韓國 컴퓨터情報學會 論文誌
第12卷 第1號, 2007. 3.

2007-12-1-3-2

비유일 외래키 조합 복합키 기반의 사실테이블 모델링과 MDX 쿼리문 작성법

유 한 주*, 이 덕 성*, 최 인 수**

A Nonunique Composite Foreign Key-Based Approach to Fact Table Modeling and MDX Query Composing

Han-Ju Yu*, Duck-Sung Lee*, In-Soo Choi**

요 약

스타 스키마는 중앙에 사실테이블이 있고 이 주변에 여러 개의 차원테이블이 감싸고 있는 형태로 되어있다. 사실테이블에서의 각 행은 여러 외래키로 구성된 하나의 복합키와 이 복합키와 관련된 여러 측정값으로 구성된다. 복합키의 구성원인 각 외래키는 각 하나씩의 차원테이블과 관련을 맺고 있다. 그런데 문제가 되는 것은 OLAP 스키마에서는 사실테이블에서의 복합키와 측정값이 1 : 1의 관계를 맺고 있는 것으로 되어있는데 비해서 실제의 애플리케이션 특히 금융 애플리케이션에 있어서는 복합키와 측정값이 1 : N의 관계를 맺게 된다는 점이다. 따라서 1 : N의 관계를 1 : 1로 만들기 위해서 예를 들면 SQL 데이터베이스에서 미리 필요한 처리를 한 다음 이 선처리 결과를 OLAP 데이터베이스에 입력하는 방법을 취하게 되는데, 이 방법에도 여러 문제가 있는 것으로 알려져 있다. 특히 어떤 경우에는 결과값이 틀리게 나오기도 한다.

본 연구에서는 1 : N의 관계를 사실테이블에 유지를 하면서도 어떠한 선처리도 하지 않고 정확한 결과값을 산출할 수 있는 사실테이블의 모델링과 MDX 쿼리문 작성법을 제안하고 있다.

Abstract

A star schema consists of a central fact table, which is surrounded by one or more dimension tables. Each row in the fact table contains a multi-part primary key(or a composite foreign key) along with one or more columns containing various facts about the data stored in the row. Each of the composit foreign key components is related to a dimensional table. The combination of keys in the fact table creates a composite foreign key that is unique to the fact table record. The composite foreign key, however, is rarely unique to the fact table record in real-world applications, particularly in financial applications. In order to make the composite foreign key be the determinant in real-world application, some precalculation might be performed in the SQL relational database, and cached in the OLAP database. However, there are many drawbacks to this approach. In some cases, this approach

• 제1저자 : 유한주
• 접수일 : 2007.1.2, 심사일 : 2007.1.22, 심사완료일 : 2007. 3.2.
* 숭실대학교 산업·정보시스템공학과 박사과정, **숭실대학교 산업·정보시스템공학과 교수
※ 본 연구는 숭실대학교 교내 연구비 지원으로 이루어졌음.

might give users the wrong results. In this paper, an approach to fact table modeling and related MDX query composing, which can be used in real-world applications without performing any precalculation and gives users the correct results, is proposed.

▸ Keyword : 1 對 多 관계(1 : N Relationship), 금융 애플리케이션(Financial Applications), 선처리(Precalculation), 다차원 분석(OLAP Analysis), MDX 쿼리 작성법(MDX Query Composing)

Ⅰ. 서 론

다차원 데이터베이스는 비즈니스 의사결정지원 분야에서 중요시 여겨지는 요소이다. 다차원 데이터베이스에서는 데이터를 계층구조에 맞게 저장시키는데, 이 계층구조의 대상이 되는 것에는 비즈니스 구조를 반영시키는 차원이 있다. 예를 들어 일반적인 제품판매에 관한 다차원 데이터베이스에는 제품, 시간, 판매지역과 같은 차원들이 계층구조로 되어있다. 다차원 데이터베이스는 두 가지의 중요한 특징이 있다. 첫째, 다차원 데이터베이스에서 계층구조란 비즈니스 데이터 자체의 계층구조이다. 이 계층구조 덕분에 사용자가 데이터를 네비게이션 하는 것이 자연스럽고 쉬워진다. 둘째, 다차원 데이터베이스에서는 하나의 차원계층 내에 여러 수준(levels)을 설정하여 이 설정된 수준별로 데이터를 축적하고 집계한다는 점이다. 이 때문에 사용자는 계층의 각 수준을 오르내리면서 분석 작업을 능률적으로 수행할 수 있게 된다[1,5,6].

다차원 데이터베이스에서 긴밀하게 활용되는 분석도구가 OLAP(OnLine Analytical Processing)이다[1,7]. OLAP은 사용자가 의사결정에 필요한 지식을 찾아내기 위해 대량의 비즈니스 데이터를 쉽게 분석할 수 있도록 도와주는 데이터베이스 도구이며, 사용자 위주의 관점으로 설계하는 계층형 데이터베이스이다[1,8,9,11]. OLAP과 분석가와의 대화 즉, 질의를 가능하게 해주는 것이 MDX(Multi-Dimensional eXpressions)인데 이 MDX는 차원과 사실테이블로 구성된 큐브에 기반하여 질의하는 언어이다. 따라서 쿼리구문의 해석이 용이하고 계산이 정확하다는 특징을 가진다[2,4].

다차원 데이터베이스를 설계하는 가장 보편화된 방법은 스타 스키마이다. 스타 스키마는 중앙의 사실테이블을 여러 개의 차원테이블이 둘러싼 형태를 취한다. 사실테이블에서의 각 행은 여러 차원의 기본키들의 조합 즉, 외래키들로 조합된 하나의 복합키로 구별되며 이 복합키와 관련된 여러 측정값으로 구성된다. 그런데 문제가 되는 것은 OLAP 스키마에서는 사실테이블에서의 복합키와 측정값이 1 : 1의 관계를 맺고 있음으로 인하여 실제의 애플리케이션 예를 들면 금융 애플리케이션과 같은 복합키와 측정값이 1 : N의 관계를 맺고 있는 경우에는 그 표현이 불가능하다는 것이다[1]. 따라서 1 : N의 관계를 1 : 1로 만들기 위해서 분석자들은 해당 질의를 실행하기 전에 1 : N 의 관계를 미리 집계함으로써 1 : 1의 관계로 변화시키는 과정을 거친다. 다시 말하자면 각 제품 또는 계정의 평균 금액이나 평균 비율로 선처리하여 1 : 1의 관계로 처리 한 다음 이 선처리 결과를 OLAP 데이터베이스에 입력하는 방법을 취하여 왔다[1,2,4]. 그러나 이 방법에는 여러 가지 문제가 있는 것으로 밝혀지고 있다. 예를 들면 제품 단가가 매일 매일 하루에도 여러 번 바뀌면서 입·출고 또는 판매되는 형태 즉, 구성원 속성의 값들이 매시간 달라지는 경우에서는 결과값이 틀리게 나오기도 한다는 것이다. 다시 말하자면 리프멤버를 구성하는 인스턴스들이 모든 리프멤버에서 일률적으로 같은 개수로 존재하게 되는 경우는 인스턴스를 따로 불리하여 처리하는 방법을 사용할 수 있지만 그렇지 않고 변동적이거나 랜덤 할 경우 이를 다룰 수 있는 방법은 현재로는 OLAP에서나 관계형 데이터베이스에서는 찾아볼 수 없다[1,3].

따라서 본 연구에서는 1 : N의 관계를 사실테이블에 유지하면서도 어떠한 선처리도 하지 않고 정확한 결과값을 산출할 수 있는 사실테이블의 모델링과 MDX 쿼리문 작성법을 제안하고자 한다.

Ⅱ. OLAP의 아키텍처

OLAP은 사용자가 의사결정에 필요한 지식을 찾아내기 위해 대량의 비즈니스 데이터를 쉽게 분석할 수 있도록 도와주는 차원과 사실테이블로 구성된 큐브에 기반 하는 분석 도구 이다. 다차원 데이터베이스의 기본적인 구조인 스타 스키마에서 다차원 모델은 사실(fact) 데이터와 차원(dimension) 데이터로 구성되며, 사실 데이터로 구성된 사실 테이블을 중심으로 여러 개의 차원 테이블이 뻗어져 나오는 형태를 취한다. 단순한 형태의 스타스키마는 (그림 1)

에서처럼 하나의 정규화 된 사실 테이블과 이에 연결된 다수의 차원테이블로 구성되는데 이런 스타 스키마는 실제 사용자가 쉽게 이해할 수 있고, 단순하게 모델링 할 수 있게 때문에 결과적으로 사용자가 차원에 대해 쉽게 질의를 작성할 수 있고, 데이터베이스는 사용자의 질의에 빠른 속도로 응답할 수 있게 되는 것이다. 하나의 사실 데이터와 여러 개의 차원 데이터는 각기 차원 데이터의 기본키(primary key)로 연결되며, 하나의 사실 데이터는 여러개의 관련된 차원의 기본키의 조합으로 구별된다. 여기서 기본키의 조합이란 외래키(foreign key)의 조합 즉, 각 차원의 리프멤버(leaf member)의 조합이 된다[1,12].

