SQL내의 중대한 총체적 해악(總體的 害惡)

장황한(redundant) : 쓸모없는(de trop), 산만한(diffuse), 과잉의(excessive), 필수적이지 않은(inessential), 완곡한(periphrastic), 반복적인(repetitious), 원하지 않는(unwanted), 불필요한(unnecessary) 등은 Chambers Twentieth Century 유의어 사전에서 발췌된 redundant의 유사어들이다(동 사전은 concise, essential, necessary 등과 같은 redundant의 반의어들의 명단도 잘 정리되어 있다).

GROUP BY와 HAVING(이후부터는 GBH로 칭하기로 한다)으로 시작하는 절(clauses)들이 SQL내에서 반복적으로 사용된다는 사실을 알고 있는지에 대한 의문을 제기할 수 있다. 다시 말해, SQL내에서 표현되고, 상기의 절들 모두 또는 어느 한 쪽이 포함된 어떠한 종류의 상식적인 질문도 그러한 절들을 사용하지 않고도 표현될 수 있다는 것이다(필자가 여기서 '상식적'이란 말로 한정을 한 이유는 나중에 설명하겠다). 필자는 중복성에 관해 설명하고 이것이 내포하고 있는 의미에 관해 논의코자 한다.

GROUP BY절

언제나 그렇듯이 필자는 우리에게 익숙해진 공급자-부품 데이터베이스를 기본 예로 이용할 것이다.

```
S {S#,SNAME,STATUS,CITY}
  PRIMARY KEY {S#}

P {P#,PNAME,COLOR,WEIGHT,CITY}
  PRIMARY KEY {P#}

SP {S#,P#,QTY}
  PRIMARY KEY {S#,P#}
  FOREIGN KEY {S#} REFERENCES S
  FOREIGN KEY {P#} REFERENCES S
```

다음은 대부분의 사람들이 '자연스럽게' GROUP BY 절을 사용할 상기의 데이터베이스에 반(反) 하는 하나의 질문이다.

Q1 : 공급된 각 부품들마다 부품 번호, 공급된 최대, 최소 수량을 검색할 것.

이런 경우 상기의 질문에 '자연스럽게' 사용되는(GROUP BY) 공식을 사용했을 경우 아마 다음과 같은 형태일 것이다.

```
SELECT SP.P#,
       MAX (SP,QTY) AS MXQ,
       MIN (SP,QTY) AS MNQ
FROM   SP
GROUP BY SP.P# ;
```

〈그림 1〉에 제시된 예제 데이터가 주어졌을 경우, 상기한 질문의 결과는 〈그림 2〉에서 나타난 바와 같다.

다음은 정반대로 동일한 질문에 GROUP BY를 사용할 필요가 없는 또다른 공식이다.

```
SELECT DISTINCT SP.P#,
       (SELECT MAX (SPX,QTY)
        FROM SP AS SPX
        WHERE SPX.P =SP.P#) AS MXQ,
       (SELECT MIN (SPX,QTY)
        FROM SP AS SPX
       WHERE SPX.P# = SP.P#) AS MNQ
FROM SP ;
```

물론 이 공식이 이전의 공식보다 조금 더 길다는 것은 사실이지만, 논리적으로 그 둘은 동일하다. 그리고 그 사실은 다음

의 예제에서 쉽게 종합할 수 있다. R {A, B,...}라는 집합과 SUM, MAX, 또는 MIN 등과 같은 집합 함수 agg가 주어지고, 그 집합 함수를 R.B에 적용할 수 있다면, 그 표현식은 다음과 같다.

```
SELECT R.A,
    agg (R.B) AS C
FROM  R
GROUP BY R.A ;
```

상기의 표현식은 동일한 의미를 지닌 다음의 표현식으로 논리적으로 변형될 수 있다.

```
SELECT DISTINCT R.A,
   (SELECT agg (RX.B)
    FROM  R AS RX
    WHERE RX.A = R.A) AS C
FROM  R ;
```

다음의 내용에서 필자는 이러한 변형을 Type 1 변형으로 간결히 지칭할 것이다.

