성능평가를 위한 다양한 분포를 갖는 질의 작업부하의 생성 기법

서 상 구*

A Technique for Generating Query Workloads of Various Distributions for Performance Evaluations

Sang-Koo Seo*

Abstract

Performance evaluations of database algorithms are usually conducted on a set of queries for a given test database. For more detailed evaluation results, it is often necessary to use different query workloads several times. Each query workload should reflect the querying patterns of the application domain in real world, which are non-uniform in the usage frequencies of attributes in queries of the workload for a given database. It is not trivial to generate many different query workloads manually, while considering non-uniform distributions of attributes' usage frequencies. In this paper we propose a technique to generate non-uniform distributions, which will help construct query workloads more efficiently. The proposed algorithm generates a query-attribute usage distribution based on given constraints on usage frequencies of attributes and queries. The algorithm first allocates as many attributes to queries as possible. Then it corrects the distribution by considering attributes and queries which are not within the given frequency constraints. We have implemented and tested the performance of the proposed algorithm, and found that the algorithm works well for various input constraints. The result of this work could be extended to help automatically generate SQL queries for various database performance benchmarking.

※ 본 논문은 2000년도 광운대학교 교내학술연구비 지원에 의하여 연구되었음.
* 광운대학교 경영정보학과(skseo@gwu.ac.kr)

1. 서 론

데이터베이스 성능 향상을 위한 질의 최적화, 데이터 저장 기법, 구체화 뷰의 활용, 최적 색인 구성 등 다양한 기법과 도구들이 개발되어 왔다[Goldstein & Larson 2001, Gupta 1997, Chaudhuri & Narasayya 1998, Labrinidis & Roussopoulos 2000]. 또한 새롭게 개발되는 알고리즘을 적용하기 위한 성능평가 작업이 다양한 분야에서 요구되고 있다[Boehme & Rahm 2001, Grinstein *et al* 2001]. 성능평가는 데이터베이스 최적화 알고리즘의 개발 시에 필요할 뿐 아니라 데이터베이스 응용 시스템의 벤치마킹 작업에서도 시스템의 안정성, 유지관리성에 대한 평가와 함께 반드시 포함되는 중요한 요소이다[Elmasri & Navathe 2000, Gray 1993].

성능평가 작업은 주로(가상의) 예제 데이터베이스와 이에 대한 질의들을 이용하게 된다. 이들 질의들과 질의의 실행 빈도 등을 질의 작업부하(workload)라고 한다[McDonell 1995]. 데이터베이스 최적화 알고리즘의 성능 평가 작업과 데이터베이스 응용 시스템의 성능 평가 작업은 동일하지 않다. 데이터베이스 응용 시스템의 성능평가는 주어진 데이터베이스에 대하여 주어진 작업부하(즉, 질의집합)를 가장 가격-효율적으로 처리할 수 있는 데이터베이스 시스템 운영환경(DBMS, H/W 플랫폼, 응용시스템)을 구성하는 것인 반면, 데이터베이스 최적화 알고리즘의 성능 분석은 여러 규모의 데이터베이스 스키마에 대한 다양한 질의 집합에 대한 알고리즘의 대응 능력을 평가할 필요가 있다.

데이터베이스 응용 시스템의 성능평가 환경으로 데이터베이스 벤치마킹 기관인 TPC의 데이터베이스 스키마와 작업부하가 종종 이용된다[TPC 1999]. TPC의 스키마와 작업부하는 TPC 회원사(들)이 제안한 명세서 초안(Draft Specification)을 TPC의 Benchmark Development Subcommittee에서 검토하고, 정식 검토 대상으로 받아들여지면 회원사들의 투표를 통하여 벤치마크 환경으로 최종 결정되는 것으로 알려져 있다[TPC 2001]. 실제 응용환경에서 스키마 크기(i.e., 테이블 수, 속성 수)와 질의부하(i.e., 질의 수, 질의의 복잡도)는 TPC의 그것보다 큰 경우가 더 일반적일 것이다. 질의 최적화, 최적 색인구성, 뷰 최적화 등 데이터베이스 알고리즘의 연구 논문에서 TPC의 TPC-H 또는 TPC-R(이전의 TPC-D)의 벤치마크용 데이터베이스 스키마와 작업부하를 대상으로 하는 성능 평가 결과를 제시하기도 한다[TPC 1999, Chaudhuri & Narasayya 1997, Ligoudistianos *et al* 1998, Gupta *et al* 1997].

여러 논문에서 TPC에서 제공하는 스키마와 질의집합을 이용하는 이유는 TPC 환경이 데이터베이스 연구자들에게 잘 알려져 있기 때문에 연구결과를 표출하기에 적합하기 때문이라고 생각된다. 하지만, 예를 들어, 색인 구성 알고리즘을 개발하는 과정에서는 알고리즘이 적절히 작동하는지를 알아보기 위하여 TPC에 주어진 *하나의* 스키마와 질의집합만으로는 다양한 작업부하에 대한 알고리즘의 색인 선택 능력을 알아보기에 부족할 것이다. 데이터베이스 관련 최적화 알고리즘의 성능을 분석하고 검증하기 위해서는 TPC-H의 환경보다 작은 규모의 작업부하는 물론 이보다 큰 규모의 작업부하도 필요할 것이다. 그러나 임의의 데이터베이스 스키마에 대하여 여러 개의 질의(e.g., > 10)를 수작업으로 직접 작성한다는 것이 매우 번거로운 일이다. 특히, 실세계에서 스키마의 속성들이 이용되는 빈도와 하나에 질의에 포함되는 속성들의 이용분포가 불균등(non-uniform)한 것이 대부분의 현상인데 이를 고려하여 질의 집합을 작성하기란 간단한 일이 아닐 것이다.

본 논문에서는 데이터베이스 알고리즘의 성능 평가 작업에 필요한 다양한 질의 작업부하의 작성을 지원하는 기법을 연구하였다. 데이터베이스 스키마 정보(테이블과 속성, 키, 외래키 등), 질의 수, 각 속성과 질의의 이용 정도(예를 들면 High, Mid, Low)가 주어졌다고 할 때, 각 질의에 이용되는 속성들(또는, 반대로 각 속성이 이용되는 질의들)을 효과적으로 선택하여 주어진 질의-속성 이용조건을 만족하는 질의 집합을 생성하는 일은 단순하지는 않다.

본 연구에서 제안하는 알고리즘은 질의당 이용 속성 수의 분포값과 속성당 이용되는 질의 수의 분포값을 이용하여 두 분포조건을 최대한 만족하는 질의 모형을 생성한다. 실행 결과를 이용하여 다양한 질의 작업부하가 필요한 벤치마크 또는 성능평가 작업에서 적절한 질의-속성 이용 분포를 따르는 질의집합을 작성하는데 유용하게 이용될 수 있을 것이다. 이러한 다양한 분포의 작업부하 생성과 관련된 관심과 연구가 부족했다고 여겨지며, 관련 연구를 찾기가 쉽지 않았다. 본 연구는 이에 대한 초기 연구로서, 앞으로 데이터베이스 알고리즘의 보다 효과적인 성능 분석 작업에 반드시 필요한 연구라고 생각된다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 작업부하의 질의모형을 정의하고 3장에서는 질의 생성 문제의 정의와 본 연구에서의 가정을 설명한다. 4장에서 질의 집합 생성 알고리즘을 제안하고 5장에서 실험 결과를 보인 후, 6장에서 결론을 맺는다.

