DOI: 10.3745/KIPSTD.2008.15-D.4.549

SQL 기반 퍼시스턴스 프레임워크

조 동 일[†] · 류 성 열[††]

요 약

기업의 웹 기반 인트라넷 시스템은 객체지향 언어로 개발되고, 데이터의 관리는 RDBMS를 이용하여 구축된다. 두 시스템은 이질적 패러다임에 기인하여 모델의 불일치성을 발생시킨다. 이 문제를 해결하고자 사용되는 ORM 프레임워크는 RDB의 테이블과 객체지향 언어의 객체를 매핑하는 구조로 응용프로그램의 개발이 복잡하고, 변경에 유연하지 못하여 기업형 인트라넷 시스템의 개발 및 유지보수에 어려움을 준다.

본 연구에서는 기존 ORM 프레임워크의 복잡성을 해소하고, 변경에 유연하여 기업의 인트라넷 시스템에 적합한 퍼시스턴스 프레임워크를 제안한다. 제안한 퍼시스턴스 프레임워크는 테이블의 엔티티와 객체를 매핑하는 매핑 메타정보가 불필요하고, 소스코드를 자동 생성하여 개발 및 유지보수의 편의성을 제공하고, 변경에 유연하다. 제안 프레임워크는 Hibernate, iBATIS와의 테스트 결과 iBATIS와는 처리속도가 비슷했으나 iBATIS는 대용량 데이터 처리시 문제를 나타냈으며, Hibernate보다 약 3배 빠른 속도를 보였다.

키워드 : 퍼시스턴스 프레임워크, 객체관계매핑, 프레임워크, SQL래핑

A Persistence Framework Based SQL

Dongil-Il Cho[†] · Sung-Yul Rhew[††]

ABSTRACT

Web-based Enterprise Intranet System is developed Object-oriented programming language and data management is constructed using RDBMS. Between Object-oriented programming language and RDBMS bring about "Object-Relational Impedance Mismatch" due to heterogeneous paradigm. To solve this kinds of problems commonly use the ORM Framework that it is converting data between incompatible type systems in databases and object-oriented programming languages, uses complex mapping metadata. It causes difficult to develop and maintain because of inflexible in changes.

This paper proposed persistence framework that solve the existing complexity of ORM framework and has more flexible in evolution of database table. This persistence framework is unnecessary meta information that connecting with entity of database table and the objects, it offers users convenience of maintenance and it allows far more flexible and affordable systems to be built because of automatically code generation. The result of testing based on the proposed persistence framework with Hibernate, iBATIS, It is similar response time with iBATIS and it has more about three times faster than Hibernate. But iBATIS has problems of mass data processing.

Key Words : Persistence Framework, ORM, Framework, SQL Wrapping

1. 서 론

객체지향 응용프로그램과 RDB 간의 연동 문제는 기업의 웹 기반 인트라넷 시스템에서 중요한 이슈가 된다. 이 둘간의 연동은 이질적인 시스템 패러다임과, 네트워크로 분리된 분산 환경, 그리고 각 솔루션마다 서로 다른 인터페이스의 영향으로 모델의 불일치성(object-relational impedance mismatch)을 발생시킨다[1]. 이런 문제를 해결하기 위해 퍼시스턴스 프레임워크(Persistence Framework)가 사용된다[2].

퍼시스턴스 프레임워크는 ORM(Object-Relational Mapping)을 이용하여 마치 메모리의 객체 정보를 변경하는 것처럼 RDB의 데이터를 변경할 수 있는 인터페이스를 제공 한다. 그리고 DBMS의 종류에 상관없이 동일하게 개발할 수 있어서 코드의 재활용성을 높일 수 있다. 하지만 ORM 프레임워크는 객체와 RDB를 매핑하기 위한 복잡한 매핑 데이터를 필요로 하고, 네이밍 기반 매핑으로 인하여 오류를 유발할 수 있다. 또한 매핑이 동적으로 일어나기 때문에 성능 저하의 원인이 되고, SQL을 운영시점에 자동으로 생성하여 쿼리를 이용한 튜닝요소를 사용할 수 없다.

기업의 비즈니스 조직은 지속적으로 변화하고, 비즈니스 요구사항 및 환경 그리고 프로세스 또는 워크플로우는 이에 적절하게 대응해야 하며, 새로운 비즈니스 요구에 신속하게

※ 본 연구는 숭실대학교 교내 연구비 지원으로 이루어졌음.
† 정 회 원 : 숭실대학교 컴퓨터공학과 박사과정
†† 종신회원 : 숭실대학교 교수
논문접수 : 2007년 12월 11일
수 정 일 : 1차 2008년 3월 3일, 2차 2008년 3월 19일, 3차 2008년 3월 25일
심사완료 : 2008년 4월 3일

적응해야 한다[3]. 이런 변화 속에 인트라넷 시스템은 변경에 유연한 구조를 가져야 한다. 또한 기업 인트라넷 시스템은 복잡한 테이블 구조를 가지고 있고 복잡한 SQL을 수행해야 한다. ORM 프레임워크가 많은 기능을 가지더라도 이런 인트라넷 환경에서 ORM 프레임워크가 가지는 복잡성은 개발 및 유지보수에 많은 어려움을 준다. 그리고 운영자 및 유지보수자 입장에서 기존 시스템의 DBMS가 바뀌는 상황은 시스템의 재개발과 함께 오는 경우가 대부분이어서 ORM 프레임워크의 여러 DBMS의 지원은 불필요할 뿐만 아니라 운영 시 처리속도 저하의 원인이 된다.

