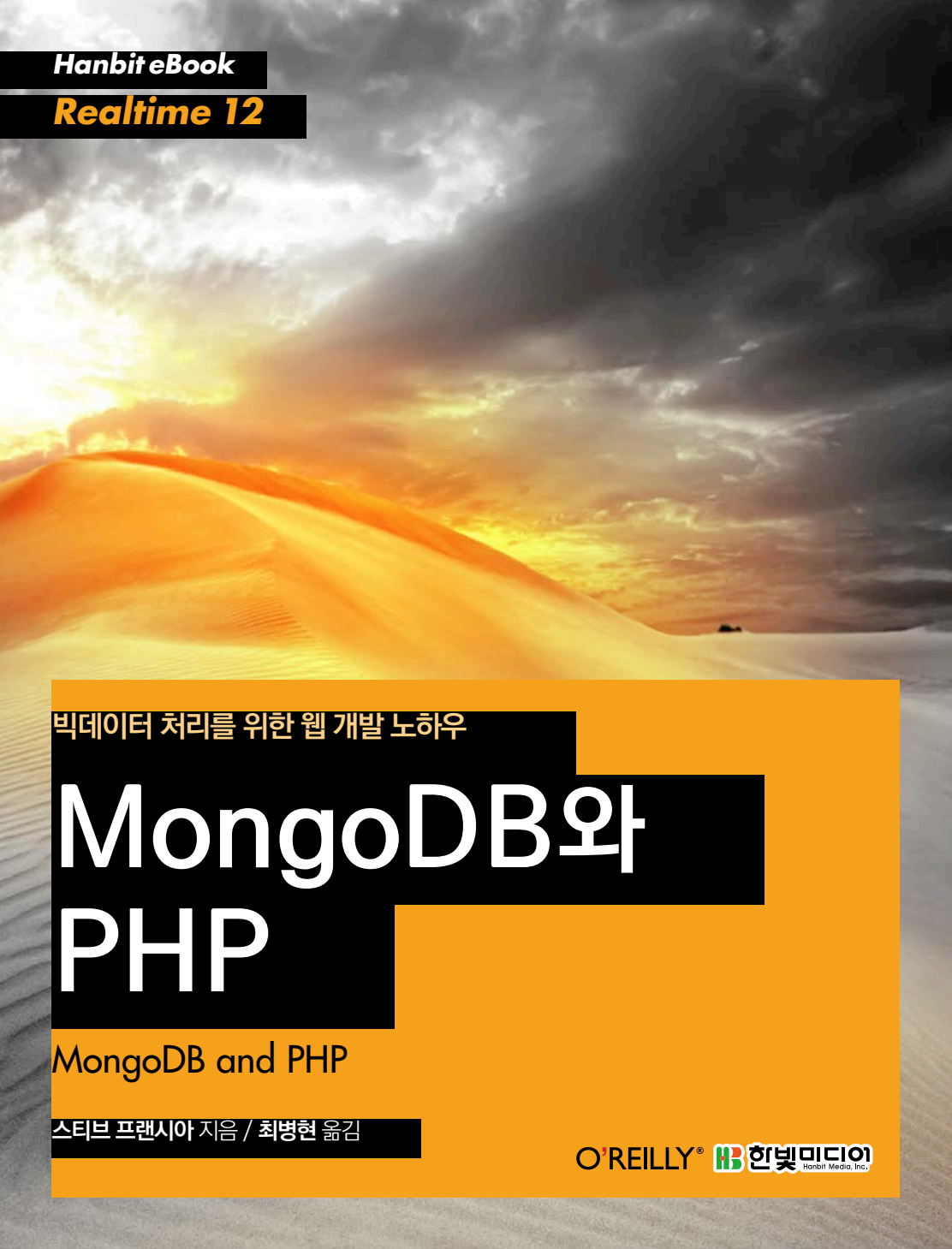
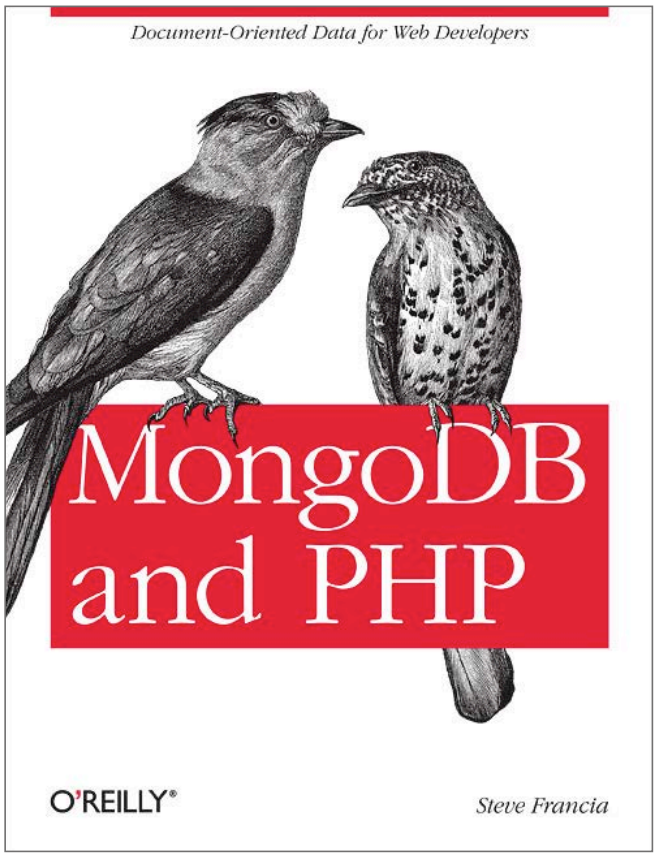

Hanbit eBook
Realtime 12
빅데이터 처리를 위한 웹 개발 노하우
MongoDB와 PHP
MongoDB and PHP
스티브 프랜시아 지음 / 최병현 옮김
O'REILLY
한빛미디어
Hanbit Media, Inc.




