- **A single file within a domain**

Page (32bit)
- **A single logical page in the file**

Slot (16bit)
- **A single data record on a page**

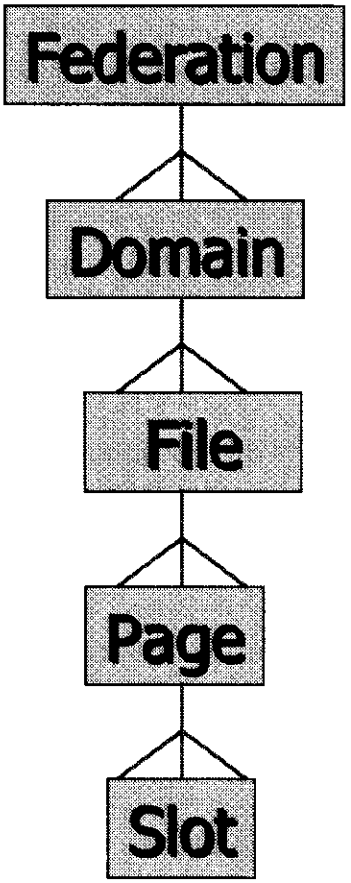


Figure 42 - OID Mapping

15.5.4 Transactions and Recovery

A variety of different approaches are used by DMBS systems for handling transactions and recovery. Once such technique is the use of log or journal files, where *un-do* or *re-do* records are stored. An alternative mechanism is the use of *shadow-pages*, where the new data are written directly to the database, but access indirectly through a page map. Modifications or new data are written to free physical pages and only become visible at commit time when the new page map is written to the database. Shadowing paging is well adapted to page-server environments, is robust to system crashes, and provides a simple clean-up mechanism. Objectivity/DB implements a form of shadowing paging, where the page map changes are logged in journal files. More recent forms of shadow paging – such as that implemented in the current prototype – obviate the need for journal files and exploit sequential writes (via appropriate logical – physical page mapping) to optimise performance.

15.5.5 Lock Server

A lockserver is required to support concurrent access from multiple clients. In the current prototype, locking is performed at the level of a file, with a queue of requestors in the case of locked resources. In an attempt to offer more scalability, the lockserver operates on the

level of a given dataserver, although it is currently implemented as a separate process. As is the case with the dataserver (pageserver), the lockserver is rather straightforward to implement, although detailed performance, reliability and scalability tests need to be performed.

15.5.6 Schema Handling

In order for the system to create or reload C++ objects in the address space of the client application, the schema of the classes has to be defined. Using the schema, the database has to perform the following tasks:

- Provide platform independent data access
  - Field position and size of attributes within a class
  - Byte ordering

- Support process address space independence
  - *vtable* and *vbase* pointer location and values[37]
  - embedded OIDs

- Enforce data consistency
  - Support bi-directional links
  - Handle lock propagation

Three distinct approaches can be taken to handle the schema definition of persistent classes:

1. The schema are defined in a special language – in other words, different to that used to write the application – and are processed by parser.

This approach is that followed by the SHORE persistent object manager, which uses the SDL language for defining schema. It is also used by relational databases, where SQL is used to define schema. An issue that must be addressed when using this technique is that of consistency between the C++ application and the database schema.

2. Using extended C++ syntax to describe the schema and process with a modified C++ parser.

An example of such an approach is Objectivity/DB.

3. Describe the schema using standard C++ syntax, as proposed by the ODMG standard.

In this case, one can either extract the schema using a modified C++ parser, as above, or else obtain the schema directly from the compiled program.

[37] See, for example, Lippman – "Inside the C++ Object Model" [40].

The current prototype implementation uses the last option, namely to extract the schema from debugging information. A standard format, used by the Sun and egcs compilers, is currently supported. Alternatively, a modified egcs front-end could also be used.

This technique supports run-time reflection for C++ classes, as follows:

- Describes classes and structs, with their fields and inheritance tree.
- Supports namespaces, typdefs, enums and templates.
- Provides the location and value of virtual function and virtual base class pointers.
- Is sufficient to allow field by field consistency checks against persistent schema at runtime.

