HARP DATA AND SOFTWARE MIGRATION FROM OBJECTIVITY TO ORACLE

A.Valassi, D.Geppert, M. Lübeck, K. Nienartowicz, M. Nowak, CERN IT-DB, Geneva, Switzerland
E. Tcherniaev, CERN PH, Geneva, Switzerland
D. Kolev, University of Sofia, Sofia, Bulgaria

Abstract

The migration of the HARP data and software from an Objectivity-based to an Oracle-based storage solution is reviewed. The project, which was successfully completed in January 2004, included three distinct phases, involving the migration of the HARP raw event data, event collection metadata and conditions data, respectively, and of the software needed to handle each of these data types. All three phases are described in detail in this paper.

INTRODUCTION

The physics goal of the HARP experiment [1] at CERN is the systematic and precise study of hadron production for beam momenta between 2 and 15 GeV/c, for target nuclei ranging from hydrogen to lead. The motivation is twofold: to acquire adequate knowledge of pion yields for the optimal design of a neutrino factory, and to improve the calculation of the atmospheric neutrino flux for a better interpretation of atmospheric neutrino experiments.

HARP took data during three months in 2001 and six more months in 2002 [2], with all subsystems in operation and no appreciable down-time. Physics data were recorded during the "spills" of the T9 beam from the PS against fixed targets, placed inside a TPC upstream of a large instrumented spectrometer. Almost 200M and 600M events were collected in 2001 and 2002, respectively. This corresponds to a total of more than 20k data-taking runs, for over 800 settings of beam momentum, beam polarity and target type, grouped in a dozen collections.

During data-taking, all HARP data were stored using Objectivity, an object database management system. Two Objectivity "federations" were used to store the data from 2001 and 2002. Because of the termination of Objectivity support at CERN and of the need for long-term storage and analysis of the data, all HARP data and software were migrated in 2003 to a different storage solution, based on the Oracle relational database management system. The migration only concerned the raw data collected on beam time, while no samples of reconstructed or simulated data had to be moved to the new storage system.

The project, which was carried out in collaboration by the Database Group of the CERN IT Department (IT-DB) and the HARP experiment, was successfully completed in January 2004. The migration involved three distinct phases, described in detail in the next sections. In the first phase, which profited significantly from the previous COMPASS migration [3-4], 30 TB of HARP raw event data were migrated in two weeks to a hybrid persistency solution, storing raw event records in "flat" files on tape and the corresponding metadata in Oracle relational tables. In the second phase, the longest to achieve in spite of the limited data volumes involved, the complex data model of HARP event collections and the software to provide read-only navigational access to event data were re-implemented for the new Oracle solution. The third phase was the easiest, as it involved the migration of conditions data from the Objectivity to the Oracle implementation of a same C++ API, which acted as a screening layer between the abstract data model and its implementation in the two persistent technologies.

RAW EVENT DATA MIGRATION

The first phase of the project involved the large-scale migration of the 30 TB of raw event data collected by HARP. The preparation for this migration started as early as March 2002 and was carried on jointly with that of the even larger migration of the 300 TB of COMPASS raw event data, so that the same storage model and most of the hardware configuration and software tools initially set up for COMPASS were eventually reused for HARP too.

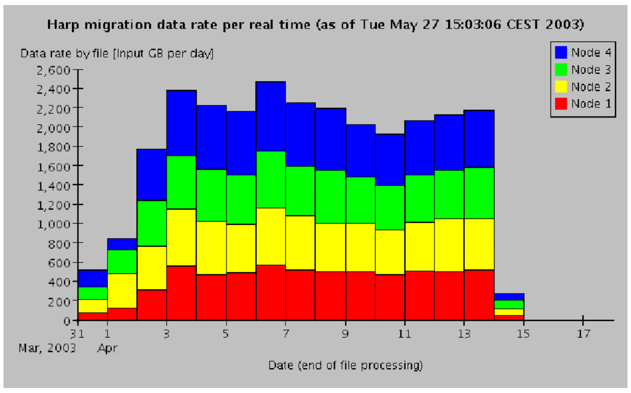


Figure 1 HARP raw event data migration speed.