이 도서는 O'REILLY의
MongoDB and PHP의
번역서입니다.

빅데이터 처리를 위한 웹 개발 노하우

# MongoDB와 PHP

빅데이터 처리를 위한 웹 개발 노하우 **MongoDB와 PHP**

**초판발행** 2013년 1월 4일

**지은이** 스티브 프랜시아 / **옮긴이** 최병현 / **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** 978-89-7914-995-1 15560 / **정가** 9,900원

**책임편집** 배용석 / **기획** 김창수 / **편집** 김병희, 안선화
**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.**

지은이_ **스티브 프랜시아**

스티브 프랜시아는 10gen의 수석 솔루션 설계자로, MongoDB에 관한 모든 드라이버 개발, 통합, 전도, 웹과 문서를 담당하고 있다. 10gen에서 일하기 전에는 MongoDB를 사용한 첫 전자상거래 사이트(MongoDB를 이용한 초기 PHP 사이트 중 하나이기도 하다)인 OpenSky를 만들었다. 또한 Portero에서 최고정보책임자(CIO)와 최고운영책임자(COO)로, Takkle에서는 개발부사장으로 일했으며, Supernerd의 창립자이자 최고기술책임자(CTO)이기도 했다.

스티브 프랜시아는 오픈 소스를 사랑한다. 그는 MongoDB, Doctrine, Symfony2, Magento, Zoop를 포함한 수십 개의 오픈 소스 프로젝트에 기여했고, 스스로 몇 가지 프로젝트를 시작하기도 했다. 현재 세계 곳곳의 콘퍼런스에서 데이터베이스, 전자상거래, 빅데이터와 애플리케이션 개발에 대해 강연하며, 블로그(http://www.spf13.com)를 활발히 운영하고 있다. 브리검영 대학교에서 학위를 받았고, 동 대학에서 동적 웹 개발 과목을 개설하여 학생들을 가르치고 있다.

옮긴이_ **최병현**

서강대학교에서 컴퓨터학을 전공했다. 현재 번역 및 프로그래밍 프리랜서로 일하고 있다.

# 저자 서문

대략 십 년마다 한 번씩 굉장히 혁명적인 기술이 개발되어 우리가 사물에 접근하는 모든 방식을 근본적으로 바꾼다. 세상 그 자체가 변하는 것이다. 처음으로 인터넷 애플리케이션 개발을 시작한 1995년 무렵을 돌아보면, 당시에 필요했던 데이터는 상대적으로 단순했다. 그 후 십 년간은 별다른 변화가 없었다. 점점 더 많은 사람이 인터넷을 사용했고 결과적으로 더 많은 데이터 저장소가 필요하게 되었지만, 모든 사용자가 같은 데이터에 접근했기 때문에 캐싱을 통해 대체로 문제를 해결할 수 있었다. 그러나 근래 들어 소셜미디어가 열매를 맺으면서, 지난 30년간 잘 써왔던 접근법이 더는 충분치 않다는 점이 명백해졌다. 미래에는 모든 데이터와 경험이 개별 사용자에 맞춰져야 할 것이며 이 과정은 대규모로 이루어질 것이다. MongoDB는 이러한 필요를 충족시키고자 개발되었다. 오늘날의 애플리케이션을 위한 데이터베이스, 오늘날의 문제를 해결하기 위한 데이터베이스, 오늘날의 데이터 규모를 다루기 위한 데이터베이스로서, MongoDB는 애플리케이션 개발 접근법을 근본적으로 변화시킬 가능성을 가지고 있다.

지난 몇 달간 나는 여유 시간 대부분을 이 책을 쓰는 데 할애하였다. 그 기간을 참을성 있게 기다려준 아내와 네 아이에게 감사를 표하고 싶다.

저자 **스티브 프랜시아**

# 역자 서문

이 책은 독자가 MongoDB를 기초부터 탄탄하게 익힐 수 있도록 도와준다. 물론 이 책만 가지고도 MongoDB를 사용해 볼 수는 있겠지만, MySQL이나 Oracle 같은 기존 데이터베이스 대신 프로젝트에 사용하려면, 저자가 추천하듯 공식 홈페이지의 개발 문서를 읽어보기를 권한다.

이 책은 실제 프로젝트에 MongoDB를 적용할 때 얻을 수 있는 장점을 독자에게 알리기 위해 쓰였다. 이 책을 읽고 MongoDB를 한번 사용해보고 싶다는 생각이 들었다면, 저자의 목적은 달성된 것이다. 한 사람의 독자로서 말하자면, 저자는 독자가 빠르게 MongoDB를 사용해볼 수 있도록 도와주면서 자신의 경험담을 섞어 MongoDB의 장점을 맛깔나게 묘사했다.

