관계형 데이터베이스에서 저장용량에 제약이 없는 경우 비정규화를 고려한 데이터베이스 설계
- A Database Design without Storage Constraint Considering Denormalization in Relational Database -

장 영 관*
Jang, Young Kwan
강 맹 규*
Kang, Maing Kyu

Abstract

Databases are critical to business information systems and RDBMS is most widely used for the database system. Normalization was designed to control various anomalies(insert, update, and delete anomalies). However normalized database design does not account for the tradeoffs necessary for the performance reason.

In this research, we model a database design problem without storage constraint. Given a normalized database design, in this model, we do the denormalization of duplicating columns in order to reduce frequent join processes. In this paper, we consider insert, update, delete, and storage cost, and the anomalies are treated by additional disk I/O cost necessary for each insert, update transaction.

We propose a branch and bound method, and show considerable cost reduction.

1. 서론

1.1 연구의 배경 및 목적

최근에 정보처리 시스템은 급격하게 증가하고 있고, 데이터는 여러 부문에서 중요한 전략적 자원이 되어가고 있다. 이전에는 데이터를 주로 화일로 관리하여 왔으나 정보처리 요구가 증가함에 따라 DBMS 사용이 급격히 증가하고 있다. DBMS를 사용함으로써 발생한 큰 변화 중의 하나는 정보가 정보관리자의 독점에서 벗어나 사용자들이 매우 쉽게 사용할 수 있게 되었다는 것이다. DBMS 중에서 특히 관계형 DBMS가 현재 가장 많이 쓰이고 있다.

데이터베이스 설계는 일반적으로 (1) 요구의 수집 및 분석, (2) 논리적 설계, (3) 물리적 설계, (4) 구현 등의 절차로 이루어진다. 그 중에서 논리적 설계 단계에서는 일반적으로 E-R 모델을 사용하며, 정규화(normalizaion)와 데이터 무결성 제약을 고려하여 모델링하고, 물리적 설계 단계에서는 논리적 설계에서 설계된 관계를 그대로 테이블로 사상(mapping)하여 구현하는

* 한양대학교 산업공학과

것이 이상적이다. 그러나 이렇게 설계한 결과는 좋은 수행도(performance)를 기대하기가 어렵다. 좋은 수행도를 얻기 위해서는 많은 경우 비정규화(denormalization)가 필요하지만 아직까지 수리적으로 모델링하여 최적화하는 연구가 크게 미진하다.

본 연구에서는 데이터베이스의 수행도를 향상시키기 위해 정규화된 데이터베이스 설계를 비정규화된 데이터베이스 설계로 조율(tuning)하는 방법을 수리적으로 모델링하고 분지한계법을 사용하여 최적해를 구한다.

1.2 기존 연구

물리적 데이터베이스 설계에 관한 연구는 주로 수직분할(vertical partitioning), 분산 데이터베이스 환경에서의 수평분할(horizontal partitioning)의 두 가지 분야로 연구되어 왔다.

Roger[13]는 논리적 데이터베이스 설계를 그대로 물리적 데이터베이스 설계로 전환하여 적절한 수행도를 얻기는 어려우므로 경우에 따라서 비정규화는 필요하다고 강조하였다. Rajiv[12]는 물리적 데이터베이스 설계와 논리적 데이터베이스 설계를 수리모형화하여 열거법으로 해를 구하였다. 물리적 설계에서는 정렬과 인덱스(indexing)를 고려하고, 논리적 설계에서는 universal 릴레이션(relation)을 분해(decomposition)하는 것을 고려하였다.

Westland[16]는 데이터베이스 정규화에 관련된 저장비용, 갱신이변비용(anomaly cost), 조회비용을 확률모형으로 분석하였다. Lee[11]는 무조건 고차의 정규화 형태로 설계하는 것 대신에 비용과 수익을 고려해서 의사결정나무 방법으로 정규화형을 결정하는 방법을 제시하였다. Lee는 비용요소로서 저장비용(storage cost), 갱신이변비용, 그리고 결합비용(join cost)을 고려하였다.

2. 물리적 데이터베이스 설계

물리적 데이터베이스 설계는 좋은 수행도를 얻을 수 있도록 저장구조와 액세스(access)경로를 설정하는 것이다. 수행도는 보통 트랜잭션(transaction)을 처리하기 위해 필요한 액세스 횟수에 의해 결정된다. 예를 들어, 높은 차수로 정규화되어 있는 테이블은 보통의 경우에 많은 로우(row)를 액세스해야 한다[9].