이제 우리는 원본(GROUP BY) 공식이 WHERE 절을 포함하고 있을때에 어떤 일이 발생하는지에 관해 생각해야 한다. Q1을 다음과 같이 연장해 보자.

Q2 : 공급된 각각의 부품마다 부품 번호, 공급된 최대, 최소 수량을 검색할 것; 그러나 공급자 S1을 통해서 공급된 물건들은 무시할 것.

첫째로 GROUP BY 공식은

```
SELECT SP.P#,
    MAX (SP.QTY) AS MXQ,
    MIN (SP.QTY) AS MNQ
FROM   SP
WHERE  SP.S# <> 'S1'
GROUP BY SP.P# ;
```

GROUP BY를 사용하지 않는 다음의 공식은 좀 더 복잡하다(아래의 공식이 GROUP BY를 사용하지 않는 유일한 공식은 아니다).

```
SELECT DISTINCT SP.P#,
    (SELECT MAX (SPX.QTY)
     FROM SP AS SPX
     WHERE SPX.P# = SP.P#
     AND  SPX.S# <> 'S1' ) AS MXQ,
    (SELECT MIN (SPX.QTY) AS
     MNQ
     FROM SP AS SPX
     WHERE SPX.P# = SP.P#
     AND SPX.S# <> 'S1' ) AS MNQ
FROM  SP
WHERE SP.S# <> 'S1' ;
```

보다시피 원본 GROUP BY 공식으로부터 주어진 WHERE 절은 SELECT 절 내의 괄호로 닫혀있는 두 표현식 안으로 복제되어야만 한다. 그 이유는 원본 공식 내의 WHERE 절은 해당 공식내에서 SELECT 절과 GROUP BY 절 양쪽 모두를 지배하기 때문이다. 그리고 해당 공식이 이러한 형태로 되어있는 이유는, SQL은 그러한 절들이 약간 비논리적인 순서로 쓰여질 것을 요구하기 때문이다.

일반적으로 SELECT-FROM-WHERE-GROUP BY 공식에서 사용되는 절들은 FROM-WHERE-GROUP BY-SELECT의 순서로 평가된다. 따라서 SQL이 사용되는 절들을 그와 같은 순서로 입력하기를 요구했다면 보다 이치에 맞았을 것이다. 그러나 실상 SQL은 그와 같은 것을 요구하지 않는다.

그 문제는 그렇다고 하고, 보다시피 Type 1 변형 룰은 WHERE 절들을 다루기 위해서 약간 연장되어야만 한다. 상기의 논의에 별도로 난해한 내용이 없으므로 그에 관한 자세한 설명을 생략하겠다.

이제 위에서 본 예제를 다시 한번 수정(修正)해 보자. 원본 질문이 다음과 같다고 가정해보자.

S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

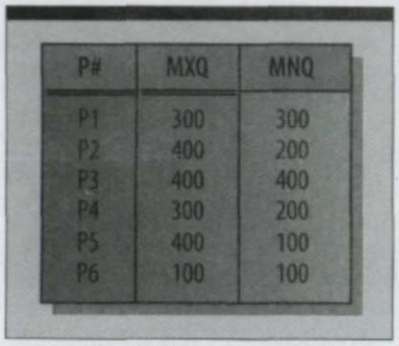
SP

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

P

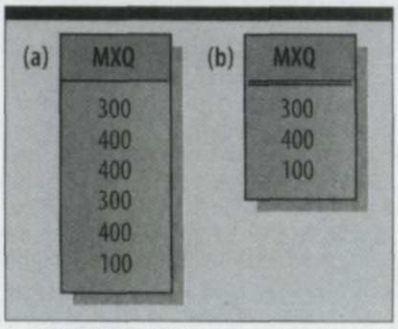
P#	PNAME	COLOR	WEIGHT	CITY
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

〈그림 1〉 공급자와 부품:데이터 가치 예



P#	MXQ	MNQ
P1	300	300
P2	400	200
P3	400	400
P4	300	200
P5	400	100
P6	100	100

〈그림 2〉 쿼리 Q1의 결과



(a)

MXQ
300
400
400
300
400
100

(b)

