Journal of the Society of Korea Industrial and Systems Engineering
Vol, 26, No. 3, pp.1－9, September 2003.

관계형 데이터베이스에서 응답시간에 제약이 있는 경우 최적 역정규화 방법

장 영 관

삼척대학교 시스템정보공학과

An Optimal Denormalization Method in Relational Database with Response Time Constraint

Young-Kwan Jang

Dept. of System and Information Engineering, Samcheok National University

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

In this research, I model a database design method considering denormalization of duplicating attributes in order to reduce frequent join processes. This model considers response time for processing each select, insert, update, delete transaction, and for treating anomalies. A branch and bound method is proposed for this model.

Keywords : Database Design, Denormalization

1. 서론 및 기존 연구

1.1 서론

데이터는 여러 부문에서 중요한 전략적 자원이 되어 가고 있다. 이전에는 데이터를 주로 화일로 관리하여 왔으나 정보처리 요구가 증가함에 따라 DBMS 사용이 급격히 증가하고 있다. DBMS 중에서 현재 가장 많이 사용하고 있는 것은 관계형 DBMS(relational DBMS)이다.

크고 복잡한 시스템에서 관계형 DBMS의 성능(performance)은 주요한 논점이 되고 있다. 성능은 트랜젝션(transaction)을 처리하는 데 소요되는 응답시간(response time)을 말하며, 일반적으로 트랜젝션을 처리하기 위해 필요한 디스크 액세스(access) 횟수에 의해 결정된다[9, 12].

DBMS의 성능을 향상시키는 문제는 데이터베이스 설계 단계에서부터 프로그램의 검증 단계까지 모든 단계에서 해결해야만 하고, 심지어 시스템이 완성되어 운용되고 있는 경우에도 환경의 변화에 따라 지속적으로 해결해야 하는 문제이다.

이전에 화일로 데이터를 관리할 때에는 성능을 개선하기가 쉬웠으나 정보 기술이 발전하고 관계형 데이터베이스를 사용함에 따라 어렵게 되었다. 특히, 관계형 데이터베이스를 사용함으로써 사용자가 직접 질의어를 사용하여 결과를 얻을 수 있게 됨에 따라 사용자들은 더 많은 조회 요구를 하게 되고 웹(Web)을 기반으로 하는 애플리케이션(application)이 증가함에 따라 시스템에 부담을 가중시키고 있다. 이러한 성능문제를 해결하고 데이터베이스의 응답시간이 최소화되도록 하기 위해 본 연구에서는 역정규화 방법을 사용하고, 수리적으로 모델링하여 분지한계법을 사용하여 최적해를 구한다.

1.2 기존 연구

Roger[17]는 논리적 데이터베이스 설계를 그대로 물리

적 데이터베이스 설계로 전환하여 적절한 성능을 얻기는 어려우므로 경우에 따라서 역정규화는 필요하다고 말하였다. Rajiv[16]는 물리적 데이터베이스 설계와 논리적 데이터베이스 설계를 수리모형화하여 열거법으로 해를 구하였다. 물리적 설계에서는 정렬과 인덱스(indexing)를 고려하고, 논리적 설계에서는 범용 릴레이션(universal relation)을 분해(decomposition)하는 것을 고려하였다. Corrigan과 Gurry[9]는 성능을 향상시키기 위해 데이터베이스 설계 단계에서부터 프로그램의 검증 단계까지 각각의 단계에서 고려하고 적용할 수 있는 매우 실제적인 방법을 제시하였다. 논리적 데이터베이스 설계 단계에서 성능을 향상시킬 수 있는 방법에 대한 연구로는 Lee[15]의 연구가 있다. Lee는 무조건 고차의 정규화 형태로 설계하는 것 대신에 비용과 수익을 고려하여 의사결정나무 방법으로 정규화형을 결정하는 방법을 제시하였다. Lee는 정규화하는 과정에서 비용과 수익을 비교하여 수익이 더 큰 경우는 정규화 과정을 계속하고, 그렇지 않은 경우에 더 이상 정규화하지 않음으로써 정규화형을 결정하는 방법을 사용하였다. Lee는 비용 요소로서 저장비용(storage cost), 이변비용, 그리고 조인비용(join cost)을 고려하였다.

논리적 데이터베이스 설계 단계와 물리적 데이터베이스 설계 단계에서 성능 향상을 위한 연구로는 Dewan[10], Cerpa[8] 등의 연구가 있다. Dewan은 논리적 데이터베이스 설계와 물리적 데이터베이스 설계를 수리모형화하여 열거법으로 해를 구하였다. 논리적 설계에서는 범용 릴레이션을 분해(decomposition)하는 것을 고려하고, 물리적 설계에서는 정렬과 인덱스를 고려하였다.

Cerpa는 1995년에 논리적 데이터베이스 설계단계 이후, 물리적 데이터베이스 설계단계 이전에 논리적 데이터베이스 설계를 수정하여 성능을 향상시키는 지침을 제시하였다. Cerpa는 실제적인 데이터베이스 설계 경험에 근거한 15 가지 유용한 지침을 제시하였다. Cerpa의 지침 중에서 역정규화에 관한 기법은 속성의 복사, 파생된 데이터를 위한 속성의 추가, 릴레이션의 조인, 특별히 필요한 데이터를 위한 릴레이션의 생성 등이다.

역정규화를 이용하여 데이터베이스의 성능을 향상시키기 위한 연구는 Rodgers[17], Janas[14], Sanders와 Shin[18] 등의 연구가 있다.