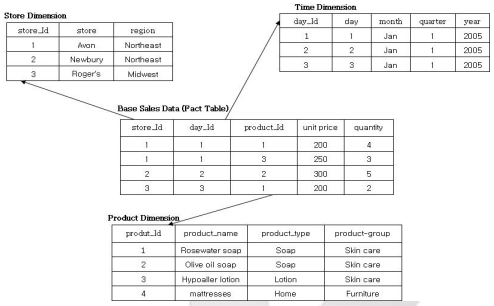


그림 1. 스타스키마
Figure 1. Star Schema

(그림 1)에서 store_Id가 1이고, day_Id가 1, product_Id가 3인 기본키를 가지는 행은 2005년 1월 1일에 상점 Avon에서 단가가 250원인 제품 Hypoaller lotion이 4개가 팔렸다는 정보를 지니고 있다. 이처럼 사실 테이블의 데이터들은 여러개의 관련된 차원의 기본키의 조합으로써 측정값의 의미를 나타낸다. 그러나 만약 같은 날 같은 상점에서 같은 제품이 단가가 240원, 수량이 5개가 팔렸다는 데이터가 추가되었다면 이 스타 스키마의 사실 테이블은 store_Id가 1이고, day_Id가 1, product_Id가 3, 그리고 측정값 unit price가 240이고 quantity가 5인 행을 추가해야 할 것이다. 이 의미는 같은 기본키를 가진 행이 두 개 이상 발생하였다는 말이 된다. 이럴 경우에 (그림 1)의 스타 스키마는 동일한 기본키를 가지는 두 개 이상의 행들을 구별해 줄 수 있는 방법이 없다[1,2,11].

이런 환경에 대처하기 위해서는 리프멤버가 기준이 되는 것이 아니라 리프멤버의 인스턴스가 기준이 되어야만 한다. 리프멤버의 인스턴스란 리프멤버를 구성하는 구성원이다. 즉, store_Id가 1이고, day_Id가 1, product_Id가 3인 기본키를 가지는 단가 250과 240을 가지는 두 개의 행을 말한다. 이처럼 리프 멤버의 인스턴스가 여러개 있을 경우는 인스턴스들의 선처리 과정이 필요한데 대개의 경우 평균값을 사용한다. 평균이란 여러 수나 같은 종류의 양의 중간 값을 갖는 수를 말한다. 제품 단가의 인스턴스들의 평균은 일반적으로 제품 차원의 리프멤버인 제품의 구성원 속성(member property)으로 정의되어왔다. 구성원 속성이란 구성원의 특성을 설명해 주는 항목들을 말한다. (그림 2)에서 보는 바와 같이 데이터베이스나 데이터웨어하우스에 있던 소스들을 선처리 하여 평균 단가와 같은 유형의 속성들을 정의한다. 이 선처리 과정이 끝나 다음에서야 비로써 OLAP 큐브를 생성하여 활용하는 것이다.

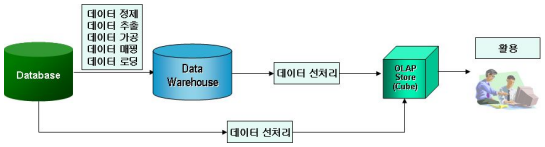


그림 2. 큐브 아키텍처
Figure 2. Cube Architecture

그러나 기존의 이 방법에는 분석자가 분석하고자 하는 기간이나 차원의 수준이 바뀌었을 경우에는 또 다른 큐브를 생성하여야만 한다는 결점이 생기게 된다. 단가가 하루에도 몇 번씩 변한다는 말은 평균 단가가 매일 변할 수 있다는 말과 같다. 이 말은 각 날짜에 해당되는 올바른 평균 단가를 계산하기 위해서는 날짜별 소스의 데이터를 추출하여 선처리(또는 선계산)하고, 이 선처리 한 데이터를 이용하여 날짜별 큐브를 생성하여야만 한다는 의미를 가진다. 따라서 이런 경우 단가를 제품차원의 구성원 속성으로써 정의한다는 것은 현실적으로 실현 불가능 하다고 할 수 있겠다.

본 연구에서는 기존의 실현 불가능한 설계 방법이 아닌 새로운 방식의 리프멤버 각각의 인스턴스를 다루는 방법과 모든 것의 출발점이라고 할 수 있는 RDB의 새로운 설계방법을 제시하고자한다. (그림 3)은 본 연구에서 제시하는 큐브 아키텍처이다.

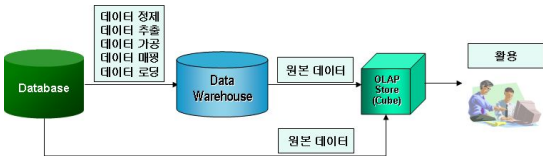


그림 3. 큐브 아키텍처의 새로운 제안
Figure 3. Revised Cube Architecture

(그림 3)에서는 (그림 2)에서의 선처리 과정이 생략되어 있는 것을 볼 수 있다. 따라서 본 연구에서 제시하는 방법으로 데이터베이스나 데이터웨어하우스를 설계한다면 OLAP 큐브를 생성하는데 필요했었던 선처리 과정을 거치지 않고도 원본 데이터를 그대로 사용하여 간단하고 올바른 분석값을 산출 할 수 있게 된다.

III. 사실 테이블 모델링

3.1 선처리 유무에 따른 차이

2장에서 기존 OLAP의 방법에서는 리프멤버의 인스턴스를 다룰 수 있는 방법이 없기 때문에 리프멤버의 올바른 평균을 구하는 것이 현실적으로 불가능 하다는 것을 제시한바 있다. 이 장에서는 왜 OLAP에서 평균을 구하는 방법에 문제점이 있는지를 보이고 이 문제점을 해결하기 위한 본 연구의 방법론을 제시하고자 한다.

평균이란 여러 수나 같은 종류의 양의 중간 값을 갖는 수를 말하며 일반적으로 산술 평균을 말한다. 즉, 여러 수의 합을 수의 개수로 나눈 값이 산술평균이다. OLAP의 MDX 식을 이용하여 평균을 구하는 방법 중 하나는 'Avg' 명령어를 사용하는 것이다[2,4,11]. (그림 4)의 내용을 살펴보자.

Market_ID	Market_Name	Country
1	U.S.A.M1	U.S.A.
2	U.S.A.M2	U.S.A.
3	Canada_M1	Canada
4	Canada_M2	Canada
5	Canada_M3	Canada

그림 4. Market차원과 Sales에 대한 사실테이블
Figure 4. Market Dimension and Sales Fact Table

(그림 4)에 대한 평균을 명령어 'Avg'을 이용해서 MDX 식으로 구하면 결과는 (그림 5)와 같다.

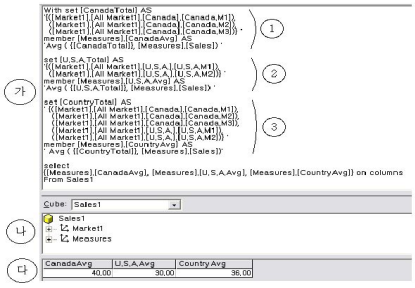


그림 5 'Avg'를 사용하여 구한 평균
Figure 5. Calculated Average Results by Using 'Avg'

MDX 도구는 대화식 질의를 하는 질의창(가), 선택한 큐브의 구조를 보여주는 큐브창(나), 질의의 결과를 나타내는 결과창(다)으로 구성되어 있다. ①은 캐나다에 있는 마켓 세 군데를 CanadaTotal이라는 이름의 집합으로 만들고 만든 집합의 평균을 계산하여 CanadaAvg라는 이름으로 나타내라는 의미를 가진다. ②는 미국에 대한 집합과 평균을 ③은 전체 5군데의 마켓들의 평균을 계산하라는 의미이다. 각 결과는 결과창에 나타난바와 같다. 그러나 'Avg' 명령어는 평균값을 구하기 위해서는 반드시 평균값을 구하는데 필요한 멤버들로 구성되어진 세트를 만들어야 한다. 따라서 'Avg' 명령어로 구한 평균은 세트를 구성하는 멤버들의 총합을 세트를 구성하는 멤버의 개수로 나눈 산술평균이 된다. 그러나 'Avg' 명령어는 회계나 재정 애플리케이션에서와 같이 리프멤버의 인스턴스가 다수 존재하는 상황에서는 현실적으로 사용불가능하다는 문제점이 있다. 따라서 본 연구에서는 'Avg' 명령어를 사용하지 않고 평균을 구하고자 하는 것이다. (그림 6)은 회계의 대차대조표계정 차원에 대한 계층도이다.

