제26회 한국정보처리학회 추계학술발표대회 논문집 제13권 제2호 (2006. 11)

경험적인 가중치를 고려한 증권시장용 계정 원장 DB의 인덱싱 방안 연구

강석희*, 최진영
고려대학교 컴퓨터정보통신대학원
e-mail:blockade, choi@korea.ac.kr

A Study on Indexing Strategy for Stock Ledger DB based on Heuristic Factor

Seog-hee Kang*, Jin-Young Choi
Graduate of School Computer and Information Technology,
Korea University

요 약

DBMS상의 인덱싱 기법은 대부분 B-트리 또는 B-트리의 변형인 T-트리 등을 주로 사용하여 왔다. 이는 트리 구성에서 최적의 균형을 유지하여 자료검색 및 저장 공간에 효율적인 방안으로 인식되어 왔으며 대부분의 범용 DBMS에서 탁월한 성능을 보여 왔다. 그러나 고성능의 효율을 요구하는 OLTP[1] 분야 중 특히 증권시장에서는 이런 일반적인 인덱싱 방법보다는 그 분야 특성에 맞는 새로운 인덱싱 방안의 적용이 성능 향상과 더불어 시스템의 최적의 효율을 얻어낼 수 있다. 본 논문에서는 증권시장용 원장(ledger)[2] DB의 새로운 인덱싱 방안으로서, 먼저 원장 DB의 접근 형태를 이해하고 경험적인 데이터를 기준으로 산출한 가중치를 이용, 새로운 인덱싱 방안을 제시하고 기존 인덱싱 방법과의 비교를 통한 성능향상의 효율성을 보여준다.

1. 서론

현재 우리는 많은 양의 자료를 보관하고 있는 데이터베이스(Database: DB)에서 자료를 효과적으로 검색하기 위하여 많은 노력을 기울여 왔다. 특히 메모리 가격 하락, 메모리 DB 지원을 위한 인덱스 구조 개발 등으로 메모리 기반 DB의 기술이 제시되어 그 검색성능은 예전 디스크 기반보다 월등히 향상되었다 [1][2]. 그럼에도 불구하고 많은 양의 데이터가 범용시스템 내에서 유지, 관리되는 경우는 항상 검색시간이 느려짐에 따라 많은 문제점으로 나타났고, 증권시장 시스템도 예외는 아니었다. 이에 증권시장용 계정원장 데이터베이스에서 요구되는 최고의 검색성능을 충족하기 위하여 그 데이터베이스에 적합한 인덱싱 기법을 제안하여 더 좋은 성능을 기대하고자 한다.

이를 위하여 먼저, 증권시장에서 적합한 데이터 적재 방안과 자료들의 검색 패턴을 분석한다. 과거 2005년 1년간 총 250여 일 동안 발생한 증권시장의 자료 6,273,560건을 수집하였으며, 데이터 접근 형태와 반복 사용비율, 재활용성 등을 분석하여 계정 원장 DB (account ledger DB)의 데이터 특성을 파악하고 고성능을 위한 새로운 형태의 인덱싱 방안을 제시한 후 각각의 성능을 비교하였다.

2. 관련 연구

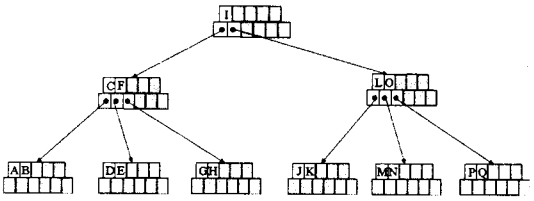
2.1 인덱싱

인덱싱이란 저장된 자료를 DB화하여 검색 시간을 단축시키는 방법으로 데이터베이스의 필수 요소이다. 성능 향상을 위해 저장 공간 절약과 탐색 시 디스크 접근 회수 감소를 목적으로 많은 연구들이 진행되었으며 인덱스의 저장 구조에 대해서는 B-트리 B^*-트리, B^+-트리 등 많은 구조가 제시되었다 [3][4][5].

2.2 전형적 B-트리의 인덱싱

1) Online Transaction Processing
2) 고객원장을 의미

B-트리의 알고리즘 복잡도는 $O(nlogn)$으로 다른 검색 알고리즘에 비하여 좋은 성능을 발휘하며 균형 잡힌 트리 구조를 이루는 경우 검색과 저장 공간 사용이 효율적이기 때문이다 [6]. 이러한 B-트리 인덱싱 기법을 사용하는 일반적인 DB는 현재 상용 DBMS의 주종을 이루고 있다. 다음은 B-트리의 예시이다.



