JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 21, No. 2 : 333~338 Feb. 2017

SQLite 모바일 데이터베이스의 검색 성능 분석을 위한 구현 및 실험

최진오*

Implementation and Experiment for Search Performance Analysis of SQLite Mobile Database

Jin-oh Choi*

Department of Embedded Software, Busan University of Foreign Studies, Busan 46234, Korea

요 약

최근 모바일 기기의 보편화로 인하여 모바일 데이터베이스를 필요로 하는 많은 새로운 모바일 응용들이 나타나고 있다. 대표적인 모바일 데이터베이스로는 SQLite, Realm 등이 있는데, 이들은 리소스 제약이 큰 소형기기에 적합한 데이터베이스 엔진들이며 리눅스 기반 모바일 기기에 많이 사용되고 있다. 이 엔진들은 대부분 오픈소스 프로그램들이며 범용 데이터베이스에 비해 가볍고 속도가 빠른 장점을 지닌다. 이 논문에서는 모바일 데이터베이스의 선두 주자인 SQLite의 성능 실험과 분석에 초점을 맞춘다. 대상인 모바일 환경을 고려하여 리눅스 기반 환경에서 실험한다. 그리고, SQLite 데이터베이스의 검색 성능을 파악하기 위한 테스트 프로그램을 구현하고 성능 실험을 실시한다. 실험은 같은 환경에서 실행되는 Oracle 데이터베이스와 비교하여 진행한다.

ABSTRACT

Recently, because of the mobile device generalization tendency, a lot of new mobile applications which need mobile database on mobile devices appear. SQLite, Realm, etc. are representative mobile databases, and they are the database engines which are suitable for small devices that have large resource restriction and are used a lot in mobile devices based on Linux. These engines are open source programs and have advantages which are lighter weight and faster speed than general purpose databases. This paper focuses on the performance experiments and analysis of SQLite, which is front runner in mobile database fields. Considering the target mobile environments, the test is performed at Linux environment. And this paper implements performance test program and performs performance experiment of SQLite database to test search efficiency. The experiment is processed by comparison with the results of a Oracle database working out at the same environment.

키워드 : SQLite, 모바일 데이터베이스, 성능 테스트, 리눅스

Key word : SQLite, Mobile Database, Performance Test, Linux

Received 08 December 2016, **Revised** 09 December 2016, **Accepted** 17 December 2016

* **Corresponding Author** Jin-Oh Choi(E-mail:jochoi@bufs.ac.kr, Tel:+82-51-509-6245)
Department of Embedded Software, Busan University of Foreign Studies, Busan 609-815, Korea

Open Access http://doi.org/10.6109/jkiice.2017.21.2.333 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

모바일 장치는 사용의 보편화와 함께 장치의 기술발전과 가격 하락으로 인한 대용량화가 급속히 진행되고 있다. 따라서 비약적으로 모바일 데이터베이스의 필요성이 커지고 있으며 모바일 기기에서 처리하는 데이터베이스 응용이 점점 많아지고 있다.

모바일 데이터베이스의 활용이 보편화되는 사례로 먼저 최근 라즈베리 파이(Raspberry Pi)나 아두이노(Arduino)를 이용한 데이터베이스 기반 서버 구축이 많이 시도되고 있다[1].

또한 모바일 장치의 센서 기술이 발전하여 모바일 장치로 수집되는 센싱 데이터를 로컬 데이터베이스에 저장하였다가 추후 마스터 데이터베이스와 동기화하는 기술이 등장하고 있다. 즉, mobile edge computing[2] 기술을 이용하는 분야가 증가하면서 모바일 데이터베이스의 활용도가 점차 증가하고 있다.

그리고 e-commerce나 m-commerce 분야에서도 모바일 디바이스의 사용이 보편화되면서 모바일 디바이스에서의 파일로 처리 불가능한 데이터 처리에 대해 모바일 데이터베이스가 중요한 해법으로 등장하게 되었다. 따라서 e-commerce 기반 독자 생존 가능한 모바일 데이터베이스 개발 수요가 급격히 늘고 있다[3].