익숙하지 않은 MongoDB를 팀 전체, 또는 일부가 일부러 공부하는 수고를 들여서라도 사용해볼 가치가 있는지 검토해보고 싶은 프로젝트 책임자들에게, 이 책이 도움될 것이다.

번역을 마치며

**최병현**

# 대상 독자 및 도서 구성

초급 | **초중급** | 중급 | 중고급 | 고급

이 책은 MongoDB의 기본(쿼리, 읽기쓰기 명령, 관리)은 물론 MapReduce, 샤딩과 같은 고급 주제도 다루고 있다. 판에 박힌 관계형 데이터베이스에서 벗어나 연산과 스케일에 최적화된 고성능 시스템을 이용해보자.

- MongoDB를 이용해 PHP 애플리케이션을 만들 때 필요한 도구들을 차근차근 배운다.
- Create, Read, Update, Delete (CRUD) 명령을 실행해보고, 쿼리문을 통해 데이터를 받아오는 방법을 배운다.
- MongoDB 셸을 통해 데이터베이스를 관리하고 데이터에 접근해 변경해본다
- 집합, 배열, 다중 문서 관련 함수들을 사용하여 동기적, 비동기적, 원자 연산을 실행해본다.
- PHP 커뮤니티 도구와 라이브러리 활용법을 배운다.
- 정규식, 집계, MapReduce, 리플리케이션, 샤딩을 사용해본다.

# 한빛 eBook 리얼타임

한빛 eBook 리얼타임은 IT 개발자를 위한 eBook 입니다.

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 내시는 선배, 전문가, 고수분에게는 보다 쉽게 집필하실 기회가 되리라 생각합니다. 또한 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 준비 중이며, 조만간 선보일 예정입니다.

2. **무료로 업데이트되는, 전자책 전용 서비스입니다.**

   종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정한 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 보완되고 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

### 3. 독자의 편의를 위하여, DRM-Free로 제공합니다.

구매한 전자책을 다양한 IT기기에서 자유롭게 활용하실 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해, 독자 여러분이 언제 어디서 어떤 기기를 사용하시더라도 편리하게 전자책을 보실 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 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 | 왜 Mongo인가?

1세대 닷컴 기업이 몰락한 원인 중 하나는 엄청난 개발비, 특히 서버 소프트웨어를 개발하는 데 사용된 개발비였다. 그러나 이 1세대 닷컴 기업의 잿더미에서 널리 사용될만한 새로운 오픈 소스 툴들이 모습을 드러냈고, 그것은 다음 세대 인터넷의 기반이 되었다. 2001년 여름 LAMP(Linux, Apache, MySQL, PHP)와 같은 새로운 약어가 회자되었는데, 이는 나아가 개발자 전 세대가 선택하는 플랫폼이 되었다. 마찬가지로 PHP와 MySQL은 결혼했다(달리 붙여 쓴 게 아니다). 이 둘은 영원히 같이할 운명처럼 보였다.

## 1.1 객체와 관계형 데이터 구조의 문제

PHP와 MySQL의 결합에는 문제가 하나 있었다. 템플릿 언어로 시작한 PHP는 발전함에 따라 점진적으로 객체를 받아들였다. 더 복잡한 애플리케이션 개발에 사용되면서 커지는 요구 사항을 맞추기 위해 지속적으로 변했고, 요구 사항이 점점 복잡해지면서는 PHP에서 사용하는 객체(또는 배열)와 MySQL(과 다른 관계형 데이터베이스)에서 사용하는 테이블, 열, 행 사이에서의 문제도 점점 자라났다. 이를 해결하기 위한 여러 툴도 만들어졌다.

이는 PHP에만 국한된 문제가 아니다. 몇십 년 동안 많은 이들이 객체를 관계형 데이터 구조로 변환하는 과정을 자동화하는 툴과 라이브러리를 만들었다. 그중에서 가장 유명한 종류는 객체 관계형 맵퍼ORM(Object Relational Mapper)라고 불린다. ORM은 SQL의 문제를 해결하기 위해 만들어졌으며, 그 홍보 문구는 다음과 같았다.

> "ORM을 사용하세요. 짜증나는 데이터 저장 관련 처리 과정을 친숙한 PHP 객체로 작업할 수 있습니다."

당시 각광받던 툴들은 약속했던 작업을 상당 부분 처리하긴 했지만, 어떤 것도 완벽하게 해내지는 못했다. 그 이유는 첫째, 사용자는 객체가 뜻하는 관계형 데이터베이스의 테이블, 열, 행 구조를 항상 기억해야만 했다. 둘째, ORM을 사용하는 데는 수고가 많이 들었다. ORM은 애플리케이션 개발을 굉장히 복잡하게 만들었고, 그 과정에 간접비용이 더 많이 드는 데다 SQL의 기능도 일부만 지원했다. 기능이 추가되면서 ORM을 배우는 것은 시간을 많이 잡아먹는 일이 되어, 차라리 SQL을 배우는 게 훨씬 빠를 정도가 되었다. 말하자면, ORM은 SQL의 문제를 많은 부분 해결했지만, 그 자체로 새로운 문제를 만들었던 것이다.

## 1.2 ORM의 문제

ORM의 목적은 단순했지만 그 문제는 절대 단순하지 않았다.