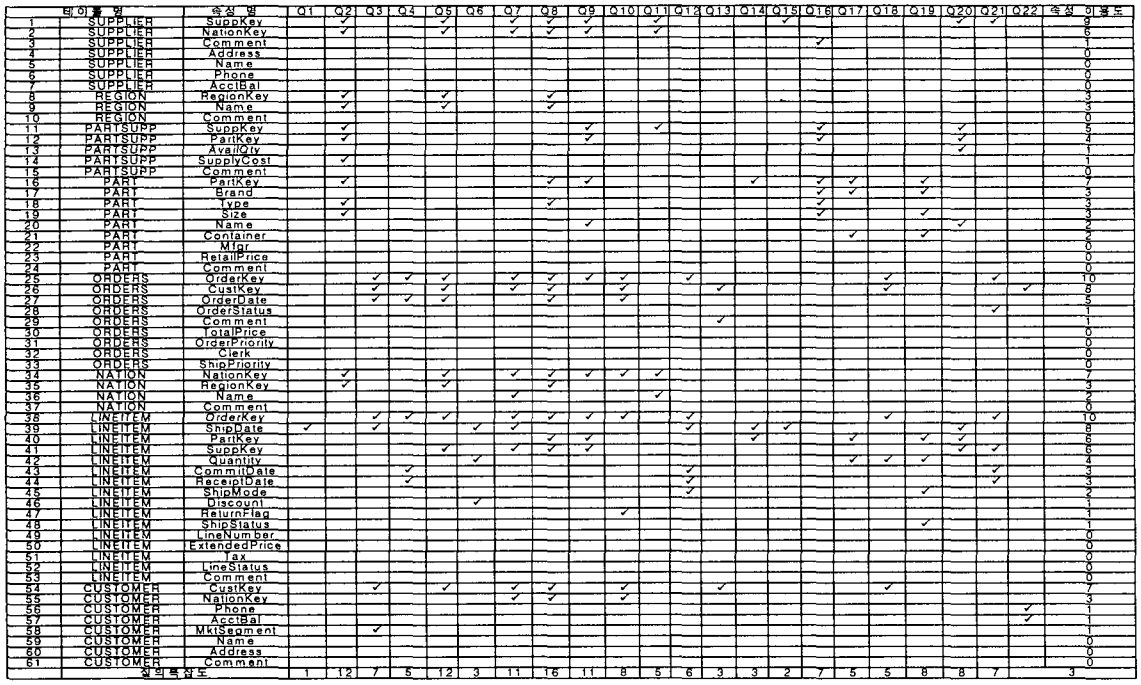
2. 작업부하의 질의집합 모형

실세계 또는 벤치마킹의 작업부하의 각 질의들을 분석해 보면 그 복잡도와 이용 패턴이 불균등한 것이 일반적인 현상이다[Darmont & Schneider 2000, TPC 1999, Gray 1993, Jain 1991]. 즉, 전체 질의집합 가운데 소수의 일부 질의들에 대한 처리비용이 전체 질의처리 비용의 대부분을 차지하는 것을 쉽게 관찰할 수 있다. 또한, 작업부하의 각 질의에서 WHERE절의 검색 조건식에 포함된 속성의 수도 질의 별로 균등하지 않다. OLTP(On-Line Transaction Processing) 응용 환경에서는 간단하고 짧은 트랜잭션을 표현하기 위하여 질의당 이용되는 속성의 수가 대부분 적지만, 데이터 웨어하우징과 같은 응용 환경에서는 검색 조건식에 많은 속성들이 이용되는 복잡한 질의가 전체 작업부하의 대부분을 차지할 수도 있다[Poess & Floyd 2000, Devlin 1997].

속성의 측면에서도 이와 유사한 불균등 분포를 관찰할 수 있다. 즉, 주 키(Primary Key), 또는 외래키(Foreign Key) 속성은 검색 절에 여타 속성들 보다 집중적으로 많이 쓰이며, 대부분의 속성은 단순히 SELECT 절의 출력용으로 이용되는 것이 일반적이다.

성능평가 기관인 TPC의 TPC-H 벤치마크의 질의와 속성 수의 사이에서도 이와 같은 분포를 확인할 수 있다[TPC 1999]. 다음의 (그림 1)은 모두 8개의 테이블, 61개의 속성으로 구성된 TPC-H의 데이터베이스 스키마에서 22개의 검색질의와 각 질의의 WHERE 검색 조건절에 포함된 각 테이블의 속성을 대략적으로 보여주고 있다. 각 행의 V 표시는 그 행의 속성이 해당 열의 질의에 이용되고 있음을 뜻한다. 제일 마지막 열의 ***속성 이용도***는 하나의 속성이 작업부하의 질의들에 이용되는 빈도 수를 나타낸다. 또한 제일 마지막 행에 집계된 수는 하나의 질의에 이용되는 속성의 수로서 그 질의의 ***복잡도***를 의미한다.

(그림 1)에서 알 수 있듯이 질의에 포함된 속성수의 분포와 또 속성이 이용되고 있는 질의



	테이블 명	속성 명	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	속성 이용도
1	SUPPLIER	SuppKey		✓			✓		✓	✓	✓		✓				✓					✓	✓		9
2	SUPPLIER	NationKey		✓			✓		✓	✓	✓		✓												6
3	SUPPLIER	Comment																✓							1
4	SUPPLIER	Address																							0
5	SUPPLIER	Name																							0
6	SUPPLIER	Phone																							0
7	SUPPLIER	AcctBal																							0
8	REGION	RegionKey		✓			✓			✓															3
9	REGION	Name		✓			✓			✓															3
10	REGION	Comment																							0
11	PARTSUPP	SuppKey		✓							✓		✓					✓				✓			5
12	PARTSUPP	PartKey		✓							✓							✓				✓			4
13	PARTSUPP	AvailQty																				✓			1
14	PARTSUPP	SupplyCost		✓																					1
15	PARTSUPP	Comment																							0
16	PART	PartKey		✓						✓	✓					✓		✓	✓		✓				7
17	PART	Brand																✓	✓		✓				3
18	PART	Type		✓						✓								✓							3
19	PART	Size		✓														✓			✓				3
20	PART	Name									✓											✓			2
21	PART	Container																	✓		✓				2
22	PART	Mfgr																							0
23	PART	RetailPrice																							0
24	PART	Comment																							0
25	ORDERS	OrderKey			✓	✓	✓		✓	✓	✓	✓		✓						✓			✓		10
26	ORDERS	CustKey			✓		✓		✓	✓		✓			✓					✓				✓	8
27	ORDERS	OrderDate			✓	✓	✓			✓		✓													5
28	ORDERS	OrderStatus																					✓		1
29	ORDERS	Comment													✓										1
30	ORDERS	TotalPrice																							0
31	ORDERS	OrderPriority																							0
32	ORDERS	Clerk																							0
33	ORDERS	ShipPriority																							0
34	NATION	NationKey		✓			✓		✓	✓	✓	✓	✓												7
35	NATION	RegionKey		✓			✓			✓															3
36	NATION	Name							✓				✓												2
37	NATION	Comment																							0
38	LINEITEM	OrderKey			✓	✓	✓		✓	✓	✓	✓		✓						✓			✓		10
39	LINEITEM	ShipDate	✓		✓			✓	✓					✓		✓	✓					✓			8
40	LINEITEM	PartKey								✓	✓					✓		✓			✓	✓			5
41	LINEITEM	SuppKey					✓		✓	✓	✓											✓	✓		6
42	LINEITEM	Quantity						✓											✓	✓	✓				4
43	LINEITEM	CommitDate				✓								✓									✓		3
44	LINEITEM	ReceiptDate				✓								✓									✓		3
45	LINEITEM	ShipMode												✓							✓				2
46	LINEITEM	Discount						✓																	1
47	LINEITEM	ReturnFlag										✓													1
48	LINEITEM	ShipStatus																			✓				1
49	LINEITEM	LineNumber																							0
50	LINEITEM	ExtendedPrice																							0
51	LINEITEM	Tax																							0
52	LINEITEM	LineStatus																							0
53	LINEITEM	Comment																							0
54	CUSTOMER	CustKey			✓		✓		✓	✓		✓			✓					✓					7
55	CUSTOMER	NationKey							✓	✓		✓													3
56	CUSTOMER	Phone																					✓		1
57	CUSTOMER	AcctBal																					✓		1
58	CUSTOMER	MktSegment			✓																				1
59	CUSTOMER	Name																							0
60	CUSTOMER	Address																							0
61	CUSTOMER	Comment																							0
	질의특성도		1	12	7	5	12	3	11	16	11	8	5	6	3	3	2	7	5	5	8	8	7		3

(그림 1) TPC-H의 질의-속성 간의 이용 관계

의 수의 분포는 균등하지 않고 매우 쏠려있음(skewed)을 확인할 수 있다. 만약 특정한 분포의 질의집합이 필요할 때, 수작업을 통하여 질의와 속성의 이용분포를 고려하여 질의집합을 생성한다는 것은 매우 힘든 일일 것이다. 본 논문에서 제안하는 알고리즘은 이와 같은 불균등 분포를 적절히 생성하는데 그 목적이 있다. 이를 통하여 다양한 질의-속성 이용분포의 질의 작업부하를 생성함으로써, 데이터베이스 알고리즘의 성능평가 및 데이터베이스 응용 시스템의 벤치마킹을 위한 효과적인 작업환경을 구축하는데 도움이 될 것이다.