본 연구에서는 기업의 인트라넷 시스템 개발 및 유지보수에서 기존 퍼시스턴스 프레임워크가 가지는 문제점을 개선한 퍼시스턴스 프레임워크를 제안한다. 제안한 퍼시스턴스 프레임워크는 객체와 관계를 매핑이 아닌 래핑(Wrapping)을 이용하여 처리하여 매핑에 필요한 복잡한 메타데이터가 필요 없는 구조를 가지고 있다. 그리고 테이블이 아닌 SQL를 래핑하여 인트라넷 시스템의 복잡한 쿼리의 수행을 편리하게 하고 유지보수성을 증대 시켰다. 또한 반복적으로 개발되는 래퍼를 자동 생성하는 빌드툴을 함께 제공해 개발 생산성을 증대 시켰다. 이후 제안한 프레임워크를 SQL래퍼라고 호칭하기로 한다.

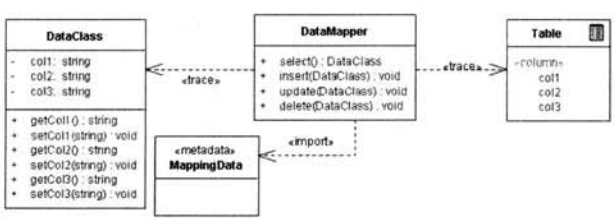
2. 관련연구

2.1 ORM Framework

객체지향 언어에서는 모델의 불일치성을 해소하기 위해 몇 가지 디자인 패턴을 제시하고 있다[4]. 대표적으로 ORM, SQL Mapper, ActiveRecord라는 디자인 패턴이 적용되고 있으며 각각 Hibernate, iBATIS, Rails가 이에 속한다.

2.1.1 Hibernate

Hibernate는 매핑정보를 통해 RDB의 테이블을 오브젝트와 매핑한다[5]. 응용프로그램에서는 마치 메모리 값을 수정하는 것처럼 RDB의 데이터를 조작할 수 있는 기능을 제공한다. Hibernate는 HQL(Hibernate Query Language)이라는 DBMS에 독립적인 질의 언어를 지원하여 RDB 종류에 상관없이 프레임워크에서 적당한 SQL로 자동 변환 한다[5]. Hibernate는 개별 도메인에서는 사용되지 않는 폭넓은 기능을 제공하기 위해 복잡한 메타정보를 필요로 한다. 이 메타정보들은 네이밍 기반으로 서로 유기적으로 연결되어 있어 작성이 힘들고, 변경 발생시 수정이 어려우며, 에러를 발생시킬 확률이 높다. 또한 테이블과 매핑되는 클래스가 존재해야 하고, 클래스 내부에 테이블의 관계와 유사한 클래스 간의 관계가 표현되어야 하기 때문에 시스템의 변화가 잦은 기업형 인트라넷 시스템에서는 개발 및 유지보수에 많은 어려움을 준다.



(그림 1) Object Relational Mapping Architecture[4]

2.1.2 iBATIS

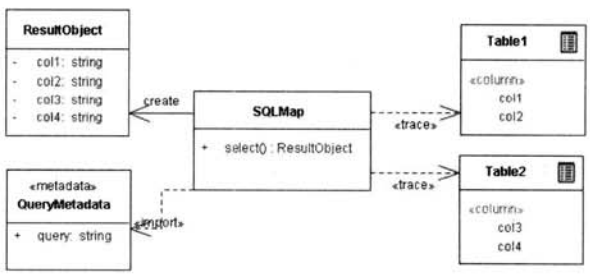
iBATIS는 메타정보를 이용한 SQL Mapping 기법을 사용한다[6].

iBATIS는 Hibernate와 같이 강력한 기능은 지원하지 못하는 반면, 구조가 간단하고, 빠른 처리성능을 가진다 [6]. 하지만 기능이 협소하고, SQL이 많아 질수록 SQL에 대응하는 입력 클래스 및 결과 클래스가 반복적으로 개발되어야 한다. 그리고 SQL과 클래스를 매핑하기 위한 매핑 데이터가 추가로 필요하고, 클래스의 재사용이 어렵다.

