원천 시스템 환경을 고려한 데이터 추출 방식의 비교 및 Index DB를 이용한 추출 방식의 구현*

- ㅅ 은행 사례를 중심으로 -

김 기 운**

A Comparison of Data Extraction Techniques and an Implementation of Data Extraction Technique using Index DB*

- S Bank Case -

Giun Kim**

Abstract

Previous research on data extraction and integration for data warehousing has concentrated mainly on the relational DBMS or partly on the object-oriented DBMS. Mostly, it describes issues related with the change data (deltas) capture and the incremental update by using the triggering technique of active database systems. But, little attention has been paid to data extraction approaches from other types of source systems like hierarchical DBMS, etc. and from source systems without triggering capability.

This paper argues, from the practical point of view, that we need to consider not only the types of information sources and capabilities of ETT tools but also other factors of source systems such as operational characteristics (i.e., whether they support DBMS log, user log or no log, timestamp), and DBMS characteristics (i.e., whether they have the triggering capability or not, etc), in order to find out appropriate data extraction techniques that could be applied to different source systems. Having applied several different data extraction techniques (e.g., DBMS log, user log, triggering, timestamp-based extraction, file comparison) to S bank's source systems (e.g., IMS, DB2, ORACLE, and SAM file), we discovered that data extraction techniques available in a commercial ETT tool do not completely support data extraction from the DBMS log of IMS system. For such IMS systems, a new data extraction technique is proposed which first creates Index database and then updates the data warehouse using the index database. We illustrates this technique using an example application.

Keyword : Data Extraction Techniques, ETT tool, Log Extraction, Triggering, Extraction, Index Database

논문접수일 : 2001년 9월 28일 논문게재확정일 : 2003년 5월 31일

* 본 연구는 2003년도 경인여자대학 교내연구지원 연구비에 의해 수행되었음.

** 경인여자대학 컴퓨터정보기술학부 인터넷비지니스

1. 서 론

데이터 웨어하우스는 데이터를 통합하고 분석적인 정보를 산출하여 기업의 의사결정을 지원하는 중요한 정보기술 기반이다. 그러나 데이터 웨어하우스를 구축하는 일은 많은 시간과 비용이 요구되며 구축 작업이 용이하지 않다. 그 이유 중의 하나는 데이터 추출 작업의 복잡성에 기인한다. 즉, 데이터의 원천과 데이터 웨어하우스 내의 테이블과의 대응이 다-대-다 관계를 갖고 있어서, 운영 환경의 데이터가 주제 중심의 데이터 웨어하우스로 옮겨지는 과정은 다수의 임시 파일과 프로그램들을 통하여 다-대-다의 대응 관계를 형성하기 때문이다. 따라서 Inmon(1992)과 Berson et al.(2000)이 지적하였듯이, 원천 시스템에서 데이터를 데이터 웨어하우스로 유입하는 추출, 정제, 적재 등의 작업은 상당한 시간이 소요된다. 평균적으로 데이터 웨어하우스를 구축하는 노력의 80%가 이들 업무에 투입된다(Hufford, 1996).

일반적으로 원천 시스템으로부터 데이터를 추출하여 목표 시스템으로 데이터를 전달하는 과정을 'ETT(Extraction/Transmission/Transformation)'라고 한다. 이러한 ETT 과정을 지원하는 대표적인 2가지 방식은, 전통적인 코딩 기법에 의해 프로그램을 개발하는 것과 추출(ETT) 도구를 사용하여 ETT 과정을 자동화하는 것이다(Chaudhuri & Dayal, 1997). 데이터 웨어하우스 구축을 위한 ETT 도구는 ETT 과정과 관련한 여러 가지 기능을 갖고 있지만 필요한 기능이 미비한 경우도 있다(김기운, 서용무, 2000). 또한, 필요한 기능이 있다고 하여도 ETT 도구들이 적용될 시스템 운용 방식 등의 시스템의 기술적 특성에 따라서 실제적으로 구현이 용이하지 않은 경우가 있어, 변환 프로그램을 추가로 작성하는 등 다른 대안을 함께 고려할 필요가 있다.

따라서 본 연구에서는 ㅅ 은행의 시스템 운영 방식 및 DBMS 환경 등의 시스템의 기술적 특성을 고려하여 ETT 도구인 PRISM을 이용한 데이터 추출 방식을 비교·검토하고 이와 관련하여 발생하는 이슈를 살펴보고, 그들에 대한 해결 방안을 찾아보고자 한다.

본 연구의 구성은 다음과 같다. 제 2장에서는 데이터 추출을 위한 정보원천 유형의 분류와 추출 방식에 대한 기존 연구를 고찰하고, 본 연구의 연구 이슈를 제기한다. 제 3장에서는 기존에 제시된 추출 방식을 ㅅ 은행의 시스템 환경에 적용하여 비교·검토한다. 제 4장에서는 제 3장에서 검토한 내용을 기반으로 발생한 이슈에 대한 새로운 추출 방식을 제안하고 그 방식에 대한 구현 예제를 통하여 제안된 방식의 적용 가능성을 살펴본다. 제 5장에서는 본 연구를 정리한다.

2. 관련 연구 고찰 및 연구 이슈

2.1 데이터 추출(Extraction) 방식

ETT와 관련한 일련의 활동들에 대한 용어의 표현은 학자에 따라 약간 다른 용어를 사용하고 있으나 그 의미는 일맥 상통하고 있다. Kelly(1994)는 데이터 추출(extract), 데이터 매핑(mapping), 데이터 재충전(refresh) 혹은 갱신(update), 데이터 전송(propagation), 데이터 변환, 데이터 적재(load)의 과정으로 설명하고 있다. Haisten(1995)은 데이터 추출(extract), 데이터 전송(transport), 데이터 매핑, 데이터 정제(clean), 변환(transform), 데이터를 적재(load)하고 저장(population)하는 활동으로 설명하고 있다. Hufford(1996)는 데이터 추출(extract), 데이터 여과(filter), 데이터 품질 확인(validate), 다른 추출 데이터와 병합(merge), 데이터 집계(aggregate), 데이터 적재(load), 오래된 데이터를 다른 저장 공간에 보관(archive)하는 단계로 설명한다. Devlin(1997)은 데이터 추출(capture), 데이터 전달(transfer), 데이터의 변환(transform), 추출된 데이터를 목표 시스템에 저장(apply)하는 과정으로 설명한다.

본 연구에서는 ETT 과정에서 데이터 추출과 관

련된 내용을 중심으로 살펴본다.

Widom(1995)은 데이터 추출의 원천 유형을 ① 변경된 데이터 부분을 Log로부터 추출할 수 있는 원천(Logged Source), ② Trigger 혹은 Active DB (Widom & Ceri, 1996) 기능을 갖춘 원천으로부터 변화를 자동적으로 통지해주는 협동적 원천(Co-operative Source), ③ 데이터 웨어하우스의 모니터(monitor)가 질의를 제기하여 추출할 수 있는 원천(Queryable Source), ④ 기존 시스템에서 연속적인 스냅샷을 비교하여 변화를 감지하여 추출하는 스냅샷[1] 원천(Snapshot Source)의 4가지 유형으로 분류하고 있다(<표 1> 참조).

Devlin(1997)은 Widom(1995)의 연구를 세분화하고 확장하여 원천 데이터를 추출하기 위한 6가지의 추출 방식을 <표 1>과 같이 제시하고, 이것을 크게 점진적(incremental)과 정적(static) 추출로 나누고 있다.