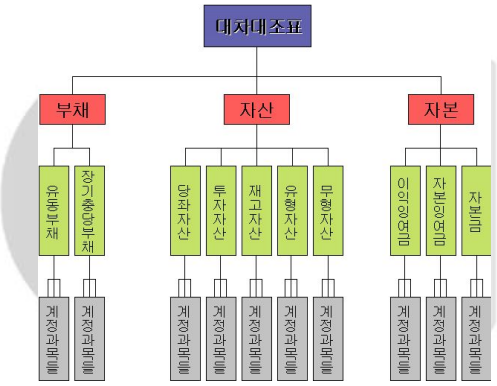


그림 6. 대차대조표의 계층도
Figure 6. A Balance Sheet Hierarchy

대차대조표계정차원과 전표 사실테이블의 예제 데이터를 보면 각각 (그림 7), (그림 8)과 같다.

계정기초 년도	계정코드	대분류	중분류	계정명	차대구분
2005	10140	자산	당좌자산	보통예금	0
2005	10220	자산	당좌자산	정기적금	0
2005	10310	자산	당좌자산	유동성 만기보유증권	0
2005	10410	자산	당좌자산	설비외상매출금	0
2005	20730	부채	유동부채	미지급비용-기타	1
2005	20710	부채	유동부채	미지급법인세	1
2005	21310	부채	유동부채	부가세예수금	1
2005	21410	부채	유동부채	가수금	1
2005	33110	자본	이익잉여금	이익준비금	1
2005	33910	자본	이익잉여금	이월이익잉여금	1
2005	34910	자본	이익잉여금	배당금	1

그림 7. 대차대조표계정차원
Figure 7. Accounts Dimension in the Balance Sheet

전표번호	전표순번	계정코드	시간 코드	차대구분	차변금액
2005052908	20	20650	149	1	11848950
2005052908	21	10140	149	0	0
2005052909	1	63910	149	0	32000
2005052909	2	20310	149	1	0
2005052910	1	21410	149	1	745620
2005052910	2	21410	149	1	2328610
2005052910	3	21410	149	1	3000000
2005052910	4	21410	149	1	940490
2005052910	5	10620	149	0	0
2005052911	1	20120	149	1	70290000
2005052911	2	20310	149	1	0
2005052912	1	10610	149	0	907500
2005052912	2	72910	149	1	0
2005052912	3	21310	149	1	0

그림 8. 전표 사실테이블
Figure 8. Fact Table for Chit Data

2장에서 기존 OLAP의 설계방법으로는 리프멤버들에 대한 멤버라고 할 수 있는 인스턴스의 숫자를 세는 것이 불가능하기 때문에 선처리 과정을 거치지 않고서는 각 계정에 대한 평균값을 구할 수 없다고 제시한바 있다. 먼저 기존 방법대로 각 리프멤버의 인스턴스들을 선처리 하여 각 계정의 평균값을 구해보도록 하자. 분석자가 (그림 7), (그림 8)의 그림으로부터 특정기간 동안의 부채계정의 유동부채를 구성하는 각 계정의 평균값을 구하고자 한다. 각 계정의 평균값을 구하는 MDX식은 (그림 9)와 같다.

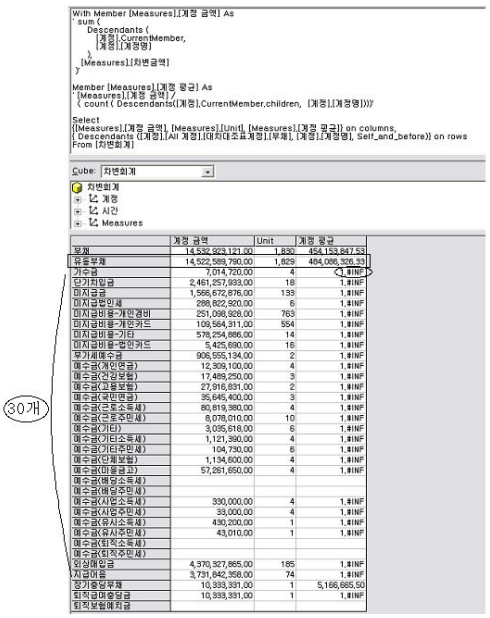


그림 9. 유동부채 계정의 평균
Figure 9. Calculated Average of Floating Liabilities Account

(그림 9)의 MDX 표현식은 분석자가 원하는 수준까지 즉, 부채계정까지 집계하여 그 집계한 총액을 멤버들의 숫자 즉, 리프멤버의 숫자로 나누어 평균을 구하라는 의미를 가진다. 결과창의 계정평균 컬럼에서 보는 바와 같이 유동부채의 평균은 전체 유동부채 금액 14,522,589,790원을 유동부채를 구성하는 멤버 개수 30으로 나눈 값인 484,086,326.33원이 되는 것은 확인할 수 있다. 그러나 선처리 하는 기존 방법은 계정 차원의 리프멤버에 속하는 계정의 인스턴스 개수를 세는 것이 불가능하다. 그 예로서 리프멤버인 가수금의 계정 평균을 보자. (그림 8)의 사실테이블에서 가수금 계정에 속하는 즉, 계정코드가 21410인 행은 총 네 개이며, 각각 745,620, 2,328,610, 3,000,000, 940,490원이라는 금액을 가지고 있다. 그러나 이들 값을 선처리 하였을 경우에는 가수금에 대한 각각의 인스턴스를 구별하지 못하게 되므로 4개의 인스턴스의 총 금액인 7,014,720원은 산정할 수 있지만 이들의 인스턴스 평균이 얼마라는 것은 알 수 없게 된다. 결과창의 "1.#INF" 이 가지는 의미는 분모가 0에 수가 나와서 INFINITE (무한수가)가 나온 경우 즉, 분모가 될 숫자가 없다는 말이다. 따라서 선처리를 하는 기존 방법은 유동부채 계정처럼 유동부채를 구성하는 하위 멤버 수준이 존재할 경우는 그 평균값을 산정할 수 있지만 하위 수준이 없는 리프멤버의 평균값을 구할 수 있는 방법이 없다는 것을 확인할 수 있었다. 따라서 선처리한 값은 가수금의 예처럼 인스턴스 정보를 알 수 없게 되므로 결과적으로 정보손실이 생긴다고 할 수 있겠다[1]. 따라서 본 연구에서는 선처리를 하지 않고도 올바른 수준별 평균값을 구하는 방법을 제시하고자한다. 본 연구에서 제시하는 방법을 사용하여 테이블을 설계하면 (그림 10)과 같다.

전표번호	전표순번	계정코드	시간 코드	차대구분	차변금액	unit
2005092401	73	20610	267	1	78400	1
2005052909	1	63910	149	0	32000	1
2005092401	74	20610	267	1	78400	1
2005052910	1	21410	149	1	745620	1
2005052910	2	21410	149	1	2328610	1
2005052910	3	21410	149	1	3000000	1
2005052910	4	21410	149	1	940490	1
2005092401	75	20610	267	1	77400	1
2005052911	1	20120	149	1	70290000	1
2005092401	76	20610	267	1	10000	1
2005052912	1	10610	149	0	907500	1

그림 10. unit 컬럼을 추가한 전표 사실 테이블
Figure 10. Fact Table with Unit Column for Chit Data

본 연구에서는 리프멤버의 인스턴스를 다루기 위해 (그림 10)에서 보는 바와 같이 사실테이블에 유니트(unit) 컬럼을 추가하였다[13,14]. 유니트 컬럼은 리프멤버의 각 인스턴스를 식별해주기 위해 추가한 컬럼이다. 유니트 컬럼을 사실테이블에 추가함으로써 발생하는 장점에 대하여 하나씩 알아보도록 하자. (그림 10)에서 추가한 유니트 컬럼을 이용하여 MDX식을 사용하면 (그림 11)의 결과를 얻을 수 있다.