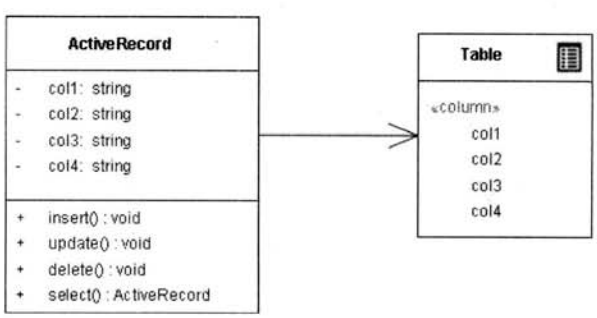
2.1.3 Rails

최근 많은 주목을 받고 있는 Ruby의 Rails 프레임워크는 객체와 테이블을 래핑으로 처리한다[7]. Rails의 ActiveRecord는 데이터와 데이터를 처리하는 메소드가 한 클래스에 존재하여 직관적인 개발이 가능하고 테이블의 관계가 소스 코드상에 기술되기 때문에 동적 매핑으로 인한 운영시점의 부하를 줄여준다.

Rails는 관계를 기술하는 매핑 데이터는 없지만 테이블과 클래스가 1대1로 래핑되고, 테이블간의 관계를 소스코드에 매크로를 이용해 기술해 주어야 한다. 그렇기 때문에 Rails는 RDB의 테이블 구조를 소스코드상에 그대로 기술해 주어



(그림 2) SQL Mapping Architecture[6]



(그림 3) Rails의 ActiveRecord

야 하는 관리의 이중성을 가지고 있다.

2.2 메타프로그래밍

퍼시스턴스 프레임워크의 테이블과 매핑되는 클래스는 정해진 매핑 룰에 의해 소스의 자동 생성이 가능하다.

Hibernate, Oracle Toplink의 경우 XDoclet을 이용한 소스코드의 생성을 지원한다. 메타 정보를 이용한 코드의 자동생성은 Meta-programming 기법을 이용한다[8].

메타프로그래밍은 도메인 독립적인 모델과 그 정보를 표현할 템플릿을 이용하여, 생성할 도메인의 정보를 입력 받아 도메인에 맞는 코드를 자동 생성한다[8].

메타프로그래밍은 소스코드를 자동 생성하여 개발 공수를 줄여주고 High Level Refactoring을 가능하게 하여 소스코드의 품질을 일괄적으로 상승시킬 수 있다. 또한 변경사항을 쉽게 반영할 수 있다.

하지만 생성된 소스코드가 완성된 소스코드가 아닌 중간 단계의 템플릿만 제공하거나, 소스코드의 생성 절차가 복잡하다면 오히려 개발 생산성 및 유지보수성을 저하시키는 원인이 되기도 한다.

2.3 퍼시스턴스 프레임워크의 성능비교

웹 기반 객체지향 프로그램 개발에서 ORM프레임워크의 중요도가 높아짐에 따라 ORM프레임워크의 성능 비교에 관한 연구가 이루어져 왔다. 웹 환경에서 운영되는 퍼시스턴스 프레임워크는 응용프로그램과 결합시켜 최종 사용자 입장에서 느끼는 처리시간을 측정하는 성능측정(Performance Test)과 ORM 프레임워크간의 지원 기능 및 요구조건을 가지고 비교할 수 있다[9, 10].

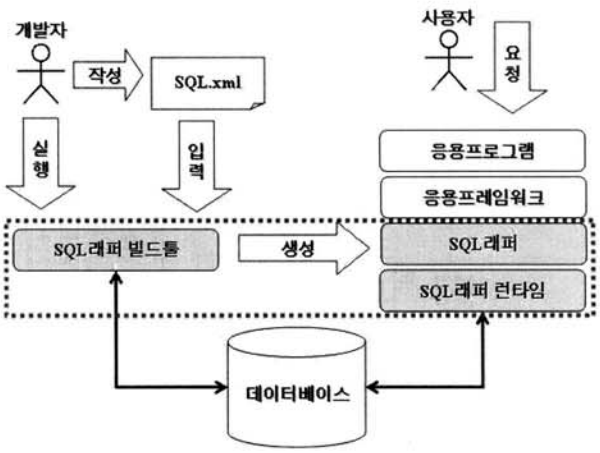
3. SQL기반 퍼시스턴스 프레임워크

기업 인트라넷 시스템은 아래와 같은 특징을 가진다고 가정한다.

- 여러 테이블을 조인하는 복잡한 SQL을 처리
- 주된 프로세스는 조회 작업
- 잦은 테이블의 변경
- 대용량 데이터의 처리

SQL래퍼는 이런 요구사항을 수용하기 위해 아래와 같은 구조를 가진다.