이러한 문제를 해결하는 방법에는 하드웨어 시스템 조율, DBMS 조율, 데이터베이스 설계 조율, 응용프로그램 조율 등이 있다[4]. 이 중에서 데이터베이스 설계 조율과 응용프로그램 조율이 가장 효과가 크다. 응용프로그램 조율은 관계형 데이터베이스의 질의언어(SQL)를 조율하는 방법으로서 데이터베이스의 물리적 설계 결과에 크게 영향을 받는다. 그러므로 좋은 수행도를 갖는 데이터베이스를 설계하기 위해서는 물리적 설계가 가장 중요한 것이라 할 수 있다[7].

데이터베이스의 물리적 설계 방법은 인덱스, clustering, hashing, 비정규화, 단편화(fragmentaion) : 수직 단편화, 수평 단편화 등이 있다. 이 중에서 인덱스, clustering, hashing의 방법은 데이터베이스의 테이블 설계를 변경하지 않고 할 수 있는 방법이고, 비정규화, 단편화는 테이블 설계를 변경하여 수행도를 향상시키는 방법이다.

비정규화의 일반적인 절차는 가장 중요하고 빈도가 높은 트랜잭션을 대상으로 하여 액세스 경로를 분석한다. 이러한 액세스 경로를 향상시키기 위해 결합 데이블을 정의하고 삽입, 갱신, 삭제, 조회, 저장 비용을 평가하여 데이터 무결성 유지 여부를 검증한다[15].

논리적 데이터베이스 설계의 결과인 정규화된 테이블 설계를 비정규화하여 설계하는 방법은 다음과 같다[13].

- 두 개의 릴레이션이 1:1의 관계에 있을 때: 두 개의 테이블을 1개의 테이블로 만든다.
- 두 개의 릴레이션이 M:N의 관계에 있을 때: 세 개의 테이블을 두 개의 테이블로 만든다.
- 두 개의 릴레이션이 1:N의 관계에 있을 때: E-R 모델에서 가장 일반적인 경우로서 참조되는 1의 관계의 컬럼을 N의 관계의 테이블에 복사하여 결합(join)을 줄일 수 있다[8].
- 매우 자세한 데이터를 가지는 릴레이션: 부모(parent) 테이블에 자식(child)을 포함시킨다.
- 파생된(derived) 컬럼 : 한 개의 컬럼을 추가하여 반복적인 계산을 피한다.
- 보고서용으로 컬럼을 발췌하여 새로운 테이블을 만든다.

3. 연구 내용

2장에서의 비정규화 방법은 특수한 상황에서만 적용할 수 있고 일반적으로 정형화하기는 매우 힘들다. 본 연구에서는 더 일반적이고 범용성이 있는 모델을 만들어서 초기의 물리적 설계 단계뿐만 아니라 데이터베이스 운용 중에 데이터베이스를 재설계할 경우에도 사용할 수 있도록 한다.

본 연구에서는 실제적으로 시스템을 구축할 때 널리 쓰이는 제 3 정규형으로 정규화된 테이블 설계가 주어져 있을 때 외부키(foreign key)관계에 있는 두 개의 테이블에서 외부키 대상이 되는 참조테이블(referenced table)로부터 컬럼을 복사함으로써 빈번한 결합을 방지하여 조회 트랜젝션의 수행도를 향상시키고 전체적인 수행도를 향상시킨다. 이 결과는 다음의 절충 관계가 있다.

- 조회 비용 감소
- 보조기억장치(디스크)의 저장비용 증가
- 삽입, 갱신, 삭제 비용 증가

데이터베이스 설계와 구현의 과정에서 디스크 저장용량에 대한 투자 결정이 이루어지고 단기적으로는 디스크 저장용량은 제한되어 있으며, 디스크 저장용량을 늘리는 것은 장기적인 의사결정 사항이다[16].

본 연구에서는 디스크 저장용량이 제한되어 있지 않은 경우에 삽입 비용, 갱신 비용, 삭제 비용, 조회 비용, 저장 비용을 수리 모형에서 고려한다. 본 연구의 비용 요소는 디스크의 액세스 횟수에 따른 비용과 디스크 저장에 따른 비용이다.

