구조화질의언어 기반 퍼시스턴스 프레임워크

조동일*, 류성열**
*숭실대학교정보과학대학원 소프트웨어공학과
**숭실대학교 IT 대학 컴퓨터학과 교수
e-mail : chodongil@ssu.ac.kr

SQL Based Persistence Framework

Dong-il Cho*, Sung-Yul Rhew**
*Dept. of Computer Science, Soongsil University Graduate School of Information Science
**Professor, Dept. of Computer Science, Sungsil University

요 약

웹기반 기업형 어플리케이션은 객체지향 언어로 개발되고, 데이터의 관리는 RDB(Relational Database)를 이용하여 구축된다. 두 시스템은 이질적 패러다임에 기인하여 모델의 불일치성(object-relational impedance mismatch)을 발생시킨다. 이 문제를 해결하고자 사용되는 객체-관계 매핑 프레임워크(ORM-Framework)는 RDB 의 테이블과 객체지향 언어의 객체를 매핑하는 구조로 복잡한 메타정보를 이용하여 동적으로 매핑하기 때문에 개발이 복잡하고, 변경에 유연하지 못하여 유지보수에 많은 어려움이 있다. 본 논문에서는 기존 ORM 프레임웍크의 복잡성을 해소하고, 변경에 유연한 퍼시스턴스 프레임워크를 제안한다. 제안되는 프레임워크는 SQL 을 래핑하는 구조로 테이블과 객체의 메타정보가 불필요하고, 정형화된 구조를 가진 래퍼의 사용으로 소스코드를 자동 생성하여 개발 및 유지보수의 편의성을 제공하고, 변경에 유연하다. 제안 프레임워크는 Hibernate, iBATIS 와의 테스트 결과 구동 매커니즘이 거의 동일한 iBATIS 와는 처리속도가 비슷했고, Hibernate 의 약 3 배 빠른 속도를 보였다. 코딩량은 Hibernate 대비 1/9, iBATIS 대비 1/4 을 나타냈다.

ORM;Framework;ActiveRecord;Metaprogramming

1. 서론

기업형 시스템에서의 응용프로그램과 RDB 간의 연동은 대용량 데이터를 다루는 시스템 구축 프로젝트의 아키텍처 설계 시 중요한 이슈를 차지한다. 객체지향 언어로 개발된 응용프로그램과 RDB 는 이질적인 시스템 패러다임과, 네트워크로 분리된 분산환경 그리고 각 솔루션 마다 서로 다른 인터페이스의 영향으로 모델의 불일치성(object-relational impedance mismatch)을 비롯한 많은 문제점이 발생한다[6].

이런 문제를 능동적으로 해소하기 위해 많은 퍼시스턴스 프레임워크가 개발되고 배포되고 있으며, 아래와 같은 장점을 가진다[6].

- ✓ Structural mapping more roust
- ✓ Less error-prone code
- ✓ Optimized performance all the time
- ✓ Vendor independence

퍼시스턴스 프레임워크은 ORM(Object-Relational Mapping)이라는 방법을 사용하여 객체지향 응용프로그램과 RDB 를 통합하는 아키텍처를 제공한다. ORM 은 마치 객체의 정보를 변경하는 것처럼 RDB 의 데이터를 응용프로그램에서 변경할 수 있게 하는 인터페이스를 제공한다. ORM 프레임워크는 안정적이고 직관적인 인터페이스를 제공하고 있지만, 그런 인터페이스를 제공하기 위해 많고 복잡한 메타데이터를 필요로 한다. 메타데이터는 복잡한 구조로 연관관계를 가지고 있어 개발이 복잡하고, 네이밍 기반 매핑으로 인하여 오류를 유발할 수 있는 여지를 제공한다. 또한 매핑이 동적으로 일어나기 때문에 성능 저하의 원인이 되고, SQL 을 운영시점에 자동으로 생성하여 DBMS 밴더에서 지원하는 쿼리를 이용한 튜닝요소를 사용할 수 없다.

