JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지 Vol. 26, No. 1: 1~6, Jan. 2022

Linux File Systems에 따른 SQLite3 데이터베이스의 검색 성능 비교

최진오*

Comparison of Search Performance of SQLite3 Database by Linux File Systems

Jin-Oh Choi*

*Professor, Division of Software, Busan University of Foreign Studies, Busan, 46234 Korea

요 약

최근 IoT 센서를 이용하여 데이터를 로컬에서 생산하고 스트림으로 제공하는 엣지 컴퓨팅(Edge Computing) 응용 분야가 넓어지고 있다. 대량으로 생산된 데이터는 실시간 처리를 위해 모바일 장치의 데이터베이스에 저장했다가 필요한 시점에 서버와 동기화된다. 이러한 응용 분야를 지원하기 위한 다양한 모바일 데이터베이스가 개발되었다. CloudScape, DB2 Everyplace, ASA, PointBase Mobile 등이며 그중 가장 널리 사용되는 대표적 모바일 데이터베이스는 리눅스 기반 SQLite3이다. 이 논문에서는 서버와 동기화 시 필요한 성능에 초점을 맞추었다. SQLite3의 정보 선택 시 필요한 검색 성능을 데이터베이스가 저장된 각 리눅스 파일 시스템의 종류에 따라 비교 분석하였다. 그래서 다양한 검색 쿼리 유형에 따라 파일 시스템별로 성능 차이를 확인하고 인덱스 사용 환경과 테이블 스캔 환경에 따라 더 적합한 리눅스 파일 시스템을 적용하는 기준을 마련하고 제시하였다.

ABSTRACT

Recently, IoT sensors are often used to produce stream data locally and they are provided for edge computing applications. Mass-produced data are stored in the mobile device's database for real-time processing and then synchronized with the server when needed. Many mobile databases are developed to support those applications. They are CloudScape, DB2 Everyplace, ASA, PointBase Mobile, etc, and the most widely used database is SQLite3 on Linux. In this paper, we focused on the performance required for synchronization with the server. The search performance required to retrieve SQLite3 was compared and analyzed according to the type of each Linux file system in which the database is stored. Thus, performance differences were checked for each file system according to various search query types, and criteria for applying the more appropriate Linux file system according to the index use environment and table scan environment were prepared and presented.

키워드 : 모바일 데이터베이스, 리눅스 파일시스템, 검색 성능, SQLite3

Keywords : Mobile database, Linux file system, Search performance, SQLite3

Received 13 October 2021, **Revised** 27 October 2021, **Accepted** 31 October 2021

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

Open Access **http://doi.org/10.6109/jkiice.2022.26.1.1** print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

IoT 디바이스가 보편화되면서 데이터 소스의 근처에서 대량의 데이터를 수집하는 일이 쉽게 가능해지고 있다. 즉, 데이터 소스와 가까운 장소에서 컴퓨팅을 수행하는 일이 수월해지고 있다. 이때, 여기서 요구되는 것은 모바일 스마트 장치, 센서, 그리고 모바일 데이터베이스 기술이다.

이러한 응용 분야를 엣지 컴퓨팅(edge computing)[1][2]이라 부르며, 이는 나아가 하이브리드 클라우드(hybrid cloud)[3]를 구축하는 근간이 된다. 하이브리드 클라우드는 기존의 데이터센터 서버에서 퍼블릭 클라우드(public cloud)를 통해서만 데이터를 수집하고 컴퓨팅하던 방식에서 그 한계를 확장하는 개념이다.

일반적으로 엣지 컴퓨팅은 모바일로 이루어진다. IoT 단말은 자체 센서를 이용하여 실시간 데이터를 고속으로 수집하여 로컬 데이터베이스에 저장하고 처리하며 컴퓨팅을 수행한다. 여기서 이 응용 기술은 IoT 장치의 성능 향상과 소형화에 그치지 않고 5G와 같은 통신 능력의 진보에 따라 같이 발전하는 분야임을 알 수 있다.

엣지 컴퓨팅이나 하이브리드 클라우드 응용에서 최근 가장 주목받는 변화 중 한 가지 사례는 실시간 스트림(stream)으로 발생하는 대용량 데이터의 처리 방법이다. 고속 실시간으로 수집된 대용량의 데이터는 손실을 막기 위해 모바일 데이터베이스에 실시간 저장된다. 저장된 데이터는 필요에 따라 가공되고 컴퓨팅 되어 데이터센터와 같은 서버에 동기화된다.

