OLAP 환경에서 스프레드시트와 피벗 테이블을 다루기 위한 SQL의 확장

신성현 [1,O] 김진호 [2] 문양세 [2] 김상욱 [3]
[1]한양대학교 전기전자통신기술연구소/[2]강원대학교 컴퓨터학부/[3]한양대학교 정보통신공학
shshin@agape.hanyang.ac.kr, {jhkim,ysmoon}@kangwon.ac.kr, wook@hanyang.ac.kr

SQL Extensions for Handling Spreadsheets and PIVOT tables in OLAP Environment

Sung-Hyun Shin[1,O], Jinho Kim[2], Yang-Sae Moon[2], and Sang-Wook Kim[3]
[1]Research Institute of Electrical and Computer Engineering, Hanyang University
[2]Department of Computer Science, Kangwon National University
[3]College of Information and Communications, Hanyang University

요 약

온라인 분석 처리(On-Line Analytical Processing: OLAP)은 데이터 웨어하우스로부터 다차원 데이터를 분석하거나 의사 결정을 위한 유용한 정보를 제공하고 있다. 데이터 분석을 위해, OLAP에서는 다차원 데이터를 표현한 스프레드시트(spreadsheet) 또는 피벗 테이블(PIVOT table)을 널리 사용하고 있다. 스프레드시트와 피벗 테이블은 서로 유사한 형태로써 분석의 기준이 되는 애트리뷰트들이 많은 구조이다. 사용자들은 흔히 사용되고 있는 SQL 구문을 이용하여 스프레드시트 또는 피벗 테이블에서 손쉬운 데이터 분석을 요구한다. 그러나, RDBMS에서 제공하는 SQL 구문의 사용으로, 이는 다차원 데이터를 효과적으로 분석할 수 없다. 그 이유는 SQL 구문이 다양한 데이터 분석의 목적으로 사용되거나, 요약된 집계 정보를 도출하는 데 한계가 있기 때문이다. 따라서, 본 연구에서는 SQL 구문을 확장하여 다차원 데이터를 표현한 스프레드시트를 손쉽게 조작하고, 요약된 집계를 계산하는 셀(cell) 구문을 제안한다. 이 방법은 스프레드시트와 피벗 테이블에서 행과 열이 교차하는 좌표(coordinate)를 이용하여, 특정 셀의 조작 및 선택한 부분/전체 영역에 대한 집계 정보를 계산하는 방법이다. 결과적으로, RDBMS에서 사용되는 SQL 구문이 친숙한 사용자들이 제안한 셀 구문을 이용하면, 다양한 관점에 따라 손쉽게 스프레드시트와 피벗 테이블을 다룰 수 있을 것으로 사료된다.

1. 서 론

온라인 분석 처리(On-Line Analytical Processing: OLAP)은 데이터 웨어하우스에 저장된 방대한 데이터를 다양한 정보로 추출하기 위해 다차원 데이터를 분석하는 기법이다[2]. OLAP에서는 이러한 데이터를 다양한 분석에 사용되는 표현 방법으로 **스프레드시트** (*spreadsheet*)와 **피벗 테이블** (*PIVOT table*)을 널리 사용하고 있다. 이들은 마이크로소프트 사의 엑셀에서 제공하는 형태와 비슷한 형태로써, 분석 기준이 되는 애트리뷰트들의 값을 테이블의 행과 열의 색인 값을 갖는 이차원 테이블로 표현된다[5].