〈표 1〉 데이터 추출 방식관련 연구

Widom(1995)의 분류	Devlin(1997)의 분류		
Log	점진적 추출	즉시 추출	User Log
Triggering			DBMS Log
			Triggering
Queryable		지연 추출	TimeStamp
			File 비교
Snapshot	정적추출		Snapshot

점진적 추출에는 변경을 일으키는 사건이 발생한 후 원천 데이터의 변화를 즉시(immediate) 추출하는 방식과 일정한 시간이 지난 후(delayed)에 추출하는 방식이 있다. 즉시 추출에는 3가지 기법이 있다. 첫째는 User Log 추출 방식이다. 이 방식은 추출 기능이 운영 시스템의 애플리케이션에 삽입되어 있어서 각 애플리케이션마다 커미트(commit)하기 전에 User Log에 운영 시스템의 변경 데이터를 연속적으로 제공한다. 그러나 이 방식은 기존의 애플리케이션에 많은 영향을 주며, 개발과 유지보수 관점에서 볼 때 비용이 많이 든다. 둘째는 추출의 기능이 애플리케이션으로부터 DBMS로 넘어가는 DBMS Log 추출 방식이다. 운영 데이터에서의 변화는 일반적으로 백업 및 복구 목적으로 변경 데이터 추출(CDC ; Change Data Capture) Log로 유지되며, 이 Log를 사용하여 변경 데이터를 추출함으로 원천 시스템에 접근하지 않고 추출할 수 있어 시스템 성능에 강점이 있다. 이 방식으로 추출할 때에 ETT 도구는 커미트 포인트(commit point) 및 롤백(roll back)을 정확하게 조절할 수 있어야 한다. 셋째는 데이터베이스에 변화가 일어날 때 자동으로 변경 데이터를 저장하기 위하여 DBMS의 Trigger 기능을 사용하는 방식이다. Trigger를 이용한 온라인 방식의 데이터 추출은 관련된 모든 데이터 원천에 대하여 능동 규칙(active rule)을 정의함으로써 수행된다(Suh & Jung, 1999). 즉, 데이터 원천에 있는 데이터 항목들의 값이 변경될 때마다 그 변화를 인식하고 삽입, 삭제, 갱신 등의 데이터 변경과 관련된 모든 종류의 조작들을 이벤트로 정의하여 그 변경 내용을 데이터 웨어하우스로 보낸다.

여기에서 DBMS Log를 이용한 추출 방식은 어떤 이벤트가 운영 데이터베이스에 있는 1개 이상의 테이블에 갱신을 일으킬 때 목표 시스템에서의 변경 연관관계를 정확히 반영할 필요가 있다. Trigger 지원에 의한 추출 방식은 자동으로 변경 데이터를 처리함으로 변경 연관관계를 비교적 용이하게 유지하여 준다. 일반적으로 User Log, DBMS Log, Trigger를 이용한 즉시 추출 기법들은 원천 데이터에서의 변경 데이터를 빠짐없이 추출할 수 있는 완전성(completeness) 때문에 점진적 데이터 추출을 위해 선호되는 기법들이다. 그러나 기술적 제한으로 인하여 이들 기법의 사용이 어려운 경우가 있는데, 이 경우에 다음의 2가지 지연추출의 방식을 고려할 수 있다(Devlin, 1997). 첫째는 타임스탬프(timestamp) 방식이다. 이 방식은 원천 레코드가 추출을 위한 기준이 되는 타임스탬프 정보를 하

1) 특정 시점에서 필요로 하는 원천 데이터를 추출하는 스냅샷 방식으로, 정적(static) 추출 방식으로 분류한다(Devlin, 1997).

나 혹은 하나 이상의 필드에 포함하여 특정 시점 이후에 처리된 데이터를 일괄 처리한다. 이 방식에 의한 추출은 원천 애플리케이션에서 제공하는 타임스탬프에 의존함으로 원천 데이터가 타임스탬프의 이력정보를 유지하고 있지 않으면 적용하기 어렵다. 두 번째는 파일비교 프로그램을 사용하여 변경 전의 데이터 사본과 변경 후 현재의 데이터 사본의 두 버전을 비교하여 차이를 변경 데이터로 간주하여 추출하는 방식이다. 이 방식은 비교적 적은 수의 변경이 발생하여도 대규모의 레코드를 비교하고 정렬해야 하므로 시스템 성능에 영향을 주며 많은 시스템 자원이 요구된다.

2.2 연구 이슈

Widom(1995), Widom et al.(1995a), Widom et al.(1996), Labio et al.(1997)은 데이터 웨어하우스 구축을 위하여 여러 정보 원천에서 정보를 수집·통합함에 있어 몇 가지 연구 이슈를 제안하고 있다. 그 중의 하나는 웨어하우스의 데이터와 관련이 있는 원천 데이터의 변화를 어떻게 감지하여 이들 변화를 데이터 웨어하우스로 전달하는가에 대한 이슈이다. 즉, 원천 데이터가 변경될 때 데이터 웨어하우스에 그 변경된 데이터를 반영하여 데이터 웨어하우스 데이터의 최신성(currency)을 유지할 필요가 있을 뿐만 아니라 원천 시스템의 성능에 영향을 최소화하면서 변경된 데이터를 목표 시스템에 전달할 필요가 있다. 이러한 문제를 해결하기 위하여 Widom(1995)은 정보 원천의 유형을 <표 1>에서와 같이 4가지 유형으로 분류하고, 이들 유형을 고려하여 데이터 변화 감지를 위한 적절한 대안을 개발할 필요가 있다고 주장하고 있다. 특히 계층형 데이터베이스와 같은 비관계형 모델에 있어서 변경 데이터를 감지하기 위한 대안을 개발할 필요가 있음을 주장하고있다.

그러나, 이질적인 원천 시스템으로부터 데이터 통합을 위한 추출과 관련하여 연구되어 왔던 기존 연구(Zhou et al., 1995a ; Roussopoulos, 1995 ; Widom, 1995 ; Widom et al., 1995a ; Hull & Zhou, 1996 ; Widom et al., 1996 ; Labio et al., 1997)들의 해결 대안들은 관계형 모델로부터의 추출에 대한 연구(Ghandeharizadeh, 1994 ; Zhou et al., 1995a ; Widom, 1995)가 대부분이고, 일부는 객체 지향형 모델로부터의 추출에 대한 연구(Widjojo, 1990 ; Doherty et al., 1995 ; Roussopoulos, 1995 ; Zhou et al., 1995a ; Zhou et al., 1995b)도 있었으나 계층형 모델에서의 추출과 통합의 해결 대안 제시는 소홀히 되어왔다. 또한, 트리거링과 룰 베이스와 같이 Active DB에서 사용되는 기법을 이용한 변경 데이터 추출 및 점진적 갱신과 관련된 이슈(Blakeley et al., 1986 ; Qian & Wiederhold, 1991 ; Gupta et al., 1993)에 대한 연구가 대부분이었으나, DB의 능동성(activeness)과 트리거 기능이 없는 환경에서의 추출과 관련된 연구는 소홀히 되었다. 아울러 원천 시스템의 기술적 환경 특성과 업무 처리 특성 등의 요소에 따라서 해결 대안이 다를 수 있고, 이들 요소를 고려할 때 기존에 제시된 추출 방식(Kelly, 1994 ; Widom, 1995 ; Devlin, 1997)은 한계가 있을 수 있다. 그러나 지금까지의 연구는 이와 같은 요소를 충분히 고려하지 못하였다.

따라서 본 연구에서는 원천 시스템 운영 방식, DBMS 환경 등과 같은 원천 시스템의 기술적 환경 특성에 따라 데이터 추출을 위한 해결 대안이 다를 수 있음을 보여주고자 한다. 또한, 이 요소를 고려하여 기존에 제시된 추출 방식을 비교하고, 이와 관련한 이슈를 해결하기 위한 추출 방식을 제안한다.

본 연구에서 고려하고 있는 원천 시스템의 기술적 환경 특성의 내용을 살펴보면 다음과 같다.

첫째, 원천 시스템의 운영 방식에서 고려할 수 있는 상황은 몇 가지가 있다. 예를 들어, 원천 시스템이 DBMS Log 혹은 User Log를 갖는 방식으로 운영하거나 Log를 갖지 않을 수 있다. 또한, 원천 시스템의 데이터가 타임스탬프를 갖도록 운영을 할 수 있고 그렇지 않을 수도 있다. 아울러, 원천 시스템의 장애시, 데이터 백업 및 복구를 위해 원

천 데이터에 대한 사본을 갖도록 복제 큐(replication queue) 방식으로 시스템을 운영할 수 있다.

둘째, DBMS 환경은 원천 시스템의 DBMS 기능 및 버전 등과 관련된다. 즉, DBMS가 트리거링 기능을 제공하는지의 여부를 고려할 수 있다.

