KIPS Trans. Softw. and Data Eng.
Vol.7, No.8 pp.287~296 pISSN: 2287-5905

https://doi.org/10.3745/KTSDE.2018.7.8.287

Automation Tool Design for PL/SQL Applications Conversion

Jungeun Jee[†] · Jeongkun Lee[††] · Yongrak Choi[†††] · Yongtae Shin[††††]

ABSTRACT

In the recent commercial DBMS market, as the users' burden and complaint which are related to high price licensing policy and late technical support service are increasingly rising, the concern for use of open source DBMS which has no problem with compatibility or stability is escalating. Due to the fact, the cases saving the cost are growing by converting Oracle Corporation's applications, which has about 60% share in the DBMS market, to an open source DBMS. However, in converting non-interchange sentences to an ANSI standard-based open source DBMS because of the use of PL/SQL in Oracle Corporation provides, a lot of manual work accompanies, so there is a lot of loss of time and money. Therefore, a tool that automatically converts PL/SQL to standard SQL is required. The proposed automation tool for the conversion of applications converts PL/SQL to Java Stored Procedure, an open source DBMS-based ANSI standard programming language. Through carrying out a test on the automation tool, it is proved that the tool will contribute to shortening time and saving cost by verifying the identity of input-output data and its reliability after correcting errors in converting to Java Stored Procedure.

Keywords : Open Source DBMS, Conversion of Applications, Automatic Conversion Tool, Automatic Conversion to Java Stored Procedure

PL/SQL 응용프로그램 전환을 위한 자동화 도구 설계

지 정 은[†] · 이 정 근[††] · 최 용 락[†††] · 신 용 태[††††]

요 약

최근 상용 DBMS 시장은 고가의 라이선스 정책과 신속하지 않은 기술 지원 서비스 등에 대한 사용자들의 부담과 불만이 커지면서, 호환성이나 안정성에 문제가 없는 오픈소스 DBMS의 사용에 대한 관심이 확대되고 있다. 이로 인해 DBMS 시장의 약 60% 점유율을 갖고 있는 오라클사 기반의 응용프로그램을 오픈소스 기반의 DBMS로 전환하여 비용을 절감하는 사례가 많아지고 있다. 그러나 오라클사에서 제공하는 PL/SQL의 사용으로 인한 비호환적 문장을 오픈소스 DBMS로 전환 시, 많은 수작업을 동반하게 되어 시간 및 비용 손실이 크다. 따라서 PL/SQL을 표준 SQL로 자동 전환해주는 도구가 요구된다. 제안하는 응용프로그램 전환을 위한 자동화 도구는 PL/SQL을 ANSI 표준 프로그래밍 언어인 Java SP (Stored Procedure)로 전환한다. 자동화 도구 테스트 실행을 통해 입출력 데이터의 동일성을 확인하고, 발생하는 Java SP 전환 오류 수정으로 신뢰성을 검증하여 응용프로그램 전환의 시간과 비용 절감에 기여할 수 있다는 것을 확인했다.

키워드 : 오픈소스 DBMS, 응용프로그램 전환, 자동 전환 도구, Java Stored Procedure로 자동 전환

1. 서 론

DBMS (Database Management System) 시장은 오라클, 마이크로소프트, IBM 사 제품 점유율이 90% 수준으로 강세이다. 특히, 오라클사는 DBMS 시장의 약 60% 정도를 점유하고 있다. 그러나 독과점적인 지위를 이용한 고가 라이선스 정책 및 유지보수 비용에 대한 사용자의 불만과 부담이 증가되고, DBMS 다변화에 대한 사용자 요구가 증대되면서, 오픈소스 DBMS 사용에 대한 관심이 커지고 있다[1]. 이로 인해 기존 오라클사 기반으로 운영되고 있는 응용프로그램을 오픈소스 DBMS로 전환하여 비용을 절감하는 사례가 많아지고 있다. 국방부 국방통합정보관리소(메가센터)의 클라우드 표준 DBMS와 정부통합전산센터의 G-클라우드 시스템을 오픈소스 DBMS로 선정하여 구축하였고, KT는 내부 시스템의 일부 시스템에 오픈소스 DBMS를 사용하고 있다.

오라클사 기반 응용프로그램은 ANSI 표준 언어인 SQL과 오라클사에서 독자적으로 제공하는 PL/SQL(Procedural Language extension to SQL)을 사용한다. ANSI 표준 SQL은 관계형

※ 이 논문은 정보통신산업진흥원의 공개SW개발지원사업으로 수행되었음.
† 준 회 원 : 숭실대학교 컴퓨터학과 박사과정
†† 비 회 원 : 솔트웨어(주) 대표이사
††† 종신회원 : 숭실대학교 SW특성화대학원 교수
†††† 종신회원 : 숭실대학교 컴퓨터학부 교수
Manuscript Received : February 20, 2018
Accepted : March 30, 2018
* Corresponding Author : Yongtae Shin(shin@ssu.ac.kr)

DBMS 데이터를 관리하기 위해 설계된 표준 언어이고[2], PL/SQL은 표준 SQL에 절차적인 프로그래밍 언어를 포함하여 만든 비표준 언어이다[3].

오픈소스 DBMS는 ANSI 표준 SQL을 사용하여 응용프로그램을 개발하지만 오라클사의 DBMS 기반의 응용프로그램들을 개발할 때에는 PL/SQL을 사용한다. 이러한 문제로 오라클사의 PL/SQL을 오픈소스 DBMS로 전환 시, PL/SQL의 비호환적 문장으로 인해 많은 수작업을 동반하게 되어 시간 및 비용 측면에서 손실이 크다[4]. 그러므로 PL/SQL을 ANSI 표준 프로그래밍 언어인 Java SP로 자동 전환해주는 도구의 필요성이 요구된다.

따라서 본 논문에서는 PL/SQL을 Java SP로 전환하는 방법을 제안하고, 자동 전환하는 과정을 테스트하여 입력 데이터 PL/SQL과 출력 데이터 Java SP 코드의 동일성을 확인한다.

본 논문은 1장 서론, 2장 응용프로그램 전환 방법과 ANTLR, 3장 응용프로그램 전환 자동화 도구 제안, 4장 자동화 도구의 테스트 실행 과정 및 결과, 그리고 5장 결론으로 구성한다.

