데이터베이스 IMPORT/EXPORT 도구 설계 및 구현

이종현, 홍승진, 김석현, 이상원
성균관대학교 컴퓨터공학

e-mail : ljh08375@hanmail.net
wow1090@naver.com
seokhyon@ece.skku.ac.kr
wonlee@ece.skku.ac.kr

Design and Development Import/Export Tools of DBMS

Jong-Hyun Lee, Seung-Jin Hong, Seok-Hyun Kim, Sang-Won Lee
Dept. of Computer Engineering, Sung-Kyun-Kwan University

요 약

DBMS 가 널리 사용되기 위해선 비용적인 측면과 서비스도 중요하지만, 성능 역시 중요하다. DBMS 의 성능에는 속도, 신뢰성, 기능성, 확장성, 가용성 등 여러 가지가 포함될 수 있지만 사용자가 직접 체감하기 쉽기 때문에 속도는 가용성과 더불어 성능의 중요한 부분 중 하나이다. 그리고 DBMS 의 속도는 사용자의 질의에 대한 응답시간뿐만 아니라 데이터를 백업하거나 데이터베이스를 import-export 하는 시간도 포함할 수 있다. 이에 본 논문에서는 CUBRID 의 DB import-export 에 대하여 속도 개선 방안을 제시하고, 도구를 설계 및 구현하여 기존의 import-export 와 처리속도를 비교 분석한다.

1. 서론

국내 기업 NHN 에서 DBMS 인 CUBRID 를 개발하여 오픈 소스로 배포하고 있다. CUBRID 는 국내 최대 규모의 인터넷 포털 업체인 NAVER 서비스에 적용되어 운영되고 있으며, 교육인적자원부를 비롯한 여러 공공기관에서 사용 중에 있다. 그러나 다양한 DBMS 가 존재하기 때문에 널리 사용되기 위해선 오픈 라이선스 같은 비용적인 측면과 서비스도 중요하지만, 성능 역시 매우 중요하다.

DBMS 의 성능에는 속도, 신뢰성, 기능성, 확장성, 가용성 등 여러 가지가 포함될 수 있다. 그러나 사용자가 직접 체감하기 쉽기 때문에 속도는 신뢰성과 더불어 성능의 중요한 부분 중 하나이다. 그리고 DBMS 의 속도는 사용자의 질의에 대한 응답시간뿐만 아니라 데이터를 백업하거나 데이터베이스를 import-export 하는 시간도 포함할 수 있다.

이에 본 논문에서는 CUBRID 의 DB import-export 에 대하여 속도 개선 방안을 제시하고, import-export 도구를 설계 및 구현하여 기존의 import-export 와 처리속도를 비교 분석한다.

본 논문의 구성은 다음과 같다. 2 장에서는 현재의 import-export 의 진행 과정과 속도 저하 요인에 대해 살펴보고, 3 장에서는 속도 개선 방안을 제시하며, 기존의 import-export 와 처리속도를 비교 분석하고 4 장에서 결론을 맺는다.

2. CUBRID 기존의 import 와 export 현황

데이터베이스를 import-export 할 때 속도를 결정 짓는 요인은 다양하지만 그 중 CUBRID 의 데이터 타입에 주목해 보겠다. 데이터 타입은 크게 문자열, 수치형, 날짜/시간, 집합형으로 분류 할 수 있다. 세부적으로 문자열은 알파벳부터 숫자, 한글, 특수문자 등 모든 문자를 말하고, 수치형은 정수형, 실수형, 부동소수점으로 나눠진다. 이처럼 다양한 데이터 타입이 있고 특성이 각각 다르지만 현재 CUBRID 2008 8.1.6 버전까지는 모두 string 형식으로 import-export 를 수행하고 있다. 그래서 문자열 형식을 제외한 다른 형식들은 모두 export 시 문자열로 변환하고 import 시 본래의 데이터 타입으로 다시 변환하는 과정이 요구되고, 이는 실행 속도의 감소와 시스템의 부하를 초래한다.

3. 개선된 import 와 export 툴

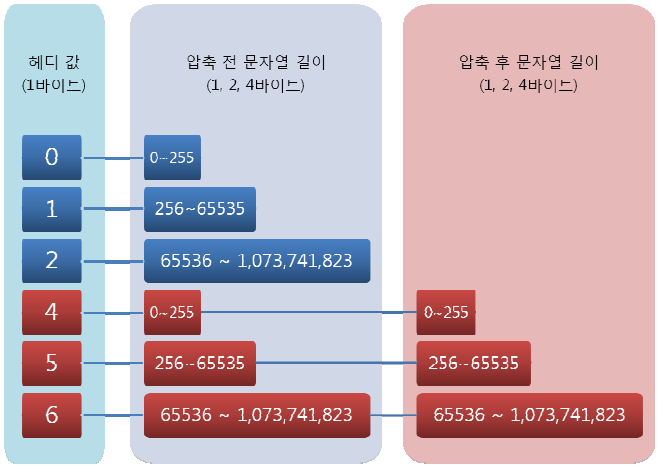
본 절에서는 2 장에서 말한 문자열 변환의 문제점을 해결하기 위해서 일괄적으로 string 형식으로 입출력 하는 것이 아닌 각 데이터 타입의 특성에 따라 좀

* 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT 연구센터 지원사업의 연구결과로 수행되었음 (NIPA-2009-(C1090-0902-0046))

더 효율적인 방식으로 import-export 를 수행하기 위한 방안을 모색한다.

