□ 기술해설 □

제품 기술 동향

데이타 웨어하우스를 위한 색인 기법

한국 인포믹스 지 용·전완기

1. 서 론

90년대 들어 데이타베이스 시장의 가장 큰 변화가 있다면 데이타 웨어하우스의 부상이다. 더불어 각 데이타베이스를 판매하는 회사들은 앞 다투어 경쟁사 보다 앞서가는 기술을 선점하기 위한 치열한 경쟁을 벌이고 있다. 이러한 가운데 데이타 웨어하우스는 국내 시장을 파고들면서 사용자로 하여금 그의 필요성을 인식하도록 하고 있다.

날로 늘어나는 데이타와 시장 성향 분석의 필요성을 직시하면서 데이타를 통한 정보화의 걸음은 한층 더 열기를 더해만 가고있다. 그래서 필자는 데이타 웨어하우스에서 사용되는 여러 가지 색인 기법에 대해서 설명하고자 한다. 아래에서 설명할 색인 기법들은 데이타 웨어하우스를 구축함에 있어 상당한 잇점을 제공하게 될 것임을 필자는 확신한다. 그리고 아래에 기술되는 내용은 Informix XPS에 제공되는 DSS 색인의 일부임을 밝혀둔다.

여러 가지 색인 기법을 설명하기 위해서 우선 아래의 테이블을 정의하고자 한다.

Bookings는 Partner_Supplier, Reservations, Membership, Booking_Agent, Destination, Promotions와 모두 조인되는 중앙 테이블이다. 데이타 웨어하우스의 데이타 모델링에서 bookings는 사실 테이블을 의미한다. 그리고 나머지 테이블을 차원 테이블이라 한다. 모든 데이타 웨어하우스 질의에 있어서 결과는 사실 테이블과 차원 테이블들 사이의 조인에 의해서 얻어진다. 그리고 여기에서는 언급하지 않지만 요약 정보를 담고있는 요약 테이블을

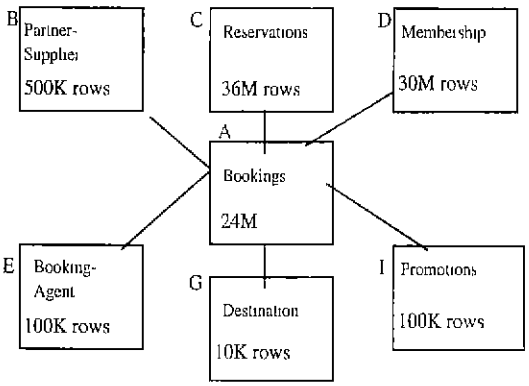


그림 1 예제 데이타베이스 테이블

통해서 원하는 통계 데이타를 얻게 된다.

2. DSS 색인

OLTP 시스템에서 의사결정을 위한 DSS (Decision Support System)는 그리 쉬운 상대는 아니다. 필요에 따라서 모든 데이타의 중복을 허용하여 원하는 자료를 원하는 형태대로 만든다 하더라도 전사적인 측면에서 이러한 데이타는 다양한 형태로 흩어져 있을 수가 있다. 무엇보다도 이러한 형태를 만들게 하는 원인은 성능의 향상을 꾀하기 위한 수단이기도 하거니와 거미줄처럼 흩어져 있는 데이타의 통합 자체가 힘들기 때문이다. 우선 OLTP 시스템에서 원하는 형태의 DSS를 지원하기 위해서는 OLTP 시스템의 성능 저하를 감수해야 한다. DSS라는 시스템은 OLTP 시스템보다도 시스템 자원을 많이 요구하기 때문이다. 그러나 DSS를 위해서 자체 시스템을 구비한다 하더라도 DBMS가 대량의 데이타를 처리할 수 있는 능력을 가

지고 있지 않다면 소용이 없다. 이러한 이유로 DSS 시스템을 위해 새롭게 등장한 기술이 병렬 처리 및 인데싱 기법이라 하겠다. 특히 DSS 시스템을 목적으로 생겨난 색인은 대량의 데이타를 빠른 시간에 처리할 수 있는 능력을 준다. 그럼 먼저 DSS 색인에 대한 설명 부터 시작하고자 한다.