본 연구에서는 질의와 속성간의 이용 분포를 ***질의집합 모형***이라고 하고, 0과 1의 2진수값으로 구성된 2차원 배열로 표현한다. 2차원 배열의 i-번째 행을 속성 a_i라 하고 j-번째 열을 질의 q_j고 하면, i행 j열의 값은 속성 a_i가 질의 q_j에 이용될 경우에는 1, 그렇지 않으면 0의 값을 갖는 것으로 한다.

3. 문제 정의

3.1 가 정

일반적으로 작업부하는 검색질의와 변경질의(삽입/갱신/삭제)로 구성된다. 본 연구에서 표현하고자 하는 질의-속성 이용 모형은 데이터베이스 색인 관리, 실체화 뷰 구성 등의 알고리즘에서 어떤 테이블의 어떤 속성들이 질의에 어느 정도로 이용되는 지에 대한 분포 모형을 생성하려는 것이므로 주로 검색질의가 해당된다. 그러나 변경질의에서도 검색조건이 이용되기 때문에 이 부분도 질의-속성의 이용분포에 고려해야 할 것이다. 즉, INSERT 문의 경우 WHERE 조건절에서 구한 튜플(들)을 테이블에 추가한다든가, UPDATE 문과 DELETE 문의 경우에 WHERE 조건절에서 구해진 튜플들에 대한 변경과 삭제가 있을 수 있는데, 이 때의 각 조건절 부분도 작업부하로서 속성의 이용도로 고려

할 필요가 있다. 본 논문에서는 작업부하로서 변경질의가 포함될 경우에 변경질의의 각 조건절에 해당되는 부분을 WHERE 절로 갖는 별도의 검색질의를 가정함으로써 질의-속성의 이용 분포 조건으로 고려하기로 한다. SELECT 절, GROUP BY, ORDER BY 절에 포함되는 속성도 질의의 처리에 큰 영향을 미치지만 작업부하로서의 속성은 WHERE절에 포함된 속성만을 표현 대상으로 한다.

일반적으로 작업부하에는 해당 질의의 이용 횟수 또는 중요도 값이 포함된다. 본 연구에서의 작업부하 생성은 각 질의 별로 이용되는 속성 수의 분포와 각 속성이 질의들에 이용되는 분포에 대한 것이므로 질의 자체의 이용횟수 또는 중요도는 고려하지 않는다. TPC 벤치마킹 등에서도 질의의 이용빈도는 별도로 주어지는 방식을 취하고 있다. 또한, 이에 앞서 작업부하 형성을 위한 데이터베이스 스키마 정보와 질의의 수는 문제의 입력 조건으로 주어진다고 가정한다. 스키마 정보는 테이블, 속성, 키, 외래키 등이 해당된다.

질의 복잡도와 속성 이용도 분포값은 두 가지 종류가 있을 수 있다. 질의 복잡도를 기준으로 설명하면 다음과 같다. 먼저 질의들 가운데는 속성을 많이 포함하는 부류와 그렇지 않는 부류들의 비율이 주어질 수 있다. 즉, 속성을 많이 포함하는 정도에 따라 HIGH, MID, LOW 의 그룹으로 나눌 수 있다. 보다 세밀한 작업부하를 생성하고자 할 때에는 VERY HIGH, VERY LOW, 등의 비율을 추가할 수도 있을 것이다. 둘째는 각 부류에서 실제로 어느 정도의 속성을 포함하는가에 대한 것이다. 이것은 속성수에 대한 백분율로 표현될 수도 있고 또는 절대적인 속성 수(또는 범위)로 주어질 수 있으며 둘 다 동일한 의미로 해석될 수 있다. 일반적으로 테이블 수와 속성 수가 많은 대규모 응용의 경우에는 많은 수의 속성을 포함하는 복잡한 질의가 있을 수 있을 것이다. 그렇다 하더라도, 각 질의 그룹 HIGH, MID, LOW 등에서 스키마 규모에 관계없이 일정 비율 만큼의 속성이 이용된다고 가정하기에는 무리일 것이다.

예를 들어, 속성수의 20%를 이용한다고 할 경우, 전체 속성수가 100개인 경우 20개의 속성이 하나의 질의의 WHERE절에 이용되는 경우는 매우 드문 일일 것이다. 따라서, 전체 속성수를 감안하여 질의 부류당 일정 범위의 속성수를 고려하는 것이 일반적일 것으로 판단된다. 예를 들면, HIGH 부류에 속하는 질의들은 20% 정도이며 이들은 대략 전체 속성 가운데 10~15개의 속성들을 이용하는 작업부하를 생각해 볼 수 있다. 따라서, 본 연구에서는 질의의 복잡도는 전체 속성수를 고려한 적절한 속성 수의 범위로 주어진다고 가정한다. 속성 이용도도 마찬가지로 HIGH, MID, LOW 등의 그룹으로 나뉘며, 전체 질의수를 고려하여 각 속성이 이용되는 질의 수의 범위가 문제의 입력으로 주어진다고 가정한다. 즉, 각 속성에는 그 속성을 이용하는 질의의 수의 범위가, 예를 들면, 5~10개와 같이 주어진다고 가정한다.

끝으로, 각 질의의 복잡도와 속성의 이용도가 HIGH, MID, LOW에 속하는 범위의 설정에 대한 것이다. 이것은 성능평가 또는 벤치마킹의 응용 분야에 따라 제안된 알고리즘의 입력으로 임의로 지정하면 될 것이다. 참고로 TPC-H의 경우 질의의 복잡도에 따라 LOW, MID, HIGH 그룹으로 나누었을 때, 각 그룹에 속한 질의 수의 분포는 정규분포 형태에 가까운 것으로 볼 수 있다. 즉, LOW와 HIGH 그룹에 속한 질의는 적은 반면 MID 그룹에 속한 질의의 수가 상대적으로 많다. 속성의 이용도 분포는 주 키와 외래키가 일반적으로 많이 이용되는 것이

보통이므로 이를 감안하여 임의로 지정하면 될 것이다.

3.2 문제 정의

본 연구에서 접근하는 문제는 2절의 질의집합 모형과 3.1절의 가정을 바탕으로 다음과 같이 정의된다.

Given

- 스키마 정보 : 속성 수 m for $a_1, a_2, \cdots, a_m$, where
  - Relation[a_i] is the relation to which the attribute a_i belongs
  - P_Key[a_i] is true if a_i is the primary key of the relation Relation[a_i]
  - F_Key[a_i] is a_j ($i \neq j$) if a_i is a foreign key and a_j is the corresponding primary key ; otherwise nill.
- 목표 질의 수 : n for $q_1, q_2, \cdots, q_n$
- 속성 이용빈도 (즉, 속성별 이용 질의 수) 분포 :
  lower_limit[a_i], upper_limit[a_i] for $1 \leq i \leq m$
- 질의의 복잡도 (즉, 질의당 속성 수) 분포 :
  lower_limit[q_j], upper_limit[q_j] for $1 \leq j \leq n$

Generate

- mxn의 2차원 비트맵 배열 $C_{i,j}$, for $1 \leq i \leq m$ and $1 \leq j \leq n$

With Weak Restrictions

- lower_limit[a_i] $\leq$ count[a_i] $\leq$ upper_limit[a_i], for $1 \leq i \leq m$
- lower_limit[q_j] $\leq$ count[q_j] $\leq$ upper_limit[q_j], for $1 \leq j \leq n$

위에서 Relation[ ], P_Key[ ], F_Key[ ]는 주어진 스키마 정보를 담고 있는 배열로 가정한다. lower_limit[ ]과 upper_limit[ ]는 각 속성의 이용빈도와 질의의 복잡도에 대한 하한 값과 상한 값을 뜻한다. 출력되는 작업부하 모형은 $m \times n$의 2차원 배열로서 항목 $C_{i,j}$는 속성 a_i가 질의 q_j에 이용될 경우 1, 그렇지 않으면 0의 값을 갖는다. 주어진 분포 값에 따라 두 조건을 모두 만족시킬 수 없을 수도 있기 때문에 약한 제약조건이라고 하였고, 따라서, 제안된 알고리즘은 주어진 질의와 속성의 분포를 최대한 만족시키는 것을 목표로 한다. 여기서, count[q_j]와 count[a_i]는 질의 q_j에 포함되는 속성의 수와 속성 a_i의 이용횟수를 각각 의미한다.