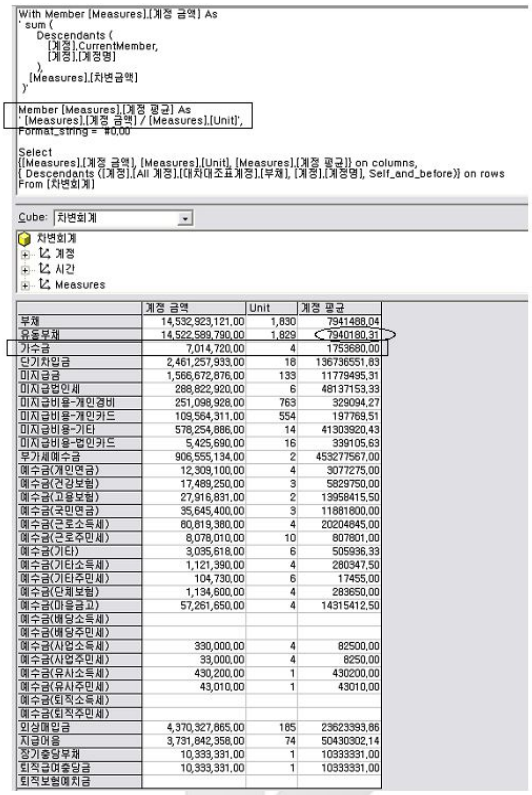


그림 11. unit 컬럼을 활용하여 구한 평균
Figure 11. Calculated Average by Using Unit Column

(그림 10)에서 추가한 유니트 컬럼은 리프멤버의 인스턴스 개수부터 모든 수준의 각 멤버 수준까지 집계(aggregation)된 인스턴스의 숫자를 셀 수 있다. (그림 11)의 MDX 식은 (그림 9)의 MDX 식에서 멤버들의 숫자로 나누는 부분을 연구에서 제시한 유니트 컬럼을 이용하여 각각의 인스턴스의 개수를 집계하여 나누는 공식으로 바뀐 것을 확인 할 수 있을 것이다. 결과창에서 리프멤버 가수금의 총금액은 7,014,720원이며 이 가수금 총액은 4개의 가수금의 금액들로서 이루어진 것임을 알 수 있다. 즉, 가수금을 구성하는 인스턴스 숫자는 4이고 이 4개의 인스턴스의 조합으로써 가수금의 총금액이 산출되었다는 것이다. 따라서 리프멤버인 가수금의 계정평균 즉, 인스턴스 평균은 7,014,720원을 4로 나눈 1,753,680원이다. 본 연구에서 제시한 유니트 컬럼을 사용하면 리프멤버의 구성원인 인스턴스에 대한 정보를 유지하게 되므로 정보의 손실이 있었던 기존방법에 비해 우월하다고 말 할 수 있다.

본 연구에서 제시한 방법을 사용하면 리프멤버의 멤버인 인스턴스의 개수뿐만 아니라 분석자가 원하는 수준의 멤버별 인스턴스의 정확한 개수까지 셀 수 있다. 다시 말하자면 1 : N의 관계를 사실테이블에 유지하면서 어떠한 선처리를 하지 않고, 정확한 결과값을 산출할 수 있다는 것이다. 따라서 연구에서 제시하는 유니트 컬럼을 사용하면 한 제품에 대해 단가가 하루에도 여러번 바뀌는 상황이라던가 예제와 같은 회계전표의 각 계정별 금액을 산출하는 유형의 문제에 대해 해결 가능하리라 생각한다.

또한 본 연구에서 제시한 유니트 컬럼과 MDX식을 사용하면 인스턴스에 1 : N의 관계를 나타낼 수 있는 장점과 함께 수준별 각 멤버의 올바른 평균인 가중평균을 구할 수 있다. 가중평균이란 각 항의 수치에 그 중요도에 비례하는 계수를 곱한 다음 산출한 평균으로서 정밀도나 들어온 양이 같지 않은 물품의 평균 가격처럼 원래의 수치가 동등하지 않다고 생각되는 경우에 주로 사용하는 값이다. 회계나 금융에 관련된 데이터들은 동등한 빈도나 금액을 가지고 있지 않으므로 동등한 관계로써 분석을 해서는 안 된다. 따라서 수준별 각 멤버의 평균을 구하고자 할 때에는 (그림 9)에서처럼 각 멤버를 구성하는 멤버의 개수만으로 구하는 산술평균이 아닌 (그림 11)에서 구한 인스턴스의 개수에 비례하는 가중치를 고려한 가중 평균으로써 구하여만 올바른 평균값을 구할 수 있다.

3.2에서는 수준별 각 멤버의 올바른 평균값인 가중 평균을 구하는 방법을 알아보도록 하겠다.

3.2 가중 평균

3.1에서는 유니트 컬럼을 사실테이블에 추가함으로써 수준별 각 멤버의 평균 금액을 구하는 방법을 알아보았다. 여기서는 3.1에서 구한 계정 금액과 유니트 컬럼을 사용하여 올바른 수준별 각 멤버의 평균값 즉, 가중평균을 구하는 방법을 알아보고자 한다.

분석자는 OLAP의 MDX를 활용하여 (그림 6)의 대차대조표계정에서 부채의 평균을 구하려 한다. 기존 설계방식에서 MDX의 결과는 부채의 총 금액을 부채의 멤버인 유동부채와 장기충당부채의 개수인 2로 나눈 금액을 산정하거나 유동부채에 해당되는 계정명의 개수와 장기충당부채에 해당되는 계정명의 개수의 합으로 부채의 총금액을 나눈 평균값을 계산 할 것이다(그림 9참조).

(그림 6)에 대한 대차대조표계정차원과 전표 사실테이블인 (그림 7)과 (그림 8)을 다시 살펴보다. (그림 7)과 (그림 8)의 데이터를 기반으로 차변회계 큐브를 생성하면 (그림 12)의 스키마 구조를 가지게 된다.

그림 12. 차변회계 큐브 스키마
Figure 12. A Cube Schema for a Debtor Account

(그림 12)의 차변회계 큐브의 데이터를 활용하여 부채의 평균을 구해보자. OLAP의 MDX 편집기를 이용하여 부채의 평균을 구하는 방법은 (그림 13)과 같다.

(그림 13)의 MDX식은 리프멤버의 금액을 대분류까지 집계하여 총 금액을 구하고 구한 금액을 중분류에 해당하는 멤버의 숫자로 나누라는 의미를 가진다. 결과에서처럼 부채의 평균은 부채의 전체 금액인 14,532,923,121원을 부채의 멤버인 유동부채와 장기충당부채의 개수 2로 나눈 7,266,461,562.5원이라는 것을 확인할 수 있다. 과연 분석자는 (그림 13)에서 구한 부채의 평균금액 7,266,461,560.5원이 올바른 부채의 평균 금액이라고 판단 할 수 있겠는가?

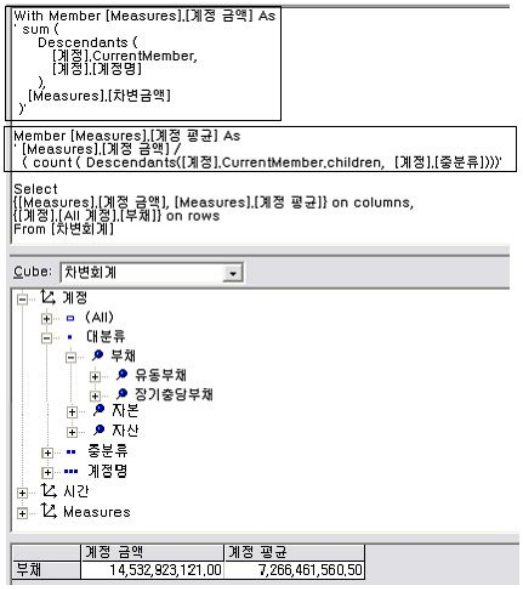


그림 13 부채의 평균
Figure 13. Calculated Average of a Debt Account

이 경우 부채의 멤버인 유동부채에 해당하는 금액과 빈도가 장기충당부채의 금액과 빈도보다 훨씬 큰 차이를 보인다면 일반적인 산술평균으로 동등하게 구한다는 것은 무의미한 값을 산출하였다고 할 수 있을 것이다. 실제로 이 데이터는 부채의 멤버 중 하나인 유동부채가 1,829번의 빈도를 가지며 14,522,589,790원이라는 금액을 차지하고 있으며 장기충당부채는 1번의 빈도와 10,333,331원의 금액을 차지하고 있다. 또 하나의 경우, 유동부채에 해당하는 금액이 0원이고 장기충당부채에 해당하는 금액이 1,000만원이라고 한다면 부채의 평균은 500만원이 나온다는 것이다. 즉, 계정이 금액을 가지고 있지 않는 경우에도 비어있는 계정이 단지 상위 수준의 멤버라는 이유만으로 평균을 구할 때 나누어주는 개수에 포함되어야 한다는 것이다. 이처럼 값이 없는 멤버는 상위 수준의 값을 나누기 위한 멤버에 포함되어서는 안 된다. 즉, 논리적으로 올바른 계산이 아니라고 할 수 있다. 전술한 두 가지 결점을 가지는 이유는 3.1에서 설명하였듯이 선처리 된 값은 그 값에 대한 정보를 가지지 못하기 때문이다.