15.5.7 C++ Language Binding

The current C++ language binding supports all main language features, such as virtual functions and templates. As opposed to the C++ binding of Objectivity/DB, no macros are needed – templates are used, as defined by the ODMG standard. Again, in contrast to Objectivity/DB, there is no need for automatic code generation – the functionality of refs, associations and iterators is all handled directly via templates. No language extensions are required, facilitating the integration of 3rd party tools, such as Rational Rose.

The binding is based upon that defined by the ODMG, although not all features (e.g. indices, bi-directional associations, lock propagation) are yet available. However, the list of classes implemented includes:

- *d_Object, d_Ref<T>, d_Ref_Any, d_Database, d_Transation, d_Varray<T>, d_Dictionary<T>, d_Error.*

The binding also solves are number of problems present in that of Objectivity/DB, namely:

- *d_Ref*s can point to both persistent and transient objects.
- Persistent-capable classes may be embedded in other persistent-capable classes.
- *d_active* and *d_deactivate* methods are supported.

The prototype is implemented in standard C++ and requires a compiler, such as egcs, that supports:

- **STL containers**
  - all containers in the current prototype are based on the STL.
- **Exceptions**
  - Used to signal error conditions, as defined by the ODMG.
- **Namespaces**
  - Implementation is in namespace "espresso".
  - C++ binding is in namespace "odmg".

15.6 Current Status

The current prototype is capable of handling the HTL histogram package. In order to build HTL with the prototype, rather than Objectivity/DB, only two changes are required:

1. The inclusion of a header file defining the ODMG d_Ref type.
2. The use of this d_Ref type.

These changes are shown in the code fragment below.