이 문제는 일견 단순해 보이지만 실제로 그리 간단한 문제는 아니다. 두 가지 분포를 모두 만족하는 작업부하의 생성 가능성 여부의 판단도 단순하지는 않다. 만약 질의 복잡도에 대한 분포 조건만 고려한다면 각 질의에 대하여 필요만큼의 속성들을 임의로 포함시키면 될 것이다. 이럴 경우, 질의의 복잡도는 만족될 수 있지만, 각 속성들이 질의에 이용되는 이용빈도에 대한 조건은 만족시키지 못하게 될 수 있다. 속성 이용도에 대한 분포 조건만 고려할 때에도 마찬가지일 것이다.

본 논문에서 접근하는 질의-속성간의 분포 문제는 0/1 정수 계획법(0/1 Integer Programming)의 문제 형태와 유사하다[Horowitz and Sahni 1978, Hillier and Lieberman 1990]. 속성들의 의미정보(즉, 주키 및 외래키)가 분포생성에 이용되어야 한다는 점에서 일반적인 0/1 정수 계획법의 문제 형태와 다소 차이가 있을 수 있다. 본 논문은 데이터베이스 알고리즘의 평가와 분석 시에 필요한 다양한 질의 작업부하에 대한 초기 연구이므로, 0/1 정수 계획법을 이용한 해결과 본 논문에서 제시할 알고리즘과의 효율성에 대한 비교는 향후 연구에서 다루

기로 한다.

다음은 문제 예로서 스키마 정보와 질의-속성의 이용 분포 조건 지정을 예시하기 위하여 간략히 구성하였다. 5장에서 이 데이터에 대한 상세한 설명과 함께 제안한 알고리즘의 실행 예를 보여줄 것이다.

● 문제 예제

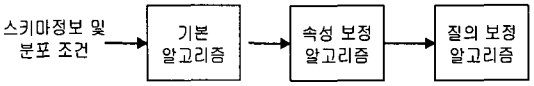
작업부하로 주어진 4개의 테이블 R_1, R_2, R_3, R_4가 각각 $a_1 \sim a_4$, $a_5 \sim a_7$, $a_8 \sim a_{10}$, $a_{11} \sim a_{14}$의 총 14개의 속성을 갖고, $\{a_1\}$, $\{a_5, a_6\}$, $\{a_8\}$, $\{a_{11}\}$가 각 테이블의 주 키이며 a_4, a_5, a_6는 외래키로서 a_{11}, a_1, a_8가 각각의 대응되는 주 키라고 하자. 작성하고자 하는 질의가 10개라고 할 때, 각 속성에는 질의들에 이용되는 이용수의 범위 조건이 주어진다. 각 속성이 질의에 이용되는 수의 범위 및 많은 질의에 이용되는 속성의 개수와 적은 질의에 이용되는 속성 개수의 범위는 임의로 지정할 수 있다. 각 질의에도 질의에 이용되는 속성 수의 범위 조건이 주어진다고 하자. 스키마의 속성정보(i.e., 주 키, 외래키)를 고려하여 속성과 질의의 이용 수의 범위 조건을(최대한) 만족하는 질의집합 모형을 구한다.

4. 질의-속성 분포 알고리즘

4.1 개 요

질의/속성 작업부하를 생성하는 알고리즘은 두 가지 접근 방법이 있을 수 있다. 첫 번째 방법은 질의(또는 속성) 별로 주어진 속성(또는 질의)의 범위 값을 그대로 이용하여 단계적으로 질의와 속성의 이용 관계를 맺어 주는 방법이고 두 번째 방법은 이들 범위 값을 확률 값으로 전환한 후 각 속성이 각 질의에 이용될 확률을 정의하는 확률적 방법이 있을 수 있다. 확률적 방법이란 한 속성이 어떤 질의에 포함될 것인지에 대한 결정은 확률 변수값을 생성하여 생성된 값이 주어진 확률범위에 포함되는지에 따라 이루어지게 하는 의미로서, 일반적으로 한 속성이 어떤 질의에 이용될 조건 확률 값이 클수록 확률적으로 그 가능성도 높겠지만 예외적인 확률 변수 값이 생성될 수도 있기 때문에, 전체적으로는 주어진 범위 조건을 따르면서도 다양한 형태의 분포가 만들어질 수도 있을 것이다. 본 연구에서는 속성과 질의의 주어진 이용 범위에 가급적 만족시키는 것을 목표로 하여, 첫 번째 방법을 시도하였다.

제안된 알고리즘은 기본 알고리즘과 보정 알고리즘으로 구성되며, 보정 알고리즘은 다시 속성보정 알고리즘과 질의보정 알고리즘으로 구성된다((그림 2) 참조). 각 질의 q_j에 대하여 q_j에 포함될 속성 개수의 범위로서 lower_limit[q_j]와 upper_limit[q_j]가 주어진다. 마찬가지로 각 속성 a_i에 대하여 a_i가 이용될 질의 수의 범위로서 lower_limit[a_i]와 upper_limit[a_i]가 주어진다. 기본 알고리즘을 이용하여, 주어진 스키마 정보와 분포 조건으로부터 각 질의에 이용될 속성들을 선택한다. 그 결과로 생성된 속성수 m, 질의 수 n에 대한 $m \times n$의 2차원 비트맵은 각 속성과 질의들이 주어진 분포조건에 모자라거나 초과하는 분포를 가질 수 있다. 이를 보정하기 위하여 속성보정 알고리즘과 질의 보정 알고리즘을 실행한다. 속성 보전 알고리즘과 질의 보정 알고리즘의 실행 순서는 무관하다.



(그림 2) 질의-속성 분포 생성절차

4.2 기본 알고리즘

기본 알고리즘은 다음과 같이 질의-속성 분

```
INPUT : lower_limit[a_i], upper_limit[a_i], lower_limit[q_j], upper_limit[q_j]
        Schema Information (Relation[a_i], P_Key[a_i], F_Key[a_i])
line 1   for each q_j in Q do
line 2       for each a_i in A do
line 3           if (count[q_j] < upper_limit[q_j] && count[a_i]< upper_limit[a_i]) {
line 4               c_ij = 1 ;
line 5               count[a_i ] ++ ;
line 6               count[q_j ] ++ ;
line 7               if (F_Key[a_i ] equals to a_p for some pi) { /* a_i is a foreign key */
line 8                   c_pj = 1 ;
line 9                   count[a_P] ++ ;
line 10                  count[q_j] ++ ;
line 11              }
line 12          } else
line 13              c_ij = 0 ;
line 14      done
line 15  done
```

(Algorithm 1) 기본 알고리즘

포를 기본적으로 지정한다. 먼저 각 질의들에 대하여 그 질의에 포함될 속성들을 임의의 순서로 하나씩 고려한다. count[a_i]와 count[q_j]는 각각 속성 a_i가 이용된 횟수와 질의 q_j에 포함된 속성의 개수를 담고 있는 배열 변수이므로, 속성 a_i가 질의 q_j에 이용되기 위해서는, 즉, 2차원 행열의 셀 c_{ij}의 값이 1로 설정되려면, count 값이 질의와 속성의 upper_limit 이내여야 할 것이다. 그렇지 않을 경우, 즉, 질의 q_j 또는 속성 a_i의 어느 한 쪽이라도 upper_limit에 이른 경우에는 셀 c_{ij}의 값을 0으로 설정한다. c_{ij}의 값이 1로 설정되면 q_j와 a_i의 count 값을 1씩 증가 시킨다.

만약, a_i가 외래키 속성인 경우에는 그에 해당되는 주키도 포함시킨다. 이는 대부분 질의들에서 일반적으로 관찰되는 유형으로서, 외래키를 통한 조인 질의를 시사한다. 그러나 제안되는 알고리즘에서는 질의 및 속성 분포의 보정과정을 통하여 외래키가 포함되지 않는 다양한 질의 형태도 생성될 수 있다.

다음에 의사코드(pseudo code) 형태의 기본 알고리즘이 나타나있다. 알고리즘에 이용된 기호는 다음과 같은 의미를 갖는다. Q는 질의 집합이고, A는 속성 집합이며, q_j와 a_i는 각 개별 질의와 속성을 의미한다. count[ ], lower_limit[ ], upper_limit[ ]의 의미는 3.2절에서 설명한 바와 같다.