따라서 전술한 상황에 대처하기 위해서는 부채의 하위 수준인 유동부채와 장기충당부채의 빈도수에 따른 가중평균을 구하여야만 한다. 수준별 멤버들의 올바른 평균값을 구하기 위해서는 3장에서 살펴본 리프멤버의 인스턴스에 대한 빈도수를 구하는 방법처럼 각 수준 멤버에 속하는 계정의 빈도수를 구하는 방법이 먼저 제시되어야 한다. 각 수준 멤버들의 계정 빈도수를 구한 다음 기준 수준의 특정 멤버에 대한 상대적인 비율을 구하고 그 비율에 대한 가중치를 부여한 값을 산정하여야만 올바른 평균값인 가중평균을 계산할 수 있다.

각 수준별 멤버의 올바른 가중평균을 산출하기 위해서도 본 연구에서 사실테이블에 추가한 유니트 컬럼이 활용된다. 예제에서 부채의 올바른 가중 평균을 구하기 위해서는 우선 부채 계정의 리프멤버들의 인스턴스의 개수를 먼저 구하여야만 한다. 이 말이 의미하는 것은 특정 기간 동안에 발생한 회계 전표 중에서 부채 계정에 속하는 모든 인스턴스를 식별하라는 뜻을 내포하고 있다. 부채의 유동부채와 장기충당부채에 해당하는 계정별 인스턴스의 개수를 구한 후에 부채의 멤버인 유동부채와 장기충당부채 수준까지 그 개수를 집계하여 두 계정의 빈도를 구하게 된다. 그리고 두 계정을 합한 즉, 부채계정의 총 빈도수에 대한 유동부채와 장기충당부채 각각의 빈도수에 대한 비율을 구한다. 부채에 대한 두 계정의 비율을 구한 다음 유동부채와 장기충당부채의 금액에 각각의 비율을 곱하여 부채계정의 올바른 평균인 가중 평균을 구한다. 상기 절차의 MDX 공식은 (그림 14)와 같다.

(그림 14)에서 부채계정에 대한 총금액은 14,532,923,121원이고 모두 1,830번이 발생하였다는 것을 확인할 수 있다. 부채를 구성하는 멤버인 유동부채는 1,830번 중 1,829번을 차지

함으로써 0.999454의 비율을 차지하며 또 다른 멤버인 장기충당부채는 1번 발생함으로써 0.000546을 차지하고 있음을 확인 할 수 있다. 또한 유동부채의 리프멤버인 가수금이 전체 부채계정 중 0.002186의 비율을 차지하고 있으며 계정 가중 평균은 15,332.72원이라는 것까지 한눈에 확인할 수 있다.

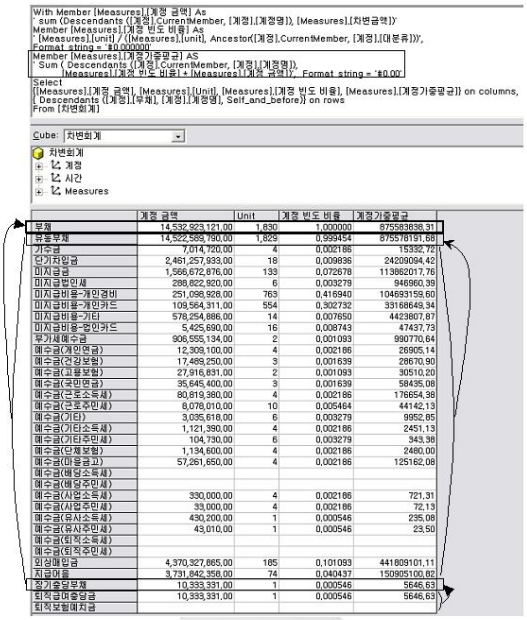


그림 14. 부채의 가중 평균
Figure 14. Weighted Average of a Debt Account

따라서 부채를 구성하는 리프멤버 각각의 비율과 해당하는 가중평균까지도 계산할 수 있게 된다. 여기서 유동부채의 가중평균은 유동부채를 구성하는 각 멤버들의 가중평균의 합으로 계산되며 그 금액은 875,578,191.68원이다. (그림 14)의 질의창의 박스부분은 본 연구에서 제시한 MDX식이다. 본래 MDX의 With Memeber를 사용하여 만드는 측정값은 그 측정값을 만드는 공식을 따르게 된다. 이 예제에서 유동부채의 계정가중평균은 기존방법을 사용했을 경우에는 유동부채 금액 14,522,589,790원에 계정 빈도 비율 0.999454를 곱한 금액인 14,14,60,55.97원을 산정한다. 그러나 정확한 유동부채의 가중평균 값은 유동부채를 구성하는 가수금부터 지급어음까지의 각 계정의 계정가중평균값들을 집계한 값을 가져와야만 한다. 따라서 본 연구에서는 올바른 값을 산정하기 위해 Sum과 Descendants를 적절하게 사용하여 구하고자 하는 올바른 가중평균값인 875,578,191.68원을 산정할 수 있게 하였다.

예제를 통해서 본 연구에서 제시한 유니트 컬럼을 사용하면 데이터 소스에 대한 선처리 과정을 거치지 않고도 각각의 인스턴스의 대한 정보 손실 없이 정확한 값을 산출할 수 있다는 것을 확인 할 수 있었다.

Ⅳ. 사례연구

이번 장에서는 3장에서 제시한바 있는 사실테이블에 유니트 컬럼을 추가하는 방법과 MDX 작성법을 채택하여 기존 데이터 소스의 선처리 과정에서 발생하는 여러 문제점을 해결하는 방법을 사례를 들어 설명한다. 사례 연구에서는 회계계정 차원을 사용하여 회계 관리 시스템을 설계하고자 한다. 2005년 1년간 발생한 전표 699개의 8,845개 데이터를 가상으로 설정하여 회계관리 시스템의 데이터로 삼았는데, 사례의 계정차원 계층도는 (그림 15)와 같다.

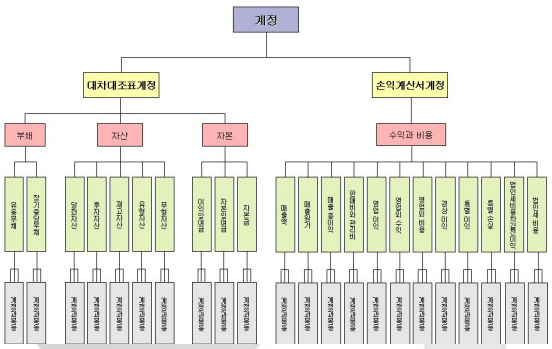


그림 15. 계정 계층도
Figure 15. An Accounts Hierarchy

본 사례의 계정차원과 시간차원, 차·대변 전표 사실테이블은 각각 (그림 16), (그림 17), (그림 18), (그림 19)와 같다.

계정기초 년도	계정코드	대분류	중분류	소분류	계정명	차대구분
2005	10140	대차대조표계정	자산	당좌자산	보통예금	0
2005	10220	대차대조표계정	자산	당좌자산	정기적금	0
2005	10310	대차대조표계정	자산	당좌자산	유동성 만기보유증권	0
2005	10410	대차대조표계정	자산	당좌자산	설비외상매출금	0
2005	10430	대차대조표계정	자산	당좌자산	공사외상매출금	0
2005	10440	대차대조표계정	자산	당좌자산	용역외상매출금	0
2005	10450	대차대조표계정	자산	당좌자산	받을어음	0
2005	10490	대차대조표계정	자산	당좌자산	매출채권대손충당금	1
2005	10520	대차대조표계정	자산	당좌자산	임직원단기대여금	0
2005	71110	손익계산서계정	수익 비	영업외 수	이자수익	1
2005	71210	손익계산서계정	수익 비	영업외 수	배당금수익	1
2005	71610	손익계산서계정	수익 비	영업외 수	외환차익	1
2005	72220	손익계산서계정	수익 비	영업외 수	대손충당금환입	1
2005	72910	손익계산서계정	수익 비	영업외 수	잡이익	1
2005	75110	손익계산서계정	수익 비	영업외 비	이자비용	0
2005	75610	손익계산서계정	수익 비	영업외 비	외환차손	0
2005	76110	손익계산서계정	수익 비	영업외 비	유형자산처분손실	0
2005	71500	손익계산서계정	수익 비	영업외 비	무형자산감액손실	0
2005	76910	손익계산서계정	수익 비	영업외 비	잡손실	0

그림 16. 계정차원
Figure 16. Accounts Dimension

시간 코드	년	분기	월	날자
1	2005	1	1	20050101
2	2005	1	1	20050102
3	2005	1	1	20050103
4	2005	1	1	20050104
5	2005	1	1	20050105
6	2005	1	1	20050106
7	2005	1	1	20050107
359	2005	4	12	20051225
360	2005	4	12	20051226
361	2005	4	12	20051227
362	2005	4	12	20051228
363	2005	4	12	20051229
364	2005	4	12	20051230
365	2005	4	12	20051231

