중첩된 버킷을 사용하는 다차원 히스토그램에 대한 개선된 알고리즘

(An Improved Algorithm for Building Multi-dimensional Histograms with Overlapped Buckets)

문 진 영 † 심 규 석 ††
(Jin-young Moon) (Kyu-seok Shim)

요 약 히스토그램은 최근들어 많은 관심을 끌고 있다. 히스토그램은 주로 상용 데이타베이스 관리 시스템에서 질의 최적화를 위해 속성의 값에 대한 데이타 분포를 추정하는데 사용되었다. 최근에는 근사 질의와 스트림 데이타에 대한 연구 분야에서 히스토그램에 대한 관심이 커지고 있다. 관계형 데이타베이스에서 두 개 이상의 속성에 대한 결합 데이타 분포를 근사시키는 가장 간단한 방법은 각 속성의 데이타 분포가 결합 데이타 분포에 독립적이라고 가정하는 속성 값 독립(Attribute Value Independence: AVI) 가정하에서 각각의 속성에 대해서 히스토그램을 만드는 것이다. 그러나 실제 데이타에서 이 가정은 잘 맞지 않는다. 따라서 이 문제를 해결하기 위해서 웨이블릿, 랜덤 샘플링, 다차원 히스토그램과 같은 기법들이 제안되었다. 그 중에서 GENHIST는 실수형 속성에 대한 데이타 분포를 근사시키기 위해 고안된 다차원의 히스토그램이다. GENHIST는 데이타 분포를 좀 더 효과적으로 근사시키기 위해서 중첩되는 버킷을 사용한다.

본 논문에서는 SSE(Sum Squared Error)를 최소화시키는 중첩되는 버킷들의 최적 빈도를 결정하는 OPT 알고리즘을 제안한다. 처음에 GENHIST에 의해 중첩되는 버킷으로 구성되는 히스토그램을 만든 후에 OPT 알고리즘에 의해서 각 버킷의 빈도를 다시 계산해서 GENHIST를 개선시킬 수 있다. 실험 결과는 OPT 알고리즘이 GENHIST에 의해 만들어진 히스토그램의 정확도를 크게 개선시킴을 보여준다.

키워드 : 히스토그램

Abstract Histograms have been getting a lot of attention recently. Histograms are commonly utilized in commercial database systems to capture attribute value distributions for query optimization. Recently, in the advent of researches on approximate query answering and stream data, the interests in histograms are widely being spread. The simplest approach assumes that the attributes in relational tables are independent by AVI(Attribute Value Independence) assumption. However, this assumption is not generally valid for real-life datasets. To alleviate the problem of approximation on multi-dimensional data with multiple one-dimensional histograms, several techniques such as wavelet, random sampling and multi-dimensional histograms are proposed. Among them, GENHIST is a multi-dimensional histogram that is designed to approximate the data distribution with real attributes. It uses overlapping buckets that allow more efficient approximation on the data distribution.

In this paper, we propose a scheme, OPT that can determine the optimal frequencies of overlapped buckets that minimize the SSE(Sum Squared Error). A histogram with overlapping buckets is first generated by GENHIST and OPT can improve the histogram by calculating the optimal frequency for each bucket. Our experimental result confirms that our technique can improve the accuracy of histograms generated by GENHIST significantly.

Key words : Histograms

· 본 연구는 첨단정보기술연구센터를 통하여 한국과학재단의 지원을 받았음.
† 비 회 원 : 한국과학기술원 전산학과
jymoon@etri.re.kr
†† 종신회원 : 서울대학교 전기컴퓨터공학부 교수
shim@ee.snu.ac.kr
논문접수 : 2002년 1월 23일
심사완료 : 2003년 2월 12일

1. 서 론

히스토그램(histogram)[1]은 릴레이션(relation)에서 하나 또는 그 이상의 속성들에 대한 데이타 분포를 버킷(bucket)으로 나누고, 각 버킷의 영역과 빈도를 저장하는 근사 요약 정보이다.

처음에 히스토그램은 데이타베이스 관리 시스템 내부의 질의 최적화기에서 질의의 근사 결과를 구하기 위해서 사용되었다. 질의 최적화기는 주어진 질의에 대해서 여러 개의 질의 처리 전략을 세우고, 그 중에서 비용이 최소인 전략을 찾는다. 따라서 질의 최적화기는 실제 질의를 수행하는 것보다 훨씬 적은 노력으로 각 질의 전략의 비용을 추정할 수 있어야 한다. 이를 위해서 데이타베이스 시스템은 요약 정보인 히스토그램을 유지하여 실제 데이타베이스가 아니라 히스토그램에 대해 질의를 수행하고 질의 처리 비용을 추정한다. 그리고 최근에는 데이타웨어하우징(datawarehousing)에서도 히스토그램에 대한 관심이 높아지고 있다. 대규모 데이타베이스에 대한 분석을 수행하는 데이타웨어하우징에서 전체 데이타베이스에 대해서 정확한 질의 결과를 구하는 것은 비현실적이므로 추정으로 인한 오차를 어느 정도 감수하고라도 빠른 시간에 근사 질의 결과를 알기 위해서는 히스토그램의 사용이 필수적이다.

하나의 속성에 대한 1차원의 히스토그램에 대해서는 참고 문헌 [1] 에서 히스토그램을 체계적으로 분류하였다. 그런데 두 개 이상의 속성에 대한 결합 데이타 분포를 근사시키는 것은 고비용이 들므로 대부분의 상용 데이타베이스 시스템에서는 각 속성의 데이타 분포가 결합 데이타 분포에 독립적이라고 가정하는 **속성 값 독립 (Attribue Value Independence: AVI) 가정** 하에서 각각의 속성에 대해 히스토그램을 만들어 전체 결합 데이타 분포를 추정한다. 그러나 함수 종속성(functional dependency)에서도 알 수 있듯이, 실제 데이타에서 속성 값 독립 가정을 만족하는 경우는 드물어서 속성 값 독립 가정 하에 결합 데이타 분포를 근사시키면 부정확해질 수 있다. 따라서 각 속성간의 종속성을 잘 반영할 수 있는 다차원의 히스토그램에 대한 필요성이 대두되었다. [2] 와 [3] 에서는 다차원 히스토그램을 제안하였고, 데이타 변환 기술인 SVD(Singular Value Decomposition) [3], 웨이블릿(wavelet) [4], DCT(Discrete Cosine Transform) [5]를 사용한 기법들이 제안되었다.

[6]에서는 실수형 속성에 대한 다차원 데이타 분포를 근사시키는 GENHIST를 제안하였다. 기존의 알고리즘들이 정수형의 속성에 대해서 데이타 분포를 근사시키는 히스토그램을 제안하였지만, 실제로 공간(spatial) 데이타베이스나 멀티미디어(multimedia) 데이타베이스의 많은 속성들이 실수형의 값을 가진다. 따라서 [6]에서는 질의 자체를 실수형에 대해서 확장하고 실수형 속성의 데이타 집합에 대한 근사 질의 결과를 구하는 알고리즘을 제안하였다.

그리고 기존의 다차원 히스토그램들이 다차원 공간을 서로 떨어진 버킷으로 분할하는 것과는 대조적으로 버킷간의 중첩(overlap)을 허용하였는데, 중첩되는 버킷을 사용하면 실제 사용하는 버킷의 수보다 더 많은 빈도를 표현 할 수 있기 때문에 동일한 개수의 버킷을 사용하는 경우 데이타 분포를 보다 정확하게 근사 시킬 수 있다. 그러나 중첩을 허용하면서 문제의 복잡도가 증가하게 되서 최적인 히스토그램을 빠른 시간 안에 찾기가 불가능하므로 휴리스틱하게 히스토그램을 만들게 되어서 데이타 분포상에서 버킷을 효과적으로 분포시키더라도 버킷의 빈도를 잘 설정하지 못함으로 인해서 데이타 분포를 효과적으로 근사시키지 못할 수도 있다.

본 논문에서는 근사 오차 SSE(Sum Squared Error)를 최소화하도록 각 버킷의 최적 빈도를 결정하는 OPT 알고리즘을 제안하였다. 먼저 GENHIST에 의해서 중첩되는 버킷으로 구성되는 히스토그램을 생성한 다음 OPT 알고리즘을 통해서 각 버킷의 최적 빈도를 계산하여 히스토그램을 개선시킨 GENHIST-OPT를 제시하고 실험 및 평가하였다. 합성 데이타 집합 및 실생활 데이타 집합에 대해 수행한 실험 결과는 OPT 알고리즘이 GENHIST를 현저하게 개선시켰음을 보여준다.

본 논문의 구성은 다음과 같다. 2. 에서는 먼저 히스토그램이 근사시킬 데이타 분포와 히스토그램에 대한 기본 개념들을 설명한 다음, 기존의 히스토그램들 중에서 근사 오차를 최소화시키는 V-Optimal[1]과 다차원의 공간을 휴리스틱 방법으로 분할하는 MHIST[3]와 실수형 속성에 대해서 중첩을 허용하는 버킷을 사용하는 GENHIST[6]에 대해서 설명한다. 3. 에서는 주어진 버킷에 대해서 근사 오차를 최소화시키는 최적 평균 빈도를 결정하는 OPT 알고리즘을 제안한다. 그리고 4. 에서는 3. 에서 제안한 OPT 알고리즘을 GENHIST에 적용하여 GENHIST의 정확도를 개선시킨 GENHIST-OPT와 GENHIST를 실생활 데이타 집합과 합성 데이타 집합에 대해 실험한 결과를 기술한다. 마지막으로 5. 에서는 본 논문의 결론을 내린다.