2. 응용프로그램 전환 방법과 ANTLR

2.1 응용프로그램 전환 방법

데이터베이스 기반의 응용프로그램 전환 방법에는 응용프로그램 재작성, 수동식 전환 및 자동식 전환이 있다.

응용프로그램을 재작성하여 전환하는 방법은 응용프로그램을 재작성한 후 다시 테스트하기 어렵고 수많은 버그가 발견된다. 또한, 이미 작성된 상당량의 코드를 버리고 재작성하게 됨으로써 많은 시간과 비용을 필요로 한다. 수동으로 전환하는 방법은 처음부터 다시 설계를 수행하고 재개발을 해야 하므로 더 많은 시간과 비용이 요구되며 개발 기간 동안에는 정상적인 업무 운용에 어려움이 있다. 자동으로 전환하는 방법은 전환 도구가 제공하는 범위에 따라 지원이 안 되는 기능이 있을 수 있으나 자동식 전환 방법은 적용이 간편하며 검증되고 신뢰된 아키텍처로 전환하므로 시간과 비용을 절감하고 오류 발생 위험 요소도 감소된다[5].

국외 자동 변환 도구들로는 Ispirer Systems의 SQLWays와 Ciphersoft사의 Composer CipherSoft가 있다. SQLWays는 마이그레이션 도구로 자동으로 다양한 플랫폼과 프로그래밍 언어 사이의 응용 프로그램 전환이 가능하고[6, 7], Composer CipherSoft는 오라클 양식과 PL/SQL 응용 프로그램을 자바 기반의 인터페이스에서 실행되는 Java 및 XML로 전환한다[8, 9].

2.2 ANTLR

파서(Parser)는 문법이 있는 텍스트 파일 형식의 문장에 대한 어휘 분석(Lexical analysis)을 통해 나온 결과로 구문 분석(Syntax analysis)을 수행하여 의미 있는 결과를 만들 수 있다[10]. 어휘 분석에서는 소스(Source) 파일을 읽어 들여 일련의 토큰(Token)을 생성하는 일을 수행한다. 구문 분석에서는 어휘 분석의 결과인 토큰을 받아 소스 파일에 대한 오류를 검사하고 올바른 문자에 대한 구문 구조를 트리 형태로 만든다[11, 12].

파서는 다양한 언어들로 생성될 수 있고 ANTLR (ANother Tool for Language Recognition), Bison, Yacc 등 여러 가지 종류가 있다. ANTLR은 Java, C++, C# 등의 언어로 작성된 문법 표현으로부터 파서를 생성해주는 도구이다[13].

본 논문에서는 Java로 구현되고 검증된 라이브러리로 품질을 보증할 뿐만 아니라, 라이선스 측면에서도 자유로운 오픈소스 소프트웨어인 ANTLR을 선정하여 파싱(Parsing)에 사용했다[14].

3. 응용프로그램 전환 자동화 도구 설계

3.1 자동화 도구

제안하는 응용프로그램 전환을 위한 자동화 도구는 PL/SQL 문법으로 개발된 응용프로그램(저장 프로시저, 함수, 트리거 등)을 ANTLR 기반의 어휘 분석 및 구문 분석 규칙을 적용하여 만들어진 문법 문서를 사용하여 ANSI 표준 프로그래밍 언어인 Java SP로 자동 전환해준다. Fig. 1은 응용프로그램 전환 자동화 도구의 개념도이다.

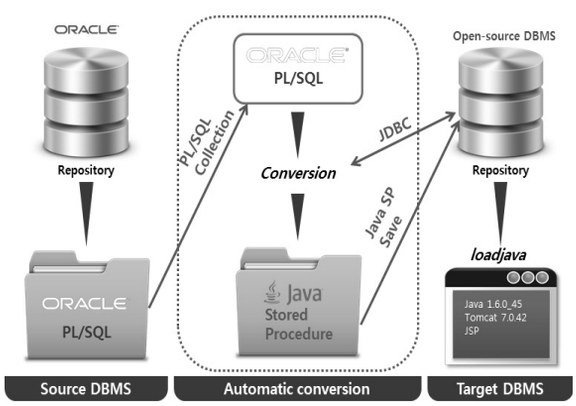


Fig. 1. Conceptual Diagram of Automation Tool for Applications Conversion

Java SP로 자동 전환을 수행하려는 소스 DBMS인 오라클 DBMS에서는 작성된 PL/SQL 문을 수집하여 자료저장소(Repository)에 저장한다. 자동 변환을 원하는 오픈소스 DBMS 중 하나를 목표(Target) DBMS로 설정한 후 자동화 도구로 응용프로그램의 자동 전환을 수행한다. 소스 DBMS의 PL/SQL 문은 자동화 도구에 의해 Java SP로 코드 전환이 수행되며 전환된 코드는 자료저장소에 저장된다. 자동화 도구에 의해 전환된 코드를 확인하려면 목표 DBMS의 자료저장소 확인할 수 있다.

3.2 전환 과정

본 논문에서는 PL/SQL 구문의 어휘 분석과 구문 분석을 체계적으로 하기 위해 ANTLR4 기술을 적용한다[11]. ANTLR

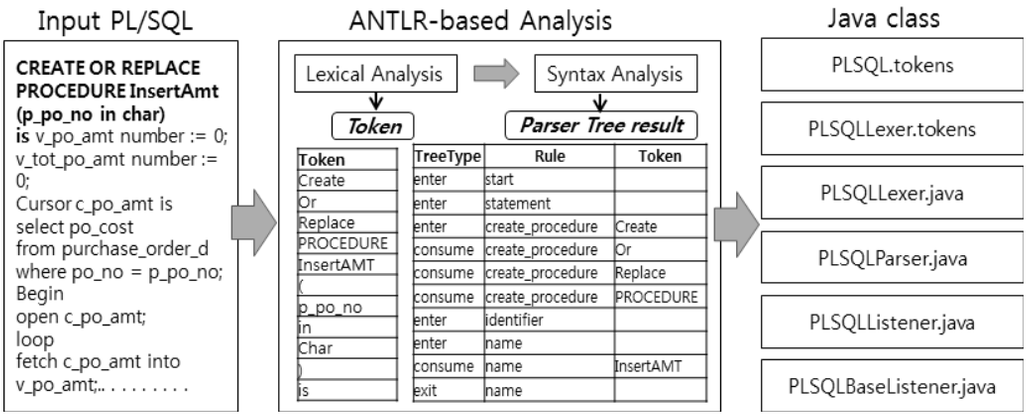


Fig. 2. Process for PL/SQL Statement Conversion

문법에 맞는 PL/SQL의 어휘와 구문 분석 규칙을 설계하였으며 Fig. 2는 PL/SQL 구문 전환 과정이다.