The benefit for HARP of the earlier COMPASS experience can be fully appreciated by considering that, while the COMPASS migration only reached its design data transfer rate of 100 MB/s in its final month of February 2003 [4], the HARP raw event data migration proceeded essentially at its full design speed of 30 MB/s (over 2 TB/day) throughout the two weeks of April 2003 during which it was executed, as shown in Figure 1. The major difference between the two projects is that the new COMPASS storage solution had to be in place before the experiment took new data in the summer 2003, while the HARP migration was performed under weaker time pressure after the experiment had completed data-taking.

Raw binary data records and event metadata

The HARP Objectivity data model, out of which the data were migrated, includes a number of persistent C++ classes representing the entities defined in the hierarchical organization of event data, such as collections, settings, runs, spills and events. Because of the parallelism of the DAQ, most runs/spills are also split into two "partial" runs/spills, containing only one out of every two events, those processed by the same event builder. For some runs, only one partial run exists as one event builder was used.

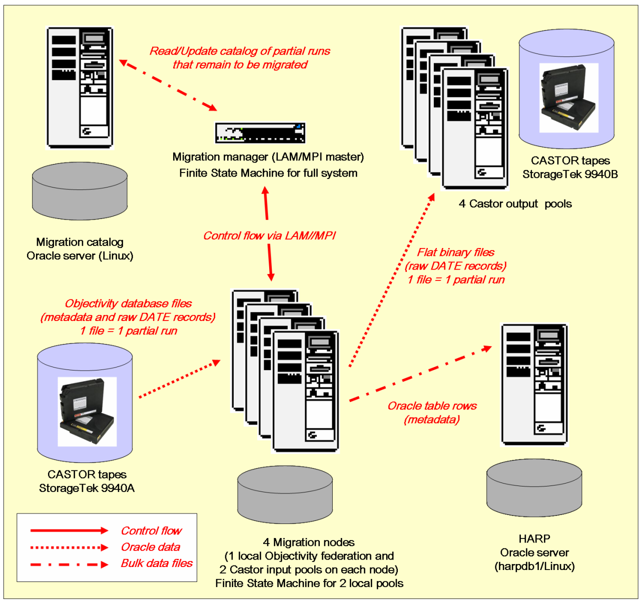
The Objectivity C++ objects associated to events, spills and runs (both partial and whole) essentially act as "headers" to allow the hierarchical navigation of the data model (e.g., from runs to the spills they contain and from spills to events): they do not contain any actual data, but only pointers to other persistent objects encapsulating the corresponding raw data records in the DATE [5] binary format, which is independent of Objectivity. In the new "hybrid" persistency solution, these binary records are sequentially stored on flat files, while the corresponding "metadata" (for each record, its size and its byte offset on file, and for each partial run, the name of the file where its records are stored, plus all navigational links between the various headers) are stored in Oracle relational tables, as shown in Figure 3 at the end of this note.

The total size of the HARP event metadata in Oracle is less than 100 GB (plus 110 GB of Oracle indexes), while the bulk of the 30 TB of raw data is accounted for by the flat files, archived on tapes in the CASTOR mass storage system. The files are transparently staged in via RFIO [6] by the C++ component providing on demand read access to the raw binary records. While this model, used for both COMPASS and HARP, has many similarities to the new persistency solution that is being developed for the LHC experiments, POOL [7], based on an object streaming layer supplemented by a relational database for object metadata, the POOL software did not reach production quality in time to be used for either of the two migrations.

It should also be noted that the possibility to store the binary records as BLOBs in Oracle had initially been considered. The idea was dropped because it implied the need to keep some Oracle data files on tape and develop a mechanism to transparently stage them in and load them in the database, similar to what was done for Objectivity using AMS [4], while it involved no obvious benefit as BLOB columns are not meant to be queried using SQL.

Partial run file migration

Partial runs played a key role in the first phase of the migration, because they correspond to separate files in the original physical partitioning model of HARP event data. An Objectivity "database" file for a partial run is by itself a complete data set, including all metadata necessary to navigate to the partial spills and events it contains, and all corresponding raw data records. As a consequence, the migration simply consisted in processing all partial run files independently, on several nodes in parallel: for each partial run and all the partial spills and events inside it, all raw DATE records were copied into a single flat file, while the corresponding metadata were stored in Oracle.