2. 관련 연구

본 장에서는 우선 히스토그램이 근사시킬 데이타 분포와 히스토그램의 기본 개념을 설명한다. 그런 다음 이전에 제안된 히스토그램들 중에서 최적화의 개념을 처음으로 도입한 V-Optimal[1], 다차원에서의 공간 분할을 제안한 MHIST[3], 실수형 속성에 대해서 중첩을 허용하는 버킷을 사용하는 GENHIST[6]를 소개한다.

2.1 히스토그램의 기본 개념

2.1.1 데이타 분포

히스토그램은 숫자 속성(numerical attribute)에 대한 데이타 분포를 근사시킨 요약 정보이다. 숫자 속성 X_i의 데이타 분포에 대해서 알아본다. 속성 X_i의 범위(domain) P_i는 X_i가 가질 수 있는 모든 가능한 값(value)들의 집합이다. 값 집합 V_i ($\subseteq P_i$)는 릴레이션 R에서 실제로 나타나는 값들의 집합으로, 이 집합의 원소 $v_i(k)$와 $v_i(j)$는 $k \le j$ 이면 $v_i(k) \le v_i(j)$ 으로 순서가 매겨진다. 즉 값 집합 $V_i = \{ v_i(k) \mid 1 \le k \le D_i \}$, $D_i = |V_i|$ 이다. $v_i(k)$의 간격(spread) $s_i(k)$는 $s_i(k) = v_i(k+1) - v_i(k)$ ($1 \le k \le D_i$)로 정의된다. (단, $s_i(D_i) = 1$ 이라고 가정한다.) 그리고 값 $v_i(k)$의 빈도(frequency) $f_i(k)$는 릴레이션 R 내에서 속성 X_i가 $v_i(k)$를 값으로 가지는 튜플의 수이다. 또 $v_i(k)$의 영역(area) $a_i(k)$는 $v_i(k) \cdot f_i(k)$으로 정의된다. 값 $v_i(k)$의 누적 빈도(cumulative frequency)는 릴레이션 R 내에서 속성 X_i가 $v_i(k)$ 이하의 값을 가지는 튜플 수이다. 즉 누적 빈도 $c_i(k)$는 $\sum_{j=1}^{k} f_i(j)$가 된다. X_i의 데이타 분포는 값과 빈도 쌍들의 집합 $T_i = \{ (v_i(1), f_i(1)), (v_i(2), f_i(2)), \ldots, (v_i(D_i), f_i(D_i)) \}$ 이다. 유사하게 X_i의 누적 데이타 분포는 값과 누적 빈도 쌍들의 집합 $T_i^c = \{ (v_i(1), c_i(1)), (v_i(2), c_i(2)), \ldots, (v_i(D_i), c_i(D_i)) \}$ 이다. 또 X_i의 확장 누적 데이타 분포(extended cumulative data distribution) T_i^{c+}는 누적 데이타 분포 T_i^c에서 $P_i - V_i$에 모든 값에 0을 빈도로 할당 한 뒤 누적빈도를 계산함으로써 값 집합을 V_i에서 P_i로 확장시킨 데이타 분포이다.

지금까지 설명한 용어들을 정리하면 표 1과 같다.

속성 X_i에 대한 1차원의 데이타 분포를 속성 $X_1, \ldots, X_n$ 에 대한 n($\ge$ 2)차원의 데이타 분포로 확장시키면 다음과 같다. 값 $v_{1,\ldots,n}(k)$은 각 차원의 조합(combination)인 $\langle v_1(k_1), \ldots, v_n(k_n) \rangle$이 되고, 결합 빈도(joint frequency) $f_{1,\ldots,n}(k)$는 릴레이션 R에서 모든 i ($1 \le i \le n$)에 대해 속성 X_i의 값이 $v_i(k_i)$인 튜플들의 개수가 된다. 그리고 속성 $X_{1,\ldots,n}$에 대한 결합 데이타 분포 $T_{1,\ldots,n}$는 값의 조합 $v_{1,\ldots,n}(k)$과 결합 빈도 $f_{1,\ldots,n}(k)$의 쌍들의 집합이다. 그리고 각각의 속성에 대해서 데이타 분포를 투영시켜서 얻은 데이타 분포를 주변 분포(marginal distribution)라고 한다. 즉 속성 Xr ($1 \le r \le n$)에 대한 주변 분포 M_r의 값 $v_r(k_r)$은 데이타 분포 $T_{1,\ldots,n}$에서 속성 X_r의 값 $v_r(k_r)$ ($1 \le k_r \le D_r$)이 되고, 빈도 $f_r(k)$는 릴레이션 R 내에서 속성 X_r이 $v_r(k_r)$를 값으로 가지는 튜플의 수이다.

예를 들어, EMP(ename, salary)라는 스키마(schema)를 가지는 EMP 릴레이션에서 salary 속성의 값과 빈도, 그리고 간격을 구하면 표 2와 같다고 하자.

표 1 데이타 분포의 용어들

용 어	기 호	의 미
relation	R	릴레이션
attribute	X_i	릴레이션의 i번째 속성
domain	P_i	속성이 가질 수 있는 값의 범위
value set	V_i	$Vi = v_i(k) \mid 1 \le k \le D_i$, where $v_i(k) < v_i(j)$ when $k < j$
number of values	D_i	$D_i = \lvert V_i \rvert$
spread	$s_i(k)$	$s_i(k) = v_i(k+1) - v_i(k), 1 \le k \le D_i$ (단 $s_i(D_i) = 1$)
frequency	$f_i(k)$	$t.X_i = v_i(k)$인 튜플 $t(\in R)$의 개수
area	$a_i(k)$	$a_i(k) = v_i(k) \cdot f_i(k)$
data distribution	T_i	$T_i = (v_i(1), f_i(1)), (v_i(2), f_i(2)), \cdots, ((v_i(D_i), f_i(D_i))$
cumulative frequency	$c_i(k)$	$t.X_i \le v_i(k)$인 튜플 $t(\in R)$의 개수
cumulative data distribution	T_i^c	$T_i^c = (v_i(1), c_i(1)), (v_i(2), c_i(2)), \cdots, (v_i(D_i), c_i(D_i))$

표 2 EMP 릴레이션의 salary 속성의 값, 빈도, 간격

Value$\{v_i\}$	10	60	70	120	140	160
Frequency$\{f_i\}$	110	90	20	30	70	80
Spread s_i	50	10	50	20	20	1

이 때 그림 1은 표 2를 바탕으로 x축이 값, y축이 빈도를 나타내는 데이타 분포를, 그림 2-(a)는 x축이 값, y축이 누적 빈도를 나타내는 누적 데이타 분포를, 그림 2-(b)는 x축이 확장된 값, y축이 누적 빈도를 나타내는 확장 누적 데이타 분포를 나타낸다.

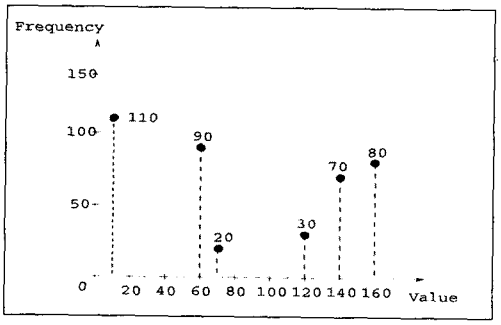
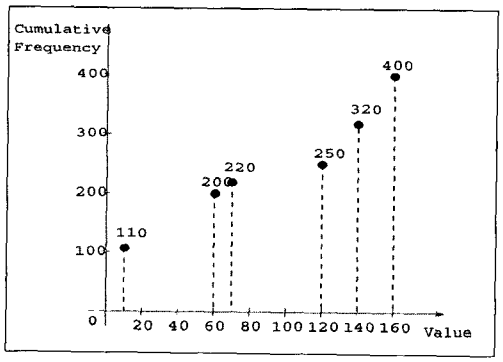
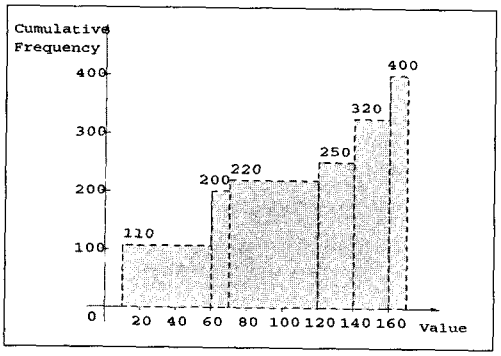


그림 1 EMP 릴레이션의 salary 속성에 대한 데이타 분포



(a) 누적 데이타 분포



(b) 확장 누적 데이타 분포

그림 2 EMP 릴레이션의 *salary* 속성에 대한 누적 데이타 분포들

2.1.2 히스토그램의 정의

속성 X_i에 대한 일차원의 히스토그램은 데이타 분포 T_i를 $\beta(\geq 1)$개의 버킷으로 분할(partition)하고 각 버킷의 빈도와 값을 근사시켜 만든다. 속성 X_1, ..., X_n에 대한 n차원의 히스토그램은 결합 데이타 분포 $T_{1,...,n}$에 대해서 일차원 히스토그램을 확장시킨 것이다.

하나의 버킷은 영역(range)과 빈도(frequency)로 구성된다. 버킷의 영역은 데이타 분포의 일부분(subset)으로 데이타 분포상의 범위를 나타낸다. 이 때 범위는 각 차원에 대해서 폐구간 [min, max]의 조합으로 나타낸다. 예를 들면, 2차원의 경우 $X_1[min_1, max_1]$ $X_2[min_2, max_2]$ (단, $v_1(1) \leq min_1 \leq max_1 \leq v_1(D_1)$, $v_2(1) \leq min_2 \leq max_2 \leq v_2(D_2)$, 로 나타낸다. 버킷의 빈도는 버킷 내의 총 빈도(total frequency)나 평균 빈도(average frequency)를 쓴다. 평균 빈도를 결정하는 것은 다음의 가정 중에서 어느 것을 따르느냐에 따라서 달라진다.