Rodgers는 1989년에 논리적 데이터베이스 설계를 그대로 물리적 데이터베이스 설계로 전환하여 적절한 성능을 얻기는 어려우므로 경우에 따라서 역정규화는 필요하다고 강조하였다. 또, Rodgers는 대부분의 데이터베이스 설계 방법론이 논리적 데이터 설계를 물리적 데이터베이스 설계로 변환하여 적절한 성능을 얻는 문제를 단지 역정규화로 인한 이변의 발생을 경고하는 정도로 그치고 있고 역정규화를 언급한 것은 역정규화가 종종 필요하다는 것의 반증이라고 하였다.

Rodgers는 역정규화의 대상으로 두 개의 릴레이션이 1:1의 관계, 두 개의 릴레이션이 M:N의 관계, 참조 데이터, 매우 자세한 데이터를 가지는 릴레이션, 파생된 데이터 등을 제시하였다. Rodgers는 성능을 향상시키기 위해 역정규화가 필요하다고 최초로 역설하였고 그 방법으로 매우 유용한 지침을 제시하였다. 그러나 Rodgers의 연구는 역정규화에 의한 부작용인 이변을 처리하는 명확한 방안을 제시하지 않았고 수리적인 모델을 제시하지 않았다. 본 연구는 역정규화에 따른 이변을 처리하는 명확한 방안을 제시하며 수리적으로 모형화하여 최적화하는 방법을 제시한다.

Janas는 1995년에 본 연구에서와 같이, 정규화된 테이블을 역정규화하는 방법으로서 속성을 중복시키는 방법을 제안하였는데 먼저 정규화를 한 다음 역정규화를 하는 방법이다. 참조하는 테이블에 참조되는 테이블의 모든 속성을 중복시켜서 조인조회(join query)의 효율을 향상시키는 것이다. 그는 참조되는 테이블의 모든 속성을 기본키(primary key)에 포함시키고, 참조하는 테이블의 외부키(foreign key)에 중복시킨 속성을 포함시킴으로써 참조무결성 제약으로 갱신이변을 방지하는 방법을 제안하였다. Janas의 역정규화 방법은 다른 연구에 비해 체계적인 좋은 방법이며, 특히 참조무결성 제약을 이용하여 갱신이변을 방지한 것은 이론적으로 기여한 바가 크다고 할 수 있다. 그러나 Janas의 방법은 참조되는 테이블의 모든 속성을 참조하는 테이블에 복사하기 때문에 디스크 저장용량이 크게 증가하고, 만약 참조되는 속성 중에서 조인조회에 거의 쓰이지 않는 속성이 있으면 디스크 저장용량의 증가 뿐만 아니라 전체적인 성능이 저하될 수 있다. Sanders와 Shin은 2001년에 일반적으로 적용할 수 있는 역정규화 방법에 대한 지침을 제시하였다.

본 연구에서는 요구되는 응답시간의 제약이 있는 경우에 데이터베이스의 성능을 향상시키기 위해 정규화된 데이터베이스 설계를 역정규화된 데이터베이스 설계로 조율(tuning)하는 방법을 수리적으로 모델링하고 분지한계법을 이용해 그 최적해를 구한다.

2. 연구 내용

논리적 데이터베이스 설계의 결과인 정규화된 테이블 설계를 역정규화하여 설계하는 방법은 다음과 같다[8, 9].

- 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 속성을 N의 관계의 테이블에 복사하여 조인(join)을 줄일 수 있다.
- 매우 자세한 데이터를 가지는 릴레이션 : 부모(parent) 테이블에 자식(child)을 포함시킨다.
- 파생된(derived) 속성 : 한 개의 속성을 추가하여 반복적인 계산을 피한다.
- 보고서용으로 속성을 발췌하여 새로운 테이블을 만든다.

위에서 나열한 역정규화 방법은 각각의 특수한 상황에서만 적용할 수 있고 일반적으로 정형화하기는 매우 힘들다. 또한 두 개의 테이블을 한 개의 테이블로 만들 경우에 삽입, 갱신, 삭제 이변(anomaly)이 발생하여 부정확한 정보가 되거나 정보를 표현할 수 없는 경우가 발생한다. 본 연구에서는 먼저 실제적으로 시스템을 구축할 때 널리 쓰이는 제 3 정규형 또는 그 이상의 정규형으로 모든 테이블을 정규화한 다음 그 테이블 설계를 가지고 역정규화하는 방법을 사용한다. 두 개의 테이블을 i, j라고 하고, 테이블 i는 테이블 j의 기본키(primary key)를 참조하는 참조 무결성 제약(referential integrity constraint)을 가지는 것을 i=>j 라고 하면 테이블 i에 테이블 j의 속성들을 복사해 둠으로써 빈번한 조인을 방지하여 조회 트랜젝션의 성능을 향상시키고 전체적인 성능을 향상시킨다. 이 결과는 다음의 절충(trade-off) 관계가 있다.

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

본 연구의 비용 요소는 디스크를 액세스하는 데 드는 시간이다. 본 수리 모형은 삽입, 갱신이변(anomaly)에 기인한 데이터 일관성 유지 비용을 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. 순차 액세스에서는 테이블의 모든 튜플을 액세스해야 하고, blocking에 의해 한 번의 액세스로 여러 block을 액세스할 수 있다. 인덱스 액세스에서는 먼저, 인덱스를 인덱스 높이(height) 만큼을 액세스한 다음 실제 데이터가 있는 데이터 block을 한 번 액세스한다.