전환하려는 PL/SQL 문이 입력이 되면 ANTLR 기반의 분석 단계인 PL/SQL의 어휘와 구문 분석이 수행된다. PL/SQL의 어휘 분석 규칙은 PL/SQ 구문을 토큰으로 재구성하며, 구문 분석 규칙은 구문 구조를 인식하기 위해 파서 트리 형태의 결과로 정의된다. 따라서 어휘 및 구문 분석 규칙을 ANTLR 텍스트 파일 형태의 문법 문서로 작성한다.

어휘 분석에서는 전환 대상인 PL/SQL의 “CREATE”, “SELECT”, “FUNCTION” 등의 PL/SQL 구문별로 재구성하여 토큰으로 식별하고, 구문 분석에서는 “CREATE FUNCTION”, “CREATE PROCEDURE” 등의 PL/SQL을 ANTLR 문법으로 정의하여 파서 트리 형태의 구문 구조를 인식한다[11].

입력된 PL/SQL 문은 ANTLR로 분석되어 PL/SQL의 토큰과 파서 트리를 자료저장소에 저장한다. 저장된 토큰과 파서 트리에 따라 PL/SQL 구문에 해당하는 Java SP 객체로 구조화하여 정의하고 Java 클래스로 설계한다. 생성된 Java 클래스는 입력된 PL/SQL 문과 의미가 같은 Java SP 코드로 자동 생성된다. 즉, ANTLR 기반 어휘 및 구문 분석 규칙으로 PL/SQL 구문 분석 파서 트리 결과인 Java 클래스가 정의된다.

ANTLR 기반으로 분석된 PL/SQL의 토큰과 파서 트리는 자료저장소에 저장하고 전환 대상 PL/SQL 분석 결과를 행별로 인식하여 Java 코드로 변환하기 위한 전환 규칙을 적용한다. Java SP 전환 규칙은 분석 규칙의 결과인 파서 트리에서 Java 클래스를 호출할 수 있도록 파서 트리를 읽어 PL/SQL 문을 트리 단계의 반복 방식으로 해당 PL/SQL 문에 해당하는 Java 객체를 호출하는 방식이다. 또한 전환 규칙은 PL/SQL 문에 해당하는 Java 객체를 호출하는 코드로 표현하며 이 과정에서 Java 객체는 PL/SQL 구문 분석 규칙을 정의한 구문에 해당하는 모든 객체를 정의한다. 정의된 Java PL/SQL 객체는 Fig. 3과 같은 Java SP 구조에 적합하게 블록 단위로 Java 코드를 생성한다.

Java SP 코드 블록은 8개의 단위로 정의되며 각 블록은 해당하는 Java SP 소스 코드를 객체로 정의하여 분석 결과에 따라 Java 객체를 호출하여 Java SP 코드 블록 위치에 적합한 Java 코드를 삽입한다. 즉, 분석된 파서 트리가 Java SP 전환 규칙에 따라 “CREATE PROCEDURE” 코드로 정의되면 Java SP 코드 블록 중 “CREATE PROCEDURE” 코드의 객체인 ‘Java Stored Procedure Block’을 호출하여 소스 코드 블록에 맞는 Java SP 코드로 완성된다.

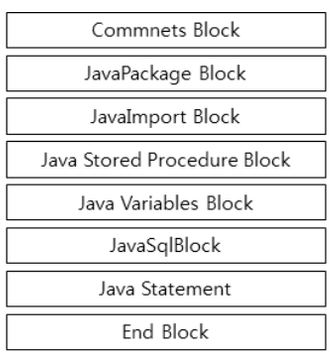


Fig. 3. Java SP Code Block

3.3 전환 도구의 구성요소

전환 도구의 구성요소는 Table 1과 같이 소스 관리(Source Manager), 전환 관리(Conversion Manager), 목표 관리(Target Manager) 그리고 어댑터(Adaptor)로 구성된다.

Table 1. Components of Conversion Tool

Component	Description
Source Manager	▪ PL/SQL source management
Conversion Manager	▪ PL/SQL source conversion
Target Manager	▪ Converted Java management - Java Compile - Java class registered in the DBMS
Adaptor	▪ RDBMS supported DBMS Adaptor

구성요소 관리에는 소스 관리, 전환 관리, 목표 관리가 있으며 어댑터는 유틸리티(Utility) 어댑터, DBMS 어댑터, DBMS 확장 등을 지원한다. 유틸리티 어댑터는 DBMS에 종속적인