알고리즘은 매우 단순하며, 이상적인 경우에 질의 분포와 속성 이용도 분포를 모두 만족시킬 수 있지만, 그렇지 못할 경우가 발생할 가능성이 매우 높다. 알고리즘은 질의 별로 주어진 upper_limit 만큼의 속성을 포함시키고 있다. 따라서, 모든 질의에 대한 속성 선택이 만족되었다고 하더라도 어떤 속성의 경우 이용도 분포의 lower_limit를 만족하지 못할 경우가 있을 수 있다. 질의에 대하여도 마찬가지로 주어진 lower_limit을 만족하지 못하는 질의들이 존재할 수 있다. 또한, 외래키에 대응되는 주 키를 포함시킴으로써 질의와 속성의 upper_limit을 초과하는 경우도 발생 가능하다. 이를 보완하기 위하여 질의 및 속성 분포에 대한 보정 작업이 필요하다.

[예제 1]

예로서 4개의 속성과 3개의 질의에 대한 이용 범위가 다음과 같다고 하자. 편의상 속성들은 모두 주 키 또는 외래키가 아닌 일반 속성으로 가정한다.

a1[1 : 2], a2[0 : 1], a3[1 : 3], a4[2 : 3]
q1[1 : 3], q2[1 : 2], q3[3 : 4]

이상의 범위 조건에 대하여 기본 알고리즘을 적용한 결과는 아래 <표 1>과 같다.

〈표 1〉 예제 1 : 기본 알고리즘 실행결과

	q1	q2	*q3*	합
a1	1	1	0	2
a2	1	0	0	1
a3	1	1	1	3
a4	0	0	1	1
합	3	2	2	

위 표에서 '1'은 해당 행의 속성이 해당 열의 질의에 포함되는 것을 의미하고 '0'은 포함되지 않는 것을 의미한다. 그림에서 속성 a1, a2, a3와 질의 q1, q2는 각각 최대수 만큼 이용되고 있고 기운 글씨체로 인쇄된 a4과 q3는 최저수에 미달되고 있음을 보이고 있다. a3와 q4는 가능하면 보정 알고리즘을 통하여 보정되어야 할 것이다.

4.3 속성 보정 알고리즘

속성 보정 작업은 각 속성의 분포 범위를 초과한 속성들과 분포범위에 미달되는 속성들에 대하여 분포 범위에 최대한 만족될 수 있도록 하는 것이다. 기본 알고리즘의 실행을 통한 질의-속성 이용을 나타내는 2차원 비트맵 c_{ij}, 각 속성에 지정된 질의수의 분포와 현재 설정 갯수, 각 질의에 지정된 속성수의 분포와 현재 포함된 속성 개수에 대한 값이 이용된다. 알고리즘은 먼저 질의 개수 범위를 초과하는 질의들에 대한 보정을 시도한다. 여기에는 두 가지 방법이 가능하다. 첫째는 이러한 속성 a_i에 할당된 질의 가운데 count[q_j] > lower_limit[q_j]인 임의의 질의 q_j, 즉, 최저 개수보다 큰 속성수를 갖는 질의를 선택하여, 질의 q_j를 속성 a_i의 이용 목록에서 제외시킨다. 즉, c_{ij} = 0로 하고, count[a_i]와 count[q_j]를 1씩 감소시킨다.

두 번째 방법은 이러한 q_j가 존재하지 않는 경우에 적용된다. 즉, a_i를 이용하는 모든 질의들이 모두 최저 분포수 또는 그 이하의 속성수를 갖는 경우에 위의 첫 번째 방법은 적용될 수 없다. 이 경우에는 a_i와 다른 속성 중에서, count[$a_{i'}$]가 upper_limit[$a_{i'}$]보다 적고, c_{ij} == 1이면서 $c_{i'j}$ == 0인 속성 $a_{i'}$와 q_j를 임의 선택하여, c_{ij} = 0로 하고, $c_{i'j}$ = 1로 바꿈으로써, count[a_i]를 1 감소시킬 수 있다. 이때 count[$a_{i'}$]는 1 증가하지만 여전히 upper_limit[$a_{i'}$]를 만족하며, count[q_j]의 수는 a_i에 의하여 1증가하고 $a_{i'}$에 의하여 1감소하여 변함이 없으므로 q_j의 속성 이용 분포를 만족한다는 점이 중요하다.

만약 이와 같은 $a_{i'}$와 q_j를 선택할 수 없는 경우에는 주어진 질의-속성 분포 조건하에서는 속성의 초과하는 질의 수를 더 이상 보정할 수 없다는 의미가 된다. 이러한 경우는 질의-속성 분포조건이 부적절하게 지정되었기 때문에 발생한다고 볼 수 있다. 설령 이러한 경우가 발생한다고 하더라도, 기본 알고리즘에서 한 속성에 대한 이용 질의 수의 범위 초과 개수는 최대한 1로 제한되는데 그 이유는 다음과 같다. 어떤 질의 q_j가 속성수 범위를 초과하는 경우는 q_j에 어떤 테이블의 외래키 a_i가 포함될 때 그에 대응되는 다른 테이블의 주 키 a_p를 추가할 때 발생할 수 있다. 이 때 count[q_j]를 1 증가시킴으

```
INPUT : 2-dimensional array c_ij, count[a_i], count[q_j], lower_limit[a_i], upper_limit[a_i],
        lower_limit[q_j], upper_limit[q_j]
        Schema Information (Relation[a_i], P_Key[a_i], F_Key[a_i])
line 1  while (there exists a_i where count[a_i] > upper_limit[a_i]) {
line 2     if (there exists q_j where c_ij == 1 && count[q_j] > lower_limit[q_j]) {
line 3           c_ij = 0 ; count[a_i] -- ; count[q_j]-- ;
line 4     } else if (there exist a_i' and q_j where c_i'j == 0 && c_ij == 1 &&
line 5                                          count[a_i' ] < upper_limit[a_i']) {
line 6           c_ij = 0 ; c_i'j = 1 ;
line 7           count [a_i]-- ; count[a_i' ]++ ; /* count[q_j] is not changed */
line 8     } else
line 9           break ;
line 10 }
line 11 while (there exists a_i where count[a_i] < lower_limit[a_i]) {
line 12    if (there exists q_j where c_ij == 0 && count[q_j ] < upper_limit[q_j ]) {
line 13          c_ij = 1 ; count[a_i ]++ ; count[q_j]++ ;
line 14    } else if (there exist a_i' and qj where c_i'j == 1 && c_ij == 0 &&
line 15                                          count[a_i' ] > lower_limit[a_i' ]) {
line 16          c_ij = 1 ; c_i'j = 0 ;
line 17          count [a_i ]++ ; count[a_i' ]-- ; /* count[q_j] is not changed */
line 18    } else
line 19          break ;
line 20 }
```

(Algorithm 2.1) 속성 보정 알고리즘

로써 이용 수의 한도를 초과하게 되면 그 다음의 반복문에서 더 이상 속성을 추가하지 않기 때문이다. 따라서, 전체적으로는 주어진 질의-속성 분포 수에 최대한 근사하게 설정된다는 점이 제안된 알고리즘의 특성이라고 하겠다.

다음은 속성의 주어진 최저 질의 이용 개수에 미달하는 속성에 대한 보정이 필요한데, 이는 초과하는 경우의 처리 방식과 유사하게 진행된다. 최저 질의 이용 개수에 미달되는 질의 a_i에 포함되지 않은 질의 가운데 질의의 속성 이용 upper_limit보다 적은 질의 q_j를 선택하여, $c_{i,j}$ = 1로 설정하고, a_i와 q_j의 이용 count를 1씩 증가시킨다. 만약 속성 a_i에 이용되지 않은 질의들이 모두 속성이용 분포의 상한값의 이용 개수를 가질 경우에는, a_i와 다른 속성 가운데, 질의 이용 분포의 하한값을 초과한 이용 개수를 갖고, $c_{i,j}$ == 0이며 $c_{i'j}$ == 1인 속성 $a_{i'}$와 질의 q_j를 선택한다. 그리고 c_{ij} = 1, $c_{i'j}$ = 0로 하며 a_i와 $a_{i'}$의 count값을 각각 1증가 및 1 감소시켜 보정하며 이 과정을 되풀이한다. 이용 질의 수의 하한값에 미달되는 속성 a_i에 대하여 a_i에 이용되지 않는 질의들이 모두 이용 속성수의 상한값 또는 그 이상을 갖고, a_i 이외의 다른 속성들이 모두 이용 질의수의 하한값 또는 그 이하를 가질 경우에는 더 이상 속성 a_i에 이용 질의 수를 추가할 수 없게 된다. 만약 이러한 속성들의 수가 지나치게 많다면, 문제의 입력으로 주어지는 질의-속성 이용 분포조건을 재설정하여 기본 알고리즘부터 다시 실행할 수도 있을 것이다.

