ETL상에서 파일 시스템을 이용한 대용량 데이터 처리 기법

정윤철
고려대학교 정보통신 대학원 소프트웨어 공학과
e-mail : ycjung@korea.ac.kr

Processing Large Data Using File System On ETL

Yunchul Jung
Dept. of Software Engineering, Korea University
Graduate school of Computer Information Communication

요 약

관계형 DBMS의 보급이 확대되면서 대형 운영시스템 구축 시에 인덱스를 사용하는 관계형 DB의 사용이 증가하고 있다. 이에 따라 Sort의 용도가 대폭 축소되고 DB에서 직접 대형 결산작업이 주로 처리되게 되었다. 그러나 대형 결산 작업 처리시 사용되는 대용량의 데이터의 경우 ETL(Extract Transformation Loading) 작업 시에는 오히려 파일 시스템을 사용하는 경우보다 성능이 저하되는 문제가 발생하기 시작했다. 본 논문에서는 ETL 작업 시 DBMS에 존재하는 대용량 데이터 처리하는 경우에 파일 시스템 상에서 flat 파일을 이용하여 처리 속도를 향상 시키고, 이와 동시에 리소스부하 문제를 해결할 수 있는 방안을 제안했다. 보다 세부적으로 DBMS에서 사용되는 sort, Join, Merge, Summary, 각종 사용자 함수 등의 다양한 기능들을 flat 파일에 적용하는 방법을 제시하였다. 또한 실험을 통해 ETL 작업 시 제안하는 기법이 처리 속도 개선과 리소스 활용성을 향상 시킴을 증명하였다.

1. 서 론

1990년대 중반 이후 기업의 전략적 정보 시스템이 국내에 도입 되면서 그 근간을 이루는 DB 시스템의 성능이 기업의 생산성과 직결되는 중요 요인으로 부각되고 있다. 기업의 전략적 정보 시스템의 성능은 DB의 속도에 따라 수십 배의 속도 차이를 낼 수 있다. 이는 DB의 튜닝상태에 따라 많은 직원들의 질의 결과를 기다려서 확인하고 다음 업무를 처리하는데 걸리는 시간이 수십 배 가량 차이가 난다는 것을 의미한다. 현재 대부분의 상용 DBMS제품들은 자체적으로 최적의 성능 관리와 튜닝을 위한 도구들을 제공하고 있다. 그러나, DB에 유입되는 데이터의 양이 방대해지고 사용자 및 SQL 응용 프로그램의 질의가 점점 복잡해 지면서 기업의 정보 시스템에 처리속도의 문제가 발생하는 경우가 발생하기 시작했다. 또한, 방대한 데이터 양 및 DB 자체의 복잡한 구조로 인해 처리속도 문제가 발생한 경우, 그 원인을 파악하고 해결하기 위한 작업은 점점 더 어려워져 가고 있다. 이러한 성능 문제는 수백 기가 바이트(GB)에서 수십 테라 바이트(TB)에 이르는 대용량 데이터 처리 환경에서 매우 중요한 문제로 대두 된다. 이러한 방대한 규모의 운영 데이터를 얻기 위해서는 BI(Business Intelligence) 인프라가 OLTP(On-Line Transaction Processing) 시스템의 성능 저하를 일으키지 않으면서, 데이터를 수집, 이동, 변환 및 저장을 할 수 있어야 한다. 그러나, 이러한 문제는 서로 다른 운영 시스템, DB, 하드웨어 플랫폼 및 네트웍 환경을 고려하면 매우 해결하기 어려운 문제이다. 이러한 복잡한 환경을 고려하면서 데이터의 추출 기능 개발, 비즈니스 로직 적용, 디버깅 및 안전한 데이터 로딩을 위해 실무 현장에서는 매우 복잡한 데이터 처리 과정을 거쳐야만 했다.