3. 수리 모형

3.1 수리모형의 가정

(1) 모든 테이블의 튜플 크기는 1 블록보다 작고 저장 용량에 제한이 없다.

중대형 데이터베이스 시스템에서 블록 크기는 일반적으로 2K, 4K, 혹은 그 이상인 경우도 있다. 그러므로 본 연구에서 제시하는 역정규화 방법에 의해 테이블의 튜플 크기가 다소 커질 수 있으나 복사할 수 있는 속성을 모두 복사하는 것이 아니라 최적의 성능을 보장하는 속성만을 복사하므로 특별한 경우가 아니면 대부분의 튜플 크기는 1 블록보다 작다고 할 수 있다. 또한 보조기억 장치의 값이 저렴하여 본연구에 의한 저장 용량의 증가는 큰 문제가 될 수 없으므로 저장 용량에 대한 제한은 없다고 가정한다.

(2) 기본키에 대한 인덱스가 존재한다.

기본키는 개체 무결성을 유지하는 제약으로서 관계형 데이터베이스에서는 인덱스로 구현하고 있다. 만약 기본키에 대한 제약이 없으면 같은 기본키를 가지는 튜플이 존재할 수 있으므로 기본키에 대한 인덱스는 필수적이라 할 수 있다.

(3) 온라인 트랜젝션을 최적화 대상으로 한다.

일반적으로 트랜젝션은 온라인 트랜젝션과 일괄작업 트랜젝션으로 구분할 수 있다. 그 중에서 일괄작업은 예비(backup)작업, 이전(migration), 새로운 DBMS 설치 등과 같은 경우에 수행되고 이러한 일괄작업은 데이터베이스 시스템에 부하가 적은 시간에 이루어지는 것이 일반적이고 온라인 트랜젝션에 비해 상대적으로 성능이 크게 문제가 되지 않으므로 본 연구에서는 온라인 트랜젝션을 최적화 대상으로 한다

3.2 기호 정의

본 연구에서 사용하는 기호는 다음과 같다. 대문자로 표시된 기호는 상수이고 소문자 i, j, m, k, t는 첨자이며, 소문자 s_i, x_{ijk}, y_{ij}, g_{ij}, z_{tjk}는 결정변수이다.

i, j, m : 테이블
k : 속성
t : 삽입, 갱신, 삭제, 조회 트랜젝션
$[a]$: a 보다 크거나 같은 가장 작은 정수
B : 블록 크기
M : 복사할 수 있는 속성의 개수보다 큰 수
W : 블록킹 계수×블록 크기
RT : 임의 액세스 1회 처리하는 평균 시간

ST : 순차 액세스 1회 처리하는 평균 시간
C_t : 트랜잭션 t의 처리 시간 제약
D_t : 삭제 트랜잭션 t의 대상이 되는 테이블
F_t : 트랜잭션 t의 발생 빈도
H_i : 테이블 i에서 기본키 인덱스의 높이
H_{ti} : 트랜잭션 t에서 사용하는 테이블 i의 인덱스 높이
H_{ij} : 테이블 j를 참조하는 테이블 i의 외부키에 대한 인덱스 높이
I_t : 삽입 트랜잭션 t의 대상이 되는 테이블
E_{ti} : 트랜잭션 t의 대상이 되는 테이블 i의 속성 집합
N_{ti} : 트랜잭션 t에서 액세스하는 테이블 i의 튜플 개수
L_i : 정규화된 테이블 i의 튜플 크기
L_{ik} : 테이블 i의 속성 k의 크기
P_t : 조회 트랜잭션 t에서 인덱스 액세스하는 테이블 집합
$\overline{P_t}$: 조회 트랜잭션 t에서 순차 액세스하는 테이블 집합
T^J : 조인이 필요한 조회 트랜잭션의 집합
T^S : 단일 테이블 조회 트랜잭션의 집합
R_i : 테이블 i의 튜플 개수
s_i : 테이블 i의 튜플 크기
U_t : 갱신 트랜잭션 t의 대상이 되는 테이블
V_{ij} : 테이블 i의 한 개의 튜플에 대응하는 테이블 j의 평균 튜플 개수

$$D_{ij}=\begin{cases}1: & \text{테이블 } j\text{를 참조하는 테이블 } i\text{의 외부키에 대한 인덱스가 있을 때}\\ 0: & \text{그렇지 않을 때}\end{cases}$$

$$\overline{D_{ij}}=\begin{cases}1: & \text{테이블 } j\text{를 참조하는 테이블 } i\text{의 외부키에 대한 인덱스가 없을 때}\\ 0: & \text{그렇지 않을 때}\end{cases}$$

$$I_{tij}=\begin{cases}1: & \text{삽입 트랜잭션 } t\text{에서 테이블 } j\text{를 참조하는 테이블 } i\text{의 외부키가 널이 아닌 값일 때}\\ 0: & \text{그렇지 않을 때}\end{cases}$$

$$U_{tij}=\begin{cases}1: & \text{갱신 트랜잭션 } t\text{에서 테이블 } j\text{의 기본키를 참조하는 테이블 } i\text{의 외부키를 널이 아닌 값으로 갱신할 때}\\ 0: & \text{그렇지 않을 때}\end{cases}$$

$$x_{ijk}=\begin{cases}1: & \text{테이블 } i\text{에 테이블 } j\text{의 속성 } k\text{가 복사될 때}\\ 0: & \text{그렇지 않을 때}\end{cases}$$

