JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지 Vol. 22, No. 12: 1571~1576, Dec. 2018

SQLite3 모바일 데이터베이스의 갱신 성능 비교

최진오*

Modification Performance Comparison of SQLite3 Mobile Databases

Jin-oh Choi*

School of Embedded IT, Busan University of Foreign Studies, Busan, 46234 Korea

요 약

최근 모바일 디바이스의 가장 주목받는 변화는 계산 성능의 획기적인 향상, 저장 용량의 대폭적인 증가, 인터넷의 상시 연결, 그리고 디스플레이 기술의 정교한 발전으로 꼽을 수 있다. 이에 따라, 모바일 디바이스를 활용한 데이터베이스 응용이 새롭게 등장하고 있다. 이러한 응용에는 모바일 서버용 데이터베이스, 에지 컴퓨팅을 위한 데이터베이스, 포그 컴퓨팅 등이 있다. 따라서 현재 출시된 모바일 데이터베이스에 주목하고 그러한 응용들에 적합한 성능을 가지고 있는지 주목하는 것이 중요하다. 이 논문에서는 대표적이고 우수한 모바일 데이터베이스인 SQLite3를 선택하여 갱신 성능 및 특성을 테스트하기 위한 실험을 실시한다. 실험 결과를 평가하기 위하여 동일한 환경에서 Oracle 데이터베이스의 결과와 비교하였다. 실험 결과 SQLite3의 Insert 성능은 개선 여지가 많았으며, Update 성능은 아주 우수한 것으로 밝혀졌다. 특히 Range Query에 우수한 성능을 보였다.

ABSTRACT

Recently, the attractive changes of mobile device are a improvement of the computing performance, dramatic improvement of storage capacity, constant connection to the internet, and sophisticated development of display technology. As a result, database applications utilizing mobile devices are emerging. These applications include databases for mobile servers, databases for edge computing, and fog computing. Therefore, it is important to pay attention to the current mobile database and pay attention to whether it has suitable performance for the applications. In this paper, the most common mobile database, SQLite3 is selected and experimented to test and understand the update performance and characteristics. The results of experiment are compared with the one of Oracle database at the same condition to evaluate the experiment. As a result, Insert Performance of SQLite3 has a lot of points to be improved and Update performance is very good. Especially, the performance of Range Query is excellent.

키워드 : SQLite3, 갱신 성능, 모바일 데이터베이스, 오라클 테스트

Keywords : Mobile Database, Oracle Test, SQLite3, Update Performance

Received 15 August 2018, **Revised** 23 August 2018, **Accepted** 12 September 2018

*** Corresponding Author** Jin-oh Choi(E-mail:jochoi@bufs.ac.kr, Tel:+82-51-509-6245)
School of Embedded IT, Busan University of Foreign Studies, Busan, 46234 Korea

Open Access http://doi.org/10.6109/jkiice.2018.22.12.1571 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

모바일 IT 환경에서 최근 가장 주목받는 변화는 계산 성능의 획기적인 향상, 저장 용량의 대폭적인 증가, 인터넷의 상시 연결, 그리고 디스플레이 기술의 정교한 발전으로 꼽을 수 있다. 이로 인하여 오늘날의 모바일 IT 기기는 모바일 인터넷 디바이스의 지위에서 모바일 고성능 컴퓨터로 그 위상이 바뀌었다. 최근에는 모바일 워크스테이션(Mobile Workstation)이 등장하여 모바일 장치가 워크스테이션 수준의 서버로 활용되는 응용분야에 사용되고 있다.

따라서 스마트폰으로 대표되는 모바일 IT 디바이스는 단말기로 작동하던 지위에서 벗어나 서버나 미들웨어의 역할도 수행할 수 있는 상황을 만들고 있다. 여기서 모바일 장치가 서버나 미들웨어의 역할을 수행하기 위해서는 데이터베이스의 설치 및 운용이 반드시 필요하다. 따라서 모바일 기기에 최적화된 모바일 데이터베이스의 필요성이 점점 커지고 있으며 새로운 모바일 데이터베이스가 지속적으로 새로 개발되고 발표되고 있다.

초기의 모바일 디바이스용 데이터베이스는 단순히 파일 시스템을 대체하는 수준에 머물러 있었으나 점차 디바이스의 발전과 응용 분야에 확대에 따라 미들웨어를 위한 데이터 고속 저장소에서부터 서버용 범용 데이터베이스까지 확장 발전하고 있다.