그림 17. 시간차원
Figure 17. Time Dimension

전표번호	전표순번	계정코드	시간 코드	차대구분	차변금액
2005010101	1	10140	1	0	175
2005102401	21	20610	297	1	20000
2005010201	1	10440	2	0	-85967625
2005103102	43	20110	304	1	30800000
2005101802	1	10440	291	0	193344253
2005010201	4	10620	2	0	3245432
2005092803	15	20590	271	1	38440
2005	5	41110	270	0	64500
2005091506	2	57210	258	0	8000000
2005121505	1	58510	349	0	90000
2005121505	2	58510	349	0	220000
2005121505	3	58510	349	0	120000
2005121505	4	58510	349	0	188600
2005090907	3	51710	252	0	27216000

그림 18. 차변 회계 사실 테이블
Figure 18. Fact Table of a Debt Account

전표번호	전표순번	계정코드	시간 코드	차대구분	대변금액
2005122802	22	15090	362	1	153472
2005010101	2	71110	1	1	175
2005121802	62	10140	352	0	700000
2005010201	2	41510	2	1	-85967625
2005010201	3	20670	2	1	3245432
2005121104	9	20210	345	1	440000
2005010201	5	20210	2	1	1386439459
2005	6	20110	2	1	1303488548
2005090502	3	20110	248	1	9900000
2005090502	6	20110	248	1	1912680
2005090502	9	20110	248	1	2013000
2005090502	12	20110	248	1	11330000
2005090502	15	20110	248	1	2112000
2005090502	18	20110	248	1	4290000
2005090502	21	20110	248	1	880000

그림 19. 대변 회계 사실 테이블
Figure 19. Fact Table of a Credit Account

본 회계 시스템에서는 금액이 없는 즉, 0값을 가진 행들이 가지는 문제점을 해결하기 위해서 차변회계와 대변회계를 분리하여 분석하고 있다. 참고로 차변금액과 대변금액의 총금액은 같다. (그림 16)에서 (그림 19)까지의 데이터를 이용하여 OLAP의 큐브를 만들면 각각 (그림 20)과 (그림 21)의 스키마 구조를 가지며 각각의 큐브 브라우저는 (그림 22)와 (그림 23)과 같다.

그림 20. 차변 회계 큐브 스키마
Figure 20. Cube Schema for a Debt Account

그림 21. 대변 회계 큐브 스키마
Figure 21. Cube Schema for a Credit Account

그림 22. 차변 회계 큐브 브라우저
Figure 22. Cube Browser for a Debt Account

그림 23. 대변 회계 큐브 브라우저
Figure 23. Cube Browser for a Credit Account

회계분석팀은 2005년 4분기의 부채계정 중 유동부채의 총 금액이 얼마가 되는가의 간단한 분석에서부터 각 수준별 멤버의 1회당 평균 금액과 분석 기간에 따른 각 계정의 비율과 같은 공식을 사용하는 복잡한 분석을 하기 위해서 OLAP의 MDX 편집기를 사용하기로 하였다.

2005년 4분기에서의 자본계정의 이익잉여금, 자산계정의 당좌자산, 부채계정의 유동부채의 금액이 얼마인지를 알고자 한다. MDX의 표현식은 (그림 24)와 같다.

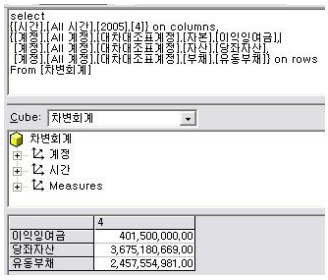
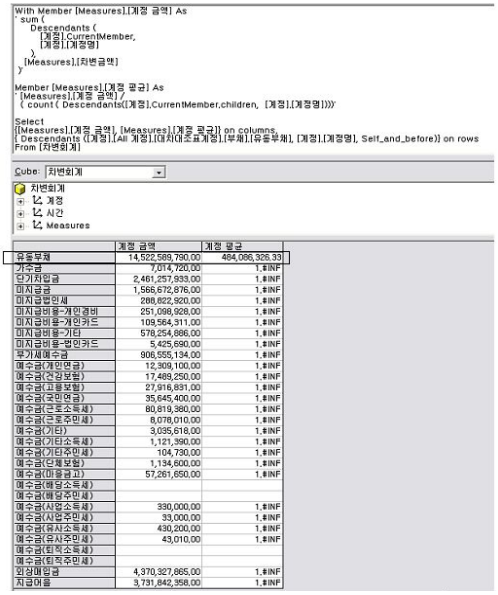


그림 24. 이익잉여금, 당좌자산, 유동부채의 금액
Figure 24. An Operating Surplus, Quick Assets, and Floating Liabilities



```
With Member [Measures].[계정 금액] As
' sum (
    Descendants (
      [계정].CurrentMember,
      [계정].[계정명]
    ),
  [Measures].[차변금액]
)

Member [Measures].[계정 평균] As
' [Measures].[계정 금액] /
 ( count ( Descendants([계정].CurrentMember,children,  [계정].[계정명]))

Select
{[Measures].[계정 금액], [Measures].[계정 평균]} on columns,
{ Descendants ([계정].[All 계정],[대차대조표계정],[부채],[유동부채], [계정].[계정명], Self_and_before)} on rows
From [차변회계]
```

Cube: 차변회계

차변회계
- 계정
- 시간
- Measures

	계정 금액	계정 평균
유동부채	14,522,589,790.00	464,066,326.33
가수금	7,014,720.00	1.#INF
단기차입금	2,461,257,933.00	1.#INF
미지급금	1,566,672,876.00	1.#INF
미지급법인세	286,822,920.00	1.#INF
미지급비용-개인경비	251,098,928.00	1.#INF
미지급비용-개인카드	109,564,311.00	1.#INF
미지급비용-기타	578,254,886.00	1.#INF
미지급비용-법인카드	5,425,690.00	1.#INF
부가세예수금	906,555,134.00	1.#INF
예수금(개인연금)	12,309,100.00	1.#INF
예수금(건강보험)	17,489,250.00	1.#INF
예수금(고용보험)	27,916,831.00	1.#INF
예수금(국민연금)	35,645,400.00	1.#INF
예수금(근로소득세)	80,819,380.00	1.#INF
예수금(근로주민세)	8,078,010.00	1.#INF
예수금(기타)	3,035,618.00	1.#INF
예수금(기타소득세)	1,121,390.00	1.#INF
예수금(기타주민세)	104,730.00	1.#INF
예수금(단체보험)	1,134,600.00	1.#INF
예수금(미을금고)	57,261,650.00	1.#INF
예수금(배당소득세)		
예수금(배당주민세)		
예수금(사업소득세)	330,000.00	1.#INF
예수금(사업주민세)	33,000.00	1.#INF
예수금(유사소득세)	430,200.00	1.#INF
예수금(유사주민세)	43,010.00	1.#INF
예수금(퇴직소득세)		
예수금(퇴직주민세)		
외상매입금	4,370,327,865.00	1.#INF
지급어음	3,731,842,358.00	1.#INF

그림 25. 유동부채의 평균 금액 산정을 위한 MDX 식
Figure 25. An MDX Query for Calculating Floating Liabilities Average

분석팀은 대차대조표계정 중 부채 계정 중 유동부채의 평균 금액을 산정하고자 MDX 식을 이용하여 분석을 시도하였다. MDX 식은 (그림 25)와 같다.

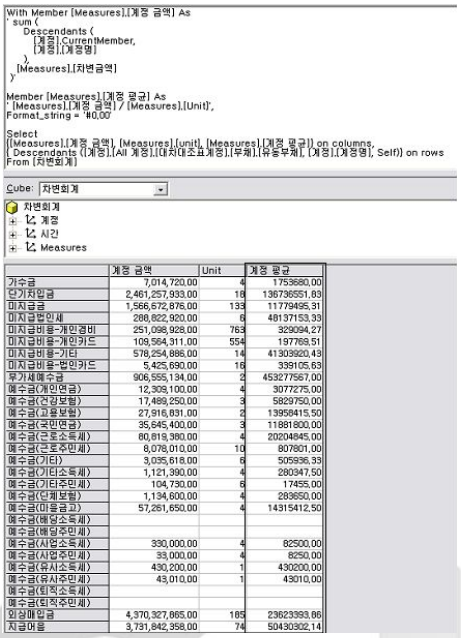
(그림 25)의 결과에서처럼 부채나 유동부채처럼 하위 수준 멤버들을 가지는 멤버는 그 평균을 구할 수 있지만 하위 수준이 없는 리프 멤버는 결과를 산출할 수 없다는 확인하였다. 그러나 분석팀은 사실테이블에는 각 리프 계정을 구성하고 있는 인스턴스들이 존재한다는 사실을 확인하였고 리프멤버 자체도 분석할 수 있을 뿐 아니라 수준별 각 멤버의 가중평균R지도 구할 수 있는 방법을 연구하게 되었다. 그 결과로써 유니트 컬럼을 사실 테이블에 추가하는 방법과 MDX 작성법을 개발하였다(그림 26참조).