### 1.2.1 끔찍하게 복잡한 ORM

Propel[01]과 Doctrine[02]은 가장 유명한 PHP용 ORM이다. Propel은 액티브 레코드 모델을, Doctrine은 하이버네이트hibernate 모델을 따랐다. 두 프로젝트 모두 상당히 규모가 컸고, 수만 줄의 코드를 포함했다. Doctrine은 SQL과 비슷한 자체 쿼리 언어인 DQL을 제공했는데, Doctrine을 사용하기 위해서는 SQL과 DQL를 모두 알아야 했다.

### 1.2.2 ORM의 떨어지는 성능

ORM의 핵심 목표는 개발자 편의 제공이다. 이 말은 ORM이 DB의 테이블, 열, 행을 개발자가 사용하는 언어의 객체 형태로 변환하도록 만들어졌다는 의미다. 이를 처리하기 위한 가장 흔한 접근법은 '액티브 레코드active record'다. 액티브 레코드는 사용하기가 아주 쉽지만, 바로 그 편의성의 구현 때문에 성능이 매우 떨어진다.

---

01 http://propelorm.org/
02 http://www.doctrine-project.org/

이러한 현상은 특히 PHP에서 심하게 나타난다. 일반적으로 적은 양의 데이터를 다룰 때 액티브 레코드를 사용하면 상당히 빠르지만, 부하나 데이터 크기가 커지면 성능이 급격히 떨어진다. 예를 들면, '루비 온 레일즈Ruby on Rails(ORM 역할을 하는 웹 프레임워크)'는 확장할 수 없어서 프로토타입 환경으로서나 쓸만하다는 비판을 듣는다. 이는 정확한 비판이지만, 좀 더 구체적으로 살펴봐야 한다. 루비 온 레일즈에서 확장할 수 없는 부분은 컨트롤러나 뷰가 아니라 바로 액티브 레코드 층이다. ORM은 런타임 중에 간접비용이 드는 계층을 추가할 뿐만 아니라 엄청난 메모리를 잡아먹는다.

### 1.2.3 ORM이 망가트린 SQL

ORM은 SQL을 단순히 숨긴 것이 아니라, SQL의 기능을 가장 기초적인 부분만 남기고 벗겨 냈다. ORM 덕분에 객체를 읽고 쓰는 연산과 흔히 CRUDCreate, Read, Update, Delete라 부르는 기본 연산은 아주 간단해졌지만, SQL의 고급 기능은 대부분 지원하지 못했다. 믿지 못하겠다면, left outer join 연산이나 데이터 집합의 평균을 구하는 집계 함수를 ORM으로 사용해보라. 많은 ORM이 트랜잭션[03] 기능조차 지원하지 않고 애플리케이션에 책임을 떠넘긴다.

### 1.2.4 복잡한 구조

ORM과 관계형 데이터베이스의 성능 결함을 해결하기 위해 MemCache가 개발되었다. 데이터 검색 속도를 상당히 효과적으로 높였기 때문에 MemCache는 업계에 빠르게 퍼졌고, 곧 확장이 필요한 애플리케이션은 물론 속도 향상을 필요로 하는 애플리케이션에서도 필수 요소가 되었다. 단일 기술로는 사실상 가장 높은 비율로 사용되는 MemCache는 거의 모든 웹 사이트와 인터넷 애플리케이션이 사용하고 있다.

---

03 (역자주) 여러 연산을 묶음 단위로 처리하는 기능

하지만 데이터에 빨리 접근하는 데는 용이하나, 애플리케이션을 간소화하는 데 MemCache는 별 도움이 되지 않는다. MemCache가 추가되면서, ORM이나 애플리케이션은 객체를 테이블, 열, 행으로 변환하는 작업은 물론, 객체를 키 값(또는 키 값 집합) 뒤에 저장하고, MemCache 데이터 입출력 및 갱신 타이밍을 관리하여 RDBMS와의 동기화를 유지하는 추가적인 로직이 필요해졌다. 이 작업은 쉬운 일이 아니라 보통 달갑잖은 문제를 남겨둔 채로 "나쁘지 않은" 상태에서 끝나곤 한다.

### 1.2.5 PHP 작업의 대부분은 CRUD 연산이다

ORM의 문제점들을 살펴보면서, 프로그래머들이 도대체 왜 ORM을 쓰는 것인지 의문스러울 것이다. 사람들이 많은 단점에도 ORM을 사용하는 이유는 단 한 가지다. PHP 애플리케이션은 대체로 CRUD 애플리케이션이다. 관계형 데이터베이스가 제공하는 풍부한 기능을 PHP 애플리케이션이 전부 사용하는 경우는 드물기 때문에, 그 기능들을 포기하는 대가로 데이터에 좀 더 쉽게 접근할 수 있다면 해볼 만한 일인 것이다. 게다가 별다른 선택지가 없기도 하다. 간단한 프로젝트에서라면 SQL을 코드에 직접 쓸 수도 있겠지만, 그럴 경우 디버깅이 어려울뿐더러 보안을 유지하기는 더욱 어렵다. 미숙한 개발자들이 별다른 처리 없이 SQL에 변수를 바로 넘기곤 해서, PHP는 SQL 인젝션 공격에 취약하기로 유명하다.