모바일 데이터베이스는 가장 선두주자로 꼽히는 SQLite가 2000년에 등장한 이래 최근까지 SQLite3로 서비스 되고 있다. 2014년에는 Realm이 새롭게 발표되었는데 현재 상용 SW로 서비스되고 있고 확장 추세에 있다. 또한 최근 모바일 장치에서 대용량 데이터의 고속 저장을 지원하는 Machbase의 Edge 모바일 데이터베이스가 발표되어 관심을 끌고 있다.

이 논문에서는 최근 관심의 초점이 모아지고 있는 모바일 데이터베이스들 중 대표 격인 SQLite3를 선정하여 모바일 데이터베이스의 성능이 어느 정도까지 발전했는지 테스트해보고자 한다. 선행 연구[1]에서 SQLite3의 검색 성능의 비교 분석에 이어, 여기에서는 SQLite3의 갱신 성능을 테스트하고 평가하기 위한 실험을 진행한다.

실험 결과를 비교 분석하기 위해 동일한 환경에서 동일한 조건으로 Oracle 데이터베이스(Oracle 11g)에 대하여서도 실험을 실시한다. Oracle에 대하여 비교 실험을 진행하는 이유는 완성도가 가장 높은 DB와의 성능 차이를 수치화해보기 위함이다. 향후 동급의 모바일 DB들과의 성능 평가를 진행할 계획이다.

Ⅱ. 관련 연구

모바일 데이터베이스의 활용이 보편화되는 사례로 먼저 최근 라즈베리파이(Raspberry Pi)나 아두이노(Arduino)를 이용한 데이터베이스 서버 구축이 많이 시도되고 있다[2]. 또한 모바일 장치의 센스 기술이 발전하여 모바일 장치로 수집되는 센싱 데이터를 로컬 데이터베이스에 저장하였다가 추후 마스터 데이터베이스와 동기화하는 기술인 mobile edge computing[3] 기법이 등장하여 활용도가 점차 증가하고 있다. 포그 컴퓨팅(Fog Computing)[4]도 유사한 개념으로 방대한 양의 센싱 데이터를 발생 근처 디바이스에 저장하고 분석하여 선별적으로 활용하고자 하는 새로운 아키텍처로서 모바일 데이터베이스를 필요로 한다.

모바일 데이터베이스 제품군으로 SQLite[5]는 2000년에 발표되어 널리 사용되며 그 성능을 인증 받고 있다. Open Source SW이며 현재 V3.23.1 버전까지 지원되고 있다. Realm[6]의 경우 2014년 새로운 모바일 데이터베이스로 출시되어 다양한 모바일 기업과 모바일 앱에서 사용되고 있으며 SQLite의 경쟁자로 등장했다. Realm은 서버, 에지, 그리고 메모리 데이터베이스로 활용이 가능하다. 또한 최근 국내 제품으로서 Edge 모바일 데이터베이스[7]가 Machbase에서 발표되었다. 이 모바일 데이터베이스는 IoT Gateway 장비에 사용되며 센서 등에 의해 생성되는 대용량의 데이터를 고속으로 저장하는 역할을 수행한다.

[1]에서 대표적인 모바일 데이터베이스인 SQLite의 검색 성능을 다양한 실험을 통해 수치화해서 보였다. 그리고 비교를 위해 Oracle 11g의 성능을 같이 평가해 보였다. 그러나 이 연구는 갱신 성능에 대한 실험과 분석이 실시되지 않았다.

이 연구에서는 SQLite3 모바일 데이터베이스에 대한 갱신 성능 평가 및 비교를 위한 구현과 실험을 진행한다. 그리고 실험 결과를 Oracle의 결과와 비교 분석하여 그 의미를 해석하고자 한다.

Ⅲ. 성능 테스트를 위한 갱신 쿼리 유형

쿼리는 갱신문을 대상으로 한다. 실험해서 평가하고자 하는 갱신 쿼리는 모든 가능한 갱신 유형을 포함하도록 구성하였다. 실험하는 갱신 쿼리 유형은 다음과 같다.