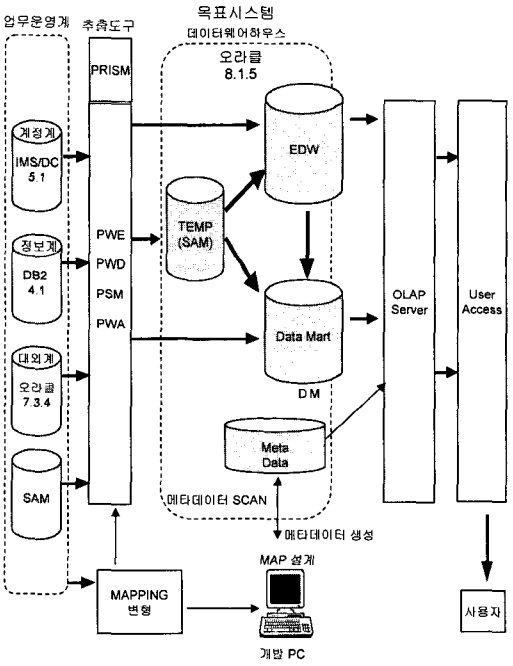
본 연구에서 ㅅ 은행의 시스템 환경을 연구 대상으로 선정한 동기는 여러 정보 원천에 분산된 이질적인 데이터의 통합된 저장소라는 데이터 웨어하우스의 정의(Inmon & Hackathorn, 1994 ; Whitten et al., 1994 ; Poe, 1996 ; Saylor & Bansal, 1995 ; Widom et al., 1995 ; Labio et al, 1997) 관점에서 볼 때 ㅅ 은행의 사례는 이러한 데이터 웨어하우스의 기본적인 정의를 충족시키는 이질적인 시스템 환경을 갖고 있어서 원천 시스템 운영방식, DBMS 환경 등의 기술적 환경 특성을 중심으로 여러 각도에서 추출방식을 비교할 수 있기 때문이다.

3. 원천 시스템의 기술적 환경 특성을 고려한 추출 방식의 비교 · 검토

3.1 ETT 시스템 구성

데이터 웨어하우스로의 ETT를 위한 추출 시스템 아키텍처는 <그림 1>과 같다. 즉, ETT를 위한 ㅅ 은행의 기본 사상은 IBM 메인프레임의 계정계(계층형 IMS/DB5.1) 및 정보계(관계형 DB2 4.1), SAM 파일과 유닉스 환경의 각종 외부 업무와 관련된 대외계(Oracle 7.3.4)로 부터 데이터를 추출하여, 데이터 웨어하우스의 Oracle DBMS(8.1.5)로 적재하는 과정을 거치도록 하는 것이다. 그림에서 목표 시스템에 있는 TEMP는 ETT 도구인 PRISM의 추출 결과를 SAM 파일 형식으로 저장하는 임시 지역 저장소이다. 이 SAM 파일로부터 목표 시스템의 DBMS인 Oracle의 SQL*Loader를 이용하여 적재한다.

ETT 도구인 PRISM의 구성 모듈은 4가지이다. 즉, ① PWE(Prism Warehouse Executive)는 원천 시스템으로부터 목표 데이터베이스로 데이터를 추출 및 변환하는 프로그램(COBOL 또는 JAVA 코드)의 생성과 유지를 지원한다. ② PSM(Prism Schedule Manager)은 ETT 과정의 스케줄 및 프로세스를 관리한다. ③ PWD(Prism Warehouse Directory)는 클라이언트/서버 환경을 통하여 웨어하우스의 업무 데이터 및 메타데이터에 대한 뷰 관리를 지원한다. ④ PWA(Prism Web Access)는 웹 브라우저를 통하여 웨어하우스의 업무 데이터 및 메타데이터에 대한 뷰 관리를 지원한다.



〈그림 1〉 추출 시스템 아키텍처

3.2 시스템별 추출 방식의 비교

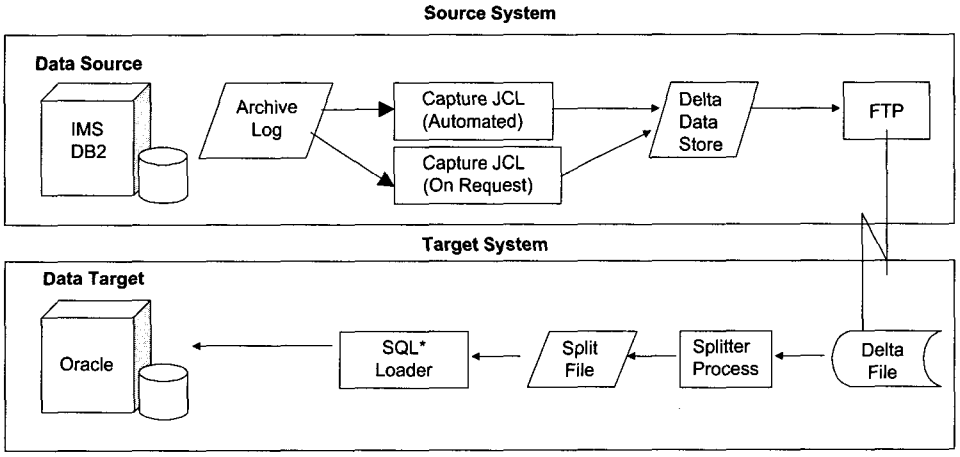
본 장에서는 원천 시스템의 기술적 환경 특성을 중심으로 ETT 도구의 기능을 함께 고려하여 ㅅ 은행의 각 시스템 유형별로 추출방식을 적용하여 비교 · 검토하고 그에 따른 이슈를 찾아보고자 한다.

ㅅ 은행의 추출 대상업무의 업무처리 특성은 주로 온라인으로 처리되고 데이터의 안정성과 빠른 응답 속도를 요구하며 데이터 량과 사용자 수가 많

〈표 2〉 ㅅ 은행의 원천 시스템 환경

기술적 환경 \ 원천 DBMS 종류	IMS	DB2	오라클	SAM 파일
Triggering	×	×	○	×
DBMS Log	○	○	△	×
User Log	×	○	×	○
Timestamp	△	△	×	×

주) ○ : 운영, × : 미운영, △ : 일부 운영.



〈그림 2〉 IMS와 DB2의 DBMS Log를 이용한 추출

다는 특징이 있다. 따라서 <표 1>에서 제시된 추출 대안 중, 원천 시스템으로부터 질의에 의한(queryable) 추출, 파일 비교, 스냅샷 방식은 원천 시스템에 부하를 주기 때문에 금융업무 특성상 바람직하지 못한 방식이다. ㅅ 은행의 각 원천 시스템 환경은 <표 2>와 같다.

IMS와 DB2는 트리거링 기능이 없기 때문에 이 방식으로 운영하지 못하고 DBMS Log 혹은 User Log 방식으로 운영하고 있다. 또한, 일부 원천 데이터는 타임 스탬프를 갖지 않고 있다. 따라서 <표 1>에서 제시한 추출방식 가운데 즉시 추출방식을 ㅅ 은행에 적용하여 검토하고자 한다.

3.2.1 DBMS Log를 이용한 추출 방식

ㅅ 은행은 ETT 도구를 이용하여 DBMS Log로부터 변경 데이터를 추출하기 위해 그 대상 원천 시스템으로 계층형 IMS와 관계형(DB2, 오라클) 데이터베이스를 고려하였다.

먼저, 계층형 IMS와 관계형 DB2로부터의 데이터 추출을 위한 기본 사상은 <그림 2>와 같이 DBMS Log인 아카이브 Log[2]에 수록된 변경 데이터를 추출하고자 하는 방식이었다. 즉, 이 Log로부터 추출된 데이터는 순 변경 데이터(delta data)를 저장하는 델타 데이터 저장소(DDS ; Delta Data Store)에 반영되고, DDS에 저장된 데이터는 FTP 전송 프로그램을 이용하여 목표 시스템으로 전송되어 목표 시스템의 임시 저장소에 저장되며 이것은 목표 시스템에 있는 오라클 DBMS의 SQL* Loader에 의해 데이터 웨어하우스로 적재되도록 하는 방식을 고려하였다.