이러한 변화에 모바일 데이터베이스 기술이 중요한 역할을 한다. 모바일 데이터베이스 기술은 제한된 리소스 환경에서 경량화 서버로 최대의 성능을 발휘하는 것이다. 현재 모바일 데이터베이스에는 개척자로 꼽히는 SQLite를 중심으로 최근 발표된 Realm에 이르기까지 다양한 제품이 사용되고 있다.

이 논문은 그 중에서 가장 앞선 모바일 데이터베이스로 주목받고 있는 SQLite 데이터베이스의 검색 성능에 초점을 맞추고자 한다. 성능 실험을 위한 테스트 프로그램을 구현하여 실험을 실시하고자 한다. 테스트 프로그램은 대표적인 쿼리 유형들에서의 성능을 실험할 수 있도록 구성된다.

실험은 대부분의 모바일 시스템에서 사용되는 리눅스 환경을 기반으로 하여 검색 성능에 초점을 맞추어 분석한다. 성능 분석은 같은 환경에서 실험한 Linux Oracle 11g와 비교하여 실시한다.

이 논문의 구성은 다음과 같다. 2장에서 관련 연구를 보이고 실험 환경과 실험 결과와 분석은 3장에서 보인다. 결론은 4장에서 맺는다.

Ⅱ. 관련 연구

SQLite[4]는 2000년 8월에 미국의 Richard Hipp의 주도의 오픈소스 프로젝트로 제작되어 V1.0을 발표하였고 현재 V3.12.2를 제공하고 있는 대표적인 모바일 데이터베이스이다. SQLite는 모바일 데이터베이스 분야의 선두주자로서 최근까지 SQLite의 성능 분석과 개선에 대한 연구가 활발히 이루어지고 있다[5,6].

최근엔 2014년 새로운 모바일 데이터베이스인 Realm[7,8]이 출시되어 다양한 모바일 기업과 모바일 앱에서 사용되고 있으며 SQLite의 경쟁자로 등장했다.

ORMLite(Object Relational Mapping)는 Java와 관계 데이터베이스 사이의 매핑을 제공하는 오픈소스 프래임워크로서 안드로이드 플랫폼에서 JDBC를 지원한다. Java와 SQLite의 연결도 지원한다.

Core Data는 Apple에서 제공되는 객체 저장 프레임워크로서 OS X와 iOS에서 구동된다. 데이터를 관계형 모델로 저장할 수 있으며 추상화된 엔터티-릴레이션 구조로 표현할 수도 있다. 저장은 XML, binary 또는 SQLite에 할 수 있다.

그 외에도 페이스북에서 제공하는 모바일 데이터베이스 플랫폼인 Parse[9], Apache 프로젝트 중 하나인 CouchDB[10] 등이 등장했다.

이 중에서 현재 가장 많이 사용되고 있는 SQLite는 현재 SQLite V3.12 까지 출시되어 있다. 안드로이드 OS를 비롯한 다양한 플랫폼에서 작동되며 다른 데이터베이스 제품에 비해 가볍고 속도가 빠르다고 한다. 주요 특징으로는 단일 데이터베이스 파일을 이용하며 서버로 구동되지 않고 라이브러리 형태의 API로 작동한다는 것이다. 동적 자료형을 지원하여 편리한 코딩이 가능한 장점도 지니고 있다.

이 논문에서는 모바일 응용에서 사용되는 쿼리 유형에서 SQLite 데이터베이스가 보이는 성능을 테스트하기 위한 ESQL 프로그램을 구현하고 실험한다. 실험한 결과의 분석을 위해 동일한 환경에서 실험한 검증된 Oracle 11g의 실험 결과와 비교한다. 비교 분석 결과로서 SQLite 데이터베이스의 장점과 단점을 파악하고 적합한 사용 환경과 향후 개선 방향을 제시한다.

Ⅲ. 실험 결과와 분석

이 논문에서 제시하는 성능 테스트를 위한 검색 쿼리 유형은 다음과 같다.