[예제 2]

앞선 [예제1]의 결과에 대하여 속성 보정 알고리즘을 실행한 결과는 <표 2>와 같다. 즉, [예제 1]의 <표 1>에서 이용범위에 미달된 속

성 a4를 보정한 것으로서, <표 1>의 질의 q1과 q2가 이미 최대수 만큼의 속성을 이용하기 때문에 단순히 셀 c41 또는 c42를 '1'로 만들 수는 없다. 대신 a1이 최저 개수보다 큰이용 수를 갖기 때문에 셀 c11을 '0'에서 '1'로 하고, 미달된 속성 a4에 대한 셀 c41을 '0'에서 '1'로 바꿈으로써 a4가 이용범위 조건에 맞도록 보정된 결과를 위 위 표에서 확인할 수 있다(기운 글씨체 참조). 질의 q4는 여전히 보정이 필요하다.

〈표 2〉 예제 2 : 속성 보정 알고리즘 실행 결과

	q1	q2	q3	합
a1	0	1	0	*1*
a2	1	0	0	1
a3	1	1	1	3
a4	*1*	0	1	*2*
합	3	2	2	

4.4 질의 보정 알고리즘

속성보정 알고리즘을 실행하면서 일부 질의에 대한 분포도 함께 보정 될 수도 있으나, 그렇지 못한 경우도 있을 수 있기 때문에 질의에 대한 보정 작업도 반드시 필요하다. 예를 들면, 모든 속성들이 분포범위를 만족한다고 할 때 질의 보정은 특별한 작업이 없을 수 있지만, 질의의 속성 이용 lower_limit에 미달되거나 upper_limit를 초과하는 질의들이 여전히 존재할 수 있다. 질의 보정 알고리즘은 위의 속성보정 알고리즘과 대칭적이다.

먼저, 외래키 선택시 해당 주 키의 추가 선택으로 인하여 upper_limit을 초과하는 질의들에 대하여 보정한다. 질의 q_j가 최대 속성수를 초과할 경우에 q_j에 선택된 속성 가운데 lower_limit을 초과하는 속성 a_i를 임의 선택하여 c_{ij}를 0으로 하고, a_i와 q_j의 count를 각각 1 감소시킨다. 만약 이러한 a_i가 존재하지 않을 경우에는 q_j에 선택된 임의의 a_i에 대하여 (즉, c_{ij} == 1) a_i를 포함하지 않고 a_i를 추가로 포함하여도 upper_ limit를 초과하지 않는 질의 $q_{j'}$ (즉, $c_{ij'}$ == 0)를 선택하여 c_{ij}는 0으로, $c_{ij'}$는 1로 하고, count[q_j]는 1감소, count[$q_{j'}$]는 1증가시킨다. 이때 a_i의 count는 1증가되고 또 1감소되었으므로 변함이 없다.

질의의 lower_limit에 미달하는 질의들에 대한 보정작업도 유사하게 수행된다. q_j의 count가 lower_limit에 미달될 때, q_j에 속하지 않은 속성들 가운데 count 값이 upper_limit이하인 속성 a_i를 임의 선택하여, c_{ij} = 1로 하고, count[a_i]와 count[q_j]를 1씩 증가시킨다. 이러한 a_i가 더 이상 없을 경우에는 앞서 설명한 방법과 유사한 방법으로 다른 질의 q_j로부터 임의의 속성 $a_{i'}$를 빌어옴으로써 1만큼 보충하게 된다. 더 이상의 보정이 불가능하면 알고리즘은 종료하게 된다.

[예제 3]

앞선 [예제 2]의 결과에서 대하여 속성 보정 알고리즘을 실행한 결과는 아래와 같다.

〈표 3〉 예제 3 : 질의 보정 알고리즘 실행 결과

	q1	q2	*q3*	합
a1	0	1	*1*	*2*
a2	1	0	0	1
a3	1	1	1	3
a4	1	0	1	2
합	3	2	*3*	

[예제 2]의 <표 2>에서 질의 q3를 보정하기 위하여 속성 a1이 최대 이용 개수 미만이면서 셀 c13이 '0' 이므로 c13를 단순히 '0'에서 '1'로 바꿈으로써 q3의 이용수가 범위 조건에 만족시킬수 있게 되었다(기운 글씨체 참조).

```
INPUT : 2-dimensional array c_ij, count[a_i], count[q_j],
        lower_limit[a_i], upper_limit[a_i], lower_limit[q_j], upper_limit[q_j]
        Schema Information (Relation[a_i], P_Key[a_i], F_Key[a_i])
BEGIN
line 1   while (there exists q_j where count[q_j] > upper_limit[q_j]) {
line 2      if (there exists a_i where c_ij == 1 && count[a_i] > lower_limit[a_i]) {
line 3            c_ij = 0 ;  count[a_i] -- ; count[q_j]-- ;
line 4      } else if (there exist q_j' and a_i where c_ij' == 0 && c_ij == 1 &&
line 5                                          count[q_j' ] < upper_limit[q_j']) {
line 6            c_ij = 0 ; c_ij' = 1 ;
line 7            count [q_j]-- ; count[qj']++ ; /* count[a_i] is not changed */
line 8      } else
line 9            break ;
line 10  }
line 11  while (there exists q_j where count[q_j] < lower_limit[q_j]) {
line 12     if (there exists a_i where c_ij == 0 && count[a_i] < upper_limit[a_i]) {
line 13           c_ij = 1 ;  count[a_i]++ ; count[q_j]++ ;
line 14     } else if (there exist qj' and ai where c_ij' == 1 && c_ij == 0 &&
line 15                                          count[q_j'] > lower_limit[q_j' ]) {
line 16           c_ij = 1 ; c_ij' = 0 ;
line 17           count [q_j]++ ; count[q_j']-- ; /* count[a_i] is not changed */
line 18     } else
line 19           break ;
line 20  }
END
```

(Algorithm 2.2) 질의 보정 알고리즘

4.5 구현 고려 사항

앞서 제안된 알고리즘의 구현 시에 몇 가지 고려할 사항이 있다. 먼저, 기본 알고리즘이 선택할 질의와 속성의 순서이다. 질의의 순서는 질의간의 분포에 대한 상호 관련성이 없기 때문에 입력 데이터로 주어진 순서, 또는, 임의의 순서대로 실행되어도 알고리즘의 실행 결과에 영향을 미치지 않을 것이다. 그러나, 속성들은 특정 테이블에 소속되어 있는 관련성이 있으며, 속성의 고려 순서는 먼저 고려되는 속성일수록 최저 분포수를 만족할 확률이 높고, 뒤에 고려되는 속성일수록 최저 분포 수를 만족하지 못할 가능성이 높다. 특히, 질의의 upper_limit가 대부분 낮게 주어질 경우에 더욱 그럴 것이다. 물론 이런 경우 보정 알고리즘을 통하여 보정되겠지만, 특정 테이블의 속성들만이 질의 수 범위를 만족하지 못하는 경우를 피하기 위해서는 속성들을 임의의 순서로 고려하는 것이 바람직할 것이다. 따라서, 본 연구에서는 속성과 질의목록을 입력 데이터 순서에 따라 배열로 표현하되 배열의 순서를 임의로 재 정렬할 수 있도록 구현함으로써 이와 같은 효과를 줄 수 있도록 하였다.

다음은 보정 알고리즘에서, 어떤 질의에 선택되었다가 보정절차에 의하여 배제시킬 속성을 선택하는데 대한 것이다. 위의 기본 알고리즘 경우에 논의한 바에 의하여, 고려되는 속성의 순서는 임의로 고려될 수 있으므로 임의 선택될 수 있다. 선택된 속성은 테이블의 주 키, 외래키

또는 일반 속성일 것이다. 질의를 보다 효과적으로 작성하는 분포를 생성하는 것이 본 알고리즘의 목적이므로, 본 연구에서는 주 키 속성이 배제 대상일 경우에는 다른 속성을 고려하는 것으로 보정 알고리즘을 구현하였다.