그러나 DB2의 DBMS Log를 이용한 추출은 별 문제가 없었으나, IMS의 DBMS Log를 이용한 추출 방식은 다음과 같은 문제가 있었다. 즉, ㅅ 은행

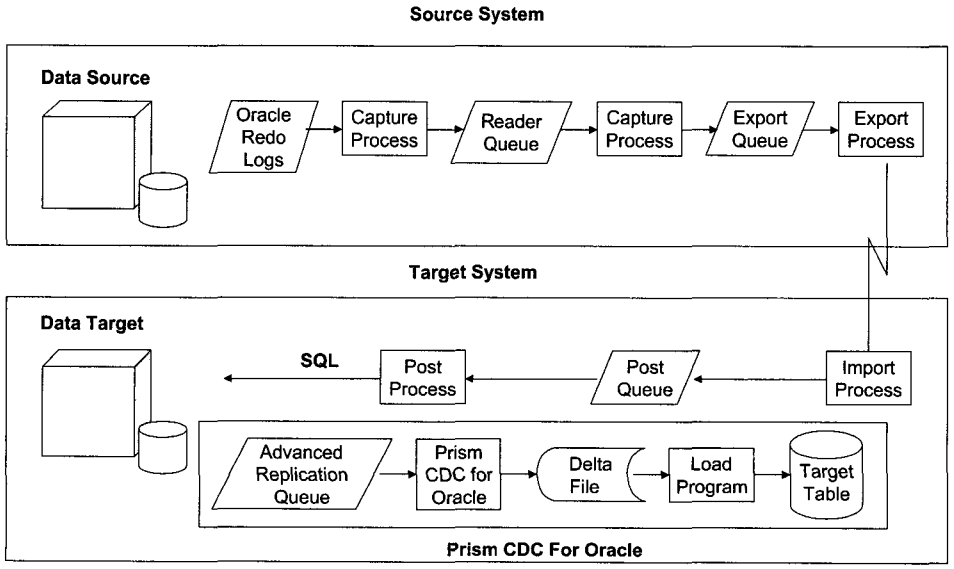
2) 각 온라인 DBMS 시스템에서 DB의 변경이 발생한 데이터의 이미지를 Archive Log 형태로 보관.

을 비롯한 대부분의 국내 금융권의 IMS의 운영 방식은 DEDB(Data Entry Database) 방식이다. DEDB는 대용량의 데이터를 신속하게 처리하기 위하여 일정하게 정해놓은 트랜잭션 단위(예 ; 1,000건 등)로 Log를 갱신하도록 설계된 IMS DB의 구조인 "fast path DB"의 일종으로, IMS DB를 사용하는 국내 금융기관의 독특한 DBMS Log 방식이다. 대표적인 대상 업무의 예는 수시 입출금의 계정성 거래로 발생하는 금융거래 내역, 원부, 고객원장 등과 같이 시스템의 신속한 응답시간(response time)을 요구하는 데이터 베이스를 처리하는 방식이다. 이 DEDB 방식에 의한 모든 트랜잭션은 커미트 정보와 현재 실행 중인 트랜잭션 정보가 DBMS Log에 쌓인다. 한 번의 트랜잭션으로 인하여 여러 개의 세그먼트에서 동시 갱신(concurrent update)이 발생한다. 이때 갱신이 완료된 정보와 갱신 중인 정보가 함께 Log에 쌓이는데 한 번의 트랜잭션에 커미트를 여러 번 실행할 수 있다. 이 경우 여러 번의 커미트에 의해 발생한 동시 갱신으로 인하여 변경된 데이터를 ETT 도구가 완전하게 빠짐없이 인식하지 못함으로써 데이터 추출 시에 일부 데이터가 누락되는 문제점이 있다.

한편, 오라클 DBMS의 경우 일부 업무에 한정하여 DBMS Log를 갖도록 운영하고 있다. 따라서 이 Log를 이용한 추출방안을 2가지 관점에서 검토하였다.

첫 번째 방식은 오라클의 데이터 복제(replication) 솔루션인 셰어플렉스(shareplex)를 이용하여 원천 시스템의 DBMS Log(Redo Log) 데이터를 목표 시스템으로 복제하고 목표 시스템에서는 복제할 때에 Advanced Replication Manager에 의해 생성된 복제 큐(replication queue)로부터 ETT 도구의 CDC 기능을 이용하여 순 변경 데이터를 추출하는 방식이다. 이 방식의 기본 사상은 <그림 3>과 같다. 이 방식은 원천 데이터의 CDC를 지원하기 위하여, 오라클의 트리거 방식과 대비하여 강점이 있는 대안으로 고려되었다. 왜냐하면 트리거 방식에 의할 경우, 원천 시스템에서 다량의 변경 데이터가 발생하면 데이터 추출 양이 증가하여 시스템 부하에 영향을 미치기 때문이다. 그러나 ㅅ 은행은 기존의 오라클 데이터 운영 방식에 Advanced Replication Queue 방식을 적용하여 오지 않았기 때문에 이 방식을 적용할 수 없었다.

또 다른 방식은 오라클 DBMS 버전 8i 이상의



<그림 3> 오라클의 셰어플렉스를 이용한 DBMS Log 추출

기능인 로그 마이너(Log Miner)를 이용하여 원천 시스템의 DBMS Log(Redo Log) 데이터를 일정한 시간 간격으로 변경 데이터를 추출하는 방식이 있다. 그러나 ㅅ 은행의 오라클 데이터베이스의 원천 시스템은 버전 7.3.4를 사용하고 있으므로 이 방식도 적용할 수 없었다.

요약하면, IMS는 DBMS Log의 운영방식을 DEDB 방식으로 운영하고 있는데, ETT 도구가 이러한 방식을 지원하지 못하는 한계가 있다. Oracel의 경우, Oracle DBMS의 Shareplex 혹은 Log Miner를 이용하여 DBMS Log로부터 데이터를 추출할 수 있다. 그러나 Shareplex를 이용하기 위해서는 복제 큐(Replication Queue) 방식의 Log 운영방식이 선행되어야 한다. 또한, Log Miner의 기능을 이용하기 위해서는 원천 시스템의 Oracle DBMS 버전이 8I 이상이 되어야 한다. 한편, DB2의 DBMS Log를 이용한 추출 방식은 문제가 없는 것으로 확인되었다.

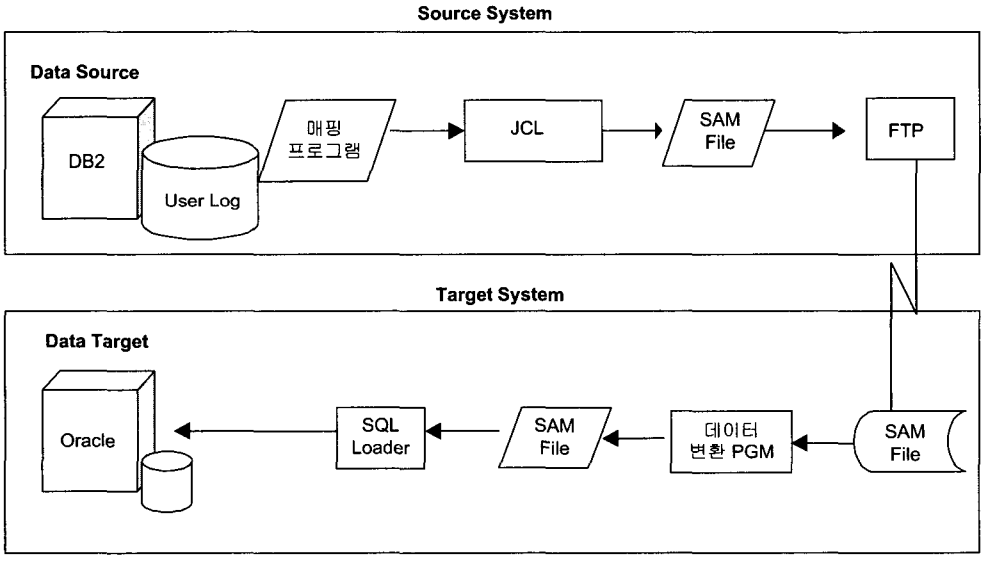
3.2.2 User Log를 이용한 추출 방식

이 방식의 기본 사상은 <그림 4>와 같다. 즉, PRISM의 PWE를 이용하여 작성한 매핑(추출) 프로그램을 IBM 메인프레임에 전송하고, 전송된 프로그램은 컴파일 및 링크를 하여 실행모듈을 생성 후 JCL(Job Control Language)를 실행하여 SAM 파일 형식의 추출 데이터를 생성한다. 이 생성된 SAM 파일은 JCL의 FTP 프로그램을 이용(ASCII 코드 변환 동시 수행)하여 목표 시스템 서버로 전송한다. 목표 시스템 서버에서는 IBM의 원천 시스템으로부터 전송된 SAM 파일에 대하여 변환 작업을 실시한 후 목표 테이블에 SQL*Loader를 이용하여 적재 프로그램에 의하여 데이터를 반영한다.