이렇게 모바일 데이터베이스를 사용하는 엣지 컴퓨팅에서 얻을 수 있는 이점은 다음과 같다. 첫째, 로컬에서 분산 방식으로 수행되므로 데이터 수집과 전송 비용을 절약할 수 있다. 둘째, 불필요한 전체 데이터를 원격 서버로 전송하는 것보다 필요한 컴퓨팅을 로컬에서 수행하므로 데이터 전송량을 줄일 수 있다. 셋째, 개인 생체 정보 등 민감한 데이터는 로컬에서 보안 처리를 거치므로 중앙 서버에 집약되는 것을 막을 수 있어 개인 프라이버시 보호가 가능하다. 마지막으로 중앙 데이터센터 서버를 모바일 데이터베이스를 통해 다운사이징을 할 수 있어서 데이터 수집의 비용적 효율적 측면에서 유리하다.

엣지 컴퓨팅 응용에서 모바일 데이터베이스는 현장에서 발생한 데이터를 저장하고 필요에 따라 가공한 후 중앙 데이터센터와 동기화하는 능력을 필요로 한다. 또한, 고속 실시간 데이터를 손실 없이 저장하기 위한 처리 능력을 갖추어야 한다. 뿐만 아니라 로컬 컴퓨팅과 서버 동기화를 위한 실시간 데이터 처리 능력도 필요로 한다.

현재 발표된 모바일 데이터베이스로는 Machbase Edge Mobile DB, Realm, CloudScape, DB2 Everyplace, ASA(Adaptive Server Anywhere), PointBase Mobile Edition, SQLite 등이 있다. 이 중 SQLite가 가장 대표적인 제품으로서 최근까지 SQLite3로 선두 자리를 지키며 다양한 모바일 응용에 사용되고 있다.

그런데, 대부분의 모바일 데이터베이스는 IoT 디바이스의 플래시 메모리를 데이터 저장장치로 사용한다. 플래시 메모리는 특정 파일 시스템으로 초기화되어 동작하며 그에 따라 서로 다른 특성을 보인다. 여기서 이 논문은 논의의 범위를 다음과 같이 좁혀 엣지 컴퓨팅에서 모바일 데이터베이스의 성능 비교를 수행하고자 한다. 먼저 SQLite3 모바일 데이터베이스를 대상으로 하며, 리눅스 기반 파일 시스템인 XFS와 Ext4, 그리고 가장 널리 사용되는 FAT 파일 시스템으로 성능 비교 대상을 좁힌다. 마지막으로 데이터 처리 유형은 로컬 컴퓨팅과 동기화에 필요한 검색 기능에 한정하여 성능을 평가하고자 한다. 즉, SQLite3의 검색 유형별 파일 시스템에 따른 성능을 측정하고 분석한다. 그 결과로서 쿼리 유형에 따른 각 파일 시스템의 장단점과 특성을 파악할 수 있을 것으로 기대된다. 이후 다른 모바일 데이터베이스와 파일 시스템으로 비교 분석 대상을 쉽게 확대할 수 있을 것이다.

선행 연구[4]에서 3가지 파일 시스템에서 SQLite3의 갱신 성능을 비교 분석하기 위해 동일한 환경에서 동일한 조건으로 비교 분석 실험을 실시하였다. 이 연구를 바탕으로 리눅스 기반의 3종류의 파일 시스템에서 동일한 조건의 검색 쿼리에 대한 성능을 테스트하고 비교 분석하고자 한다.

Ⅱ. 관련 연구

모바일 데이터베이스의 성능에 대한 비교 분석은 실험 보고서와 연구에 의해 많이 소개되어왔다. 다만 최근 다량 출시된 모바일 데이터베이스 제품을 포함한 연구

결과는 인터넷의 보고서 자료[5] 외에는 아직 찾기 어렵다. [5]에서는 Room Lib., GreenDao, Realm, ObjectBox 등에 대한 create, select, update, 그리고 delete 쿼리의 성능을 CRUD(안드로이드의 전형적 쿼리 유형) 연산으로 비교해서 보이고 있다. 이 결과로부터 쿼리 유형에 따른 가장 적합한 모바일 데이터베이스나 라이브러리를 선택할 수 있는 기준을 찾을 수 있다. 하지만 모바일 데이터베이스가 사용하는 저장 공간의 파일 시스템에 따른 성능 비교 분석은 아직 연구 결과나 보고서를 찾아보기 어렵다.