전표번호	전표순번	계정코드	시간 코드	차대구분	차변금액	unit
2005010101	1	10140	1	0	175	1
2005102401	21	20610	297	1	20000	1
2005010201	1	10440	2	0	-6596625	1
2005103102	43	20110	304	1	30800000	1
2005101802	1	10440	291	0	193344253	1
2005010201	4	10620	2	0	3245432	1
2005092803	15	20590	271	1	36440	1
2005[illegible]	[illegible]	51100	293	0	5467**	1
2005091506	2	57210	258	0	800000	1
2005121505	1	58510	349	0	30000	1
2005121505	2	58510	349	0	220000	1
2005121505	3	58510	349	0	120000	1
2005121505	4	58510	349	0	188500	1
2005090907	3	51710	252	0	27216000	1

그림 26. 유니트 컬럼을 추가한 차변회계 사실 테이블
Figure 26. Fact Table with Unit Column for a Debtor Account

본 연구에서 추가한 유니트 컬럼을 사용하여 구한 리프 멤버의 평균은 (그림 27)과 같다.

(그림 27)의 결과에서처럼 유니트 컬럼을 사용하면 유동부채에 해당하는 리프멤버 각각의 인스턴스에 대한 올바른 평균을 구할 수 있다는 것을 확인하였다.



```
With Member [Measures].[계정 금액] As
' sum (
    Descendants (
      [계정].CurrentMember,
      [계정].[계정명]
    ),
  [Measures].[차변금액]
)

Member [Measures].[계정 평균] As
' [Measures].[계정 금액] / [Measures].[Unit]',
Format_string = '#0.00'

Select
{[Measures].[계정 금액], [Measures].[unit], [Measures].[계정 평균]} on columns,
{ Descendants ([계정].[All 계정],[대차대조표계정],[부채],[유동부채], [계정].[계정명], Self)} on rows
From [차변회계]
```

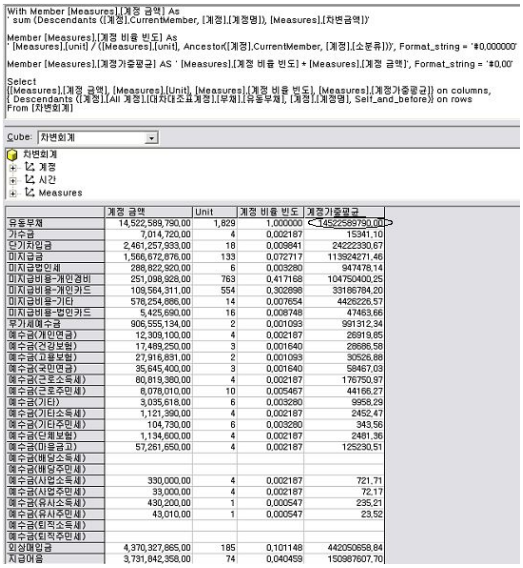
Cube: 차변회계

차변회계
- 계정
- 시간
- Measures

	계정 금액	Unit	계정 평균
가수금	7,014,720.00	4	1753680.00
단기차입금	2,461,257,933.00	18	136736551.83
미지급금	1,566,672,876.00	133	11779495.31
미지급법인세	286,822,920.00	6	48137153.33
미지급비용-개인경비	251,098,928.00	763	329094.27
미지급비용-개인카드	109,564,311.00	554	197769.51
미지급비용-기타	578,254,886.00	14	41303920.43
미지급비용-법인카드	5,425,690.00	16	339105.63
부가세예수금	906,555,134.00	2	453277567.00
예수금(개인연금)	12,309,100.00	4	3077275.00
예수금(건강보험)	17,489,250.00	3	5829750.00
예수금(고용보험)	27,916,831.00	2	13958415.50
예수금(국민연금)	35,645,400.00	3	11881800.00
예수금(근로소득세)	80,819,380.00	4	20204845.00
예수금(근로주민세)	8,078,010.00	10	807801.00
예수금(기타)	3,035,618.00	6	505936.33
예수금(기타소득세)	1,121,390.00	4	280347.50
예수금(기타주민세)	104,730.00	6	17455.00
예수금(단체보험)	1,134,600.00	4	283650.00
예수금(미을금고)	57,261,650.00	4	14315412.50
예수금(배당소득세)			
예수금(배당주민세)			
예수금(사업소득세)	330,000.00	4	82500.00
예수금(사업주민세)	33,000.00	4	8250.00
예수금(유사소득세)	430,200.00	1	430200.00
예수금(유사주민세)	43,010.00	1	43010.00
예수금(퇴직소득세)			
예수금(퇴직주민세)			
외상매입금	4,370,327,865.00	185	23623393.86
지급어음	3,731,842,358.00	74	50430302.14

그림 27. 리프멤버의 평균을 구하는 MDX 식
Figure 27. An MDX Query for Calculating Leaf Members Average

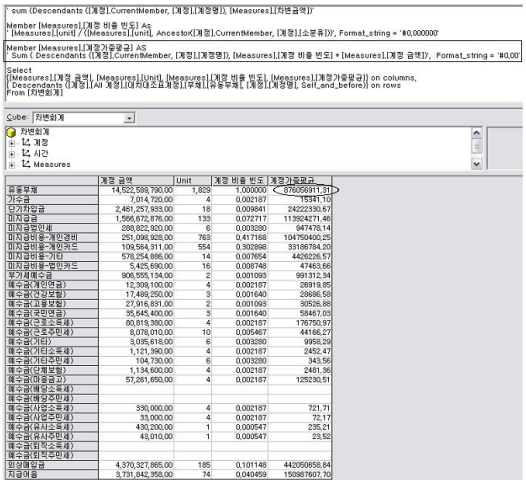
회계분석팀은 (그림 27)의 결과로부터 한 가지 값에 대한 의구심이 들기 시작하였다. (그림 25)에서 구한 유동부채의 평균이 과연 올바른 평균이라고 할 수 있겠는가라는 생각이 들었기 때문이다. 유동부채는 30개의 멤버로 구성되어 있다. 따라서 일반적으로 유동부채의 평균은 유동부채의 총금액을 멤버의 개수인 30으로 나눈 값이 산정된다. 그러나 사실 테이블로부터 유동부채를 구성하는 멤버들의 빈도수와 금액을 확인한 결과 이 값들은 현저하게 차이가 난다는 것이 확인되었다. 그런 이유로 유동부채나 그 상위 수준의 평균은 동등한 값으로 나누는 산술평균으로서는 평가되어서는 안된다는 결론에 도달하였다. 따라서 분석팀은 유동부채의 평균은 그것을 구성하는 각 계정의 빈도수에 비례하는 가중치를 부여한 가중평균으로 구해야만이 올바른 평균을 구하는 것이라고 결정짓고 연구에서 제시한 유니트 컬럼을 이용하여 유동부채를 기준으로 유동부채에 해당하는 각 계정의 빈도수에 대한 비율을 구하고 구한 각각의 비율을 각 멤버의 금액에 곱하여 가중평균을 구하였다. (그림 28)은 상기 절차의 MDX식을 나타낸 것이다.



	계정 금액	Unit	계정 비율 빈도	계정가중평균
유동부채	14,522,589,790.00	1,829	1.000000	7945225897900
가수금	7,014,720.00	4	0.002187	15341.10
단기차입금	2,461,257,933.00	18	0.009841	24222330.67
미지급금	1,566,672,876.00	133	0.072717	113924271.46
미지급법인세	288,822,920.00	6	0.003280	947476.14
미지급비용-개인경비	251,098,928.00	763	0.417168	104750400.25
미지급비용-개인카드	109,564,311.00	554	0.302898	33186784.20
미지급비용-기타	578,254,886.00	14	0.007654	4426226.57
미지급비용-법인카드	5,425,690.00	16	0.008748	47463.66
부가세대수금	906,555,134.00	2	0.001093	991312.34
예수금(개인연금)	12,309,100.00	4	0.002187	26919.85
예수금(건강보험)	17,489,250.00	3	0.001640	28686.58
예수금(고용보험)	27,916,831.00	2	0.001093	30526.88
예수금(국민연금)	35,645,400.00	3	0.001640	58467.03
예수금(근로소득세)	80,819,360.00	4	0.002187	176750.97
예수금(근로주민세)	8,078,010.00	10	0.005467	44166.27
예수금(기타)	3,035,618.00	6	0.003280	9958.29
예수금(기타소득세)	1,121,390.00	4	0.002187	2452.47
예수금(기타주민세)	104,730.00	6	0.003280	343.56
예수금(단체보험)	1,134,600.00	4	0.002187	2481.36
예수금(미들금고)	57,261,650.00	4	0.002187	125230.51
예수금(배당소득세)				
예수금(배당주민세)				
예수금(사업소득세)	330,000.00	4	0.002187	721.71
예수금(사업주민세)	33,000.00	4	0.002187	72.17
예수금(유사소득세)	430,200.00	1	0.000547	235.21
예수금(유사주민세)	43,010.00	1	0.000547	23.52
예수금(퇴직소득세)				
예수금(퇴직주민세)				
외상매입금	4,370,327,865.00	185	0.101148	442050658.84
지급어음	3,731,842,358.00	74	0.040459	150987607.70