그러나 ㅅ 은행의 IMS DB와 오라클 DB의 경우, 이들 원천 시스템이 User Log 방식으로 운영하지 않고 있기 때문에 User Log를 이용한 데이터 추출 방식을 적용할 수 없었다. 그러나 DB2의 경우, 시스템 운영방식이 User Log를 갖도록 운영하고 있으므로 DB2는 User Log를 이용한 추출방식의 적용이 가능하였다.

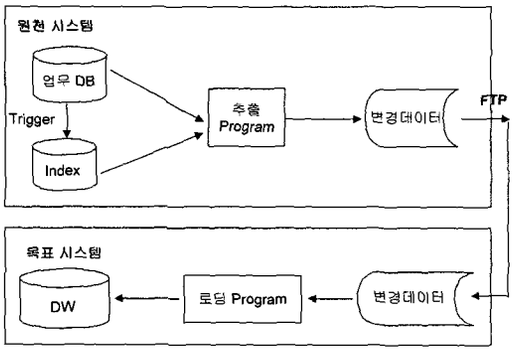
3.2.3 Trigger 기능을 이용한 추출 방식

Trigger는 사용자의 접속이나 사용된 애플리케이션에 관계없이 Trigger와 관련된 삽입, 삭제, 갱신 문장이 수행될 때 DBMS에 의해 묵시적으로 실행



〈그림 4〉 User Log를 이용한 추출

된다. 그런데 ㅅ 은행의 원천 시스템인 IMS와 DB2의 DBMS는 이러한 Trigger 기능이 없으나, 오라클 DBMS는 Trigger 기능을 제공하고 있기 때문에 오라클 DBMS 환경의 업무에만 적용 가능하였다.



〈그림 5〉 Trigger를 이용한 데이터 추출

```
CREATE OR REPLACE TRIGGER
                    Trigger_handle_index_trade
AFTER DELETE OR INSERT OR UPDATE ON trade
FOR EACH ROW
DECLARE
 /* variables are declared to be used in the
    following block */
BEGIN
  IF DELETING THEN
     INSERT INTO trigger_handle_index_trade
          (timestamp, log_key_trade_no, log_gubun, … )
     VALUES (SYSDATE, :trade.trade_no, "D", …) ;
 ELSEIF INSERTING THEN
     INSERT INTO trigger_handle_index_trade
          (timestamp, log_key_trade_no, log_gubun, …)
     VALUES (SYSDATE, :trade.trade_no, "I", …) ;
 ELSE UPDATING THEN
     INSERT INTO trigger_handle_index_trade
          (timestamp, log_key_trade_no, log_gubun, …)
 VALUES (SYSDATE, :trade.trade_no, "U", …) ;
 ENDIF ;
END ;
```

〈그림 6〉 Trigger를 이용한 능동규칙 정의 예

ㅅ 은행의 오라클 DBMS의 Trigger 방식은 After Trigger 기능을 이용하고 있으므로, 이 방식을 이용하여 데이터 추출 방식을 도출하였다. 이 방식의 기본 사상은 <그림 5>와 같다. 즉, 원천 시스템으로부터 변경 데이터가 발생하면 트리거를 이용하여 온라인 방식으로 변경 데이터의 정보, 즉 변경 데이터에 대한 키 값, 이벤트 정보와 타임스탬프 정보 등을 Index 데이터베이스에 저장된다. 원천의 변경 데이터에 대한 키 정보를 Index DB에 저장하기 위하여, 오라클의 시스템 구문을 이용하여 능동규칙(active rule)을 정의하는 형식은 <그림 6>과 같다. 이와 같이 DBMS의 Trigger 기능에 의해 저장된 변경 데이터의 정보를 이용하여 필요한 데이터를 추출할 수 있도록 추출 프로그램을 이용한다. 이렇게 추출된 데이터는 FTP를 이용하여 적재 방식에 의해 목표 시스템에 전송된다. 목표 시스템에 전송되는 변경 데이터는 SAM 파일의 형식으로 일단 목표 시스템의 임시 저장소에 저장된다. 여기서 필요에 따라 다른 데이터와 병합을 하거나 데이터 변형 작업을 수행한 후 목표 테이블에 SQL* Loader에 의하여 데이터를 반영한다.

4. Index DB를 이용한 추출방식의 구현

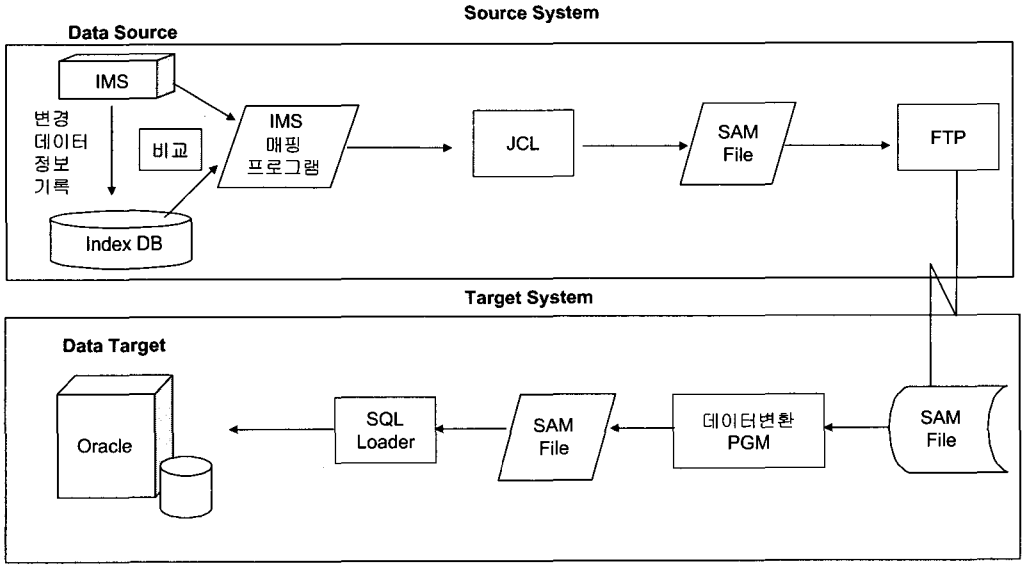
제 3장에서 ㅅ 은행의 원천 시스템 환경별로 추출 방식을 비교하였다. 즉, 원천 시스템이 DB2인 경우는 DBMS Log나 User Log를 이용한 추출 방식이 가능하고 오라클은 Triggering 방식에 의한 추출이 가능하였다. 그러나 원천 시스템 운영 방식이 DEDB 방식의 DBMS Log를 갖는 IMS DB로부터의 데이터 추출은 기존에 제시된 추출 방식으로는 한계가 있음을 확인하였다.

따라서 본 연구에서는 기존의 연구(Widom, 1995 ; Devlin, 1997)에서 제시되지 않았던 Index DB를 이용한 데이터 추출 방식을 ㅅ 은행에 적용하여, 구현 예제를 통하여 그 방식을 설명한다. 이 Index DB는 원천 시스템의 어떤 데이터에서 변경이 발생하였는가에 대한 정보를 알려주기 위한 데이터베이스로, <표 3>에서 보듯이 변경 데이터의 원천 데이터베이스 명, 변경 데이터의 키 값, 변경 데이터의 Log 시간 등의 정보를 갖고 있다.