Read/Update catalog of partial runs that remain to be migrated
Migration manager (LAM/MPI master)
Finite State Machine for full system
Migration catalog
Oracle server (Linux)
CASTOR tapes
StorageTek 9940B
4 Castor output pools
Control flow via LAM//MPI
Flat binary files
(raw DATE records)
1 file = 1 partial run
Objectivity database files
(metadata and raw DATE records)
1 file = 1 partial run
Oracle table rows
(metadata)
CASTOR tapes
StorageTek 9940A
Control flow
Oracle data
Bulk data files
4 Migration nodes
(1 local Objectivity federation and
2 Castor input pools on each node)
Finite State Machine for 2 local pools
HARP
Oracle server
(harpdb1/Linux)

Figure 2 HARP raw event data migration hardware setup

The topology of the hardware and software framework for the HARP migration, shown in Figure 2, is almost the same as that previously used for COMPASS. As in that case, a LAM/MPI communication protocol is used to synchronise the dispatching of jobs on the conversion nodes, all individually modelled as finite state machines. The main difference is that 4 conversion nodes were used for HARP instead of 11, which explains the lower data transfer rate. A complete description of this framework is presented in Reference [3] and will not be repeated here.

EVENT COLLECTION MIGRATION

The second phase of the project involved the migration into Oracle of the metadata for the higher level HARP data model entities: collections, settings, "whole" runs and spills. The first three of these are stored in the tables at the top of Figure 3, while the seven tables at the bottom represent the data extracted from the previous migration of Objectivity partial run files (federations, partial runs and spills, events, partial run records, event and record types). While the volume of additional data migrated does not reach 1 GB, this phase took many months to complete as it also involved extensive internal consistency cross-checks on these data and the development of the software to provide read-only navigational data access.

Note that a table for "whole" spills is not present in the schema, because the only extra attribute of spills with respect to partial spills (i.e. the information whether their start-of-burst and end-of-burst records come from the first or second partial spill) is the same for all spills in a run and is stored in the run table. Similarly, a table for the run records associated to a whole run is not needed, as it is enough to store in the run table the "ID" of the partial run from which they are taken. Spills and run records, instead, are represented by Oracle "views", virtual tables whose rows are dynamically queried from other "base" tables.

Views and "materialized" views (special stored tables whose rows, derived from other tables, are automatically refreshed by the system) are also extensively used to simplify or speed up the access to other quantities that are otherwise available only through aggregate operations or complex joins, such as the number of physics events in a run. Other performance-enhancing Oracle features are also widely used in the HARP schema: indexes are added on the most frequently queried columns, such as the type of an event, and partitioning by ranges of run numbers is used for all tables on which it can be applied. SQL bind variables are used throughout the code to minimise the network roundtrips to the server for both write (bulk insertion) and read (result set caching) data access, implemented using the Oracle C++ client library (OCCI).

Migration of event collection metadata

The migration of event collection metadata proceeded by looping hierarchically through all the data in each federation, from collections to settings, to runs and finally to spills. This was necessary, for instance, because in the HARP Objectivity data model it is possible to know which runs a given setting contains, but it is not possible, given a run, to determine the setting it belongs to. In the Oracle model, conversely, each run is stored together with the setting, and even the collection, it belongs to. Note that the data model in Figure 3 is not fully "normalized" and clearly exhibits some redundancies, introduced either for convenience or for performance reasons. The partial run table, for instance, contains a collection field (added during the second phase of the migration), even if this can also be obtained by navigating back through the run and setting tables. To prevent any inconsistency, integrity constraints based on redundant combinations of unique and foreign keys were introduced, not shown in Figure 3.

Extensive cross-checks of the internal consistency of the Objectivity data were also performed. Runs that in Objectivity had been assigned to more than one setting, for instance, had to be handled on a case by case basis, while "fake" collections were introduced to group settings that were found to belong to no real collection. Also, during the "merge" of partial runs and spills into whole runs and spills, special treatment had to be used for those runs whose two partial runs contain a different number of partial spills, and for runs whose run records had to be reassigned to a different partial run than that indicated in Objectivity, as the latter had been lost or had failed its migration to Oracle during the first phase of the project.

Oracle implementation of C++ read access

One of the most interesting aspects of the project was the re-implementation using Oracle of the C++ event selection and hierarchical event looping, within the HARP Gaudi framework. The challenge was to fully exploit the server-side SQL query capability of the Oracle server for event selection, with no modification required to the end-user code and job options for batch analysis. The goal was successfully achieved by the re-implementation from scratch of the two software components responsible for the data retrieval from persistent storage and for the event loops and selection. In the Objectivity-based system, these tasks are handled by the two packages called ObjyHarp and EventSelector. The latter has a software dependency on the former: event selection, for instance, is handled by class FOEventSelection, a function object taking as argument an instance of the ObjyEventBase class, modelling persistent Objectivity events. Looping on events satisfying a selection is achieved as a full scan, i.e. by loading into C++ memory all Objectivity events and then applying on each event the FOEventSelection cut.