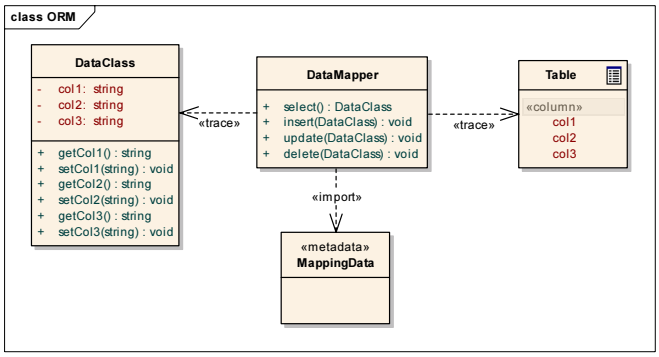
본 논문에서는 객체지향 응용프로그램과 RDB 의 통합을 SQL 을 래핑하는 구조의 퍼시스턴스 프레임워크를 제안 한다. 이 구조는 데이터의 매핑에 필요한 복잡한 메타데이타가 필요 하지 않다. SQL 래퍼는 템플릿 형태로 SQL 만으로 자동으로 생성하여 ORM 과 같은 인터페이스를 제공하면서도 변경에 유연하다. 래핑은 운영시에 발생하지 않고 생성된 소스코드에 의해 내장되어 운영시 매핑으로 발생할 수 있는 성능 저하가 없어 보다 빠른 처리성능을 발휘할 수 있다.

본 논문의 구성은 다음과 같다. 2 장 관련연구에서는 대표적인 ORM 프레임워크인 Hibernate 와 Apache 의 iBATIS 의 시스템 아키텍처를 알아보고, Metaprogramming 과 Object Wrapping 관련 연구를 알아본다. 3 장 제안 모델에서는 제안하는 퍼시스턴스 프레임워크의 시스템 아키텍처를 제시한다. 4 장 비교분석에서는 Hibernate, iBATIS 와 제안 프레임워크를 테스트를 거쳐 비교하고 비교 결과를 분석한다. 마지막

으로 5 장에서는 결론 및 향후 연구 과제를 제시한다.

2. 관련연구

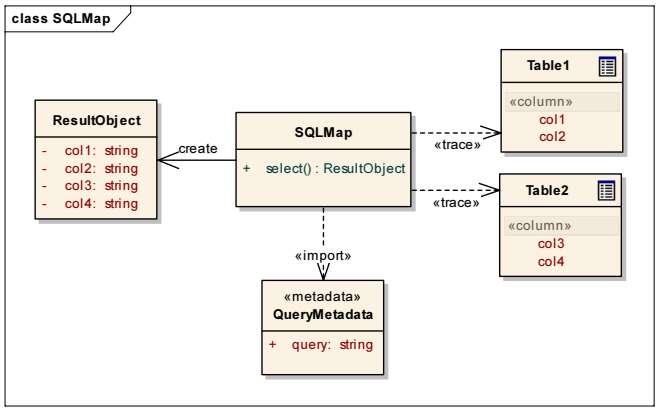
객체지향 언어는 모델의 불일치성을 해소하기 위해 몇 가지 디자인 패턴을 제시하고 있다[3]. 이들 디자인 패턴을 적용한 퍼시스턴스 프레임워크는 데이터의 관리를 중요시하는 기업시스템에서 주요하게 쓰여지고 있으며 Hibernate, Apache 의 iBATIS, Oracle 의 Toplink 가 주로 사용되고 있다.



(그림 1) Object Relational Mapping Architecture[3]

Hibernate 는 메타정보를 통해 RDB 의 테이블을 오브젝트와 매핑한다[1]. 마치 객체의 변수를 수정하는 것처럼 RDB 의 정보를 생성, 조회, 수정, 삭제 할수 있는 기능을 제공한다. HQL(Hibernate Query Language)이라는 RDB 독립적인 질의 언어를 지원하여 RDB 종류에 상관없이 프레임워크에서 적당한 SQL 로 자동 변환 한다[1]. 하지만 이런 기능들을 제공하기 위해 많은 메타정보를 필요로 하며, 이 메타정보들은 네이밍 기반으로 복잡하게 연결되어 있어 작성하기 힘들고, 변경 발생시 수정이 어려우며, 에러를 발생시킬 확률이 높다. 그리고 테이블과 매핑되는 클래스가 존재해야 하고 클래스 내부에 테이블의 관계와 유사한 클래스간의 관계가 표현되어야 하기 때문에 시스템의 변화가 잦은 기업시스템의 변화에 반영이 힘들다.