## 1.3 연산에 최적화된 MongoDB

애플리케이션 개발자들이 데이터 저장소를 실제로 사용하는 연산에 최적화시키면 어떨지 상상해본 적 있는가?

2007년, 엘리엇 호로비츠Eliot Horowitz와 드와이트 메리맨Dwight Merriman(10gen 공동 창립자)이라는 훌륭한 개발자 두 사람이 바로 그 일에 착수했다. 둘 모두

DoubleClick[04]에서 일했는데, 드와이트는 CTO로 엘리엇은 엔지니어로 재직했다. 그곳에서 초당 수십만 개의 광고를 뿌리고 추적하는 시스템을 만들면서, 엘리엇은 큰 용량의 데이터를 다루고 많은 연산을 처리하는 동시에 데이터의 확장도 가능한 시스템을 기존의 데이터베이스 기술로 만들고자 할 때 부딪치는 어려움을 상세하게 알고 있었다. 또한 현재의 관계형 데이터베이스에 부족한 것이 무엇인지도 알고 있었다. 그들은 연산과 확장에 최적화된 데이터베이스 개발에 착수했다. 그것이 바로 MongoDB다.

MongoDB는 기능성을 유지하면서 수평적 확장을 허용하는 동시에 개발자가 최대한 간편하게 사용할 수 있도록 한다는 설계 철학 위에 만들어졌다.

MongoDB 개발을 시작하면서 두 사람은 관계형 데이터베이스가 제공하는 기능을 살펴, 어떤 기능이 없어도 되며 어떤 기능이 여전히 개발자들에게 필요할지를 고민했다. 관계는 수평적 확장을 불가능하게 만들고 다중 테이블 트랜잭션으로는 분산 클러스터에서 처리하기 힘들었다. 그들은 어떻게 하면 개발자가 데이터베이스를 더 쉽게 사용할 수 있을지 고민했다.

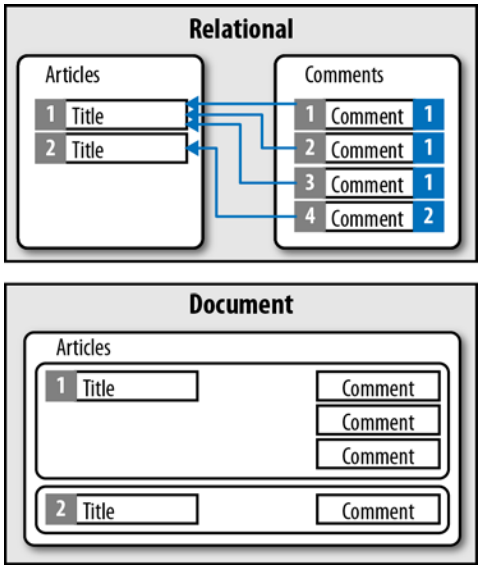
키 값 저장 방식은 훌륭하지만, 대부분의 경우 더 많은 기능이 필요했다. 가끔은 키 값이 아닌 다른 값으로 데이터에 접근할 수 있어야 했다. 오늘날 대부분의 언어가 객체를 통해 연산하는데, MongoDB가 객체를 닮은 데이터 구조를 사용하면 어떨까?

04 http://www.google.com/doubleclick/

### 1.3.1 MongoDB는 문서 데이터베이스다

엘리엇 호로비츠와 드와이트 메리맨은 MongoDB를 '문서 데이터베이스 document database'[05]의 일종으로 개발하기로 결정했다. 가장 높은 수준의 구성에서 MongoDB는 관계형 데이터베이스와 매우 비슷하지만, 데이터 자체를 좀 더 자세히 살펴보면 데이터가 저장되는 방식에 중요한 변화가 있다는 것을 알 수 있다. 데이터베이스, 테이블, 열, 행 대신 데이터베이스, 컬렉션, 문서로 저장되는 것이다(그림 1-1 참조).

**그림 1-1** 데이터 구성도 – 관계형 구성 VS 문서 기반 구성




05 (역자주) 문서 데이터베이스는 테이블과 같은 구조에 데이터를 저장하지 않고 느슨하게 정의된 문서에 저장한다. RDBMS는 테이블에 열을 새로 추가하려면 테이블 자체의 정의를 변경해야 한다. 따라서 널 값이라도 열이 기존의 모든 레코드에 추가된다. 이는 RDBMS의 엄격한 스키마 기반 설계로 인한 것이다. 그러나 문서를 사용하면 모든 문서를 변경하지 않고도 개별 문서에 속성을 새로 추가할 수 있다. 이는 문서 데이터베이스가 스키마를 사용하지 않도록 설계되기 때문이다.

### 1.3.2 문서 == 배열

문서 데이터베이스라는 부정확한 용어를 들으면 사람들은 흔히 PDF 파일이나 워드 문서를 떠올린다. 이때의 문서란 사실 PHP의 배열과 같다.

#### 데이터베이스

MongoDB가 데이터를 데이터베이스로 묶는 방식은 대부분의 관계형 데이터베이스가 하는 방식과 같다. 관계형 데이터베이스를 다뤄본 경험이 있는 사용자라면 MongoDB도 같은 방식으로 사용하면 된다. 관계형 데이터베이스 시스템 RDMBS(Relational Database Management System)에서 데이터베이스는 테이블, 저장된 프로시저, 뷰 등의 집합이다. 반면 MongoDB에서 데이터베이스는 컬렉션의 집합이다.