$$y_{ij}=\begin{cases}1: & \text{테이블 } i\text{에 테이블 } j\text{의 속성이 한 개 이상 복사될 때}\\ 0: & \text{그렇지 않을 때}\end{cases}$$

$$g_{ij}=\begin{cases}1: & \text{참조하는 테이블 } i\text{에 복사한 속성을 참조되는 테이블 } j\text{에서 한 개 이상 갱신할 때}\\ 0: & \text{그렇지 않을 때}\end{cases}$$

$$z_{tij}=\begin{cases}1: & \text{조회 트랜잭션 } t\text{에서 액세스하는 테이블 } j\text{의 모든 속성이 조인되는 테이블 } i\text{에 복사될 때}\\ 0: & \text{그렇지 않을 때}\end{cases}$$

3.3 수리 모형의 정식화

본 수리 모형은 모든 삽입, 갱신, 삭제 및 조회 트랜잭션을 처리하기 위해 필요한 I/O 시간의 합을 최소화하는 문제로서 다음과 같이 표현된다. 목적식의 모든 비용은 디스크 액세스 횟수와 액세스 방법(임의, 순차)에 따라 임의(random) 액세스(인덱스 액세스)인 경우는 액세스 횟수×RT가 처리시간이 되고, 또 순차액세스인 경우는 액세스 횟수×ST가 처리시간이 된다. 본 수리모형에서 $s_i=L_i+\sum_{j\neq i}\sum_k L_{jk}x_{ijk}$ 으로서 튜플의 크기를 나타낸다.

Min

$$\sum_t F_t \sum_{i\in I_t} RT[1+\sum_{j\neq i} y_{ij}(H_j+1)] \quad \cdots\cdots (3.1)$$

$$+\sum_t F_t \sum_{i\in U_t}[RT\cdot N_{ti}(H_i+2)$$
$$+\sum_{j\neq i} RT\cdot y_{ij}\cdot U_{tij}\cdot N_{ti}(H_j+1)$$
$$+\sum_{j\neq i} g_{ji}\cdot N_{ti}\{D_{ji}\cdot RT\cdot V_{ij}(H_{ji}+2) \quad \cdots\cdots (3.2)$$
$$+\overline{D_{ji}}\left(ST\left\lceil\frac{s_j\cdot R_j}{W}\right\rceil+RT\cdot V_{ij}\right)\}\Big] \quad \cdots\cdots (3.3)$$
$$+\sum_t F_t \sum_{i\in D_t} RT\cdot N_{ti}(H_i+2) \quad \cdots\cdots (3.4)$$
$$+\sum_t F_t[RT\sum_{i\in P_t} N_{ti}(1-z_{tji})(H_{ti}+1)$$
$$+ST\sum_{i\in\overline{P_t}}(1-z_{tji})\left\lceil\frac{s_i\cdot R_i}{W}\right\rceil\Big] \quad \cdots\cdots (3.5)$$

s.t.

$$\sum_k x_{ijk}-M\cdot y_{ij}\leq 0 \qquad \forall i,j \quad \cdots\cdots (3.5)$$
$$\sum_{k\in E_{ij}} x_{ijk}-M\cdot g_{ij}\leq 0 \qquad \forall i,j \quad \cdots\cdots (3.6)$$

$$z_{tij}=\begin{cases}\prod x_{ijk} & k\in E_{tj},\ i,j\in P_t\cup\overline{P_t},\ i=>j,\ t\in T^j\\ 0 & i,j\in P_t\cup\overline{P_t},\ i=>j,\ t\in T^s\end{cases} \quad \cdots\cdots (3.7)$$

$$z_{tij} + z_{tjm} \le 1 \quad i, j, m \in P_t \cup \overline{P_t}, \quad i => j, \; j => m, \; t \in T^j \quad \cdots\cdots (3.8)$$

$$x_{ijk}, \; y_{ij}, \; g_{ij}, \; z_{tij} \in \{0,1\} \quad \forall i, j, k \quad \cdots\cdots (3.9)$$

$$\sum_{i \in I_t} RT[1 + \sum_{j \ne i} y_{ij}(H_j + 1)] \le C_t \quad \forall t \quad \cdots\cdots (3.10)$$

$$\begin{aligned} &\sum_{i \in U_t} [RT \cdot N_{ti}(H_i + 2) \\ &+ \sum_{j \ne i} RT \cdot y_{ij} \cdot U_{tij} \cdot N_{ti}(H_j + 1) \\ &+ \sum_{j \ne i} g_{ji} \cdot N_{ti}\{ D_{ji} \cdot RT \cdot V_{ij}(H_{ji} + 2) \\ &+ \overline{D_{ji}}\left(ST \lceil \frac{s_j \cdot R_j}{W} \rceil + RT \cdot V_{ij}\right) \}] \le C_t \end{aligned} \quad \cdots (3.11)$$