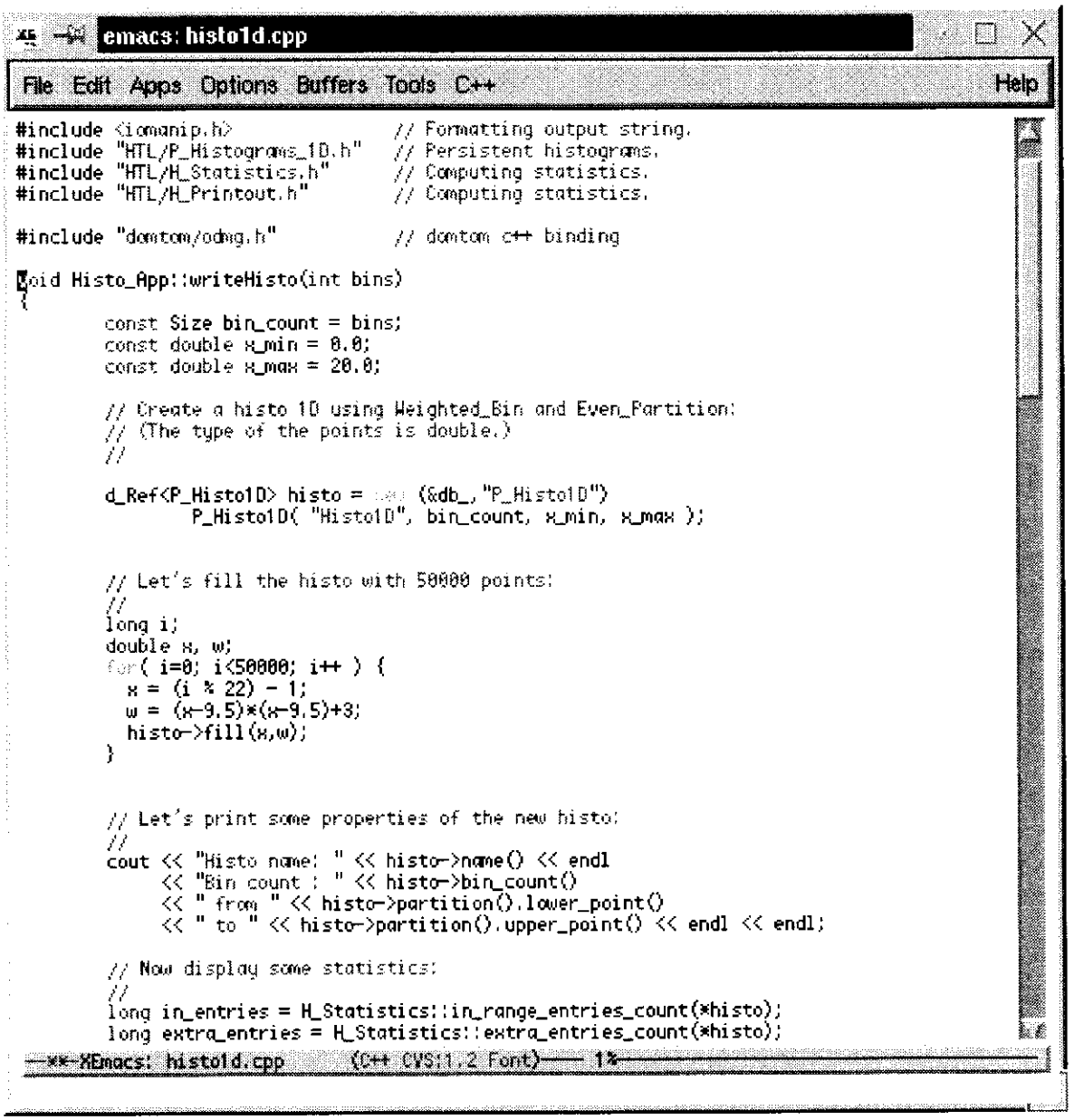


Figure 43 - HTL Built with Alternative Persistent Object Manager

Using Python/tk, a simple graphical browser has been built, as shown in Figure 44 below. This browser shows the class inheritance and member data of the persistent HTL objects that have been stored.