1. Point Query : 기본키에 대한 단일 레코드 검색
2. Set Query : 복수 개 레코드 검색(단일 조건)
   1) index 없을 경우
   2) index 있을 경우
3. Range Query : 순차 검색
4. Multiple Condition Query : 복수 조건 검색
   1) index 없을 경우
   2) index 있을 경우
5. Join Query : 조인 검색
   1) index 없을 경우
   2) index 있을 경우

Point Query는 기본키 값을 조건으로 한 레코드의 검색으로서 하나의 레코드 검색 유형이다. 기본키에는 인덱스가 설정되어 있으므로 주로 기본키 인덱스 성능을 알 수 있는 쿼리 유형이다.

Set Query는 키가 아닌 값을 조건으로 무작위의 여러 레코드가 검색되는 유형이다. 키가 아닌 필드에 대하여 인덱스 유무에 따른 성능의 차이를 테스트한다.

Range Query는 기본키 필드에 대하여 일정 범위안의 순차 데이터를 검색하는 검색 유형이다.

Multiple Condition Query는 where절의 조건이 2개 이상 필드에 대한 질의로 쿼리로서 통계정보를 이용한 질의 최적화 여부를 판별할 수 있다. 인덱스 유무에 따른 성능의 차이도 실험한다.

Join Query는 조인 성능을 판별하기 위한 유형이다. 역시 인덱스 유무에 따른 성능의 차이를 테스트한다.

실험 환경은 다음과 같다. 리눅스는 Fedora 11, Kernel 2.6.29.4 버전을 사용한다. 실험 컴퓨터는 CPU Intel Core Duo 2.83Gh, 그리고 2Gb Memory를 가진다. SQLite는 V3.12.2 32bit 버전을 이용한다. Oracle은 11g (11.2.0.1.0) 32bit 버전을 사용하여 테스트를 진행한다. SQLite의 성능 실험을 위한 구현은 ESQL/C로 작성하여 gcc로 컴파일하였다. Oracle의 성능 실험을 위한 구현은 Pro*C로 구현하였고 Oracle의 proc와 gcc로 컴파일하였다. 실험 데이터는 임의로 생성한 1백 만 개의 레코드 테이블(8개의 필드)과 20만개의 레코드를 가진 테이블(8개의 필드)을 사용하였다. 키가 아닌 필드들은 동일한 값들을 최대 20%까지 가진다.

실험 결과에서 X축은 측정 방법의 종류를, Y축은 초(Sec.)의 단위를 가진다.

3.1. Point Query

그림 1은 Point Query에 대한 실험 결과이다. 무작위로 1백 만 개 중 1개의 기본키 값에 대한 접근 속도를 측정한 결과이다. 가벼운 SQLite가 Oracle보다 접근 속도가 빨랐다.

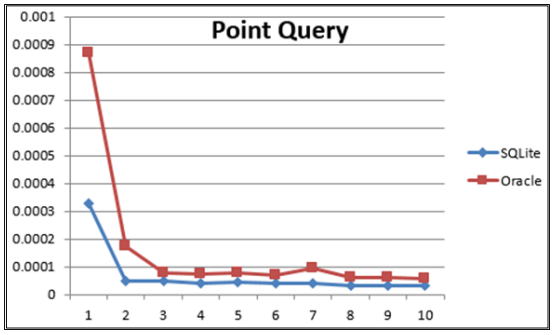

Fig. 1 Point Query Result

3.2. Set Query

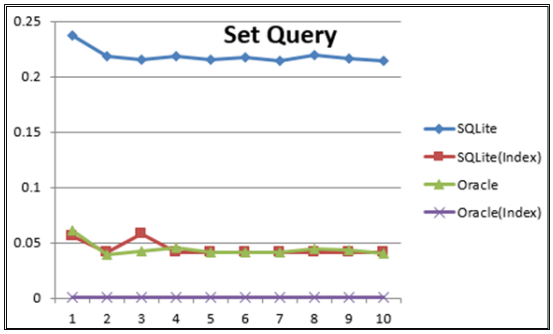

Fig. 2 Set Query Result

그림 2는 키가 아닌 필드에 대해 무작위로 조회한 Set Query 결과이다. 검색 결과는 최대 20% 복수개의 레코드들일 수 있다. 인덱스가 없는 경우의 검색에서 Oracle이 매우 우수한 결과를 보였다. 인덱스가 있는 경우에도 Oracle이 우수하였다. 인덱스 생성이 SQLite는 약

5배 성능 향상을 보인 반면 Oracle은 70배의 향상을 보였다.