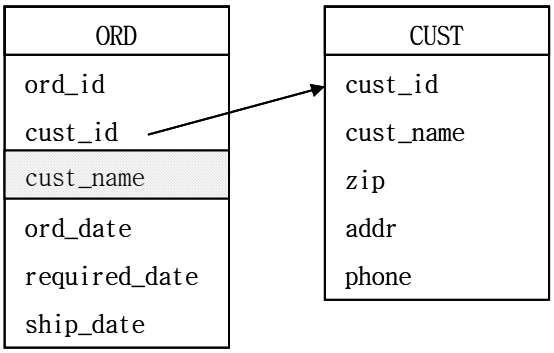
$$\sum_{i \in D_t} RT \cdot N_{ti}(H_i + 2) \le C_t \quad \forall t \quad \cdots\cdots (3.12)$$

$$\begin{aligned} &RT \sum_{i \in P_t} N_{ti}(1 - z_{tji})(H_{ti} + 1) \\ &+ ST \sum_{i \in \overline{P_t}} (1 - z_{tji}) \lceil \frac{s_i \cdot R_i}{W} \rceil \le C_t \quad \forall t \end{aligned} \quad \cdots\cdots (3.13)$$

x_{ijk} 는 결정변수로서 테이블 i에 테이블 j의 속성 k를 복사할 것인지 여부를 표시한다. 만약 x_{ijk} = 1 이라고 결정되면 테이블 i에 새로운 속성이 한 개 추가된다. x_{ijk}가 고려 대상이 되기 위해서는 (i,j,k)가 다음의 세 가지가 동시에 만족할 경우이다. 그렇지 않으면 변수 x_{ijk} 는 고려할 필요가 없다.

- 테이블 i에는 테이블 j의 기본키를 참조하는 외부키 속성이 있어야 한다.
- 테이블 i와 테이블 j를 조인하여 조회하는 트랜젝션이 있어야 한다.
- 그 조회 트랜젝션에서 테이블 j의 속성 k에 대한 조회 요구가 있어야 한다.

<그림 1>에서 ORD 테이블의 cust_id 속성은 CUST 테이블의 기본키인 cust_id를 참조하는 외부키고, ORD 테이블(i) 에 CUST 테이블(j)의 cust_name 속성(k)를 복사한 역정규화 설계를 나타낸다.



<그림 1> 역정규화한 테이블 설계의 예

3.3.1 삽입 트랜젝션 처리 비용

삽입 비용은 테이블 i에 새로운 튜플을 삽입할 때 발생하는 액세스 횟수와 복사된 속성의 값을 유지시키기 위한 테이블 j에 대한 액세스 횟수에 임의 액세스 시간을 곱한 것으로서 수식(3.1)과 같이 표현된다. 여기서 테이블 j는 기본키에 의해 1개의 튜플을 액세스 한다. 테이블 j에 대한 액세스는 기본키에 대한 높이(H_j) 만큼 인덱스를 액세스하고 또, 데이터 블록을 한 번 액세스 한다. y_{ij}는 추가된 결정변수로서 테이블 i에 테이블 j의 속성이 하나라도 복사하면 1이 되는 변수이다. 예를 들어, <그림 1>에서 ORD에 새로운 튜플이 삽입되는 경우에 데이터 일관성을 유지하기 위해 cust_name을 CUST 테이블에서 기본키에 의해 조회하는 비용이 추가된다.

3.3.2 갱신 트랜젝션 처리 비용

갱신 비용은 복사한 테이블의 외부키가 인덱스되어 있을 경우에 다음과 같이 표현된다. 갱신은 먼저 기본키에 대한 인덱스의 높이 만큼 액세스하고, 데이터 블록을 1 회 조회하고 1 회 갱신하는 액세스를 해야 한다. 또한 복사해 둔 속성이 있는 경우에 데이터 일관성을 유지하기 위한 액세스 비용이 추가되어 수식(3.2)와 같다. 예를 들어, <그림 1>에서 ORD의 cust_id가 갱신되는 경우에 데이터 일관성을 유지하기 위해 CUST 테이블에서 1개의 튜플을 조회해야 한다. 또한 CUST 테이블의 cust_name이 갱신되는 경우에 ORD에 복사한 cust_name을 모두 갱신해야 한다.

3.3.3 삭제 트랜젝션 처리 비용

삭제 비용은 수식(3.3)과 같이 표현되며, 본 연구에서는 속성을 복사해 두는 역정규화 방법을 사용하기 때문에 삭제 이변이 발생하지는 않는다.

3.3.4 조회 트랜젝션 처리 비용

조회 비용은 다음과 같이 순차 액세스의 경우와 인덱스 액세스 의 경우의 두 가지로 수식(3.4)와 같이 표현된다. 여기서 z_{tji}는 추가된 결정 변수로서 조인조회에서 필요한 테이블 i의 속성이 모두 테이블 j 에 복사되어 있으면 1이 되어 테이블 i를 액세스하지 않아도 조인조회의 결과를 얻을 수 있도록 한다.

3.3.5 제약 조건

수식 (3.5) 및 (3.6)에서 M은 상수로서 크기가 큰 수를 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. 수식 (3.7)의 제약조건은 i=>j 의 관계에 있는

테이블의 조인조회 트랜젝션을 처리하기 위한 제약으로서 만약, 조인조회에서 필요한 테이블 j의 모든 속성이 테이블 i에 복사되어 있으면 z_{tji}는 1로 제약되어 그 조회를 처리하기 위한 테이블 j의 액세스는 0이 될 수 있도록 한다. 수식 (3.8)은 세 개의 테이블 이상을 조인하는 조회 트랜젝션에 대한 제약이다. 이 수식은 만약, 조회 트랜젝션에서 필요한 테이블 i의 속성들이 테이블 j에 복사되어 있고 또, 테이블 j의 속성들이 테이블 m에 복사되어 있다고 가정하면, 이 트랜젝션을 처리하기 위해서는 테이블 i와 테이블 j 중에서 적어도 한 개는 액세스해야만 하고 또, 테이블 j와 테이블 m 중에 적어도 한 개는 액세스하도록 제약한다. 수식 (3.9)에서의 결정변수 x_{ijk}, y_{ij}, g_{ij}, z_{tji} 는 모두 1 또는 0 의 값을 갖는 이진 정수이다.