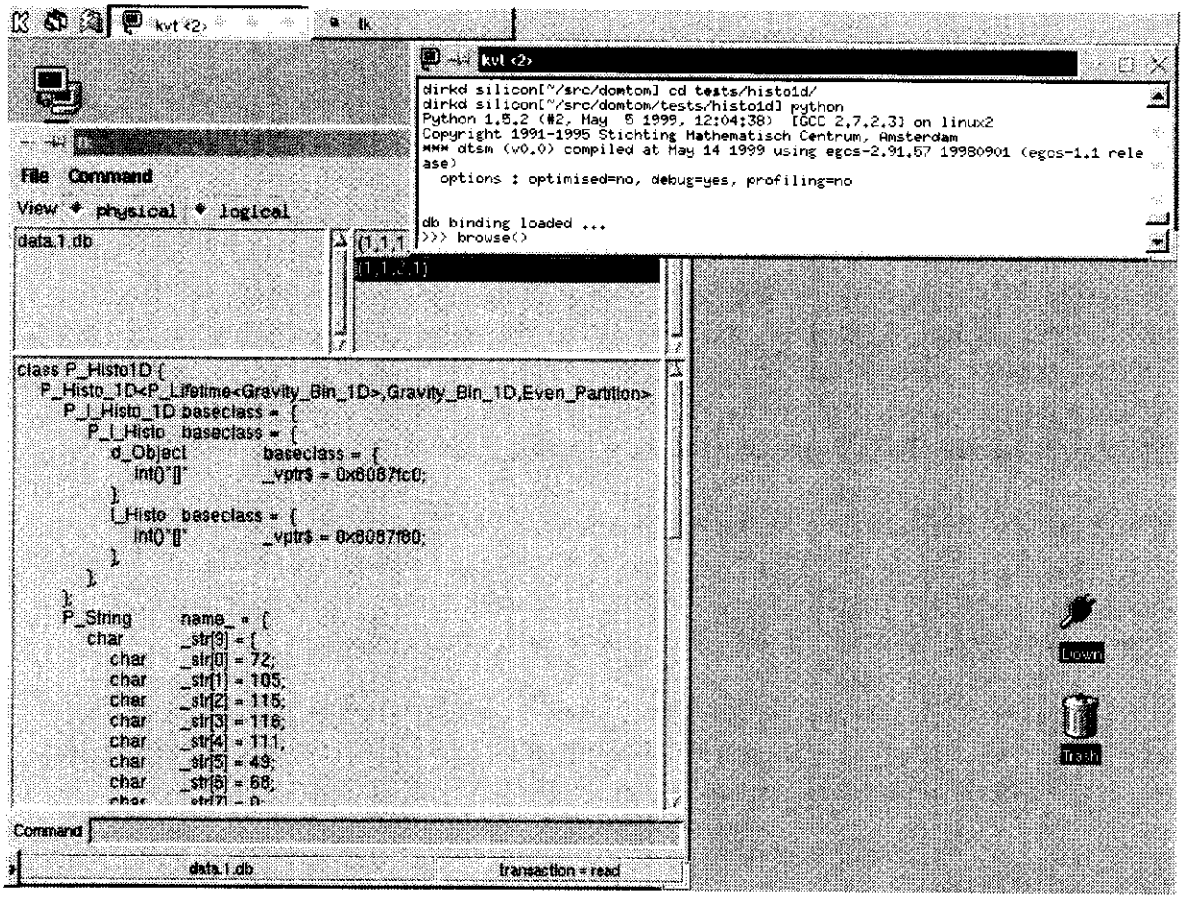


Figure 44 - Python Browser

15.7 Future Work

As stated above, there is no intention to develop a product at this stage. The investigations that have been performed have been strictly limited to the on-going risk analysis, in order to enable estimates of the manpower that would be required to develop a full alternative system to be made. However, it is intended that the system be developed to the stage that the various components (object manager, schema handler, lock server, data server etc.) can be used together and have sufficient functionality to store HTL histograms and event tags. It is felt that the requirements of histograms and event tags are sufficiently complex as to enable useful estimates to be made. The target date for such a version is October 1999.

15.8 Manpower Issues

The prototype alternative persistent object manager described above has been implemented as a background activity as part of the on-going risk analysis of the RD45 project over the

past 6-9 months. Although far from being a complete product – for which the exact requirements have not yet been finalised – it does suggest that an alternative solution, consistent with the overall strategy of RD45[38] could be indeed be built, should it be determined that such a step is necessary. Such a system is likely to require a minimum of 10 man-years, and perhaps as much as the "tens of man-years" foreseen by the CMS collaboration in 1996.

Today's estimates of the effort to bring a full commercial system to market are around 50 man-years. A non-commercial system – in particular, one designed to meet the needs and exploit the characteristics of HEP data – would almost certainly require less effort. It is likely that effort could be found within the LHC experiments and others, such as BaBar. A minimum of 1FTE / experiment could be expected. Coupled with a small team at CERN, 5-10 FTEs does not look impossible. It would still be very tight to produce a complete system by 2001 – indeed, experience suggests that any large software project requires at least three years for a stable system to be produced. However, a minimal system by 2001 and a complete solution by 2003 would seem to be feasible.

The possibility of collaborative development with users outside of the HEP community should also be considered. There are certainly other areas of physics that have similar needs – e.g. astrophysics, plasma physics etc. It is also possible that Universities and research institutes, such as INRIA, would also be prepared to provide effort.

These issues clearly require much further study and discussion. However, the first results are both positive and reassuring – it seems likely that an alternative solution could be built, if required and that the manpower requirements are not off-scale.

15.9 Conclusions

In direct response to the risk analysis requested by the LCB, the RD45 collaboration has developed a prototype alternative persistent object manager, with a view to estimating the effort required to build a production version of such a system, should the need arise. Initial results are encouraging and suggest that a complete system is indeed feasible. Further work in this area clearly requires careful discussion. However, it is important to stress that at least one realistic alternative to Objectivity/DB should be available on the timescale of 2001 and is considered particularly important given the continued small size of the ODBMS market.

[38] Single interface to all data, transparent navigation, scalable from the requirements of a single-user the full production system etc.

