Hanbit eBook

Realtime 24

실무 예제로 다가가는 MySQL 쿼리 작성

MySQL 퍼포먼스 최적화

성동찬 지음

한빛미디어 Hanbit Media, Inc.

MySQL 퍼포먼스 최적화

: 실무 예제로 다가가는 MySQL 쿼리 작성

지은이_ **성동찬**

절대 깨어지지 않는 고가용성 데이터 시스템을 만드는 것이 목표인 DBA다. KTH에서 5년 간 '개발+DBA' 업무를 수행하였으며, 현재는 카카오에서 DBA로 활동하고 있다.

- 블로그 : http://gywn.net
- 트위터 : @gywndi

MySQL 퍼포먼스 최적화 실무 예제로 다가가는 MySQL 쿼리 작성

초판발행 2013년 3월 29일

지은이 성동찬 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 6월 24일 제10-1779호
ISBN 978-89-6848-602-9 15000 / **정가** 9,900원

책임편집 배용석 / **기획** 김병희 / **편집** 김연숙
디자인 표지 여동일, 내지 스튜디오 [밈], 조판 김현미
마케팅 박상용, 박주훈, 정민하

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanb.co.kr / **이메일** ask@hanb.co.kr

Published by HANBIT Media, Inc. Printed in Korea

지금 하지 않으면 할 수 없는 일이 있습니다.
책으로 펴내고 싶은 아이디어나 원고를 메일(ebookwriter@hanb.co.kr)로 보내주세요.
한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.

저자 서문

MySQL은 예전부터 있었고 지금도 해외에서는 잘 활용되고 있는 데이터베이스다. 하지만 유독 국내에서는 규모가 작은 서비스에만 적당한 데이터베이스로 치부되어왔다. 필자도 처음에는 서비스에 따른 데이터베이스 선정 가이드를 작성할 때 아무 생각 없이 "데이터 500만 건 미만에서만 MySQL을 사용한다"라는 어이없는 결론을 내리기도 했다. 과연 MySQL의 성능이 뒤떨어지는 것일까? 대답은 "NO!" MySQL로 서비스를 운영하면서 느낀 점은 MySQL의 특성을 모르고 사용을 해왔기 때문에 성능이 좋지 않은 것이지, 다른 데이터베이스에 비해 대용량 데이터 처리 성능이 뒤떨어지는 것은 아니었다는 사실이다.

최근 SNS 서비스가 화두가 되면서 MySQL이 급부상하였다. 구글에서는 이미 적극적으로 MySQL을 사용하고 있으며, SNS의 대표주자인 트위터 역시 MySQL을 기반으로 Gizzard라는 데이터 스토어를 만들어 사용하고 있다. 대형 SNS 업체들이 왜 MySQL을 선정하여 활용하고 있을까?

SNS 데이터의 가장 큰 특징은 엄청난 속도로 데이터가 누적되며 개인화된 데이터가 상당 부분 차지한다는 것이다. 갈수록 비대해지는 데이터를 상용 라이선스로 해결하기에는 분명 엄청난 비용이 소요된다. 상용 DBMS(Database Management System)을 사용하면 하드웨어 가격보다 소프트웨어 라이선스 가격이 몇 배 이상인 경우가 대부분이라 SNS 데이터 처리를 위해 상용 DBMS를 사용하는 것은 상당히 부담스럽다. 게다가 커뮤니티 성격이 강한 SNS 데이터 특성상 결제 데이터만큼 민감한 데이터도 아니다.

이 책에서는 필자가 MySQL로 서비스를 운영, 설계, 튜닝하면서 적용한 사례를 바탕으로 MySQL의 장점과 성능 최적화 방법을 설명할 것이다. MySQL을 사용하기에 앞서 반드시 알아야 할 항목부터 대용량 데이터를 수집/처리하기 위한 방법까지,

개인적인 경험에 근거하여 정리하였다. 원론적인 데이터베이스 분석이나 설명은 하지 않았지만 도서 내용을 이해하는 데 어려움은 없을 것이다. 필자는 소설책 읽듯이 편안한 마음으로 읽고 이해한 후 MySQL을 사용했으면 하는 마음으로 집필하였다. MySQL을 사용하는 데 도움이 되기를 바란다.

매번 늦은 원고 집필로 고생을 하는 한빛미디어 관계자 분들께 죄송한 마음에 앞서 처음 책을 쓰는 두려운 상황 속에서 용기를 주고 도와주신 분들께 진심으로 감사하다는 말을 전하고 싶다. 아울러 늦은 밤 퇴근해 들어오는 나에게 웃음을 잊지 않고 현관문 앞에서 빼꼼 얼굴을 내밀며 반갑게 맞이해주는 내 아기 효주(gywn)와 평일에 많은 시간을 함께 보내지 못함에도 언제나 나를 응원해주고 용기를 북돋아주는 아내에게 깊은 감사를 전한다.