■ **연속 값 가정(Continuous Value Assumption)**

버킷 내의 값들은 버킷 영역 안의 전체 값들의 집합이다.

■ **균등 간격 가정(Uniform Spread Assumption)**

버킷 내의 값들은 동일한 간격로 떨어져 분포한다.

예를 들어, 버킷의 영역이 [1, 10]이고 이 영역에서 빈도의 합이 10일 때 연속 값 가정에 의하면 [1, 10] 의 모든 값 1, 2, ···, 10에 값이 존재하므로 이 버킷의 평균 빈도는 1이 되고, 균등 간격 가정에 의하면 별개의 값이 2개 존재한다고 할 때, 값 1과 10에 값이 존재하므로 이 버킷의 평균 빈도는 5가 된다. 그림 3는 각 가정에 의해서 버킷 내의 데이타 분포가 어떻게 근사시키는지를 보여준다.

앞에서 EMP 릴레이션의 salary 속성에 대한 데이타

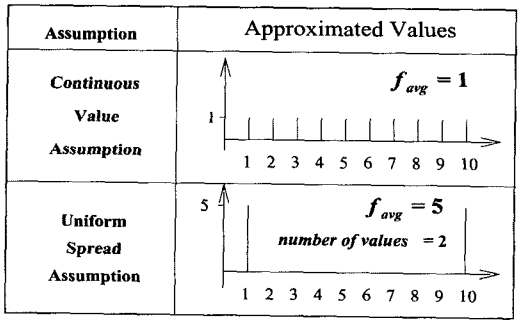


그림 3 버킷 내에서 값과 빈도의 근사

분포를 균등 간격 가정(uniform spread assumption)에 의해서 bucket1[10, 10], bucket2[60, 120], bucket3[140, 160]의 3개의 버킷으로 근사시키면 다음과 같다. bucket1에는 하나의 값과 빈도가 존재하므로 그 빈도 자체를 평균 빈도로 둔다. bucket2의 빈도는 값 60, 70, 120의 빈도 90, 20, 30의 평균 빈도인 46.67로 하고, 값은 [60, 120]에서 3개의 값이 균일 간격으로 있다고 가정하여 60, 90, 120에 평균 빈도 46.67이 존재한다고 추정한다. bucket3의 빈도는 값 140과 160의 빈도 70, 80의 평균 빈도인 75로 하고, 값은 [140, 160]에서 2개의 값이 균일 간격로 있다고 가정하여 140과 160에 평균 빈도 75가 존재한다고 추정한다. 그림 4에서 bucket1, bucket2, bucket3에 의해서 근사된 데이타 분포를 보여준다.

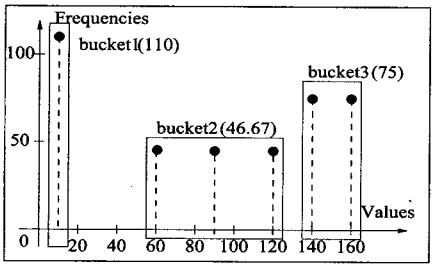


그림 4 *salary* 속성에 대한 히스토그램에 의해 근사된 데이타 분포

2.1.3 히스토그램의 정확도

히스토그램의 정확도는 실제 데이타 분포에 대한 질의 결과와 히스토그램에 의해 근사된 데이타 분포에 대한 질의 결과의 차인 오차(error)로 측정할 수 있다. S_i를 질의 q_i의 실제 질의 결과, S_i'을 히스토그램에 의해 근사된 질의 결과라고 할 때, 하나의 질의 q_i에 대한 대표적인 오차 측정 척도들은 다음과 같다.

표 3 하나의 질의에 대한 오차 측정 척도

절대 오차(Absolute Error)	상대 오차(Relative Error)
$e_i^{abs} = \lvert S_i - S_i' \rvert$	$e_i^{rel} = \frac{e^{abs}}{S_i} = \frac{\lvert S_i - S_i' \rvert}{S_i}$

따라서 크기가 |Q|인 질의 집합 Q에 대한 히스토그램의 평균 에러 E는 다음과 같다.

표 4 질의 집합에 대한 오차 측정 척도

평균 절대 오차 (Average Absolute Error)	평균 상대 오차 (Average Relative Error)
$E^{abs} = \frac{1}{\lvert Q \rvert} \cdot \sum_{i=1}^{\lvert Q \rvert} \lvert S_i - S_i' \rvert$	$E^{rel} = \frac{1}{\lvert Q \rvert} \sum_{i=1}^{\lvert Q \rvert} \frac{\lvert S_i - S_i' \rvert}{S_i}$

2.2 기존의 히스토그램들

2.2.1 V-Optimal

V-Optimal은 각 버킷에 의해서 근사된 빈도와 데이타 분포의 실제 빈도의 차를 제곱한 값들의 합이 최소화되는 히스토그램이다. V-Optimal(F,F)[7]는 근사 오차를 최소화시키기 위해서 데이타 분포를 빈도로 정렬하고 연속된 빈도를 같은 버킷에 둔다. 그리고 각 버킷에 나타나는 모든 값을 가지고 있다고 가정한다. 참고 문헌 [7]에서는 근사된 데이타 분포의 근사 오차(approximate error)와 최적 히스토그램(optimal histogram)을 다음과 같이 정의하였다.

■ 데이타 분포의 근사 오차

f_i가 실제 데이타 분포에서의 빈도이고 f_i'이 히스토그램에 의해 근사된 데이타 분포에서의 빈도라고 할 때, 데이타 분포의 근사 오차는 다음과 같다.

$$\text{error} = \sum_{i=1}^{B} (f_i - f_i')^2$$

■ 최적 히스토그램

릴레이션 R에 대해서 $\mathbb{H}_R$를 히스토그램의 집합일 때, $\mathbb{H}_R$중에서 히스토그램 $H_R \in \mathbb{H}_R$에 의해 근사된 데이타 분포의 근사 오차가 최소라면 히스토그램 $H_R \in \mathbb{H}_R$는 최적 히스토그램이다.

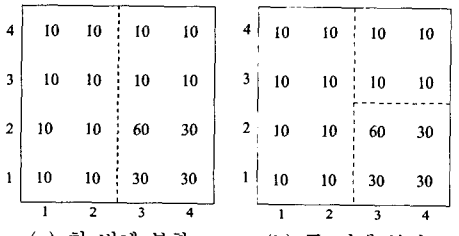
히스토그램에 의해 근사된 분포의 근사 오차는 *SSE*라고 부른다. [7]에서는 각 버킷 내의 모든 값들을 기록한다는 가정과 최적성의 정의하에서 V-Optimal(F, F)이 등가 조인(equality join)과 선택 질의(selection query)의 결과 크기를 추정하는데 최적임을 보였다. 그러나 버킷 내의 모든 값들을 기록한다는 가정은 히스토그램이 작은 저장 공간을 사용하여야 한다는 점에서 비현실적이다. 그리고 V-Optimal(F,F)를 만드는 효율적인 알고리즘도 제안되지 않았다.

[8]에서는 1차원의 데이타 분포를 값으로 정렬한 다음, 동적 프로그래밍 기법을 이용해서 시간 복잡도 $O(N^2 \cdot B)$(N은 데이타 분포의 크기, B는 사용하는 버킷 수)에 V-Optimal(V, F)를 구하는 알고리즘을 제안하였다. 그러나 참고 문헌 [9]에서 주어진 오차 측정 척도(error metric)이 최소가 되도록 2차원 공간을 임으로 분할하는 것은 NP-hard임을 증명됨으로써 실제로 2차원 이상에서 V-Optimal(V, F)를 생성하는 것이 비현실적임이 증명되었다.

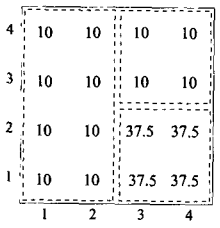
2.2.2 MHIST

[3]에서 제안된 MHIST는 결합 데이타 분포를 β개의 서로 떨어진(disjoint) 버킷으로 분할하여 만드는 다차원의 히스토그램이다. 각 단계에서 가장 분할될 필요가 있

는 속성을 찾아서, 그 속성에 해당하는 차원의 공간을 p 등분하는 알고리즘을 MHIST-p라고 한다. 예를 들어, 분할될 필요성이 있는 속성을 정하기 위해서 MaxDiff [1]를 사용하여 MHIST-2를 만들면, 우선 각 속성의 주변 분포(marginal distribution)를 구한 뒤에 인접한 값들 간의 차가 가장 큰 속성를 분할할 속성으로 정하고, 차가 가장 큰 점을 중심으로 데이타 분포를 둘로 분할한다. 이와 동일한 방법을 재귀적으로 수행하여 데이타 분포가 사용할 버킷 수만큼 분할될 때까지 계속하고, 각각의 버킷의 빈도는 분할된 데이타 분포 내의 총 빈도 또는 평균 빈도로 설정한다.



(a) 첫 번째 분할 (b) 두 번째 분할



(c) 근사된 데이타 분포

그림 5 MHIST

그림 5는 MHIST-2의 생성 과정과 만들어진 MHIST-2에 의해 근사된 데이타 분포를 보여준다. MaxDiff를 사용하여 버킷 3개를 사용하는 MHIST-2를 만드는 과정이 그림 5-(a)와 그림 5-(b)이다. 우선 첫 단계에서 x축과 y축의 주변 분포를 구해보면 x축과 y축 모두 각각 40, 40, 110, 80이 되기 때문에 70으로 주변 분포의 차가 최대가 지점인 2와 3사이를 수직으로 분할한다. 여기에서는 x축으로 분할하였다. 두번째 단계에서는 둘로 분할된 데이타 분포에 대해서 왼쪽 부분은 데이타가 모두 10으로 같으므로 분할될 필요가 없고 오른쪽 부분은 x축으로 주변 분포가 110과 80 이고 y축으로 주변 분포가 20, 20, 90, 60 이므로 y축으로 2와 3사이에 분할되어 그림 5-(b)와 같이 된다. 그림 5-(b)와 같이 분할된 각 영역을 버킷으로 잡고 값과 빈도를 균등하게 근사시켜면 그림 5-(c)와 같이 된다.