16 Standards Activities

In the context of RD45, CERN has *associate membership* of the Object Management Group (OMG) and is a *reviewer member* of the Object Database Management Group (ODMG). CERN is also represented in the IEEE Computer Society Executive Committee on Mass Storage – the body to which the various standards sub-groups (SSSWG) report.

Regular attendance at the IEEE Mass Storage Symposia has been continued: two papers were presented at the 16th Symposium, held in March 1999 in San Diego. These symposia provide up-to-date information on storage hardware and software and allow contacts with other sites to be established. On the other hand, the IEEE SSSWG, which recently dropped the existing set of draft standards for a new list of somewhat different scope, is not felt to be of direct importance to HEP. Even if draft standards are completed on target, it is not clear if conforming – and hence interoperable – products will be developed early enough to affect the initial production phase of the LHC.

In contrast to previous years, no ODMG meetings were attended. The reasons for this are twofold:

1. Most of the ODMG meetings conflicted with CERN events, such as the LCB workshop on LHC Computing in Barcelona.
2. The ODMG itself has focussed almost exclusively on Java, and changed its charter to be less Object Database specific. ODMG now stands for Object ***Data*** Management Group.

It is expected that a new revision of the ODMG standard – V3.0 – will be released during 1999 or early 2000. However, given that vendor compliance to the standard does not seem to be improving, regular attendance at the meetings no longer appears appropriate. However, as is also the case with the OMG and IEEE, continued membership is still recommended, if only for timely access to information on the database (or object / storage as appropriate) industry / market.

17 Future Activities

17.1 Introduction

Although many of the initial questions concerning the potential use of an ODBMS and MSS as the basis of a HEP data store have been answered – and much additional information will also be gained from the production experience of experiments such as BaBar, a number of important outstanding issues remain. These are primarily related to the on-going risk analysis and to the LHCC milestone regarding a choice of ODBMS in 2001. Whilst it is clear that production support for Objectivity/DB-based services must have high priority, we cannot afford to ignore the conclusions of the risk analysis, nor can we delay in preparing for the 2001 ODBMS selection. Thus, we believe that the future activities of the project should include:

1. Full support for Objectivity/DB-based production services.
2. A continuation of the risk analysis and proposed actions.
3. Preparation for the 2001 ODBMS selection.

17.2 Production Activities

Support of Objectivity/DB as part of a full production database service is now considered essential by several experiments. A significant amount of important data is already stored in Objectivity/DB, including event and calibration data, construction data produced by CMS/Cristal and soon simulation data.

In addition, the following items are considered of particularly high priority:

1. Transparent non-blocking interface with HPSS supporting at least 10 users. This is urgent and is required as soon as possible.
2. Tools and support for users with the capability to export, extract and replicate data and schema; manipulate data and schema outside the production database while accessing data and schema from the production database; import data and schema back into the production system.
3. Fully functional, reliable, high quality database system including very large database support, checking and control of new releases, a query language, Java support with C++ interworking and management tools.

17.3 Research Activities

It is felt that the presence of a framework for making the decision of a database system in 2001 (the timeframe set by the LHCC) is crucial. Such a decision should be based not only

on technical issues and practical experience, but also support and resource requirements, licensing issues (if applicable), risk analysis and so forth.

In addition, it is felt important that the current "white paper" series be maintained and extended, that future meeting and workshops are held and that contacts with industry and other ODBMS users be maintained. In short, those activities traditionally carried out by RD45 that did not explicitly address a particular LCB milestone or recommendation should continue.

In other words, the proposed R&D activities would include:

1. Support for the choice of database system at the end of 2001 including risk analysis, change of technology and migration issues, and gaining experience from other experiments such as BaBar, Fermilab experiments, NA45, etc.
2. Manpower estimate for an Alternative Persistent Object Manager. The aim is to determine the manpower needed to build a full system over a two-year period. The work could consist of a limited implementation followed by risk analysis.
3. A database independent software layer based largely on the ODMG interface standard.
4. The analysis and revision of LHC database requirements by mid 2000 which includes possible new issues and takes into account the knowledge gained since the original requirements were collected in 1995.