DSS 색인은 단말 노드(leaf node)에서 비트맵(bitmap)이나 RID(Row IDentifier) 목록으로 저장할 수 있는 B-트리 구조이다. 단말 노드 값에 해당하는 테이블 행이 많을 수록 비트맵일 가능성이 높아진다. OLTP에서 사용하는 형태의 보통 색인처럼 DSS 색인도 조합하여 사용할 수 있는데, DSS 색인으로 Informix는 더 일반적인 색인 작성 기능을 적용하였다.

일반화된 색인의 개념을 이해하려면 색인을 새로운 방법으로 생각해 보는 것이 좋다. 색인이 담고 있는 정보는 개념상 질의의 결과로 볼 수 있도록 되어있다. 즉,

```
CREATE INDEX (index_name) ON
T(col1, col2, col3) ;
```

는 다음 질의의 결과로 볼 수 있다.

```
SELECT    T.col1, T.col2, T.col3, ROWID
FROM      T ORDER BY col1, col2, col3
```

T.col1,T.col2, T.col3가 각각 생성되면 한 색인 항목이 해당하는 ROWID 목록(또는 비트 맵)으로 된다. 새 DSS 색인를 사용하면 이 색인은 다음과 같이 만들어진다.

```
CREATE DSS INDEX (index_name)
on  T  (SELECT AS KEY column1, column2,
column3 FROM T) ;
```

테이블 T의 ROWID는 'on T' 절에 포함되고 ORDER BY는 'AS KEY' 목록에 포함되어 있다.

이제 이 개념을 확장하여 다른 질의들을 포함시키면 아주 유용한 색인 목록의 기반을 갖게 되는 것이다.

2.1 가상 칼럼 색인

가상 칼럼 색인은 몇 개의 칼럼을 통해서 해야하는 계산이나 부분 문자열 조작처럼 칼럼의 일부로 만들어지는 색인이다. 다시 Bookings 테이블을 사용하면 RENTA_CODE 필드의 마지막 두 글자가 임대 유형을 나타낸다. 따라서 다음과 같은 색인은 임대 유형의 제한을 평가하는 데 사용할 수 있다.

```
CREATE DSS INDEX    RENTA_TYPE_I
on BOOKINGS
(SELECT AS KEY BOOKINGS.RENTAL_
CODE[7,8] FROM BOOKINGS) ;
```

위의 색인은 RENTAL_CODE의 마지막 두 bytes를 이용하여 유용한 색인을 만드는 예이다. 이를 이용하여 특정 값을 조회할 수 있는 키를 만들 수 있다. 기존의 B 트리 색인에서는 구현하기 힘든 기능으로 DSS 색인만이 제공하는 유용한 기능이다.

가상 칼럼 색인의 다른 예제로는

```
CREATE DSS INDEX TOTAL_INSUR-
ANCE_I
on BOOKINGS (SELECT AS KEY
BOOKINGS.LIFE_INS_AMT +
BOOKINGS.TRIP_CNCL_INS_AMT +
BOOKINGS.HEALTH_INS_AMT + BOOK-
INGS.AUTO_INS_AMT
FROM BOOKINGS);
```

이러한 경우 아래의 조건

```
where BOOKINGS.LIFE_INS_AMT +
  BOOKINGS.TRIP_CNCL_INS_AMT +
  BOOKINGS.HEALTH_INS_AMT +
  BOOKINGS.AUTO_INS_AMT > 100
```

은 아주 쉽게 처리할 수 있다. 미리 계산된 식을 색인으로 가지고 있음으로써 사용자의 질의시 실행해야 하는 모든 연산작업을 색인을 통해서 구현할 수 있음은 상당한 성능의 효과를 가져다 줄 것은 당연하다. 이 색인은 보통 총액에 관심이 있을 때 가장 유용한 색인이다.

2.2 가상 칼럼

가상 칼럼이라 함은 말 그래로 실제로 물리적인 디스크의 영역을 보유하지는 않지만 사용

자에게는 마치 하나의 실제 칼럼처럼 보여주는 칼럼을 말한다. 가상 칼럼을 쉽게 참조하려면 데이타베이스에서 칼럼을 실제로 정의해야 한다.