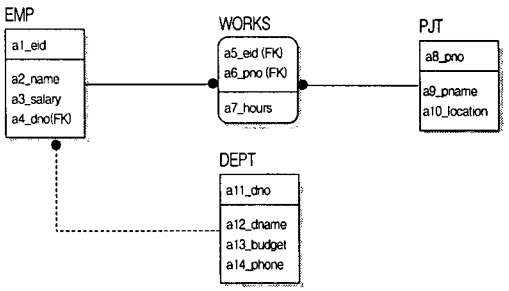
5. 실 험

제안된 알고리즘은 SunOS 에서 GNU C/C++ Version2.8.1로 구현하였다. 본 절에서는 예제 입력 데이터에 대한 기본 알고리즘과 보정 알고리즘을 수행한 결과를 보이고, 생성된 결과인 질의-속성 이용분포를 이용하여 SQL 문을 작성하는 예를 보인다.

5.1 입력 데이터

실험에 이용된 데이터베이스 스키마에 대한 ER(Entity-Relationship) 다이어그램과 속성 이용도 분포 조건이 아래 (그림 3)과 <표 4>에 각각 나타나있다. 4개의 테이블과 모두 14개의 속성으로 구성되어 있으며, 각 속성의 이용 질의수에 대한 분포가 표에서 하한값 (lower_limit)과 상한값 (upper_limit)으로 주어져 있다.

표에서 속성 이름은 편의상 앞부분만 표기하였다. 속성의 질의수에 대한 분포는 High 4개, Middle 5개, Low 5개로 임의 구성하였다. TPC-H의 질의-속성 분포에서는 주 키 또는 외래키의 경우 High 분포에 주로 해당되고, 일반 키는 대부분 Low에 해당되었다. 물론 이에 관계없이



(그림 3) 예제 데이터베이스의 ER Diagram

임의로 지정하여도 알고리즘에는 영향을 미치지 않는다. 각 나타나있다. 4개의 테이블과 모두 14개의 속성으로 구성되어 있으며, 각 속성의 이용 질의수에 대한 분포가 표에서 하한값 (lower_limit)과 상한값(upper_limit)으로 주어져 있다. 표에서 속성 이름은 편의상 앞부분만 표기하였다. 속성의 질의수에 대한 분포는 High 4개, Middle 5개, Low 5개로 임의 구성하였다. TPC-H의 질의-속성 분포에서는 주 키 또는 외래키의 경우 High 분포에 주로 해당되고, 일반 키는 대부분 Low에 해당되었다. 물론 이에 관계없이 임의로 지정하여도 알고리즘에는 영향을 미치지 않는다.

실험의 질의의 속성수 분포는 <표 5>와 같다. 통상 질의의 속성수 분포 조건은 정규분포, 즉, 일부 질의들이 이용 속성수가 매우 적거나 매우 많고, 대부분 질의는 중간 형태로 보았다. 이 분포 또한 임의로 지정하여 실험할 수도 있으나, 본 실험에서는 정규분포와 유사하도록 Low 1개, Low-Mid 2개, Middle 4개, Mid-

〈표 4〉 예제 속성 이용도 분포 조건

데이블	EMP				WORKS			PJT			DEPT			
속 성	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	a13	a14
하한값	6	4	0	5	7	5	4	6	2	0	5	2	1	2
상한값	8	6	2	6	6	7	6	9	4	3	9	6	4	3
분 포	H	M	L	M	H	M	M	H	L	L	H	M	L	L

〈표 5〉 예제 질의 복잡도 분포 조건

질 의	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_{10}
하한값	5	3	7	1	5	9	7	5	3	5
상한값	7	5	9	3	7	12	9	7	5	7
분 포	M	L-M	M-H	L	M	H	M-H	M	L-M	M

High 2개, High 1개의 질의들로 구성된 분포로 설정하였다.

5.2 알고리즘 실행 결과

(그림 4)는 알고리즘의 수행 결과를 보여준다. 좌측 그림은 기본 알고리즘만 수행한 결과이고 우측 그림은 보정 알고리즘을 추가로 수행한 결과이다. 이해의 편의상 각 속성과 질의 보정에서 속성 또는 질의의 번호 순서대로 고려하도록 실험하였다. 각 표에서 제일 오른편 열은 각 행의 1의 수, 즉, count[a_i]를 의미한다. 또, 제일 마지막 행은 각 열의 1의 수의 합계로서, count[q_j]를 의미한다.

(그림 4) 좌측 그림에서는 질의의 분포 값이 대부분 만족하고 있다. 즉, 질의 q_1, q_4와 속성 a_{14}를 제외한 각 질의와 속성은 lower_limit와 upper_limit 개수 이내의 속성수 및 질의수를 포함하고 있다. 질의 q_1과 q_4는 최대 속성수 (upper_limit)를 초과하고 있으며, 속성 a_{14}는 최저 질의수 (lower_limit)에 미달하고 있다. 보정 알고리즘을 수행한 결과가 우측에 나타나 있으며, 변경된 c_{ij} 및 count값이 기운 글씨체로 표시되어 있다. 보정과정을 거친 결과 질의 q_1과 q_4, 속성 a_2, a_3, a_4, a_{11}, a_{14}의 count값이 조정되었으며, 모두 주어진 질의-속성 분포범위를 만족하고 있음을 알 수 있다. 한가지 유의할 점

속성	질의 (q_1 ~q_{10})										count[a_i]
a_1	1	1	1	1	1	1	1	1	0	0	8
a_2	1	1	1	1	1	1	0	0	0	0	6
a_3	1	1	0	0	0	0	0	0	0	0	2
a_4	1	1	1	1	1	1	0	0	0	0	6
a_5	1	0	1	0	1	1	1	1	1	1	8
a_6	1	0	1	0	1	1	1	1	1	0	7
a_7	0	0	1	0	0	1	1	1	1	1	6
a_8	1	0	1	0	1	1	1	1	1	1	8
a_9	0	0	1	0	0	1	1	1	0	0	4
a_{10}	0	0	0	0	0	1	1	1	0	0	3
a_{11}	1	1	1	1	1	1	0	0	1	1	8
a_{12}	0	0	0	0	0	1	1	0	0	1	3
a_{13}	0	0	0	0	0	1	1	0	0	1	3
a_{14}	0	0	0	0	0	0	0	0	0	1	*1*
	8	5	9	*4*	7	12	9	7	5	7	

〈기본 알고리즘만 실행한 결과〉

속성	질의 (q_1 ~q_{10})										count[a_i]
a_1	1	1	1	1	1	1	1	1	0	0	8
a_2	*0*	1	1	1	1	1	0	0	0	0	*5*
a_3	*0*	1	0	0	0	0	0	0	0	0	*1*
a_4	1	1	1	*0*	1	1	0	0	0	0	*5*
a_5	1	0	1	0	1	1	1	1	1	1	8
a_6	1	0	1	0	1	1	1	1	1	0	7
a_7	0	0	1	0	0	1	1	1	1	1	6
a_8	1	0	1	0	1	1	1	1	1	1	8
a_9	0	0	1	0	0	1	1	1	0	0	4
a_{10}	0	0	0	0	0	1	1	1	0	0	3
a_{11}	1	1	1	1	*0*	1	0	0	1	1	*7*
a_{12}	0	0	0	0	0	1	1	0	0	1	3
a_{13}	0	0	0	0	0	1	1	0	0	1	3
a_{14}	*1*	0	0	0	0	0	0	0	0	1	*2*
	7	5	9	*2*	7	12	9	7	5	7	

〈보정 알고리즘을 추가로 수행한 결과〉

(그림 4) 알고리즘 실행 결과 예시

은 보정 알고리즘을 거친다고 하더라도, 모든 경우에 주어진 분포를 항상 만족시킬 수 있는 것은 아니라는 점이다. 극단적으로, 질의는 범위분포가 낮게 주어져 있고, 속성들은 모두 범위분포가 높게 주어진 경우, 이들 분포조건을 만족시키는 질의-속성 분포는 존재하지 않을 수도 있다. 따라서, 제안된 알고리즘은 질의-속성 수의 분포 조건을 다양하게 입력하고 성능평가에 적합한 질의-속성 분포를 쉽게 생성해 내는데 이용될 수 있을 것이다.

