ISSN (Online) : 2454-4159
International Journal of Advanced Research in Science Management and Technology
Volume 2, Issue 6, June 2016

A Review on - Generating Constant Data in Real Time Environment to Save Transactions Using Xml as Static

Abhijit Ahirrao[1], Prof. Kailash Patidar[2] , Prof. Manoj Yadav[3]

PG Student, Dept. of CSE, SSSUTMS, Sehore, M.P., India[1]
Professor & Head, Dept. of CSE, SSSUTMS, Sehore, M.P., India[2]
Assistant Professor, Dept. of CSE, SSSUTMS, Sehore, M.P., India[3]

ABSTRACT— "Where data (state) outlives the process that created it". Persistence is one of the fundamental concepts in application development. In programming, persistence refers specifically to the ability to retain data structures between program executions, such as, for example, an image editing program or a word processor saving their documents to avoid data loss in the event of power failures.

Object Persistency is an Open source Object-relational mapping framework for Java. Versions 3.2 and later provide an implementation for the Java Persistence API. Driven by XML a configuration file to configure data connectivity and map classes to database tables and DAO is thereby used to hide complex ORM.

- Java/SQL code generation tools.
- Developer writes code to call API.
- API executes necessary SQL at runtime.

Object Persistency Eliminate need for repetitive SQL. It Work with classes and objects instead of queries and result sets. Handles all create-read-update-delete (CRUD) operations using simple API; no SQL Generates DDL scripts to create DB schema (tables, constraints, sequences). Supports over 20 RDBMS; change the database by tweaking configuration files.

KEYWORDS- Object Persistency, Java Persistence API, SQL, CRUD (create-read-update- delete)

I. INTRODUCTION

Object Persistency is an Open source Object-relational mapping framework for Java. Versions 3.2 and later provide an implementation for the Java Persistence API. Driven by XML configuration files to configure data connectivity and map. Classes to database tables. Not a Java/SQL code generation tool. Developer writes code to call API. API executes necessary SQL at runtime.

Key Features:-

1) Natural programming model – Object Persistency supports natural OO idiom; inheritance, polymorphism, composition and the Java collections framework.

2) Support for fine-grained object models – A rich variety of mappings for collections and dependent objects.

3) No build-time byte code enhancement – there's no extra code generation or byte code processing steps in your build procedure.

4) Extreme scalability – Object Persistency is extremely perform ant, has dual-layer cache architecture, and may be used in a cluster.

ISSN (Online) : 2454-4159
International Journal of Advanced Research in Science Management and Technology
Volume 2, Issue 6, June 2016

5) The query options – Object Persistency addresses both sides of the problem; not only how to get objects into the database, but also how to get them out again. Support for "conversations" – Object Persistency supports both long-lived persistence contexts, detach/reattach of objects, and takes care of optimistic locking automatically.

6) Free/open source – Object Persistency is licensed under the LGPL (Lesser GNU Public License).

7) EJB 3.0 – Object Persistency implements the Java Persistence management API and object/relational mapping options, two members of the Object Persistency team are active in the expert group.

Why Use Object Persistency?

- Eliminate need for repetitive SQL.
- Work with classes and objects instead of queries and result sets.
- More OO, less procedural.
- Mapping approach can resist changes in object/data model more easily.
- Strong support for caching.
- Handles all create-read-update-delete (CRUD) operations using simple API; no SQL Generates DDL scripts to create DB schema(tables, constraints, sequences).
- Flexibility to hand-tune SQL and call stored procedures to optimize performance.
- Supports over 20 RDBMS; change the database by tweaking configuration files.

In programming, persistence refers specifically to the ability to retain data structures between program executions, such as, for example, an image editing program or a word processor saving their documents to avoid data loss in the event of power failures.

Almost all applications require persistent data. If an information system didn't preserve data when it was powered off, the system would be of little practical use. When we talk about persistence in Java, we're normally talking about storing data in a relational database using SQL. We'll start by taking a brief look at the technology and how we use it with Java. Armed with that information, we'll then continue our discussion of persistence and how it's implemented in object-oriented applications.