The potential use of a mainstream ORDBMS products, such as ORACLE 8*i* or a later release, should also be considered, if HEP is to benefit from a widely-used technology.

17.4 Summary

In addition to offering production services, we believe that the RD45 project should address the issues identified by the on-going risk analysis requested by the LCB, and should assist in the preparation for the choice of ODBMS for the LHC experiments that is scheduled for 2001. These issues are reflected in the proposed milestones, presented in section 18 below.

18 Proposed Milestones for 1999-2000

1. Together with IT/PDP group, continue to offer Objectivity/DB-based production services to meet the needs of ATLAS, CMS, CHORUS, COMPASS, NA45 and others.

2. Together with the LHC collaborations and in conjunction with the MONARC project, develop criteria by which an "ODBMS selection" can be made in 2001, in time for the ATLAS/CMS LHCC milestone.

3. Provide a concrete solution to the needs of end-user and "home-institutes" (non-RC), particularly for histogram, tag and calibration data. [Also simulated data?]

4. As part of the on-going "Risk Analysis", understand how such a solution could be extended, if necessary, to meet the needs of the main production sites. A requirements document, architectural overview and manpower estimate should be produced.

19 Conclusions

The RD45 project was approved in 1995 to investigate and propose solutions to the problems of handling the persistent objects of the LHC experiments: event data, calibration data, histograms and so forth. Strong emphasis has been placed on the potential use of standards-conforming, widely-used (commodity) solutions. At an early stage of the project, a potential solution, based upon an Object Database Management Group (ODMG)-compliant Object Database (ODBMS), coupled with a Mass Storage System (MSS) built according to the IEEE Computer Societies Reference Model for Mass Storage Systems, was identified. This potential solution has been the primary focus of our activities, although we have continued to monitor and evaluate alternatives. The preferred components of this solution are built on top of Objectivity/DB and HPSS, coupled with a small quantity of HEP-specific code.

Although numerous experiments have used Objectivity/DB over the past few years, a major milestone was achieved during the last year as the BaBar experiment entered production. The experience of BaBar – where some 100TB of data are expected per year – will clearly be extremely important in understanding if the specific systems identified above will be capable – perhaps in a future release – of handling the event data of the LHC experiments. Although it is too early to draw definitive conclusions from their experience, the fact that the initial production period has been successful is an important step in validating the overall approach.

Coupled with the experience of running experiments such as BaBar is the on-going risk analysis. Given the state of the ODBMS market – and indeed the lack of commodity solutions also in the MSS area – it appears likely that alternative, presumably "home-grown", solutions need also to be considered and the effort to develop and support such systems carefully evaluated.

20 Previous Milestones and Recommendations

We list below the milestones and recommendations from previous reviews of the RD45 project.

20.1 Milestones at the end of the second year (1997)

1. Demonstrate that an ODBMS can satisfy the requirements of typical simulation, reconstruction and analysis scenarios with data volumes of up to 1TB.
2. Investigate the impact on the every-day work of the end-user physicist when using an ODBMS for event data storage. The work should address issues related to individual developers' schema and collections for simulation, reconstruction and analysis.
3. Demonstrate the feasibility of using an ODBMS and MSS at data rates sufficient for ATLAS and CMS 1997 test-beam requirements.

In addition, the project is asked to include the following activities in its work-plan:

- Participate in the US-based related projects to gain experience in wide-area replication and use of a MSS.
- Investigate issues concerning porting data and applications from Objectivity to an alternative vendor's ODBMS product.
- Extend the performance comparisons made for the previous year's third milestone to investigate the potential advantages of adopting a new ODBMS based approach as the query and access method for physics analysis.
- Measure rates at which Objectivity can be populated with the aim of simulating event data input from a data acquisition system.
- Provide a guide to ODBMS usage for non-expert physicists and explain how it impacts their work.
- Provide guidance to LHC collaborations in the development of related database applications, such as management of calibration data etc.

