데이터베이스 갱신을 위한 스케줄링 알고리즘에 관한 연구

박희숙*

A Study on Scheduling Algorithm for Refreshing Database

Hee-Sook Park*

요 약

대규모 데이터베이스 시스템에는 다양한 종류의 데이터들이 공존하고 있으며, 사용자에게 정확하고 유용한 정보를 제공하기 위해 데이터의 신선도를 유지하는 문제는 중요한 이슈로 등장하고 있다. 대부분의 이런 문제의 해결책은 적절한 시간내에 요청된 갱신질의어를 얼마나 효율적으로 수행하는가 하는 것이 관건이다. 본 논문에서는 데이터의 신선도 유지와 기아상태의 공정성을 유지하기 위한 갱신 스케줄링 알고리즘을 제안한다. 제안된 알고리즘은 요청된 갱신질의어 실행시간 할당을 위해 목표 갱신 비율을 일정한 스케줄링 주기마다 재계산한다. 본 논문에서 제안된 알고리즘을 사용함에 따라 데이터들의 공정성과 신선도를 유지할 수 있다. 동적, 반-동적, 정적 데이터의 갱신처리 결과를 보여주기 위해 통합 웹사이트를 구현하였다.

ABSTRACT

There are coexisting various kinds of data in the large scale database system, the maintenance problem of freshness of data is emerging important issue that provide correctness and usefulness information to users. Most solution of this problem depends on how execute effectively required refreshing query within timely time. In this paper, we propose the refreshing scheduling algorithm to retain the freshness of data and fairness of starvation of requested refresh queries. Our algorithm recompute a rate of goal refreshing a every period to assign execution time of requested refreshing query so that we can keep the freshness and fairness of data by using proposed algorithm. We implement the web sites to showing the results of refreshing process of dynamic and semi-dynamic and static data.

키워드

Semi-dynamic data, Dynamic data, Static data, Refresh scheduling algorithm

* 부경대학교 전자컴퓨터정보통신공학부

접수일자 2008. 11. 03

Ⅰ. 서 론

대규모 데이터베이스에는 다양한 형태의 데이터들이 저장되어 있으며, 이들은 여러 가지 문서의 형태로 사용자들에게 유용한 정보를 제공한다. 사용자에게 제공되는 문서의 형태는 일반적으로 가장 큰 네트워크인 인터넷을 통하여 HTML문서 혹은 XML문서의 형태로 제공 될 수 있다. 이때 이들 정보를 제공하는 근원이 되는 데이터베이스 내부에 저장되어 있는 데이터의 형태로는 정적 데이터(Static data), 반-동적 데이터(Semi-dynamic data), 동적 데이터(Dynamic Data)등의 그룹으로 구별될 수 있다. 여기서 데이터의 구별은 데이터의 갱신 주기에 따라 구별되어진다. 정적 데이터는 데이터가 처음 저장된 이후 시간이 지나도 그 내용이 변하지 않거나 아주 드물게 그 내용이 변화하는 것을 말한다. 반-동적 데이터란 임의의 특정 시간에 사용자들에 의한 데이터의 수정요구가 발생하는 경우만 변경되는 데이터를 말한다. 마지막으로 동적 데이터란 짧은 시간 주기 동안 계속해서 데이터의 변화가 일어나는 것을 말한다. 사용자들은 언제나 자신들에게 정확한 최신의 정보가 제공되기를 원한다. 따라서 최신의 정보를 데이터베이스 내부에 유지하는 것은 중요한 문제가 된다. 데이터베이스 내부에 최신의 정보가 유지되기 위해서는 데이터의 신선도(Freshness)를 유지하는 것이 가장 중요하다. 이러한 신선도 유지 문제는 적절한 갱신 알고리즘을 통하여 이루어진다. 시간의 경과와 오버로드 조건하에서 기존[1][2]에 제안된 알고리즘들은 갱신 요청들에 대하여 잦은 갱신 miss를 야기하기 때문에 데이터들에 대한 적절한 신선도를 유지하기가 어렵다. 따라서 본 논문에서는 데이터의 신선도 유지와 데이터 갱신 작업에 있어서 기아상태(Starvation)의 공정성(Fairness)을 유지하기 위한 방법으로 목표 갱신비율을 일정한 스케줄링 주기마다 재계산하여 요청된 갱신 질의어 실행시간을 할당하는 갱신 알고리즘을 제안하고 이를 통하여 데이터의 신선도 유지문제의 성능을 개선하고자 한다. 또한 갱신 알고리즘의 성능평가 및 데이터들의 갱신 결과를 사용자에 제공하기 위한 방법으로 통합 웹사이트를 구현한다.