최근 데이터 웨어하우스(Data Warehouse)는 용량은 급격히 커지고 있다. 가장 큰 이유는 초기 데이터의 활용으로 인하여 수익이 증대되고 플랫폼도 그 워크로드를 감당할 수 있게 되면 데이터 활용 용도가 새롭게 생겨나고 이에 따라 데이터 웨어하우스의 요구사항이 늘어나는 선순환 구조를 지니기 때문이다[1]. 데이터 웨어하우스 구축 시 대용량의 데이터를 처리하기 위하여 DBMS에게 많은 것을 맡겨야 하는 데이터 웨어하우스의 메커니즘적인 특수성 때문에 최대한 단순 명료해야 하지만 단순 명료하게 데이터 웨어하우스를 구축하는 것 또한 해결하기 쉽지 않은 문제이다. 이러한 문제에 따라 데이터 웨어하우스 구축을 위해 반드시 필요한 ETL 작업은 큰 폭의 시스템 성능 저하를 일으키는 문제를 지니게 되었다. ETL 작업은 다양한 소스시스템(Source System)으로부터 필요한 데이터를 추출(Extract)하여 사용자의 요건에 맞게 변환(Transformation)작업을 거친 후 타겟시스템(Target System)으로 전송 및 로딩

(Loading)하는 모든 과정을 말한다.[2].

위에서 설명한 BI 인프라가 OLTP 시스템에서 성능 저하 없이 데이터를 수집하는 문제와 증가하고 있는 데이터 웨어하우스의 효율적인 처리를 위해 본 논문에서는 구조적으로 훨씬 간단하고 시스템 자원이 DBMS와 분리된 파일시스템을 활용한 기법을 제안하고자 한다. 파일 시스템을 이용할 경우 자원의 배타적 활용이 가능하여 시스템의 속도 저하 문제와 리소스 활용성을 높일 수 있다.

본 논문에서는 데이터 웨어하우스를 구축하기 위한 필수적인 작업인 ETL 기능을 중심으로 기업의 정보시스템의 성능을 유지하고 대용량의 데이터를 사용자가 원하는 시간에 처리할 수 있는 기법을 제안하며, ETL 기능 구축 작업 시에 OLTP 시스템에 부하를 주지 않고 대량의 데이터를 빠르게 처리할 수 있는 방법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 데이터 웨어하우스 구축과 관련한 관련 연구를 설명하며, 3장에서는 제안하는 기법을 설명한다. 4장에서는 3장에서 설명한 기법을 실험을 통해 증명하고, 5장에서는 본 논문의 결론을 맺는다.

2. 관련연구

데이터 웨어하우스용으로 사용되는 관계형 DBMS인 오라클에서 성능을 획기적으로 향상하기 위해 실체화뷰(materialized view), 분할(partitioning), 비트맵 및 비트맵 조인 인덱스(bitmap and bitmap join index), 테이블 압축(table compression), Star 질의 최적화(star query optimization), 동적 SQL 메모리 관리(dynamic SQL memory management) 등의 기술들이 추가 되고 강화되었다.[3,4,5]. 이러한 기술 등을 통해 결국 해결하고자 하는 바는 본 논문에서 언급하고 있는 대용량 데이터 처리시의 성능 저하 문제와 같지만, 본 논문에서는 파일 시스템을 활용하여 성능향상을 위해 DMBS 자체의 성능 향상이 아닌 파일 시스템 활용을 통한 성능 향상을 통해 보다 간단 명료하면서도 효율적인 기법을 제안한다.

ETL작업의 성능을 향상시키기 위하여 오라클9i는 CDC(Change Data Capture), 외부 테이블, UPSERT, Multitable-Insert, 테이블 함수의 다섯 가지 확장 기능을 제공한다.[6,7,8]. 그러나, DB에 존재하는 데이터 처리는 DB의 기능 및 성능에 의존적일 수 밖에 없는 것이 현실이다. 위에서 설명한 두 가지의 오라클의 대응 기법은 전적으로 DBMS의 기능 및 성능에만 의존 하기 때문에 DBMS의 성능 향상이 없이는 어떠한 성능 향상도 기대할 수 없다는 단점이 존재 한다. 본 논문은 이러한 방법에 탈피하여 DB에서 사용하는 시스템 보다는 상대적으로 자원이 풍부한 파일 시스템을 적극적으로 활용하는 방법을 제안한다.