#### 컬렉션

컬렉션은 관계형 데이터베이스 양식에서 테이블에 해당하는 것이다. 대부분의 경우 컬렉션은 테이블처럼 생각하면 된다(테이블이라고 부르지만 않는다면). 컬렉션은 문서와 인덱스의 모음으로, 테이블과 마찬가지로 컬렉션에도 인덱스가 있다.

#### 문서

MongoDB에서는 주요소를 '문서'라고 부른다. 문서는 관계형과 직접적인 연관이 없으며, 관계형 데이터베이스의 테이블과 달리 미리 정의된 스키마가 없다. 또한 문서는 데이터가 저장되는 장소라는 점에서 부분적으로 행과 같고, 문서마다(테이블마다가 아니라) 스키마가 정의된다는 점에서 부분적으로 열과 같다.

문서를 쉽게 이해하는 방법은 그것을 다차원 배열의 일종으로 생각하는 것이다. 배열은 값들과 각 값에 해당하는 키 값 집합을 가진다. 값들은 그들 자신이 또 다른 배열이 될 수 있다. 실제 구현이라는 측면에서 보면, 하나의 MongoDB 문서는 하나의 JSON 배열이다. 문서는 객체를 비롯하여 배열이나 다차원 배열 같은 다른 PHP 데이터 타입과 굉장히 잘 맞는다.

PHP 배열 구조는 다른 어떤 데이터 타입보다 문서 구조와 관련 있는데, 거의 완벽하게 일대일로 대응된다. PHP 배열의 특성을 살펴보면, PHP 배열은 '키 => 값' 관계는 물론 열거형 키(0,1,2… 같은)도 허용한다. 두 가지 방식을 전부 한 배열에서 사용할 수 있을 뿐만 아니라, 두 키 값이 한 배열 안에서 공존할 수도 있다. 추가적인 특징을 보면, PHP는 순서 없는 배열을 허용하지 않는다. MongoDB는 데이터를 저장할 때 JSON을 사용하는데, JSON은 PHP 배열과 여러모로 다르다. 자바스크립트의 JSON 표기에 따르면, JSON은 서로 다른 두 구조인 리스트(순서와 키가 없는 값들)와 해시(키/값 짝들)로 이루어져 있다. 그러나 실제로 사용할 때는 이런 차이가 거의 드러나지 않는다.

### 1.3.3 CRUD연산에 최적화된 MongoDB

MongoDB는 연구소에서 작성된 게 아니라, 현실의 문제를 해결하기 위해 작성되었다. MongoDB의 연산 과정은 최적화되어 아주 효율적이다. 많은 노력을 통해 MongoDB는 두세 가지 측면에서 최적화되었다. MongoDB를 사용해보면, 문서 구조가 매우 효과적이란 것을 알 수 있을 것이다. 사용자는 문서 하나에 아주 많은 관련 데이터를 구조 그대로 저장하고, 정규화하거나, 쿼리문으로 받아올 수 있다. 관계형 데이터베이스를 이용할 때는 객체 데이터 하나를 받아오기 위해 열 개가 넘는 테이블에 접근해야 했지만, MongoDB에서는 많은 경우 단 하나의 문서만으로 해결된다. 대부분의 CRUD 연산이 매우 단순한 저장, 검색, 삭제 연산이 된다.

### 1.3.4 개발자를 위한 최적화된 인터페이스

MongoDB 문서가 사실상 하나의 PHP 객체나 배열이기 때문에, 새로운 문서를 생성하는 것이 간단하다. 그저 새로운 PHP 배열이나 객체를 생성하고 저장하기만 하면 된다. 이 책의 많은 부분에서 PHP로 MongoDB와 연동하는 다양한 방법을 설명할 것이다. 많은 개발자가 익숙한 관계적 사고방식을 수정해야 할 수도 있겠지만, MongoDB 인터페이스는 사용하기 편하고 매우 자연스럽게 느껴질 것이다. 인

터페이스는 대체로 예측한 대로 작동해서 훨씬 효율적으로 개발할 수 있다.

### 1.3.5 최적의 성능

MongoDB는 근본부터 철저히 고성능 데이터베이스로 설계되어, 비슷한 연산에서 관계형 데이터베이스에 비해 주목할 만큼 향상된 성능을 제공한다. 많은 애플리케이션의 성능을 상당히 개선할 수도 있다(20배 이상의 성능 향상도 드문 일이 아니다). 이는 MongoDB의 주요 연산이 빠를 뿐 아니라, 훨씬 더 간결하기 때문에 가능하다. 예를 들면, 관계형 데이터베이스에 블로그 포스터를 입력insert하는 경우 여러 테이블에 대한 insert 명령이 필요할 수 있다. 포스트 테이블에서 한 번, 태그 테이블에서 두세 번, 포스트-태그 관계 테이블에서 두세 번, 카테고리 테이블에서 한 번, 미디어 테이블과 연관 테이블에서 한 번씩 등과 같이 더 많은 테이블이 필요할 수도 있다. 똑같은 일을 MongoDB에서는 단 하나의 문서를 작성하여 처리할 수 있다.

