Overview of C/C++ DB APIs

Dirk Düllmann, IT-ADC

Database Workshop for LHC developers

27 January, 2005

Oracle Language Bindings

Goal: Access to database with C/C++ as programming language

- Database functionality
  - Connection handling
  - Transaction handling
  - Session configuration
  - Statement execution
- Database data
  - Several levels of presenting the database data
  - Relational Model
    - API gives access to tables and rows objects
  - Object Model
    - API gives access to C++ objects including their dynamic type, methods,

Oracle Call Interface - OCI

- The base of many higher level tools
  - Used extensively by Oracle and third party tools
    - Stable API
    - Stable C ABI
  - Used to implement POOL RAL
- Low level of abstraction
  - Many knobs for optimisation
  - Many calls, many arguments
- Very complete
  - Anything Oracle can do can be done with OCI
- Significant learning effort to get efficient
- Good tool for expert developers who need to focus
    - on the last bit of performance
    - on database internals
- Does it pay off for your project?
- Do you envisage to use any other database vendor?

Precompilers - Pro*C

- Provided a convenient way of translating enhanced “C”..
  - to a program calling a lower level Oracle functions
  - hiding some of the complexity
- Works well for static SQL
  - Stable data model
  - Not much context which influences the SQL statements
  - Not a good option for SQL which needs to be created dynamically
- Source code is not standard “C”
- Portability ?
- My personal impression
  - With C++ one can hide low level complexity in a more standard way
    - without language extensions
  - Keeping the SQL generation still extensible by the user

OCCI

- A better OCI for C++
- Higher abstraction level
  - Classes and objects instead of just C functions
- C++ library bound to a particular C++ compiler version
  - Problems on Linux with rapid / non-backward compatible compiler evolution
- Can be used in two “modes”
  - Providing access to relational concepts (Table /Row)
    - This mode has been used in the “old” ConditionsDB implementation and by
  - Mapping table data to C++ objects
    - C++ header and implementation files generated from SQL object definition (Oracle tool ott)
    - Intrusive into physics code and after initial evaluation largely abandoned by experiments

POOL RAL and Object Storage

- Language bindings developed by the LCG Persistency framework project
  - Database vendor neutral (Oracle, MySQL, SQLight)
  - Component based (extensible to new back-ends via plugins)
- POOL Relational Abstraction Layer
  - High level C++ API
  - Relational Level
    - Access to data in tables, rows
  - Base for POOL component and Conditions Database implementation
- POOL Object Storage
  - Standard POOL C++ interface
    - As for object streaming to ROOT files
    - Simplifies moving data between file and RDBMS storage
  - Access to data on object level
    - Stores and retrieves transient C++ objects
    - Object mapping to tables done automatically