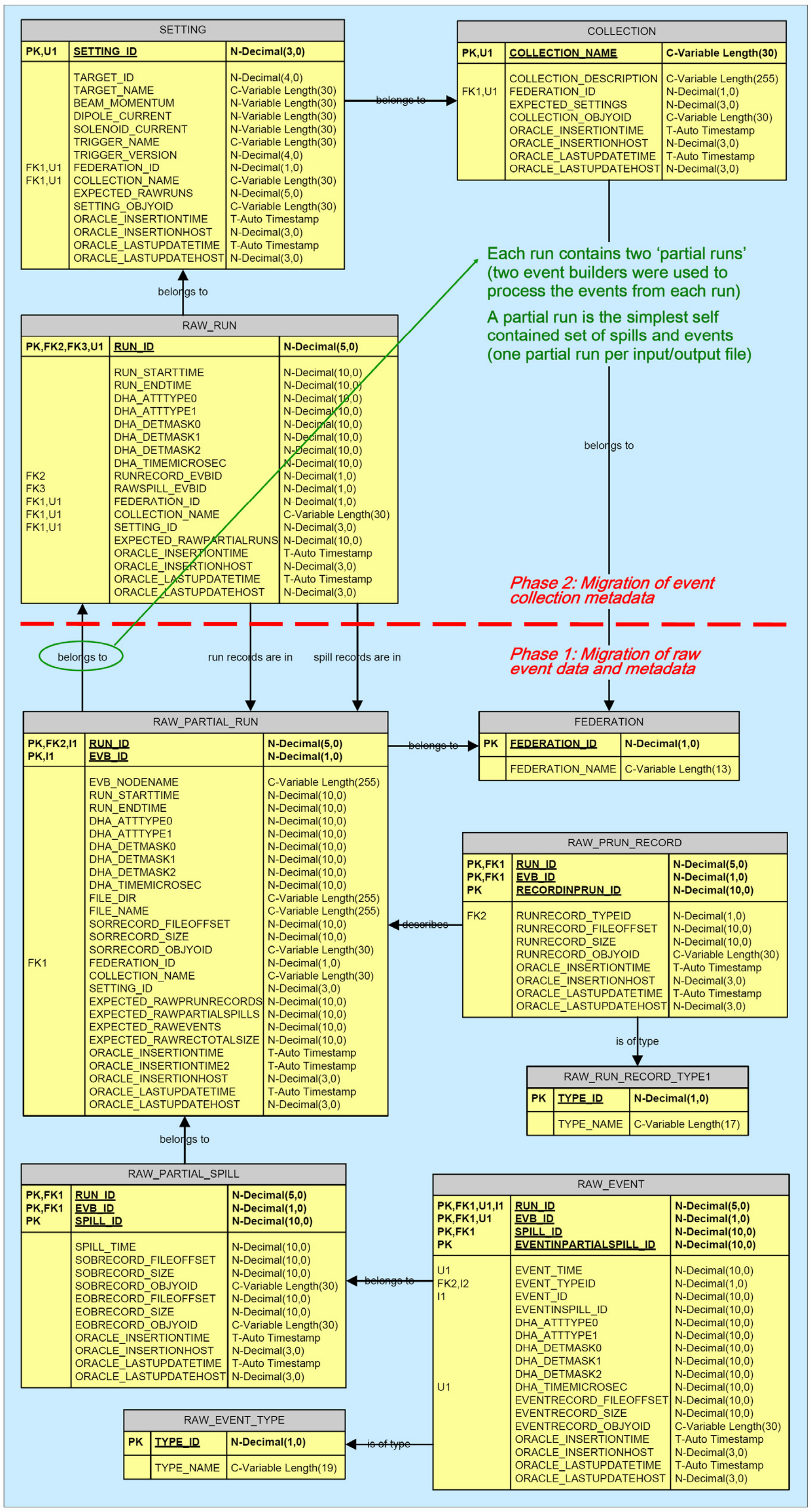
The approach followed in the new Oracle-based system is radically different. A third package is introduced, called PersistentEventModelAndSelection, where a generic C++ interface IEventHeader models persistent event headers independently of the technology they are stored in, while another interface IEventSelection represents an event selection that can be applied to any such event header. With respect to FOEventSelection, IEventSelection also has a method to describe itself as a string. A generic class HierarchicalEventLoopMgrBase for hierarchical event loops is also included in this base package, together with the machinery to translate into instances of a generic selection class the selection criteria specified in the user job option files. The very thin packages OracleHarp and OracleEventSelector simply implement these interfaces and overload some of the methods defined in the base class, for the Oracle case. In particular, the description of an Oracle event selection returns a "WHERE" clause string, while the loop algorithm is modified so that this clause is included in the SQL query that is executed on the server to retrieve the Oracle event headers.