OLAP에서는 사용자에게 요구한 질의의 분석 결과를 제공하기 위해, 스프레드시트와 피벗 테이블을 흔히 사용하고 있다. 하지만, 이들은 단순히 보여주는 보고서 역할로만 제공하고 있어, 적극적인(active) 분석을 요구하는 환경에는 어려움이 있다. 이에 따라, 스프레드 시트와 피벗 테이블을 이용하여 사용자에게 다양하고 편리한 데이터 분석을 지원하기 위해 다양한 방법들이 연구되었다. Witkowski 등[7]은 스프레드시트의 대표적인 예로 엑셀을 이용하고, 제안된 구문으로 엑셀의 셀을 조작할 수 있는 방법들을 제시하였다. Witkowski 등[8]은 스프레드시트의 다차원 배열 기반 분석 및 계산 기능을 접목하기 위해 SQL을 확장시키는 방법을 소개하였다. Agrawal 등[1]에서는 관계형 테이블에 저장된 테이블을 단순한 피벗 테이블 형태의 구조로 변환하는 방법을 제안하였다. 그러나, 이들 방법은 다차원 분석이 아닌 단순한 데이터 분석에 이르고 있으며, 스프레드시트에서 질의 처리시 SQL 구문이 복잡해지는 경향이 발생한다. 다차원 분석을 위해 OLAP 환경에서는 MDX 언어를 사용한다. 그러나, 이 언어는 큐브를 대상으로 사용되므로, 관계형 테이블을 사용하는 본 연구와 차이가 있다.

본 연구에서는 관계형 데이터베이스의 테이블에서 변환된 스프레드시트와 피벗 테이블의 데이터를 직접 조작할 수 있는 셀(cell) 구문을 제안한다. 제안한 방법은 SQL을 확장하여 스프레드시트에서 행과 열이 교차하는

좌표(*coordinate*)를 이용하는 방법으로, 간결하고 손쉽게 스프레드시트를 다룰 수 있다. 그러므로, 스프레드시트에서 사용자의 여러 관점에 따라 원하는 영역에서 다양한 집계 정보를 계산할 수 있다.

본 논문의 구성은 다음과 같다. 제2장에서는 본 연구와 관련된 기존 연구를 소개한다. 제3장에서는 스프레드시트의 개념과 이를 조작하기 위한 셀 구문을 제안한다. 제4장에서는 제안한 셀 구문을 통해 요약된 집계 정보를 계산하는 방법에 대해 설명한다. 마지막으로, 제5장에서는 결론을 맺는다.

2. 관련 연구

기존연구에서는 OLAP 응용에서 다차원 데이터를 사용자들에게 자연스럽게 표현하는 방법으로 스프레드시트 형태로 제시하였다[1,7,8]. 우선, 참고문헌[7,8]에서는 SQL문을 확장하여 관계형 DBMS에서 스프레드시트 형태의 테이블을 정의하고 조작할 수 있도록 하였다. 또한, Agrawal 등[1]에서는 RDBMS에 저장된 테이블을 단순한 피벗 테이블 형태의 테이블로 유도할 수 있는 연산을 제안하였다.

그러나, 스프레드시트 방식의 많은 장점에도 불구하고, RDBMS에서 제공하는 SQL은 OLAP 분석을 위해서 간단한 데이터의 요약된 집계 정보를 위한 질의를 복잡하게 만드는 문제점이 있다. 그 이유는 다양한 정보 분석을 목적으로 만들어지지 않았기 때문이다. 이를 해결하기 위해, Gray 등[5]은 RDBMS에서 다차원 데이터를 분석하기 위해 *Cube*와 *Rollup* 연산 등을 제안하여 기존 SQL 언어에 확장시킴으로 집계 데이터를 표현하였다. 또한 Cunningham 등[4]과 Sonting 등[3]은 기존 SQL 언어에 *PIVOT* 연산을 추가하여 저장된 소스 테이블의 행과 열을 변경하여 단순한 스프레드시트 형태로 제공하는 방법을 제안하였다. 그러나, 이들 연구는 복잡한 구문으로 작성해야 하므로 일반 사용자가 이용하기에는 매우 어렵고, 이에 따라 다양한 분석을 요구하는 OLAP 환경에는 적합하지 않다.

OLAP 환경에서는 효과적으로 다차원 질의를 작성할 수 있도록 MDX(Multi-Dimensional eXpressions)가 개발되었다[6]. 그러나, 사실 테이블(fact table)과 차원(dimensional table)로 구성된 큐브에 대하여 질의하는 언어이므로, 관계형 테이블에서 피벗 테이블을 대상으로 조작하는 본 연구와는 차이가 있다.

3. 시트 셀에 대한 주소 표기

본 장에서는 스프레드시트의 시트에 구성되는 셀을 조작 및 연산하기 위한 새로운 셀 절을 설명한다.