더 간결하고 빠른 연산에 더해, MongoDB는 메모리 맵 파일도 자주 활용한다. 지나친 단순화일 수도 있겠지만, 본질적으로 MongoDB는 모든(또는 RAM이 감당할 수 있을 만큼의) 데이터의 읽기와 쓰기를 메모리 캐싱을 통해서 처리한다. MongoDB를 사용하면 대개는 MemCache를 굳이 사용하지 않아도 된다.

### 1.3.6 최적의 간결성

아무리 복잡한 구조의 데이터 객체라도, MongoDB는 완전히 최적화된 생성, 읽기, 수정, 삭제 연산을 제공한다. 이전 섹션에서 설명했듯 복잡한 join이나 다중 테이블 트랜잭션이 필요한 연산들은 많은 경우 훨씬 더 간단한 스키마로 처리할 수 있고 그 결과 연산과 모델 층을 훨씬 더 간소화할 수 있다. 그리고 캐시를 사용하면 데이터를 갱신할 필요가 없어서 애플리케이션 자체뿐 아니라 구조도 간결해진다.

### 1.3.7 일관성의 가치

MongoDB는 MySQL이나 PostgreSQL, 그 외 많은 관계형 데이터베이스와 마찬가지로 완전한 일관성을 가지는 데이터베이스다. 반면 NoSQL에 해당하는 많은 데이터베이스는 결과적 일관성을 가진다. 이는 MongoDB와 구별되는 점이다. 결과적 일관성을 가지는 데이터베이스는 '다중 마스터 데이터베이스'라고도 불린다. 이들 스스로는 완전한 일관성을 보장한다고 주장하지만, 이때의 '일관성'이란 재정의된 것일 뿐 기존의 정의에는 못 미친다.

결과적 일관성을 가지는 데이터베이스가 필요한 곳이 있긴 하지만, 대개의 개발자들은 이 데이터베이스를 사용할 때 어떤 기능을 포기하게 되는지 자각하지 못한다. 단순히 데이터 손실 문제만이 아니라 기능적 측면에서도 그렇다. 완전한 일관성을 가진 데이터베이스에서는 값을 쉽게 증가시킬 수 있고 충돌 걱정 없이 항목을 덧붙일 수 있다. 이런 연산이 MongoDB에서는 사소한 일인 반면, 결과적 일관성을 가진 데이터베이스에서는 애플리케이션에 엄청나게 많은 로직과 처리를 추가하지 않고는 불가능한 일이다.

이 차이를 분명히 보여주기 위해 간단한 예를 들겠다. 각 투표자(모든 사용자는 한 번만 투표할 수 있다)의 사용자명을 추적하고 총계를 낼 수 있는 간단한 투표 애플리케이션 개발을 가정해보자. 로직은 상당히 단순하다. 사용자명이 배열 안에 없으면, 총계를 증가시키고 그 사용자명을 배열에 덧붙인다. MongoDB에서 이 연산은 매우 단순한 연산이지만, 결과적 일관성을 지닌 데이터베이스에서는 수행하지 못한다.

# 2 | PHP, MongoDB, 그리고 사용자

이번 장에서는 MongoDB와 PHP를 이용하기 위한 기본적인 지식을 알아보겠다. 2장이 끝낼 때쯤이면, 드라이버를 설치하고 MongoDB를 데이터 저장소로 사용하는 PHP 애플리케이션을 작성하는 방법을 익힐 수 있을 것이다.

## 2.1 리눅스 또는 맥OS X에서 드라이버 설치하기

사용 환경에 따라 설치 방법은 달라질 수 있다. 운영체제는 PHP와 관련 있으므로 사용하는 운영체제에 대한 기본적인 이해가 필요하다. 다음에 설명하는 일반적 설치 방법이 다양한 사용자 환경에 대처하는 데 충분한 정보를 담았기를 바란다.

### 2.1.1 드라이버 확인하기

드라이버를 새로 설치하기 전에 이미 설치된 드라이버가 있는지 먼저 확인해야 한다. 점점 많은 리눅스 배포판이 MongoDB 드라이버를 기본 설치본에 포함하고 있다. 다음 명령은 MongoDB 드라이버 설치 여부를 확인할 수 있는 몇 가지 정보를 출력한다.

```
php --re mongo
```

만약 MongoDB 드라이버가 설치되어 있지 않다면, 다음과 같은 메시지를 볼 수 있을 것이다.

```
Exception: Extension mongo does not exist
```

### 2.1.2 드라이버 설치하기

PHP MongoDB 드라이버를 설치하는 방법에는 몇 가지가 있다. Zend Server 사

용자라면 바로 시작할 수 있을 것이다(Zend Server 자체로 MongoDB 드라이버를 포함하고 있기 때문이다). 어떤 배포판은 MongoDB 드라이버를 포함하는 자체 deb 패키지 또는 rpm 패키지를 지원한다. 이런 패키지를 이용할 수도 있겠지만, 추천하지는 않는다. 추천하는 방법은 PECL[01]을 사용하여 드라이버를 설치하는 것이다. 왜냐하면 모든 시스템에서 같은 방법을 사용할 수 있고 쉽게 최신 버전으로 업그레이드할 수 있기 때문이다.