Extensive tests were performed to ensure that the same results as in the old Objectivity implementation could be obtained using the new Oracle-based software. The base algorithms of PersistentEventModelAndSelection were even tested after re-implementing the new approach for Objectivity in a thin adapter package ObjyEventSelector, which was very fast to develop and proved very useful.

CONDITIONS DATA MIGRATION

The final phase of the project involved the migration of a few GB of conditions data, time-dependent non-event data describing the state of the detector at the time events were taken. This phase was the easiest and took less than a month in total, because the data, initially stored using the Objectivity implementation of a generic C++ API for Conditions Database access, were simply migrated to the Oracle implementation of the same API [8]. Three classes of data were migrated: calibration and alignment data, for which the names of the files containing the actual data were stored in the database, and controls data for the T9 beam and the HARP detector, both encoded as vectors of numbers streamed into binary records in the database.

The tool used to migrate the data was based on the C++ API only in part, as direct access to the Oracle back-end had to be used to speed up the insertion of the 30M rows of data with different intervals of validity. The component providing read access to conditions data from the HARP framework, instead, only needed minimal modifications as the large majority of the code originally written for the Objectivity back-end only depended on the abstract C++ API and could be reused unchanged for the Oracle case.



belongs to
belongs to
belongs to
Each run contains two 'partial runs' (two event builders were used to process the events from each run)
A partial run is the simplest self contained set of spills and events (one partial run per input/output file)
Phase 2: Migration of event collection metadata
Phase 1: Migration of raw event data and metadata
belongs to
run records are in
spill records are in
belongs to
describes
is of type
belongs to
belongs to
is of type

SETTING

PK,U1	SETTING_ID	N-Decimal(3,0)
	TARGET_ID	N-Decimal(4,0)
	TARGET_NAME	C-Variable Length(30)
	BEAM_MOMENTUM	N-Variable Length(30)
	DIPOLE_CURRENT	N-Variable Length(30)
	SOLENOID_CURRENT	N-Variable Length(30)
	TRIGGER_NAME	C-Variable Length(30)
	TRIGGER_VERSION	N-Decimal(4,0)
FK1,U1	FEDERATION_ID	N-Decimal(1,0)
FK1,U1	COLLECTION_NAME	C-Variable Length(30)
	EXPECTED_RAWRUNS	N-Decimal(5,0)
	SETTING_OBJYOID	C-Variable Length(30)
	ORACLE_INSERTIONTIME	T-Auto Timestamp
	ORACLE_INSERTIONHOST	N-Decimal(3,0)
	ORACLE_LASTUPDATETIME	T-Auto Timestamp
	ORACLE_LASTUPDATEHOST	N-Decimal(3,0)

COLLECTION

PK,U1	COLLECTION_NAME	C-Variable Length(30)
	COLLECTION_DESCRIPTION	C-Variable Length(255)
FK1,U1	FEDERATION_ID	N-Decimal(1,0)
	EXPECTED_SETTINGS	N-Decimal(3,0)
	COLLECTION_OBJYOID	C-Variable Length(30)
	ORACLE_INSERTIONTIME	T-Auto Timestamp
	ORACLE_INSERTIONHOST	N-Decimal(3,0)
	ORACLE_LASTUPDATETIME	T-Auto Timestamp
	ORACLE_LASTUPDATEHOST	N-Decimal(3,0)