그리고 리눅스 기반 파일 시스템들은 새로 개발되거나 기능이 개선되어 업그레이드되는 경우 그 특성과 개선 기능에 대해서는 공식적인 분석 자료나 보고서로 확인하고 비교해볼 수 있다. 그러나 이 논문에서 초점을 맞추는 플래시 메모리에 구현된 파일 시스템에서 모바일 데이터베이스의 검색 쿼리 유형별 성능 비교 분석 자료는 역시 아직 널리 연구되지 않아 찾아보기 어렵다.

가장 널리 사용되는 대표 모바일 데이터베이스는 SQLite[6]이다. Open Source 소프트웨어로서 2021년 현재 SQLite3 V3.36 버전까지 발표되었다. 파일 시스템의 경우 리눅스 기반 XFS[7]가 2014년 Red Hat Enterprise Linux 7의 기본 파일 시스템으로 사용되면서 기존 Ext4[8]를 대체하게 되었다. 현재 대부분의 리눅스가 두 파일 시스템을 동시에 지원하고 있다.

[9]에서 Ext4, btrfs, 그리고 XFT의 성능 테스트를 보여주고 있지만, 모바일 데이터베이스의 쿼리 유형과는 무관하다. 이 논문에서 실험하고자 하는 파일 시스템은 호환성에 문제점을 안고 있는 brtfs를 제외하고 XFS, Ext4, 그리고 가장 널리 알려진 FAT 파일 시스템을 실험하여 비교 대상으로 삼고자 한다.

이 논문에서 대표적인 모바일 데이터베이스인 SQLite3를 대상으로 검색 쿼리에 대해 서로 다른 파일 시스템에서 어떠한 성능 차이를 보이는지 실험하고 결과를 분석한다.

Ⅲ. 실험을 위한 검색 쿼리 유형

검색 성능을 실험하기 위해 가능한 쿼리 유형을 모두 포함하도록 구성하였다. 실험하는 검색 쿼리 유형은 다음과 같다.

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

‘Point Query’는 기본키 값을 조건으로 하나의 레코드를 검색하는 쿼리를 말한다. SQLite3는 기본키에 autoindex를 생성하므로 이 성능을 테스트할 수 있다. ‘Set Query’는 기본키가 아닌(인덱스가 없는) 필드를 검색하는 쿼리로서, 전체 테이블을 검색해야 하는 쿼리 종류이다. ‘Range Query’는 기본키 필드에 대하여 일정 범위 안의 연속 데이터를 검색하는 쿼리이다. 기본키에 대하여 B+-Tree 인덱스가 생성되어 있으므로 역시 이 성능을 테스트할 수 있다. ‘Multiple Condition Query’는 기본키가 아닌 필드들에 대하여 검색 조건이 2개 이상인 쿼리이다. 마지막으로 ‘Join Query’는 조인 성능을 테스트하기 위한 쿼리이다.

실험을 위한 데이터로 임의의 값을 가진 100만 개의 레코드 테이블과 조인 테스트를 위한 100만 개의 임의 레코드를 가진 테이블을 사용하였다. 레코드 크기는 각각 10필드 108바이트와 8필드 87바이트로 구성하였다. 각 레코드는 키를 제외한 필드들에 랜덤(Random) 스트링과 숫자를 입력하였으며, 키가 아닌 필드들은 동일한 값들을 최대 20%까지만 가지도록 조정하여 최대한 고른 분포를 가진 테이블을 생성하여 사용하였다.

Ⅳ. 실험 결과

다음은 실험에 사용한 환경이다.

1. 리눅스: CentOS 8.2.2, Kernel 4.1.8, 64bit
2. 실험 컴퓨터: CPU Intel Core i5-6500, 3.2GHz, 8Gb Memory
3. SQLite: SQLite3 V3.36 for Linux 64bit, ESQL/C
4. File System: 각각 2Gb의 XFS, Ext4, 그리고 FAT 파일 시스템 플래시 메모리

실험은 샘플 데이터베이스를 구축하고 검색 성능 측정을 위한 코드 구현을 통해 진행되었다. 구현은 SQLite3의

ESQL/C 방식으로 코딩하고, SQLite3의 라이브러리를 이용하여 gcc V8.4.1로 컴파일하였다. 실험 수치는 100회 반복한 실험 결과를 평균하여 측정한 것이다. 그래프에서 X축은 측정 횟수를, Y축은 수행 시간(초)을 표시한다.

