Template Metaprogramming an Object Interface to Relational Tables

Giuseppe Attardi, Antonio Cisternino

Dipartimento di Informatica, corso Italia 40, I-56125 Pisa, Italy
{attardi,cisterni}@di.unipi.it

Abstract. We present a general technique to support reflection in C++, exploiting template metaprogramming techniques. The technique is used for building an object interface to relational database tables. By just annotating a class definition with meta information, such as storage attributes or index properties of fields, a programmer can define objects that can be stored, fetched or searched in a database table. A high-performance, full text search engine has been built with this technique.

Introduction[1]

An object oriented interface library to a relational database table must be capable of storing objects of any class into its rows. The library must therefore know the structure of the objects in order to perform serialization. However table schema definition and table usage are independent operations, of which the compiler is unaware. Hence data operations require detailed instructions for reconstructing objects fetched form a table or supplying detailed information about their class. This can be avoided if the library can exploit introspection [1] for determining the attributes of a class and their types, and use intercession [1] to modify the objects. Such solution is more efficient and convenient than traditional embedded database languages and relieves programmers from much burden. A full object-oriented database can be built with limited effort on top of this interface and in fact we used it for implementing IXE, a fully featured, high performance class library for creating customized, full-text search engines.

The needed reflection facilities have been achieved in C++ by exploiting template metaprogramming techniques, without extending the language or the compiler, as in other proposals [3].

We provide both *static reflection*, where metaclass information is only used at compile time to produce class specific code; and *dynamic reflection*, where metaclass objects exist at runtime. Static reflection involves no runtime computations, while dynamic reflection allows defining classes dynamically from a metaclass assembled from field descriptions and other information. Dynamic reflection is necessary for instance in an interactive SQL interpreter.

[1] This research has been supported in part by a grant from Ideare SpA.

The Object Interface to Relational Tables

Relational DBMS use tables for storing relations, expressed in SQL-like statements like this:

```
create table Documents (
    name          varchar(2048),
    body          varchar(65536),
    size          INT,
    PRIMARY KEY(name), FULLTEXT(body)
)
```

We represent such table by a C++ class, supplying meta-information, in particular storage attributes or index properties, about each attribute as follows:

```
class Document {
  public:
    char*         name;
    Text<65536>   body;
    int           size;
  META(Document,
    (VARKEY(name, 2048, Field::unique),
     KEY(body, Field::fulltext),
     FIELD(size))
    );
};
```

`META`, `KEY`, `VARKEY` and `VARFIELD` are macros that exploits template metaprogramming for creating a metaclass for the class. The template class `Table` implements a relational table for storing objects of a specified class. Here is how to create such table and insert into it an object `doc` of class `Document`:

```
Table<DocInfo> table("db/table");
table.insert(doc);
```

The table can be queried obtaining a cursor for accessing the results of the query, similarly to GigaBase [2]. For example:

```
Query query("size < " + size + " and text matches 'PDF'");
QueryCursor<DocInfo> cursor(collection, query);
while (cursor.hasNext())
    dt = cursor.get()->title;
```

Method `get()` returns a genuine object, whose methods can be invoked directly.

References

1. K. Czarnecki, U.W. Eisenacker, *Generative Programming – Methods, Tools, and Applications*. Addison Wesley, Reading, MA, 2000.
2. K.A. Knizhnik, *The GigaBASE Object-Relational database system*, http://www.ispras.ru/~knizhnik.
3. S. Chiba. *A metaobject protocol for C++*. Conference Proceedings of Object-Oriented Programming Systems, Languages and Applications, pp. 285-299, ACM Press, 1995.