FairDB SQL Persistency Scheme

Denis Bertini[1]
[1]Scientific Computing ,GSI, Darmstadt, Germany

Introduction

Conceptually FairDB [1] handles data as table i.e in a two-dimensional data structure with cells organized in rows and columns.
This is the reponsability of the FairDB framework to deliver an automatic parameter object to relational table mapping and a uniform API to access these object regardless of the internal relational representation.

Object to Table Mapping

FairDB eases the transition parameter object to the relational data model by implementing an automatic mapping procedure according to the following rules:

- **Class** definition corresponds to table definition
- **Columns** are the physical equivalent of the attributes
- **Rows** are the physical equivalent of objects instances

Additionnaly each record (rows or unit block of rows) on a table is uniquely identified. The purpose of the unique identifier is to act as the *primary key* on the table where it is defined and to be referenced as the *foreign key* by other related tables.
Figure [1] shows an example of object to table mapping in the case where the target parameter class is a simple class or is related through inheritance.

Figure 1: One-to-One Parameter object to relational table mapping.

SQL I/O scheme

FairDB provides a programmatic and uniform template based API to write and access parameter object. Like the Java SQL interface system (JDBC)[2], each data retrieval produces a pointer giving read access to a results table which corresponds to a subset of the underlying database table (Figure [2]). Each Row of the results table corresponds to a object, the type of which is user defined and table-specific.

Minimizing I/O with Level2 Cache

Another important point is to minimise I/O. Some requests, particularly for detector relevant parameters, can pull in large amounts of data but users must not load it once at the start of the job and then use it repeatedly since it may not be valid for all the data they process. Also multiple users may want access to the same data and it would be inefficient for each to have their own copy.
To deal with both of the above problems, the interface uses the concept of handle or proxy . When accessing a particular table, a table-specific pointer object is created. the corresponding object is usually very small is suitable to be stack based and passed by value, thus reducing considerably the risk of a memory leak.
During construction of the pointer, a request for data is passed down through the interface and the results table, which could be large, is created on the heap. The interface places the table in its cache and the user's pointer is attached to the table, but the table is owned by the interface, not the user.

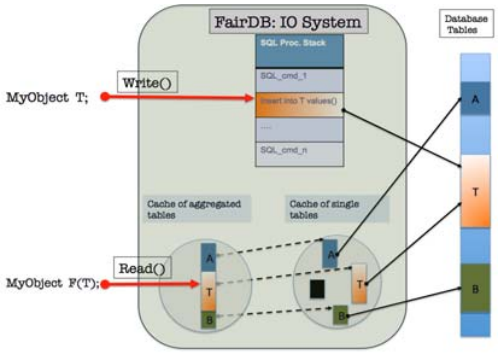

Figure 2: FairDB SQL I/O persistency scheme. Different level of Caches are used to store recently accessed table in on disk.

References

[1] FairRoot Virtual Database (User Manual). `https://panda-wiki.gsi.de/foswiki/pub/Computing/PandaRoot/FairRootVirtualDatabase.pdf`
[2] http://docs.oracle.com/javase/tutorial/jdbc/

 doi:10.15120/GR-2014-1-IT-12