RAW_RUN

PK,FK2,FK3,U1	RUN_ID	N-Decimal(5,0)
	RUN_STARTTIME	N-Decimal(10,0)
	RUN_ENDTIME	N-Decimal(10,0)
	DHA_ATTTYPE0	N-Decimal(10,0)
	DHA_ATTTYPE1	N-Decimal(10,0)
	DHA_DETMASK0	N-Decimal(10,0)
	DHA_DETMASK1	N-Decimal(10,0)
	DHA_DETMASK2	N-Decimal(10,0)
	DHA_TIMEMICROSEC	N-Decimal(10,0)
FK2	RUNRECORD_EVBID	N-Decimal(1,0)
FK3	RAWSPILL_EVBID	N-Decimal(1,0)
FK1,U1	FEDERATION_ID	N Decimal(1,0)
FK1,U1	COLLECTION_NAME	C-Variable Length(30)
FK1,U1	SETTING_ID	N-Decimal(3,0)
	EXPECTED_RAWPARTIALRUNS	N-Decimal(10,0)
	ORACLE_INSERTIONTIME	T-Auto Timestamp
	ORACLE_INSERTIONHOST	N-Decimal(3,0)
	ORACLE_LASTUPDATETIME	T-Auto Timestamp
	ORACLE_LASTUPDATEHOST	N-Decimal(3,0)

RAW_PARTIAL_RUN

PK,FK2,I1	RUN_ID	N-Decimal(5,0)
PK,I1	EVB_ID	N-Decimal(1,0)
	EVB_NODENAME	C-Variable Length(255)
	RUN_STARTTIME	N-Decimal(10,0)
	RUN_ENDTIME	N-Decimal(10,0)
	DHA_ATTTYPE0	N-Decimal(10,0)
	DHA_ATTTYPE1	N-Decimal(10,0)
	DHA_DETMASK0	N-Decimal(10,0)
	DHA_DETMASK1	N-Decimal(10,0)
	DHA_DETMASK2	N-Decimal(10,0)
	DHA_TIMEMICROSEC	N-Decimal(10,0)
	FILE_DIR	C-Variable Length(255)
	FILE_NAME	C-Variable Length(255)
	SORRECORD_FILEOFFSET	N-Decimal(10,0)
	SORRECORD_SIZE	N-Decimal(10,0)
	SORRECORD_OBJYOID	C-Variable Length(30)
FK1	FEDERATION_ID	N-Decimal(1,0)
	COLLECTION_NAME	C-Variable Length(30)
	SETTING_ID	N-Decimal(3,0)
	EXPECTED_RAWPRUNRECORDS	N-Decimal(10,0)
	EXPECTED_RAWPARTIALSPILLS	N-Decimal(10,0)
	EXPECTED_RAWEVENTS	N-Decimal(10,0)
	EXPECTED_RAWRECTOTALSIZE	N-Decimal(10,0)
	ORACLE_INSERTIONTIME	T-Auto Timestamp
	ORACLE_INSERTIONTIME2	T-Auto Timestamp
	ORACLE_INSERTIONHOST	N-Decimal(3,0)
	ORACLE_LASTUPDATETIME	T-Auto Timestamp
	ORACLE_LASTUPDATEHOST	N-Decimal(3,0)

FEDERATION

PK	FEDERATION_ID	N-Decimal(1,0)
	FEDERATION_NAME	C-Variable Length(13)

RAW_PRUN_RECORD

PK,FK1	RUN_ID	N-Decimal(5,0)
PK,FK1	EVB_ID	N-Decimal(1,0)
PK	RECORDINPRUN_ID	N-Decimal(10,0)
FK2	RUNRECORD_TYPEID	N-Decimal(1,0)
	RUNRECORD_FILEOFFSET	N-Decimal(10,0)
	RUNRECORD_SIZE	N-Decimal(10,0)
	RUNRECORD_OBJYOID	C-Variable Length(30)
	ORACLE_INSERTIONTIME	T-Auto Timestamp
	ORACLE_INSERTIONHOST	N-Decimal(3,0)
	ORACLE_LASTUPDATETIME	T-Auto Timestamp
	ORACLE_LASTUPDATEHOST	N-Decimal(3,0)

RAW_RUN_RECORD_TYPE1

PK	TYPE_ID	N-Decimal(1,0)
	TYPE_NAME	C-Variable Length(17)