본 논문은 다음과 같이 구성된다. 2장의 관련연구에서는 기존의 갱신 스케줄링 알고리즘 대하여 간략하게 소개하고, 3장에서는 본 논문에서 제안하고 있는 갱신 스케줄링 알고리즘에 대하여 기술하며, 4장에서 실험결과에 대하여 설명하고 마지막 5장에서는 결론을 논의한다.

Ⅱ. 관련연구

본장에서는 기존의 제안된 2가지 갱신 스케줄링 알고리즘인 EDF(Earliest Deadline First)스케줄링 알고리즘과 MSF(Most Starved First)스케줄링 알고리즘에 대하여 소개한다.

EDF 스케줄링 알고리즘은 다양한 범위의 실시간 시스템들에 적용되어지며 기존 연구보고서들에서 다른 측정기준들과 관련하여 최적화된다는 것을 볼 수 있다. 그러나 오버로드 조건에서 그것은 매우 열악하게 수행되었다. 그 이유는 EDF가 각각의 작업 데이터들에게 가장 높은 우선순위를 부여하는 방식을 사용하기 때문이며 이것은 그들의 데드라인(deadline)을 miss하는 것에 가까운 것이 되기 때문이다. 만약 첫 번째 데이터가 데드라인을 miss하게 된다면 연속적으로 데이터들이 데드라인을 miss하게 되어 도미노 효과를 야기한다. 이것은 대부분의 갱신요청에서 데이터를 갱신하지 못하는 것이 되기 때문에 낮은 신선도를 얻게 된다[1][2].

MSF 스케줄링 알고리즘의 주요 목표는 질의어 집합들이 데이터를 갱신하기 위하여 갱신작업의 공정성을 최대화하는 것이다. 즉, MSF는 가능한 miss한 갱신요청들의 합을 낮게 유지하는 것을 목표로 한다. 갱신 요청들은 다른 시점에서 각각 PENDING, FUTURE, SATISFIED, MISSED등 4가지 상태를 가지며, 만약 갱신요청에 대한 시작 시간이 현재 시간보다 늦으면 요청상태는 FUTURE가 되고, 요청 시작 시간이 현재 시간보다 빠르고 다음 요청시간이 현재 시간보다 늦으면 PENDING이 된다. 만약 갱신 질의어 실행시간이 데드라인 이전에 완전히 종료된다면 SATISFIED상태, 그렇지 않으면 MISSED상태가 된다[1]. MSF 스케줄링 알고리즘은 기아상태의 균등한 유지로 갱신작업의 공정성은 높게 유지되지만 갱신 주기가 길어질수록 데이터베이스의 신선도가 현저히 저하된다. 따라서 본 논문에서는 MSF 스케줄링 알고리즘을 개선한 것으로 갱신 요청비율에 따라 갱신을 수행하여 갱신의 공정성과 모든 데

이터의 신선도를 최적으로 유지 가능한 개선된 갱신 알고리즘을 제안하고자 한다.

Ⅲ. 시스템 구성 모델과 알고리즘 제안

3.1 시스템의 구성요소

본 논문에서 제안한 알고리즘의 구현에 사용된 시스템의 논리적 구성도는 그림 1과 같다.

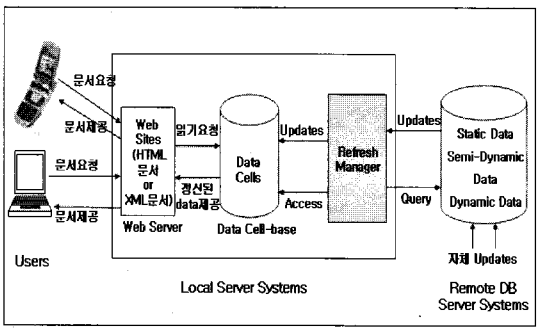

그림 1. 시스템의 논리적 구성도
Fig. 1 Logical architecture of Systems

시스템의 구성은 사용자부, 로컬 서버 시스템부, 원격 데이터베이스 서버 시스템부로 구성된다.

