CORAL

CORAL (**CO**mon **R**elational **A**bstraction **L**ayer) is a software toolking written by ?CERN' s ?LCG Application Area. It provides access to relational databases using a C++, SQL-free API. Currently it implements four different RDBMS flavors: SQLite, MySQL, Frontier and Oracle.

We are considering using CORAL as a foundation for DBMSStorage. It is expected CORAL in particular will help us:

- encapsulating variations between different RDBMS flavors (i.e. variations in SQL extensions, APIs, etc.) and
- managing multiple database connections.

It implements type conversion between rdbms and c++, variable binding, result-set prefetching, some bulk operations (LSST will need more), database service indirection, service failover, database monitoring and many other features we would otherwise have to implement ourselves.

Links to CORAL documentation:

- ?CORAL
- ?CORAL doxygen docs

CORAL is used by 3 out of 4 LHC experiments: Atlas, CMS and LHCb. It is the foundation for any database access from persistency frameworks of the above 3 experiments.

CORAL brings several dependencies: SEAL (package developed at CERN) and boost 1.33.1 plus the RDMBS library or libraries required. There is work in progress on removing the SEAL dependency, it is expected to be completed this summer (2007).

More notes about our experience with CORAL + LSST:

- Building CORAL
- Using CORAL
- Extending CORAL
- CORAL Overhead

Argument for removing CORAL

After we gained experience in the use of CORAL during DC2, we have realized several limitations that it brings:

- Building CORAL from source is very difficult. As a result, the standard methods used for LSST package distribution, which involve recompiling from source, could not be used.
- Binary distribution of CORAL limits platform support. It proved difficult to support CORAL on MacOS and Linux64.
- CORAL does not provide all the needed facilities. In particular, "LOAD DATA INFILE" and "CREATE TABLE ... LIKE ..." functionalities offered by MySQL (and similar functions in other RDBMS packages) were not accessible, although they could be added to CORAL. Adding these to our own version produces a fork that would have to be incorporated into the main trunk at a later date,

plus it brings in the building from source issue. Requesting that these be added to the supported version leads to uncertain delivery dates.
- Type conversion in CORAL was outside our control. This led to issues with supporting DECIMAL fields and CHAR(1) fields versus TINYINT fields.
- If our scalable parallel database architecture prototype is to be used with the rest of the LSST software, it must either present a CORAL-compatible interface (which might be difficult) or be used via its own API outside CORAL, negating much of the usefulness of CORAL.

Removing CORAL would lose some of the advantages mentioned above. There are mitigations for most of these, however:

- Porting from one RDBMS to another is simplified by encapsulating all RDBMS-specific code in the `DbStorageImpl` and `DbTsvStorage` classes. This reduces the amount of code that needs to be rewritten to support a new RDBMS substantially.
- The above classes allow for multiple simultaneous database connections, and they can be enhanced relatively easily to support connection pooling for databases that operate more efficiently that way.
- Variable binding is typically provided by the RDBMS API. A wrapper must be implemented to provide this functionality in a portable fashion, but it is of relatively low complexity and allows us to control type conversion.
- Result set prefetching will not be implemented, but the underlying RDBMS API may provide it.
- Database service indirection can be handled by an orchestration layer in combination with Policy-specified database server locations.
- Database failover can be handled, often more robustly, via IP takeover mechanisms at the network layer rather than at the application layer.
- Database monitoring is likely to be handled by an overall system monitoring mechanism; CORAL's functionality in this area may not be needed.