예를 들어, 가상 칼럼 TOTAL_INSURANCE를 만들면,

```
ALTER TABLE BOOKINGS
ADD VIRTUAL COLUMN TOTAL_INSURANCE
        (BOOKINGS.LIFE_INS_AMT+
        BOOKINGS.TRIP_CNCL_INS_AMT+
        BOOKINGS.HEALTH_INS_AMT+
        BOOKINGS.AUTO_INS_AMT);
```

이 칼럼은 데이타베이스에서 공간을 차지하지는 않는다. 칼럼 정의는 질의 도구가 볼 수 있도록 저장된다. 이 가상 칼럼을 이용한 색인 작성은 다음과 같이 쓸 수 있다.

```
CREATE DSS INDEX TOTAL_INSURANCE_I
    on BOOKINGS
(SELECT AS KEY TOTAL_INSURANCE
FROM BOOKINGS);
```

사용자는 TOTAL_INSURANCE나 네 가지 금액의 합에 제한을 둘 수도 있다. 이러한 기능은 질의시 실행해야 하는 연산작업을 색인을 통해서 실행할 수 있는 획기적인 기법이라 할 수 있다.

2.3 색인에서의 CASE 문

색인 CASE는 칼럼을 유효 범위로 묶는 일종의 가상 칼럼 색인이다. 예를 들어, 일반적으로 각 종류의 보험이 구매되었는지 아닌지를 묻는 경우라면, 각 보험 종류에 따라 색인을 분류하면 처리 속도가 빨라진다. 네 가지 보험액을 각각 구매된 것(values>0)과 구매되지 않은 것(values=0)으로 구분한다면, 생명 보험에 대해 다음과 같이 가상 칼럼을 만들 수 있다.

```
ALTER TABLE BOOKINGS
ADD VIRTUAL COLUMN LIFE_PURCHASED
(CASE
   WHEN BOOKINGS.LIFE_INSURANCE_
   AMT>0 then 'YES'
   WHEN BOOKINGS.LIFE_INSURANCE_
   AMT=0 THEN 'NO'
   FROM BOOKINGS)
```

그리고 색인을 만들면 아래와 같다.

```
CREATE DSS INDEX LIFE_PURCHASED_I
on BOOKINGS
   (SELECT AS KEY LIFE_PURCHASED
   FROM BOOKINGS);
```

네 가지 보험액 칼럼 모두에 같은 작업을 마치면 사용자는 어떤 조합의 보험 종류가 구매되었는지(또는 구매되지 않았는지) 가상 칼럼을 참조하여 알 수 있다. 각 가상 칼럼은 두 가지 값(YES와 NO)만 취하므로, 색인은 비트맵으로 저장된다. 아주 빨리 OR 연산을 할 수 있고 다른 색인과 함께 사용할 수 있다.

2.4 이전 조인된 색인(Pre-Joined Index)와 선택적 색인(Selective Index)

DSS 색인은 select 문에서 조인 집합을 지원하므로 관리자는 질의를 미리 조인할 수 있다. 예를 들어, 사용자가 보통 단골 고객 카드 회원만 고려하면 FREQUENT_CUST_IND 값을 기초로 Bookings 테이블에 미리 조인된 색인을 만들 수 있다.

```
CREATE DSS INDEX CARD_HOLDERS_I
ON BOOKINGS
(SELECT AS KEY MEMBERSHIP.FREQUENT_CUST_IND
FROM BOOKINGS, MEMBERSHIP
WHERE BOOKINGS.MEMBERSHIP_ID=
MEMBERSHIP. MEMBERSHIP_ID)
```

이 색인은 MEMBERSHIP 테이블과 조인하지 않고 단골 고객 카드 회원이 포함된 Bookings에 행 목록을 만든다. 예제 질의에서처럼 회원 이름을 구할 때는 MEMBERSHIP과 조인이 이루어진다. 하지만 이 색인을 사용하면 Bookings에서 검색할 행 수가 줄어들므로 스캔(scan)이 더욱 빨라진다. Bookings에서 카

드가 없는 회원에 해당하는 행이 이미 삭제되었으므로 더 적은 수의 조인으로 작업할 수 있다.