1. Insert Query
   1) Point Insert
   2) Range Insert
2. Update Query
   1) Point Update, with no index
   2) Point Update, with index
   3) Range Update, with no index
   4) Range Update, with index
   5) Full Table Update
3. Delete Query
   1) Point Delete, with no index
   2) Point Delete, with index
   3) Range Delete, with no index
   4) Range Delete, with index
   5) Full Table Delete

'Point' 쿼리는 한 레코드를 Insert하거나 키(Key) 값을 조건으로 한 레코드를 검색하여 Update하거나 Delete하는 쿼리를 말한다. 'Range' 쿼리는 복수개 레코드 Insert 또는 일정 범위안의 순차 데이터를 검색하여 Update하거나 Delete하는 쿼리를 말한다. Range의 범위는 100개, 500개, 1,000개, 5,000개, 10,000개, 50,000개, 100,000개, 300,000개, 그리고 600,000개를 포함한다. 그리고 전체 테이블 갱신은 실험 데이터의 레코드 1,000,000개 전체를 갱신하는 쿼리를 말한다.

여기서 Insert 쿼리에서 Index가 없을 경우와 Index가 있을 경우를 구분하여 실험하는 것은 큰 성능 차이가 없어 생략하였다.

Ⅳ. 실험 결과

실험 환경은 다음과 같다.

1. 리눅스: CentOS 6.7, Kernel 2.6.32., 64bit
2. 실험 컴퓨터: CPU Intel Xeon E5-2690 Core 8, 2.60GHz, 32Gb Memory
3. SQLite: SQLite3 V3.25.2
4. Oracle: Oracle 11g (Ver 11.2.0.1.0)

SQLite3의 성능 실험을 위한 구현은 ESQL/C로 작성하여 gcc Ver. 4.4.7로 컴파일 하였다. Oracle의 성능 실험을 위한 구현은 Pro*C로 구현하였고 Oracle의 proc 전처리기와 gcc로 컴파일 하였다.

데이터베이스 실험 환경은 다음과 같다. 먼저 Insert 쿼리의 테스트를 위해서 8개의 필드를 가진 레코드 당 88byte 크기의 테이블을 생성하였다. Update 쿼리와 Delete 쿼리를 실험하기 위한 테이블은 역시 8개의 필드를 가진 레코드 당 88byte 크기의 테이블을 사용하였다. 각 레코드는 키를 제외한 필드들에 랜덤(Random) 스트링과 숫자를 입력하여 생성하였고 1백 만 개의 임의 레코드를 생성하여 저장하였다. 키가 아닌 필드들은 동일한 값들을 최대 20%까지만 가지도록 조정하여 최대한 고른 분포를 가진 테이블을 생성하여 사용하였다.

실험 결과에서 X축은 측정 방법의 종류를, Y축은 초(Sec.)의 단위를 가진다. Insert 쿼리에서 X축은 실행 회수를 표시한다. Update 쿼리와 Delete 쿼리에서 Point 쿼리에 대한 실험 결과 그래프의 X축도 실행 회수를 표시한다. 다만 Range 쿼리의 실험 결과 그래프에서 X축은 Range의 범위에 따른 실험 종류를 구별한다.

실험 결과 Insert 쿼리는 그림 1에서와 같이 SQLite3와 Oracle 모두 일정 시간 간격으로 큰 지연 현상을 보였다. 이는 버퍼 처리에 따른 지연으로 보인다. 버퍼가 다 찬 경우 실제 Flush 처리가 필요하기 때문이다. 100개의 Insert 쿼리를 처리하는 성능 비교에서는 그림 1과 같이 SQLite3의 Insert 쿼리 처리 성능보다 Oracle의 성능이 월등히 좋았다. SQLite3의 한 Insert 쿼리 처리 평균 시간이 0.049sec이며 Oracle은 0.007sec로 약 7배의 성능 차이를 보였다. 더구나 실험 결과 SQLite3의 10만 개 레코드 입력 처리 시간이 4,345sec(0.043sec/1rec.)였고 Oracle은 6.6sec이었다. SQLite3의 경우 100개 레코드 입력 처리와 10만 개 레코드 입력 처리 시간 평균이 거의 비례하였으나 Oracle은 평균 0.007sec이므로 10만개 레코드 처리 시간은 714초로 추정되었으나 실제 처리 시간은 6.6sec로 아주 우수하였다. Oracle 11g에 대한 Oracle사의 자료[8]에 따르면, 11g 버전은 'Batch Insert'

에 로그(Log) 용량을 줄이는 'Direct Path Insert' 기법을 도입하여 처리 효율을 획기적으로 높이고 있다.