3. 제안하는 기법

모든 상용 DB에서는 DBMS에 존재하는 모든 데이터들을 flat 파일로 추출할 수 있는 기능과 flat파일을 DBMS로 적재를 할 수 있는 기능들을 지원하는 최적화된 툴을 제공한다. 본 논문에서는 ETL(Extract Transformation Load)상에서 E(Extract)와 L(Load)에 관한 기능 처리에 관하여 DBMS에서 제공하는 최적화된 툴의 기능을 이용하여 처리한다. 따라서, 본 논문에서는 E와 L을 제외한 T(Transformation)의 성능 향상에 주목한다. T는 최적화된 툴을 제공하지 않기 때문에 시스템 성능을 결정 지을 수 있는 중요한 요소이다. 본 논문에서 고려하고 있는 T의 작업은 한글 컨버전, EBCDIC컨버전, Sorting, 조인, 머지, lookup 등의 기본적인 작업과 특정 업무를 위한 사용자 비즈니스 로직 적용 작업을 포함한다.

3.1 개요

Flat 파일을 처리하는 과정에 있어서 제일 먼저 해야 할 작업은 처리하고자 하는 flat 파일의 layout을 지정을 하는 일이며 이 작업을 FFD생성이라 한다. FFD생성은 flat파일 처리 작업시 반드시 선행 되어야 하며, FFD를 어떻게 생성하느냐 하는 문제가 성능에 중요한 영향을 미친다. 두 번째 작업인 Process Script는 FFD에 따라 구성된 컬럼을 이용하여 사용자 비즈니스 로직을 적용하는 단계로서 변환, 가공을 위하여 사용자는 flat파일 처리를 위한 전용 스크립트를 이용하여 비즈니스 로직을 적용한다. 이후 작성된 FFD및 Process Script를 실행 하면 최적화된 C코드가 내부적으로 생성, 컴파일, 실행을 한다. 실행 이후 사용자가 요구한 결과가 제대로 생성되었는지 출력결과를 검증한다. 그림 1은 위에서 설명한 제안하는 기법의 프로세스를 나타낸다.

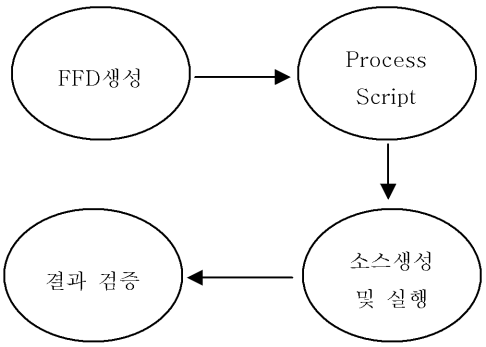

그림1. 제한하는 프로세스

3.2 FFD (File Format Description)생성

본 논문에서는 flat 파일을 테이블 형태의 DB데이터를 컬럼 정보를 제외하고 텍스트로 저장한 것으로 정의한다. 이 flat파일은 컬럼 정보가 없으므로 가공 처리하기 위

해서는 FFD를 생성하여 테이블의 컬럼 정보를 flat 파일에 제공하여야 한다. 작업을 하기 위해서는 반드시 테이블 형태의 DB데이터를 flat 파일 형태로 변환시켜야 하는데, 이 flat파일은 처리 속도를 위해서 컬럼 정보를 제외하고 DB로부터 추출 된다. 그러므로 별도로 FFD를 생성하여 이 flat파일에 컬럼 정보를 주어야 한다.

	주민등록번호	거래코드	매체코드	단말사용자직번	통화코드	원화거래금액	외화거래금액	취소거래여부
1	3302021019113	PDS1D091	18	0000000	000	0	0	N
2	3302042108814	PDS1D091	18	0000000	000	0	0	N
3	3302161058027	PDS1D091	18	0000000	000	0	0	N
4	3302232023621	PDS1D091	18	0000000	000	0	0	N
5	3303031047726	PDS1D091	18	0000000	000	0	0	N
6	3303061052414	PDS1D091	18	0000000	000	0	0	N
7	3303101850828	PDS1D091	18	0000000	000	0	0	N
8	3304031068517	PDS1D091	18	0000000	000	0	0	N
9	3304041046510	PDS1D091	18	0000000	000	0	0	N
10	3305051048011	PDS1D091	18	0000000	000	0	0	N
11	3305071912017	PDS1D091	18	0000000	000	0	0	N
12	3713081067011	PDS3D11A	18	0000000	410	0	0	N
13	3713081117118	PDS1D091	18	0000000	000	0	0	N
14	3713102260214	PDS1D091	18	0000000	000	0	0	N
15	3713172052714	PDS1D091	18	0000000	000	0	0	N
16	3713172231720	PDS1D091	18	0000000	000	0	0	N