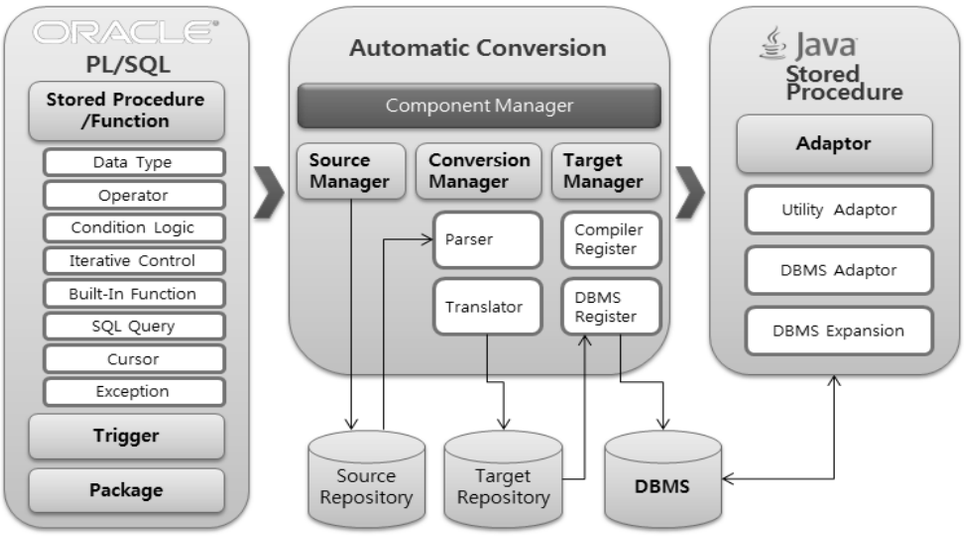


Fig. 4. Connection of Components

내장 함수(Built-in function)를 전환된 Java에서 처리 가능하도록 하고, DBMS 어댑터는 RDBMS에 따른 연결 문자열(Connect String), JDBC 드라이버 등을 전환된 Java에서 처리 가능하도록 한다.

소스 DBMS의 PL/SQL을 오픈소스 DBMS의 Java SP로 전환하기 위해 Java와 JDBC를 사용하여 플랫폼과 DB에 독립적으로 구성한다. 또한, PL/SQL 구문의 효과적인 분석 및 전환을 적용하기 위해 Fig. 4와 같이 전환 도구의 구성요소들로 설계한다.

소스 관리는 오라클사에서 사용하는 저장 프로시저/함수(Stored Procedure/Function), 트리거(Trigger), 패키지(Package) 등의 PL/SQL을 소스 자료저장소에 저장하여 조회/삽입/삭제/수정 등의 작업을 수행할 수 있도록 관리한다.

전환 관리는 소스 자료저장소에 저장된 PL/SQL을 Java SP로 전환하여 그 결과를 목표 자료저장소에 저장하여 관리한다. 즉, SQL Query, 커서(Cursor), 예외처리(Exception), 트리거, 패키지 등에서 Java 코드로 전환된 결과를 저장한다. 이 때, 파싱은 ANTLR로 어휘 및 구문 분석을 하며, 분석 결과인 파서 트리는 Java 코드로 전환된다.

목표 관리는 전환된 Java 코드를 컴파일하고 요청에 따라 목표 DBMS로 적재(Loading)한다.

3.4 주요 기능

본 논문에서 제안하는 응용프로그램 전환 자동화 도구의 기능은 Table 2와 같다.

응용프로그램 전환 자동화 도구는 PL/SQL 구문의 수집 및 조회, 수집된 PL/SQL 파싱, 자료저장소에 저장된 파싱된 결과를 검토하여 문법 분석, 분석된 문법을 Java 코드로 전환한 결과와 로그 저장 및 조회, 전환된 Java에서 사용하는 테이블과 CRUD 조회, 전환할 PL/SQL과 전환된 Java 코드 조회 및 전환율 조회, PL/SQL과 Java에서 사용하는 테이블과 CRUD 비교 등의 기능을 제공한다.

Table 2. Key Functions

Function	Description
PL/SQL collection/inquiry	▪ PL/SQL collection and inquiry
PL/SQL Parsing	▪ Parsing the collected PL/SQL and storing it in Repository
PL/SQL syntax analysis	▪ Reviewing the result of parsing and checking the syntax
PL/SQL conversion analysis	▪ Check if the code passing the syntax analysis can be converted to Java code
PL/SQL CRUD inquiry	▪ Inquiring into the table name used in PL/SQL and CRUD of the type of use - C(Create), R(Read), U(Update), D(Delete)
PL/SQL conversion	▪ Converting the analyzed code to a Java code and storing the result and log
Inquiry into the result /log of conversion	▪ Inquiring into the result/log of the conversion to a Java code
Java CRUD inquiry	▪ Inquiring into the table used in the converted Java code and the type of use
Inquiry before/after conversion	▪ Inquiring before conversion (PL/SQL) and after conversion (Java code)
Inquiry into the conversion rate	▪ Inquiring into conversion rate of PL/SQL to Java code
CRUD comparison	▪ Comparing the table used in the PL/SQL environment and Java code environment and the type of use

4. 응용프로그램 전환 자동화 도구 테스트

4.1 테스트 환경

응용프로그램 전환 자동화 도구의 테스트 환경은 Table 3과 같다.

Table 3. Test Environment

Item	Description
OS	▪ Windows 7(64bits)
Source DBMS	▪ Oracle 12CR1
Languages	▪ Visual Studio 2010 ▪ ANTLR version 4.1 ▪ Apache Tomcat 7.0.42
Target DBMS	▪ Cubrid V9.1.0.0212

운영체제는 윈도우 XP 이상 버전에서 수행 가능하고, PL/SQL 수집은 윈도우와 Java 버전과 관련이 없다. 목표 DBMS는 자료저장소를 지원하고, 목표 DBMS인 큐브리드는 자체 Java를 가지고 있지 않으므로, 서버에 설정된 Java VM을 경로에 기술하여 사용했다.

4.2 테스트 실행

테스트는 Fig. 5와 같이 소스 DBMS의 DB명은 PDBORCL, 목표 DBMS의 DB명은 PurchaseOrder로 구축하여 자동화 도구로 응용프로그램 전환을 수행한 후, 웹브라우저에서 PurchaseOrder의 데이터 조회/입력/수정 실행 결과를 확인했다.

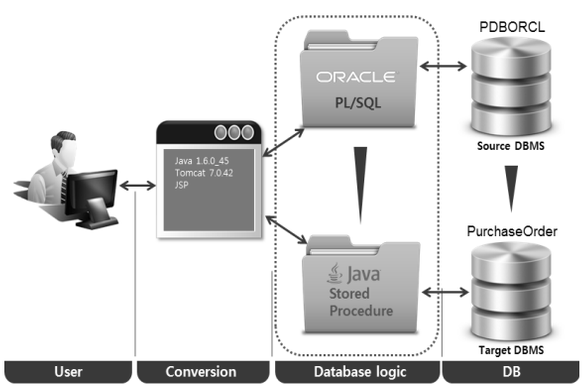


Fig. 5. Test Environment Configuration

1) PDBORCL 구축

구매발주 업무를 사용하여 PDBORCL을 구축했으며, 구매발주 업무 설계를 위한 논리 데이터 모델은 Fig. 6, 물리 데이터 모델은 Fig. 7과 같다.