(그림 1) 'A'부터 'Q'까지 삽입한 B-트리 예시

그림 1에서 'A'부터 'Q'까지 순차적으로 삽입할 경우 높이가 3인 B-트리가 만들어진다. 이 B-트리는 데이터가 삽입 될 때마다 각 노드의 균형을 유지하며 분할이 되므로 노드가 포함하고 있는 데이터와 무관하게 일정한 규칙으로 트리 전체가 구성된다.

3. 증권시장 원장 DB의 데이터 사용 형태 분석

본 논문에서는 일반적으로 범용 DB를 구축하는 사례와는 달리 증권시장에서 사용되는 여러 가지 DB 자료 중 증권회사 시스템의 마스터 성격의 DB인 원장(ledger) DB를 분석하였다. 오프라인 처리 위주의 증권회사 지점 A와 온라인 처리 위주의 증권회사 지점[3] B로 나누어 2가지 그룹의 2005년도 249일의 증권영업일[4] 데이터를 가지고 특성을 분석하였다.

<표 1>은 총 영업일에 대하여 사용된 레코드의 비율을 나타내는데, 사용 비율이 95%라 함은 특정 레코드가 영업일 249일 중 236일 동안 지속적으로 사용되었음을 의미한다. 이 표에서 알 수 있듯이 총 원장 DB 레코드 절반 정도는 한 번도 사용되지 않았으며, 사용일이 50% 이상인 레코드 수는 전체 레코드의 10% 정도에 그치는 것을 알 수 있어, 상위 10%이하의 레코드만이 자주 사용 된다는 것을 알 수 있다.

<표 1> 레코드별 총 영업일 사용 비율

사용 비율	A 그룹		B 그룹	
	레코드 수	누적비율	레코드 수	누적비율
100%	6	0.13%	130	0.71%
95%~	8	0.29%	125	1.40%
90%~	10	0.50%	136	2.14%
85%~	8	0.67%	185	3.15%
80%~	18	1.04%	172	4.10%
75%~	14	1.34%	140	4.86%
70%~	17	1.69%	177	5.83%
65%~	19	2.09%	187	6.85%
60%~	18	2.46%	200	7.95%
55%~	33	3.15%	245	9.29%
50%~	18	3.53%	225	10.52%
45%~	31	4.17%	300	12.16%
40%~	24	4.67%	270	13.64%
35%~	59	5.91%	294	15.25%
30%~	45	6.84%	380	17.33%
25%~	86	8.64%	446	19.77%
20%~	97	10.66%	517	22.60%
15%~	164	14.09%	704	26.46%
10%~	223	18.74%	1,066	32.29%
5%~	904	37.60%	2,891	48.12%
0%~	2,990	100.00%	9,476	100.00%
합계	4,792		18,266	

이러한 특성은 월별 통계에서도 나타나는데 2005년도 월별 사용 레코드를 보면 A그룹과 B그룹이 각각 최대 6%와 15% 미만임을 알 수 있다 <표 2>.

<표 2> 월별 레코드 건수 대비 사용 된 레코드 건수

월	A 그룹			B 그룹		
	Access 된 레코드 수	DB의 총 레코드 수	사용 비율	Access된 레코드 수	DB의 총 레코드 수	사용 비율
1	157	3,588	4.38%	1,261	9,150	13.78%
2	183	3,640	5.03%	1,383	9,512	14.54%
3	170	3,722	4.57%	1,322	9,885	13.38%
4	150	3,794	3.95%	1,189	10,199	11.66%
5	146	3,853	3.78%	1,223	10,528	11.61%
6	170	3,917	4.33%	1,476	11,806	12.50%
7	185	4,014	4.60%	1,596	12,607	12.66%
8	168	4,183	4.01%	1,537	13,059	11.77%
9	188	4,281	4.38%	1,631	13,537	12.05%
10	205	4,410	4.64%	1,880	14,460	13.00%
11	225	4,513	4.99%	2,051	16,040	12.79%
12	252	4,689	5.37%	2,203	17,717	12.44%

결국 <표 1>과 <표 2>에서 마스터 DB를 차지하고 있는 대다수의 레코드에서 단지 5~15%의 레코드들만이 사용되고 있음을 알 수 있다 (규칙 1).