사용자부는 유선 또는 무선방식의 서버 접속을 통하여 갱신된 데이터들을 포함하는 문서들을 제공 받는다. 제공 받는 문서는 HTML문서 또는 XML문서의 형태가 된다.

로컬 서버 시스템부의 구성 요소들은 다음과 같다

① 웹서버(Web Server) : 웹서버는 많은 웹사이트들로 구성되어 있으며, 각 웹사이트들은 정적 데이터, 반-동적 데이터, 동적 데이터들을 포함하는 웹 페이지들 또는 XML문서들을 포함하고 있다. 이들 문서들은 사용자의 요청에 의해 사용자들에게 제공이 된다. 또한 이들 문서의 내용 일부는 데이터 셀- 베이스에 제공된 정보들을 포함하게 된다.

② 데이터 셀-베이스(Data Cell-base) : 데이터 셀- 베이스는 데이터 셀들의 집합으로 구성 되며 이들은 테이블의 형태로 구현한다. 테이블의 구성 요소는 {Id, Refresh_period, Last_refresh_time, Refresh_request_count, Real_refresh_count}이다. 여기서 Id는 셀의 식별자, Refresh_period는 갱신 주기, Last_refresh_time은 갱신질의 실행시간, Refresh_request_count는 갱신 요청 횟수, Real_refresh_count은 실제 갱신횟수를 나타낸다.

각 웹사이트의 문서들은 문서의 내용으로 데이터 셀-베이스의 내용을 참조하기 때문에 데이터 셀-베이스는 캐시와 같은 역할을 할 수 있음을 의미한다.

③ 갱신관리자(Refresh Manager) : 갱신관리자는 데이터 셀들과 데이터베이스의 매핑 유지관리 설정 및 데이터 셀-베이스내에 구성될 데이터 셀들의 집합을 결정하는 역할을 한다. 또한 웹사이트의 데이터 갱신작업 수행 및 갱신 질의어의 수행을 위한 갱신 스케줄링 알고리즘을 수행해야 할 책임이 있다[1].

원격 데이터베이스 서버 시스템부는 관계형 데이터베이스 안에 저장된 테이블 또는 뷰의 형태로 구성한다. 그들의 내용은 응용프로그램들에 의해 자체 업데이트가 가능하다.

3.2 갱신 스케줄링 알고리즘의 제안

사용자에게 항상 유용한 정보를 제공하기 위해서는 데이터의 신선도 유지가 가장 중요하다. 따라서 데이터의 신선도 유지는 모든 갱신 요청이 발생했을 때 만족할 수 있도록 적절한 방법으로 갱신 질의어 집합의 갱신처리를 실행 할 수 있어야 한다. 이것은 갱신 스케줄링 알고리즘의 성능에 따라 차이가 난다. 데이터베이스에 대한 다양한 갱신 패턴들의 주기적 차이와 실행시간 때문에 몇몇 갱신 요청들은 갱신 실행을 miss하는 경우가 발생하게 된다. 이것은 갱신 요청들의 기아상태를 유발하게 되며, 이러한 기아상태는 데이터들의 신선도를 저하시키는 직접적인 원인으로 작용하게 된다. 본 논문에서는 이와 같은 기아 상태를 완화하기 위해 일정한 주기마다 예상 갱신비율(R)을 계산하고 이것을 기준으로 각 데이터의 갱신 비율과 비교하여 최소 갱신비율을 갖는 데이터를 우선적으로 갱신처리 함으로써 각 그룹별 데이터 갱신에 대한 공정성이 유지되도록 하는 알고리즘을 제안한다. 예상 갱신비율(R)의 계산은 다음과 같은 공식을 사용한다.

$$R = \frac{Re_c}{Rq_c} \quad (1)$$

식(1)에서 R은 데이터의 예상 갱신비율을, Re_c는 데이터의 실제 갱신 카운트를 Rq_c는 갱신 요청 카운트를 의미한다. 식(1)의 갱신 요청 카운트(Rq_c)는 다음과 같은 공식으로 계산한다.