II. LITERATURE SURVEY

In last few decades there was rapid shift observed while writing software solutions. Object oriented model is now globally adapted and preferred by most of the application developers. Object persistence plays a key role in designing data model, business objects working with other business objects. Object persistence could be very efficient if designed correctly. A typical design leads extra overheads in terms of cost, resource utilization, and time. Considering the importance of object persistence, it is very essential to concentrate more on this area. Gateway-based, Object-relational database and Object-oriented Database are the three major groups of solutions available to implement object persistence.

In this paper, we discuss about the features available in Object Persistence methodologies, how and where we should use them efficiently based on the application requirements. Our discussion continues further on positive and negative sides of object persistence methodologies by considering limitations and different application requirements.

When an object is being created by an application, the scope of object is limited to the application life cycle. With the end of application, object life line also ends. The reason is, object is being stored temporarily in main memory. To keep object alive, application need to store the object in persistence storage.

Object persistence refers to the concept of saving the state of an object so that it can be restored and used at a later time. An object that does not persist basically dies when it goes out of scope. Maintaining the state of an object is called persistence. There are many ways to implement persistence in applications. A simple example is to use text file as storage. All required information can be stored on file by using file write operation. This information can be retrieved when restore is required by performing file read operation. This solution is not efficient and might be good for small applications, where limited information saving is required and information is not changing frequently. Representing information on simple text file becomes very complicated, where information to be stored is very complex in nature. Lot of development efforts will be wasted to maintain this. However text files are very flexible, easy to implement, can be accessed by more than one program, but they are not object friendly. Object oriented programs offer variety of relationship with other objects, for example inheritance and references with other objects. The challenge here is how we can represent and maintain the different kind of relationships on text file.

Another solution is to use relational database as storage, but due to limitations of databases, it is not a best solution. This can be a good choice when application need database related functionalities like, transaction with rollback, record locking and indexing. Databases are generally expensive; managing them is even more difficult. Object oriented instances are basically structured in hierarchical and relational database structured in tabular format. These are two different structures and a difference in two approaches is known as “impedance mismatch” problem.

Solution towards achieving persistence in object-oriented applications categories in three classes: the gateway-based method adds object-oriented programming access to persistent data stored using traditional non object-oriented data stores, the object-relational DBMS method enhanced the extremely popular relational data model by including object-oriented modelling features, and the object-oriented DBMS method adds persistence support to objects in an object-oriented programming language.

[1] Clarence J M Tauro, N Ganesan, Ritesh Kumar Sahai, Sandhya Rani A “Comparative Study on Object Persistence Methods” in International Journal of Computer Applications (0975 – 8887) Volume 42– No.7, March 2012.

Object-Oriented development is bottom-up approach, which focus on data instead of procedures that manipulates the data. Object-Orientation is an efficient, powerful, reliable and well-known technique that mimics a network of real life processes, scalable applications, tasks, and business rules of a domain. An object present in a software application consumes some amount of memory space and exists for a specific time. When an object is being created by an application, the scope of object is limited to the application life cycle. When the application terminates, object’s life also terminates. The reason is, object is being stored temporarily in main memory. To keep object alive, application need to store the object in persistence storage.

The concept of maintaining the state of an object is termed as Object-Persistence. Object-Persistence refers to the concept of saving the state of an object so that it can be restored and used at a later time. An object that does not persist basically dies when it goes out of scope. A persistent object continues to exist, even when the application that created or used the object has finished execution.

In an application, persistence can be implemented in many ways, and few of them are listed below:

- Storing object in a simple text file.

- Storing object in an indexed sequence access mechanism.
- Storing object in relational database.
- Storing in an Object-Oriented database.

A very simple way is to use text file as storage. All required information can be stored on file by doing file write operation. This stored information can be retrieved when restore is required by performing file read operation. This kind of persistence mechanism might be good for small applications, where limited information saving is required and information does not change frequently. But, representing complex information in a text file is very complicated. Lot of development efforts will be wasted in maintaining this. However text files are very flexible, easy to implement, can be accessed by more than one program, but they are not object friendly. Object-Oriented programs exhibit various kinds of relationships with other objects, for example inheritance and references with other objects. The challenge here is how we can represent and maintain the different kinds of relationships on text file while saving the object.