공간을 절약하고 반응 시간을 줄이려면 색인을 선택적(즉, 카드 회원이 아닌 사람은 찾지 않도록)으로 만든다. 다음과 같은 WHERE 절을 추가하여 색인 명령문을 만들면 더욱더 효과적인 필터링(filtering)의 기능을 동시에 사용할 수 있게 된다.

```
WHERE MEMBERSHIP.FREQUENT_CUST
_IND='Y'
```

선택적 색인은 널(Null) 값을 색인으로 작성할 수 없다는 것을 표시하는 데도 사용된다. 예를 들어 행 대부분이 칼럼에 대하여 널(Null) 값을 가질 때나, 질의가 Null 칼럼과 전혀 관련이 없을 때 아래와 같은 조건절을 색인과 함께 사용하면 유용한 기능이 될 것이다.

```
WHERE MEMBERSHIP.FREQUENT_CUST
_IND IS NOT NULL
```

3. Skip Scan

이전에는 테이블의 작은 하위 집합을 액세스할 때만 색인을 사용하는 것이 유익했다. 색인이 행 목록을 나타낼 수 있다 해도, 그 목록이 길면 행을 각각 무작위로 액세스하는 데는 시간이 아주 많이 걸렸다. 행을 검색하는 알고리즘은 행을 디스크에 있던 순서대로 만들지 않아 같은 페이지를 여러 번 읽어야 하는 경우도 있었다.

Informix는 색인으로 표시된 행만 읽고, 행을 데이타베이스에 나타나는 순서대로 읽도록 하므로 테이블의 행을 효율적으로 스캔한다. 이 알고리즘은 skip scan이라고 불린다. 페이지를 두번 읽는 일은 없으며, 페이지를 무작위로 읽는 것이 아니라 순차적으로 읽으므로 I/O 자원 요구가 줄어든다. 순차적 스캔이므로 read-ahead가 가능하다. I/O가 감소될 뿐 아니라 색인 칼럼에서 필터링하지 않아도 되므로 CPU 요구도 감소한다.

4. 해시 세미조인(Hash Semi-Join)

해시 세미 조인은 다중 테이블 조인의 조인 속도를 향상시키기 위해 만들어진 Right Deep Tree 해시 조인의 최적화된 형태이다.

원래의 Informix Left Deep Tree 해시 조인(그림 2 숫자는 처리 순서를 표시)은 두 단계에 걸쳐 한 번에 두 테이블을 조인한다. 먼저, 첫 구축 단계에서 한 테이블의 선택된 열과 행들의 조를 만들고, 조인된 칼럼(조인 키)을 바탕으로 해시 테이블을 만든다. 이 해시 테이블은 임시 색인의 효과를 갖는다. 두번째 탐색 단계에서는 두번째 테이블의 필터링된 집합이 해시 테이블(다시 한 번 조인 키를 바탕으로)로 탐색하여 조인할 행을 찾는다. 두 개 이상의 테이블이 포함된 조인에서는 첫번째 조인 결과를 사용하여 모든 테이블이 한 번 조인될 때까지 다음 테이블에 대한 조인 단계를 수립한다.

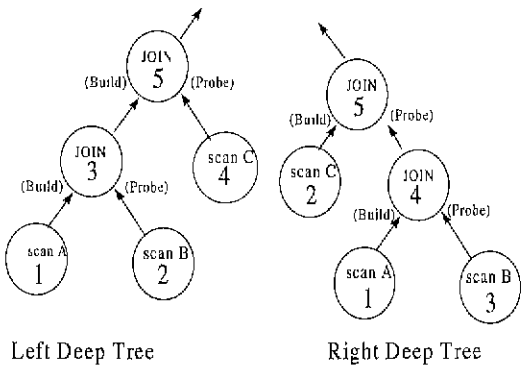


그림 2 Left/Right Deep Tree

여러 개의 작은 테이블과 조인할 큰 테이블이 있을 때는 Right Deep Tree 해시 조인이 사용하기 편리하다. 이 경우 구축 단계는 각 작은 테이블에서 필터된 조에서 이루어진다. 그런 다음 탐색 단계가 큰 테이블을 사용하여 시작된다. 큰 테이블의 필터링된 각 조는 작은 구축 테이블을 하나하나 탐색하는 데 사용된다. 이 방법에서 데이타 중간 집합이 없고 모든 테이블 조인이 동시에 일어난다(위 숫자는 처리 순서를 나타내지만 여러 세그먼트는 동시에 처리될 수 있다).

큰 테이블 하나가 여러 개의 작은 테이블에 조인되는 특수한 조인(아래 그림 참고)이 있을 경우, 이런 형식의 조인에서 가장 주목할 만한 것은 스타 스키마의 스타 조인이지만, 그 경우에만 제한되는 것은 아니다.