3.1 스프레드시트와 피벗 테이블의 개념

본 절은 OLAP에서 사용되는 관계형 테이블에서 스프레드시트와 피벗 테이블으로 변환 과정을 기반으로 설명한다. 스프레드시트와 피벗 테이블은 서로 유사한 형태이며 그림 1과 같이, 2차원 테이블 형태로 표현한 모델을 나타낸다. 본 논문에서는 스프레드시트와 피벗 테이블의 형태는 동일한 구조로 고려하며, 이들 용어는 혼용하여 사용한다.

이러한 스프레드시트와 피벗 테이블은 여러 시트들로 구성되며, 하나의 시트는 각각 지역, 제품, 기간 등과 같은 데이터 집합으로 구성된다. 그리고, 이들은 같은 성격을 갖는 데이터들끼리 모여서 축을 이루고 있으며, 이러한 축의 위치는 좌표(coordinate)라고 한다. 여기서, 사용되는 축은 행의 축과 열의 축으로 구분된다. 특히, 행의 축은 사용자의 분석 관점을 나타내는 집합으로 구분하고, 열은 분석하고자 하는 측정 기준인 계산 가능한 측정값으로 구분한다.

예를 들어, 그림 1은 두 개의 축으로 이루어진 데이터 집합인 시트를 보여주고 있다. 행 축은 *Country*와 *Product* 차원으로, 열 축은 *Year* 축으로 구성되어 있다. 행 축의 좌표는 (*Korea,TV*), (*Korea,Radio*), …, (*Japan, RAM*)과 같이 *Country*와 *Product*의 구성원을 이용하여 구성된다. 또한, 열 축의 좌표는 (*2003,FH*), (*2003,LH*), …, (*2007,LH*)로 구성된다. 또한, 행과 열 축의 교차점을 **셀**(*cell*)이라고 한다. 셀은 여러 가지 정보를 가지고 있다. 그림에서 셀이 가지는 정보는 *Product*의 총 판매량이다.

행 축 – Country, Product　　열 축 – Year, halfterm

Country	Product	2003		2004		2005		2006		2007	
		FH	LH	FH	LH	FH	LH	FH	LH	FH	LH
Korea	TV	18	12	21	16	15	17	19	22	20	22
Korea	Radio	12	15	14	13	25	16	16	13	19	23
Korea	RAM	8	12	12	17	21	19	14	24	17	19
Japan	TV	15	21	17	20	13	22	17	10	18	21
Japan	Radio	10	18	24	22	14	12	20	17	21	16
Japan	RAM		16	23	19	17	23	18	19	22	14

시트 – Product, Measure　　셀

FH: first half of the year
LH: latter half of the year

그림 1. 시트에 구성된 데이터 집합(*ProductSh*).

다음으로, 이러한 피벗 테이블을 관계형 데이터베이스에 실제 저장된 관계형 테이블 구조로 변환할 수 있는 방법을 소개한다. 피벗 테이블은 구조와 형태가 다르기 때문에 관계형 테이블로 직접 저장할 수 없다. 그 이유는 피벗 테이블은 분석의 기준이 되는 애트리뷰트의 수가 많기 때문에, 컬럼의 제약(최대 1,024개)이 있는 관계형 테이블로 직접적으로 저장할 수 없으므로, 이를 처리하는 방법이 필요하다. 이에 따라, Chen 등[3]은 관계형 테이블에서 피벗 테이블로 변환하는 방법을 관계 대수식으로 정의하였다.

$$HT = [\bowtie_{i=1}^{n} \pi_{Oid,M}(\sigma_{A='A_j'} VT)] \Rightarrow \mathrm{PIVOT}_{A \text{ on } M}^{[A_1,\ldots,A_k]}(VT)$$

여기서, 피벗 테이블 *PT*는 (*Oid*, A_1, A_2, …, A_k)의 스키

마를 가지며, 관계형 테이블 VT는 (Oid, A, M)의 스키마를 가지며, VT에 저장되는 튜플은 (O_i, A_i, M_i)로 표현된다. 위의 식을 보면, 관계형 테이블 RT를 애트리뷰트 A의 값에 대해 PIVOT을 하면, 피벗테이블로 표현됨을 알 수 있다. 이에 따라, 본 연구에서는 관계형 테이블에서 변환된 피벗 테이블을 직접 조작하여 처리할 수 있는 방법을 제시한다.