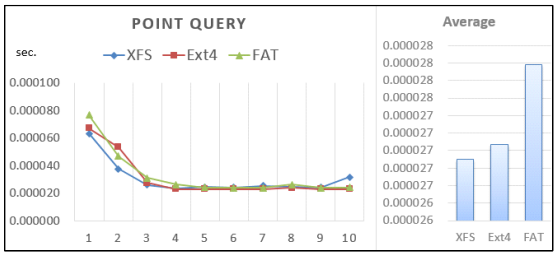

Fig. 1 Performance of Point Query

Point Query는 기본키 레코드 하나를 검색하는 쿼리로서 실험 결과 그림 1과 같이 파일 시스템 종류와 관계없이 비슷한 성능을 보였다. 전체 평균 성능은 XFS가 가장 좋았으며 FAT이 가장 느렸으나 그 차이는 1μs 정도였다. 데이터베이스가 하나의 파일로 구성되어 있는 상황에서 Point Query는 데이터베이스의 구조나 데이터 변경 없이 인덱스 접근을 통한 조회 작업만 진행하는 쿼리이다. 따라서 파일 시스템의 종류와 상관없이 한 레코드 탐색에는 시간이 거의 비슷하게 소요되는 것으로 판단된다. 다만 FAT 파일 시스템의 경우 큰 데이터베이스 파일 크기로 인해 긴 FAT 체인이 형성되어 상대적으로 효율이 조금 떨어지는 것으로 보인다.

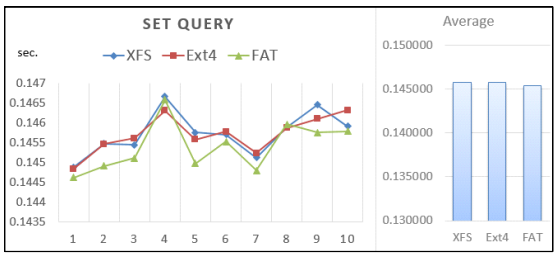

Fig. 2 Performance of Set Query

Set Query는 인덱스가 없는 필드에 대한 복수 개 레코드 검색 쿼리로서 그림 2와 같은 성능을 보였다. 세 파일 시스템에서 거의 비슷한 성능을 보였으나 FAT 파일 시스템이 미세하게(200μs) 우수하였다. Set Query는 인덱스가 없는 필드에 대한 전체 레코드를 스캔하는 쿼리 종류이다. XFS와 Ext2 파일 시스템은 전체 테이블을 스캔하기 위해서 i-node의 i_block 구조[10](데이터 블록 인덱스 리스트)를 참조해야 하는데 간접 블록에 접근할 때에는 추가 I/O가 발생한다. 반면 FAT 파일 시스템은 FAT 체인만 따라가며 전체 테이블 스캔을 비교적 간단히 진행할 수 있는 장점이 존재한다. 따라서 전체 테이블 스캔에는 FAT 파일 시스템이 다소 유리한 것으로 보인다.

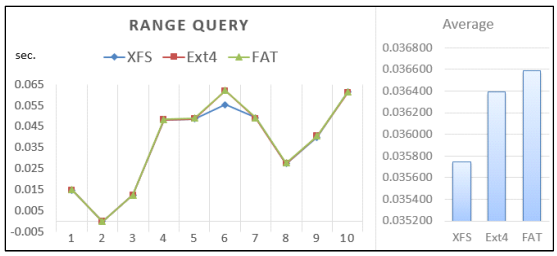

Fig. 3 Performance of Range Query

Range Query는 기본키 필드에 대한 범위 검색 쿼리로서 그림 3과 같은 실험 결과를 보였다. 검색 Range는 100개 범위에서 100만 개(테이블 전체)까지를 커버하게 실험하였다. 실험에서 비교 파일 시스템들은 거의 유사한 성능을 보였다. 하지만 Point Query에서 보듯이 인덱스를 사용한 검색이므로 XFS와 Ext4가 상대적으로 우수하였다. SQLite3는 Range Query에 B^+-Tree를 이용한다. Range Query를 수행하기 위해서는 먼저 Point Query를 수행한 후 범위 내에서 순차 블록 접근을 수행하게 된다. XFS나 Ext4 파일 시스템의 경우 직접 접근이 가능한 데이터 블록 인덱스 구조로서 Point Query에 유리하다.