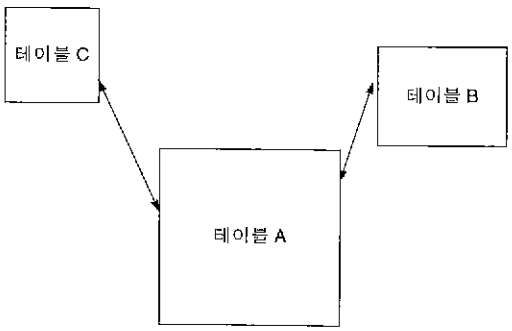


그림 3 여러개의 작은 테이블과 조인된 큰 테이블

새로운 색인 작성 방법을 Right Deep Tree와 함께 사용하면 이 해시 세미 조인을 다루는데 특히 효율적인 알고리즘을을 제공할 수 있다. 다른 작은 테이블들과 조인된 큰 테이블의 경우, 작은 테이블은 보통 메모리에 해시될 수 있으므로 Right Deep Tree가 가장 적합한 조인이다. 모든 테이블이 조인될 때쯤이면 마지막 결과는 상당히 작아지지만, 표준 Right Deep Tree로는 테이블을 탐색할 때 아주 많은 수의 조로 시작하게 된다. Informix는 Hash Semi-Join을 사용하여 추가 단계에서 조인이 시작되기 전에 될 수 있으면 조의 집합을 줄인다.

기본 Right Deep Tree에서처럼 해시 세미 조인의 구축 단계는 작은 테이블 각각의 열/행 선택에서 수행된다. 그런 다음 조인 키는 큰 테이블을 skip scan하게 된다. 다시 말해, 작은 테이블 각각의 조인 키 목록(큰 테이블의 색인)은 조합하여 큰 테이블에서 skip scan하는데 사용된다. 그런 식으로 조인되지 않는 조는 탐색 단계가 시작되기 전에 큰 테이블에서 삭제된다. skip scan의 결과 조는 이미 해시된 각 테이블을 계속하여 탐색하는 데 사용한다.

해시 세미 조인을 설명하기 위해서 테이블 A, B, C(그림 4)를 보자. A는 A.col1=B.col1인 칼럼에서 B와 조인되고, A.col2=C.col2인

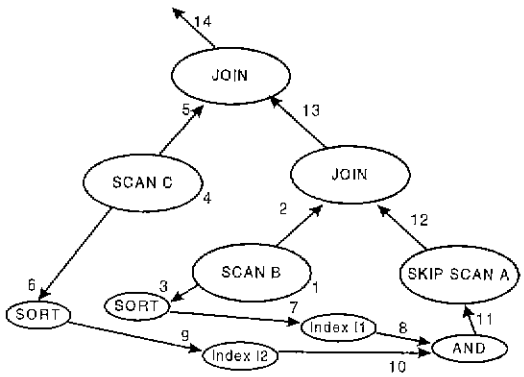


그림 4 조인된 SQL 문의 프로세스 처리 순서

칼럼에서는 C와 조인된다. 이 칼럼에 테이블 A에 대한(col1의 I1와 col2의 I2) 색인을 만든다.

위 그림은 질의 트리를 단순화하여 나타낸 것으로 right-deep hash join을 포함한다. 그림의 숫자는 다양한 단계의 처리 순서를 나타내지만, 동시에 일어나는 단계가 많다.

1단계에서 테이블 B가 액세스되고, 행 선택과 열 선택 후에 조는 교환 연산자(나타나지 않음)에 전달되어 다시 그 조를 해시 조인 구축 단계를 실행하는 조인 연산(2단계)에 전달한다. 교환 연산자는 또한 조인하는 칼럼(조인 키)을 취하여 정렬 연산자에 전달(3단계)하여 키를 정렬하고 중복을 삭제한다. 같은 과정이 테이블 C(4단계)에서 시작되고 교환이 조를 각 구축 단계(5단계)로, 정렬 연산자(6단계)로 전달한다. 테이블 B와 C가 모두 액세스되면 각 정렬 연산자는 중복을 삭제한 정렬된 키 목록을 만든다. 7단계와 8단계에서는 A(I1의 모든 해당 비트 맵에서 AND 연산 수행)의 I1 색인에서 조회하여, B에서 발견한 모든 키에 일치하는 A의 행의 비트 맵을 만든다.