3.2 셀 구문의 정의 및 소개

본 절에서는 제안한 셀 절을 정의하고, 셀 조작을 위한 구문을 소개한다.

```
<the query block for SELECT-FROM-WHERE>
  SELECT          <attribute_list>
  FROM            <table_list>
<the structure of the CELL clause>
  WHERE           (<formula>, <formula>, …, <formula>)
  formula = function
  function = (cell_position = function_name (cell_position_list))
  cell_position_list = cell_position | cell_postion : cell_position
  cell_position = sheet_name[parameter]
  parameter = [first_parameter, last_parameter]
  first_parameter = row_position
  last_parameter = column_position
```

그림 2. 시트 조작을 위한 SQL CELL 구문.

셀 조작을 위한 구문은 SQL 구문을 확장하여 그림 2와 같이 정의한다. 이러한 구문은 기존의 SQL 구문과 시트를 조작하기 위해 WHERE 구문을 분리하여 처리한다. 이러한 셀 구문을 이용한 질의는 다음의 정보를 포함해야 한다.

- SQL문의 이용: 정보를 검색하는 기본 문으로 SELECT-FROM-WHERE 블록을 지정한다.
- 시트(테이블) 명: FROM 절 외에, 참조하는 시트 명을 WHERE 절에 기술한다.
- 셀 주소 표기: 셀의 주소를 명시하기 위해, 행과 열의 축으로 주소 위치를 명시한다.

셀 주소를 지정하는 방법은 다음과 같이 표현할 수 있다. 우선, 하나의 셀을 지정하기 위한 방법은 *Sheet* [*ridx:cidx*]으로 표현할 수 있다. 여기서, 행 *ridx*와 열 *cidx*를 축으로 셀 주소를 표현한다. 다음으로, 셀의 범위를 주어 지정하기 위한 방법은 *Sheet* ([$ridx_i$,$cidx_j$]:[$ridx_m$, $cidx_n$])으로 표현할 수 있다. 여기서, $ridx_i$ < $ridx_j$이며, $cidx_j$ < $cidx_n$이다. 마지막으로, 각 축에 대해 셀 전체를 지정하는 방법은 *Sheet*[*ridx*,*], *Sheet* [*,*cidx*]이며, 전체 셀을 지정하는 방법은 *Sheet* [*,*]으로 표현할 수 있다. 표1은 본 논문에서 사용하는 일반적인 표기와 이에 대한 정리 및 의미를 정리한 것이다.

표 1. 주요 표기법

기호	정의/의미
Sheet	지정된 시트(테이블)의 이름
Sheet[$ridx_i$: $cidx_j$]	시트 *Sheet* 의 행 $ridx_i$ 과 열 $cidx_j$ 에 대한 셀 지정
$ridx_i$, $cidx_j$	$ridx_i$ 는 다수 행의 멤버($ridx_i = \{row_{i1}, row_{i2} ... row_m\}$) $cidx_j$ 는 다수 열의 멤버($cidx_j = \{col_{j1}, col_{j2} ... col_{jn}\}$)
Sheet([$ridx_i$, $cidx_j$] : [$ridx_m$, $cidx_n$])	i 번째 행과 j 번째 열에서 m 번째 행과 n 번째 열까지 구성된 셀 부분 지정 ($ridx_i < ridx_m$, $cidx_j < cidx_n$)
Sheet[$ridx_i$, *], *Sheet*[*, $cidx_j$]	i 번째 행과 모든 열로 구성된 셀 부분 지정하거나, 모든 행과 j 번째 열로 구성된 셀 부분 지정