MXQ
300
400
100

〈그림 3〉 쿼리 Q3의 결과 (a)Group By 사용 (b)회피 Group By

Q3: 공급된 각각의 부품들마다 공급된 최대 수량을 검색할 것. 그러나 부품 번호는 검색하지 말 것.

```
GROUP BY 공식 a):

SELECT MAX (SP.QTY) AS MXQ
FROM   SP
GROUP BY SP.P# ;
```

Type 1 변형 룰을 적용해서 우리는 다음과 같은 공식 b)를 얻는다.

```
SELECT DISTINCT (SELECT   MAX
                         (SPX.QTY)
                 FROM SP AS SPX
                 WHERE SPX.P# = SP.P#)
                 AS MXQ
FROM  SP ;
```

상기의 두 가지 SQL 표현식들로 얻어진 결과는 〈그림 3〉에서 제시된다. 여러분이 보는 것과 같이 상기의 두 표현식은 동일한 결과를 산출하지 않기 때문에 그들은 정확히 동등하지는 않다. 다시 말해서 Type 1 변형은 위와 같은 특별한 경우에 유효하지 않다. 그러나 이렇게 동등성에 있어서 결함이 발생하는 진정한 이유는, GROUP BY 공식이 제공하는 결과는 관계형이 아니기 때문이다.

GROUP BY 공식에서 도출된 결과는 중복 열들을 포함하고 있기 때문에 관계형이 아니다. 덧붙여 말한다면 이러한 중복 열들은 필수적이다. 예를 들어 '300'이라고 입력된 두 열들은 서로 다른 의미를 지니고 있다. 전자는 어느 부품 하나의 최대 공급 물량이 300이라는 뜻이고, 후자는 또 다른 어느 부품의 최대 공급 물량이 300이라는 것을 뜻한다.

그리고 '필수 중복'은 연관 모델의 기본 개념을 상당히 이탈한다. 실상 필수 중복은, 모든 정보가 명백히 관계형 값의 형태로만 주어져야 한다는 정보 이론에 위배된다. 더구나 SQL이 '필수 중복'의 개념을 포함하고 있다는 사실은 SQL은 이제까지도, 그리고 지금도 완전히 연관적이지 않다는 점을 보여주는 매우 강력한 증거이다.

참고 : '필수 중복'은 SQL내에서 지금 논의되고 있는 형식 이외에도 여러 가지 형식으로 발생한다는 점을 간단히 지적하고자 한다. 가령, 다음과 같은 단순한 표현식에서조차도 일반적으로 '필수 중복'을 수반하는 결과를 제공한다.

```
SELECT CITY
FROM  S ;
```

Q3에 관하여 좀더 심도있게 논의해 보자. 이 질문이 진정 의미하는 것은 무엇인가? 해당 질문이 진정 요구하는 것은 SP 내의 최대 선적 수량들의 집합이다. 우리가 위에서 본 GROUP BY를 사용하지 않는 공식은 이러한 정보를 올바로 산출한다. 본 공식은 어느 특정 부품의 최대 수량이 얼마인지를 나타내지 않는다. 그러나 그러한 정보는 요구되지 않았다.

GROUP BY 공식 또한 상기의 공식과 동일한 정보를 제공한다(특정 부품의 최대 수량이 얼마인지를 나타내지 않은 상태에서 최대 선적 수량들에 관한 정보를 제공하는 것). 그런 점에서 볼 때, 여러분은 아마도 상기의 공식이 수용할 만하다는 사실에 동의할 것이다. 그러나 몇몇 사람은 GROUP BY 공식은 여섯부분의 상이한 정보를 제공하므로 이 공식을 GROUP BY를 사용하지 않는 공식보다 선호할 만하다고 계속 주장할 수도 있다. 그러나 GROUP BY 공식은 이러한 추가 정보를 비연관적 형태로 제공한다는 점에 주목하기 바란다(위에서 지적했듯이 이 공식은 이러한 추가 정보를 '필수 중복'의 형태로 제공한다).

필자의 생각에 우리가 실제 그러한 추가 정보를 얻기 원한다면 우리는 그러한 정보를 적합한 연관 형식으로 제공하는 새로운 질문을 이용해야 할 것이다(Q1이 그러한 요구를 만족시킬 것이다). 필자는 잠시 후 이 점에 관해 다시 논의할 것이다.