2.2.3 GENHIST

GENHIST는 실수형 속성에 대해서 중첩되는 버킷을 사용하여 데이타 분포를 근사시킨 히스토그램이다.

기존의 알고리즘들은 속성의 자료형이 정수라고 가정하고, 정수형 속성에 대해서 히스토그램을 생성하고 정수형의 질의에 대해서 근사 결과를 구한다. 그러나 실제로 릴레이션의 많은 속성들이 실수형이다. 예를 들어, 기상 데이타의 습도, 풍속, 강우량의 속성들이 실수형이다. 그리고 최근에 많은 관심을 모으고 있는 공간 데이타베이스와 멀티미디어 데이타베이스의 속성들도 실수형의 값을 가진다. 따라서 이러한 실수형 속성의 데이타 분포를 근사시키는데 기존의 정수형 속성의 데이타 분포를 근사시키는 알고리즘들을 사용하기 위해서는 각 실수형의 값을 정수형으로 맵핑시키는 과정이 필요하다. 그러나 몇몇 특정 값이 높은 빈도를 가지는 정수형 속성의 데이타 분포에 적합하도록 고안된 알고리즘의 경우에는 실수형으로 맵핑시키더라도 값이 낮은 빈도로 흩어져 있는 실수형 속성의 데이타 분포를 효과적으로 근사시키지 못할 수도 있다. 따라서 [6]에서는 실수형 속성의 데이타 분포를 근사 시키고, 실수형의 질의에 대해서 근사 결과를 구하도록 확장하였다.

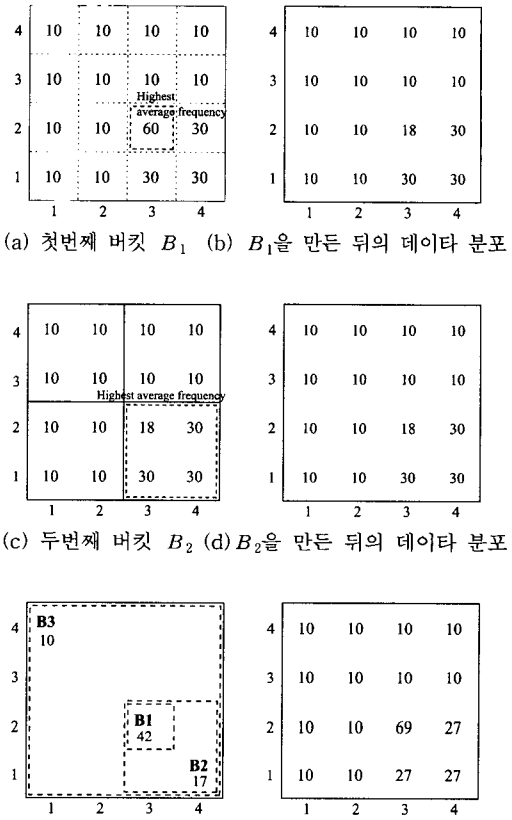
그리고 일반적으로 히스토그램은 데이타 분포를 중첩되지 않는 버킷으로 분할하고 버킷 내부에 데이타가 균등하게 분포되어 있다고 가정한다. 그런데 결합 데이타 분포를 중첩되지 않는(disjoint) 버킷으로 분할하면, 차원이 증가하기 때문에 공간의 부피도 지수승으로 증가함에 따라 필요한 버킷의 수도 지수승으로 증가하는 문제점가 발생 한다. 예를 들어, 5차원에서 각 차원을 4등분하여도 $45 = 1024$ 개의 버킷이 필요하다. 그리고, 버킷을 사용하여 중첩되지 않게 데이타 분포를 분할하는 경우 단지 B개의 영역만을 B개의 버킷의 평균 빈도로 근사시킬 수 있다. 그리고 차원이 높아질수록 데이타들이 산재해서 분포하므로 이렇게 분할하여 균등 분포 가정에 의해서 데이타 분포를 근사시키면, 정확도가 많이 떨어지게 된다. 이런 문제를 해결하기 위해서 참고 문헌 [6]에서는 데이타 분포를 서로 교차하지 않는 버킷으로 분할(partition)한다는 기존의 히스토그램의 개념을 확장해서, 처음으로 중첩된 버킷의 사용을 허용하는 GENHIST를 제안하였다. 버킷들이 서로 중첩되면 중첩된 영역의 빈도는 이 영역과 교차(intersect)하는 버킷들이 서로 나누어 가져야 한다. 따라서 중첩된 버킷의 평균 빈도는 그 영역의 총 빈도가 아니라 그 영역에서 버킷이 실제로 할당받은 총 빈도에 의해서 결정된다. 그래서 데이타

분포에서 특정 영역의 빈도는 그 영역과 교차하는 모든 버킷들의 평균 빈도의 합으로 추정한다.

GENHIST의 알고리즘은 다음과 같다. n 개의 튜플을 가지는 d 차원의 데이타 분포 T가 있다고 가정하자. GENHIST의 세 가지 입력 변수는 각 차원을 분할하는 개수인 ζ, 각 반복(iteration)에서 만드는 버킷의 수인 b_ζ, 그리고 ζ를 감소시키는 비율을 조정하는 α이다. 각 반복에서 GENHIST는 데이타 분포 T를 각 차원에 대해 ζ개로 분할한 뒤에 각 분할(partition)의 평균 빈도를 구한다. 그리고 평균 빈도가 가장 높은 분할 b_ζ개를 선택한다. 각 분할 $c(\in b_\zeta)$는 영역 내에 n_c개의 튜플을 가진다고 하자. 각각의 분할 $c(\in b_\zeta)$에 대해서 c에 이웃하는 분할들의 평균 빈도 f_n를 구해서 이 값이 분할 c의 평균 빈도 f_c보다 크면, 분할 c에 포함되는 $(f_c-f_n)*n_c$ 개의 튜플들을 임의로 골라 데이타 분포 T에서 제거한 다음, 분할 c의 영역을 버킷의 영역으로 하고, 제거한 튜플 수를 버킷의 전체 빈도로 하여 버킷을 만든다. 다음 반복을 위해서 GENHIST는 ζ를 식 $\lfloor \zeta \cdot \min(|R|/(|R|+S), \alpha) \rfloor$ (S는 각 반복에서 제거된 튜플의 수이고, |R|는 데이타 분포 R에 남아있는 튜플의 수이다.)을 통해 다시 계산한다. 만약 ζ가 1보다 작으면 데이타 집합 전체의 영역을 포함하는 버킷을 하나 만든다. 만약 R에 하나 이상의 튜플이 존재하고 ζ가 1보다 크면, 다음 반복을 수행한다.

그림 6에서는 ζ가 4, ζ가 0.5, 그리고 ζ가 1일 때, GENHIST를 가지고 주어진 데이타 분포를 근사시키는 예를 보여준다. 그림 6-(a)에서는 데이타 분포를 각 차원에 대해서 4등분하였을 때 평균 밀도가 가장 높은 분할을 점선으로 보여준다. 이 분할을 둘러싼 주변 분할들의 평균 빈도가 18이므로, 이 분할은 주변보다 48만큼이 높다. 따라서 첫번째 버킷 B_1은 이 분할을 영역으로 하고, 전체 빈도는 60에서 48을 뺀 18이 된다. 그림 6-(b)는 버킷 B_1을 만들고 달라진 데이타 분포를 보여준다. 첫번째 수행 이후에 다시 계산된 ζ값 2를 가지고 동일하게 수행을 하면 그림 6-(c)에서 점선으로 표시된 분할이 버킷 B_2가 되고, 버킷 B_2를 만들고 달라진 데이타 분포는 그림 6-(d)가 된다. 그리고 다시 계산된 ζ값이 1이므로 데이타 분포 전체를 영역으로 하고 나머지 튜플을 전체 빈도로 하는 버킷 B_3을 만든다. 그림 6-(e)가 위의 과정을 통해서 만들어진 버킷 B_1, B_2, B_3로 구성된 GENHIST를 나타내고, 그림 6-(f)가 이 GENHIST에 의해서 근사된 데이타 분포를 나타낸다.

그리고 기존의 히스토그램은 각 차원별로 버킷 영역의 최소값과 최대값 그리고 그 버킷의 총 빈도 또는 평균 빈도를 저장하므로, d차원의 경우 버킷당 2d+1의 수를 저장해야 한다. 그런데 GENHSIT는 데이타 분포 전체 영역을 ζ등분하였을 때, 등분의 수인 ζ와 몇 번째 분할인지를 나타내는 인덱스를 알면 버킷의 영역을 알 수 있으므로 차원에 관계없이 ζ와 인덱스 그리고 총 빈도 세 가지만을 저장하면 된다. 따라서 GENHIST방식으로 버킷 정보를 저장하게 되면 같은 크기의 저장 공간을 가지고 히스토그램을 생성하는 경우 차원이 증가함에 따라 상대적으로 더 많은 버킷을 사용할 수 있다.



(a) 첫번째 버킷 B_1 (b) B_1을 만든 뒤의 데이타 분포

(c) 두번째 버킷 B_2 (d) B_2을 만든 뒤의 데이타 분포

(e) 생성된 히스토그램 (f) 근사된 데이타 분포

그림 6 GENHIST

3. OPT 알고리즘

본 장에서는 먼저 버킷간의 중첩을 허용함으로써 얻을 수 있는 이득에 대해 설명한 다음에 근사 오차 SSE를 최소화 시키도록 각 버킷의 평균 빈도를 최적화시키는 알고리즘인 OPT를 설명한다.