$$Rq_c = \sum_{i=1}^{n} \frac{T}{P_i} \tag{2}$$

식(2)에서 T(갱신 관리자의 1회 스케줄링의 실행시간 * 스케줄링 횟수)는 정해진 시간을 의미하며 P_i는 각 데이터의 갱신주기를 나타낸다. 식(1)에서 사용된 실제 갱신 카운트(Re_c)의 계산은 다음과 같은 공식을 적용한다.

$$P_s = \sum_{i=1}^{n} P_i \tag{3}$$

$$E_t = ((P_s/P_i)/\sum_{k=1}^{n} \frac{P_s}{P_k})* T \tag{4}$$

$$R_c = E_t/Avg(Eq_i) \tag{5}$$

$$Re_c = Min(Rq_c, R_c) \tag{6}$$

위의 식에서 P_s는 데이터들의 갱신주기의 합을 P_i는 셀-베이스내에 저장된 각 셀의 갱신주기이다. E_t는 각 셀에 할당된 질의어 실행시간을 의미한다. Avg(Eq_i)는 예상 갱신 비율 결정 이후부터 각 셀의 질의어 실행시간 평균을 타나내며, R_c는 각 셀의 갱신 횟수를 나타낸다. 다음 그림 2는 각 데이터 셀들에 대한 갱신요청을 발생시키는 스케줄링 알고리즘을 표현한 것이다.

```
IF (Current_time-Last_refresh_time) < Refresh_period
  Cell_refresh_count=Cell_refresh_count+1
else  if refresh_ok == true
  Cell_refresh_count+=Refresh_request_count
 else
  Cell_refresh_count+=Refresh_request_count
  Last_refresh_time=
     Refresh_period*Refresh_request_count
end if
 end if
```

그림 2. 갱신 스케줄링 알고리즘
Fig. 2 Algorithm of refresh scheduling

본 논문에서 제안하는 갱신 스케줄링 알고리즘은 데이터의 갱신주기가 짧을수록 갱신처리의 실행 기회를 더 많이 할당 받고 갱신주가 길수록 실행 기회를 적게 할당 받게 된다. 따라서 기아상태가 특정형태의 데이터에 편중되지 않으므로 데이터 그룹별로 갱신비율에 대한 공정성과 적절한 신선도가 유지 될 수 있다. 특정 데이터의 갱신 요청이 실행된 후 데이터의 갱신 비율은 증가하여 갱신된 데이터의 갱신 우선순위는 결과적으로 낮아지게 된다.

Ⅳ. 실험결과

4.1 시스템의 구현

실험에 사용된 시스템 사양은 Sun Os 5.8® 운영체제와 웹서버로 Apache를 탑재한 Sun blade 1500® UltraSPARC III®, 메모리 512MB, CPU 속도 1-GHz, HDD 80GB로 구성된 하드웨어 시스템과 원격 데이터베이스 시스템으로 1대 Oracle® 10g와 3대의 My-SQL Database System을 사용하였다. 갱신관리자 구현을 위해 JAVA 프로그래밍언어를 사용하였으며, 갱신된 결과를 보여주기 위한 웹사이트 구현을 위해 JSP 프로그래밍 언어를 사용하였다. 데이터베이스 구현은 동적 데이터를 위해 주식정보를 반-동적 데이터를 위해 뉴스 속보를 정적 데이터를 위해 지역별 날씨정보를 사용하였다.

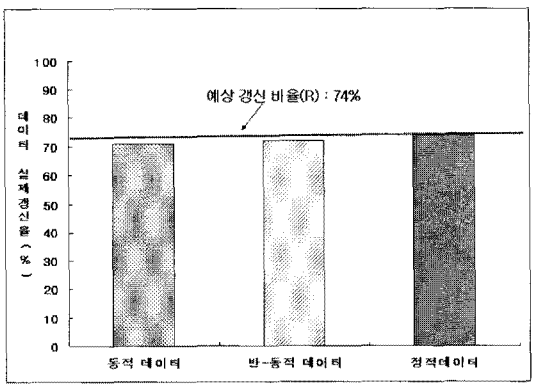

그림 3. 예상 갱신 비율에 따른 실제 갱신율
Fig. 3 Real refreshing rates by expected refresh ratio

그림 3은 제안된 갱신 스케줄링 알고리즘을 이용하여 실험한 결과로서 실제 갱신 비율이 예상갱신 비율에 근접함을 보여주고 있다.