본 수리 모형은 데이터베이스에서 매우 중요한 개념이고 필수적으로 지켜져야 하는 데이터의 일관성(consistancy)을 유지하기 위한 비용을 주어진 상수로 계산하는 것이 아니라 물리적인 액세스 방법에 따른 비용으로 수리 모형에 포함시켜 수식화한다.

본 연구에서의 액세스 방법은 순차 액세스(segment scan access)와 인덱스 액세스(indexed scan access)의 두 가지를 고려한다. 그 이외의 액세스 방법인 clustering, hashing 은 본 연구를 확장하여 쉽게 적용할 수 있다. 순차 액세스의 액세스 횟수는 $\frac{T_i \cdot R_i}{W}$, 인덱스 액세스의 액세스 횟수는 $(1+H_i)N_i$ 로 추정할 수 있다[6].

4. 수리 모형

4.1 가정

본 연구에서의 가정은 다음과 같다.

- 정규화된 설계가 주어져 있다.
- 기본키에 대한 인덱스가 존재한다.
- 인덱스 갱신 비용을 고려하지 않는다.
- CPU 시간, 주기억 장치의 한계는 고려하지 않는다.
- 결합, 정렬 비용은 고려하지 않는다.
- block에는 데이터만 저장되어 있고 정의된 길이보다 적은 컬럼도 정의된 길이만큼 고정길이로 저장된다.
- 모든 테이블의 로우 크기는 1 block 보다 작다.
- 모든 결합조회 트랜잭션은 두 개의 테이블만을 결합한다.
- 모든 갱신, 삭제 트랜잭션은 기본키를 액세스 경로로 하여 이루어지고 외부키에 의해 참조되는 테이블에 대한 삭제 트랜잭션은 없다.

4.2 기호 정의

본 연구에서 사용된 기호는 다음과 같다.

i, j, m : 테이블

k : 컬럼

R_i : 테이블 i의 로우 개수

L_i : 정규화된 테이블 i의 로우 크기

L_{ik}: 테이블 i의컬럼 k의 크기

A : 디스크 액세스를 1회 하는 데 소요되는 비용

S : 1 byte 저장 비용

B : block 크기

W : blocking 계수×block 크기

H_i : 테이블 i에서 기본키 인덱스의 높이(height)

H_{ti} : 트랜잭션 t에서 사용하는 테이블 i의 인덱스 높이

V_{ij} : 테이블 i의 한 개의 로우에 대응되는 테이블 j의 평균 로우 개수

F_t : tranaction t의 발생 빈도

z_{tji} 1 : 조회 트랜잭션 t에서 액세스하는 테이블 i의 모든 컬럼이 결합되는 테이블 j에 복사되어 있을 때

0 : 그렇지 않은 경우

y_{ij} , g_{ij} 1 : 테이블 i에 테이블 j의 컬럼이 적어도 한 개 복사되어 있을 때

0 : 그렇지 않은 경우

x_{ijk} 1 : 테이블 i에 테이블 j의 컬럼 k가 복사되어 있을 때

0 : 그렇지 않은 경우

K_i : 테이블 i의 외부키 컬럼집합
K_{ti} : 트랜잭션 t에서 액세스하는 테이블 i의 컬럼 집합
U_{ti} 1 : 갱신 트랜잭션 t에서 j의 기본키를 참조하는 i의 외부키를 갱신할 때
0 : 그렇지 않은 경우
I_t : 삽입 트랜잭션 t에서 갱신하는 테이블 집합
U_t : 갱신 트랜잭션 t에서 갱신하는 테이블 집합
D_t : 삭제 트랜잭션 t에서 갱신하는 테이블 집합
P_t : 조회 트랜잭션 t에서 인덱스 액세스하는 테이블 집합
Q_t : 조회 트랜잭션 t에서 순차 액세스하는 테이블 집합
N_{ti} : 조회 트랜잭션 t에서 액세스하는 테이블 i 의 로우 개수

4.3 수리 모형의 정식화

본 수리 모형은 삽입 비용(Ci), 갱신 비용(Cu), 삭제 비용(Cd), 조회 비용(Cq), 저장 비용(Cs)의 합을 최소화하는 문제로서 다음과 같이 표현된다.

$$Min\ (Ci + Cu + Cd + Cq + Cs)$$

s.t.

$$\sum_k x_{ijk} - M \cdot y_{ij} \le 0$$

$$\sum_{k \in K_a} x_{jik} - M \cdot g_{ji} \le 0$$

$$z_{tji} = \prod_{i \in K_a} x_{jik}$$

본 수리 모형에서의 x_{ijk} 는 결정변수로서 테이블 i에 테이블 j의 컬럼 k를 복사한다는 변수이다. 만약 $x_{ijk} = 1$ 이라고 결정되면 테이블 i에 새로운 컬럼이 한 개 추가된다. x_{ijk} 가 결정변수가 될 수 있는 조건은 다음의 세 가지가 동시에 만족하여야 한다.