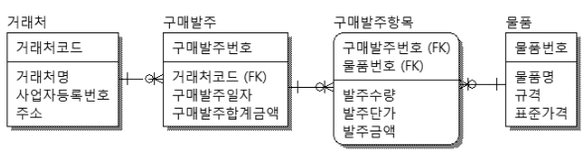


Fig. 6. Logical Data Model

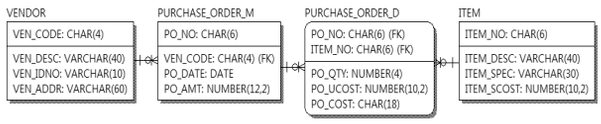


Fig. 7. Physical Data Model

PDBORCL 테이블들에 초기 데이터로 Table 4와 같은 데이터를 입력했다.

Table 4. Input Data

Table	Data
VENDOR	▪ “V123”,”대한민국”,”1234567890”,”서울특별시” ▪ “V234”,”숭실대학교”,”2345678901”,”서울특별시 동작구 상도동”
ITEM	▪ “A12345”, ”컴퓨터”, ”123*345”, ”1000000.00” ▪ “B12345”, ”모니터”, ”111*222”, ”500000.00” ▪ “C12345”, ”키보드”, ”222*333”, ”100000.00” ▪ “D12345”, ”마우스”, ”333*444”, ”50000.00”
PURCHASE_ORDER_M	▪ PO_NO, VEN_CODE, PO_DATE : Inserted in the input screen ▪ PO_AMT : Sum total of purchased order amount of ITEM_NO and PO_NO
PURCHASE_ORDER_D	▪ PO_QTY, PO_UCOST : Inserted in the input screen ▪ PO_COST : PO_QTY *PO_UCOST

구매발주 테이블과 구매발주항목 테이블의 일부 속성은 입력화면에서 삽입했다. 그리고 구매발주 테이블의 구매발주 합계금액(PO_AMT) 속성은 PL/SQL의 저장 프로시저로 작성했고, 구매발주항목 테이블의 발주금액(PO_COST) 속성은 PL/SQL의 트리거로 작성하여 처리했다.

구매발주 테이블의 구매발주합계금액 속성에 대한 저장 프로시저인 InsertAmt는 물품 테이블의 물품번호(ITEM_NO)와 구매발주 테이블의 구매발주번호(PO_NO)별 구매발주항목이 존재하면 구매발주합계금액(같은 발주번호의 발주금액 합계)을 변경한다.

```
CREATE OR REPLACE PROCEDURE InsertAmt (p_po_no in
char)
 is v_po_amt number := 0;
    v_tot_po_amt number := 0;
    Cursor c_po_amt is
      select po_cost
      from purchase_order_d
      where po_no = p_po_no;
 Begin
    open c_po_amt;
    loop
      fetch c_po_amt into v_po_amt;
      exit when c_po_amt%NOTFOUND;
      v_tot_po_amt :=v_tot_po_amt + v_po_amt;
    end loop;
```

```
    close c_po_amt;
    update PURCHASE_ORDER_M
    set po_amt = v_tot_po_amt
    where po_no = p_po_no;
end InsertAmt;
```

구매발주항목 테이블의 발주금액 속성에 대한 트리거인 UpdateCost는 구매발주항목에 입력되는 발주수량(PO_QTY)과 발주단가(PO_UCOST)의 곱으로 발주금액을 변경한다.

```
CREATE OR REPLACE TRIGGER UpdateCost
  BEFORE INSERT OR DELETE OR UPDATE ON
PURCHASE_ORDER_D
  FOR EACH ROW
BEGIN
  :new.po_cost := :new.po_qty * :new.po_ucost;
END UpdateCost;
```

2) PurchaseOrder 구축

PurchaseOrder구축을 위한 데이터 모델은 Fig. 6과 Fig. 7을 사용한다.

3) 응용프로그램 전환 자동화 도구 실행

Fig. 8과 같이 PDBORCL에서 PurchaseOrder로 Converting한 후, Loadjava를 수행한다.

Converting은 PDBORCL에서 PL/SQL로 작성된 저장 프로시저와 트리거를 SAM File로 수집하여 자료저장소에 저장한 후 Java 코드로 전환하는 과정이다. Loadjava는 전환된 결과를 목표 DBMS에서 지원하는 Java 버전으로 컴파일한 후에 목표 DBMS로 적재하는 과정이다.