다음으로 시트의 셀을 간단히 계산하여 처리할 수 있는 셀 구문을 설명한다. 그림 3는 사용자 질의에 대한 예를 나타낸다. 우선, Q1은 "2007년에 국가 *Korea*에서 생산하는 *TV*의 판매금액은 2008년에는 50% 상승한 판매금액"라는 의미이다. 여기서, 우변의 계산하는 식 *Productsh*[*Krea.TV*,2007] 1.5은 지정된 좌변 *ProductSh*[*Korea.TV*,*20 08*] 에 결과 값을 저장한다. Q2는 "2006년도와 2007년도에 국가 *Korea*에서 판매하는 *RAM*의 판매금액을 총 금액"라는 의미이다. 여기서, 오른편 *Productsh* [*Korea. RAM*,2006]+*Productsh* [*Korea.RAM*,2007] 의 결과 값은 왼편 *Productsh*[*Korea. RAM*,2008]의 셀에 결과 값을 저장한다. 그리고, Q3은 "2004년에서 2006년도에 판매되는 *Radio*의 총금액을 계산하여 2008년도에 판매금액"라는 의미이다.

```
SELECT   Country, Product, 2003, 2004, 2005, 2006, 2007
FROM     Products
WHERE    (
   Q1 : Productsh[Korea.TV, 2008] = Productsh[Korea.TV, 2007]×1.5,
   Q2 : Productsh[Korea.RAM, 2008] = Productsh[Korea.RAM, 2006]
                                   +Productsh[Radio.RAM, 2007]
   Q3 : Productsh[Japan.Radio, 2008] =
        avg(Productsh[Japan.Radio, 2005]: Productsh[Japan.Radio, 2007])
)
```

그림 3. 셀 조작을 위한 질의의 예.

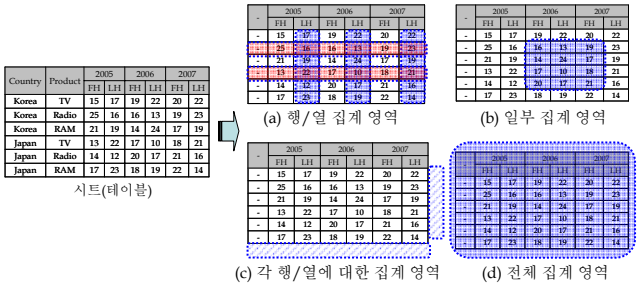


그림 4. 집계를 위한 여러 영역.

4. 셀 구문을 이용한 집계 영역 계산

본 장에서는 열과 행의 축으로 구성하는 시트에서 여러 관점으로 집계를 계산하는 방법을 설명한다.

4.1 행/ 열 부분 집계 연산

본 절에서는 행 또는 열의 축으로 부분 셀 영역을 지정하여 집계를 계산하는 방법을 소개한다. 우선, 특정한 행에 대하여 집계하는 구문을 설명한다. 열의 전체 수가 n개라고 할 때, 특정한 행의 축으로 전체 열의 값을 계산하기 위해서는 $sheet[ridx_i,cidx_j]$: $sheet[ridx_i,cidx_n]$ (여기서, $j<n$)으로 표현되거나, 표1에 따라 간단히 $sheet[ridx_i,*]$으로 표현이 가능하다. 예를 들면, *ProductSh*[*Korea.TV*, 2003.*FH*]:*ProductSh*[*Korea.TV*,2007.*LH*]이며, 간단히 *ProductSh* [*Korea.TV*,*]으로 표현된다. 또한, 제안한 주소 표기법을 사용하면 특정한 행에 대하여 전체 값이 아닌, 일부분의 값을 계산할 수 있다. 이는 $sheet[ridx_i,cidx_j]$: $sheet[ridx_{k+1},cidx_n]$ (여기서, $i<k+1$)으로 표현할 수 있다. 아래의 구문은 오른편에서 계산된 결과값을 왼편에서 지정된 셀 위치로 저장된다. 여기에서, *All*은 왼편의 식에 따라 행의 축을 기준으로 마지막 열 다음에 추가된 집계 결과값을 표현하기 위해 사용된다. 아래의 구문은 "2003년부터 2007년에 *Korea*에서 판매하는 *TV*의 총 판매수량"이라는 의미를 기술한 예이다.

```
WHERE (
    ProductSh[Korea.TV, All.All] =
        SUM(ProductSh[Korea.TV, 2003.FH] : ProductSh[Korea.TV, 2007.LH])
)
```