3.1 구현 내용

import-export 과정에서 데이터베이스의 자료들을 처리하기 위해서는 필수적으로 디스크의 입출력을 해야 한다. 보조기억장치의 연산속도가 상대적으로 느리기 때문에 이 과정이 중앙처리장치나 주기억장치의 연산에 비해 상대적으로 월등히 많은 시간을 소모하게 된다.[2] 따라서 다소 연산이 많아지더라도 파일에 입출력 되는 데이터의 크기를 줄이는 것이 속도에 더 도움이 될 수 있다는 가정하에, 문자열은 압축하여 크기를 줄이고 (그림 1) 과 같이 헤더를 만들도록 한다.[3]



(그림 1) 문자열 헤더의 구성

integer 형은 4 바이트, short 형은 2 바이트의 공간을 고정적으로 할당하고 여기에 바이너리 형식으로 저장함으로써 공간을 절약할 것이다. 4 바이트의 integer 가 최대 4,294,967,296 까지 숫자를 표현할 수 있으므로 최대에 가까운 큰 숫자의 경우 10 바이트에서 4 바이트로 저장공간을 40%로 줄일 수 있다. 하지만 0~999 사이의 숫자의 경우에도 4 바이트를 할당하면 저장공간의 손실이 발생하므로 0~15 는 1 바이트, 16~255 는 2 바이트 등 숫자의 크기에 따라 가변적으로 할당하고 header 를 둬서 길이 정보를 저장한다. 날짜/시간 역시 integer 형의 데이터 타입으로 저장되므로 앞에서의 integer 와 같은 방식을 따른다.

3.2 속도 비교 분석

<표 1>의 수치형 DATA 에는 short, integer, float, double, numeric 데이터 타입의 다섯 개의 속성을 가진 테이블에 더미 데이터 100 만개를 넣고 export-import 한 결과를 넣었다. 그리고 문자열 DATA 에는 char(10) 과 varchar 데이터 타입의 두 개 속성을 가진 테이블에 더미 데이터 10 만개를 넣고 import-export 한 결과를 넣었다. 현실성 있는 테스트를 위해 varchar 에는 3000 자의 영문 뉴스기사를 넣었다.

<표 1>에서 수치형 DATA 의 경우 큰 수와 작은 수가 비슷한 비율로 분포 되어 있을 때 시간이 176% 빨라지고, 파일 크기도 기존의 2/3 로 줄어들었다. 문자열 DATA 의 경우 압축을 통해 용량은 줄어 들지만 lzo 압축 알고리즘 특성상 비싼 압축 비용으로 인해 export 시에는 성능 향상이 크지 않은 것을 볼 수 있었다. 하지만 lzo 압축 알고리즘은 압축의 해제가 빠르기 때문에 import 시에 export 보다 상대적으로 큰 시간 단축 효과를 볼 수 있었다.

<표 1> import-export 도구의 속도 변화

		기존 exportdb	exportdb	향상 비율
수치형 DATA	수행시간	11.359초	6.459초	약 176% 향상
	파일 크기	50,607,812byte	34,000,208byte	약 67%로 감소
		기존 importdb	importdb	향상 비율
	수행시간	55.536초	36.871초	약 151% 향상
		기존 exportdb	exportdb	향상 비율
문자열 DATA	수행시간	18.920초	18.562초	약 102% 향상
	파일 크기	326,500,201byte	217,628,798byte	약 66%로 단축
		기존 importdb	importdb	향상 비율
	수행시간	2분 7.916초	1분 38초	약 131% 향상

4. 결론

수치형 데이터들에 대해서는 바이너리 형식으로 저장하고 문자열은 압축하여 출력 파일의 크기를 감소시킨 결과, 테스트 결과에서 볼 수 있듯이 대체적으로 속도가 향상된 것을 알 수 있었다. 하지만 단순 문자의 반복이나, 모든 인스턴스에 한 자리 수만 들어 있는 경우 등 특수한 경우에 대해선 속도가 더 빨라질 수도 느려질 수도 있는데, 데이터의 길이나 패턴에 맞춰 파일 크기 단축의 비율이 달라지기 때문이었다.

결과적으로 크기나 패턴이 유사한 비율로 골고루 분포해 있는 데이터 들에 대해서는 성능의 향상이 일어났다. 하지만 데이터베이스는 다양한 환경이 있을 수 있다. 그래서 특수한 데이터베이스의 경우도 무시할 수 없을 것이므로 실제 프로그램에 적용하기 위해서는 사전에 충분한 조사와 테스트가 선행 되어야 할 것이다.

참고문헌

[1] Mark A. Roth , Scott J. Van Horn “Database compression”, ACM SIGMOD Record, 22:3,p.31-39, 1993
[2] Till Westmann , Donald Kossmann , Sven Helmer , Guido Moerkotte “The implementation and performance of compressed databases”, ACM SIGMOD Record, Vol.29 No.3, p.55-67, Sept. 2000
[3] Balakrishna R. Iyer, David Wilhite, "Data Compression Support in Databases", Proceedings of the 20th VLDB Conference, Santiago, Chile, 1994
[4] G.V. Cormack. "Data compression on a database system", Communication of the ACM, 28:12, 1985.