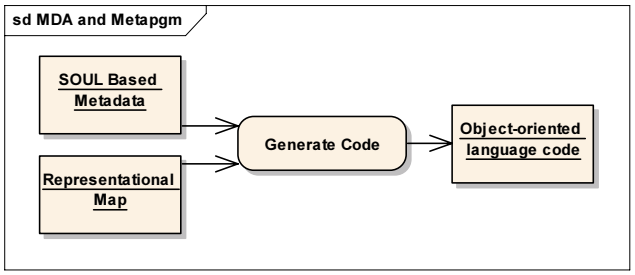
iBATIS 는 메타정보를 이용한 클래스와 SQL 을 매핑하는 SQL Mapping 기법을 사용한다.



(그림 2) SQL Mapping Architecture[2]

iBATIS 는 Hibernate 와 같이 강력한 기능은 지원하지 못하는 반면, 간단한 구조를 가지고, 빠른 처리성능을 가진다는 장점이 있다. 하지만 기능이 협소하고, SQL 이 많아 질수록 그에 대응하는 클래스도 함께 개발되어야 하며, 쿼리 및 데이터 클래스의 재사용이 힘들다[2].

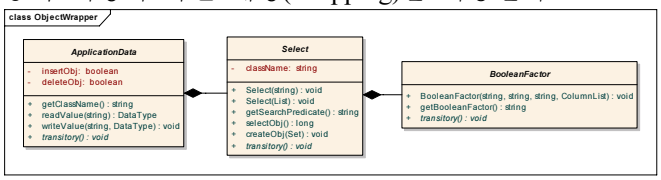
퍼시스턴스프레임워크의 메타정보는 개발툴이 정의될 경우 소스파일의 자동생성이 가능하다. Hibernate, Oracle toplink 의 경우 XDoclet 을 이용한 소스코드의 생성을 지원하며, iBATIS 의 경우 구현되어 있지는 않지만 SQL 을 이용한 클래스 코드의 자동생성이 가능하다. 이처럼 메타 정보를 이용한 코드의 자동생성은 Metaprogramming 기법을 이용한다[4].



(그림 3) Metadata 를 이용한 OOP Code Generate[4]

이 기법은 도메인 독립적인 정보와 그 정보를 표현할 Template 을 가지고, 생성할 도메인의 정보를 입력받아 도메인에 맞는 코드를 자동 생성한다[4]. 본 논문의 제안 모델은 SQL 을 이용하여 클래스 코드를 생성하는데 이 기법을 이용한다.

ORM 의 매핑은 메타정보를 Cache 에 저장해 두고 요청정보를 분석하여, 매핑 정보를 Cache 에서 조회하여 질의를 실행, 결과를 보여줄 클래스를 동적으로 로딩하여 데이터를 매핑한다. 개발단계에 이미 결정되는 객체대 관계의 매핑을 운영시에 일으킴으로써 불필요한 성능 저하와 Cache 의 병목현상을 가져올수 있다. 이런 문제점을 해소하기 위해 본 논문에서는 동적 매핑이 아닌 래핑(wrapping)을 사용한다.



(그림 4) Object Wrapper Design[5]

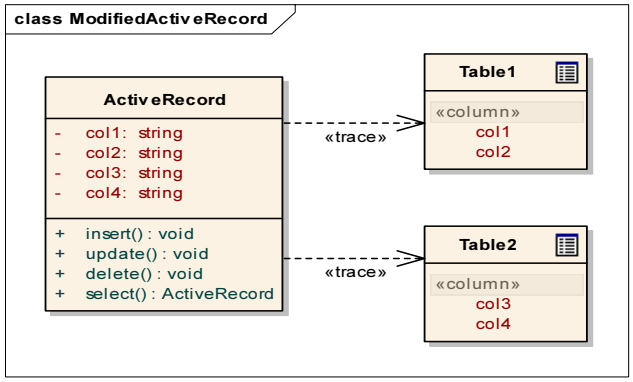
최근 많은 주목을 받고 있는 Ruby 의 Rails 프레임워크에서는 객체와 테이블을 래핑을 이용한 기법으로 처리한다[7]. ActiveRecord 라는 디자인 패턴을 이용하여 데이터 관리와 그 데이터를 처리하는 메소드를 한 클래스에서 처리 한다. 운영시에 동적으로 매핑하는 것이 아닌 Compile 시점에 매핑을 결정해서 운영시점의 부하를 줄인다. Rails 의 경우는 관계를 기술하는 메타정보는 없지만, 테이블의 관계를 소스코드상에 기술해 주어야 하기 때문에 데이터모델과 개발 소스코드가 밀접하게 연관되어, ORM 과 마찬가지로 개발

이 복잡하고, 모델 구조의 변화가 발생하였을 때 소스코드를 그에 맞게 변경해주어야 하는 단점이 있다.