〈표 3〉 Index DB 스키마

레벨	Key	항목명	SYMBOL	시작위치	속성			설 명
					TYPE	항수	BYTE	
1			SETTMAS1	1				
3		LOG_LENGTH	SETTMASL	1	B	15	2	
3	**Key**	**LOG_KEY**	SETTMASK	3	C	72	72	
5		**LOG_DATE**	FETTDATE		C	8	8	로그 일자(8)
5		**LOG_DBD_NAME**	FETTDBDN		C	8	8	DB NAME(데이터베이스 명)
5		**LOG_ROOT_KEY**	FETTRTKC		C	45	45	ROOT KEY 값
5		**LOG_TIME**	FETTTIME		C	11	11	로그 시간(11)
3		LOG_ROOT_GUBUN	SETTRTGU	75	C	1	1	ROOT 구분(I, D, R, B)*
3		LOG_DEP_GUBUN	SETTDPGU	76	C	1	1	Dependent(자식) 구분
3		LOG_DEP_SEG_NAME	SETTDPSN	77	C	8	8	Dependent Segment NAME
3		LOG_DEP_KEY_값	SETTDPKC	85	C	50	50	Dependent KEY 값

주) * I : Insert, D : Delete, R : Replace, B : Blank.



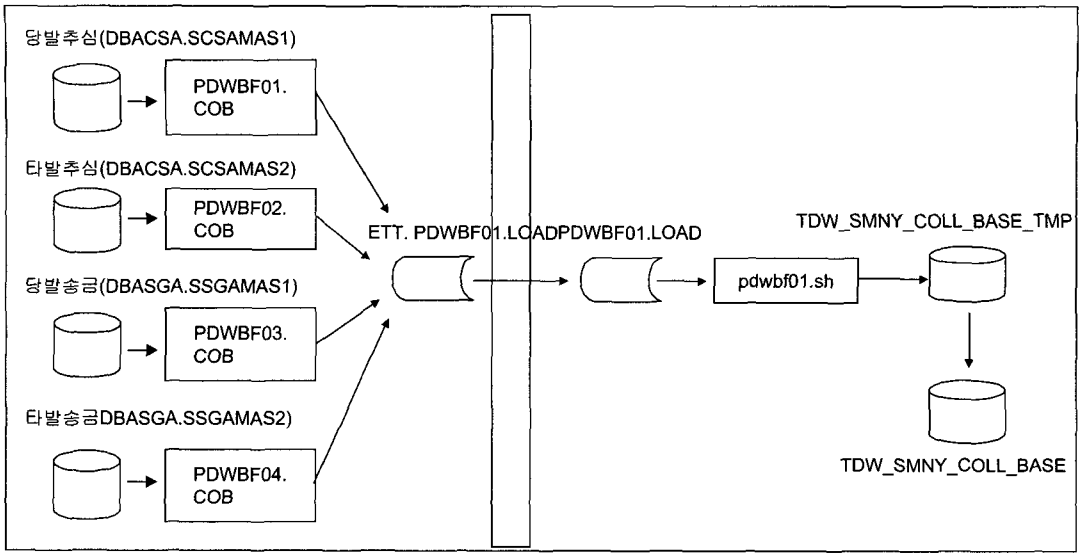
〈그림 7〉 Index DB를 이용한 추출방식

4.1 Index DB를 이용한 추출 방식[3)]

IMS 데이터베이스는 OLTP 성격의 계정계 업무, 즉 예금 등 고객의 입출금과 관련된 업무를 취급하므로 시스템의 성능이 매우 중요하다. 따라서 시스템에 영향을 최소화하면서 데이터 추출의 완전성을 위한 방식으로 <그림 7>과 같은 Index DB를 이용한 추출방식을 ㅅ 은행 환경에 적용하였다.

이 방식의 기본 사상은 별도의 Index DB 생성 프로그램을 작성하여 IMS 시스템에 새로운 Index 용 데이터베이스를 생성하고, 이 Index 데이터베이스에 변경 데이터의 키 정보와 타임 스탬프 정보를 수록하고 이것을 이용하여 추출하는 방식이다.

3) 본 방식은 DEDB 방식의 DBMS Log를 갖는 IMS/DB로부터 데이터 추출을 위한 대안으로 연구자가



〈그림 8〉 외국환 업무 매핑 흐름도

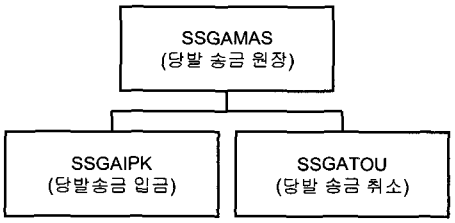
Index 데이터베이스의 세그먼트 스키마는 <표 3>과 같이 기본 인덱스(Primary Index)는 "Log 일자 + DB 명 + 키 값 + Log 시간"으로 구성된다.

원천 데이터에서 트랜잭션이 발생하면 Log 프로그램을 이용하여 그 원천 데이터의 데이터베이스 명(DB NAME)과 그 세그먼트의 키 값, 그리고 일자 및 시간이 인덱스 데이터베이스에 등록된다. 여기서 Log 일자와 Log 시간은 원천 데이터의 변경과 관련하여 추출을 하기 위한 기준이 된다. 즉, 일정한 시점에 추출을 하고 그 이후의 시간을 확인하여 이에 해당하는 원천 데이터를 찾아 추출하게 된다.

4.2 추출 예제

IMS DB의 데이터 추출과 관련하여 ㅅ 은행에 적용하여 구현한 업무는 "외국환" 관련 업무다. 이 업무와 관련한 매핑 흐름은 <그림 8>에서 보여 주는 바와 같이, 4개의 원천 시스템으로부터 목표 시스템의 "송금추심기본(TDW_SMNY_COLL_BASE)" 테이블로의 ETT 과정을 나타낸다. 그 중에서 "당발송금내역(DBASGA. SSGAMAS)" 업무를 중심으로 설명한다. 원천 데이터베이스 명(DB Name)은 "DBASGA(송금원장)"이다. 이것의 계층구조는 <그림 9>와 같다.

<그림 9>에서 세그먼트 "SSGAMAS(당발 송금원장)"는 부모(parent)가 되고, 세그먼트 "SSGAIPK(당발 송금 입금)"와 "SSGATOU(당발 송금 취소)"가 자식(child)이 되는 계층형 데이터베이스이다. 설명을 단순화하기 위해 자식에 해당하는 "SSGAIPK"와 "SSGATOU"의 스키마는 생략한다. 부모 세그먼트 "SSGAMAS"의 키 필드는 "SSGAMASK"이다(<표 4> 참조).



〈그림 9〉 송금원장(DBASGA) DB의 계층 구조

송금과 관련하여 트랜잭션이 발생하게 되면 원천 데이터 베이스인 "당발 송금 원장(SSGAMAS)"

ㅅ 은행에 제안하여 ㅅ 은행의 DW 팀과 함께 구현한 방식이다.

〈표 4〉 원천 데이터(당발송금내역 ; SSGAMAS) 거래 내용

SSGAMASK(키)			FJRO(외화 송금)									
FDSGREF(송금구분)			FDSGNYK(송금내역)					FERINYK(의뢰인 내역)			FSAGYNC (삭제처리여부)	FTOUYNC (취소처리여부)
FBUJREF (지점 구분)	FGURREF (거래 구분)	FSEQREF (일련 번호)	FCHIYMD (취급 일자)	FRWACOD (통화 코드)	FSOGGUM (송금 금액)	FJHCOD (지급은행 코드)	FGJECOD (결제은행 코드)	FCUSBUN (고객번호)	FERINME (성명)	FJIGGUK (지급국가 코드)		
100	ABC	00001	20000720	USD	1000000	KDB	KFB	10030000000001	KGU	008	N	N
100	DEF	00002	20000720	USD	1000	CITYB	JPM	10030000000001	KGU	001	N	N
210	DEF	00001	20000720	USD	500	NYB	NYB	10030000000003	MTS1	001	N	N
200	DEF	00002	20000720	KRW	20000000	CHB	LAB	10030000000001	SYM	002	N	N
:					:	:						
100	ABC	00003	20000720	USD	1000000	KDB	KFB	10030000000001	KGU	008	Y	Y
150	DEF	00003	20000720	JPY	1000	NMB	JPM	10030000000004	JY	002	N	N
130	GHI	00004	20000720	USD	1500	SNB	SNGB	10030000000005	YJS	005	N	N
130	GHI	00005	20000720	USD	1500	SNB	SNGB	10030000000005	YJS	005	N	Y

의 갱신이 이루어지고 트랜잭션 발생과 동시에 Index DB 생성 프로그램에 의하여 데이터베이스 명인 "DBASGA"와 세그먼트의 키 값, 그리고 트랜잭션이 발생한 날짜와 트랜잭션 시간이 Index DB에 등록된다.