- SQL을 래핑하여 복잡한 SQL을 응용프로그램에서 쉽게 처리할 수 있도록 한다
- RDB의 구조에 영향을 받는 매핑 및 메타데이터를 제거하여 RDB의 변경에 유연성을 제공한다
- 동적 리플렉션(Runtime Reflection)을 사용하지 않고, 데이터 클래스의 부피를 최소화 한다
- 빌드툴을 제공하여 래퍼의 개발을 자동화 한다



(그림 4) 제안프레임워크 구성도

3.1 제안 프레임워크 구성도

제안 프레임워크를 개발자와 사용자 관점에서 분리하면 다음 그림과 같다.

개발자는 SQL을 작성하고, SQL래퍼 빌드툴을 이용하여 SQL래퍼를 생성한다. SQL래퍼 빌드툴은 SQL과 데이터베이스 정보를 이용하여 SQL래퍼를 생성한다.

사용자는 웹을 통해 응용프로그램에 서비스를 요청한다. 응용프로그램은 SQL래퍼를 이용하여 데이터베이스와 통신한다. SQL래퍼 런타임은 SQL래퍼가 구동할 수 있는 구동환경을 제공한다.

3.2 SQL래퍼

SQL래퍼는 복잡한 매핑 메타데이터와, 성능 저하를 일으키는 동적 매핑을 제거하기 위해 ActiveRecord패턴을 이용하여 SQL을 래핑하는 구조를 가진다.

ActiveRecord 패턴은 다음과 같이 정의된다.

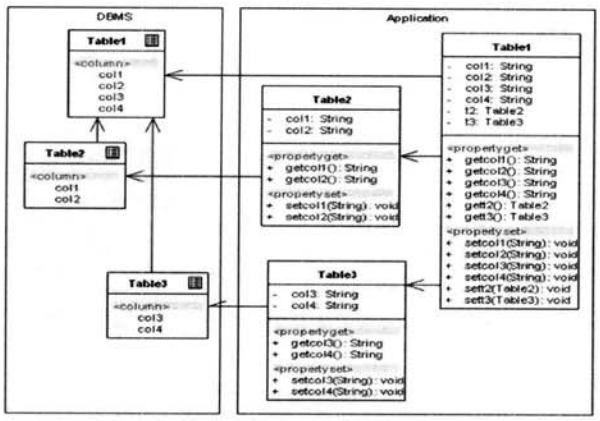
데이터베이스의 테이블 또는 뷰를 래핑하고 데이터베이스 엑세스 로직과 도메인 로직이 추가된 오브젝트[4]

ActiveRecord는 모델의 정보를 클래스로 래핑하고 데이터를 처리하는 메소드를 함께 가지고 있다.

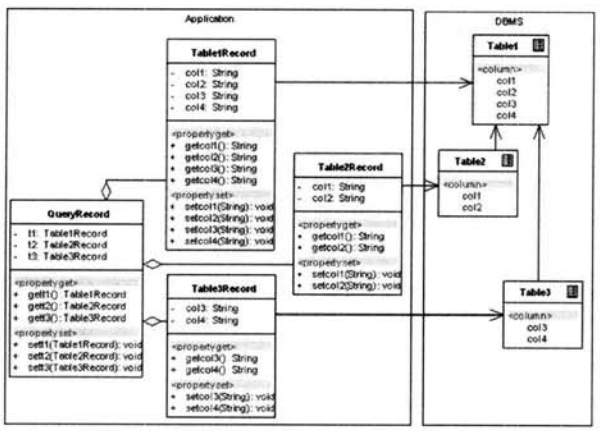
Rails에서 ActiveRecord 패턴을 구현하고 있다. Rails는 테이블을 ActiveRecord로 래핑하고, 테이블간 연관성을 클래스에 매크로로 정의한다. 그리고 테이블과 ActiveRecord 간의 매핑을 네이밍 기반으로 처리하기 때문에 매핑 데이터가 불필요하다. 하지만 Rails는 클래스 대 테이블을 1대1로 래핑하고, 테이블의 연관성을 클래스에 기술해 주어야 하기 때문에 테이블 변경 시 소스코드의 변경이 불가피하다.

반면 SQL래퍼는 테이블이 아닌 SQL을 래핑한다. SQL의 래퍼 클래스는 테이블을 조인하는 SELECT SQL을 래핑하고 SQL의 From절에 기술되는 테이블들의 래퍼를 포함하는 구조를 가진다. (그림 5)와 (그림 6)은 Rails와 SQL래퍼의 구조를 도식화한 그림이다.