참고 : 아마도 우리는 두번째 공식(GROUP

BY를 사용하지 않은 공식)에서 DISTINCT 절을 탈락시킴으로써 두 공식간의 동등성을 지키고자 할 것이다. 하지만 그러한 계획은 어떤 문제도 해결하지 못한다. 그 이유는 첫째, 그 결과는 이제 셋 또는 여섯열이 아닌 열두열을 지닐 것이므로 수정된 공식은 원래의 두공식 중 어느 쪽과도 논리적으로 동등하지 않을 것이다. 둘째로, 그러한 열두열은 다시 '필수 중복'을 포함할 것이며, 그러므로 수정된 공식은 원본 GROUP BY 공식과 마찬가지로 진정한 연관형 공식일 수 없다.

마지막으로 Q3과 같은 질문은 합법적이지만 아주 상식적이지는 못하다는 점을 꼭 지목하고 싶다. 이렇게 말한 이유는, 이러한 질문은 우리가 무시되기를 바라지 않는 정보를 무시되도록 효과적으로 요구하기 때문이다(가령 위의 예제에서 무시된 정보는 각 부품의 최대 수량이 얼마인지 등과 같은 정보이다).

일반적인 SQL 형식에서 이러한 질문들은 전형적으로 일련의 GROUP BY 공식들을 유도하는데, 이러한 공식들에선 특정 칼럼이 GROUP BY 절내에서는 언급되고 상응하는 SELECT 절에는 언급되지 않는다. 그리고 이러한 공식들이 바로 우리의 Type 1 공식과 제대로 호환되지 않는 종류이다. 우리의 공식은 '상식적인' 질문들에 적용되었을 때 제 역할을 다한다.

HAVING 절(Clause)

이제 HAVING 절을 살펴보자. 다음은 대부분의 사람들이 '자연스럽게' HAVING 절을 사용하는 질문의 하나이다.

Q4 : 하나 이상의 공급자로부터 공급된 부품들 각각의 부품 번호를 검색할 것.

이 질문을 위한 GBH 공식은 아마 다음과 같을 것이다(GBH는 HAVING 절을 포함하고 있는 GROUP BY 공식을 의미한다는 것을 기억하기 바란다).

```
SELECT SP.P#
FROM   SP
GROUP BY SP.P#
HAVING COUNT (*) > 1 ;
```

〈그림 1〉의 예제 데이터가 주어졌을 때, 상기 질문의 결과는 〈그림 4〉에 제시된 바와 같다.

다음은 정 반대로 동일한 질문을 위한 non-GBH 공식이다(다시 말해, GROUP BY나 HAVING 절을 사용하지 않는 공식을 의미한다).

```
SELECT DISTINCT SP.P#
FROM   SP
WHERE (SELECT COUNT (*)
       FROM  SP AS SPX
       WHERE SPX.P# = SP.P#) > 1 ;
```

먼저 기술한 경우와 같이 이 공식 또한 GBH 공식보다 장황하다. 그러나 논리적으로 이 둘은 동등하다. 그리고 전의 예와 같이 이 예로부터 일반화하기가 용이하다. 이전 단락에서 논의됐던 동일한 개념을 이용했을 때 표현식은 다음과 같다.

```
SELECT R.A
FROM   R
```

P#	MXQ	MNQ
P1	300	300
P2	400	200
P3	400	400
P4	300	200
P5	400	100
P6	100	100

〈그림 4〉 쿼리 Q4의 결과

(a) MXQ	(b) MXQ
300	300
400	400
400	100
300	
400	
100	

〈그림 5〉 쿼리 Q7의 결과 (a)사용 Group By Having (b)불사용 Group By Having

```
GROUP BY R.A
HAVING agg (R.B) comp scalar ;
```

comp는 스칼라 비교 연산자이며 scalar는 스칼라 값을 가진 표현식이다. 상기의 표현식은 다음과 같은 동등한 표현식으로 변형될 수 있다.

```
SELECT DISTINCT R.A
FROM   R
WHERE (SELECT agg (R.B)
       FROM  R AS RX
       WHERE RX.A = R.A) comp
       scalar ;
```

앞으로 이러한 변형을 Type 2 변형이라고 지칭하겠다. 이제 다시 원본(HAVING) 공식이 WHERE 절을 포함하고 있을 때 어떤 현상이 발생하는지를 고려해

야만 한다. Q4를 다음과 같이 연장해 보자.