IMS 데이터의 추출 과정을 예를 들어 설명하면 다음과 같다.

만약, 'KGU'라는 고객이 '2000년 07월 20일 오전 10시'에 '100' 지점에서 '일백만$'을 송금하였다면, <표 4>와 같이 첫 번째 트랜잭션이 발생하여 원천 데이터베이스에 첫 번째 레코드로 기록된다. 이와 동일한 방법으로 13시까지 거래된 내용이 <표 4>와 같이 4건이 발생하면 인덱스 DB에 이들 4건에 대한 관련 정보가 기록된다. 하루에 2회씩 즉, 13시와 업무가 마감된 밤에 변경 데이터의 추출 작업이 수행된다면, 13시까지 발생한 거래 4건에 대하여 1차 추출을 한다.

그런데 'KGU'라는 고객이 '100' 지점에서 오전에 보낸 송금 '일백만$'는 업무 담당자의 오류로 잘못 처리하여 삭제를 할 필요가 있다고 가정하자. 이 경우 원천 데이터베이스의 'FSAGYNC(삭제처리 여부)'와 'FTOUYNC(취소처리 여부)' 필드에 'Y' 값이 입력되고, <표 5>와 같이 Index DB의 LOG_ROOT_GUBUN 필드에는 삭제(Delete) 정보를 나타내는 'D'가 등록된다. 여기에서 LOG_ROOT_GUBUN 필드는 부모 세그먼트의 갱신관련 이벤트 정보를 기록한다. 또한, 'YJS'라는 고객은 '130' 지점에서 '1500$'의 송금을 의뢰하여 거래처리를 하였다. 그 후 바로 취소 요청이 발생하였다면, 원천 데이터베이스의 'FTOUYNC(취소처리 여부)' 필드에 'Y' 값이 입력되고, <표 5>와 같이 Index DB의 LOG_ROOT_GUBUN 필드에는 취소(Delete) 정보를 나타내는 'D'가 등록된다. 이와 같은 트랜잭션 내용은 <표 4>와 같으며, LOG_DEP_ GUBUN 필드의 'B' 값은 자식(child) 세그먼트의 데이터에서 변경이 없으므로 공란(Blank)을 의미하는 값 'B'가 등록된다(<표 5> 참조).

원천 데이터에 대한 변경 정보에 대하여 다시 2차 추출 작업을 할 때에는 원천 데이터베이스에 있는 모든 데이터를 읽을 필요가 없이, 13시 이후부터 발생한 변경 데이터에 대해서만 추출 작업이 수행된다. 예를 들어, 13시 이후에 발생한 변경 데이터가 <표 4>와 같이 4건이 발생하였다면, ETT 도구의 PWE에서 COBOL로 코딩한, IMS 매핑 변환(추출) 프로그램인 "당발송금(PDWBF03)"을 이용하여 13시 이후에 등록된 인덱스 DB의 날짜와 시간 정보를 갖고 원천 데이터베이스에서 이들에 관련된 정보만을 추출하여 SAM 파일 형식으로 저장

〈표 5〉 Index DB 등록 정보

LOG_KEY				LOG_ROOT_GUBUN (부모세그먼트 갱신이벤트)*	LOG_DEP_GUBUN (자식세그먼트 갱신이벤트)*	중 략
LOG_DATE (로그 일자)	LOG_ROOT_NAME (부모 세그먼트 명)	LOG_ROOT_KEY (부모 세그먼트 키)	LOG_TIME (로그 시간)			
20000720	SSGAMAS	100ABC00001	10 : 00 : 00	I	B	~
20000720	SSGAMAS	100DEF00002	10 : 30 : 10	I	B	~
20000720	SSGAMAS	200DEF00001	11 : 40 : 00	I	B	~
20000720	SSGAMAS	200DEF00002	13 : 00 : 00	I	B	~
20000720	SSGAMAS	100ABC00003	14 : 30 : 10	D	B	~
20000720	SSGAMAS	50DEF00003	15 : 20 : 00	I	B	~
20000720	SSGAMAS	300GHI00004	16 : 25 : 30	I	B	~
20000720	SSGAMAS	300GHI00005	16 : 25 : 30	D	B	~

주) * I : Insert, D : Delete, B : Blank.

한다. FTP 프로그램을 이용하여 목표 시스템 서버로 전송한다. 목표 시스템 서버에서는 IBM의 원천 시스템으로부터 전송된 SAM 파일의 변환을 실시한 후 목표 테이블에 SQL*Loader에 의하여 데이터를 반영한다.

5. 결 론

데이터 웨어하우스 구축을 위한 작업에 있어서, 자동화 도구인 ETT 도구를 사용함으로써 추출 프로그램 작성 등의 노력을 최소화하고 모든 데이터의 구조와 변화를 체계적으로 관리할 수 있기 때문에 ETT 과정의 효율성을 높일 수 있다(Chaudhuri & Dayal, 1997). 그러나 ETT 도구를 사용하는 것이 ETT와 관련한 모든 문제를 해결해 주는 것이 아니다. 왜냐하면, ETT 도구를 적용하여 추출하고자 하는 대상 업무의 업무처리 특성 및 원천 시스템의 기술적 환경 등에 따라서 추출 작업이 용이하게 이루어지지 않는 경우가 있기 때문이다. 따라서 본 연구에서는 정보 원천 유형의 분류(Widom, 1995), 데이터 추출방식(Devlin, 1997) 등의 기존에 제시된 추출 방식을 ㅅ 은행의 원천 시스템의 종류별로 적용하여 추출 방식을 비교하고 적용 가능한 추출방식을 분석하였다. 그런데 정보 원천 유형 가운데 질의에 의한 데이터 추출과 스냅샷에 의한 데이터 추출은 고려하지 않았으며, 파일비교 방식도 검토 대안에서 제외하였다. 왜냐하면, 추출 대상 업무는 데이터 량이 많고 업무 복잡도가 높은 계정계 업무로, 신속한 업무 처리를 요구하는 금융 업무의 특성을 고려할 때 이들 원천으로부터 직접 데이터를 추출하는 작업은 시스템에 많은 부하를 주어 시스템 성능에 영향을 미치기 때문이다. 한편, 원천 시스템이 DB2인 경우, DBMS Log와 User Log를 이용한 변경 데이터에 대한 추출이 모두 가능하였으나, IMS DB는 DEDB 방식의 구조를 갖는 DBMS Log를 이용한 변경 데이터 추출은 ETT 도구가 이 방식을 완전하게 지원하지 못하므로 일부 데이터의 누락이 발생하는 문제점이 있었다. 따라서 ㅅ 은행은 전체적인 추출 전략에 있어 DBMS Log를 이용한 추출 방식을 택하지 않기로 하였다. ㅅ 은행의 원천 시스템 유형별로 적용한 추출 방식을 정리하면 <표 6>과 같다.

본 연구의 의의는 다음과 같다.

첫째, 본 연구에서는 ETT 도구를 사용한 추출 작업에 있어서 ETT 도구의 기능을 고려하고, 적용하고자 하는 추출 대상 업무의 원천 시스템 환경, 즉 시스템 운영방식 및 DBMS 환경에 따라 데이터 추출을 위한 적용 방식이 다를 수 있음을 보

〈표 6〉 ㅅ 은행의 원천 시스템 유형별 추출방식

원천시스템 DBMS	추출방식	가 능 여 부	설 명	적용 추출방식
IMS	Trigger	×	DBMS가 Trigger 기능 없음	Index DB를 이용한 추출
	DBMS Log	×	DEDB의 운영방식을 ETT 도구가 충분히 지원하지 못하므로, 추출 시 데이터 누락	
	User Log	×	User Log 방식으로 운영하지 않았음	
	TimeStamp	×	기존의 원천 데이터는 TimeStamp 정보가 없음	
DB2	Trigger	×	DBMS가 Trigger 기능 없음	User Log
	DBMS Log	○	추출에는 문제가 없음	
	User Log	○	User Log 방식에 의한 시스템 운영	
	TimeStamp	×	기존의 원천 데이터는 TimeStamp 정보가 없음	
Oracle	Trigger	○	임시 Index DB를 만들기 위한 Triggering 적용	Trigger 기능을 이용한 Index DB 생성에 의한 추출
	DBMS Log (Redo Log)	×	오라클의 Shareplex를 이용하여 DBMS Log인 Replication Queue를 만들어 추출하는 방식이나 이 방식으로 운영하지 않았음	
	DBMS Log (Redo Log)	×	오라클 8I 이상에서 Log Miner를 이용하는 방식이나 원천 DBMS 버전이 ORACLE 7.3.4임	
	TimeStamp	×	기존의 원천 데이터는 TimeStamp 정보가 없음	
SAM 파일	User Log	○	User Log 방식으로 운영	User Log

여주고 있다. 따라서 원천 시스템 운영 방식을 중심으로 원천 시스템 환경을 고려하여 ㅅ 은행의 시스템에 실제 적용하여 추출방식을 비교·검토하고 이에 따른 이슈를 확인하여 적용 가능한 추출 방식을 살펴봄으로써 데이터 웨어하우스를 구축하고자 하는 기업에게 참고 사례를 제시하고 있다.