다음은 특정한 열의 축으로 집계하는 구문을 설명한다. 특정한 행의 축으로 집계를 계산하는 방법뿐만 아니라, 특정한 열에 대해서도 집계 계산이 가능하다. 튜플의 전체 수가 m개라고 할 때, 특정한 열의 전체 셀 값을 계산하기 위해서는 $sheet[ridx_i,cidx_j]$:$sheet$ $[ridx_m,cidx_j]$ (여기서, $i<m$)으로 표현되거나, 간단히 $sheet[*,cidx_j]$ 으로 표현이 가능하다. 예를 들면, "2007년 상반기에 판매되는 모든 지역과 모든 제품의 총 판매 수량"의 의미로는 *ProductSh*[*Korea.TV*,2007.*FH*]:*ProductSh*[*Japan.RAM*,2007.*FH*]이며, 간단히 *ProductSh* [*,2007.*FH*]으로 표현된다. 또한, 하나 이상의 열 부분 집계를 계산하는 방법을 계산할 수 있으며, 이러한 계산은 $sheet[ridx_i,cidx_j]$:$sheet$ $[ridx_m, cidx_{k+1}]$ (여기서, $j<k+1$)으로 표현할 수 있다. 아래의 구문은 "2007년 상반기에 판매되는 모든 지역과 모든 제품의 총 판매수량"이라는 의미로 기술한 예이다. 여기서, 왼편의 *All.All*은 새로 생성된 행 축의 좌표이며, 마지막 열 값의 다음으로 새로 결과값을 저장한다.

```
WHERE (
    ProductSh[All.All, 2007.FH] =
        SUM(ProductSh[Korea.TV, 2007.FH] : ProductSh[Japan.RAM, 2007.FH])
)
```

4.2 특정 부분 집계 연산

특정한 셀 영역을 지정하여 집계 연산을 수행하는 방법은 여러 방법들로 수행될 수 있다. 우선, 2차원 영역을 계산하는 방법과 인접하지 않는 행 또는 열의 부분 영역을 선택하여 집계를 계산하는 방법 등이 있다. 우선, 행과 열 축의 쌍의 교차 영역으로 구성된 2차원 서브 영역을 집계 계산하는 방법을 설명한다. RDBMS에서는 이러한 영역에 대한 집계를 계산하기 어렵다. 이에 따라, 2차원 작은 영역의 전체 값을 계산하기 위해서는 $sheet[ridx_i,cidx_j]$:$sheet[ridx_{i+k}, cidx_{j+l}]$ (여기서, $i<i+k<m$, $j<j+l<n$)으로 표현이 가능하다. 예를 들어, 질의 "2004년부터 2006년까지 *Korea*에서 판매하는 *TV, Radio, RAM*의 판매 수량"은 다음과 같이 셀 구문으로 표현한다.

```
WHERE (
    ProductSh[All.All, 2007.FH] =
        SUM(ProductSh[Korea.TV, 2004.FH] : ProductSh[Korea.RAM, 2006.FH])
)
```