3. 제안모델

제안 프레임워크는 기존의 문제점인 복잡한 메타정보와, 성능 저하를 일으키는 동적 매핑을 제거하기 위해 ActiveRecord 패턴을 변형하여 SQL 을 래핑하는 구조를 이용한다. ActiveRecord 패턴은 아래와 같이 정의 된다.

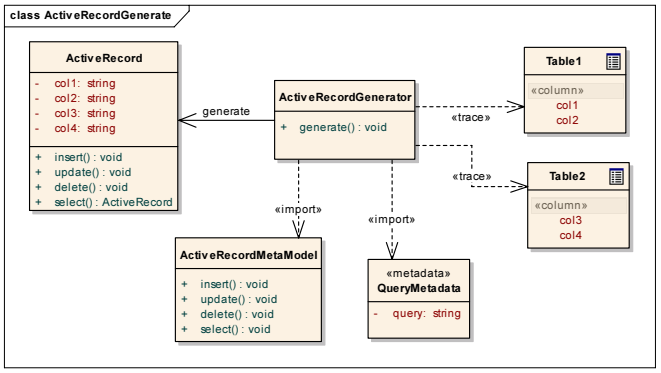
An object that wraps a row in a database table or view, encapsulates the database access, and adds domain logic on that data[3]



(그림 5) Modified ActiveRecord Architecture

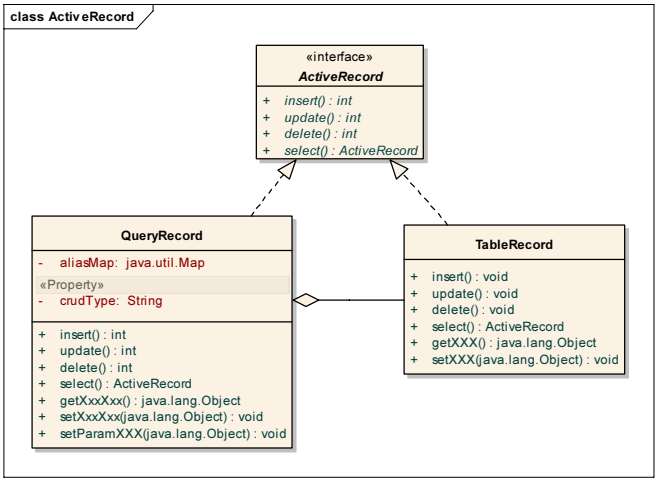
ActiveRecord 는 테이블의 정보를 객체로 래핑하여 데이터와 데이터를 처리하는 행위를 한 객체에서 처리한다. 이 방법 역시 테이블간 연관성이 소스에 포함 되어 구현되어야 한다.

본 논문에서는 ActiveRecord 의 방식을 변형한 SQL 래퍼를 이용한다. RDB 와 응용프로그램의 구조적 연관성을 없애고, 응용프로그램은 데이터를 처리하는 행위에만 집중하게 한다. 즉 데이터베이스의 데이터를 처리하기 위한 SQL 을 ActiveRecord 로 래핑하고 관련 데이터를 처리하도록 하는 구조를 가진다. 이 구조는 테이블 구조가 변경 되더라도 쿼리만 변경하고 소스는 빌드툴을 이용하여 자동 생성하는 것으로 변경의 반영이 완료 된다. 그리고 RDB 밴더에서 제공하는 SQL 을 이용한 튜닝 요소도 반영이 가능하다.



(그림 6) ActiveRecord MetaProgramming

빌드툴에서는 SQL 의 결과와 RDB 의 정보를 RDB 로 부터 조회하여 ActiveRecord 를 생성한다. 생성시 RDB 로 부터 조회 SQL 에서 이용된 테이블 리스트 및 그 테이블의 컬럼 리스트등을 함께 조회하여 SQL 에서 사용된 테이블의 ActiveRecord 도 함께 생성한다.