집필을 마치며

지은이 **성동찬**

대상 독자

초급 | **초·중급** | 중급 | 중·고급 | 고급

다른 DBMS로 개발했던 개발자나 이제 막 IT 문턱으로 넘어온 초급자, 그리고 MySQL에서 복잡한 쿼리 성능을 조금이라도 높이고 싶어 하는 독자를 위한 책이다.

이 책은 MySQL을 사용할 때 반드시 알고 있어야 할 사항을 간략하게 소개하고 사례를 통해 쿼리 작성 시 주의해야 할 것들을 설명한다. 데이터베이스의 학문적인 내용보다는 경험 사례를 바탕으로 소개함으로써 독자들이 필자가 겪었던 문제와 비슷한 문제에 봉착했을 때 해결 방안에 대한 영감을 얻는 데 중점을 두었다.

이 책은 실무에서 사용하는 용어를 사용했다. 초급자나 처음 MySQL을 접하는 독자라면 용어가 낯설거나 어색할 수 있지만, 필자가 사용한 용어는 실무에서 관용적으로 사용하는 것이니 익숙해지면 실무에 도움이 될 것이다.

실습 환경 및 도서 구성

실습 환경

다음 웹 사이트에서 각자의 테스트 운영체제에 맞는 MySQL 버전을 다운받아 설치한다. 필자는 MySQL 5.1 버전을 사용하였다.

- MySQL 다운로드 : http://dev.mysql.com/downloads/mysql/
- MySQL 설치 방법 : http://gywn.net/2011/12/mysql-installation-on-linux/

필자는 대부분의 테스트 및 실무를 CentOS 5.x 버전에서 수행하였으나 MySQL 설치 후에는 특별히 OS와 연관된 요소가 없으므로 선호하는 환경에서 설치하면 된다.

이 책은 필자의 경험을 바탕으로 집필하였으며, 이 책에 나오는 결과는 필자가 현장에서 직접 테스트하여 얻었지만 실습할 수 있도록 원본 데이터를 제공하지는 않는다. 현장에서 사용하는 데이터베이스의 데이터를 공개한다는 것이 얼마나 위험한 일인지 잘 알고 있을 것이다.

도서 구성

1장에서는 MySQL에 대한 경험을 공유하기 전에 반드시 짚고 넘어가야 할 특성을, 2장에서는 최적화에 앞서 쿼리 성능을 진단할 수 있는 방안에 대해서 설명한다. 3장과 4장에서는 쿼리 작성 시 주의해야 할 사항에 대해서 설명하고, 5장에서는 쿼리 튜닝을 넘어서 스키마 레벨에서 최적화할 수 있는 팁에 대해서 설명한다. 마지막으로 6장에서는 스토리지 엔진을 변경하여 최적화한 사례를 소개한다.

한빛 eBook 리얼타임

한빛 eBook 리얼타임은 IT 개발자를 위한 eBook입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾는 것도 쉽지 않습니다. 또한 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1. **eBook Only - 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다.**

   500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 수 있는 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내는 선배, 전문가, 고수 분에게는 보다 쉽게 집필할 수 있는 기회가 될 수 있으리라 생각합니다. 또한 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 하고 있습니다.

2. **무료로 업데이트되는, 전자책 전용 서비스입니다.**

   종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나 저자(역자)와 독자가 소통하면서 보완하여 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3. 독자의 편의를 위하여 DRM-Free로 제공합니다.

구매한 전자책을 다양한 IT 기기에서 자유롭게 활용할 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해 독자 여러분이 언제 어디서 어떤 기기를 사용하더라도 편리하게 전자책을 볼 수 있도록 하기 위함입니다.

4. 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다.

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 힘든 전자책이 아니라, 전자책의 환경에 가능한 한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가도록 하겠습니다.

차례

1 | MySQL의 특징

MySQL의 성능을 운운하기에 앞서 MySQL 솔루션의 특성을 정확하게 인지해야 한다. 사용하는 시스템의 특성을 전혀 모르고 성능을 운운하는 것은 눈을 가리고 눈길을 빠르게 질주하는 것과 같다.

이 장에서는 MySQL의 전체적인 모습과 MySQL을 사용하기 전에 반드시 알아야 할 사항에 대해서 살펴본다.

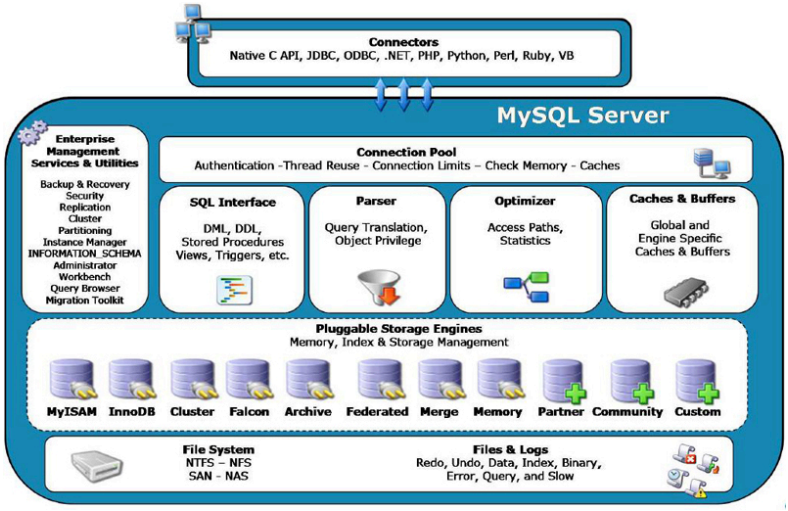
1.1 MySQL은 전체적으로 어떻게 생겼나?