Object-Persistence can also be implemented by another mechanism in which relational databases are used for object storage. This approach is most appropriate for storing business data as it can be easily formatted into rows and columns. This mechanism can be a good choice when application needs database related functionalities like, transaction with rollback, record locking and indexing. Databases are generally expensive; managing them is even more difficult. Object-Oriented instances are basically structured in hierarchal order and relational database structured in tabular format. These are two different structures and a difference in two structures is known as "impedance mismatch" problem. Due to these limitations of databases, this mechanism is not a best solution.

Implementation mechanism for Object-Persistence can be broadly classified into three categories. First one is the gateway-based approach which adds Object-Oriented programming access to persistent data stored using traditional non Object-Oriented data stores. The second mechanism is the Object-Relational DBMS approach which enhanced the extremely popular relational data model by adding Object-Oriented modelling features and the final mechanism is the Object-Oriented DBMS approach which adds persistence support to objects in an Object-Oriented programming language. OODBMS approach is more appropriate for unformatted and/or scientific information.

[2] Clarence J M Tauro, Ritesh Kumar Sahai, Sandhya Rani A. " Object Persistence Techniques - A Study of Approaches, Benefits, Limits and Challenges" in International Journal of Computer Applications (0975 – 8887) Sept. 2014.

III. PROBLEM DEFINITION

The popularity of databases has increased manifold in the past few years. Java has become the preferred choice of developers for developing secure, flexible, and scalable database driven web applications. These web applications require objects to be associated with appropriate databases. Hibernate, along with other persistence technologies associate's objects with the appropriate database in a simple, straight forward and natural way.

The concept of maintaining the state of an object is termed as Object-Persistence. Object-Persistence refers to the concept of saving the state of an object so that it can be restored and used at a later time. An object that does not persist basically dies when it goes out of scope. A persistent object continues to exist, even when the application that created or used the object has finished execution.

In an application, persistence can be implemented in many ways, and few of them are listed below:

- Storing object in a simple text file
- Storing object in an indexed sequence access mechanism
- Storing object in relational database
- Storing in an Object-Oriented database

ISSN (Online) : 2454-4159
International Journal of Advanced Research in Science Management and Technology
Volume 2, Issue 6, June 2016

IV. PROPOSED SOLUTION

In this paper we proposed the efficient system which is feasible for generating the constant data in real time environment which reduces the number of transactions. The existing system faces the problems:

1. Unified Development Process:

Close language binding offers scalability and design flexibility that simplifies the lives of system analysts and designers. The good point of Object-Oriented DBMS is that it supports object oriented language and applies the object-orientation concept in database as well. Object-Oriented DBMS follows semantically same concept like Object-Oriented modeling and design, Object-Oriented analysis, and Object-Oriented languages allowing a unified conceptual approach during the whole development cycle.

2. Transparency and Object-Fault:

Persistent of object should be independent of how application program modified and manipulates that object. Persistence independence, also referred as transparency, requires that it is indistinguishable whether programming code is operating on persistent or transient data.

3. Data Source Related Issues:

JDBC provides standard for communication between Java based application and relational database. Several frameworks have been designed with the focus on independence from data store type, however still JDBC-compliant data stores captured the popularity and most of attention and efforts. JDBC itself is quite convenient and simple set of interfaces, the actual implementations tend to deviate from the standard by providing proprietary extensions in some cases or postponing certain important implementations like metadata until better times.

4. Concurrency Control:

In recent development, concurrency control becomes basic need for scalable applications, which allows multiple users to access the data storage at the same time. Effective and efficient concurrency control mechanism enables guaranteed data integrity, and consistent information received by users.

5. Machine independence:

During computation operation files are being created in different systems and on different type of machines. These files are being used by other systems with different type of machines. This requires continuous tracking of differences in size and byte order of integer, and floating-point representation.