3.3. Range Query

그림 3은 기본키 필드에 대한 Range Query의 실험 결과이다. 검색 레코드 수를 100개에서 5십 만 개까지 변화시키며 측정하였다. Oracle이 데이터 크기가 큰 경우에 우수한 성능을 보였다.

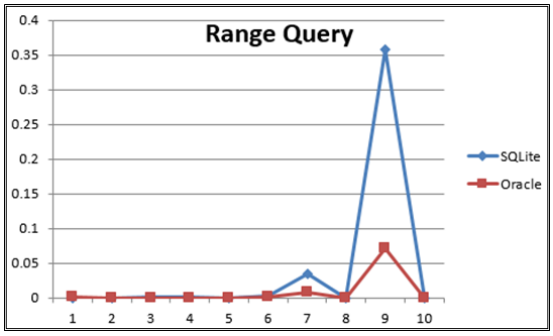

Fig. 3 Range Query Result

3.4. Multi-Conditions Query

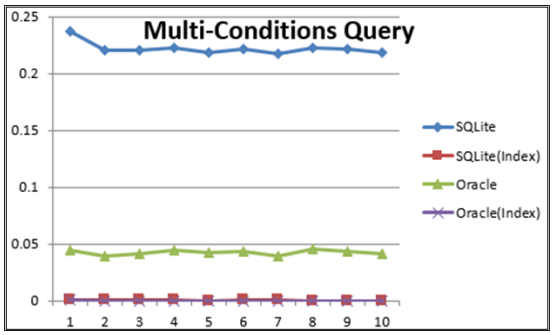

Fig. 4 Muti-Conditions Query Result

그림 4는 Multi-Conditions Query의 실험 결과이다. 복수 필드 값들의 조합으로 무작위 조회하고 측정하였다. 실험 결과 데이터의 통계 정보를 적극 사용하는 Oracle이 우수한 성능을 보였다. 인덱스가 있을 경우 두 데이터베이스는 비슷한 성능을 보였는데 Oracle은 500배의 향상을 보인 반면 SQLite는 약 2,000배 성능 향상을 보였다.

3.5. Join Query

그림 5는 외래키에 인덱스가 없는 경우의 Join Query의 실험 결과이다. 실험은 100만 개의 레코드를 가진 테이블과 20만 개의 외래키를 가진 테이블 간 조인을 실행하였다. 실험 결과, Oracle의 강력한 쿼리 최적화기능을 감안하더라도 SQLite에서 인덱스가 없는 조인은 치명적인 약점으로 보였다.

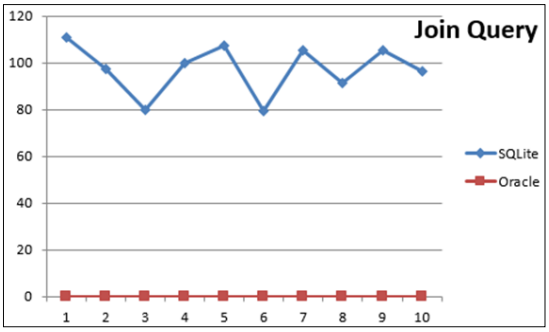

Fig. 5 Join Query Result (No Index)

그림 6은 인덱스를 사용한 Join Query의 결과로서 두 데이터베이스 간 성능 차이가 거의 나지 않는 것으로 측정되었다.