수식 (3.10)에서 (3.13) 까지는 각각의 삽입, 갱신, 삭제 및 조회 트랜젝션을 처리하는 데 필요한 디스크 액세스 시간에 대한 제약으로 응답시간 제약을 수식으로 표현한 것이다.

4. 분지한계법을 이용한 알고리듬

4.1 분지한계법의 기본 개념

분지한계법은 기본적으로 열거법(enumeration)의 일종으로서 그 중에서 부분열거법에 속한다. 분지한계법은 각 단계에서 최적해가 포함되어 있을 가능성이 가장 커보이는 부분집합을 선정하여, 이 부분집합에 계산량을 줄일 수 있는 어떤 전략을 사용하여 두 개 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.

4.2 본 연구의 분지한계법

4.2.1 분지 전략

본 연구에서의 분지는 기본적으로 복사할 대상이 되는 속성을 복사한다(x_{ijk} = 1)와 복사하지 않는다(x_{ijk} = 0)로 하면 된다. 본 연구에서는 분지효율을 높이기 위하여 부분분지 기준을 정의하여 x_{ijk} = 1, x_{ijk} = 0로 분지하지 않고 x_{ijk} = 1 또는 x_{ijk} = 0 중에서 한 개로만 분지할 수 있도록 한다. 두 개의 테이블(i, j, i=>j)을 조인하는 조회 트랜젝션에서 액세스 비용을 줄이려면 j에서 필요한 모든 속성을 테이블 i에 복사해야만 한다. 이 경우 만약 복사대상이 되는 속성을 사용해서 두 개의 테이블(i, j)을 조인하는 여타 조회 트랜젝션이 없으면 부분분지를 할 수 있는 경우가 되며, 분지는 후보노드의 값과 동일한 한 개로만 분지한다.

후보노드는 다음의 순서에 의해 선택한다.

(1) 부분분지를 할 수 있는 노드
(2) 같은 E_{ti}에 속하는 노드
(3) 조인조회 비용이 가장 큰 노드

4.2.2 한계 전략

최저한계(lower bound)는 가능한 한 최적해에 가까운 값이어야 하고, 구하는 방법은 간단하고 그 계산량이 적어야 한다. 본 연구에서는 제약식의 개수를 줄이는 방법으로 제약식을 완화하여 두 개의 완화된 문제(*RP1*, *RP2*)를 정의해 각각의 문제를 최적화하여 그 목적식의 합을 최저한계로 한다.

RP1 은 다음과 같다.

Min

$$\begin{aligned}
&\sum_t F_t \sum_{i\in I_t} RT[1+\sum_{j\neq i} y_{ij}(H_j+1)] \\
&+\sum_t F_t \sum_{i\in U_t} [RT\cdot N_{ti}(H_i+2) \\
&+\sum_{j\neq i} RT\cdot y_{ij}\cdot U_{tij}\cdot N_{ti}(H_j+1) \\
&+\sum_{j\neq i} g_{ji}\cdot N_{ti}\{D_{ji}\cdot RT\cdot V_{ij}(H_{ji}+2) \\
&+\overline{D_{ji}}\left(ST\lceil \frac{s_j\cdot R_j}{W}\rceil + RT\cdot V_{ij}\right)\}] \\
&+\sum_t F_t \sum_{i\in D_t} RT\cdot N_{ti}(H_i+2) \quad \cdots\cdots (4.1)
\end{aligned}$$

s.t.

$$\sum_k x_{ijk} - M\cdot y_{ij} \le 0 \quad \forall i,j,\ (i,j,k)\in\Omega \quad \cdots\cdots (4.2)$$

$$\sum_{k\in E_{ij}} x_{ijk} - M\cdot g_{ij} \le 0 \quad \forall i,j,\ (i,j,k)\in\Omega \quad \cdots\cdots (4.3)$$

$$x_{ijk},\ y_{ij},\ g_{ij},\ z_{tij} \in \{0,1\} \quad \forall i,j,k \quad \cdots\cdots (4.4)$$

여기서, Ω는 이미 분지한 노드의 집합이다.

RP2 는 다음과 같다.

Min

$$\begin{aligned}
&\sum_t F_t[RT \sum_{i\in P_t} N_{ti}(1-z_{tji})(H_{ti}+1) \\
&+ ST \sum_{i\in \overline{P_t}} (1-z_{tji})\lceil \frac{s_i\cdot R_i}{W}\rceil] \quad \cdots\cdots (4.5)
\end{aligned}$$

s.t.

$$z_{tij} = \begin{cases} \prod x_{ijk} & k \in E_{tj},\ i,j \in P_t \cup \overline{P_t},\ i => j,\ t \in T^j \\ 0 & i,j \in P_t \cup \overline{P_t},\ i => j,\ t \in T^s \end{cases} \quad \cdots (4.6)$$

$$z_{tij} + z_{tjm} \le 1 \quad i,j,m \in P_t \cup \overline{P_t},\ i => j,\ j => m,\ t \in T^j, (i,j,k) \in \Omega \quad \cdots (4.7)$$

$$x_{ijk},\ y_{ij},\ g_{ij},\ z_{tij} \in \{0,1\} \quad \forall i,j,k \quad \cdots (4.8)$$

4.2.3 분지한계법 절차

본 연구에서 제안하는 분지한계법의 절차는 다음과 같다. 여기서, 분지노드는 이미 분지한 노드 중에서 새로운 분지를 시작하는 노드를 말한다.