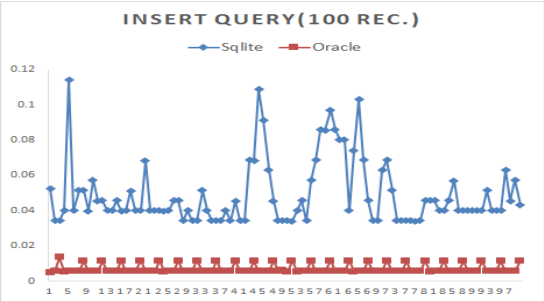

Fig. 1 Performance of Insert Query

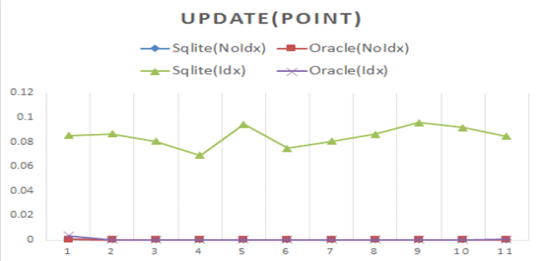

Fig. 2 Performance of Update Query: Point

Update-Point 쿼리는 그림 2와 같이 SQLite3와 Oracle 모두 비슷한 성능을 보였다. Index 처리 오버헤드 때문에 Index가 없는 경우에 우수한 성능을 보였다. 특히 SQLite3의 경우 Index가 있을 경우 Update 쿼리 처리 오버헤드가 없을 경우에 비해 442배로 컸다. Oracle은 3배 차이로 Index 오버헤드가 발생했다.

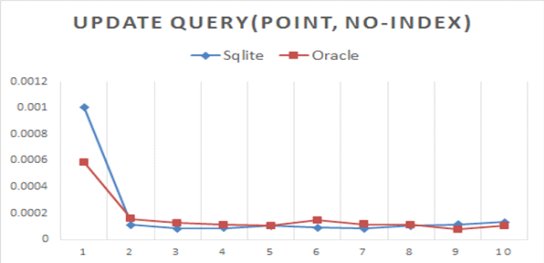

Fig. 3 Performance of Update Query: Point, No-Index

그림 3에서 Index가 없는 경우의 Point Update 쿼리 성능을 보였다. SQLite3와 Oracle은 거의 유사한 결과를 보였다.

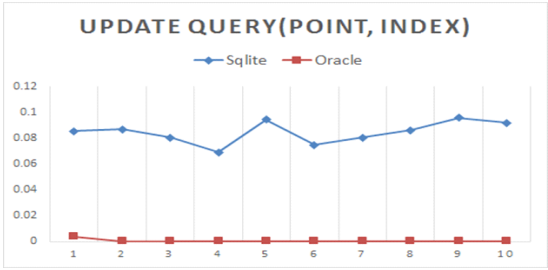

Fig. 4 Performance of Update Query: Point, With-Index

Index가 있는 경우의 Point Update 쿼리 성능은 그림 4에서 보듯이 SQLite3와 Oracle은 큰 차이를 보였다. SQLite3의 평균 Update 처리 시간은 0.08sec이었고 Oracle은 0.0005sec로서 약 169배의 차이를 보였다. 수정한 필드가 기본키이므로 B^{+}-트리의 수정 성능의 차이로 보인다.

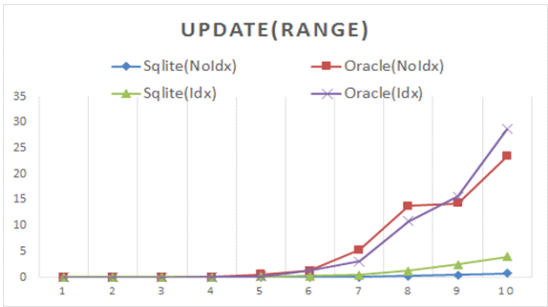

Fig. 5 Performance of Update Query: Range

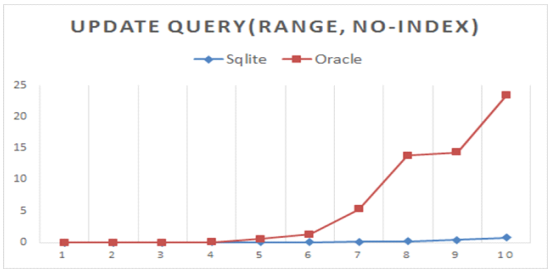

Fig. 6 Performance of Update Query: Range, No-Index

그림 5는 Range Update 쿼리의 실험 결과를 보였다. X축 번호가 커질수록 Range가 커지며 10번째 실험 결과는 전체 테이블을 수정하는 쿼리이다. 이 실험에서는

SQLite3가 연속된 대량 범위의 데이터를 수정할 때 Oracle보다 우수한 성능을 보였다. Index에 따른 오버헤드는 SQLite3가 6배, Oracle이 1.1배를 보였다.