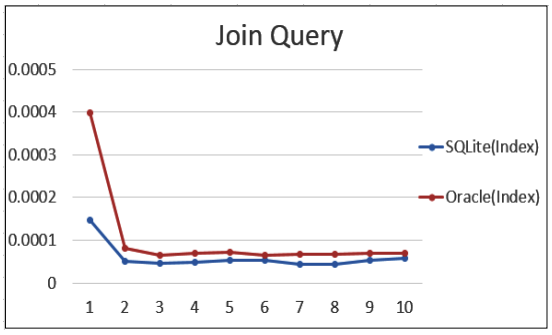

Fig. 6 Join Query Result (With Index)

3.6. 성능 분석

그림 7은 두 데이터베이스의 검색쿼리 유형에 따른 각각의 평균 실행 시간과 비율을 도식화 한 결과이다. 단, 여기서는 인덱스를 사용한 Set Query, Multi-Conditions Query, 그리고 Join Query의 경우는 제외하였다.

	SQLite	Oracle	Gap Rate
Point Query	0.0000686	0.000161	0.425031
Set Query	0.2187357	0.043981	4.973436
Range Query	0.0396043	0.00806	4.913624
MC Query	0.2220494	0.042318	5.247137
JOIN Query	102.34	0.029192	3505.707

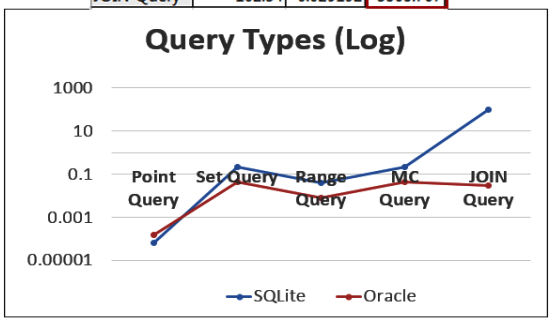

Fig. 7 Analysis of Performance Rates

그림 7에서 인덱스를 이용하여 하나의 데이터만을 검색하는 Point Query의 경우 경량의 SQLite가 오히려 우수하였다. Join Query를 제외한 나머지의 경우 Oracle이 대략 5배 정도의 우수한 성능을 보였다. 즉, 대량의 자료를 검색해야 하거나 인덱스 없이 데이터 처리할 경우 Oracle이 SQLite에 비해 5배 정도 성능이 우수하다. 이는 Oracle의 복잡하고 고도화된 질의 최적화 기능이 이유로 판단된다. 그리고, Join Query의 경우 SQLite는 인덱스가 반드시 필요한 것으로 분석되었다.

그림 8은 Set Query, Multi-Conditions Query, 그리고 Join Query에서 사용되는 두 데이터베이스의 인덱스 성능을 비교 분석한 결과이다.

	SQLite	Oracle
Set Query	4.922832	67.70443
MC Query	2005.866	468.6401
JOIN Query	1629950	95.12023

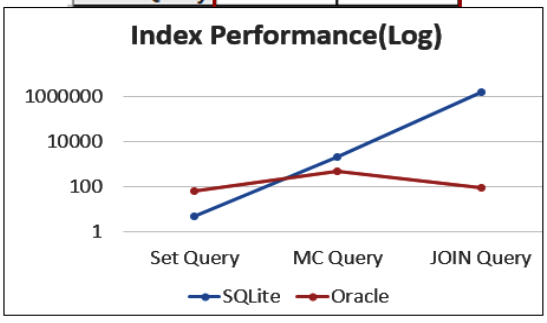

Fig. 8 Analysis of Index Performance

그림 8은 각 검색쿼리 유형에서 인덱스의 사용으로 빨라진 비율을 도식화한 결과이다. Oracle은 Multi-Conditions Query에서 성능 향상이 가장 두드러졌으며, SQLite의 경우 Set Query를 제외하고는 인덱스의 사용이 성능을 드라마틱하게 향상시켰다.

Ⅳ. 결 론

최근 모바일 환경의 보편화와 이에 따른 모바일 기기를 활용하는 새로운 모바일 응용 분야들이 대거 등장함에 따라 모바일 데이터베이스의 중요성이 커졌다. 새로운 모바일 데이터베이스들이 잇따라 등장하고 있으며 기존 제품들도 다시 주목을 받고 있다.