MySQL을 잘 사용하려면 먼저 MySQL의 전체적인 구조를 이해해야 한다. 데이터베이스Database, DB는 기능에 따라 여러 항목으로 나눌 수 있겠지만 여기서는 '서버 엔진'과 '스토리지 엔진', 두 가지로 나누어서 설명할 것이다.

두 엔진을 간략하게 정리하면 서버 엔진Sever Engine은 클라이언트의 요청을 받아 SQL을 처리하는 DB 자체의 기능적인 역할을 담당한다. 그리고 스토리지 엔진Storage Engine은 서버 엔진이 필요한 데이터를 물리 장치에서 가져오는 역할을 한다.

그림 1-1의 중간 부분(SQL Interface, Parser, Optimizer, Cache&Buffers)이 서버 엔진이고 하단에 나열되어 있는 부분이 스토리지 엔진이다.

그림 1-1 MySQL 서버의 엔진 구조(출처: http://dev.mysql.com)



1.1.1 서버 엔진

서버 엔진은 사용자가 쿼리를 날렸을 때, DB가 SQL을 이해할 수 있도록 쿼리를 재구성하는 '쿼리 파싱'과 디스크나 메모리 같은 물리적인 저장장치와 통신하는 스토리지 엔진에 데이터를 요청하는 업무를 담당한다.

또한 스토리지 엔진에서 받아온 데이터를 사용자 요청에 맞게 처리하거나 접근 제어, 쿼리 캐시, 옵티마이저[01] 등의 역할을 수행한다. 즉, 물리적인 저장장치와 직접적으로 통신하는 역할을 제외하고 사용자와 MySQL 사이에서 발생하는 데이터 처리 프로세스의 대부분을 담당한다.

다음 세션에서 다룰 스토리지 엔진이 물리적인 저장장치에서 데이터를 읽어오는 역할을 수행한다면, 서버 엔진은 스토리지 엔진에서 가져온 데이터를 처리하는

01 SQL이 들어왔을 때 연관된 데이터를 빠르게 찾아내기 위해 최적의 실행을 유도하는 DB 내부의 기능이다.

역할을 담당한다. Table Join, Group By, Order By와 같은 일반적인 SQL 처리부터 Function/Procedure, Trigger, Constraint 등과 같은 기능도 서버 엔진에서 수행된다.

1.1.2 스토리지 엔진

스토리지 엔진은 물리적인 저장장치에서 데이터를 읽어오는 역할을 담당한다. MySQL은 다른 DBMS(Database Management System)와는 다르게 스토리지 엔진이 플러그인 방식으로 동작한다(필요한 기능이 있다면 전원에 플러그를 꽂듯이 필요한 스토리지 엔진을 간단하게 설치하여 바로 사용할 수 있다). 플러그인 방식이기 때문에 여러 개의 스토리지 엔진을 설치하여 사용할 수도 있다(MySQL 5.5에서는 스토리지 엔진 8개가 기본적으로 설치되어 있다). 필요하다면 3rd Party 스토리지 엔진을 별도로 설치하여 사용할 수도 있다.

다음은 MySQL Community 버전(MySQL 5.5 기준)을 처음 설치했을 때 제공되는 스토리지 엔진이다.

- InnoDB
- MyISAM
- MRG_MYISAM
- BLACKHOLE
- CSV
- MEMORY
- FEDERATED
- ARCHIVE

서버 엔진은 스토리지 엔진의 API를 호출하며 실제 필요한 데이터를 요청하고 조작한다. 그래서 API에는 구현되어 있지만 스토리지 엔진에서는 지원하지 않는 경우도 있다. 예를 들어 InnoDB에서 인덱스 타입을 해시(Hash)로 정의하여 생성했는데, 실제 스토리지 엔진에서는 B-Tree 인덱스로 생성되는 경우[02]다.

---

02 최근 Maria 진영에서 카산드라(Cassandra) 클러스터와 통신할 수 있게 하는 카산드라 스토리지 엔진이나 풀텍스트 인덱싱 서버인 스핑크스(Sphinx)와 연동하여 데이터를 처리할 수 있는 스핑크스 스토리지 엔진을 예로 들 수 있다. MariaDB에 대한 내용은 "https://mariadb.org/"에서 볼 수 있다.