그림 6에서 Index가 없는 경우의 Range Update 쿼리의 성능을 보였다. SQLite3가 Oracle에 비하여 Range가 커질수록 우수한 성능을 보였다. 둘 사이의 평균 성능은 각각 0.15sec와 5.88sec로서 37배의 차이를 보였다.

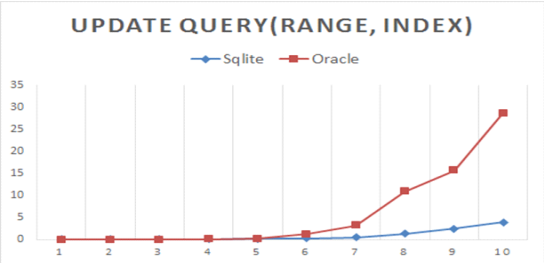

Fig. 7 Performance of Update Query: Range, With-Index

그림 7에서 Index가 있는 경우의 Range Update 쿼리의 성능을 보였다. 이 결과는 Index가 없는 경우와 거의 유사하였다. Range가 커지면 Oracle의 성능이 급격히 저하되는 현상을 보였다.

그림 8과 같이 Delete-Point 쿼리도 SQLite3와 Oracle 모두 비슷한 성능을 보였다. Index 처리 오버헤드 때문에 Index가 없는 경우에 우수한 성능을 보였다. Index 오버헤드는 SQLite3의 경우 약 1.1배, Oracle의 경우 약 1.5배의 차이를 보였다.

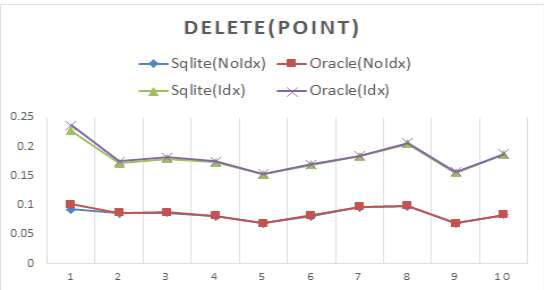

Fig. 8 Performance of Delete Query: Point

그림 9에서 Index가 없는 경우의 Point Delete 쿼리 성능을 보였다. 평균 처리 시간이 SQLite3가 0.08sec, Oracle이 0.001sec로서 약 57배의 성능 차이로 Oracle이 우수하였다.

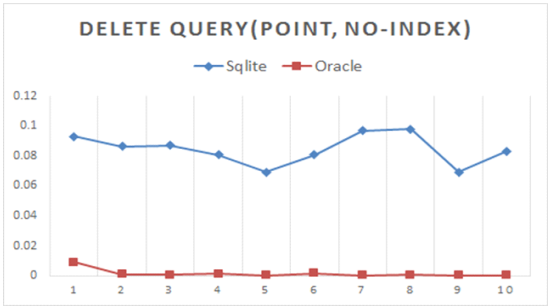

Fig. 9 Performance of Delete Query: Point, No-Index

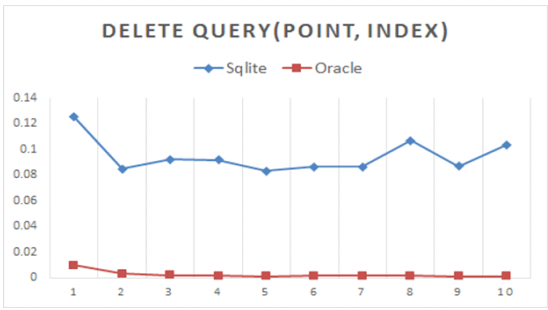

Fig. 10 Performance of Delete Query: Point, With-Index

Index가 있는 경우의 Point Delete 쿼리 성능을 그림 10에서 보였다. 평균 처리 시간이 SQLite3가 0.09sec, Oracle이 0.002sec로서 Oracle이 약 42배 우수하였다.