SQL래퍼에서는 RDB 테이블 간의 관계를 클래스 또는 매핑 정보에 기술하지 않고 SQL을 래핑하여 처리한다. 이와 유사하게 iBATIS의 SQLMapper는 SQL과 클래스를 매



(그림 5) Rails의 ActiveRecord



(그림 6) SQL래퍼의 ActiveRecord

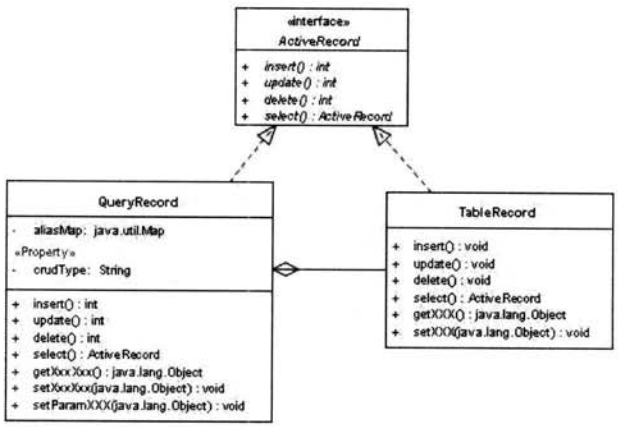
핑하여 RDB 테이블의 관계를 클래스에 기술하지 않아도 되는 장점이 있다. 하지만 테이블과 매핑되는 클래스가 없기 때문에 테이블에 접근하는 모든 SQL을 개발해 주어야 하는 불편함이 있다.

SQL래퍼는 결과적으로 Rails의 ActiveRecord를 이용하여 매핑 메타데이터를 제거하고, 테이블단위 CRUD를 쉽게 처리할 수 있게 하였고, iBATIS의 SQLMapper의 SQL 직접 접근 방법을 이용해 SQL를 직접 래핑하여 복잡한 쿼리를 쉽게 처리할 수 있는 구조를 제공한다. 그리고 SQL을 직접 래핑하기 때문에 DBMS 밴더에서 제공하는 SQL을 이용한 튜닝 요소도 사용할 수 있다.

(그림 7)은 SQL과 테이블을 래핑하는 클래스의 상위클래스의 클래스 다이어그램이다.

QueryRecord는 조회 SQL을 감싸는 래퍼 클래스이고, TableRecord는 각 테이블의 데이터를 감싸는 래퍼 클래스이다. QueryRecord는 SQL의 From절에 기술된 테이블들을 TableRecord를 통해 포함하는 구조로 생성된다. 일반 SELECT의 처리 이외에 INSERT, UPDATE, DELETE의 처리는 별도의 SQL작성 없이 TableRecord를 이용해 처리한다.

이 구조는 데이터와 데이터를 처리하는 메소드를 한 클래스에서 제공하면서 하위 클래스인 SQL래퍼는 데이터만 가



(그림 7) SQL래퍼 상위 클래스

지고, 데이터의 처리는 상위클래스에서 담당하여 클래스를 경량화 시킬 수 있다. 또한 클래스의 포함관계가 생성시 이미 정해지므로 동적 리플렉션 없이 객체 생성이 가능하다.

3.3 SQL래퍼 빌드툴

반복적인 래퍼의 개발은 빌더 툴을 제공함으로써 자동화할 수 있다. 테이블의 구조가 변경되면 변경을 반영할 SQL만 변경하고 소스코드는 빌더 툴을 이용하여 자동 생성함으로써 변경의 반영이 완료된다.

빌더 툴을 이용하기 위한 SQL작성은 아래와 같은 룰을 가지고 일반 쿼리와 유사하게 작성된다.

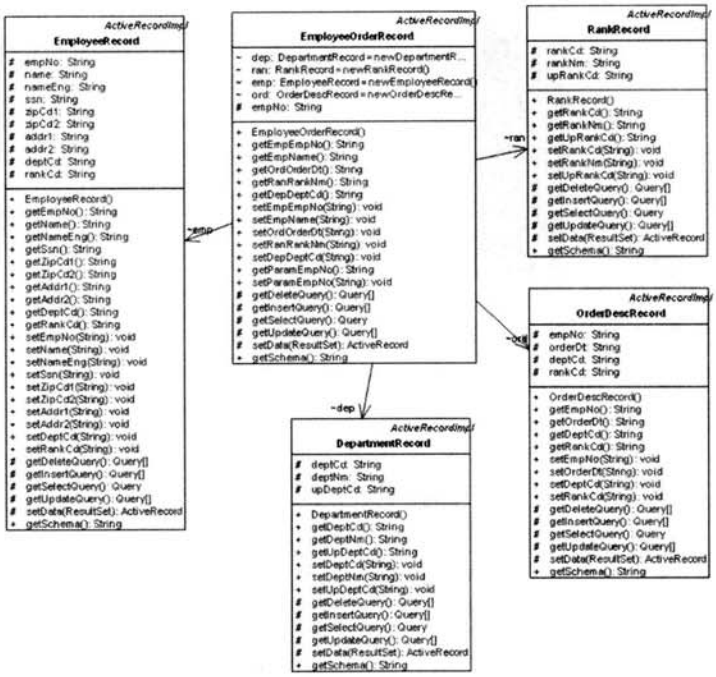
- 파라미터 : "{:parameter_name@parameter_type}"
- 조회 컬럼 : "테이블 알리아스@컬럼알리아스"

- 작성 예 :