MySQL은 다른 상용 DBMS와는 다르게 DB 엔진에 완벽하게 최적화되어 있지 않다. 이는 다양한 스토리지 엔진에서 동작할 수 있어야 하기 때문이다. 그래서 데이터 처리가 일부 비효율적일 수도 있지만 확장성 면에서는 다른 어떠한 DBMS보다 유연하다. 지금도 새로운 플러그인이 개발되고 있기 때문에 어떤 DBMS보다 강력하다고 할 수 있다.

1.2 MySQL에서 스토리지 엔진이란 무엇인가?

MySQL은 다양한 스토리지 엔진을 제공한다. MySQL의 가장 강력한 특징 중 하나가 바로 스토리지 엔진의 다양성이다. 만약 스토리지 엔진별 특성을 이해하고 적재적소에 적용할 수 있다면 강력한 성능은 물론이고 장비 효율성도 높일 수 있다. 반대로 서비스 특성에 맞지 않는 스토리지 엔진을 사용하면 예기치 않은 문제가 발생할 수도 있다. 때로는 테이블 설계나 SQL 작성보다 스토리지 엔진 선정전략이 전체 쿼리 수행 시간이나 장비 사용 효율에 큰 영향을 미치기도 한다.

MySQL은 플러그인 형식의 스토리지 엔진을 지원하며 기본적으로 스토리지 엔진 8개 정도를 제공한다. 현재 사용 중인 스토리지 엔진은 다음과 같이 조회할 수 있다(필자는 MySQL 5.1로 테스트하였다. MySQL 버전에 따라 출력 내용이 다를 수 있다).

```
mysql> SHOW ENGINES;
+------------+---------+------------------------+--------------+-----+------------+
| Engine     | Support | Comment                | Transactions | XA  | Savepoints |
+------------+---------+------------------------+--------------+-----+------------+
| InnoDB     | YES     | Supports transactions,.. | YES          | YES | YES        |
| MRG_MYISAM | YES     | Collection of identica.. | NO           | NO  | NO         |
| BLACKHOLE  | YES     | /dev/null storage engi.. | NO           | NO  | NO         |
| CSV        | YES     | CSV storage engine     .. | NO           | NO  | NO         |
| MEMORY     | YES     | Hash based, stored in  .. | NO           | NO  | NO         |
| FEDERATED  | NO      | Federated MySQL storag.. | NULL         | NULL| NULL       |
```

```
| ARCHIVE    | YES       | Archive storage engine.. | NO           | NO   | NO         |
| MyISAM     | DEFAULT   | Default engine as of M.. | NO           | NO   | NO         |
+------------+-----------+--------------------------+--------------+------+------------+
```

플러그인 형식의 스토리지 엔진이기 때문에 3rd Party에서 제공하는 다른 스토리지 엔진(SolidDB, Sphinx)을 간단하게 설치할 수 있다.

이 책에서는 다양한 스토리지 엔진 중 MyISAM, InnoDB, Archive에 중점을 두어 설명할 것이다(MyISAM, InnoDB, Archive는 실무에서 많이 사용하고 있는 대표적인 스토리지 엔진이다). 표 1-1은 스토리지 엔진별 특성을 나타낸 것이다.

표 1-1 주요 스토리지 엔진의 특징

	MyISAM	InnoDB	Archive
스토리지 제한	256TB	64TB	None
트랜잭션	No	Yes	No
Locking 레벨	Table	Row	Row
인덱스	B-Tree	B-Tree	No
Cache	Index	Data/Index	No
파티셔닝	Yes	Yes	Yes
Cluster Index	No	Yes	No
Foreign Key	No	Yes	No

1.2.1 MyISAM 스토리지 엔진

MyISAM은 MySQL에서 가장 오래된 스토리지 엔진이다. 파일 기반 스토리지 엔진이며 데이터에 대한 키, 즉 인덱스만 메모리에 올려서 처리한다. 데이터는 메모리에 적재하지 않고 디스크에서 바로 접근한다. 트랜잭션을 지원하지 않고 테이블 단위 잠금Table Level Lock으로 데이터 변경을 처리한다. 따라서 특정 테이블의 여러 세션에서 데이터를 변경하면 성능이 상당히 저하된다. 특정 세션이 테이블의 데이터를 변경하는 동안 다른 세션은 앞선 변경 작업이 끝날 때까지 대기하기 때문이다.

그렇다고 MyISAM이 다른 스토리지 엔진에 비해 성능이 뛰떨어지는 것은 아니다. 텍스트 전문을 검색할 수 있는 '풀텍스트 인덱싱Fulltext Indexing'과 지리 정보를 처리할 수 있는 '지오메트릭 스파셜 인덱싱Geometric Spatial Indexing' 같은 좋은 기능도 제공한다. 단, 풀텍스트 인덱싱와 지오메트릭 스파셜 인덱싱을 사용하면 테이블 파티셔닝을 사용할 수 없다는 제약이 있다. MySQL은 애초에 저사양 서버에서 구동하기 위한 목적으로 고안된 스토리지 엔진이기 때문에 다양한 기법으로 데이터 사이즈를 줄인다. 그중 대표적인 방법이 프리픽스Prefix 인덱스 압축 기법으로 키 사이즈를 최소화한다.