그림 2. DB원본 데이터

```
3302021019113  PDS1D091 180000000      000     0     0N
3302042108814  PDS1D091 180000000      000     0     0N
3302161058027  PDS1D091 180000000      000     0     0N
3302232023621  PDS1D091 180000000      000     0     0N
3303031047726  PDS1D091 180000000      000     0     0N
3303061052414  PDS1D091 180000000      000     0     0N
3303101850828  PDS1D091 180000000      000     0     0N
3304031068517  PDS1D091 180000000      000     0     0N
3304041046510  PDS1D091 180000000      000     0     0N
3305051048011  PDS1D091 180000000      000     0     0N
3305071912017  PDS1D091 180000000      000     0     0N
3713081067011  PDS3D11A 180000000      410     0     0N
3713081117118  PDS1D091 180000000      000     0     0N
3713102260214  PDS1D091 180000000      000     0     0N
3713172052714  PDS1D091 180000000      000     0     0N
3713172231720  PDS1D091 180000000      000     0     0N
```

그림 3. DB에서 추출된 flat파일

그림 2는 컬럼 정보를 가진 DB 데이터를 보여주며, 그림 3은 그림 2의 원본 데이터로부터 추출된 flat 파일을 보여준다. 위의 그림들에서 보이는 바와 같이 DB데이터를 flat파일로 변환하면 컬럼 정보가 제외된 형태로 저장이 된다. 하지만 flat파일 처리를 하기 위해서는 반드시 컬럼정보의 속성이 반드시 지정이 되어야 하며 이 컬럼정보를 기반으로 하여 원하는 데이터 가공처리 한다. 그림 3의 flat파일에 대한 컬럼의 정보를 정의하는 FFD는 아래의 그림 4와 같다.

이름	데이터형	구분자	차례위치	시작위치	크기
주민번호	ASCII	,	1	1	13
거래코드	ASCII	,	2	2	8
매체코드	ASCII	,	3	3	2
단말사용자직번	ASCII	,	4	4	7
통화코드	ASCII	,	5	5	3
원화거래금액	ASCII	,	6	6	1
외화거래금액	ASCII	,	7	7	1
취소거래여부	ASCII	,	8	8	1

그림 4. DB에서 추출된 Flat파일의 FFD정보

본 논문에서 제안하는 기법에 따르면 모든 flat파일은 반드시 FFD를 정의하여야 한다. 이는 DB의 원본 데이터는 그림 4와 같이 FFD정보를 정의하는 것뿐만 아니라 예상되는 출력 결과에 대한 FFD도 실행 전 미리 지정을 하여야 한다는 것이다. DB에서 table을 사용자가 원하는 형식에 맞게 create한 후 데이터를 insert, update하는 것과 같다고 할 수 있다.

3.3 Process Script 작성

Process script는 사용자가 작성하는 script를 제공하는 역할을 한다. FFD에서 정의한 각각의 출력 컬럼에 대하여 입력 컬럼을 입력으로 하여 Process Script를 작성하게 된다. 이 Process Script는 각종 변수와 함수들로 구성이 되어 있으며 여러 가지 복잡한 수식이나 연산을 원활하게 표현할 수 있도록 한다. 변수와 함수를 표현하기 위한 다양한 명령어들을 제공하므로, 여러 가지 복잡한 수식이나 연산들을 원할 하게 처리할 수 있다. 이러한 방법을 통해 중복되고 복잡한 작업을 간단하게 표현함으로써 작업의 효율을 높일 수 있다. 그림 5는 특정 조건을 충족시키기 위한 Process Script작성 예제이다.

조건 :
A라는 컬럼은 B라는 컬럼의 첫번째 문자부터 3자리를 자른 후의 결과가 “가”이면 “ABC”라는 결과를 얻고, “나”이면 “DEF”라는 결과를 얻고, 나머지는 “ZZZ”를 얻는다.

Process Script :
A = decode(substring(B, 1, 3), “가”, “ABC”, decode(B, “나”, “DEF”, “ZZZ”))

그림 5. Process Script 사용 예제

그림 5처럼 다양한 사용자 Built-In함수를 제공함으로써 사용자가 복잡한 수식이나 연산을 쉽고 원활하게 사용할 수 있도록 하는 기능을 제공한다. Built-In함수는 그림 5에서처럼 decode, substring등의 C함수를 미리 library형태로 제공이 되며 소스 컴파일/링크시 Built-In 함수 library가 최적화 된 C코드와 함께 사용이 된다.