<표 3>은 영업일 기준 과거 5일간 사용되었던 레코드가 당일 다시 사용되는 비율을 보여준다. A그룹의 자료에서는 약 50% 수준의 레코드들이 D-1 부터 D-5 까지 다시 사용되고 있고, B그룹은 70% 수준임을 알 수 있다. D합은 D일에 사용된 레코드를 D-1부터 D-5 중 하루라도 사용된 경우를 말하는데 놀랍게도 A그룹은 80%이상, B그룹은 대략 90%이상이라는 수치가 나온다 <표 3>.

3) 온라인 증권사의 지점은 주로 은행과 연계된 가상의 지점임

4) 주식거래가 가능한 일자. 국공휴일 등 영업을 하지 않는 휴일을 제외한 월~금요일을 말함

<표 3> 1년간 월 평균 재사용비중 (%)

월		1	2	3	4	5	6	7	8	9	10	11	12
A 그룹	D-1	61	62	60	60	58	62	61	57	63	59	59	59
	D-2	58	59	57	55	57	59	57	54	59	56	56	58
	D-3	54	56	55	54	53	57	54	52	57	54	53	56
	D-4	53	55	54	52	52	55	53	51	55	53	52	55
	D-5	51	54	54	51	51	54	53	49	52	52	50	54
	D합	84	86	86	84	83	85	91	85	84	84	84	84
B 그룹	D-1	74	73	73	71	73	74	76	75	75	74	75	74
	D-2	72	71	71	69	70	71	73	72	72	72	72	71
	D-3	69	69	69	67	68	70	71	71	70	70	70	70
	D-4	68	67	68	66	66	68	69	69	69	69	69	69
	D-5	66	66	67	65	64	66	68	68	67	67	67	68
	D합	92	92	92	91	91	91	92	92	92	92	92	91

그 결과는 한번 사용된 레코드는 다음 영업일에 다시 사용될 확률이 높으며, 이전 영업일의 간격이 크면 건수가 감소하고 직전 영업일에 가까울수록 높다는 것과 총 5일 동안의 기준으로 볼 때 익일 사용될 레코드는 이전기록의 80%이상을 예측할 수 있다는 것을 알 수 있다 (규칙 2).

규칙 1. 마스터 DB의 레코드 전체가 골고루 사용되지 아니하고 일부분 레코드만이 자주 사용된다.

규칙 2. 한번 사용된 레코드는 당일 혹은 익일 또는 인접 영업일에 다시 사용될 확률이 높다.

4 경험적 비균형 인덱싱 (Heuristic Un-Balanced Indexing)

4.1 개념

인덱스 검색 시 최적의 효율을 위해서는 B-트리상의 검색 깊이를 줄여 노드 순회 시간을 줄이는 것이 전체적인 응답 시간을 감소시킬 수 있는 하나의 요소가 될 것이다. 따라서 DB 적재 시 생성되는 B-트리 인덱스를 인위적으로 위치를 지정하고, 빈번히 사용되는 데이터를 검색 깊이가 낮은 루트 근처에 위치시키고 거의 사용되지 않는 데이터를 단말 노드에 위치하도록 구성한다면, 매번 발생하는 레코드 접근 요청마다 최고의 성능을 발휘할 수 있을 것으로 예상할 수 있는 것이다.

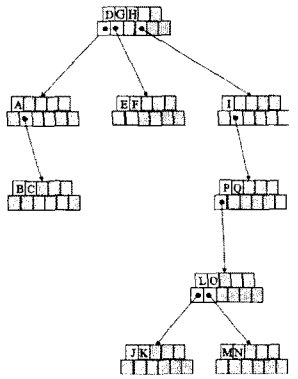
앞에서 예시한 (그림 1)에 대하여 각각의 레코드에 대하여 사용 건수 정보가 있다면 그 B-트리는 (그림 2)처럼 변경될 것이고 검색 시 깊이가 낮아져 좀 더 나은 효율을 기대할 수 있다.

본 논문에서는 B-트리를 비교 대상으로 선택하였는데 이는 B-트리가 메모리 기반에서는 모든 데이터를 단말 노드에 배치할 필요가 없으므로 공간낭비가 적기 때문에 B+트리보다 효율적이기 때문이다. 따라서 B-트리로 인덱싱 한 경우와 가중치를 고려한 B-트리 또한 그 응용한 알고리즘 등 다음의 몇 가지를 각각 비교하였다.