여기서 잠깐_ 프리픽스 인덱스 압축 기법이란?

인덱스 첫 번째 값을 통째로 저장하고, 그 이후 변경 값을 차례로 인덱싱하는 방식이다. 다음과 같이 두 개의 문자열(또는 스트링)을 인덱싱한다고 가정해보자.

Ex) Perform과 Performance 데이터 인덱싱

첫 번째 문자열은 일반적인 인덱스와 마찬가지로 Perform을 키 값으로 가지고 있다. 이 상태에서 Performance라는 문자열을 인덱싱할 경우, 앞 단계에서 중복된 문자(Perform)는 참조해야 할 글자 수(7자)와 필요한 나머지 문자열(ance)을 키로 가지게 된다.

① Perform : 인덱싱 문자열(Perform)을 저장

② 7, ance : 중복된 문자열(Perform)의 글자 수를 저장한 후에 나머지 데이터를 저장

프리픽스 인덱스 압축을 사용하면 키 사이즈는 대폭 줄어들고 메모리 효율은 높아진다. 그러나 키 역순으로 데이터를 찾아야 할 경우에는 성능 이슈가 있다. 원하는 데이터를 찾으려면 이전 단계에 저장된 인덱스 내용이 필요하다. 따라서 MyISAM은 로그 수집 또는 단순 SELECT에는 적합하나 동시 데이터 처리에는 한계가 있다.

1.2.2 InnoDB 스토리지 엔진

MySQL에서 유일하게 트랜잭션을 지원하는 스토리지 엔진으로, 일반적인 서비스에 가장 많이 사용하는 엔진이다. '다중 버전 동시성 제어 메커니즘Multiversion Concurrency Control, MVCC'[03]을 제공하고 행 단위 잠금으로 데이터 변경 작업을 수행하기 때문에 연관이 없는 데이터를 다른 사용자가 변경할 수 있다.

MyISAM은 인덱스만 메모리에 올리지만 InnoDB는 인덱스와 데이터를 모두 메모리에 올린다는 것이 가장 큰 차이점이다. 메모리에 인덱스와 데이터가 적재되어 있기 때문에 메모리 버퍼 크기(InnoDB_Buffer_Pool_Size)가 DB 성능에 큰 영향을 미친다.

03 MVCC는 여러 개의 복제본을 이용하여 하나의 데이터를 처리함으로써 서로의 READ/WRITE를 방해하지 않도록 하는 방식이다.

여기서 잠깐_ '테이블 단위 잠금'과 '행 단위 잠금'의 차이

이 둘의 차이는 데이터 변경 시 어느 수준까지 락Lock을 걸어 다른 사용자가 데이터를 변경하지 못하게 유지할 것인가다.

화장실에 네 개의 칸이 있다고 가정해보자. 처음에는 네 칸 모두 사용할 수 있다. 이때 한 사람이 들어가서 화장실 문을 잠가버리면 그 칸은 다른 사람이 이용하지 못하는 상태가 된다. 물론 나머지 세 개의 칸은 여전히 사용할 수 있다. 바로 이러한 경우가 DB에서는 '행 단위 잠금Row Level Lock'이라고 이해하면 된다.

어느 날 화장실로 들어오는 파이프 관이 파열되어 물이 단수되는 사고가 발생하였다. 다시 화장실을 사용하려면 파열된 파이프 관을 고쳐야만 한다. 화장실을 고치러 온 수리공은 제일 먼저 화장실 문 앞에 '수리 중'이라는 팻말을 올려놓고, 수리가 완료되기 전까지는 화장실을 이용하지 못하도록 제한한다. 화장실 칸을 사용할 수는 있지만 다른 사람이 사용하지 못하도록 수리공이 화장실 전체 칸을 선점해놓는 것이 바로 '테이블 단위 잠금Table Level Lock'과 비슷한 상태라고 이해하면 된다.

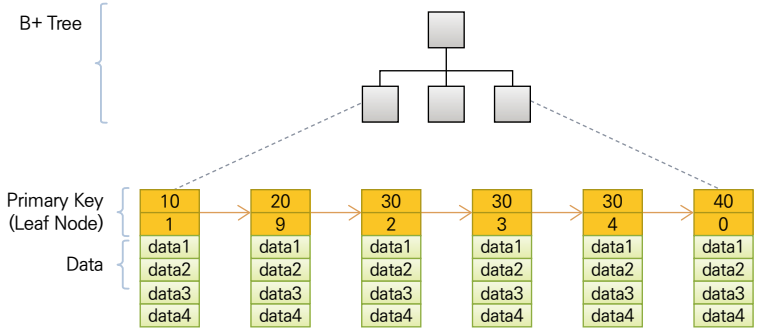
테이블 단위 잠금에서 특정 사용자가 데이터 입력/수정/삭제 작업을 수행하면 해당 테이블 전체에 잠금이 걸려 다른 사용자는 대기해야 한다. 따라서 테이블 변경 작업을 동시에 하면 성능이 상당히 저하된다.