다음은 인접하지 않는 행 또는 열의 축으로 부분 셀 영역을 지정하여 집계를 계산하는 방법을 소개한다. 이 방법은 애트리뷰트의 동일한 값인 튜플을 그룹화하는 SQL에서 제공하는 GROUP BY 절과 유사하게 행의 축으로 또는 열의 축으로 그룹화할 수 있다. 이에 따라, 본 연구에서 제안하는 셀 구문은 이와 유사한 기능을 확장하여 기술할 수 있다. 이는 모든/일부 행에 대하여 특정 열들의 값을 기준으로, 이 부분 집단들에 대하여 독립적으로 집계함수를 적용할 수 있다. 예를 들어, 특정 지역에서 판매되는 모든/일부 제품의 총 판매량을 구한다고 하자. 이런 경우, 특정 지역을 위한 별도의 그룹을 만들고, 모든/일부 제품에 대하여 집계 함수를 적용한다. 우선, 행의 축으로 그룹화하는 표현 방법을 설명한다. 기본적인 주소 표기법는 $sheet[*,cidx_j]$으로 표현된다. 여기서, 행의 축이 여러 값으로 이루어질 경우는 구분자 '.'을 이용하여 상위 또는 하위 부분을 구별하여 그룹화할 수 있다. 예를 들어, 질의 "2007년 상반기에 *Korea*에서 판매하는 모든 *Product*의 총 판매수량"는 SUM(*Productsh* [*Korea.**,2007.*FH*]) 으로 표현된다. 여기서, *Korea.**는 행의 좌표로서 명시된 {*Korea.TV, Korea.Radio, Korea.RAM*}을 의미한다. "2007년 상반기에 *TV*를 *Country*에서 판매하는 모든 *TV*의 총 판매수량"은 SUM(*Productsh*[*.*TV*,200 7.*FH*])로 표현할 수 있다. 여기서, 행의 축 좌표 *.*TV*는 {*Korea.TV, Japan. TV*}를 의미한다.

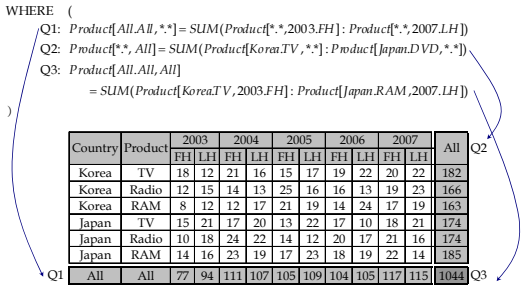
```
WHERE (
    Productsh[Korea.RAM, All] = SUM(Product[Korea.TV, 2007.*]),
    Productsh[Japan.TV, All] = SUM(Product[*.TV, 2006.FH] : [*.TV, 2007.LH])
)
```

다음은 열의 축으로 그룹화하는 주소 표기법의 방법을 설명한다. 행의 축에서 그룹화하는 방법과 유사하게 표현되며, 기본적인 주소 표기법은 $sheet[ridx_i,*]$ 이다. 예를 들면, 질의 "2007년에 *Korea*에서 판매되는 *TV*의 총 판매수량"은 SUM(*Productsh*[*Korea.TV*,2007.*FH*]: *Productsh* [*Korea.TV*,2007.*LH*]) 이다. 또한, 표1에 따라 그룹화 표시 '*'을 이용하여 간결히 SUM(*Productsh*[*Korea.TV,2007.**])

으로 표현된다. 그리고, 질의 "모든 해의 하반기에 *Korea*에서 판매되는 *TV*의 총 판매수량"은 SUM(*Productsh*[*Korea.TV*,2003.*FH*]:*Productsh*[*Korea.TV*,2007.*LH*])이며, 간결히 SUM (*Productsh*[*Korea.TV*,*.*LH*]) 으로 간단히 표현될 수 있다.

WHERE (
Product[*All.All*,2007.*LH*]
= *SUM*(*Product*[*Korea.TV*,2007.*FH*]:Pr*oduct*[*Korea.TV*,2007.*LH*]),
Product[*Korea.TV*,*All*]
= *SUM*(*Product*[*Korea.TV*,2003.*FH*]:Pr*oduct*[*Korea.TV*,2007.*LH*])
)

행과 열의 축으로 그룹화하는 표현 방법을 이용하여, 다양한 집계 영역들을 표현 할 수 있다. 예를 들어, 질의 "모든 해의 상반기에서 판매되는 모든 *TV*의 총 판매수량"은 SUM(*Productsh*[*Korea.TV*,*.*LH*]:*Productsh* [*Japan.TV*,*.*LH*]) 으로 표현될 수 있다. 여기서, 행의 축 좌표인 *Korea.TV*와 *Japan.TV*는 전체 *Country*에서 일부 *TV*를 선택하고 그룹화하게 되므로, 이를 간결히 표현하면 표1에 따라, SUM(*Productsh*[*.*,*.*LH*]) 으로 표현이 가능하다.