실험 결과 기아상태 완화를 위해 예상 갱신 비율(R)을 74% 정도로 두었을 때 데이터의 신선도가 최대로 유지됨을 알 수 있었다. 표 1은 시간대별로 갱신요청에 따른 데이터별 갱신 공정성을 24시간 실험한 결과이다.

표 1. 시간대별 갱신비율 비교
Table 1. Refresh rates a time

시간(h) / 공정성(%)	3	6	9	12	15	18	21	24
동적 데이터	91.0	91.9	92.3	92.0	92.1	92.2	92.3	92.2
반-동적 데이터	95.2	95.3	95.4	95.3	95.2	95.2	95.2	95.3
정적 데이터	98.2	97.9	97.9	98.3	98.0	98.0	98.3	98.5

그림 4는 갱신처리 이전의 동적 데이터를 그림 5는 제안한 갱신 스케줄링 알고리즘을 사용하여 갱신 처리를 수행한 이후의 동적 데이터를 보여준 것이다.

그림 6과 그림 7은 반-동적 데이터의 갱신 이전과 갱신 이후의 결과를 보여준 것이다.

그림 4. 데이터 갱신 이전의 동적 데이터
Fig. 4 Dynamic data before data refreshing

그림 5. 데이터 갱신 이후의 동적 데이터
Fig. 5 Dynamic data after data refreshing

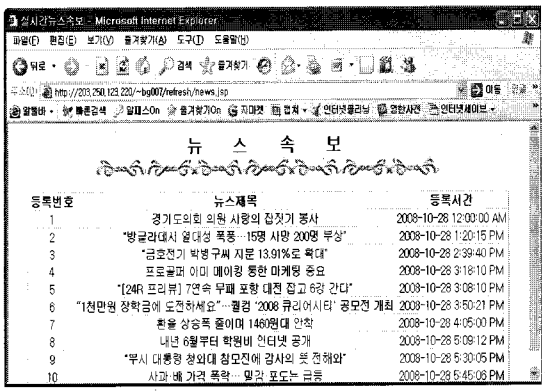

그림 6. 데이터 갱신 이전의 반-동적 데이터
Fig. 6 Semi-Dynamic data before data refreshing

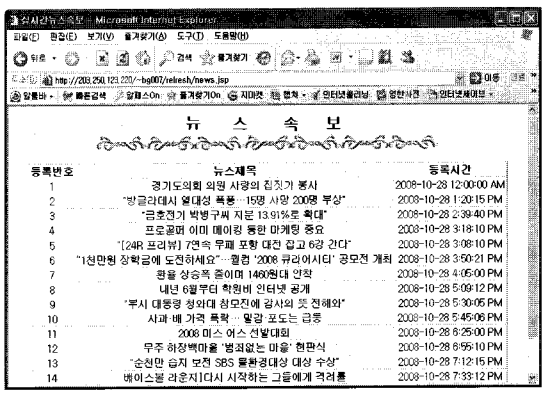

그림 7. 데이터 갱신 이후의 반-동적 데이터
Fig. 7 Semi-Dynamic data after data refreshing

그림 8과 그림 9는 정적 데이터의 갱신 이전과 갱신 이후의 결과를 보여준 것이다.