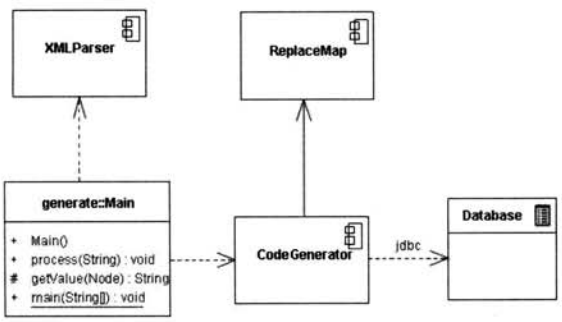
```
...
<query>
<name>Employee_Order</name>
<sql> SELECT EMP.EMP_NO   AS EMP@EMP_NO
     , EMP.NAME         AS EMP@NAME
     , ORD.ORDER_DT     AS ORD@ORDER_DT
     , RAN.RANK_NM      AS RAN@RANK_NM
     , DEP.DEPT_NM      AS DEP@DEPT_CD
  FROM EMPLOYEE     EMP
     , ORDER_DESC ORD
     , RANK         RAN
     , DEPARTMENT DEP
 WHERE EMP.EMP_NO  = {:emp_no@String}
   AND ORD.EMP_NO  = EMP.EMP_NO
   AND RAN.RANK_CD = ORD.RANK_CD
   AND DEP.DEPT_CD = ORD.DEPT_CD
ORDER BY ORD.ORDER_DT DESC</sql>
...
```

작성된 SQL은 XML로 감싸지고, 래퍼 클래스는 개별 SQL을 래핑한다. SQL의 조회 컬럼과, 조건 컬럼들은 (그림 8)과 같이 클래스의 필드 및 메소드로 자동 생성된다.

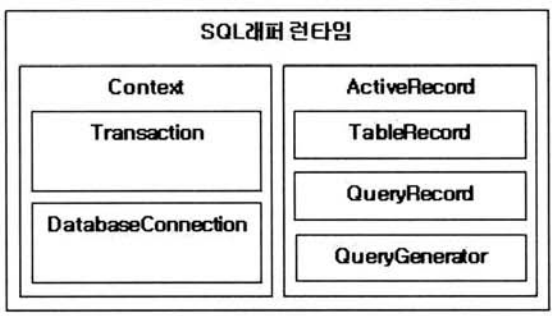
중앙 상위에 "EmployeeOrderRecord"가 SQL을 래핑한 래퍼이고, "EmployeeOrderRecord"가 포함 하는 "EmployeeRecord",



(그림 8) 빌드툴로 생성된 SQL래퍼



(그림 9) SQL래퍼 빌드툴의 컴포넌트 연관도



(그림 10) SQL래퍼 런타임 구조

"DepartmentRecord", "RankRecord", "OrderDescRecord"는 SQL의 From절에 기술된 테이블들을 처리하는 테이블 래퍼이다.

SQL래퍼 빌드툴은 (그림 9)와 같은 구조를 가진다.

빌드툴은 SQL을 임의의 파라미터를 설정하여 실행하고 그 결과와 RDB의 데이터베이스 메타정보로부터 추출된 정보를 이용하여 ActiveRecord를 생성한다. 생성시 SQL의 메타정보와 데이터베이스 메타정보를 이용하여 래퍼클래스를 자동 생성한다.

3.4 SQL래퍼 런타임

SQL래퍼 런타임은 SQL래퍼의 구동 환경을 제공한다.

SQL래퍼 런타임은 Context모듈과 ActiveRecord 모듈로 구성된다.

Context 모듈은 개별 ActiveRecord 모듈의 생명주기와 트랜잭션(Transaction)을 관리 한다.

ActiveRecord모듈은 SQL래퍼의 상위클래스를 제공하고, Context모듈과 상호작용하여 SQL래퍼를 통한 응용프로그램의 요청을 처리한다.

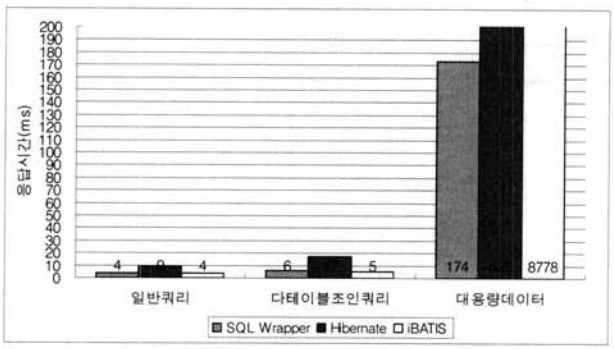
4. 성능 및 속성 비교

본 장에서는 기존 퍼시스턴스 프레임워크와 본 연구에서 제안하는 프레임워크간의 성능 및 속성을 비교 분석한다. 성능테스트 및 LOC의 비교는 개발 언어와 운영환경 등에 많은 영향을 받기 때문에 비교 대상 프레임워크는 SQL래퍼가 개발된 Java기반 프레임워크 중 많이 사용되는 Hibernate와 iBATIS로 하였다.