- 테이블 i 에는 테이블 j의 기본키를 참조하는 외부키 컬럼이 있어야 한다.
- 테이블 i 와 테이블 j를 결합하여 조회하는 트랜잭션이 적어도 한 개 있어야 한다.
- 그 트랜잭션에서 테이블 j의 컬럼 k에 대한 조회 요구가 있어야 한다.

그림 1은 비정규화된 테이블 실계의 예로서 Order 테이블의 Customer_id는 Customer 테이블의 Customer_id를 참조하는 외부키이고, Order 테이블의 Customer_name은 Customer 테이블의 Customer_name을 복사한 것이다.

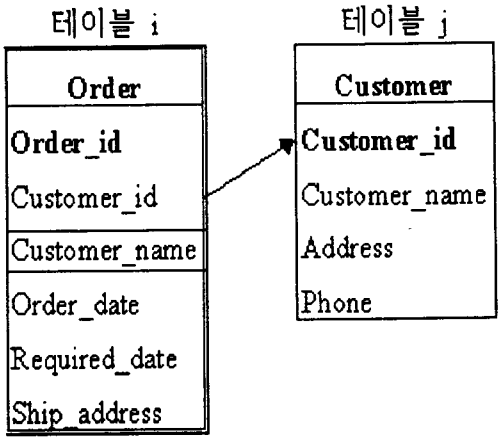

그림 1. 비정규화된 테이블 설계의 예

4.3.1 삽입 비용(Ci)

삽입 비용은 테이블 i에 새로운 로우를 삽입할 때 발생하는 액세스 횟수와 복사된 컬럼의 값을 유지시키기 위한 테이블 j에 대한 액세스 횟수에 따른 비용으로서 다음과 같은 식으로 표현된다. 여기서 테이블 j는 기본키에 의해 1개의 로우를 액세스한다.

$$\sum_t F_t \sum_{i \in I_t} A\,[1+\sum_{j \neq i} y_{ij}(1+H_j)]$$

여기서 y_{ij}는 추가된 결정변수로서 테이블 i에 테이블 j의 컬럼이 하나라도 복사하면 1이 되는 변수이다. 그림 1에서 Order에 새로운 로우가 삽입되는 경우에 일관성을 유지하기 위해 Customer_name을 Customer 테이블에서 조회하는 비용이 추가된다.

4.3.2 갱신 비용(Cu)

갱신 비용은 복사한 테이블의 외부키가 인덱스되어 있을 경우에 다음과 같이 표현된다.

$$\sum_t F_t \sum_{i \in U_t} A[\,(2+H_i)+\sum_{j \neq i} y_{ij} \cdot U_{tij}\,(1+H_j)+\sum_{j \neq i} V_{ij} \cdot g_{ji}\,(2+H_j)]$$

여기서 g_{ij}는 추가된 변수로서 만약 x_{ijk}가 하나라도 1이면 g_{ij}는 1이 된다. 그림 1에서 Order의 customer_id가 갱신되는 경우에 데이터의 일관성을 유지하기 위해 Customer 테이블에서 1개의 로우를 조회해야 한다. 또한 Customer 테이블의 Customer_name이 갱신되는 경우에 Order에 복사한 Customer_name을 갱신해야 한다. 복사한 테이블의 외부키가 인덱스되어 있지 않을 경우는 다음과 같이 표현된다.

$$\sum_t F_t \sum_{i \in U_t} A[\,(2+H_i)+\sum_{j \neq i} y_{ij} \cdot U_{tij}\,(1+H_j)+\sum_{j \neq i} g_{ji}\,(\frac{T_j \cdot R_i}{W}+V_{ij})]$$

여기서, $T_j = L_j + \sum_{m \neq j} \sum_k L_{mk} x_{jmk}$ 로서 테이블 j의 로우 크기를 나타낸다.