이에 반해 행 단위 잠금은 변경하고자 하는 Row에만 락을 걸어 해당 Row를 제외한 다른 사용자의 데이터 변경 작업에 영향을 미치지 않는다. 테이블 단위 잠금과는 달리 테이블을 동시에 변경할 수 있다.

Primary Key[04]는 '클러스터 인덱스Cluster Index'[05]로 구성된다. 쉽게 풀어서 말하면 Primary Key 순서로 디스크에 저장되어 있다는 것을 의미한다. 그림 1-2처럼 Primary Key(주황색 칼럼)가 구성되어 있다면 실제 디스크에도 주황색 순서로 데이터가 저장되어 있다.

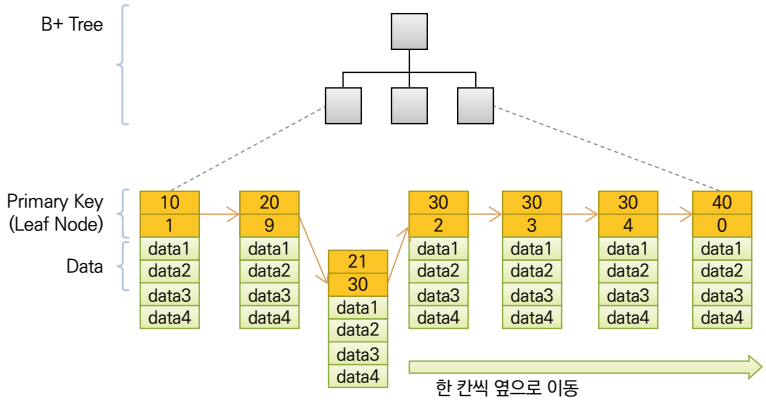
04 테이블의 식별자 역할을 하는 제약 조건으로 테이블에 하나만 설정할 수 있다.
05 클러스터 인덱스란 인덱스 순서로 데이터가 저장되어 있는 구조를 말한다.

그림 1-2 Primary Key를 이용한 클러스터 인덱스 구성



이 상태에서 Primary Key가 (21, 30)인 데이터가 유입되면 인덱스를 비교한 후 삽입된다. 이 경우 Primary Key가 20과 30인 데이터 중간에 삽입된다.

그림 1-3 Primary Key가 (21, 30)인 데이터가 유입된 경우의 클러스터 인덱스 구성



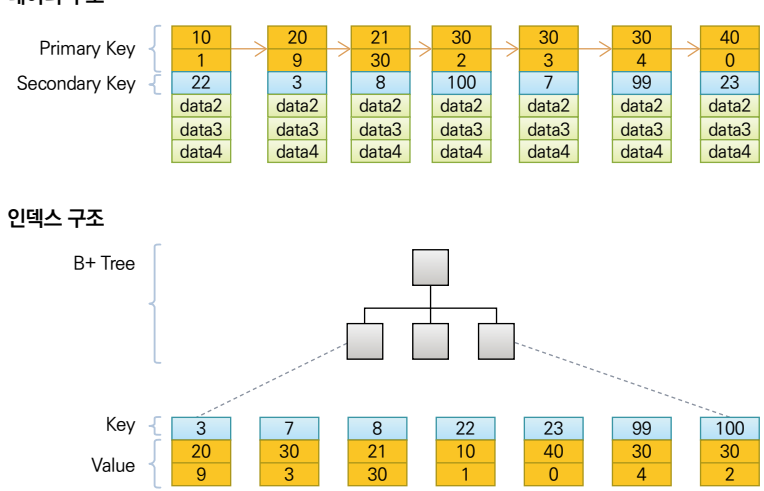
만약 Primary Key가 정의되어 있지 않으면 Unique Key[06]가 클러스터 인덱스로 구성된다. Primary Key와 Unique Key가 테이블 내에 정의되어 있지 않으면 내부

06 해당 칼럼에 입력되는 데이터가 유일하다는 것을 보장하기 위한 제약 조건이다. 한 테이블에 여러 개 설정할 수 있으며, Primary Key는 Unique Key에 포함된다.

적으로 6바이트 크기의 키를 만들어서 Primary Key로 사용한다. 물론 사용자가 볼 수 있는 키 값은 아니다. 하지만 MySQL의 데이터 복제Replication 기능을 사용하고 있는 중이라면 사용을 권고하지 않는다. 업데이트가 많은 서비스에서는 복제 데이터 동기화 지연이 발생할 수 있기 때문이다. Primary Key가 클러스터 인덱스로 구성되어 있기 때문에 데이터를 찾아가는 가장 빠른 경로는 Primary Key를 통해서 접근하는 것이다.