단계 0 : [초기화]
분지노드를 비어 있음으로 표지하고 단계 2로 간다.

단계 1 : [분지노드 결정]
아직 분지하지 않은 분지노드 중에서 최저한계비용이 가장 적은 노드를 찾는다. 만약 그 노드가 현재까지의 최적해보다 크거나 같으면 알고리듬을 끝낸다. 더 이상 분지할 노드가 없으면 끝낸다.

단계 2 : [후보 노드 결정]
단계 1에서 분지노드를 x_{ijk} 라고 하면 x_{ijk} 를 사용하는 조인 조회 트랜젝션 중에서 아직까지 분지하지 않은 속성(k^c)을 후보노드로 한다. 만약 그 k^c가 부분 분지 기준에 부합되면 분지는 분지노드 x_{ijk} 의 결정변수 값과 동일하게 분지한다. 그러한 k^c가 없으면 조인을 사용하는 조회 트랜젝션 중에서 조인 비용을 가장 크게 줄일 수 있는 속성을 후보노드로 한다.

단계 3 : [가능해 여부 결정]
3장의 수식(3.10) 부터 수식(3.13) 까지의 응답시간에 대한 제약식을 사용하여 각각의 트랜젝션의 응답 시간이 설정된 시간 제약 이내인지 검사하여 불능해(infeasible solution)면 분지를 끝내고 단계 1로 간다.

단계 4 : [후보노드의 최저한계비용 계산]
본 연구에서 제시한 한계전략을 사용하여 최저한계를 계산한다. 만약 분지가 잎노드(leaf node)까지 되었으면 가능해(feasible solution)로 표지하고 현재까지의 최적해보다 작으면 최적해를 갱신한다.

단계 5 : [분지노드 집합에 분지한 후보노드 삽입]
분지된 후보노드의 분지가 끝나지 않았으면 분지노드 집합에 삽입한다.
단계 1로 간다.

5. 수치 예제

본 수치 예제는 <표 1>에서와 같이 정규화된 테이블 설계가 주어져 있을 때 역정규화하여 최적의 설계를 구하는 문제이다. 정규화된 테이블 설계에서 ord 테이블의 customer_id는 customer 테이블의 customer_id를 참조하는 외부키이며, ord_detail의 ord_id는 ord 테이블의 ord_id를 참조하는 외부키, 그리고 product_id는 product 테이블의 product_id를 참조하는 외부키이다(모든 외부키는 인덱스 되어있고 그 높이는 기본키와 동일한 것으로 가정한다). block 크기는 1024 byte, blocking 계수는 4로 하고, V_{13}=1, V_{24}=5, V_{34}=5 로 주어진 것으로 가정하고 임의 액세스 시간 *RT*는 30 ms, 순차 액세스 시간 *ST*는 15 ms로 한다. 본 예제에서의 삽입, 갱신, 삭제 트랜젝션은 각각 한 개의 튜플을 그 대상으로 하고, 모든 트랜젝션은 3,000 ms 이내의 응답 시간 제약을 가지는 것으로 한다.

<표 1> 예제 테이블

테이블 번호	테이블 이름	튜플 개수	기본 키높이	속성 번호	속성이름	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	standard_price	10
				4	product_size	5
3	ord	5,000	2	1	ord_id	5
				2	customer_id	5
				3	required_date	7
				4	ord_date	7
				5	ship_address	100
4	ord_detail	25,000	3	1	ord_id	5
				2	product_id	5
				3	quantity	5
				4	unit_price	10

<표 2> 예제 삽입 트랜젝션

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

<표 3> 예제 갱신 트랜젝션

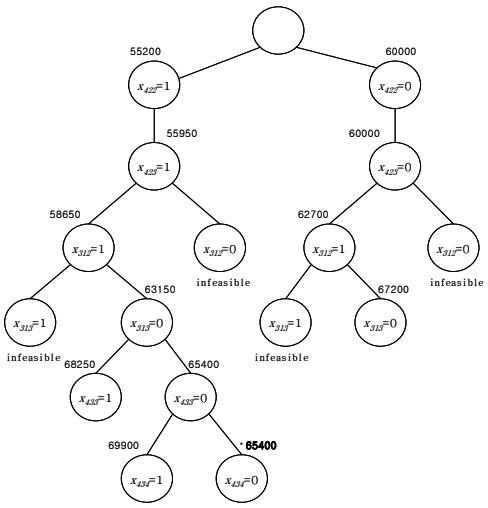
트랜젝션 번호	테이블 번호	속성 번호	발생 횟수
1	2	3	1
2	3	3	1
3	4	3,4	2

<표 4> 예제 삭제 트랜젝션

트랜젝션 번호	테이블 번호	발생 횟수
1	4	2

<표 5> 예제 조회 트랜젝션

트랜젝션 번 호	테이블 번 호	속성번호	액세스 방 법	인덱스 높 이	액세스 튜플 개수	발생 횟수
1	2	3	인덱스	2	1	100
2	3	3	인덱스	2	1	50
	1	2,3	인덱스	2	1	
3	4	3	인덱스	3	10	10
	2	2,3	순 차	-	5,000	
4	4	3	인덱스	3	1	5
	3	3	인덱스	2	5	
5	1	2,3,4	인덱스	2	1	100
6	3	1,2,3,4,5	순 차	-	5,000	5
7	4	2,3	인덱스	3	5	1
	3	4	인덱스	2	10	
	1	2	순 차	-	5,000	