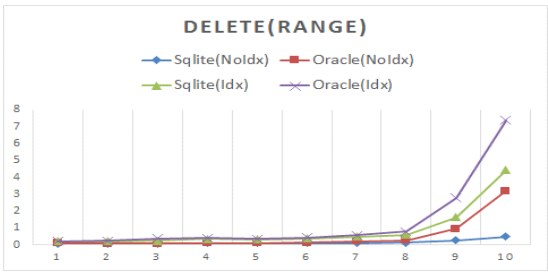

Fig. 11 Performance of Delete Query: Range

Range Delete 쿼리도 Range Update 쿼리와 아주 비슷한 성능 양상을 보였다. 그림 11에서 SQLite3가 Oracle보다 대량 데이터 처리에서 우수한 성능을 보임을 알 수 있다. Index가 있는 경우는 모두 처리 성능이 떨어졌다. Index 오버헤드는 SQLite3의 경우 2.4배, Oracle은 1.3배 차이를 보였다.

Ⅴ. 결 론

이 논문에서는 SQLite3 데이터베이스의 갱신 쿼리 처리 성능을 실험하고 분석해 보였다. 실험 결과를 평가하기 위해 실험 결과를 동일한 환경의 Oracle의 것과 비교하여 살펴보았다.

실험결과 SQLite3는 Insert에서 Oracle과 큰 성능 차이를 보였다. 평균 약 7배의 성능 차이를 나타내었다. 특히 대용량 레코드의 Insert인 Bach Insert에서 Oracle과의 큰 차이를 보였다. 이는 SQLite3에 대용량 데이터 입력에 대한 성능 개선 전략이 없는 증거로 보인다. 따라서 대용량 센싱 데이터를 고속 입력 처리해야 하는 모바일 에지 데이터베이스로 부적합한 것으로 판단된다.

Update 쿼리에서는 SQLite3의 성능이 Oracle과 대등하거나 보다 우수한 결과를 보였다. 특히 Range Update 쿼리에서는 SQLite3의 성능이 월등히 우수하였다. Delete 쿼리에서는 Oracle과의 가장 큰 성능 격차를 보였다. 이는 Delete 쿼리 처리에 SQLite3의 성능 개선 여지가 많다는 것을 의미한다. 1개 레코드에 대한 SQLite3의 자체 갱신 쿼리 성능은 Insert 쿼리는 평균 0.049sec, Point Update 쿼리는 0.0002sec, 그리고 Point Delete 쿼리는 0.084sec로 Update 쿼리 성능이 가장 우수하였다.

실험 결과를 바탕으로 향후 Android 환경에서 새로운 모바일 데이터베이스와의 성능 실험을 실시할 계획이다. 모바일 환경에서 모바일 데이터베이스간 성능 분석 결과를 도출할 계획이다.

ACKNOWLEDGEMENT

This work was supported by the research grant of the Busan University of Foreign Studies in 2018

REFERENCES

[1] J. Choi, “Implementation and Experiment for Search Performance Analysis of SQLite Mobile Database,” *Journal of The Korea Institute of Information and Communication Engineering*, vol. 21, no. 2, pp. 333-338, Feb. 2017.

[2] H. Lee, J. Oh, “Design and Implementation of a Small Server Room Environment Monitoring System by Using the Arduino,” *Journal of The Korea Institute of Electronic Communication Science*, vol. 12, no. 2, pp. 385-390, Feb. 2017.

[3] K. Wang, H. Yin, W. Quan, and G. Min, “Enabling Collaborative Edge Computing for Software Defined Vehicular Networks,” *Journal of IEEE Network*, vol. 32, no. 5, pp. 112-117, May 2017.

[4] B. Rad, A. Shareef, “Fog computing: A Short Review of Concept and Applications,” *International Journal of Computer Science and Network Security*, vol. 17, no. 11, pp. 68-74, Nov. 2017.

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

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

[7] MachBase Database. IoT Gateway for Edge Analytics [Internet]. Available: http://machbase.com/product-iot-gateway.

[8] Oracle Help Center. Oracle 11g Database new Features [Internet]. Available: https://docs.oracle.com/cd/B28359_01/server.111/b28279/chapter1.htm#NEWFTCH1.

최진오(Jin-Oh Choi)

1991년 부산대학교 컴퓨터공학과 공학사
1995년 부산대학교 컴퓨터공학과 공학석사
2000년 부산대학교 컴퓨터공학과 공학박사
2000년~ 부산외국어대학교 임베디드IT학부 교수
※관심분야 : Mobile Database 응용, Mobile Edge Computing, Embedded System