이 논문에서는 이러한 환경 변화에 따라 모바일 데이터베이스의 선두를 지켜온 SQLite 데이터베이스의 검색 성능 분석에 초점을 맞추었다.

가능한 검색 쿼리 유형을 모두 포함하도록 실험환경을 구축하고 구현 실험하여 성능을 측정하였다. 실험은 모바일 환경을 고려하여 Linux에서 실시하였고 동일한 실험을 Oracle 데이터베이스에서 반복하여 그 결과와 비교 분석하였다.

구현 실험과 성능 분석 결과 첫째, SQLite의 경량성으로 인하여 Point Query에서 무거운 Oracle보다 우수하다는 점을 확인하였다. 둘째, 그러나 나머지 경우에는 대략 5배 안팎의 성능 차이를 보이며 Oracle이 우수하였다. 이는 복잡하고 다양한 Oracle의 질의 최적화 기능이 인덱스가 없거나 대량의 데이터를 처리하는 경우에 우수하다는 분석이 가능했다. 셋째, SQLite는 인덱스를 반드시 사용해야 한다는 점이다. 특히 조인 처리에서 인덱스를 사용하지 않으면 급격한 성능 저하 현상이 나타났다.

이러한 이 논문의 실험 결과는 향후 SQLite의 검색 쿼리 성능 향상에 활용될 수 있을 것으로 생각한다. 또한 향후 Realm 등 경쟁 모바일 데이터베이스와의 성능 비교도 필요해 보인다. 그리고 안드로이드 등 다양한 모바일 플랫폼 환경에서 동일한 실험으로 성능 비교 분석이 필요할 것으로 생각된다.

ACKNOWLEDGEMENT

This work was supported by the research grant of the Busan University of Foreign Studies in 2016

REFERENCES

[1] S. Kim, J. Lee, M. Sin, S. Kim and Y. Kim, “Development of music & AV Server with Raspberry PI,” in *Proceedings of KSMTE*, pp. 117-117, 2015.

[2] M. Beck, M. Wemer and S. Feld, “Mobile Edge Computing: A Taxonomy,” in *Proceedings of 6th Conference on Advances in Future Internet*, pp. 48-54, 2014.

[3] A. Sripriya and R. Dhanapal, “Mobile software solutions using mobile database design methodology,” in *Proceedings of International Conference on ICCCNT(IEEE)*, pp. 1-5, 2010.

[4] Architecture of SQLite [Internet]. Available: http://www.sqlite.org/arch.html.

[5] D. Tuan, S. Cheon and Y. Won, “On the IO characteristics of the SQLite transactions,” in *Proceedings of the International Conference on Mobile Software Engineering and Systems*, Austin Texas, pp. 214-224, 2016.

[6] H. Ouarnoughi, J. Boukhobza, P. Olivier, L. Plassart and L. BellatrecheOu, “Performance analysis and modeling of SQLite embedded databases on flash file systems,” in *Design Automation for Embedded Systems*, Springer, Vol. 17, No. 3-4, pp. 507-542, Oct. 2014.

[7] Realm Mobile Database [Internet]. Available: http://realm.io/docs/#getting-started.

[8] Android Working with Realm Database - Replacing SQLite & Core Data [Internet]. Available: http://www. androidhive. info/2016/05/android-working-with-realm-database-replacing-sqlite-core-data/.

[9] Parse Server Guide [Internet]. Available: http://parseplatform.github.io/docs/parse-server/guide/.

[10] Apache CouchDB 2.0 Documentation [Internet]. Available: http://couchdb.apache.org/en/2.0.0/.

최진오(Jin-Oh Choi)

1991년 부산대학교 컴퓨터공학과 공학사
1995년 부산대학교 컴퓨터공학과 공학석사
2000년 부산대학교 컴퓨터공학과 공학박사
2000년~ 부산외국어대학교 임베디드소프트웨어학과 교수
※관심분야 : 모바일 App., Database App., Embedded System