3.1 중첩된 버킷의 사용

[6]에서 중첩된 버킷의 사용을 제안하기 전에는 대부분의 히스토그램이 서로 떨어진(disjoint)한 버킷을 사용하여서 데이타 분포를 분할했다. 그러나 다음의 간단한 예를 통해서도 버킷을 중첩시키는 경우에 더 적은 버킷을 사용하여 최적 히스토그램을 만들 수도 있음을 알 수 있다. 주어진 데이타 분포 T = {(10,10),(20,30), (30,20)}에 대해, 중첩되는 버킷 2개를 가지고 히스토그램을 만드는 경우와 버킷과 중첩되지 않는 버킷 2개로 히스토그램을 만드는 경우를 생각해보자. 위의 데이타 분포를 중첩되지 않은 2개의 버킷으로 히스토그램을 만드는 경우에는 원래 데이타 분포와 동일하게 근사될 수 없다.

그림 7-(a)에서처럼 서로 떨어진 2개의 버킷을 사용하면bucket2에서 빈도 30과 20이 균등 빈도 가정에 의해 25로 근사되어 *SSE*가 50이 된다. 그러나, 그림 7-(b)에서처럼 중첩된 2개의 버킷을 사용하여 bucket1 [10, 20]의 평균 빈도를 10, bucket2[20, 30]의 평균 빈도를 20으로 두면 값 20에 해당되는 영역이 bucket1과 bucket2에 의해서 30으로 근사되면서 원래 데이타 분포와 동일한 분포로 근사되므로 *SSE*는 0이 된다. 즉, 버킷간에 중첩을 허용하면 실제 만든 버킷의 수보다 많은 빈도를 표현할 수 있어서 동일한 개수의 버킷을 사용하는 경우 데이타 분포를 보다 정확하게 근사시킬 수 있다.

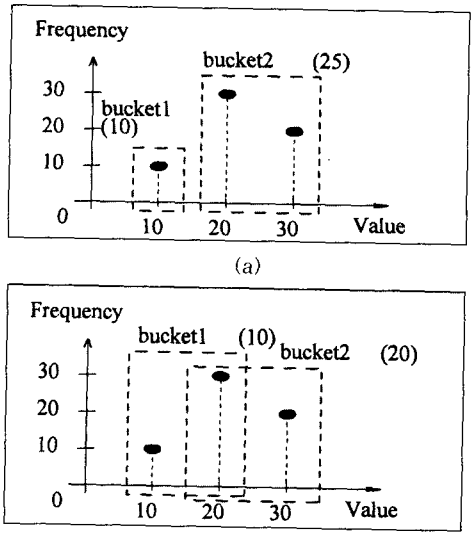


그림 7 (a) 떨어진 버킷을 사용한 경우
(b) 중첩된 버킷을 사용한 경우

그리고 다차원에서 버킷간의 중첩을 허용하면 사각형의 버킷을 이용하여 사각형이 아닌 공간을 근사 시킬 수 있다. 그림 8-(a)처럼 데이타 분포상의 일부분이 주위보다 빈도가 높거나 낮은 경우 전체를 차지하는 버킷 안에 빈도의 차가 심한 부분에 버킷을 둠으로써 사각형이 아닌 형태의 공간의 빈도도 비교적 정확하게 근사시킬 수가 있다.

그림 8-(b)처럼 X[3, 3] Y[3, 3] 영역에 버킷 B_1와 X[3, 4] Y[3, 4] 영역에 버킷 B_2와 X[1, 4] Y[1, 4] 영역에 버킷 B_3를 두고 각각의 평균 빈도를 30, 20, 10으로 두면 실제 데이타 분포와 동일하게 근사 시킬 수 있다. (각 버킷의 최적 평균 빈도 30, 20, 10가 어떻게 계산되는지는 뒤에 중첩된 버킷의 평균 빈도를 최적화 시키는 OPT 알고리즘에서 설명하기로 한다.)

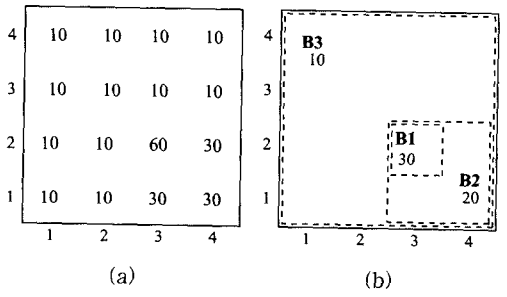


그림 8 (a) 실제 데이타 분포
(b) 중첩된 버킷을 사용한 히스토그램

위에서 언급한 장점을 활용하게 위해서 본 논문에서는 **중첩되는 버킷을 사용하는 히스토그램**을 생성하는 문제를 다룬다.

3.2 버킷의 평균 빈도를 최적화 시키는 OPT 알고리즘

2장에서 언급되었듯이 최적 히스토그램(optimal histogram)이란 히스토그램의 집합에서 오차 측정기준(error metric)을 최소화시키는 히스토그램을 의미한다. 참고문헌 [7]에서는 오차 측정 기준으로 *SSE*를 사용하였는데, *SSE*는 원래 데이타 분포와 근사된 데이타 분포간의 차를 나타내므로 *SSE*가 작아질수록 원래 데이타 분포에 가깝게 근사되었음을 의미한다. 따라서 본 논문에서는 N_B개의 버킷이 주어졌을 때 *SSE*를 최소화시키도록 각 버킷의 평균 빈도를 결정하는 알고리즘을 제안한다. 기존의 MHIST나 GENHIST에서는 최소화를 정확하게 하지 않고 근사적으로 계산하므로 본 논문에서 제안하는 OPT 알고리즘만이 처음으로 *SSE*를 최소화할 수 있는 알고리즘이다. 기존의 알고리즘들은 휴리스틱을 쓰기

때문에 *SSE*의 정확도를 OPT와 수식적으로 비교하기는 굉장히 어렵다. 하지만 4장에서 보여 주듯이 여기서 제안된 OPT 알고리즘은 평균 상대 오차를 기존의 알고리즘에 비해서 많이 향상시킴을 볼 수 있으므로 OPT 알고리즘의 우수성을 알 수 있다.

버킷이 서로 중첩되는 것을 허용하지 않는 경우에는 버킷의 빈도는 균일 빈도 가정에 의해서 평균 빈도로 나타낸다. 그러나 GENHIST처럼 버킷이 서로 중첩되면 중첩되는 부분의 빈도를 어느 버킷이 얼마만큼 할당 받을지의 알 수 없기 때문에 각 버킷의 빈도를 결정하는 것이 문제가 된다. GENHIST의 경우에는 휴리스틱하게 주변 분할들의 평균 빈도보다 큰 만큼을 그 버킷의 평균 빈도로 한다. 중첩된 버킷의 평균 빈도는 그 영역의 총 빈도가 아니라 그 영역에서 버킷이 실제로 할당받은 총 빈도에 의해서 결정된다. 그래서 데이타 분포에서 특정 영역의 빈도는 그 영역과 교차하는 모든 버킷들의 평균 빈도의 합으로 추정한다. 따라서 데이타 분포는 버킷별로 근사되는 것이 아니라 버킷에 의해 만들어지는 각각의 영역별로 근사된다. 이 때 버킷에 의해 근사되는 각각의 영역을 필요 영역(required region) 이라고 정의한다.

정의 1

데이타분포를 나타내는 전체 영역을 각 축(axis)에 평행하게 나누는 NB개의 버킷들에 의해서 전체 데이타 분포의 영역이 분할되는데, 같은 버킷들에 의해서 근사되는 영역 각각을 **필요 영역(required region)**이라고 한다.

필요 영역의 정의에 의해서 각 필요 영역은 데이타 분포상의 각 점이 어떤 버킷의 영역에 포함되는지를 각 버킷에 대한 불린(boolean)연산으로 표시하면 구할 수 있다. 그러므로 최대 2^{NB}개의 필요 영역이 만들어 질 수 있다. 그러나 실제로 버킷간에 중첩되지 않거나 한 버킷이 다른 버킷의 내부에 존재하는 포함 관계로만 중첩되는 경우에는 데이타 분포 전체를 포함하는 버킷이 있는 경우 N_B개의 필요 영역이, 데이타 분포 전체를 포함하는 버킷이 없는 경우 N_B+1개의 필요 영역이 만들어진다.

그림 9에서는 2개의 버킷 B_1과 B_2에 의해 만들어지는 4개의 필요 영역 R_1, R_2, R_3, R_4를 보여준다. 필요 영역 R_1과 R_3의 경우에는 각각 B_1과 B_2에 의해서만 근사되는 영역이므로 각각 B_1과 B_2의 평균 빈도를 가지고 이 영역의 빈도를 추정한다. 필요 영역 R_2의 경우에는 B_1과 B_2가 중첩되는 영역이므로 B_1과 B_2의 평균 빈도의 합으로 이 영역의 빈도를 추정한다. 필요 영역 R_4는 공간상으로는 좌측 하단과 우측 상단 두 부분으로 나뉘지만 둘다 B_1과 B_2 어느 버킷에 의해서도 근사되지 않는 영역이므로 하나의 필요 영역이 되고, 이 영역의 빈도는 0으로 추정된다.

Bucket / Required region	B1	B2
R1	Y	N
R2	Y	Y
R3	N	Y
R4	N	N

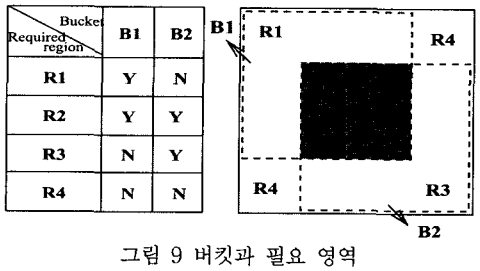


그림 9 버킷과 필요 영역

3.2.1 중첩되는 버킷을 사용하였을 때 SSE 계산하기

*SSE*를 계산하기 전에 앞으로 사용될 몇 가지 기호들을 표 5에 정리하였다. 부가적인 기호들은 필요할 때 소개될 것이다.