특히, XFS 파일 시스템은 I/O 처리 효율을 높이기 위한 추가 기능을 갖추고 있어 대용량 Range Query에서 Ext4 파일 시스템보다 비교적 우수한 성능을 보이는 것으로 판단된다. XFS 파일 시스템은 높은 처리량이 필요한 응용의 경우 캐싱을 하지 않고 직접 처리할 수 있는 직접 I/O 기능을 지원한다[7].

그림 4에서 Multi-Condition Query의 실험 결과를 보이고 있다. Multi-Condition Query는 인덱스가 없는 복수 필드에 대한 다중 조건 질의이다. 인덱스가 없는 경우 FAT 파일 시스템은 테이블의 구성 데이터 블록을 FAT 체인을 따라 단순히 스캔하면 되기 때문에 조금 더 나은 결과를 보인다. 하지만 XFS나 Ext4 파일 시스템은 그림 2의 Set Query와 같이 파일 시스템의 복잡성에 대한 성능 저하가 발생하는 것으로 보인다.

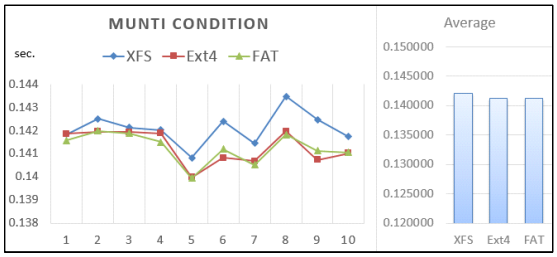

Fig. 4 Performance of Multi-Condition Query

그림 5는 Join Query의 실험 결과를 보인 그래프이다. 이 실험은 100만 개의 레코드를 가진 두 테이블 사이에서 조건절을 수행하여 평균 100개의 레코드들의 Join을 수행한다. 이 쿼리는 참조하는 테이블의 외래키 값에 대하여, 참조받는 테이블의 기본키 값을 Point Query로 조회하는 것이 주된 수행 내용이다. 이어서 조인 결과 테이블이 생성된다.

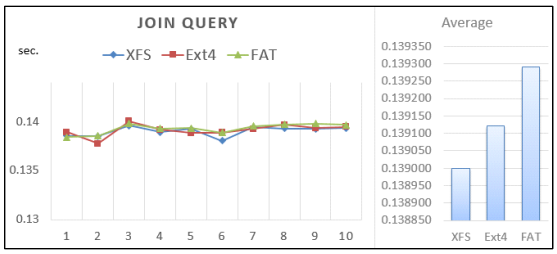

Fig. 5 Performance Join Query

이 실험에서 XFS 파일 시스템이 가장 우수한 성능을 보였고 이어서 Ext4의 성능이 나았고, FAT의 성능이 상대적으로 가장 낮았다. Join Query도 인덱스를 이용하는 질의이다. 조인의 나머지 수행 내용은 파일 시스템에 상관없이 진행되는 작업들이다. 따라서 조인 쿼리의 성능은 그림 1의 Point Query와 비슷한 성능 추세를 보였다. 해당 외래키 값의 기본키 레코드를 찾는 성능이 Join Query 성능의 결정 인자이기 때문으로 판단된다.

Fig. 6 Average Performances of File Systems

그림 6에서 각 검색 쿼리 별 파일 시스템의 성능과 파일 시스템 별 검색 쿼리 성능을 보이고 있다. 먼저 검색 쿼리 별 성능에서 Point Query가 가장 성능이 좋았고 다음으로 Range Query, Join Query 순이었다. 인덱스를 사용하지 않는 Set Query와 Multi-Condition Query는 성능이 상대적으로 좋지 않았다.