4.1 성능 테스트

성능 테스트 환경은 다음과 같다.

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



(그림 11) 시나리오 별 프레임워크 별 평균응답시간

- Hibernate 3.2
- iBATIS 2.3.0.677

테스트는 기업 인트라넷 주된 프로세스인 조회작업을 테스트 한다. 세 종류의 SELECT문을 3회에 걸쳐100회씩 실행하고, 수행 시간을 최대값 5건과 최소값 5건을 뺀 나머지의 평균 수행시간을 측정 하였다.

테스트 쿼리는 아래와 같다.

1. 4개 테이블의 조인하는 조인 쿼리
2. 7개 테이블의 조인이 필요한 다 테이블 조인 쿼리
3. 4개 테이블이 조인해야 하는 210만 건의 대량 데이터를 조회 쿼리

각 퍼시스턴스 프레임워크 별로 동일한 SQL을 사용하여 데이터베이스 상에서의 처리 속도의 차이를 제거하고 퍼시스턴스 프레임워크의 처리 속도 차이만 발생하도록 하였다.

〈표 1〉 Line of Code(LOC) 및 기능비교

Framework	관련 파일	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

일반조인쿼리를 실행 하였을 때는 Hibernate의 경우 0~37ms의 분포를 보였고, iBATIS와 SQL래퍼의 경우 0~10ms를 보였다. iBATIS의 경우 최초 호출 시 Cache에 등록하기 위한 추가시간이 필요하였다. 다 테이블 조인쿼리의 결과는 일반조인쿼리를 수행할 때 걸리는 시간과 크게 다르지 않았다.

대량데이터 처리에서 iBATIS는 OutofMemoryError를 발생시키며, 수행이 중지 되었다. 건수를 줄여서 조회하면 900ms 이상의 많은 시간이 소모 되어 대량 데이터 처리에서는 서버의 메모리 사용에 과부하가 발생하였다. Hibernate의 경우는 310ms~700ms의 높은 수행시간을 보였고, SQL래퍼는 160ms~180ms의 수행시간을 보였다.

4.2 LOC 비교

다음은 테스트를 하기 위해 구현한 파일의 수와 구현이 필요한 부분의 코딩라인이다. Java Code는 Sun의 "Code Conventions for the JavaTM Programming Language"를 준수하였으며,

〈표 2〉 속성비교

	Tool	S	H	I
Support	Persists arbitrary classes	Y	Y	Y
	Supports grouping	Y	Y	Y
	Supports aggregate functions	Y	Y	Y
	Maintains single identities for objects	Y	Y	
	Resolves circular identities	Y	Y	
	Generates mapping and objects	Y	Y	
	Supports many-many and one-many ass	Y	Y	Y
	Collection support	Y	Y	Y
Require	Can fetch assoc. Objects using outer joins	Y	Y	Y
	Supports optimistic locking/versioning		Y	
	Support unit-of-work obj.1v1.transactions	Y	Y	Y
	Persistence through private fields	Y	Y	Y
	Persistence through accessors	Y	Y	Y
	Requires code generation/byte code processing	Y	Y	
	Requires runtime reflection			Y
	Requires Object-Relation association metadata		Y	Y

[S : SQL래퍼, H : Hibernate, I : iBATIS]

LOC에 주석과 자동 생성되는 소스코드와 환경 설정에 관계되는 메타데이터는 포함시키지 않고, 개발해야 하는 클래스와 매핑 메타데이터는 포함시켰다.

SQL래퍼는 소스코드를 자동으로 생성하여 LOC는 없고, 소스를 자동 생성하기 위해 작성한 query부분이 LOC에 포함되었다. SQL래퍼와 Hibernate의 경우 위의 코드 만으로 Insert나 Update, Delete의 처리가 가능하지만 iBATIS의 경우 추가 개발이 필요하다.

4.3 속성 비교

속성 비교 항목은 기존 퍼시스턴스 프레임워크 성능비교 자료를 토대로 하였으며, 결과는 아래와 같다.

SQL래퍼는 기능적인 측면에서는 Hibernate보다 뒤지나 iBATIS보다 월등한 기능성을 보였다. 요구조건에서 SQL래퍼는 가장 적은 요구조건을 필요로 하였고, Hibernate와 iBATIS는 동일하였다.

5. 결론 및 향후 과제

기업의 웹 기반 인트라넷 시스템은 객체 지향 응용프로그램과 RDB간의 모델의 불일치성을 해결하기 위해 퍼시스턴스 프레임워크를 사용한다.

퍼시스턴스 프레임워크는 응용프로그램에 RDB의 데이터를 메모리의 객체를 조작하는 것과 같은 인터페이스를 제공하여, 개발의 편의성을 제공한다. 그러나 개별 도메인에서는 불필요한 많은 기능을 제공하기 위한 복잡한 매핑 데이터는 개발 및 유지보수를 어렵게 하고 동적 매핑으로 인해 처리 속도를 떨어지게 한다.