기존의 중첩을 허용하지 않는 히스토그램에 의해 근사된 데이타 분포의 *SSE*는 다음과 같이 계산된다.

$$\text{SSE} = \sum_{i=1}^{D} (f_i - f_i')^2 \qquad (1)$$

$$= \sum_{j=1}^{N_B} \sum_{v \in S_{B_j}} (f_v - F_{B_j})^2. \qquad (2)$$

수식 1은 실제 빈도와 근사된 빈도의 차를 제곱한 것들을 각각 합한다는 *SSE* 원래 의미를 나타낸다. 버킷간에 중첩이 없는 경우에는 데이타 분포상의 하나의 빈

표 5 기호

기호	설명	기호	설명	기호	설명
B_j	버킷 j	R_i	필요 영역 i	N_B	버킷의 총 수
N_R	필요 영역의 총 수	S_{B_j}	B_j가 차지하는 공간	S_{R_i}	R_i가 차지하는 공간
f_v	데이타 분포의 각 점 v의 빈도	T_{B_j}	B_j의 총 빈도	D_{B_j}	B_j 내의 각 점의 수
F_{B_j}	B_j의 평균 빈도	T_{R_i}	R_i의 총 빈도	D_{R_i}	R_i 내의 각 점의 수
F_{R_i}	R_i의 평균 빈도				

도는 하나의 버킷에 의해서 근사되기 때문에 수식 2에서와 같이 각각의 버킷에 대해서 *SSE*를 계산한다. 그런데 버킷 간에 중첩을 허용하게 되면 데이타 분포상의 하나의 빈도는 교차하는 모든 버킷들의 평균 빈도의 합으로 근사되기 때문에 *SSE*를 계산하는 식이 확장되어야 한다.

d차원의 데이타 분포을 N_B개의 버킷으로 근사시킬 때, 이 히스토그램의 *SSE*는 다음과 같다.

$$SSE = \sum_{i=1}^{N_B} \sum_{v \in S_{R_i}} (f_v - FR_i)^2 \quad (3)$$

$$= \sum_{i=1}^{N_B} \sum_{v \in S_{R_i}} (fv - \sum_{j=1}^{N_B} \delta_{ij} FB_j)^2$$

$$\text{where } \delta_{ij} = \begin{cases} 1 & \text{if } S_{R_i} \subseteq S_{B_i} \\ 0 & otherwise \end{cases} \quad (4)$$

$$= \sum_{i=1}^{N_B} \sum_{v \in S_{R_i}} (fv - \sum_{j=1}^{N_B} \delta_{ij} \left(\sum_{k=1}^{N_B} \delta_{kj} T_{R_k} W_{kj} / \sum_{k=1}^{N_B} \delta_{kj} D_{R_k} \right))^2. \quad (5)$$

수식 3에서 각각의 필요 영역에 대해서 *SSE*를 계산한다. 즉, 각 필요 영역 R_i 내의 각 점에 대해서 실제 빈도와 근사된 빈도간의 차를 제곱한 것들을 합산한다. 각 필요 영역 R_i의 평균 빈도는 필요 영역 R_i와 교차하는 모든 버킷들의 평균 빈도의 합이므로 수식 4에서는 F_{R_i}를 δ_{ij}와 F_{B_j}로 치환한다. 여기서 δ_{ij}는 버킷 B_j와 필요 영역 R_i 간의 관계를 나타내는데, 필요 영역 R_i가 버킷 B_j 영역의 일부분이면 값으로 1을 가지고 그렇지 않으면 0을 가진다. 수식 5에서는 각 버킷의 평균 빈도 F_{B_j}를 필요 영역 R_k의 총 빈도 T_{R_k}와 필요 영역 R_k내의 각점의 수 D_{R_k}로 치환하는데, 버킷간의 중첩을 허용함으로써 중첩되는 영역의 총 빈도 중에서 실제로 할당받을 빈도를 가지고 평균 빈도를 구해야 한다. 수식 5에서 W_{kj}는 T_{R_k}중에서 실제로 B_j에 할당되는 비율을 의미한다. 따라서 $\sum_{j=1}^{N_B} W_{kj} * \delta_{kj} = 1$ 을 만족해야하는데, 이것은 T_{R_k}가 각 버킷에 할당되는 비율인 W_{kj}의 합은 1이 되어서 결국 한 필요 영역의 빈도는 그 영역과 교차되는 모든 버킷의 빈도의 합이 되어 총 빈도에는 변함이 없음을 의미한다. W_{kj} 값을 어떻게 정하느냐에 따라서 각 버킷의 평균 빈도가 결정되고, 이에 의해서 히스토그램의 *SSE*가 계산된다.

3.2.2 SSE를 최소화시키는 버킷의 평균 빈도 구하기

W_{kj} 값이 정해지면 각 버킷의 평균 빈도가 결정되므로, 문제는 *SSE*를 최소화시키는 W_{kj} $\frac{\partial^2}{\partial W_{kj}^2} SSE \geq 0$를 구하는 것이다. 이므로, $\frac{\partial}{\partial W_{kj}} SSE = 0$ 을 만족하는 W_{kj}의 값이 *SSE*를 최소화시킨다. 따라서 최적의 W_{kj}를 구하기 위해서 모든 W_{kj} ($1 \leq j \leq NB$, $1 \leq k \leq NR$)에 대해 수식 5를 편미분(즉, $\frac{\partial}{\partial W_{kj}} SSE = 0$) 해야하는데, 수식 $\sum_{j=1}^{N_B} W_{kj} * \delta_{kj} = 1$에 의해서 수식 내의 W_{kj}끼리는 의존 변수이기 때문에 바로 편미분할 수 없다. 따라서 $\delta_{kh} \neq 0$이고 $1 \leq h \leq NB$인 임의의 W_{hk}를 $1 - \sum_{j=1, j \neq h}^{N_B} W_{kj} * \delta_{kj}$으로 치환해서 나머지 미지수 W_{kj} 들을 독립변수화시킨 다음, 수식 5를 편미분을 해서 1차 방정식을 만든다(즉, $\frac{\partial}{\partial W_{kj}} SSE = 0$). 이렇게 구한 1차 방정식은 $\sum_{j=1}^{N_B} \sum_{k=1}^{N_B} c_{kj} W_{kj} = d_{kj}$ (c_{kj}와 d_{kj}는 상수)의 형태가 된다. 따라서 편미분에 의해서 얻어진 방정식의 수는 O(NBNR)이고 방정식에서 O(NBNR)개의 미지수를 가진다. W_{kj}를 계산하는데 충분한 수의 방정식이 있으므로, 가우스 소거법(Gaussian elimination method)과 같은 수치 해석학에서 1차 방정식을 푸는 알고리즘들을 이용할 수 있다. 가우스-조던 소거법(Gauss-Jordan elimination method)[10]은 n개의 미지수가 있는 n개의 1차 방정식의 해를 구할 때 $O(n^3)$의 시간이 걸린다. 따라서 가우스-조던 소거법을 사용하면 $O((N_B N_R)^3)$가 걸려서 $N_B N_R$개의 W_{kj}를 구할 수 있다. W_{kj}를 구하고 나면 $\sum_{k=1}^{N_B} \delta_{kj} T_{R_k} W_{kj} / \sum_{k=1}^{N_B} \delta_{kj} D_{R_k}$를 통해서 각 버킷의 평균인 F_{B_j}를 구할 수 있다.

주어진 버킷에 대해서 최적의 평균 빈도를 구하는 알고리즘을 이용하여, 그림 7-(b)에 표시된 버킷 B_1, B_2의 최적 평균 빈도를 구하는 방법은 다음과 같다. 버킷 B_1과 B_2에 의해서 만들어지는 필요 영역을 R_1[1, 1] R_2[2, 2] R_3[3, 3]라고 할 때, 필요 영역 R_2에 버킷 B_1과 버킷 B_2가 중첩되므로 빈도 30이 각 버킷에 나누어지게 된다. 30이 B_1에 할당되는 비율을 W_{21}, B_2에 할당되는 비율을 W_{22}라고 하면 $W_{21}+W_{22}$는 1이 되고, 버킷 B_1의 평균 빈

도는 $(10+30W_{21})/2$이고, 버킷 B_2의 평균 빈도는 $(30W_{22}+20)/2$이다. 그런데 버킷 B_2의 평균 빈도에서 W_{22}를 $1-W_{21}$로 치환해주면 $(50-30W_{21})/2$가 된다. 그러면 *SSE*를 계산해보면 다음과 같다.

$$SSE = ((10 - \frac{10+30W_{21}}{2})^2 + 0 + (20 - \frac{50-30W_{21}}{2})^2 = \frac{(30W_{21}-20)^2}{2}$$

수식 7을 W_{21}에 대해서 편미분하여 1차식을 구하면 $30(30W_{21}-10)=0$ 이 되고 이 일차식의 해는 $W_{21}=\frac{1}{3}$이므로 이를 위에서 W_{21}이 들어가는 다항식으로 표현한 각 버킷의 평균 빈도에 대입하면 버킷 B_1의 최적 평균 빈도는 10, 버킷 B_2의 최적 평균 빈도는 20이 구해진다.