그림 28. 잘못 계산한 유동부채의 가중평균
Figure 28. Weighted Average in Error for Floating Liabilities

(그림 28)의 결과에서처럼 유동부채를 1로 보고 유동부채를 구성하는 멤버들 각각의 빈도수에 대한 비율을 계정금액에 곱하여 계정가중평균을 산정한 것을 확인할 수 있다. 그러나 유동부채의 계정가중평균 금액인 14,522,589,790원은 올바른 가중평균의 금액이 아니라는 것이 확인 되었다. 올바른 유동부채 계정의 가중평균이란 유동부채를 구성하는 각 멤버 즉, 가수금부터 지급어음에 해당되는 30개의 계정가중평균을 모두 합한 금액이 산출되어야 한다. 그러나 본래 with member에 의해서 만들어진 측정값은 그 측정값을 산정하기 위한 공식에 의해서 금액을 산출하므로 유동부채의 총금액에 유동부채의 비율 1을 곱한 값이 산정되어 틀린 값을 가져오게 되었다는 것을 확인하였다. 따라서 분석팀은 오랜 MDX 구문 연구에 끝에 Sum과 Descendants를 적합하게 활용한 MDX식을 완성하였다. 그 MDX 식과 결과는 (그림 29)에서 보여 진다.



	계정 금액	Unit	계정 비율 빈도	계정가중평균
유동부채	14,522,589,790.00	1,829	1.000000	876056911.31
가수금	7,014,720.00	4	0.002187	15341.10
단기차입금	2,461,257,933.00	18	0.009841	24222330.67
미지급금	1,566,672,876.00	133	0.072717	113924271.46
미지급법인세	288,822,920.00	6	0.003280	947476.14
미지급비용-개인경비	251,098,928.00	763	0.417168	104750400.25
미지급비용-개인카드	109,564,311.00	554	0.302898	33186784.20
미지급비용-기타	578,254,886.00	14	0.007654	4426226.57
미지급비용-법인카드	5,425,690.00	16	0.008748	47463.66
부가세대수금	906,555,134.00	2	0.001093	991312.34
예수금(개인연금)	12,309,100.00	4	0.002187	26919.85
예수금(건강보험)	17,489,250.00	3	0.001640	28686.58
예수금(고용보험)	27,916,831.00	2	0.001093	30526.88
예수금(국민연금)	35,645,400.00	3	0.001640	58467.03
예수금(근로소득세)	80,819,360.00	4	0.002187	176750.97
예수금(근로주민세)	8,078,010.00	10	0.005467	44166.27
예수금(기타)	3,035,618.00	6	0.003280	9958.29
예수금(기타소득세)	1,121,390.00	4	0.002187	2452.47
예수금(기타주민세)	104,730.00	6	0.003280	343.56
예수금(단체보험)	1,134,600.00	4	0.002187	2481.36
예수금(미들금고)	57,261,650.00	4	0.002187	125230.51
예수금(배당소득세)				
예수금(배당주민세)				
예수금(사업소득세)	330,000.00	4	0.002187	721.71
예수금(사업주민세)	33,000.00	4	0.002187	72.17
예수금(유사소득세)	430,200.00	1	0.000547	235.21
예수금(유사주민세)	43,010.00	1	0.000547	23.52
예수금(퇴직소득세)				
예수금(퇴직주민세)				
외상매입금	4,370,327,865.00	185	0.101148	442050658.84
지급어음	3,731,842,358.00	74	0.040459	150987607.70

그림 29. 올바른 유동부채의 가중평균
Figure 29. Weighted Average for Floating Liabilities

(그림 29)의 결과에서처럼 유동부채의 올바른 가중평균 값인 876,056,911.31원을 산정할 수 있었다.

V. 결론

다차원 데이터베이스는 비즈니스 의사결정지원 분야에서 중요시 여겨지는 요소이다. 다차원 데이터베이스에서는 데이터를 비즈니스 구조에 맞게 설계하고 저장한다. 다차원 데이터베이스를 설계하는 가장 보편화된 방법인 스타 스키마는 사실 테이블의 기본키와 측정값이 1 : N의 관계를 맺고 있는 경우에는 표현이 불가능하기 때문에 기존 방법에서는 선처리 과정을 거쳐야만 하였다. 그러나 이 방법은 인스턴스의 정보를 알 수 없게 되므로 결과적으로 정보손실이 생긴다.

본 연구에서 제시한 유니트 컬럼과 MDX 작성법을 사용하면 사례와 같이 회계전표의 각 계정별 금액을 산출하는 유형의 문제에 대한 해결이 가능하다는 것을 확인할 수 있었으며 인스턴스의 1 : N의 관계를 나타냄으로써 얻어지는 이점과 함께 수준별 각 멤버의 올바른 평균인 가중평균을 구할 수 있다는 것을 확인할 수 있었다.

본 연구에서 제시하는 방법으로 데이터베이스나 데이터웨어하우스를 설계한다면 OLAP 큐브를 생성하는데 필요했었던 선처리 과정을 거치지 않고도 원본 데이터를 그대로 분석 작업에 사용하여 간단하고 올바른 분석값을 산출 할 수 있게 된다. 이러한 점에서 본 연구에서 제시한 방법은

현대 사회에서처럼 수많은 분석 작업이 이루어지는 환경에서 기업의 시간과 비용에 대한 부담을 보다 줄여 줄 수 있을 것을 것이라고 생각하며 아울러 OLAP의 적용분야를 더욱 넓힐 수 있을 것으로 예상한다.

참고문헌

[1] Thomsen, E., "OLAP Solutions", 2nd Edition, pp.375-386, Wiley, 2002.

[2] Spofford, G., et al., "MDX Solutions", pp. 295-305, Wiley, 2001.

[3] Tang, Z. and MacLennan, J., "Data Mining with SQL Server 2005", pp.265-272, Wiley, 2005.

[4] Spofford, G., et al., "MDX Solutions", 2nd Edition, pp.471-477, Wiley, 2006.

[5] Craig, R.S., et al., "Microsoft Data Warehousing",pp.63-66, Wiley, 1999.

[6] Kroenke, D.M., "Database Processing", 10th Edition, pp.547-549, Prentice Hall,2006.

[7] Berson, A., et al., "Building Data Mining Applications for CRM", pp.89-92, McGraw Hill, 2000.

[8] Inman, W.H., "Building the Data Warhouse," 2nd Edition, pp.127-128, Wiley, 1996.

[9] Turban, E., et al., "Introduction To Information Technology", 5th Edition, pp.424-427, Wiley, 2006.

[10] Koutsoukis, N.S. et al., "Adopting on-line analytical processing for decision modeling : the interaction of information and decision technologies", Elsevier science B.V.,, Decision Support Systems, 26권, 1호, 1쪽 1999.

[11] 권오주, "OLAP Solutions +α SQL Server Analysis Services", pp.40-54, 대림, 2001.

[12] 조재희, 박성진, "OLAP 테크놀로지", Sigma Insight, 2003.

[13] 이덕근 등, "실 위치지정자 자격으로서의 멤버특성을 활용한 개인화 작업", 한국컴퓨터정보학회논문지, 제10권, 제3호, pp.101-110, 2005.

[14] 이덕근 등, "DW에서의 질의어처리 성능향상을 위한 데이터 구조화 방법", 한국컴퓨터정보학회논문지, 제10권, 제1호, pp.7-13, 2005.

저 자 소 개

유 한 주

2001년 남서울대학교 산업공학과 졸업(학사)

2004년 숭실대학교 MIS 졸업(공학석사)

2005년 ~ 현재 숭실대학교 산업·정보시스템공학과 박사과정 재학중

<관심분야> MIS, DW, OLAP, MDX, CRM

이 덕 성

1990년 전남대학교 산업공학과 졸업(학사)

1995년 전남대학교 산업공학 졸업(석사)

2005년 ~현재 숭실대학교 산업·정보시스템공학과 박사과정 재학중

<관심분야> MIS, DW, OLAP, MDX, CRM

최 인 수

1985년 서울대학교 산업공학 졸업(공학박사)

1980년 ~ 현재 숭실대학교 산업·정보시스템공학과 교수

<관심분야> MIS, DW, OLAP, MDX, CRM