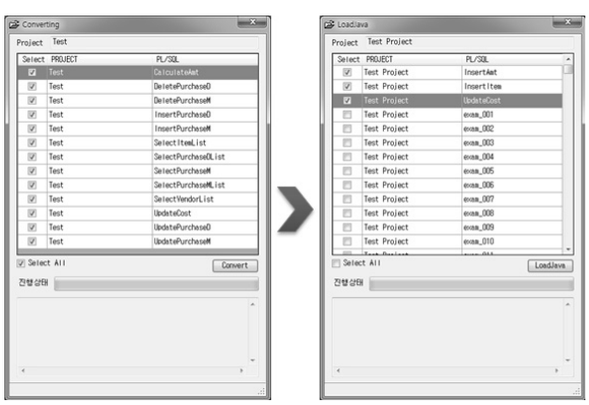


Fig. 8. Loadjava Execution after Converting

4) 전환된 Java 코드

PL/SQL 구문의 저장 프로시저 InsertAmt는 Java SP 코드 블록에서 “CREATE PROCEDURE” 코드 객체인 “Java Stored Procedure Block”을 호출하여 Java 코드로 InsertAmt.java 파일을 생성한다.

```
public class InsertAmt {
    public static void InsertAmt(
    String p_po_no) throws SQLException
    {
     BigDecimal v_po_amt = new BigDecimal(0);
     BigDecimal v_tot_po_amt = new BigDecimal(0);
     Connection _conn =
DriverManager.getConnection("jdbc:default:connection");
     PreparedStatement _pstmnt    = null;
     String                _sqlQuery = null;
     boolean _isFound = false;
     int rowsUpdated  = 0;
     String _str_c_po_amt = "select po_cost from
purchase_order_d where po_no = ?";
     _pstmnt = _conn.prepareStatement(_str_c_po_amt);
     _pstmnt.setString(1,p_po_no);
     ResultSet _rs_c_po_amt = _pstmnt.executeQuery();
     while(true) {
          _isFound = _rs_c_po_amt.next();
          if(_isFound) { v_po_amt =
_rs_c_po_amt.getBigDecimal(1); }
          if(!_isFound) { break; }
          v_tot_po_amt = v_tot_po_amt.add(v_po_amt);
     }
     _pstmnt.close();
     _sqlQuery = "update PURCHASE_ORDER_M
                  set po_amt = ? where po_no = ?";
     _pstmnt = _conn.prepareStatement(_sqlQuery);
     _pstmnt.setBigDecimal(1,v_tot_po_amt);
     _pstmnt.setString(2,p_po_no);
     rowsUpdated = _pstmnt.executeUpdate();
    }
}
```

PL/SQL 구문의 트리거 UpdateCost는 전환된 Java 코드로 UpdateCost.java 파일을 생성한다.

```
public class UpdateCost {
    public static void UpdateCost
      (String _old_po_no,
       String[] _new_po_no,
       BigDecimal _old_po_seq,
       BigDecimal[] _new_po_seq,
       String _old_item_no,
       String[] _new_item_no,
       BigDecimal _old_po_qty,
       BigDecimal[] _new_po_qty,
       BigDecimal _old_po_ucost,
       BigDecimal[] _new_po_ucost,
       BigDecimal _old_po_cost,
       BigDecimal[] _new_po_cost
      )throws SQLException {
      _new_po_cost[0] =
```

```
            _new_po_qty[0].multiply(_new_po_ucost[0]);
        Connection _conn =
DriverManager.getConnection("jdbc:default:connection");
        PreparedStatement _pstmnt  = null;
        String                _sqlQuery = null;
        _sqlQuery = "update PURCHASE_ORDER_D"
                  + "          set ITEM_NO = ?"
                  + "              ,PO_QTY = ?"
                  + "              ,PO_UCOST = ?"
                  + "              ,PO_COST = ?"
                  + "       where PO_NO = ?"
                  + "         and PO_SEQ = ?";
        _pstmnt = _conn.prepareStatement(_sqlQuery);
        _pstmnt.setString(1,_new_item_no[0]);
        _pstmnt.setBigDecimal(2,_new_po_qty[0]);
        _pstmnt.setBigDecimal(3,_new_po_ucost[0]);
        _pstmnt.setBigDecimal(4,_new_po_cost[0]);
        _pstmnt.setString(5,_old_po_no);
        _pstmnt.setBigDecimal(6,_old_po_seq);
        _pstmnt.executeUpdate();
        _pstmnt.close();
        _conn.close();
    }
}
```

트리거는 Java를 지원하지 않으므로, 트리거의 전환된 Java 코드 'Java Class'를 저장 프로시저로 등록하여 실행했다. 즉, 트리거에서 저장 프로시저를 호출하고, 호출된 저장 프로시저에 'Java Class'를 선언한다.

전환된 트리거 UpdateCost.class를 저장 프로시저 proc_UpdateCost로 등록하고, 트리거 UpdateCost에서 저장 프로시저 proc_UpdateCost를 호출하여 PL/SQL 구문의 트리거에 대응되는 Java 코드 전환을 처리했다.

```
CREATE TRIGGER UpdateCost
  AFTER INSERT ON PURCHASE_ORDER_D
  EXECUTE call proc_UpdateCost(obj.po_no, obj.po_no,
obj.po_seq, obj.po_seq, obj.item_no, obj.item_no, obj.po_qty,
obj.po_qty, obj.po_ucost, obj.po_ucost, obj.po_cost, obj.po_cost);
```

```
CREATE Procedure InsertAmt(p_po_no in char)
    AS LANGUAGE JAVA
    NAME 'InsertAmt.InsertAmt(java.lang.String)';
```

```
CREATE PROCEDURE proc_UpdateCost(
 OLD_PO_NO IN VARCHAR, NEW_PO_NO IN OUT VARCHAR,
 OLD_PO_SEQ IN NUMERIC, NEW_PO_SEQ IN OUT NUMERIC,
 OLD_ITEM_NO IN VARCHAR, NEW_ITEM_NO IN OUT
VARCHAR,
 OLD_PO_QTY IN NUMERIC, NEW_PO_QTY IN OUT NUMERIC,
 OLD_PO_CUOST IN NUMERIC, NEW_PO_UCOST IN OUT NUMERIC,
 OLD_PO_COST IN NUMERIC, NEW_PO_COST IN OUT NUMERIC)
 AS LANGUAGE JAVA NAME
 'UpdateCost.UpdateCost(java.lang.String, java.lang.String[],
java.math.BigDecimal, java.math.BigDecimal[], java.lang.String,
java.lang.String[], java.math.BigDecimal,
java.math.BigDecimal[], java.math.BigDecimal,
java.math.BigDecimal[], java.math.BigDecimal,
java.math.BigDecimal[])';
```

5) 결과 확인

웹브라우저의 로컬 웹페이지에서 테스트 결과를 확인하기 위해 DBMS, JDBC_CLASS, DB_URL, DB_USER, DB_PASSWORD를 Java SP를 사용하는 DB로 변경한 후, 목표 DBMS의 properties를 %CATALINA_HOME%\webapps\PurchaseOrder\config로 수정했다.

Fig. 9는 PurchaseOrder에서 거래처, 물품, 구매발주 테이블의 데이터를 조회하는 화면이다.