그리고, 다음의 예는 중첩된 버킷을 사용하는 GENHIST를 OPT 알고리즘을 통해서 개선시킬 수 있음을 보여준다. 동일한 그림 8-(a)의 데이타 분포에 대해서 그림 8-(b)에서 처럼 주어진 3개의 버킷으로 히스토그램을 만들었을 때, GENHIST는 그림 6-(c)에서와 같이 데이타 분포를 근사시켜서 *SSE*가 108이 되고, 앞의 GENHIST와 동일한 3개의 버킷에 대해서 OPT 알고리즘을 통해서 각 버킷의 평균 빈도를 최적화 시켜서 만든 히스토그램은 그림 8-(b)로 데이타 분포를 완벽하게 재현하여서 SSE가 0이 된다.

3.3 OPT 알고리즘의 시간 공간 복잡도

중첩된 버킷의 평균 빈도를 최적화 시키는 OPT 알고리즘에서 크기 N인 데이타 분포를 한 번 스캔하여 주어진 버킷에 의해서 만들어지는 필요 영역을 구하는데 시간이 O(N) 걸린다.

그리고 편미분하여 얻은 $O(N_BN_R)$ 개의 1차 방정식의 해 W_{kj}를 구할 때 $O((N_BN_R)^3)$이 걸린다. 따라서 중첩된 버킷의 평균 빈도를 최적화시키는 *OPT* 알고리즘의 시간 복잡도는 $O(N + (N_BN_R)^3)$이다. 그리고 *SSE*를 미지수 W_{kj}를 포함하는 다항식으로 나타낼 때 같은 필요 영역내의 근사된 빈도는 동일하므로 필요 영역의 수 N_R만큼 항이 나온다. 그리고 다항식으로 표현되는 *SSE*를 각각의 W_{kj}에 대해서 편미분하여 구한 (N_BN_R)개의 1차 방정식을 저장해야 하므로 따라서 중첩된 버킷의 평균 빈도를 최적화시키는 OPT 알고리즘의 공간 복잡도는 $O(N_B N_R^2)$이 된다.

기존의 GENHIST 알고리즘은 여러가지 입력 변수를 어떻게 지정 하느냐에 따라서 수행속도가 다르게 되며 또 만들어진 히스토그램의 정확도도 달라지게 된다. 그렇기 때문에 GENHIST의 알고리즘의 수행속도를 정확이 나타내기는 힘들지만 휴리스틱을 써서 수행속도가 빠르기 때문에 OPT 알고리즘의 수행속도는 GENHIST 알고리즘에 비해 느리게 된다. 하지만 GENHIST 알고리즘에서는 아무리 시간을 많이 준다 하더라도 각 버킷들의 평균 빈도를 최적화 시키지 못하므로 OPT 알고리즘이 히스토그램을 생성하는데 더 걸리는 시간은 만들어진 히스토그램을 이용하여 데이타웨어 하우징에서 계속해서 들어오는 근사 질의 처리시에 계속 더 정확한 결과를 제공하므로 OPT 알고리즘의 수행시간의 단점에도 불구하고 더 유용하고 우수하게 사용될 수 있음을 알 수 있다.

4. 실 험

3장에서 제안한 **OPT 알고리즘**에 의해 버킷의 평균 빈도를 최적화 시킴으로써 중첩된 버킷으로 구성된 히스토그램의 정확도를 개선시킴을 보이기 위해서 실생활(real-life) 데이타 집합과 합성(synthetic) 데이타 집합에 대하여 실험하였다.

4.1 실험 대상 알고리즘

GENHIST와 OPT 알고리즘으로 GENHIST의 버킷의 평균 빈도를 최적화시킨 GENHIST-OPT를 비교한다.

- GENHIST

GENHIST의 입력 변수인 ζ의 초기값, b_ζ, α는 참고 문헌 [6]에서의 설명에 따라서 값을 설정했다. α는 참고 문헌 [6]에서 사용한 그대로 d차원에서 $\alpha = (1/2)^{1/d}$로 하였고, ζ의 초기값은 ζ등분한 하나의 분할의 크기가 2가 되는 것에서 10이 되는 것까지 해보았고, 주어진 ζ의 초기값과 α의 값으로 최대 B개의 버킷을 잡을 수 있도록 설정하여, 히스토그램을 만들고 같은 B개의 버킷을 사용한 *GENHIST*중에서 가장 좋은 성능을 보이는 것으로 비교하였다. 그리고 2장에서 언급했듯이 저장 공간상으로 GENHIST는 하나의 버킷당 차원에 관계없이 3개의 숫자를 저장한다.

- GENHIST-OPT

GENHIST-OPT는 GENHIST와 동일한 버킷을 가지고 OPT에 의해서 각 버킷의 최적 평균 빈도를 다시 계산한 히스토그램이다. 그리고 저장 공간 상으로 GENHIST와 동일한 방식으로 버킷을 저장하므로 하나의 버킷당 차원에 관계없이 3개의 숫자를 저장한다.

4.2 질의 집합과 오차 측정기준

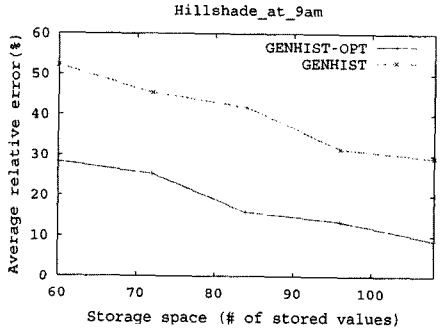
각각의 알고리즘들의 정확도를 측정하기 위해서 각 데이타 집합 별로 질의 집합 Q를 만들었다. 각각의 질의 q는 범위 질의로 각 차원의 속성 X에 대해서 min ≤ X ≤ max의 형태를 가진다. 그리고 각 질의의 선택율(selectivity)이 [min_selectivity, max_selectivity] 되는 질의들로 질의 집합을 구성하였다. 각 데이타 집합 별로 선택율의 범위를 달리한 것은 데이타 집합을 설명하는 부분에서 이를 언급하기로 한다. 하나의 데이타 집합에 대해서 위와 같은 질의로 구성된 질의 집합 Q의 크기|Q|는 10,000으로 하였다. 그리고 정확도를 측정하기 위해서 2장에서 소개한 평균 상대 오차를 사용하였다.

4.3 실생활 데이타 집합

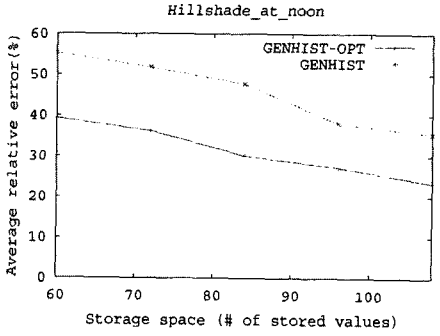
실생활 데이타로는 UCI KDD Repository[1]의 Forest Cover Type을 사용하였다. 이 데이타 집합은 GENHIST의 성능 평가를 위해서 참고 문헌 [6]에서 사용한 것으로 590,000의 튜플로 되어 있고 그 중에서 10개가 숫자 속성으로 되어 있다.

1차원에서는 Forest Cover Dataset에서 속성 Hillshade_at_9am과 속성 Hillshade_at_noon을 각각 추출하여서 근사시킬 데이타 분포로 사용하였고, 2차원에서는 Slope와 Hiilsahde_at_9am을 Slope과 Hillshade_at_noon을 추출하여서 근사시킬 2차원의 결합 데이타 분포로 사용하였다. 그리고, 1차원에서는 선택율이 최소 0.5%에서 최대 1%가 되는 범위 질의를 임의로 10,000개를 생성하여서 질의 집합을 만들고, 2차원에서는 선택율이 최소 1%에서 최대 10%가 되는 범위 질의를 임의로 10,000개를 생성하여서 질의 집합을 만들었다.

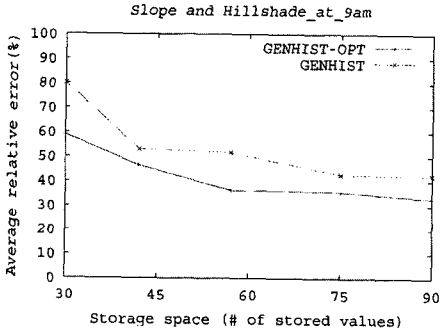
4.4 실생활 데이타 집합에 대한 결과



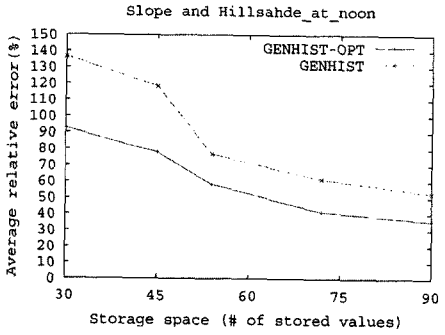
(a)



(b)



(c)



(d)

그림 10 실생활 데이타 집합에 대한 평균 상대오차

그림 10-(a)와 10-(b)는 실생활 데이타 Forest Cover Type의 속성 Hillshade_at_noon과 속성 Hillshade_at_9am에 대해 GENHIST와 GENHIST-OPT 의 평균 상대 오차를 나타낸 그래프이다. 저장 공간을 60에서 110으로 늘려가면서 평균 상대 오차의 변화를 살펴보았다. GENHIST-OPT가 GENHIST에 비해서 그림 10-(a)에서 저장 공간이 110일 때 보여주듯이 최대 69%까지 정확도가 향상되었다. 그림 10-(c)와 10-(d)는 실

1) http://kdd.ics.uci.edu/summary.data.type.html

생활 데이타 Forest Cover Type의 속성 Slope과 속성 Hillshade_at_noon, 속성 Slope과 속성 Hillshade_at_9am 에 대해 생성한 2차원의 GENHIST와 GENHIST-OPT의 평균 상대 오차를 나타낸 그래프이다. 저장 공간을 30에서 90으로 늘려가면서 평균 상대 오차의 변화를 살펴보았다. GENHIST-OPT가 GENHIST에 비해서 최대 34%까지 정확도가 향상되었다.