Primary Key 외의 보조 인덱스는 Primary Key를 값으로 가진다. 즉, 인덱스를 통해 Primary Key를 가져오고, 가져온 Primary Key로 실제 데이터에 접근한다. 이전 예제 그림에서 세 번째 칼럼에 인덱스를 추가했다고 생각해보자. 물론 인덱싱을 한 칼럼이 키가 된다. 여기서 재미있는 사실은 다른 DBMS와는 다르게 보조 인덱스는 Primary Key를 키에 상응하는 값으로 가진다는 것이다. InnoDB에서는 Primary Key가 데이터를 찾아가는 주요 값이기 때문에 보조 인덱스는 자연스럽게 Primary Key를 값으로 가지게 되는 것이다.

그림 1-4 Primary Key를 이용한 InnoDB의 데이터 구성



1.2.3 Archive 스토리지 엔진

로그 수집에 적합한 스토리지 엔진이다. 데이터가 메모리상에서 압축되고 압축된 상태로 디스크에 저장되기 때문에 행 단위 잠금이 가능하다. 단, 한 번 INSERT 된 데이터는 UPDATE와 DELETE를 사용할 수 없으며 인덱스를 지원하지 않는다. 모든 로그 수집에 적합한 엔진은 아니지만 원시 로그, 즉 가공이 한 번 필요한 데이터 수집에는 상당히 효율적이다. 테이블 파티셔닝도 지원하므로 일별 또는 월별로 데이터를 관리할 수 있다.

필자는 통계 서비스 기초 데이터를 백업할 때 Archive 스토리지 엔진을 사용하여 디스크 용량을 약 1/10 정도 절약한 적이 있다. 트랜잭션이 크게 중요하지 않은 통계 나 로그 분석 서비스의 기초 데이터 수집에 적합한 스토리지 엔진이다.

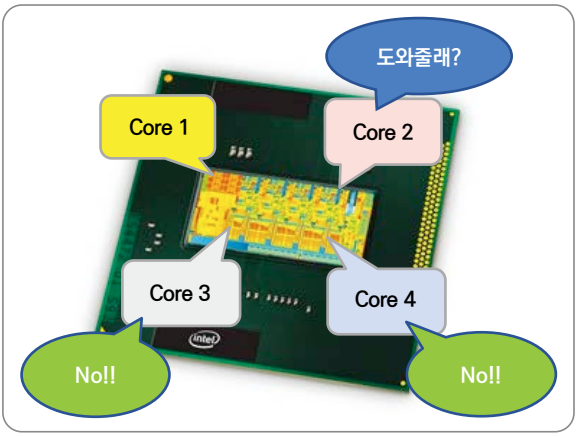
1.3 MySQL은 데이터를 어떻게 처리할까?

MySQL 쿼리를 최적화하기 전에 MySQL 쿼리 연산의 기본적인 특성에 대해서 살펴보자. 특성을 명확하게 알아야만 쿼리를 작성할 때 불필요한 연산을 최대한 줄일 수 있다.

1.3.1 MySQL에서는 모든 SQL을 단일 코어에서 처리한다

MySQL은 SQL을 병렬 처리하지 않는다. 3rd Party 스토리지 엔진을 설치하여 병렬 처리를 할 수는 있지만, 기본적인 스토리지 엔진은 단일 코어로만 데이터를 처리한다. 그림 1-5처럼 Core 2와 Core 4는 데이터를 처리하는 상태가 아니어도 다른 코어의 연산을 처리하지 않는다. 따라서 MySQL 입장에서는 CPU 코어 개수를 늘리는 Scale-Out 보다는 단위 처리량이 좋은 CPU로 Scale-Up 하는 것이 훨씬 좋다.

그림 1-5 단일 코어 처리



1.3.2 MySQL은 테이블 조인을 Nested Loop Join 알고리즘으로만 처리한다

Nested Loop Join은 선행 테이블 A의 조건 검색 결과값 하나하나를 테이블 B와 비교하여 조인하는 방식이다. 프로그램적으로 풀자면 for문 안의 for문과 유사하다. 결국 처리할 데이터가 적으면 수행 속도가 빠르지만 테이블 A나 테이블 B 중 하나라도 연산해야 할 데이터가 많아지면 쿼리 효율이 기하급수적으로 떨어진다.

다음은 Nested Loop Join 알고리즘의 유사 코드pseudo code다(Nested Loop Join으로 조인을 처리한다).

먼저 t1(①)은 '범위 스캔Range Scan'[07]으로 조인의 시작 데이터를 추출한다. t1에서 도출한 데이터는 t2의 인덱스(②)를 통해 연관된 데이터를 가져온다. 그리고 마지막으로 t3에서 데이터를 교체하는데, t3은 적당한 인덱스가 없는 경우에 실행된다. 이 경우 테이블 풀스캔을 하는데, t3 테이블에 있는 모든 데이터를 일일이 체크하고 적합한 데이터를 가져와서 최종적으로 클라이언트에 전달한다.

07 일정 범위에 있는 데이터를 추출하는 방법이다.