<그림 2> 분지한계법를 이용한 해법

본 예제에서 정규화한 테이블 설계에서 총 응답 시간은 69,315 ms 이다. 본 연구에서 제안한 수리모형을 사용하여 <그림 2>에서와 같이 본 연구에서 제안한 분지한계법을 사용하여 최적해를 구하면 65,400 ms로서 원래의 테이블 설계보다 3,915 ms의 응답시간이 감소하였다. 역정규화된 설계는 ord_detail에 product의 product_name, standard_price, 또 ord에 customer의 customer_name을 복사해 두는 것이다.

6. 결 론

본 연구는 응답시간에 제약이 있는 경우에 제 3 정규형으로 설계된 데이터베이스를 역정규화하여 최적의 성능을 가지는 데이터베이스를 설계하는 방법이다. 본 연구는 참조하는 테이블에 참조되는 테이블의 속성을 복사하는 역정규화 방법을 사용하였다. 본 연구에서는 최적의 성능을 보장하는 속성만을 복사하여 데이터베이스 설계를 하기 위해 수리적으로 모델링하고 분지한계법을 이용하여 그 최적해를 구하였다. 본 연구의 역정규화는 조회 뿐만 아니라 삽입, 갱신, 삭제 트랜젝션을 처리하는 데 필요한 시간을 고려하였고 데이터베이스의 필수 조건인 데이터 일관성을 유지하는 데 필요한 시간도 고려하였다. 본 연구에서는 효율적인 분지전략 및 한계전략을 제시하였고, 그 결과 예제에서 보는 바와 같이 분지 개수가 크게 감소한 것을 알 수 있다. 만약, 시스템의 규모가 매우 큰 경우는 “80-20 법칙”을 적용하여 20% 정도의 트랜젝션만을 최적해의 대상으로 고려하면 본 연구의 모델을 그대로 적용하여 그 최적해를 구할 수 있을 것이다. 향후 본 연구에서 제안한 수리모형에 대한 효율적인 해법에 대한 연구가 필요하다.

참고문헌

[1] 장영관, 강맹규, “관계형 데이터베이스에서 저장용량에 제약이 없는 경우 비정규화를 고려한 데이터베이스 설계,” *공업경영학회지*, 제 19 권, 37 집, 1996.

[2] 장영관, *관계형 데이터베이스의 성능 향상을 위한 비정규화 방법*, 박사학위 논문, 한양대학교 대학원, 서울 1996.

[3] Armstrong, E., S. Bobrowski, J. Frazzini, B. Linden, and M. Pratt, *ORACLE7 Server Application Developer's Guide*, Oracle Corporation, Redwood, CA, 1992.

[4] Batini, C., S. Ceri, and S. B. Navathe, *Conceptual Database Design*, The Benjamin/Cummings Publishing

Company, Inc., Redwood, CA, 1994.

[5] Bobrowski, S., E. Armstrong, C. Closkey, and B. Linden, *ORACLE7 Server Administors Guide*, Oracle Corporation, Redwood, CA, 1992.

[6] Bobrowski, S., E. Armstrong, C. Closkey, B. Linden, and M. Pratt, *ORACLE7 Server Concepts Manual*, Oracle Corporation, Redwood, CA, 1992.

[7] Chu, W. W., Fellow, IEEE, 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.

[8] Cerpa, N., "Pre-physical Data Base Design Heuristics," *Information & Management*, Vol. 28, No. 6, pp. 351-359, 1995.

[9] Corrigan, P. and M. Gurry, *ORACLE Performance Tuning*, O'Reilly & Associates, Inc., Sebastopol, CA, 1993.

[10] Dewan, R. M. 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.

[11] Elmasri, R. and S. B. Navathe, *Fundamentals of Database System*, The Benjamin/Cummings Publishing Company, Inc., Redwood, CA, 1994.

[12] Hawryszkiewycz, I. T., *Relational Database Design*, Prentice Hall, Englewood Cliffs, NJ, 1990.

[13] Hoffer, J. A., "An Integer Programming Formulation of Computer Data Base Design Problems," *Information Sciences*, Vol. 11, pp. 29-48, 1976.

[14] Janas, J. M., "A Systematic Approach to Database Denormalization," *iti'95. Proceedings of the 17th International Conference on Information Technology Interfaces*, pp.323-328, 1995.

[15] Lee, H., "Justifying Database Normalization : A Cost/Benefit Model, *Information Processing & Management*," Vol. 31, No. 1, pp. 59-67, 1995.

[16] 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.

[17] Rodgers, U., "Denormalization : Why, What, and How?," *Database Programming & Design*, Vol. 2, No. 12, pp. 46-53, 1989.

[18] Sanders, G. L. and S. Shin, "Denormalization Effects on Performance of RDBMS," *Proceedings of the 34th Hawaii International Conference on System Sciences*, 2001.

[19] Teorey, T. J., *Database Modeling & Design : The Fundamental Principles*, Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1994.

[20] Westland, J. C. "Economic Incentives for Database Normalization," *Information Processing & Management*, Vol. 28, No. 5, pp. 647-662, 1992.