본 연구에서는 변경이 잦은 기업 인트라넷 시스템에서 기존 퍼시스턴스 프레임워크의 문제점을 개선한 퍼시스턴스 프레임워크를 제안하였다. 제안 프레임워크는 객체와 RDB를 SQL을 이용하여 래핑하는 구조로 운영 처리속도를 증가시켰고, SQL을 기반으로 소스코드를 자동 생성하여 개발 및 유지보수성을 증대시켰다.

웹 기반 시스템은 하루가 다르게 복잡, 다양해 지고있으며, 시스템을 도입하는 도메인 또한 특성이 뚜렷해 지고 있다. 기존 퍼시스턴스 프레임워크가 많은 개발 편의성과 높은 성능을 제공하고는 있지만, 하나의 프레임워크가 가지는 너무 많은 기능성 때문에 개개의 도메인에서는 필요 없는 기능을 위한 인적, 하드웨어적 자원의 낭비가 심하다. 개개의 도메인별 특성에 맞고 빠른 처리 성능과 개발자 및 유지보수자 모두에게 편의성 제공하기 위해서는 각 도메인의 특성에 맞는 기능과 개발 방법을 제공하기 위한 퍼시스턴스 프레임워크의 인터페이스 개선에 관한 연구가 요구된다.

참 고 문 헌

[1] Gavin King, Christian Bauer., 'Hibernate Object/Relational Persistence for idiomatic Java,' Red Hat Middleware, 2004.

[2] Carvalho, S.R., Vianna e Silva, M.J., Melo, R.N., 'Persistent object synchronization with active relational databases,' Technology of Object-Oriented Languages and Systems. pp.53-62, 1999.

[3] Herbert Weber, Asuman Sunbul and Julia Padberg, 'Evolutionary Development of Business Process Centered Architectures Using Component Technologies,' IEEE International Conference on Systems Integration IDPT, Volume 5, Issue 3, pp.13-24, 2000.

[4] Marin Fowler, David Rice, Matthew Foemmel and Edward Hieatt, Robert Mee., 'Pattern of Enterprise Application Architecture,' Addison-Wesley Professional, pp.160-183, 2002.

[5] Red Hat Middleware, 'Hibernate Reference Documentation Version 3.2.2,' Red Hat Middleware, White-Paper, 2007.

[6] Apache Software Foundation, 'iBATIS Data Mapper Version 2.0 Developer Guide,' Apache Software Foundation, White-Paper, 2006.

[7] Steve Vinoski, 'Enterprise Integration with Ruby', IEEE Internet Computing, Vol.10, No.4, pp.91-95, 2006.

[8] Tom Mens and Tom Tourw, 'A declarative evolution framework for object oriented design patterns,' In Proceedings of Int. Conf. on Software Maintenance. IEEE Computer Society Press, pp.570, 2001.

[9] Sabu M.Thampi, 'Performance Comparison of Persistence Frameworks,' eprint arXiv of Cornell University : 0710.1404, 2007.

[10] Alexander Schirrer, 'Object-Relational Mapping, Theoretical Background and Tool Compatison,' Vienna University of Technology, 2004.

[11] Daniel A. Menascé, 'Load Testing of Web Sites,' IEEE Educational Activities Department. Vol 6, issue 4, pp.70-74, 2002.

[12] Marc Stampfli, 'Efficient Object-Relational Mapping for JAVA and J2EE Applications - or the impact of J2EE on RDB,' Oracle Software, 2004.

조 동 일

e-mail : chodongil@yahoo.co.kr

2003년 수원대학교 기계공학과(학사)

2008년 숭실대학교 정보과학 대학원 (공학석사)

2008년~현 재 숭실대학교 컴퓨터공학과 박사과정

2003년~현 재 (주)토마토시스템 기술연구소 선임연구원

관심분야 : BPM, EA, SOA

류 성 열

e-mail : syrhew@ssu.ac.kr

1976년 숭실대학교 전자계산학과(학사)

1980년 연세대학교 산업대학원 전자계산학과 (공학석사)

1997년 아주대학교 대학원 컴퓨터공학과 (공학박사)

1982년~1995년 숭실대학교 전자계산연구소 및 중앙전자계산소 소장

1992년~1993년 한국 정보과학회 총무 이사

1997년~1998년 George Mason University 객원 교수

1998년~2001년 숭실대학교 정보과학대학원 원장

1998년~현 재 숭실대학교 소프트웨어 공학센터 소장

2006년~현 재 공정거래위원회 성과관리위원회위원

2006년~현 재 기획예산처 정보화예산협의회 위원

2007년~현 재 사회복지정보화추진 위원회 위원

1981년~현 재 숭실대학교 교수

관심분야 : 소프트웨어 유지보수, 소프트웨어 재사용, 소프트웨어 재공학/역공학, 오픈소스 소프트웨어