4.3.3 삭제 비용(Cd)

삭제 비용은 다음과 같이 표현되며, 본 연구에서는 컬럼을 복사해 두는 비정규화 방법이므로 삭제 이변(anomaly)이 발생하지 않는다.

$$\sum_t F_t \sum_{i \in D_t} A(2+H_i)$$

4.3.4 조회 비용(Cq)

조회 비용은 다음과 같이 순차 액세스와 인덱스 액세스의 두 가지로 나누어 다음과 같이 표현된다.

$$\sum_t F_t \cdot A[\sum_{i \in Q_t}(1-z_{tji}) \frac{T_i \cdot R_i}{W} + \sum_{i \in P_t}(1-z_{tji}) \cdot (1+H_{ti})N_{ti}]$$

여기서 z_{tji}는 추가된 결정 변수로서 결합 조회에서 필요한 i 테이블의 컬럼이 모두 j 테이블에 복사되어 있으면 1이 되어 i 테이블을 액세스하지 않아도 결합조회의 결과를 얻을 수 있다.

4.3.5 저장 비용(Cs)

저장 비용은 다음과 같이 표현되고, 주어진 정규화된 테이블의 로우 크기에 복사된 칼럼의 크기를 합하고 로우 개수를 곱하여 계산한다.

$$\sum_i \sum_{j \neq i} \sum_k S(L_i + L_{ik} \cdot x_{ijk})R_i$$

4.3.6 제약 조건

제약 조건은 다음과 같이 표현된다.

$$\sum_k x_{ijk} - M \cdot y_{ij} \leq 0$$

$$\sum_{k \in K_a} x_{jik} - M \cdot g_{ji} \leq 0$$

여기서 M은 상수로서 크기가 큰 수(big M)을 나타내며, 추가된 결정변수 y_{ij} 및 g_{ij}에 대한 제약조건이다. 만약, x_{ijk}가 하나라도 1이면 y_{ij} 및 g_{ij}는 1이 되도록 제약한다.

$z_{tji} = 0$ $i, j \in Q_t \cup P_t$, t는 단일 테이블 조회 트랜잭션

$z_{tji} = \prod_{i \in K_a} x_{jik}$ $i, j \in Q_t \cup P_t$, j는 i를 참조하는 외부키 테이블, t는 결합조회 트랜잭션

이 제약조건은 결합조회 트랜잭션을 처리하기 위한 제약으로서 만약, 결합조회에서 필요한 테이블 i의 모든 속성이 j에 복사되어 있으면 z_{tji}는 1로 제약되어 그 조회를 처리하기 위한 테이블 i의 액세스는 0이 될 수 있도록 한다.

5. 분지한계법을 이용한 알고리듬

5.1 분지한계법의 기본 개념

분지한계법은 기본적으로 열거법(enumeration)의 일종으로서 그 중에서 부분열거법에 속한다[1]. 분지한계법은 각 단계에서 최적해가 포함되어 있을 가능성이 가장 커보이는 부분집합을 선정하여, 이 부분집합에 계산량을 줄일 수 있는 어떤 전략을 사용하여 두개 이상의 새로운 부분집합으로 분할한다. 이를 분지(branch)라 한다.

각 분지된 문제에서 그 부분집합에서의 해의 한계를 계산한다. 만약 해의 한계를 구하는 중에 최적해를 구했으면 알고리듬을 끝낸다. 그렇지 않으면 후보문제들의 한계를 사용하여 새로 분지할 부분집합을 선정하고 위의 과정을 반복한다.

5.2 본 연구의 분지한계법

본 연구에서의 분지는 복사할 대상이 되는 컬럼을 복사한다($x_{ijk} = 1$), 또는 복사하지 않는다($x_{ijk} = 0$)로 하면 된다. 본 연구에서 개발한 분지한계법의 절차는 다음과 같다.

단계 0 : [초기화]

후보노드를 비어 있음으로 표지하고 단계 2로 간다.

단계 1 : [후보노드를 결정한다]

아직 분지하지 않은 후보노드 중에서 최저한계비용이 가장 적은 노드를 찾는다. 만약 그 노드가 현재까지의 최적해 노드의 최저한계비용보다 크거나 같으면 단계 1을 반복한다. 만약 분지하지 않은 후보노드가 없으면 끝낸다.

단계 2: [분지할 노드를 결정한다]

단계 1에서 후보노드를 x_{ijk} 라고 하면 x_{ijk}를 사용하는 결합조회 트랜잭션 중에서 아직까지 분지하지 않은 k를 분지노드로 한다. 만약 그러한 k가 없으면 결합을 사용하는 조회 트랜잭션 중에서 분지하지 않은 새로운 테이블 i와 테이블 j를 찾아서 테이블 j의 임의의 컬럼 k를 분지노드로 한다.