RAW_PARTIAL_SPILL

PK,FK1	RUN_ID	N-Decimal(5,0)
PK,FK1	EVB_ID	N-Decimal(1,0)
PK	SPILL_ID	N-Decimal(10,0)
	SPILL_TIME	N-Decimal(10,0)
	SOBRECORD_FILEOFFSET	N-Decimal(10,0)
	SOBRECORD_SIZE	N-Decimal(10,0)
	SOBRECORD_OBJYOID	C-Variable Length(30)
	EOBRECORD_FILEOFFSET	N-Decimal(10,0)
	EOBRECORD_SIZE	N-Decimal(10,0)
	EOBRECORD_OBJYOID	C-Variable Length(30)
	ORACLE_INSERTIONTIME	T-Auto Timestamp
	ORACLE_INSERTIONHOST	N-Decimal(3,0)
	ORACLE_LASTUPDATETIME	T-Auto Timestamp
	ORACLE_LASTUPDATEHOST	N-Decimal(3,0)

RAW_EVENT_TYPE

PK	TYPE_ID	N-Decimal(1,0)
	TYPE_NAME	C-Variable Length(19)

RAW_EVENT

PK,FK1,U1,I1	RUN_ID	N-Decimal(5,0)
PK,FK1,U1	EVB_ID	N-Decimal(1,0)
PK,FK1	SPILL_ID	N-Decimal(10,0)
PK	EVENTINPARTIALSPILL_ID	N-Decimal(10,0)
U1	EVENT_TIME	N-Decimal(10,0)
FK2,I2	EVENT_TYPEID	N-Decimal(1,0)
I1	EVENT_ID	N-Decimal(10,0)
	EVENTINSPILL_ID	N-Decimal(10,0)
	DHA_ATTTYPE0	N-Decimal(10,0)
	DHA_ATTTYPE1	N-Decimal(10,0)
	DHA_DETMASK0	N-Decimal(10,0)
	DHA_DETMASK1	N-Decimal(10,0)
	DHA_DETMASK2	N-Decimal(10,0)
U1	DHA_TIMEMICROSEC	N-Decimal(10,0)
	EVENTRECORD_FILEOFFSET	N-Decimal(10,0)
	EVENTRECORD_SIZE	N-Decimal(10,0)
	EVENTRECORD_OBJYOID	C-Variable Length(30)
	ORACLE_INSERTIONTIME	T-Auto Timestamp
	ORACLE_INSERTIONHOST	N-Decimal(3,0)
	ORACLE_LASTUPDATETIME	T-Auto Timestamp
	ORACLE_LASTUPDATEHOST	N-Decimal(3,0)

Figure 3. Simplified Oracle relational data model for the HARP event collection metadata. Only the most important constraints are displayed. In the columns on the right, data types are shown as generic ODBC types rather than as Oracle types. The diagram also does not include any views and materialized views, or any of the auxiliary tables used for bookkeeping the progress of the migration or the physical partitioning of the above tables according to ranges of run numbers.

SUMMARY

The migration of the HARP data and software out of Objectivity has been successfully completed. The new Oracle-based hybrid solution has been used ever since in HARP for physics analysis. The success of the migration proves the viability of Oracle to handle large volumes of physics data. Oracle will also be used to store the samples of HARP DST data that are being produced.

The experience gained in this project may be useful in the future, as the need for data migrations is common to all projects that rely on the long-term availability of the data and may arise again at LHC.

ACKNOWLEDGEMENTS

We would like to thank our colleagues from the IT-DB, IT-DS and IT-ADC groups at CERN for contributing to the design and setup of the migration infrastructure. We also wish to thank Simone Giani for his feedback and support to the project and Ioannis Papadopoulos for invaluable discussions about the Objectivity-based HARP software.

REFERENCES

[1] HARP Collaboration, "Proposal to study hadron production for the neutrino factory and for the atmospheric neutrino flux", CERN-SPSC/99-35, November 1999.
[2] HARP Collaboration, "Status report of the HARP experiment", CERN-SPSC-2004-18, June 2004.
[3] M. Lübeck et al., "An Overview of a Large-Scale Data Migration", IEEE/NASA MSST 2003, San Diego, January 2003.
[4] M. Nowak et al., "Objectivity Data Migration", CHEP 2003, La Jolla, March 2003.
[5] ALICE DAQ, "DATE User's Guide", http://aldwww.cern.ch/
[6] J. D. Durand et al., "CASTOR operational issues and new developments", these proceedings.
[7] POOL - The LCG Persistency Framework, http://pool.cern.ch/
[8] A. Valassi et al., "LCG Conditions Database Project Overview", these proceedings.