20.2 Milestones at the end of the first year (1996)

1. Identify and analyse the impact of using an ODBMS for event data on the Object Model, the physical organisation of the data, coding guidelines and the use of third party class libraries.
2. Investigate and report on ways that Objectivity/DB features for replication, schema evolution and object versions can be used to solve data management problems typical of the HEP environment

3. Make an evaluation of the effectiveness of an ODBMS and MSS as the query and access method for physics analysis. The evaluation should include performance comparisons with PAW and Ntuples.

20.3 Initial Milestones and Recommendations (1995)

RD45 (P59) should be approved for an initial period of one year. The following milestones should be reached by the end of the first year.

1. A requirements specification for the management of persistent objects typical of HEP data together with criteria for evaluating potential implementations.
2. An evaluation of the suitability of ODMG's Object Definition Language for specifying an object model describing HEP event data.
3. Starting from such a model, the development of a prototype using commercial ODBMSs that conform to the ODMG standard. The functionality and performance of the ODBMSs should be evaluated.

It should be noted that the milestones concentrate on event data. Studies or prototypes based on other HEP data should not be excluded, especially if they are valuable to gain experience in the initial months.

21 Glossary

ADAMO - a system, developed in the ALEPH collaboration, based on the Entity-Relationship (ER) model.

ADSM - A storage management product from IBM

AFS - the Andrew (distributed) filesystem

CASE - Computer Aided Software Engineering

CORBA - the Common Object Request Broker Architecture, from the OMG

CORE - Centrally Operated Risc Environment

CWN - Column-wise Ntuple

CTP - Computing Technical Proposal

DFS - the OSF/DCE distributed filesystem, based upon AFS

DMIG - the Data Management Interface Group

EDMS - Engineering Data Management System

GB - 10^9 bytes

HPSS - High Performance Storage System - a high-end mass storage system developed by a consortium consisting of end-user sites and commercial companies

IEEE - the Institute of Electrical and Electronics Engineers

KB - 2^{10} (1024) bytes - normally referred to as 10^3 bytes

LCB - LHC Computing Board

LCRB - LHC Computing Review Board

LIGHT - Life Cycle Global Hypertext

MB - 10^6 bytes

MSS - a Mass Storage System

NFS - the Network Filesystem, developed by Sun

ODBMS - an Object Database Management System

ODMG - the Object Database Management Group, a group of database vendors and users that develop standards of ODBMSs

OID - Object Identifier

OMG - the Object Management Group

OOFS - the Objectivity/DB Open FileSystem

OQL - the Object Query Language defined by the ODMG

ORB - an Object Request Broker

OSM - Open Storage Manager: a commercial MSS

PAW - the Physics Analysis Workstation

PETASERVE - a MSS based upon OSM

PB - 10^{15} bytes

RWN - Row-wise Ntuple

SHORE - Scalable Heterogeneous Object REpository

SQL - Standard Query Language: the language used for issuing queries against databases

SSSWG - the Storage System Standards Working Group

STL - the Standard Template Library: part of the draft C++ standard albeit in a modified form

TB - 10^{12} bytes

TOOLS.H++ - a former de-facto standard container/collection class library, largely made redundant by the collections provided in the standard C++ library

VLDB - Very Large Database

VLM - Very Large Memory

VMLDB - Very Many Large Databases

XBSA - the draft X/Open Backup Services Application Program Interface

22 References

[1] RD45 - A Persistent Object Manager for HEP, LCB Status Report, March 1998, CERN/LHCC 98-11

[2] RD45 Project Execution Plan, 1997-1998, April 1997, CERN/LCB 97-10

[3] RD45 - A Persistent Object Manager for HEP, LCB Status Report, March 1997, CERN/LHCC 97-6

[4] Object Databases and their Impact on Storage-Related Aspects of HEP Computing, the RD45 collaboration, CERN/LHCC 97-7

[5] Object Database Features and HEP Data Management, the RD45 collaboration, CERN/LHCC 97/8

[6] Using and Object Database and Mass Storage System for Physics Analysis, the RD45 collaboration, CERN/LHCC 97-9