단계 3 : [분지노드의 최저한계비용을 계산한다]

4장의 목적식을 사용하여 아직 분지하지 않은 노드의 결정변수는 조회 트랜잭션에서는 1로 하여 계산하고 삽입, 갱신, 삭제 트랜잭션은 0으로 하여 계산하여 가능성이 있는 최저한계비용을 계산한다.

단계 4 : [후보노드 집합에 분지한 노드를 삽입한다]

분지가 잎새노드(leaf node)까지 되었으면 가능해로, 아니면 불능해로 표지하고 분지한 노드를 후보노드에 삽입하고 단계 1로 간다.

6. 수치 예제

본 수치 예제는 표 1과 같이 정규화된 테이블 설계가 주어져 있을 때 비정규화하여 최적의 설계를 구하는 문제이다. 표 2, 3, 4, 5는 각각의 트랜잭션을 표시한 것이다. block 크기는 1024, blocking 계수는 4로 하고, V_{13}=1, V_{24}=5, V_{34}=5 로 주어진 것으로 가정한다. 또, 단위 액세스 비용은 $1, 단위 저장비용은 $0.1/Kbyte 로 가정한다. 표 1에서 컬럼 이름이 진하게 되어있는 것은 외부키이고 인덱스되어 있다.

표 1. 예제 테이블

테이블 번호	테이블 이름	로우 개수	기본키 높이	컬럼 번호	컬럼 이름	길이
1	customer	5,000	2	1	customer_id	5
				2	customer_name	20
				3	address	100
				4	phone	15
2	product	5,000	2	1	product_id	5
				2	product_name	30
				3	product_size	5
				4	standard_price	10
3	order	5,000	2	1	order_id	5
				2	**customer_id**	5
				3	order_date	7
				4	required_date	7
				5	ship_address	100
4	order_detail	25,000	3	1	**order_id**	5
				2	**product_id**	5
				3	quantity	5
				4	unit_price	10

표 2. 예제 삽입 트랜잭션

트랜잭션 번호	테이블 번호	발생 횟수
1	1	1
2	2	1
3	3	10
4	4	50

표 3. 예제 갱신 트랜잭션

트랜잭션 번호	테이블 번호	컬럼 번호	발생 횟수
1	2	3	1
2	3	4	1
3	4	3,4	2

표 4. 예제 삭제 트랜잭션

트랜잭션 번호	테이블 번호	발생 횟수
1	4	2

표 5. 예제 조회 트랜잭션

트랜잭션 번호	테이블 번호	칼럼 번호	액세스 방법	인덱스 높이	액세스 로우 개수	발생 횟수
1	2	3	인덱스	2	1	100
2	3	3	인덱스	2	1	50
	1	2	인덱스	2	1	
3	4	3	인덱스	3	10	10
	2	2,3	순차	-	5,000	
4	4	3	인덱스	3	1	10
	3	3	인덱스	2	5	
5	1	2,3,4	인덱스	2	1	100
6	3	1,2,3,4,5	순차	-	5,000	5

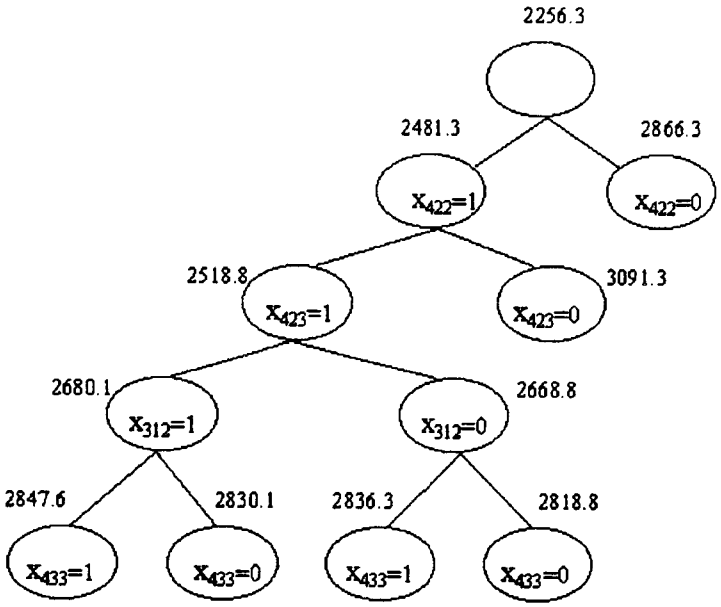

그림 2. 분지한계법을 이용한 해법

본 예제에서 주어진 정규화 설계에서의 총 비용은 $3166.3 이고 본 연구에서의 비정규화한 설계의 최적해는 $2818.8 으로서 $347.5 의 비용이 감소하였다.
최적해는 order_detail에 product의 product_name, product_size를 복사해 두는 비정규화된 테이블 설계이다.