- 일반적인 B-트리 인덱싱의 경우(Alg 1)
  - 전체 레코드를 인덱싱 한 경우
- 경험적인 B-트리 인덱싱의 경우 (Alg 2)
  - 전일 사용된 레코드만으로 인덱싱하고, 추가되는 레코드는 일반적인 B-트리로 구성
- 준 가중치 기반의 비균형 트리 인덱싱의 경우 (Alg 3)
  - 최초 설정은 가중치에 의해 지정된 인덱싱 하고, 추가되는 레코드는 일반적인 B-트리로 구성
- 가중치 기반의 비균형 트리 인덱싱의 경우 (Alg 4)
  - 최초 설정과 추가 모두 가중치에 의해 인덱싱하는 경우로 매 검색 시마다 가중치변화에 의해 인덱스 트리가 변경되는 형태

아울러서 효율을 평가하는 요소는 노드 검색에 필요한 깊이 순회의 회수로 정하며 시간 개념을 적용하지 아니하였다.

	접근	순위
A	175	7
B	130	11
C	141	9
D	200	2
E	180	5
F	182	4
G	211	1
H	199	3
I	179	6
J	53	16
K	49	17
L	113	12
M	70	14
N	59	15
O	100	13
P	138	10
Q	147	8



(그림 2) 과거 레코드 사용 결과를 기반 한 B-트리

4.2 경험치와 가중치

본 논문에서는 이전에 사용된 경험이 있는 일별 레코드는 어느 정도의 비중이 있으며, 과연 얼마동안의 자료를 바탕으로 산출되어야 하는 것인가라는 문제가 있는데 이는 영업일 기준 D-5일을 선정하고 각각의 영업일은 피보나치수열을 값을 이용하여 직전일부터 5, 3, 2, 1, 1의 가중치를 적용하여 우선순위를 결정하도록 수치화 하였다.

	D-5(1)	D-4(1)	D-3(2)	D-2(3)	D-1(5)	가중치
A 레코드	12	6	31	0	0	80
B 레코드	5	3	6	8	9	89

D-5일보다는 직전일일수록 가중치가 커지며, 직전일 사용된 건수가 크면 클수록 가중치는 더욱 커지게

된다. 여기서 산출한 가중치는 각 레코드를 DB에 적재할 때 그 노드 위치를 결정하게 되는 요인으로 사용된다.

4.3 비균형 트리 (Un-Balanced Tree)

앞 절에서 언급한 가중치를 기준으로 수치가 큰 데이터를 루트 노드 또는 그 근처 노드에 위치하도록 하여 검색 깊이를 최소화토록 구현하였으므로 균형을 상실하게 되어 비균형 트리를 구성하게 된다.

5. 성능 비교 및 결론

5.1 성능 비교

각 알고리즘별 시뮬레이션의 통계자료에 대한 환경 및 기준은 다음과 같다.

입력	A, B그룹의 일일 검색 건수 (724,029 + 5,549,531건)
출력	일일 평균 깊이, 노드 수, 노드당 데이터 수
측정방법	최초 적재를 위해 삽입한 후 검색 시의 깊이 순회 합을 산출
판단기준	검색 시 요구 최소 평균 깊이
노드 당 최대 데이터 수	5

<표 4>에서 총 노드 수는 당연히 [알고리즘 1]의 경우 가장 크게 나타났다.

<표 4> 월 평균 총 적재 노드 수

월	A 그룹				B 그룹			
	Alg 1	Alg 2	Alg 3	Alg 4	Alg 1	Alg 2	Alg 3	Alg 4
1	1,780.76	78.55	55.20	58.25	4,567.95	634.15	465.95	481.35
2	1,805.06	91.06	68.35	71.00	4,748.41	698.00	505.18	511.59
3	1,842.82	86.05	63.32	64.82	4,935.55	672.09	486.14	492.14
4	1,872.10	76.10	54.35	55.45	5,092.55	603.10	436.90	441.25
5	1,889.62	75.10	53.05	53.81	5,256.62	612.05	449.62	452.62
6	1,909.38	83.76	61.67	62.95	5,895.95	753.48	540.43	537.90
7	1,953.57	92.76	66.71	68.29	6,295.57	797.76	580.29	622.81
8	2,036.73	87.32	60.91	63.23	6,521.50	780.09	563.32	575.91
9	2,084.62	92.86	70.24	72.57	6,760.33	814.24	602.71	607.71
10	2,147.25	106.15	75.50	77.60	7,221.85	956.95	688.50	692.15
11	2,197.27	115.32	83.23	85.50	8,012.18	1036.36	745.59	753.95
12	2,282.62	132.43	91.67	94.57	8,851.48	1126.19	801.90	811.19