3.4 소스생성 및 실행

소스 생성 및 실행 단계에서는 최고의 성능을 위하여 사용자가 입력한 데이터를 바탕으로 최적의 코드를 생성하여 실행한다. 이를 위해 매핑 작업을 수행하게 되는데 매핑 작업에서는 각각의 컬럼에 타입변환이나, 숫자연산, String가공 등 다양한 작업을 Process Script를 통하여 사용자는 원하는 형태로 변환하게 된다.

출력 FFD에 정의된 각각의 컬럼에 대하여 모든 입력 컬럼을 가지고 사용자가 원하는 출력파일에 대한 출력 컬럼 별 연산을 적용하게 되는데, 이를 컬럼간 메핑작업이라 한다. 매핑 작업이 완료 된 후 사용자가 실행을 하면 작성된 매핑 내용을 기반으로 하여 최적화된 C코드

를 생성된다. 이 C코드는 C컴파일러에 의하여 컴파일 및 링크 작업을 거친 후 실행 모듈이 생성, 실행을 하게 된다.

```
Process Script :
decode(substring(B, 1, 1), "가", "ABC",
decode(B, "나", "DEF", "ZZZ"))

최적화된 C코드 생성 :
if ( strcmp(substring(B, 1, 1) , "가") == 0)
        return "ABC";
else
        if (strcmp(B, "나") == 0)
                return "DEF";
        else    return "ZZZ";
```

그림 6. 최적화 된 C코드 생성

그림 6은 사용자 Built-In함수에서 제공하는 decode 문을 이용하여 B라는 컬럼을 가공처리 하였다. 실행이 되면 이 부분은 최적화된 C코드 생성 부분처럼 C코드가 생성이 되어 처리 된다.

3.5 결과 검증

작업을 통하여 나온 결과물이 정확한지를 검증하는 것은 무엇보다 중요한 작업이다. DB에서는 레코드 count나, 특정 조건의 값을 조회하는 등의 다양한 검증 방법이 있지만 flat파일에서는 DB처럼 다양한 경우의 검증을 하기란 쉽지 않다.

본 논문에서 제시하는 검증에 대한 방법은 레코드 count로 알 수 있는 결과 건수에 대한 검증은 결과 log 값으로 제공을 하지만, 특정 조건 값에 대한 결과 검증 방법은 출력 파일에 대한 특정 컬럼이 sorting되이 있는 경우, 해당 컬럼에 대하여 Binary search 알고리듬을 통하여 조회를 하고, 그렇지 않은 경우에는 출력 파일에 대하여 순차적으로 결과파일을 읽어서 조회하는 방법을 제시한다. 그림 7은 검증 방법을 위한 실제 SQL문 예제이다. 그림 7의 예제에서처럼 DB에서의 SQL문과 유사하게 처리를 하며 where조건에 명시한 컬럼이 sorting 되어 있으냐 없느냐에 따라 달리 적용이 되며 검증 속도 또한 차이가 많이 난다.

검증을 위한 SQL : SELECT 주민등록번호, 책임자승인여부 FROM "/tmp/data.dat" WHERE 거래코드 = " PD999" AND 통화코드 = "000"

결과 파일인 '/tmp/data.dat'에서 FFD로 정의된 컬럼 중 '주민등록번호', '책임자승인여부' 라는 컬럼 값을 조회하는 조건으로서 FFD의 다른 컬럼인 '거래코드'가 'PD999"이고 '통화코드'라는 컬럼의 값이 '000'인 컬럼만 조회하는 SQL질의문이다.

그림 7. 실행 결과 검증

4. 성능 평가

성능 평가를 위해 본 논문에서 제안한 기법과 SQL을 활용한 기법을 비교 평가 하였다. 이를 위해 HP9000 L2000-44, 4 PA 8500 CPUs @ only 440MHz, 8GB RAM의 하드웨어가 사용되었으며, DB는 Oracle9.2를 사용하였다. 성능 비교는 처리 속도에 대한 비교 실험을 실시하였다. 그림 8은 ETL처리 과정에서 SQL문 처리 방식과 flat파일 방식 처리의 성능을 평가한 자료이다.