(그림 7) 자동생성되는 ActiveRecord Class

래퍼는 SQL 을 기준으로 다테이블을 처리하는 래퍼와 단일 테이블을 처리하는 래퍼를 분리 한다. 위 (그림 9)에서 QueryRecord 는 조인등의 복잡한 쿼리 구조를 처리하는 SQL 을 감싸는 ActiveRecord 이고, TableRecord 는 단일 테이블을 감싸는 ActiveRecord 이다. QueryRecord 는 감싸는 SQL 의 테이블 포함구조를 TableRecord 를 포함하는 형식으로 생성된다. 일반 SELECT 의 처리 이외에 INSERT, UPDATE, DELETE 의 처리는 별도의 SQL 작성 없이 자동으로 생성된다. 이렇게 생성된 클래스는 어떤 메타정보도 없이 즉시 사용 가능하다.

퍼시스턴스 프레임워크는 독립적으로 구동할 수 없기 때문에 테스트를 위해 응용프레임워크를 함께 구현 하였다. 응용프레임워크는 개별 퍼시스턴스 프레임워크를 이용한 응용프로그램에 동일한 환경을 부여함으로써 일관된 코드작성을 가능하게 한다.

4. 비교분석

본 장에서는 Hibernate, iBATIS 를 제안 프레임워크와 성능 테스트를 통해 비교한다. 본 논문에서 제안하는 프레임워크를 “SQL 래퍼”라고 호칭한다.

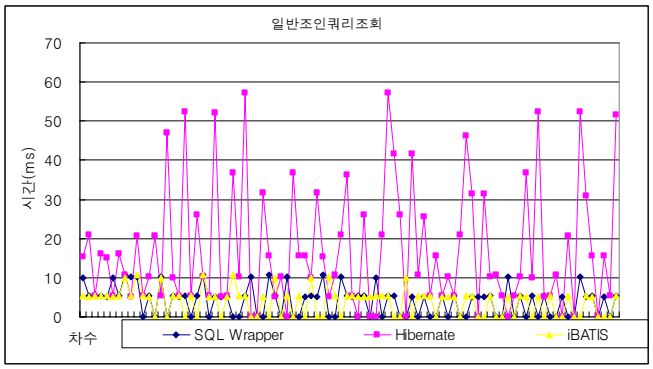
프레임워크는 Java 로 개발되었으며 동일한 환경에서 테스트 하기 위해 Hibernate 와 iBATIS 도 Java 버전을 사용 하였다. 테스트 환경은 아래와 같다.

- ✓ ServletContainer : Tomcat 5.5, Heap Size 512m
- ✓ JDK : Sun J2sdk 1.4.2_15
- ✓ Database : Oracle 10g Express Edition
- ✓ Hibernate 3.2
- ✓ iBATIS 2.3.0.677

테스트는 세종류의 SELECT 문을 3 회에 걸쳐 100 회씩 실행하고, 수행 시간을 최대/최소값 5 건씩을 뺀 나머지 평균 수행시간을 측정 하였다. 그리고 똑같은 기능을 구현하기 위해 개발해야 할 Java 소스와 메타정보 파일의 LOC(Line Of Code)를 비교 하였다.

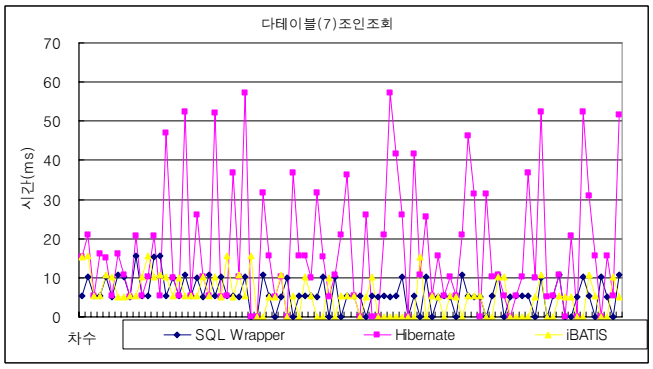
SQL 은 4 개 테이블을 조인하는 일반 조인쿼리, 7 개

테이블을 조인하는 다테이블 조인 쿼리, 4 개 테이블이 조인하고 대량 데이터(210 만건)를 조회하는 쿼리를 이용하여 테스트 하였다.