둘째, IMS DB를 사용하는 국내 금융기관의 경우, DEDB 방식의 DBMS Log로 시스템을 운영하는 경우가 있다. 그러나 ETT 도구의 기능을 고려할 때, 기존의 추출방식은 이러한 DBMS Log로부터 추출을 지원하는데 한계가 있을 수 있다. 따라서 본 연구에서는 이러한 문제를 해결하기 위해 Index DB를 이용한 추출 방식을 ㅅ 은행에 적용하여 그 대안의 가능성을 살펴봄으로써 ㅅ 은행과 유사한 시스템 환경을 갖는 기업에게 데이터 웨어하우스 구축에 대한 참고 사례를 제공하고 있다.

셋째, 본 연구에서 적용해본 Index DB를 이용한 추출 방식은 기존 연구에서 제안된 Log를 이용한 추출 방식보다는 원천 시스템의 성능에 부담을 더 주지만, 원천의 변경 데이터에 대한 키 정보를 Index DB가 갖고 있으므로 변경 데이터를 추출하기 위하여 원천 시스템을 전부 검색하지 않고도 키 정보에 의해 변경 데이터만을 추출할 수 있으므로 원천 시스템의 성능에 대한 영향을 최소화할 수 있다는 장점이 있으며, 기존에 제시된 추출 방식의 한계점을 해결하는 대안으로 고려할 수 있다.

그러나, 본 연구를 보다 객관화하기 위해서는 본 연구에서 적용한 ETT 도구뿐만 아니라 여러 개의 ETT 도구를 다양한 시스템 환경에 적용하여 비교·분석할 필요가 있으나 현실적으로 어려움이 있었던 점을 밝힌다.

참 고 문 헌

[1] 김기운, 서용무, "데이터 웨어하우스 ETT 도구들의 평가 및 검증", 「경영정보학연구」, 제10권, 제2호(2000. 6), pp.213-236.

[2] Suh, Yongmoo and Jung, Chulyong, "Data

Integration for DW Consruction", 「정보기술과 데이터베이스 저널」, 제4권, 제2호(1999), pp.79-95.

[3] Berson, A., S. Smith and K. Thearling, *Building Data Mining Applications for CRM*, McGraw-Hill, 2000.

[4] Blakeley, J.A., Larson, P.-A. and Tompa, F.W., "Efficiently updating materialized views," *In Proc. ACM SIGMOD Symp. on the Management of Data*, (1986), pp.61-71.

[5] Chaudhuri, S. and Dayal, U., "An Overview of Data Warehousing and OLAP Technology," *SIGMOD Record*, Vol.26, No.1 (March 1997), pp.65-74.

[6] Devlin, B., *Data Warehouse : from Architecture to Implementation*, Addison-Wesley Inc., 1997.

[7] Doherty, M., Hull, R. and Rupawalla, M., "Structures for manipulating Proposed Updates in Object-Oriented Databases," *Technical Report*, Computer Science Department, Univ. of Colorado, (Oct. 1995), pp. 306-317.

[8] Ghandeharizadeh, S., Hull, R. and Jacobs, D., "Heraclitus (Alg, C) : Elevating deltas to be first-class citizens in a database programming language," *Technical Report USC-CS-94-581*, Computer Science Department, Univ. of Southern California, 1994.

[9] Gupta, A., Mumick, I.S. and Subrahmanian, V.S., "Maintaining views incrementally," *In Proc. ACM SIGMOD Symp. on the Management of Data*, (1993), pp. 157-166.

[10] Haisten, M., "Designing a Data Warehouse," *InfoDB*, Vol.9, No.2(April 1995), pp. 2-9.

[11] Hufford, D., "Metadata Repositories : The Key to Unlocking Information in Data Warehouses," in : Barquin, R.C. and Edelstein H.A.(Eds.) : *Planning and Designing The Data Warehouse*, Prentice Hall PTR, New Jersey, (1997), pp.225-262.

[12] Hull, R. and Zhou, G., "A Framework for Supporting Data Integration Using the Materialized and Virtual Approach," In *Proc. ACM SIGMOD Symp. on the Management of Data*, (Jun. 1996), pp.481-492.

[13] Inmon, W.H., *Building the data ware house*, John Wiley & Sons, Inc., New York, 1992.

[14] Inmon, W.H. and Hackathorn, R.D., *Using the data warehouse*, John Wiley & Sons, Inc., New York, 1994.

[15] Kelly, S., *Data Warehousing : The Route to Mass Customization*, John Wiley & Sons, Inc., New York, 1994.

[16] Labio, W.J., Zhuge, Y., Wiener, J.J., Garcia-Molina, H. and Gupta, H., Widom, J., "The WHIPS Prototype for Data Warehous Creation and Maintenance," *Processings of the ACM SIGMOD conference*, Tuscon, Arizona, May 1997.

[17] Qian, X. and Wiederhold, G., "Incremental recomputation of active relational expressions," *IEEE Trans. on Knowledge and Data Enginering*, Vol.3, No.3(Sep. 1991), pp.337-341.

[18] Roussopoulos, N., "Supporting Data Integration and Warehousing Using H2O," *IEEE Data Engineering*, (1995), pp.29-40.

[19] Salor, M.J. and Bansal, S.K., "Open Systems Decision Support : An Overview," *Data Management Review*, Vol.5, No.1(January 1995), pp.20-24.

[20] Whitten, J.L., Bentley, L.D. and Barlow, V. M., *Systems Analysis and Design Meth-*

ods, 3rd ed., Burr Ridge, IL : Irwin, 1994.

[21] Widjojo, S., Hull, R. and Wile, D.S., "A specification approach to merging persistent object bases," In Al Dearle, Gail Shaw, and Stanley Zdonik, editors, *Implementing Persistent Object Bases*, Morgan Kaufmann, Dec. 1990.

[22] Widom, J. and Ceri, S., *Actilve Database Systems*, Morgan Kaufmann Publishers, California, 1996.

[23] Widom, J., Labio, W.J., Zhuge, Y., Wiener, J.J., Garcia-Molina, H. and Gupta, H., "A System Prototype for Warehous View Maintenance," *Processings of the ACM Workshop on materialized Views : Techniques and Applications*, Montreal, Canada, (June 1996), pp.26-33.

[24] Widom, J, Zhuge, Y., Garcia-Molina, H., Hammer, J. and Labio, W.J., "The Stanford Data Warehousing Project," *IEEE Data Engineering Bulletin*, (June 1995a), pp.41-48.

[25] Widom, J, Zhuge, Y., Garcia-Molina, H., Hammer, J. and Gupta, H., "View Maintenance in a Warehousing environment," *Processings of the ACM SIGMOD conference*, San Jose, California, (May 1995b), pp.316-327.

[26] Widom, J, "Research Problems in Data Warehousing," *Processings of the4th Int'l Conference on Information and Knowledge Management (CIKM)*, November 1995.

[27] Zhou, G., Hull, R. and King, R., "Squirrel Phase 1 : Generating Data Integration Mediators that Use Materialization," *Technical Report CU-CS-793-95*, Computer Science Department, Univ. of Colorado, Nov. 1995a.

[28] Zhou, G., Hull, R., King, R. and Franchitti, J.-C., "Using Object Matching and Materialization to Integrate Heterogeneous Database," *Technical Report*, Computer Science Department, Univ. of Southern California, 1995b.