# Table of Contents

CORAL

- A ‘COmmon Relational Abstraction Layer’................................. 1
- CORAL and Database Deployment on the Grid........................ 1
- What CORAL is NOT................................................................... 1
CORAL

A 'COmmon Relational Abstraction Layer'

The primary goal of CORAL is to provide functionality for accessing data in relational databases using a C++
and SQL-free API, shielding the user from the technology-specific APIs and removing at the same time the
need to submit directly SQL commands. Therefore CORAL allows the development of software components
that can be used without any code modification or conditional constructs against multiple relational
technologies.

A user is not expected to be an expert on all possible optimization techniques relevant to a particular relational
technology. This is handled as much as possible by the RDBMS-specific implementations of the CORAL
interfaces. In fact, a user is expected to be familiar only with the basic concepts of relational database systems,
while the only SQL fragments that would ever have to be provided are the condition clauses in queries. On the
other hand, the CORAL interfaces have been designed such that a user is guided towards standard "best"
practices in RDBMS programming, such as variable binding, result-set prefetching, bulk operations, etc.

The CORAL API is limiting the set of generated SQL statements to a relational server to a rather small subset
of whatever would be possible through an interface for issuing arbitrary SQL statements. Thanks to this
feature, applications which are based on CORAL are easier to monitor and tune.

CORAL and Database Deployment on the Grid

CORAL is expected to be used by applications running on a grid-enabled and distributed environment. In
such environment there is a need to

- allow for physical database service indirection,
- separate the database connectivity from the authentication mechanisms,
- enable certificate-based authentication mechanisms,
- enable client-side connection pooling and service failover,
- enable client-side monitoring.

The architecture and design of CORAL provides the relevant infrastructure and mechanisms addressing these
issues.

What CORAL is NOT

- A general purpose C++ connectivity library, such as ODBC, JDBC, the python or perl DBI.
- A system to accomodate all possible use cases for accessing relational data. It is restricted to the use
cases relevant to the data handling and analysis of the LHC experiments.
- A system to perform C++ object I/O with an RDBMS.

-- AlexanderKalkhof - 21-Jan-2010

Child topics (in alphabetical order): CoralConnectionStrings CoralConnectionSvcConfig CoralExpressionParser CoralFaultTolerance CoralLFCReplicaSvc CoralMonitoring CoralOracleAccess CoralReadOnlyModes CoralServer CoralSqlToCppMapping

This topic: Persistency > Coral
Topic revision: r4 - 2010-08-18 - AndreaValassi
What CORAL is NOT