알고리즘 1-1 Nested Loop Join 알고리즘
(출처: http://dev.mysql.com/doc/refman/5.5/en/nested-loop-joins.html)

```
for each row in t1 matching range {                  // ❶
  for each row in t2 matching reference key {        // ❷
    for each row in t3 {
      if row satisfies join conditions,
      send to client
    }
  }
}
```

여기서 중요한 것은 Nested Loop Join은 루프Loop문이 세 개 겹친 형태라는 것이다. 알고리즘 1-1처럼 데이터를 처리하면 매번 데이터 접근 요청을 해야 하기 때문에 상당히 비효율적이다. 앞에서 모든 조인을 Nested Loop Join으로만 처리한다고 했는데, DB 내부에서는 이것보다 한 단계 업그레이드된 'Block Nested Loop Join' 방식(조인 버퍼[08] 개념을 도입)으로 처리한다. 테이블 조인 시 필요한 데이터를 메모리에 일시적으로 저장하여 효율적으로 데이터에 접근한다.

알고리즘 1-2 Block Nested Loop Join 알고리즘
(출처: http://dev.mysql.com/doc/refman/5.5/en/nested-loop-joins.html)

```
for each row in t1 matching range {
  for each row in t2 matching reference key {
    store used columns FROM t1, t2 in join buffer
    if buffer is full {
      for each row in t3 {
        for each t1, t2 combination in join buffer {
          if row satisfies join conditions,
            send to client
```

08 조인 버퍼Join Buffer는 Nested Loop로 테이블에 접근하기보다는 조인이 필요한 데이터들을 버퍼에 채우고 한 번에 테이블에 접근하여 조인을 효율적으로 개선한 방법이다.

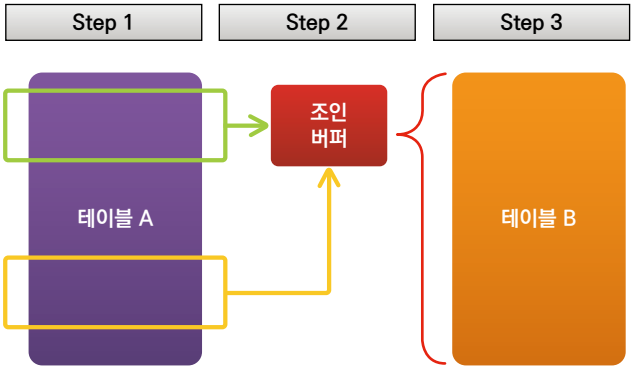
```
        }
      }
      empty buffer
    }
  }
}

if buffer is not empty {
  for each row in t3 {
    for each t1, t2 combination in join buffer {
      if row satisfies join conditions,
      send to client
    }
  }
}
```

그림 1-6을 이용하여 Block Nested Loop Join을 간단하게 설명하겠다. 다음과 같은 쿼리가 요청됐다고 가정해보자.

```
SELECT a.r1, b.r2
FROM TABEL_A a
INNER JOIN TABLE_B ON a.r1 = b.r2
```

그림 1-6 Block Nested Loop Join의 데이터 처리



먼저 테이블 A에서 조인할 데이터 대상을 찾는다(Step 1). 그리고 테이블 A에 있는 조인할 데이터를 조인 버퍼가 가득 찰 때까지 채운다(Step 2). 그림 1-6에서는 연두색 사각형이 조인 버퍼를 가득 채우는 데이터 양이다.

Step 2의 조인 버퍼가 가득 채워지면 테이블 B의 데이터를 스캔하면서 조인 버퍼에 있는 데이터와 매칭되는지 체크한다. 만약 데이터가 일치하면 조인 결과값으로 내보낸다. 테이블 B의 데이터를 스캔할 때는 테이블 풀스캔, 인덱스 풀스캔, 인덱스 범위 스캔 등으로 데이터에 접근한다. 즉, 테이블의 인덱스 구조 혹은 쿼리 형태에 따라 다양하게 접근한다.

조인 버퍼 안의 모든 데이터를 비교하는 첫 번째 과정이 종료되면 조인 버퍼를 비우고 노란색 부분에 해당하는 데이터 연산을 동일하게 시작한다. 이러한 과정은 조인 버퍼에 더 이상 데이터를 채울 수 없는 시점, 즉 테이블 A 조건에 해당하는 데이터를 모두 처리할 때까지 반복해서 수행한다. 여기서 테이블 B의 스캔하는 횟수는 '조인 버퍼에 데이터가 적재되는 횟수'와 동일하다.

MySQL에서는 단일 코어에서 Nested Loop Join 방식으로 데이터를 처리한다는

것을 꼭 기억하기 바란다.

지금까지는 MySQL을 사용하기에 앞서 반드시 알고 있어야 할 특성에 대해서 정리를 하였다. 2장에서 쿼리 프로파일링에 대해서 알아본 후에 3장부터는 실제 사례를 바탕으로 MySQL을 잘 활용할 수 있는 팁을 설명하겠다.