비슷한 방식으로 9단계와 10단계에서 테이블 C의 I2에서 조회하는 동안 비트 맵이 형성된다. 두 비트 맵은 논리적으로 AND 되어 B.C와 조인되는 A의 모든 행의 비트 맵을 만든다. 이 때 A의 다른 색인이 조합되어 조인되는 행 수를 줄일 수 있다. 이 마지막 비트 맵이 형성되면 skip-scan으로 A의 해당 행을 읽는데 사용된다. 열 선택과 행 선택은 A에서 일어난다.

이제 되돌아가서 A와 조인되는 B와 C에서

조를 선택해야 한다. 각 테이블 A의 조는 먼저 B의 구축 테이블을 탐색하는 B(12단계)의 조인으로, 다음에는 C(13단계)와 조인된 탐색 단계로 보내진다. 이 두 조인은 병렬로 동시에 일어날 수 있다. 그 결과 조인된 조는 14단계에서 다음 단계(Group-By 또는 Order-By)로 전달된다.

5. 결 론

앞에서 DSS 시스템을 위해 특별히 고안된 여러가지 색인과 기능에 대해서 간단히 살펴보았다. OLTP와는 그 성격을 달리하는 OLAP 시스템은 데이타 양의 방대함과 더불어 다양화되어가는 사용자들의 요구 사항에 대처하기 위한 대책으로 우리에게 다가오기 시작하면서 그의 중요성을 인식하기 시작했다. 과거에 IMS와 같은 시스템을 통해서 다각적으로 DSS에 대한 해결책을 찾으려 했지만, 사용자의 요구 사항에 대한 대응을 하기에는 아직 미약했던 여건을 인지한 여러 DBMS vendor들은 각자 선두적인 기술의 보유를 위해 많은 투자를 해야만 했고, 특히 INFORMIX는 병렬 처리 기술에 있어 타 회사 보다는 앞선 기술로 고객에게 다가서고 있음은 이러한 끝임없는 투자와 기술 개발의 결과라 하겠고, 더욱이 DSS 시스템을 위한 색인 기법의 적용은 대량의 데이타를 처리해야 하는 시스템에 있어 상당한 성능의 향상을 얻을 수 있도록 하는 중요한 기능으로 부상하고 있다. 이글을 마무리 하면서 여러분들이 앞으로 한 번쯤은 사용하게될 OLTP및 OLAP 시스템을 위한 데이타베이스의 선택에 있어 무엇이 진정으로 필요한 기능인가를 우선적으로 따져보기를 바란다. 그리고 그의 미래와 기술 전개 방향을 눈여겨 살펴볼 수 있는 여유를 가지기를 바란다.

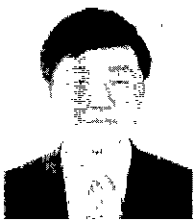
참고문헌

[1] Ralph Kimball, "The Data Warehouse Toolkit," John Wiley and Sons. 1996.

[2] Patrick O'Neil and Goetz Graefe, Multi-Table Joins Through Bitmapped Join Indices, SIGMOD Record, September, 1995. pp. 8-11.

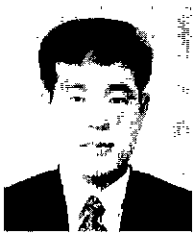
[3] INFORMIX White Paper, "INFORMIX-OnLine Extended Parallel Server"("for Loosely Coupled Cluster and Massively Parallel Processing Architectures"), July 1995.

[4] Patrick O'Neil, Patrick, And Goetz Graefe, "Multi-Table Joins Through Bitmapped Join Indices," SIGMOD Record, September, 1995. pp. 8-11.



지 용

1982 한양대학교 전자공학과 전자공학사
1984 한국과학기술원 전산학과 석사
1984~88 대우전자 PC개발부 근무
1988~90 한국마이크로소프트 개발부 근무
1991~95 대우기술 연구소 근무
1995~현재 한국인포믹스 고객지원부 부장



전 완 기

1991 창원대학교 전산학과 졸업
1991~1995 대우 기술 근무 (Informix Database 기술지원)
1995 한국 인포믹스 전문 기술지원부
1995 국내 최초 SDS와 데이터 웨어하우스 Pilot 실시
1996~현재 데이타 웨어하우스 컨설팅 담당
1996 삼성 화재 데이타 웨이하우스 벤치마크 실시
1997 현대 자동차 데이타 웨어하우스 프로젝트 실시
현재 한국 인포믹스 데이타 웨어하우스 팀 근무