[7] Where are Object Databases Heading? CERN/RD45/1996/4

[8] Why Objectivity/DB? CERN/RD45/1996/6

[9] Objectivity/DB Database Administration Issues. CERN/RD45/1996/7

[10] RD45 - A Persistent Object Manager for HEP, LCRB Status Report, March 1996, CERN/LHCC 96-15

[11] Object Databases and Mass Storage Systems: The Prognosis, the RD45 collaboration, CERN/LHCC 96-17

[12] Object Data Management. R.G.G. Cattell, Addison Wesley, ISBN 0-201-54748-1

[13] DBMS Needs Assessment for Objects, Barry and Associates (release 3)

[14] The Object-Oriented Database System Manifesto M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, and S. Zdonik. In Proceedings of the First International Conference on Deductive and Object-Oriented Databases, pages 223-40, Kyoto, Japan, December 1989. Also appears in [19].

[15] Object Oriented Databases: Technology, Applications and Products. Bindu R. Rao, McGraw Hill, ISBN 0-07-051279-5

[16] Object Databases - The Essentials, Mary E. S. Loomis, Addison Wesley, ISBN 0-201-56341-X

[17] An Evaluation of Object-Oriented Database Developments, Frank Manola, GTE Laboratories Incorporated

[18] Modern Database Systems - The Object Model, Interoperability and Beyond, Won Kim, Addison Wesley, ISBN 0-201-59098-0

[19] Objets et Bases de Données - le SGBD O_2, Michel Adiba, Christine Collet, Hermes, ISBN 2-86601-368-9

[20] Object Management Group. The Common Object Request Broker: Architecture and Specification, Revision 1.1, OMG TC Document 91.12.1, 1991.

[21] Object Management Group. Persistent Object Service Specification, Revision 1.0, OMG Document numbers 94-1-1 and 94-10-7.

[22] The Object Database Standard, ODMG-93, Edited by R.G.G.Cattell, ISBN 1-55860-302-6, Morgan Kaufmann.

[23] ADAMO Reference Manual, CERN ECP

[24] HBOOK - Statistical Analysis and Histogramming Package - CERN Program Library Long Writeup, Y250

[25] PAW - the Physics Analysis Workshop - CERN Program Library Long Writeup, Q121

[26] ATLAS Computing Technical Proposal, CERN/LHCC 96-43

[27] CMS Computing Technical Proposal, CERN/LHCC 96-45

[28] HEPDB - A Distributed Database Management System, CERN Program Library Long Writeup Q180

[29] C++ Object Databases - Programming with the ODMG Standard, David Jordan, Addison Wesley, ISBN 0-201-63488-0, 1998

[30] The CMS H2 Object Oriented Reconstruction Project.

[31] The CMS X5 Object Oriented Reconstruction Project, http://cmsdoc.cern.ch/~lucia/tbeam/OO_staff/x5oo.html.

[32] Design Patterns: Elements of Reusable Object-Oriented Software, Gamma, Helm, Johnson, Vlissides, http://consult.cern.ch/books/0201633612.

[33] STL Tutorial and Reference Guide: C++ Programming with the Standard Template Library, David R. Musser, Atul Saini, Addison Wesley, ISBN 0-201-63398-1

[34] Object Databases in Practice, Prentice Hall, ISBN 0-13-899725-X.

[35] Proceedings of the 1998 CERN School of Computing, CERN Yellow Report 98-08.

[36] Proceedings of the 1999 CERN School of Computing, to be published.

[37] Reclustering of HEP Data, Martin Schaller, in proceedings of SSDBM '99.

[38] K. Holtman, P. van der Stok, I. Willers: Automatic Reclustering of Objects in Very Large Databases for High Energy Physics, Proc. of IDEAS '98, Cardiff, UK, p. 132-140, IEEE 1998.

[39] Object-Relational DBMSs – The Next Great Wave. Michael Stonebraker with Dorothy Moore, Morgan Kaufmann, ISBN 1-55860-397-2.

[40] Inside the C++ Object Model, Stanley B. Lippman, Addison Wesley, ISBN 0-201-83454-5.