5.3 질의 작성 예

생성된 질의-속성 분포를 참고하여 다양한 방법으로 질의를 작성할 수 있을 것이다. 동일한 속성수를 이용하여 작성할 수 있는 질의문의 수는 무수히 많을 것이다. 이러한 질의는 각각 그 복잡도와 질의형태에 따라 다양하게 구성될 수 있으며, 질의와 속성간의 주어진 이용분포를 (최대한) 따른다는 점에서 벤치마킹, 색인 선택, 구체화 뷰의 선택 알고리즘 등의 성능평가 작업의 작업부하로 훌륭히 이용될 수 있을 것이다. 간단한 SPJ(Selection-Projection-Join) 형태의 질의만을 고려한다면 이와 같은 분포 데이터와 스키마 정보를 이용하여 질의를 자동 생성하는 것도 가능할 것이다. 물론 이를 위해서는 속성의 도메인 정보 등도 추가로 필요할 것이다. 본 절에서는 위와 같이 생성된 질의집합 모형 가운데 q_1과 q_5에 대한 SQL 작성 예를 보여 준다. 나머지도 유사하게 작성할 수 있을 것이다. 아래의 예에서 비트맵에 '1'로서 포함된 속성들이 WHERE절의 조건식에 이용되고 있다. 참고로 SELECT절에 포함되는 속성들은 질의-속성 분포와 무관하다.

- 질의 Q_1.

  [속성이용 비트맵 (a_1~a_{14}) :
  1 0 0 1 1 1 0 1 0 0 1 0 0 1]

```
SELECT  a2_name, a9_pname, a10_location
FROM    EMP, DEPT, WORKS, PJT
WHERE   a4_dno = a11_dno
  AND   a1_eid = a5_eid
  AND   a6_pno = a8_pno
  AND   a14_phone LIKE '%5430'
```

- 질의 Q_5.

  [속성이용 비트맵 (a_1~a_{14}) :
  1 1 0 1 1 1 0 1 0 0 0 0 0 0]

```
SELECT  a9_pname, sum(a7_hours)
FROM    EMP, WORKS, PJT
WHERE   a4_dno = 12345
  AND   a1_eid = a5_eid
  AND   a6_pno = a8_pno
  AND   a2_name LIKE '%Smith'
GROUP BY   a9_pname
```

6. 결 론

데이터베이스 알고리즘의 성능평가 및 벤치마킹 작업에는 다양한 작업부하를 손쉽게 생성하는 것이 매우 필요하다. 이들 작업부하는 각 질의에 이용되는 속성 수와 그리고 각 속성이 질의에 이용되는 수가 다양한 분포형태로 요구될 수 있다. 주어진 질의-속성 수의 분포를 따르는 질의집합 모형을 수작업으로 구성하기에는 질의수와 속성수가 많아질수록 매우 힘든 작업이 아닐 수 없으며, 또한 만족스러운 분포를 생성하기란 쉬운 작업이 아니다.

본 연구에서는 데이터베이스 분야의 성능평가, 벤치마킹 등에 이용될 수 있는 질의 작업부하의 효과적인 작성을 지원하기 위하여, 주어진 질의-속성 분포를 최대한 만족하는 질의집합 모형을 생성하는 알고리즘을 제안하였다. 스키

마 정보(테이블, 속성, 주키, 외래키), 속성의 이용도 분포, 질의 수, 질의의 복잡도 분포 값이 주어졌을 때, 제안된 알고리즘은 질의와 속성간에 이용 관계를 2차원 비트맵 배열로 추상화시키고, 질의와 속성의 이용 개수에 대한 분포조건을 최대한 만족시키기 위하여, 각 질의 별로 속성 수를 최대한 할당한 후, 분포 범위를 만족시키지 못하는 질의와 속성들에 대하여 이를 보정하는 방식으로 고안되었다.

실험을 통하여 제안된 알고리즘이 주어진 분포조건을 만족시키는 질의 모형을 적절히 생성함을 보였으며, 그 결과를 이용하여 SQL 질의문을 작성한 예를 보였다. 대규모 질의 작업부하를 필요로 하는 경우에 본 연구의 알고리즘을 적절히 이용하여 주어진 질의-속성 이용 분포를 따르는 질의집합 모형을 생성함으로써, 효과적인 질의 작업부하 작성을 지원할 수 있을 것이다.

본 연구 결과로서 주어진 질의 및 속성의 분포를(최대한) 만족하는 질의모형으로부터 실제 SQL 질의를 생성하는 작업도 그리 쉬운 일은 아닐 것이다. 단순 SPJ(Select-Project-Join) 형태의 SQL 질의문을 대상으로 한다면, 본 연구 결과를 토대로 속성의 도메인 정보를 추가로 이용하여 SQL 질의 집합을 자동적으로 생성해 내는 연구도 가능할 것으로 보인다. 또한 본 논문에서 소개된 알고리즘과 질의-속성 이용분포 문제 모형에 대한 0-1 정수 계획법의 일반적 해법과의 효율성에 대한 비교 연구와 질의-속성 이용 개수에 대한 확률 값에 근거하여 질의집합 모형을 생성하는 확률적 방법의 알고리즘에 대한 연구도 의미 있을 것으로 생각된다.

참 고 문 헌

[1] Boehme, T. and E. Rahm, "XMach-1 : A Benchmark for XML Data Management," Proceedings of German Database Conference BTW2001, Berlin 2001 (also at http://dbs.uni-leipzig.de/en/projekte/XML/XmlBenchmarking.html).

[2] Chaudhuri, S., and V. Narasayya, "An Efficient, Cost-Driven Index Selection Tool for Microsoft SQL Server," Proceedings of Int'l Conference on Very Large Data Bases, 1997.

[3] Chaudhuri, S., and V. Narasayya, "AutoAdmin : What-if Index Analysis Utility," Proceedings of ACM SIGMOD Conference, 1998.

[4] Darmont, J. and M Schneider, "Benchmarking OODBS with a generic tool," Journal of Database Management, Vol.11, Issue 3, July, 2000.

[5] Devlin, B., *Data Warehouse : from Architecture to Implementation*, Addison Wesley, 1997.

[6] Elmasri, R and S.B. Navathe, *Fundamentals of Database Systems*, Addison-Wesley, 2000.

[7] Goldstein, J. and P. Larson, "Optimizing Queries using Matrialied View," ACM SIGMOD RECORD, 30(2), 2001.

[8] Gray, J. (ed.), *The Benchmark Handbook for Database and Transaction Processing Systems*, Morgan Kaufmann Publishers, Inc., 1993.

[9] Grinstein, G., P. Hoffman, and R. Pickett, "Benchmark Development for the Evaluation of Visualization for Data Mining," Information Visualization in Data Mining and Knowledge Discovery, August, 2001.

[10] Gupta, H., *et al*, "Index Selection for OLAP," Proceedings of ICDE, 1997.

[11] Gupta, H., "Selection of Views to Materialize in a Data Warehouse," Proceedings of Int'l Conference on Data Technology(ICDT), 1997.

[12] Hillier, F.S., and G.J. Lieberman, *Introduction to Operations Research* (5Ed.), McGraw-Hill, 1990.

[13] Horowitz, E. and S. Sahni, *Fundamentals of Computer Algorithms*, Computer Science Press, 1978.

[14] Labrinidis, A. and N. Roussopoulos, "Web View Materialization," Proceedings of ACM SIGMOD Conference, 2000.

[15] Ligoudistianos, S., D. Theodoratos, and T. Sellis, "Experimental Evaluation of Data Warehouse Configuration Algorithms," Proceedings of DEXA Conference, 1998.

[16] Jain, R., *The Art of Computer Systems Performance Analysis*, John Wiley & Sons, Inc., 1991.

[17] McDonell, K., "Benchmark frameworks and tools for modelling the workload profile," Performance Evaluation, Feb. 1995.

[18] Poess, M., and C. Floyd, "New TPC Benchmarks for Decision Support and Web Commerce," ACM SIGMOD Record, 29(4), December, 2000.

[19] TPC, TPC-H Benchmark Specification, Transaction Processing Performance Council, http://www.tpc.org/hspec.html, 1999.

[20] TPC, TPC Policies, http://www.tpc.org/information/about/documentation/spec/TPC_Policies_v5.2.pdf, Oct. 2001.

■ 저자소개

서 상 구

1984년에 서울대학교 컴퓨터공학과를 졸업하고 1986년 한국과학기술원 전산학과 석사를 졸업하며 1995년에 한국과학기술원 전산학과 박사학위를 취득하였고, 1999년부터 광운대학교 경영정보학과에 근무하고 하고 있으며 주요 관심분야는 데이터베이스 최적화, 웹 데이터베이스, 데이터웨어하우징 등이다.