A Database Independent JDO Persistence Framework with User-defined Mapping Specification Languages

Liu Yang
Monash University, Malaysia
yllu8@student.monash.edu

Lim Tong Ming
Monash University, Malaysia.
lim.tong.ming@infotech.monash.edu.my

Abstract - Object-Oriented Programming Languages (OOPL) and object-oriented programming environments (OOPE) are becoming the chosen platform used by today's developers. Nevertheless, object-oriented programming languages such as C++ and Java do not provide a suitable interface that allows objects in an application to be saved into a persistence mechanism seamlessly. This research provides a middleware solution by designing and implementing a robust LYJDO Persistence Framework (LYJDO PFW) in order to promote the migration of conventional database technologies to object-oriented database technology. This persistence framework designed by implementing Java Data Object (JDO) concepts which is proposed by Sun Microsystems for transparently persisting persistent objects. And in order to achieve the flexible mapping between the persistent classes and underlying legacy databases capability, a complete Mapping Specification Language (MSL) is designed and implemented in the proposed persistence framework. As a result, developers could define their own mapping strategies by using the proposed persistence framework. To sum up, the proposed persistence framework is an OOPL-independent, database-independent JDO implementation with user-defined mapping mechanism.

Keywords: Object-Oriented Programming Language, Persistence Framework, Java Data Object, Mapping Specification Language.

1 Introduction

Generally, most enterprise applications generally manage large amount persistent data and a set of complex business rules that operate on these data. The data needs to be persistent because it needs to be around for multiple runs of the enterprise application which means for the entire lifetime of the particular business corporation. Over the lifetime of the company there may be multiple changes to the business rules and processes. The persistent data stays the same even if there is a major shift in the company - like installing a new system, the persistent data will have to be migrated to the new system. For these purposes, we need some storage systems to maintain and manipulate these persistent data. There are currently many storage mechanisms in used, such as Rational Database Management System (RDBMS), Object-Oriented Database Management System (OODBMS), Multidimensional Database Management System (MDBMS) and so on.

Today, the most popular storage mechanism used in most of the business application development is Relational Database Management Systems (RDBMS). It has been accepted in many medium-to-large organizations because it allows data to be managed and manipulated effectively and efficiently. The popularity of Object-Oriented Programming Language (OOPL) in the development of business applications has tremendously increased last a few years, but data were still managed in large legacy database like RDBMS which do not support object-oriented (OO) characteristics such as inheriting, collection, complex object etc. Hence the developers must embed SQL statements in their applications in order to map the persistent classes to tables in relational database or other persistent stores so that classes can be managed persistently. If the developers want to make a new application with different object-oriented programming languages and databases, they have to rewrite all the SQL statements within the application in order to manage objects persistently in the RDBMS. Therefore, we need a flexible and extensible framework providing OOPL-independent and database-independent mapping mechanism in order to store, retrieve and manipulate the persistent objects between the persistent classes and data structures of underlying legacy database.

For this purpose, a persistence framework must be taken up in the first place. Generally, persistence framework is a general-purpose, reusable, and extendable set of types that provides functionality to support persistent objects. It could translate objects into records (for some other form of structured data such as XML) and save them in a database, and translate records into objects when retrieving from persistent storage. In this research, Java Data Object (JDO) will be implemented as a mechanism to develop the persistence framework in order to perform transparent persistence. In next section, overview of RDBMS and its weakness will be presented. This is followed by the motivation of this research, scope of the research, research objective and methodology.