Q5 : (공급자 S1을 제외한) 하나 이상의 공급자들로부터 공급된 부품들 각각의 부품 번호를 검색할 것.

다음은 그에 상응하는 GBH 공식이다.

```
SELECT SP.P#
FROM   SP
WHERE SP.S# <> 'S1'
GROUP BY SP.P#
HAVING COUNT (*) > 1 ;
```

전과 마찬가지로 상기의 공식에 상응하는 non-GBH 공식은 좀 더 복잡하다.

```
SELECT DISTINCT SP.P#
FROM   SP
WHERE SP.S# <> 'S1'
AND   (SELECT COUNT (*)
FROM Sp AS SPX
WHERE SP.P# = SP.P#
AND   SPX.S# <> 'S1' ) > 1 ;
```

원본 GBH 공식에서 도출된 CLAUSE 절은, non-GBH 공식의 CLAUSE 절내에 괄호로 묶인 표현식 안으로 복제되어야만 한다. 그 이유는 GBH 공식내의 WHERE 절은 해당 공식 내의 GROUP BY 절과 HAVING 절 양쪽을 모두 지배하기 때문이다. 그러므로 Type 2 변형 룰은 Type 1 변형 룰과 마찬가지로 WHERE 절들을 다루기 위해서 연장되어야만 하는 것이다. 그에 관한 상세한 내용은 생략하겠다.

다음은 약간 더 복잡한 예이다.

Q6 : 하나 이상의 공급자들로부터 공급된 부품들 각각의 제품 번호를 검색하고, 각 부품의 총 공급 수량을 검색할 것.

다음은 상기의 질문을 위한 GBH 공식이다.

```
SELECT SP.P#,
       SUM (SP.QTY) AS TQY
FROM  SP
GROUP BY SP.P#
HAVING COUNT (*) > 1 ;
```

Type 1과 Type 2 변형 룰들을 적용함으로써 우리는 다음과 같은 결과를 얻는다.

```
SELECT DISTINCT SP.P#,
     (SELECT SUM (SPX.QTY)
      FROM   SP AS SPX
      WHERE SPX.P# = SP.P#) AS TQY
FROM   SP
WHERE (SELECT COUNT (*)
       FROM SP AS SPX
       WHERE SPX.P# = SP.P#) > 1 ;
```

그리고 마지막 예로

Q7 : 하나 이상의 공급자로부터 공급된 부품 각각의 총 공급 수량을 검색할 것, 그러나 부품 번호는 제외할 것.

이 경우 GBH 공식 a)는 아래와 같다.

```
SELECT SUM (SP.QTY) AS TQY
FROM   SP
GROUP  BY SP.P#
HAVING  COUNT (*) > 1 ;
```

변형된 형태 b)는 다음과 같다.

```
SELECT DISTINCT (SELECT SUM
(SPX.QTY)
                FROM SP AS SPX
                WHERE SPX.P# = SP.P#)
AS TQY
  FROM   SP
  WHERE (SELECT COUNT (*)
         FROM SP AS SPX
         WHERE SPX.P# = SP.P#) > 1 ;
```

상기의 두 표현식을 평가하여 얻은 결과는 〈그림 5〉에서 제시된다. 이전의 경우와 마찬가지로 이 두 표현식도 동일한 결과를 지니지는 않으며, 그러므로 엄밀히 말해 동등하다고 할 수 없다. 그러나 방금 우리가 살펴본 상황은 Q3에서 논의됐던 상황과 대단히 유사하다.

다시 말해서, Q7은 Q3이 지니고 있는 문제와 동일한 이유로 인해 난점이 있다. 따라서 이전 단락에서 있었던 Q3에 관한 논의는 여기서도 적용 가능한 것이며, 필자는 이 문제에 관한 추가적 언급이 필요 없을 줄 믿는다.