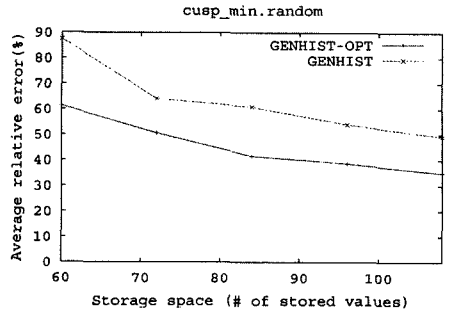
4.5 합성 데이타 집합

합성 데이타 분포 ZIPF-DATA는 참고 문헌 [1]에서 사용한 Zipf 분포를 따르는 데이타 분포의 집합이다. Zipf 분포에 대해서는 참고 문헌 [11]에 자세히 설명되어 있다. 본 논문에서는 1차원의 합성 데이타 집합을 전체 튜플 수 T는 50만개, 값의 수를 200개, 값의 범위를 [1, 1000]으로 하고, 빈도 집합은 z=1인 Zipf 분포로 값 집합에서 간격은 z=1인 cusp_max와 cusp_min 각각에 대해서 생성하였다. 이것은 참고 문헌 [3]에서 사용했던 합성 데이타를 동일하게 재현한 것이다. 2차원의 합성 데이타 집합은 전체 튜플 수 T는 50만개, 값의 수를 200개, 값의 범위(domain)를 [1, 1000]으로 하고, 결합 빈도 집합은 z=1인 Zipf 분포로 각 차원의 값 집합은 z=1인 cusp_max와 cusp_min을 따르도록 생성하였다.

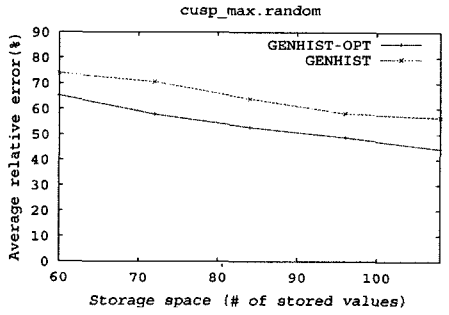
그리고, 1차원에서는 선택율이 최소 0.5%에서 최대 1%가 되는 질의를 임의로 10,000개를 생성하여서 질의 집합을 만들고, 2차원에서는 선택율이 최소 1%에서 최대 30%가 되는 질의를 임의로 10,000개를 생성하여서 질의 집합을 만들었다. 2차원의 경우 선택율이 1차원에 비해서 많이 높아진 것은 2차원의 데이타 집합이 1차원에 비해서 훨씬 분포가 치우쳐 있어서, 빈도가 높은 점들이 질의 집합에 들어가기 위해서 선택율을 높였다.

4.6 합성 데이타 집합에 대한 결과

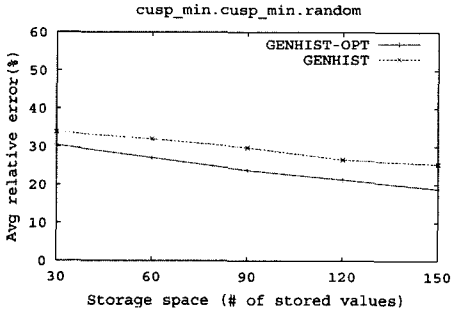
GENHIST-OPT가 GENHIST보다 정확도가 10% 이상 향상되었다. 그림 11-(a)와 (b)는 합성 데이타 집합에 대해서 1차원의 GENHIST-OPT와 GENHIST의 평균 상대 오차를 나타낸 그래프이다. 저장 공간을 60에서 110으로 늘려가면서 평균 상대 오차의 변화를 살펴보았다. GENHIST-OPT가 GENHIST에 비해서 최대 29%까지 정확도가 향상되었다. 그림 11-(c)와 (d)는 합성 데이타 집합에 대해서 2차원의 GENHIST-OPT와 GENHIST의 평균 상대 오차를 나타낸 그래프이다. 저장 공간을 30에서 150으로 늘려가면서 평균 상대 오차의 변화를 살펴보았다. GENHIST-OPT가 GENHIST에 비해서 최대 36%까지 정확도가 향상되었다.



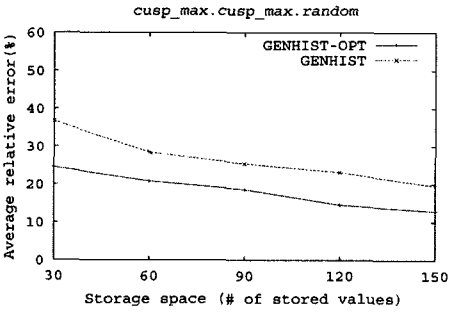
(a)



(b)



(c)



(d)

그림 11 합성 데이타 집합들에 대한 평균 상대 오차

5. 결 론

기존의 히스토그램들이 데이타 분포를 버킷으로 분할하였으나 GENHIST와 같이 중복된 버킷을 사용하면 다차원 데이타 분포를 좀더 효과적으로 근사 시킬 수 있으므로, 본 연구에서는 버킷 간에 중첩을 허용하고 주어진 버킷에 대해서 근사 오차 SSE(Sum Squared Error)를 최소화시키도록 버킷의 최적 평균 빈도를 결정하는 OPT 알고리즘을 제안하였다. 그리고 GENHIST에 의해서 중첩된 버킷들로 구성된 히스토그램을 OPT에 의해서 각 버킷의 최적 평균 빈도를 계산해서 GENHIST를 개선시켰다. 실생활(real-life) 데이타들과 합성(synthetic) 데이타들에 대해서 테스트한 실험 결과는 GENHIST의 중첩된 버킷의 평균 빈도를 알고리즘 OPT에 의해 구한 최적 평균 빈도로 재설정함으로써 기존의 GENHIST의 정확도를 현저하게 향상시킴을 보여주었다.

참 고 문 헌

[1] V. Poosala, Y. E. Ioannidis, P. J. Haas, and E. J. Shekita, "Improved histograms for selectivity estimation of range predicates", In Proc. Int'l Conf. on Management of Data, ACM SIGMOD, Montreal, Canada, pp. 294-305, June 1996.

[2] M. Muralikrishan and D. J. DeWitt, "Equi-depth histograms for estimatng selectivity factors for multi-dimensional queries," In *Proc. Int'l Conf. on Management of Data, ACM SIGMOD*, Chicago, Illinois, pp. 28-36, June 1988.

[3] V. Poosala and Y. E. Ioannidis, "Selectivity estimation without the attribute value independence assumption", In *Proc. the 23rd Int'l Conf. on Very Large Data Bases*, Athens, Greece, pp. 486-495, August 1997.

[4] Y. Matias, J. S. Vitter, and M. Wang, "Wavelet-based histograms for selectivity estimation", In *Proc. Int'l Conf. on Management of Data, ACM SIGMOD*, Seattle, Washington, pp. 448-459, June 1998.

[5] J. Lee, D. Kim, and C. Chung, "Multi-dimensional Selectivity Estimation Using Compressed Histogram Information", In *Proc. Int'l Conf. on Management of Data, ACM SIGMOD*, Philadelphia, Pennsylvania, pp. 205-214, June 1999.

[6] D. Gunopulos, G. Kollios, V. J. Tsotras, and C. Domeniconi, "Approximating multi-dimensional aggregate range queries over real attributes", In *Proc. Int'l Conf. on Management of Data, ACM SIGMOD*, Dallas, Texas, pp. 463-474, June 2000.

[7] E. Ioannidis and V. Poosala, "Balancing histogram optimality and practicality for query result size estimation", In *Proc. Int'l Conf. on Management of Data, ACM SIGMOD*, San Jose, California, pp. 233-244, May 1995.

[8] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel, "Optimal histograms with quality guarantees", In *Proc. the 24th Int'l Conf. on Very Large Data Bases*, New York, NY, pp. 275-286, August 1998.

[9] S. Muthukrishnan, V. Poosala, and T. Suel, "On rectangular partitionings in two dimensions: Algorithms, complexity, and applications", In *Proc. Int'l Conf. on Database Theory*, Jerusalem, Israel, pp. 236-256, January 1999.

[10] S. A. William, H. Press, B. P. Flannery, and W. T. Vettrling, *Numerical recipes in C: The art of scientific computing*, Cambridge University Press, 1993.

[11] V. Poosala, *Histogram-Based Estimation Techniques in Database Systems*, Ph. D. dissertation, University of Wisconsine-Madison, 1997.

문 진 영

1996.3~2000.2 경북대학교 컴퓨터공학과 (B.S.) 2000.3~2002.2 한국과학기술원 전자전산학과 전산학전공 (M.S.) 2002.2~현재 한국전자통신연구원 컴퓨터소프트웨어연구소

심 규 석

1986년 서울대학교 전기공학과 수석 졸업 (학사) 1998년 University of Maryland, College Park (석사) 1993년 University of Maryland, College Park (박사) 1994년 - 1994년 Federal Reserve Board, Research Staff 1994년 - 1996년 IBM Almaden Research Center, Restearch Staff 1996년 - 2000년 Bell Laboratories, Member of technical Staff 1999년 - 2002년 KAIST 전산학과 조교수 2001년 - 2002년 Microsoft Research, Visiting Scientist 2002년 - 현재 서울대학교 전기컴퓨터 공학부 부교수 현재 VLDB저널과 KAIS저널의 Editorial Board ACM SIGKDD Curriculum Committee Member ACM SIGMOD, VLDB, ICDE, ACM SIGKDD, PAKDD, ICDM, EDBT을 비롯 다수의 국제 학술대회의 Program Committee Member임 관심 분야는 데이타마이닝, 데이타베이스, XML, Stream Data