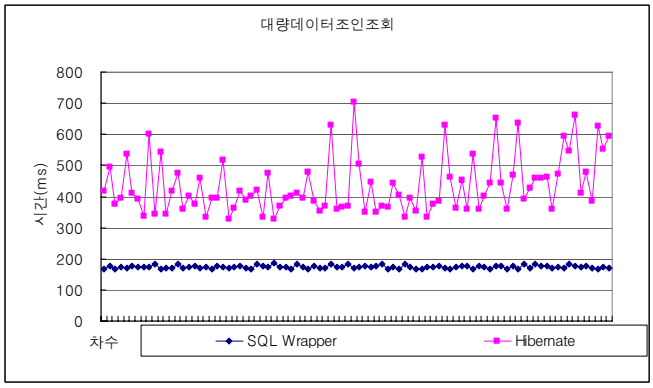
(그림 8) 일반조인쿼리 수행결과

일반조인쿼리를 실행 하였을 때는 Hibernate 의 경우 0 ~ 37ms 의 분포를 보였고, iBATIS 와 SQL 래퍼의 경우 0 ~ 10ms 으로 안정적인 분포를 보였다. iBATIS 의 경우 최초 호출시 Cache 에 등록하기 위한 추가시간이 필요하였다.



(그림 9) 다테이블조인쿼리 수행결과

다테이블 조인쿼리의 결과는 일반 조인쿼리를 수행할때 걸리는 시간과 크게 다르지 않았다.



(그림 10) 대량데이터 조인쿼리 수행결과

대량데이터 처리 시 iBATIS 는 OutofMemoryError 를 발생시키며, 수행이 중지 되었다. 건수를 줄여서 조회하면 900ms 이상의 많은 시간이 소모 되어 대량데이터 처리에서는 서버의 메모리 사용에 과부하가 발생하였다. Hibernate 의 경우는 310ms ~ 700ms 의 높은 수행시간을 보였고, SQL 래퍼는 160ms ~ 180ms 의 빠르고 안정적인 수행시간을 보였다.

아래는 테스트를 하기 위해 구현한 파일의 수와 구현이 필요한 부분의 코딩라인이다. Java Code 는 Sun 의 "Code Conventions for the JavaTM Programming Language"을 준수하여 코딩하고 LOC 에 주석은 제외하였다.

<표 1> Line of Code(LOC) 및 기능 비교

Framework	관련파일수	LOC
SQL 래퍼	ActiveRecord : 10 query.xml : 1	0 89
Hibernate	POJO : 13 Table.hbm.xml : 9	661 139
iBATIS	POJO : 3 QueryMapper.xml : 3	299 138

SQL 래퍼 와 Hibernate 의 경우 위의 코드 만으로 Insert 나 Update, Delete 의 처리가 가능하지만 iBATIS 의 경우 추가 개발이 필요하다.

5. 결론 및 향후 연구과제

제안 퍼시스턴스 프레임워크는 기존 프레임워크에서 취하고 있는 응용 프로그램과 RDB 의 밀접한 연관성을 부여하는 아키텍처를 취하지 않고 RDB 의 테이블 구조와 상관없이 응용프로그램은 데이터를 처리하기 위한 행위 만을 관리하고, 그 행위를 SQL 을 이용하여 처리함으로써 기존의 프레임워크 보다 빠른 처리 속도와 구조의 단순성을 제공한다. 그리고 Metaprogramming 을 이용하여 소스코드를 자동으로 생성함으로써 개발의 편의성과 변경에 유연한 구조를 가지게 되었다.

제안 프레임워크의 구조는 MDA(Model Driven Architecture)등의 모델 기반 아키텍처의 부분으로 이용이 연구되면 복잡한 DB 작업이 많은 기업 시스템에서 사용자 인터페이스 부터 RDB 까지의 소스코드의 자동생성이 가능해 개발 생산성 및 유지보수에 많은 잇점을 제공할 수 있을 것으로 보인다.

참고문헌

[1] Hibernate Reference Documentation Version 3.2.2
[2] iBATIS Data Mapper Version 2.0 Developer Guide
[3] Marin Fowler, David Rice, Matthew Foemmel and Edward Hieatt, Robert Mee., "Pattern of Enterprise Application Architecture", Addison-Wesley Professional
[4] Tom Mens and Tom Tourw´, "A Declarative Evolution Framework for Object-Oriented Design Patterns", Proceedings of the IEEE International Conference on Software Maintenance (ICSM'01)
[5] Sonia Bergamaschi, Alessandra Garuti, Claudio Sartori and Alberto Venuta., "Object Wrapper: an Object-Oriented Interface for Relational Databases", 23rd EUROMICRO Conference '97 New Frontiers of Information Technology
[6] Gavin King, Christian Bauer., "Hibernate Object/Relational Persistence for idiomatic Java", 2004
[7] Steve Vinoski, "Enterprise Integration with Ruby", IEEE Internet Computing, 2006.