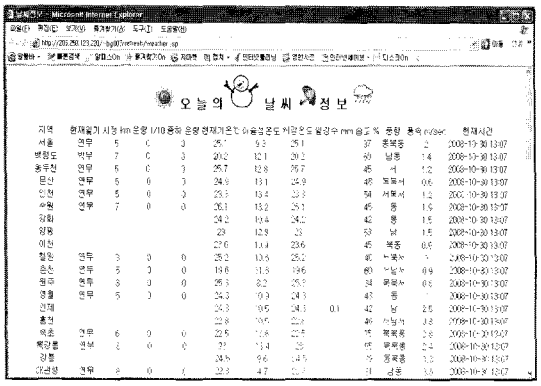

그림 8. 데이터 갱신 이전의 정적 데이터
Fig. 8 Static data before data refreshing

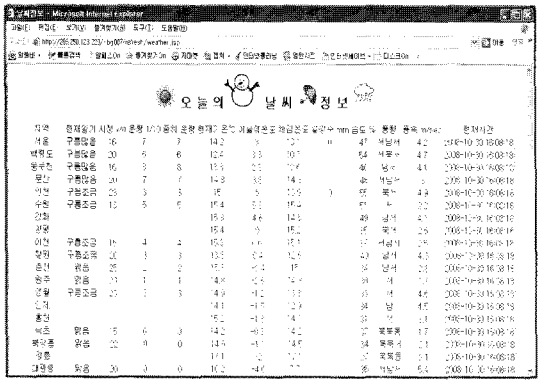

그림 9. 데이터 갱신 이후의 정적 데이터
Fig. 9 Static data after data refreshing

4.2 기존 방법과의 비교 분석

실험결과 기존의 MSF 스케줄링 알고리즘[1]은 항상 우선순위 할당 정책에 따라서 보류된 갱신요청들을 인출하여 수행하기 때문에 각 데이터들의 기아 상태의 횟수가 균등하게 유지되어 데이터 갱신비율의 공정성은 높게 유지할 수 있다. 그러나 갱신 주기가 길어질수록 MISSED되는 갱신 요청횟수가 늘어남에 따라 데이터의 신선도는 떨어지는 것으로 나타났다. 반면에 본 논문에서 제안한 갱신 스케줄링 알고리즘으로 갱신 처리를 수행하는 경우 각 요청 데이터들의 갱신은 예상갱신비율과 실행주기에 따라 갱신 작업을 수행하기 때문에 정적, 반-동적, 동적 데이터들에 대한 갱신 처리의 공정성과 모든 데이터들에 대한 신선도를 최적으로 유지할 수 있었다.

V. 결 론

사용자에게 정확하고 안정적인 정보를 제공하기 위해서 데이터의 신선도를 유지하는 것은 중요한 문제이다. 따라서 본 논문에서는 동적, 반-동적, 정적 데이터의 효율적인 갱신 작업을 수행하기 위한 방법으로 예상 갱신비율(R)을 주기적으로 재계산하여 갱신 작업에 반영하는 스케줄링 알고리즘을 제안하고 이를 구현하였다. 제안된 알고리즘을 이용하여 데이터의 갱신 작업을 수행함으로써 데이터의 갱신비율에 대한 공정성이 유지된다는 것을 보였다. 또한 기존 방법과 달리 갱신주기가 길어져도 세 가지 형태의 데이터들에 대한 최적의 신선도가 유지됨을 보였다.

참고문헌

[1] Haifeng Liu and Wee Keong Ng and Ee-Peng Lim, "Improving the Fairness of Timely Refresh of Web Views", Proceeding of the 7th International Conference on Database Systems for Advanced Applications (DASFAA 2001), April 2001

[2] K. Jeffay and D. Stanat and C. Martel, "On non-preemptive Scheduling of Periodic and Sporadic Tasks", In Procedings of th 12th IEEE Real-Time Systems Symposium, 1991

[3] Haifeng Liu and Wee Keong Ng and Ee-Peng Lim,"Keeping a Very Large Website Up-to-date: Some Feasibility Results", International Conference on Electronic Commerce and Web Technologies(EC-Web '2000), Greenwich, UK, September 2000

[4] Rob Peter, "Database Systems(8E)", Thomson, 2008

[5] Jon Kleinberg, "Algorithm Design", Addison Wesley, 2006

[6] H. M. Deitel and P. J. Deitel and T. R. Nieto and T. Lin, P. Sadhu, "XML How TO PROGRAM", Prentice Hall,

2000
[7] Elmasri, "Fundamentals of Database Systems 5/E", Addison-Wesley, 2007

저자소개

박희숙(Hee-Sook Park)

1995. 2. 한국방송통신대학교 전자계산학과(이학사)
1998. 2. 경남대학교 교육학과 (교육학석사)
2006. 2. 부경대학교 컴퓨터공학과(공학박사)
※관심분야 : 데이베이스 인덱싱 성능 개선 문제, 궤적 분할 인덱싱 기법, 객체 지향 데이터베이스, RFID와 데이터베이스 응용기술,