그림 8을 통해 Flat 파일을 통한 처리의 경우 SQL을 통해 처리한 경우 보다 빠른 처리 속도를 보임을 알 수 있다. 예를 들어 100건 정도의 ETL 작업을 수행 할 경우에 약 4배 가량의 속도 향상을 보여주는 것을 볼 수 있다.

이렇게 파일 시스템을 이용한 기법이 SQL기법 보다 성능이 좋아지는 원인은 다음과 같은 이유로 결론 지을 수 있다.

첫째는 DBMS와 flat파일의 데이터 구조적 차이에서 오는 자원의 활용이다. Flat파일이 DBMS보다 매우 심플한 구조로서 이에 따른 CPU와 I/O사용률, 메모리 사용의 자유도, 다른 작업과 동시 수행할 수 있는 환경 등의 시스템 자원을 적극 활용할 수 있다.

둘째는 파일 시스템에서의 소팅 처리이다. Flat파일의 소팅 처리는 파일 시스템의 메모리 사용의 최적화와 I/O방식의 효율화를 가져올 수 있으며, 멀티 프로세스 같은 병렬처리를 극대화 함으로서 성능의 극대치를 높일 수 있다.

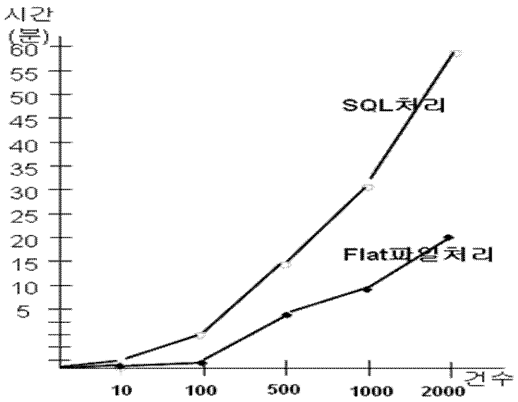

그림 8. 처리 속도 비교

5. 결론

본 논문에서 제시한 flat파일 처리 형식은 DB에 있는 대량의 데이터를 처리하는 많은 장점을 가지고 있다. 첫째 시스템 부하를 분산하는 측면이다. 많은 자원을 소비하는 DB와의 업무 분리가 자연스럽게 이루어져 시스템 부하를 낮춤으로 인하여 하드웨어 및 소프트웨어에 대

한 시스템 투자 비용을 절감할 수 있다. 둘째 성능 향상으로 인하여 ETL작업의 활용 작업인 배치 작업의 주기를 줄일 수 있고, ON-Line작업과 Batch작업을 분리로 안정적 On-Line환경을 보장하면서 빠른 Batch작업이 가능하다. 본 논문에서는 flat파일을 활용하여 ETL 작업 처리 속도를 향상시킬 수 있는 방법을 제안하였으며 실험을 통해 성능 향상을 보였다. 향후 연구로 DB의 대용량 데이터를 파일 시스템 상에서 flat파일을 이용한 방법의 활용도를 높이기 위해, SQL문에서 제공하는 많은 다양한 기능을 flat 파일을 활용하여 구현하는 것에 대한 연구를 수행할 예정이다.

6. 참고문헌

[1]Oracle Corp., "ETL Processing within Oracle9i,"Oracle Technical White Paper 2002

[2] Bill Inmon, "Building the Data Warehouse". John Wiley, 1992

[3]이상원, "데이터웨어하우스를 위한 데이터베이스 기술 소개", 정보과학회지 제21권 제10호, 2003. 11, pp. 12 ~ 22 (11pages)

[4]Randall G. Bello, Karl Dias, Alan Downing, James Feenan, Jim Finnerty, William D. Norcott, Harry Sun, Andrew Witkowski, Mohamed Ziauddin., "Materialized Views in Oracle," In Proceedings of VLDB, 1998

[5]Meikel Pss and Dmitry Potapov, "Table Compression in Oracle," Proceedings of VLDB 2003

[6]Oracle Corp., Oracle9i Data Warehousing Guide, June 2001

[7]Oracle Corp., Oracle9i Reference, June 2001

[8]김은주, 용환승, 이상원, "데이터웨어하우스 성능 모니터링을 위한 DBMax의 확장", 한국정보과학회 2002년도 가을 학술발표논문집 제29권 제2호(Ⅰ), 2002. 10, pp. 262 ~ 264 (3pages)