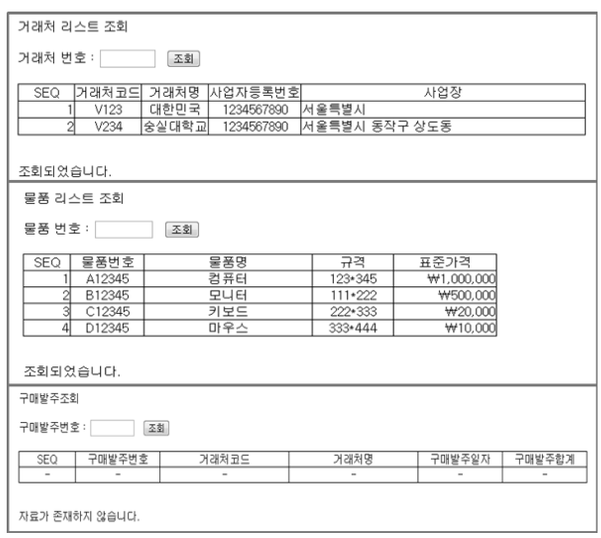


Fig. 9. Data Query Screen

Fig. 10과 같이 구매발주 합계와 발주금액의 데이터의 정확성을 확인하기 위해 구매발주입력 화면에서 구매발주를 위한 데이터를 입력했다.(입력 화면에서 조회는 되지 않는다.)

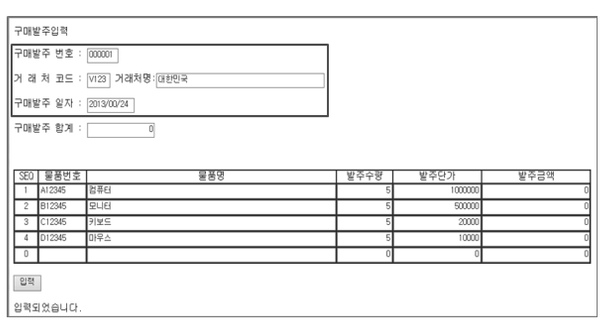


Fig. 10. Purchase-order-entry Screen

구매발주 테이블에서 수정된 구매발주합계금액(PO_ AMT)은 Fig. 11과 같이 변경된 값으로 확인했다.

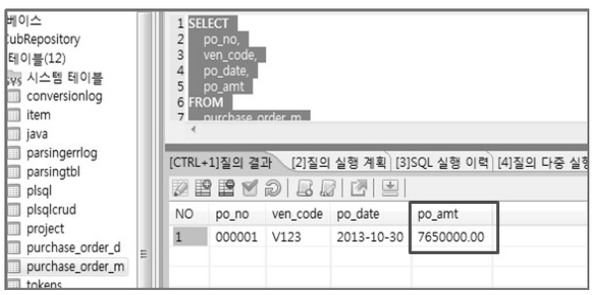


Fig. 11. Changed po__amt

구매발주항목 테이블의 수정된 발주금액(PO_COST)은 Fig. 12와 같이 변경된 값으로 확인했다.

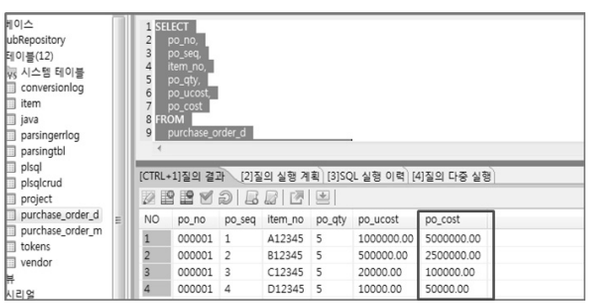


Fig. 12. Changed po__cost

구매발주조회 화면에서 구매발주번호별 항목을 조회하여 수정된 내용이 반영된 구매발주합계를 Fig. 13과 같이 확인했다.

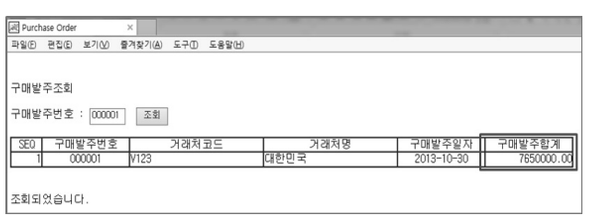


Fig. 13. Purchase-order-inquiry Screen

4.3 자동화 도구 비교

응용프로그램 전환 자동화 도구 테스트 실행으로 전환할 PL/SQL 구문과 전환된 Java SP의 동일한 데이터 검증으로 신뢰적인 전환 결과를 확인했다.

제안하는 자동화 도구는 최적화된 변환을 제공하기 위해 ANTLR 기반 어휘 및 구문 분석을 하여 구조화된 파서 트리의 결과로 Java 코드로 변환한다. SQLWays와 Composer Ciphersoft의 어휘 및 구분 분석 방식은 알려지지 않았다.

제안하는 자동화 도구는 최적화된 전환을 위해 일관성 있고 효율적으로 코드를 변환하여 제공한다. SQLWays의 자동화 작업은 지능적이고 유지보수 가능한 코드를 생성하나 다양한 기술 플랫폼에 대한 전문 지식이 필요하고 복잡하며 위험이 높다[7]. Composer Ciphersoft의 자동화 작업은 개방된 플랫폼으로 이동 가능함으로써 원시 Java/XML 코드 생성이 가능하다[9].

모든 도구가 전환 과정에서의 에러 처리는 가능하나, 가격에서는 제안하는 자동화 도구는 오픈소스 소프트웨어인 반면 SQLWays는 유연한 가격 책정 전략을 제공하고 있으며[7], Composer Ciphersoft는 라이센스 제품 또는 제공되는 서비스로 사용할 수 있다[9].

제안하는 자동화 도구와 기존 전환 방법의 비교는 Tabel 5와 같다.

제안하는 자동화 도구가 기존 전환 방법보다 간편한 수행으로 사용의 편리성을 제공하고, 전환 과정에서 발생하는 오류는 수정 가능하고, ANTLR을 사용하여 신속한 어휘 분석과 구문 분석 과정으로 처리 시간 단축 및 비용 감소 가능성을 확인했다.