7. 결론

본 연구에서는 데이터베이스의 수행도를 향상시키기 위한 방법 중에서 정규화된 데이터베이스 설계를 비정규화된 데이터베이스 설계로 조율하는 방법을 수리적으로 모델링하고 분지한계법을 사용하여 최적해를 구하였다.

본 연구의 비정규화는 조회뿐만 아니라 삽입, 갱신, 삭제, 저장 비용을 고려하였고 데이터베이스의 필수 조건인 일관성을 유지하는 데 드는 비용도 고려하였다.

데이터베이스 설계는 빈번하게 하는 것이 아니고 그 설계 주기가 매우 크므로 최적해를 구하는 시간이 다소 많이 걸리더라도 가급적 최적해를 구해야 한다.

본 연구의 최악의 complexity는 $O(2^n)$으로서 지수형이지만 4장에서의 조건에 맞는 결정변수의 수는 많지 않으며, 또한 중요한 트랜잭션에 대해서만 데이터를 수집하여 분석하게 되므로 본 연구의 최적해법은 매우 유용한 해법이라고 할 수 있다.

참 고 문 헌

1. 강맹규, *네트워크와 알고리듬*, 박영사, 서울, 1991.
2. Armstrong, E., S. Bobrowski, J. Frazzini, B. Linden, and M. Pratt, *ORACLE7 Server Application Developer's Guide*, Oracle Corporation, Redwood, CA, 1992.
3. Batini, C., S. Ceri, and S. B. Navathe, *Conceptual Database Design*, The Benjamin/Cummings Publishing Company, Inc., Redwood, CA, 1994.
4. Bobrowski S., E. Armstrong, C. Closkey, and B. Linden, *ORACLE7 Server Administors Guide*, Oracle Corporation, Redwood, CA, 1992.
5. Bobrowski, S., E. Armstrong, C. Closkey, B. Linden, and M. Pratt, *ORACLE7 Server Concepts Manual*, Oracle Corporation, Redwood, CA, 1992.
6. Chu, W. W. and Ion T. I., "A Transaction-Based Approach to Vertical Partitioning for Relational Database Systems," *IEEE Transactions on Software Engineering*, Vol. 19, No. 8, pp. 804-812, 1993.
7. Corrigan P. and M. Gurry, *ORACLE Performance Tuning*, O'Reilly & Associates, Inc., Sebastopol, CA, 1993.
8. Elmasri, R. and S. B. Navathe, *Fundamentals of Database System*, The Benjamin/Cummings Publishing Company, Inc., Redwood, CA, 1994.
9. Hawryszkiewycz, I. T., *Relational Database Design*, Prentice Hall, Englewood Cliffs, NJ, 1990.
10. Hoffer, J. A., "An Integer Programming Formulation of Computer Data Base Design Problems," *Information Sciences*, Vol. 11, pp. 29-48, 1976.
11. Lee, H., "Justifying Database Normalization: A Cost/Benefit Model, *Information Processing & Management*," Vol. 31, No. 1, pp. 59-67, 1995.
12. Rajiv M. D. and B. Gavish, "Models for the Combined Logical and Physical Design of Databases," *IEEE Transactions on Computers*, Vol. 38, No. 7, pp. 955-967, 1989.
13. Rodgers, U., "Denormalization: Why, What, and How?," *Database Programming & Design*, Vol. 2, No. 12, pp. 46-53, 1989.
14. Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-Oriented Modeling and Design*, Prentice-Hall International Inc., Cliffs, NJ, 1991.
15. Teorey, T. J., *Database Modeling & Design: The Fundamental Principles*, Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1994.
16. Westland, J. C. "Economic Incentives for Database Normalization," *Information Processing & Management*, Vol. 28, No. 5, pp. 647-662, 1992.