<표 5> 월별 검색 시 평균 깊이

월	A 그룹				B 그룹			
	Alg 1	Alg 2	Alg 3	Alg 4	Alg 1	Alg 2	Alg 3	Alg 4
1	5.56	2.73	2.26	1.85	6.53	4.55	3.64	3.15
2	5.64	2.82	2.47	1.94	6.51	4.53	3.58	3.20
3	5.61	2.80	2.05	1.88	6.51	4.55	3.52	3.10
4	5.57	2.64	1.91	1.61	6.54	4.56	3.37	3.01
5	5.66	2.74	1.99	1.70	6.55	4.55	3.35	2.95
6	5.68	2.69	2.09	1.83	6.51	4.59	3.47	3.11
7	5.64	2.79	2.44	2.07	6.52	4.64	4.50	3.79
8	5.71	2.75	2.10	1.74	6.48	4.56	3.60	3.16
9	5.70	2.85	2.30	1.98	6.52	4.55	3.54	3.07
10	5.63	3.20	2.20	1.84	6.53	5.29	3.70	3.19
11	5.67	3.56	2.32	1.81	6.94	5.50	3.54	3.08
12	5.70	3.69	2.41	1.92	7.57	5.55	3.67	3.30

성능을 나타내는 자료로 월별 검색 시 평균 깊이는 <표 5>와 같다. [알고리즘 1]에 비하여 [알고리즘 2, 3, 4]가 2배 이상 향상되었음을 알 수 있고 [알고리즘 3, 4]는 일반적인 B-트리 사용의 경우보다 좀 더 성능이 좋아졌음을 알 수 있다. 다만 [알고리즘 4]가 [알고리즘 3]의 경우보다 더욱 향상되었지만 추가 비용[5]을 고려하지 않았으므로 단순히 [알고리즘 3]과 [알고리즘 4]를 비교할 수는 없다.

5.2 결론

이로써, 일반적인 B-트리의 인덱싱에 비하여 새로 제시한 [알고리즘 2, 3, 4]가 우월함을 살펴보았다. 본 논문의 방안은 다소 추가 메모리 저장 공간을[6] 요구하고 추가적인 사전작업이 선행되어야 하지만 성능 향상 요구 측면을 고려한다면 효과적인 방안이라 할 수 있다. 더욱이 예전에 메모리의 가격 대비 효율 면에서 취약했던 때와는 달리, 성능이 강조되는 시점이며 기술발전으로 가격이 디스크와 비교해 메모리가 경쟁력 있게 자리를 잡아가고 있으므로 더욱 그러하겠다. 다만, 경험치와 가중치를 결정하는 효과적인 연구 또한 좀 더 많이 이루어져야 하겠다.

참고문헌

[1] The TimesTen Team, "In-Memory Data Management for Consumer Transactions The TimesTen Approach" , In Proceedings of the 1999 ACM SIGMOD international conference on Management of data, pp.528-529, 1999.

[2] 이용욱, "대용량 DB의 고성능 OLTP지원을 고려한 기존 DBMS의 비교연구", 데이터베이스연구회 KDBC(SIGDB-KISS), pp. 2-5, 2004.

[3] R. Bayer. Symmetric binary b-trees: data structure and maintenance algorithms. Acta Informatica, 1, pp. 290-306, 1972.

[4] Knuth, D, The art of Computer Programming, Vol 3 : sorting and Searching, Addison-Wesley, Reading, Massachusetts, 1973.

[5] M. Blasgen, R. Casey, and K. Eswaran. "An Encoding Method for Multi-field Sorting and Indexing". Communications of ACM, 20(11):846-876, Nov. 1977.

[6] Douglas Comer, "Ubiquitous B-Tree", ACM Computing Surveys, Vol. 11, Issue 2, pp. 121-137, 1979.

5) 실시간 가중치계산에 의한 인덱스트리 변경으로 발생되는 오버헤드시간을 말함

6) 경험치와 가중치의 사전 계산에 의한 정렬 작업 공간, 노드당 가중치 관리용 추가 메모리 등