5. 결 론

본 논문에서는 PL/SQL로 개발된 응용프로그램을 ANTLR 기반 분석 기술을 이용하여 ANSI 표준 기반 오픈소스 DBMS의 Java SP로 전환하는 응용프로그램 전환 자동화 도구를 제안하고 테스트를 수행하여 응용프로그램이 자동 전환됨을 확인했다. 테스트는 오픈소스 DBMS인 큐브리드로 수행했다. 실행 과정에서 트리거의 자동 전환에서 문제점이 발생되었으나, 이는 큐브리드의 엔진 문제로 Java SP를 수정함으로써

Table 5. Tools Comparison

Features	Proposed Automation Tool	SQLWays	Composer Ciphersoft
Lexical & Syntax analysis using ANTLR	Use (quick & accurate analysis)	Null	Null
Optimized Conversion	Consistent and Efficient Code	Intelligent and Maintainable Code	Native Java/XML Code
Errors in Conversion Process	Modifiable	Modifiable	Modifiable
Complexity	Low	High	Medium
Cost	Free	Flexible Price	Licensed Product

완전하게 전환됨을 보였다. 따라서 제안하는 응용프로그램 전환 자동화 도구는 PL/SQL 응용프로그램을 오픈소스 DBMS의 응용프로그램으로 전환 시 발생 가능한 위험요소를 제거했고, 자동 전환 수행으로 기존 전환 방법보다 시간, 노력 그리고 비용 절감이 가능하다는 것을 보였다. 테스트에서 적용한 큐브리드 외에 PostgreSQL, MariaDB 등의 오픈소스 DBMS로 전환하는 경우에도 동일하게 적용이 가능하다.

본 논문에서 제안한 응용프로그램 전환 자동화 도구는 오픈소스로 공개하고 다양한 기술지원, 자료 한글화, 커뮤니티 활성화 등을 제공하고 있다. 라이선스 분쟁 또는 신속하지 못한 기술지원 서비스 등과 같은 불만으로 인해 사용자들은 호환성이나 안정성에 문제가 없는 오픈소스 DBMS에 대한 관심이 점차 확대되고 있으며, 이는 공공기관, 기업 및 학교 등 다양한 분야로 확산되고 있으며 제품의 시장 점유율도 증가하고 있다. 따라서 본 논문에서 제시하는 응용프로그램 전환 자동화 도구를 활용하여 실제 업무에 적용한다면 DBMS 다변화에 큰 영향을 줄 수 있을 것이다.

향후에는 오픈소스 DBMS의 활용을 높일 수 있는 추가 기능과 응용프로그램 전환 자동화 도구의 성능을 높일 수 있는 연구가 추가적으로 요구된다.

References

[1] 2014 Year of the database industry market analysis report, Korea Database Agency, Dec. 2014. (in Korean)

[2] ANSI/ISO/IEC 9075-14:2006, "Database languages – SQL - Part14: XML-Related Specifications (SQL /XML)," *International Organization for Standardization/International Electrotechnical Commission*, Mar. 2011.

[3] Sheila Moore, *Oracle Database PL/SQL Language Reference*, 11g Release2, Oracle, 2013.

[4] Open Source Software company handbook, *Korea OSS Promotion Forum*, 2014. (in Korean)

[5] J. k. Lee, A Study of Modeling Automatic Translator from PL/SQL to Java Stored Procedure Based on ANTLR, *Soongsil University*, Jun. 2014. (in Korean)

[6] SQLWays's Application conversion [Internet], http://www.-ispirer.com/application-conversion

[7] Ispirer Systems Ltd, "Oracle to MySQL Migration," *White Paper*, Mar. 2009.

[8] Composer CipherSoft [Internet], http://composer technologies.com/products/composer-ciphersoft

[9] OpenText Composer CipherSoft [Internet], https://www.opentext.com/what-we-do/products/specialty-technologies/opentext-composer/opentext-composer-ciphersoft

[10] Y. Zhao, T. Wang, X. Ni, X. Wang, and Z. Xie, "Syntactic Representation Transformation in Operator Design Method Based on ANTLR Tool," *Computer and Information Technology(CIT), 2012 IEEE 12th International Conference on.*, pp.115-118, Oct. 2012.

[11] Terence Parr, *Language Implementation Patterns*, The Pragmatic Bookshelf, 2010.

[12] Terence Parr, *The Definitive ANTLR 4 Reference*, 2nd Edition, The Pragmatic Bookshelf, 2013.

[13] T. J. Parr and R. W. Quong, "ANTLR: A predicated -LL(k) Parser Generator," *Software-Practice and Experience*, Vol.25(7), pp.789-810, 1995.

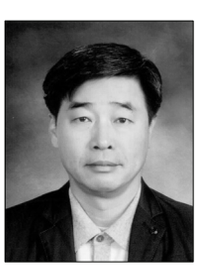
[14] D. Cao and D. Bai, "Design and implementation for SQL parser based on ANTLR," *2010 2nd International Conference on Computer Engineering and Technology*, Vol.4, pp.276-279, 2010.

지 정 은

https://orcid.org/0000-0002-2560-5909
e-mail : belouga@ssu.ac.kr
2001년 수원대학교 전자계산학과(학사)
2009년 숭실대학교 정보과학대학원(석사)
2009년~현 재 숭실대학교 컴퓨터학과 박사과정
관심분야 : IoT, 네트워크 보안, 웹 보안, 데이터베이스

이 정 근

https://orcid.org/0000-0001-5327-7896
e-mail : jklee@saltware.co.kr
1985년 동국대학교 컴퓨터공학과(학사)
2004년 연세대학교 공학대학원(석사)
2014년 숭실대학교 IT정책학과(박사)
2003년~현 재 솔트웨어(주) 대표이사
관심분야 : 클라우드 컴퓨팅, IT융합, SOA 아키텍처



최 용 락

https://orcid.org/0000-00002-3867-3425
e-mail : ylchoi58@ssu.ac.kr
1985년 숭실대학교 전자계산학과(학사)
1996년 숭실대학교 정보과학대학원(석사)
2002년 숭실대학교 컴퓨터학과(박사)
2012년~현 재 숭실대학교 SW특성화대학원 교수
관심분야 : 데이터베이스, 전략정보기획, SW 엔지니어링

신 용 태

https://orcid.org/0000-0002-1199-1845

e-mail : shin@ssu.ac.kr

1985년 한양대학교 산업공학과(학사)

1990년 Univ. of Iowa 컴퓨터학과(석사)

1994년 Univ. of Iowa 컴퓨터학과(박사)

1995년~현 재 숭실대학교 컴퓨터학부 교수

관심분야 : IoT, 정보보호, 컨텐츠보안, 모바일인터넷, 차세대인터넷기술