다음으로 파일 시스템 별 쿼리 성능이다. 여기서 Multi-Condition 쿼리 결과와 Set Query는 유사 쿼리 결과를 중복 누적할 수 있어 하나만 반영하였다. 결과는 전반적으로 XFS 파일 시스템이 우수하였고 이어서 Ext4, FAT 파일 시스템 순이었다.

V. 결 론

이 논문에서는 SQLite3 데이터베이스의 파일 시스템 별 검색 쿼리 처리 성능을 실험하고 비교해 보였다. 실험 결과와 평균값을 제시하여 파일 시스템에 따른 SQLite3의 검색 쿼리 성능을 평가할 수 있는 근거 자료로 사용할 수 있도록 하였다.

실험에서 확인한 결과는 먼저 Point Query와 Range Query에서의 파일 시스템 종류별 성능의 차이이다. 이 차이는 작지만 분명히 구별되었다(XFS:FAT=1:1.04). 이 쿼리들은 SQLite3의 기본키 인덱스를 이용하는 쿼리로서 XFS가 가장 우수하였고 이어서 Ext4, 그리고 FAT 파일 시스템 순이었다. 이것은 특정 블록을 찾거나, 그 후 연속해서 후속 블록을 순차적으로 찾는데 XFS 파일 시스템이 우수한 성능을 보인다는 뜻이다.

다음으로 Set Query와 Multi-Condition Query의 성능 역시 큰 성능 차이는 없었지만 파일 시스템별로 특성을 보였다(XFS:FAT=1.004:1). 이 쿼리들은 인덱스가 존재하지 않는 필드에 대한 검색으로서 FAT 파일 시스템이 비교적 우수하였다. 이것은 인덱스가 없어 테이블 전체를 스캔해야 하는 경우에는 구조가 간단한 FAT 파일 시스템이 유리하다는 것을 알 수 있다.

마지막으로 Join Query는 XFS, Ext4, 그리고 FAT 파일 시스템 순으로 성능이 우수하였다. 따라서 Join Query는 Point Query 성능과 비례한다는 것을 알 수 있었다.

실험을 통해 비교 파일 시스템 중 SQLite3와 같은 모바일 데이터베이스 환경에서 인덱스를 사용할 경우

XFS가 가장 우수하다는 것을 보였다. FAT 파일 시스템은 호환성이 뛰어나고 간단하지만 이러한 환경에 사용되기에 접합하지 않은 파일 시스템으로 파악되었다.

실험 결과를 바탕으로 향후 일반 필드에 인덱스를 사용한 경우와 비교 분석할 필요가 있다. 또한, SQLite3 이외의 모바일 데이터베이스에 대한 성능 실험과 결과 분석을 진행하는 것도 필요하다.

ACKNOWLEDGEMENT

This work was supported by the research grant of the Busan University of Foreign Studies in 2021.

REFERENCES

[1] N. Abbas, Y. Zang, A. Taherkordi, and T. Skeie, "Mobile Edge Computing: A Survey," *Journal of IEEE Internet of Things*, vol. 5, no. 1, pp. 450-465, 2018.

[2] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C. Hsu, "Edge Server Placement in Mobile Edge Computing," *Journal of Parallel and Distributed Computing*, vol. 127, pp. 160-168, 2019.

[3] What is Hybrid Cloud? [Internet]. Available: https://www.netapp.com/ko/hybrid-cloud/what-is-hybrid-cloud/.

[4] J. Choi, "Comparison of Update Performance by File System of Mobile Database SQLite3," *Journal of The Korea Institute of Information and Communication Engineering*, vol. 24, no. 9, pp. 1117-1122, Sep. 2020.

[5] Android Databases Performance Tests-CRUD [Internet]. Available: https://proandroiddev.com/android-databases-performance-crud-a963dd7bb0eb.

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

[7] The XFS File System [Internet]. Available: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/storage_administration_guide/ch-XFS.

[8] An introduction to Linux's EXT4 filesystem [Internet]. Available: https://opensource.com/article/17/5/introduction-Ext4-filesystem.

[9] A. Naohiro and K. Kenji, "File Systems are Hard to Test – Learning from Xfstests," *IEICE Transactions on Information ans Systems*, vol. 102, no. 2, pp. 269-279, 2019.

[10] Ext4 Disk Layout [Internet]. Available: https://Ext4.wiki.kernel.org/index.php/Ext4_Disk_Layout.

최진오(Jin-Oh Choi)

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