Country	Product	2003		2004		2005		2006		2007		All
		FH	LH	FH	LH	FH	LH	FH	LH	FH	LH	
Korea	TV	18	12	21	16	15	17	19	22	20	22	182
Korea	Radio	12	15	14	13	25	16	16	13	19	23	166
Korea	RAM	8	12	12	17	21	19	14	24	17	19	163
Japan	TV	15	21	17	20	13	22	17	10	18	21	174
Japan	Radio	10	18	24	22	14	12	20	17	21	16	174
Japan	RAM	14	16	23	19	17	23	18	19	22	14	185
All	All	77	94	111	107	105	109	104	105	117	115	1044

그림 5. 부분 및 전체 셀 영역에 대한 집계 계산.

4.3 전체 집계 연산

본 제안한 주소 표기법은 시트 전체를 셀 영역으로 지정하여 부분 또는 전체 집계를 계산할 수 있다. 이는 시트 내의 모든 집계 영역을 표시하고, 전체의 집계 값을 계산하는 방법과 행의 축 또는 열의 축에서 모든 좌표의 셀을 계산하는 방법으로 구분한다.

그림 **5**는 시트 내의 부분 및 전체 집계를 계산한 결과를 보이고 있다. 이러한 결과를 유도하기 위한 질의는 다음과 같다. 우선, 질의 Q1은 열의 축을 기준으로 각각의 집계 셀 영역을 명시하여 집계를 계산하는 의미이며, 질의 Q2는 행의 축을 기준으로 집계를 계산하는 의미이다. 그리고, 질의 Q3은 전체 시트에 대해 모든 집계 값을 계산하는 의미이다.

5. 결 론

OLAP 환경에서 다차원 데이터의 효율적인 분석과 집계된 요약 정보를 표현하기 위해 스프레드시트와 피벗 테이블이 널리 사용되고 있다. 본 연구에서는 관계형 데이터베이스에 저장된 테이블에서 스프레드시트 또는 피벗 테이블을 변환하고, 이들을 조작하기 위해 SQL 구문을 확장한 셀 구문을 제안하였다. 제안한 구문은 스프레드시트에서 행과 열의 축의 좌표를 이용하여 셀을 조작하기 위한 구문이다. 이로써, OLAP에서 사용되는 스프레드시트뿐만 아니라, RDBMS 내부적으로 제안한 구문이 적용된다면 관계형 테이블도 쉽게 조작할 수 있는 방법이라 사료된다. 향후 연구로는 제안한 구문을 실제 구현을 통해 실용적인 방법임을 보이는 것이며, 여러 시트와의 조인 연산 등을 고려하는 것이다.

참고 문헌

[1] Agrawal, R., Somani, A., and Xu, Y., "Storage and Querying of E-Commerce Data," In *Proc. of the 27th Int'l Conf. on Very Large Data Bases*(VLDB), Roma, Italy, pp. 149-158, 2001.

[2] Chaudhuri, S. and Dayal, U., "An Overview of Data Warehousing and OLAP Technology," *SIGMOD Record*, 26(1): 65-74, 1997.

[3] Chen, S. and Rundensteiner, E. A., "GPIVOT: Efficient Incremental Maintenance of Complex ROLAP Views," In *Proc. of the 21st Int'l Conf. on Data Engineering*(ICDE), pp. 552-563, 2005.

[4] Cunningham, C., Graefe, G., and Galindo-legaria, C. A., "PIVOT and UNPIVOT: Optimization and Execution Strategies in an RDBMS," In *Proc. of the 30th Int'l Conf. on Very Large Data Bases*(VLDB), Toronto, Canada, pp. 998-1009, 2004.

[5] Gray, J., et al., "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals," *Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 29-53, 1997.

[6] Spofford, G., et al., *MDX Solutions*, 2nd Edition, pp. 471-477, 2006.

[7] Witkowski, A., et al., "Business Modeling Using SQL Spreadsheets," " In *Proc. of the 29th Int'l Conf. on Very Large Data Bases*(VLDB), Berlin, Germany, pp. 1117-1120, 2003.

[8] Witkowski, A., et al., "Spreadsheets in RDBMS for OLAP," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, San Diego, CAA, pp. 52-63, 2003.