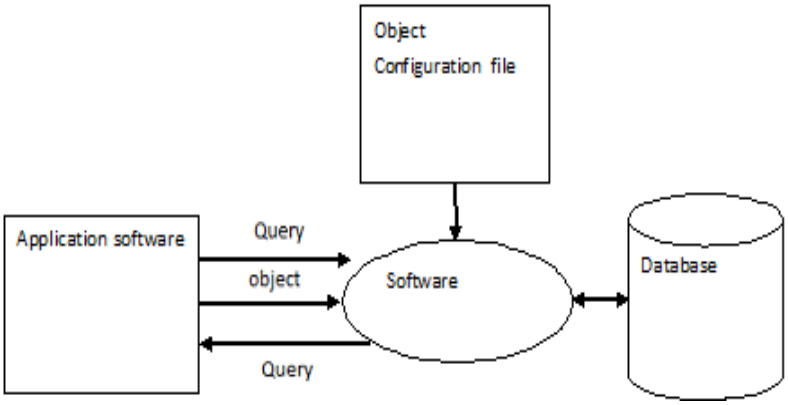

Fig. 1 Basic Working DFD

V. EXPECTED RESULTS

As per the expectations the project is developed to overcome the problem of object inconsistency, the main aim of the topic is to reduce the transaction in database using the Object Oriented DBMS and the Object Relational DBMS. The idioms of the Object Oriented Database i.e. inheritance, polymorphism, composition and the Java collections framework. The support is provided for applications by the various DBMSs for sharing data between concurrent users, crash recovery, advanced transaction models (long transactions, versioning, nested transactions), and distributed access to data.

ACID transactions:

OODBMSs support the conventional type of short transactions termed ACID transactions. OODBMSs do support various types of locking. The standard lock types are page locks and object locks (also known as record locks in RDBMSs). GOP System provide limited support for ACID (atomicity, consistency, isolation, and durability) transactions since the object cache maintained at the application is loosely coupled to the DBMS. ORDBMSs support all the traditional lock types available in relational DBMS (tuple, page, and table locks).

VI. CONCLUSION

The Object Persistency provides Java SE and Java EE developers with a single, standardized mechanism for database Persistence. Object Persistency is not tightly tied with any underlying database. “Write once persists anywhere" using Object Persistency can achieved by changing the dialect in configuration xml file. Object Persistency is used to persist the objects in the database.

Object persistency depends on object serialization capabilities of the underlying programming language. For Object Persistency provide a robust and reliable foundation for effective management of object. Creating a separate object persistence layer using ORM is great way of improving developer productivity and application quality and specifying maintenance. You might never want to write another line of SQL again.

REFERENCES

[1] **Clarence J M Tauro, N Ganesan, Ritesh Kumar Sahai, Sandhya Rani A “Comparative Study on Object Persistence Methods” in International Journal of Computer Applications (0975 – 8887) Volume 42– No.7, March 2012.**

[2] **Clarence J M Tauro, Ritesh Kumar Sahai, Sandhya Rani A. “ Object Persistence Techniques - A Study of Approaches, Benefits, Limits and Challenges” in International Journal of Computer Applications (0975 – 8887) Sept. 2014.**

[3] **K.R. Dittrich, "Object-Oriented Database System : The Notions and the issues, in” : Dittrich, K.R. and Dayal, U. (eds): Proceedings of the 1986 International Workshop on Object-Oriented Database Systems http://aldex.co.uk/respec.html.**

[4] **Edwards, Mike. "SDO and Java Persistence Architecture (JPA)". Open SOA. osoa.org. Retrieved 5 May 2011.**

[5] **"Hibernate.org - Java Persistence with Hibernate". JBoss. 2009**

[6] **"Hibernate implements the Java Persistence object/relational mapping and persistence management interfaces"**

[7] **"Manning: Java Persistence with Hibernate". Manning. "Gavin King -- the founder of the Hibernate project"**

[8] **"JBoss.com - Industry Leadership". JBoss. Retrieved 2008-11-17. "JSR 220, EJB 3.0 Spec Committee, Gavin King, Bill Burke, Marc Fleury" http://www.javaworld.com/javaworld/jw-05-persistence.html.**