물론 PECL이 미리 설치되고 환경 설정이 되어 있어야 한다. 이 방법들은 이 책의 범위를 넘어서므로 설명은 생략한다. 많은 배포판이 PECL을 포함하고 있지만 PECL이 설치되어 있지 않아 "command not found" 같은 메시지를 보게 되는 경우도 있을 것이다. 그럴 땐 각 운영체제에 따른 설치 가이드를 온라인상에서 쉽게 찾을 수 있으니 참고하기 바란다. 운영체제나 환경 설정에 따라서는 각각 설치할 때마다 "sudo pecl" 명령으로 관리자 권한을 부여해야 할 수도 있다.

PHP MongoDB 클라이언트 확장판은 다음 PECL 명령을 이용하여 설치할 수 있다.

```
pecl install mongo
```

모든 것이 제대로 작동했을 경우, 다음과 같은 메시지를 볼 수 있다.

```
Build process completed successfully
Installing '/usr/lib/php/modules/mongo.so'
install ok: channel://pecl.php.net/mongo-1.0.4
You should add "extension=mongo.so" to php.ini
```

다음 한 줄을 php.ini 설정 파일에 추가하면 모든 준비가 끝난다.

```
extension=mongo.so
```

01 http://pecl.php.net/

### 2.1.3 드라이버 업그레이드하기

드라이버 업그레이드는 조금 더 복잡하다. 시스템의 일관성이 상당히 중요하기 때문에 설치한 방식과 같은 방식으로 업그레이드를 진행해야 한다. 앞서 말했듯이 설치할 때는 PECL을 사용할 것을 추천한다. PECL를 사용하면 업그레이드 방법이 매우 단순해진다.

```
pecl update-channels
pecl upgrade mongo
```

이제 웹 서버를 다시 시작해서 새 확장판을 적용하면 된다.

## 2.2 윈도우에서 드라이버 설치하기

MongoDB는 윈도우를 완벽하게 지원하는 몇 안 되는 NoSQL 계열 솔루션이다. PECL 역시 윈도우에서 잘 동작하므로 이미 PECL이 설치되어 있다면 위 방식을 시도해볼 수 있다. 또는 MongoDB 프로젝트에서 배포하는, 미리 컴파일된 버전의 윈도우용 드라이버를 사용할 수도 있다. 이 드라이버는 "https://github.com/mongodb/mongo-php-driver/downloads"에서 다운받을 수 있다. 필요한 dll(스레드 세이프[02]한 버전이나 일반 버전)을 php 플러그인이 저장되어 있는 폴더에 넣고 편집기로 php.ini 파일을 열어 확장 부분에 추가한 파일을 적으면 된다.

AMP(Apache MySQL/MongoDB PHP)를 윈도우에서 설치하는 방법에는 여러 가지가 있지만, Uniform Server를 이용하는 방법을 추천한다. Uniform Server는 올인원 솔루션이고 귀찮은 설정 없이 설치할 수 있다. 대부분의 경우, 그저 압축을 풀고 실행하면 된다. Uniform Server 6가 지원하는 MongoDB 플러그인은 MongoDB 서버, MongoDB PHP 드라이버, 간단한 브라우저 기반 관리 프로그램인 phpMoAdmin을 제공한다. Uniform Server는 여러 서비스를 시작, 중단,

02 여러 스레드를 사용해도 안전하다는 의미다.

관리할 수 있는 윈도우 인터페이스도 제공한다. Uniform Server에 대한 더 자세한 정보는 해당 웹 사이트인 "http://www.uniformserver.com"에서 찾아볼 수 있다.

## 2.3 데이터베이스 연결하기

이제부터는 MongoDB를 설치하여 사용 가능하다고 가정하고 진행하겠다. 설치 과정에 대한 설명은 이 책의 범위를 넘어서며 MongoDB 설치 방법을 설명하는 좋은 정보들이 이미 많이 존재한다. 대표적으로 MongoDB 문서를 추천하는데, "http://mongodb.org"에서 이 문서의 최신판을 항상 받을 수 있다.

### 2.3.1 MongoDB 데이터베이스 서버 연결하기

PHP에서 MongoDB에 연결하는 방법은 다른 데이터베이스에 연결하는 방법과 매우 흡사하다. 기본 호스트는 localhost고, 기본 포트는 27017이다. 기본 설정을 사용한다면, 둘 다(또는 하나만) 연결 함수에서 생략할 수 있다.

- localhost에 있는 MongoDB 데이터베이스 서버에 27017번 포트를 이용하여 연결하기

```
$connection = new Mongo();
```

- 원격 호스트에 기본 값과 다른 커스텀 포트를 이용하여 연결하기

```
$connection = new Mongo( "172.20.10.8:65018" );
```

### 2.3.2 데이터베이스 선택하기

생성된 데이터베이스 서버와 연결을 이용해 데이터베이스에 접근할 수 있다. 접근 방법은 다음과 같다.

```
$db = $connection->selectDB('dbname');
```

흔히 그렇듯, 위와 같은 일을 하는 다른 방법이 있다. MongoDB는 정의된 사용법 외에도 축약형을 지원한다. 데이터베이스를 선택하는 방법에도 축약형이 있다.

```
$db = $connection->dbname;
```

> **NOTE_** 사용자가 생성되지 않은 데이터베이스를 선택하려고 하면, MongoDB는 에러 대신 입력된 이름으로 새로운 데이터베이스를 생성한다. 때문에 사용하는 이름을 제대로 확인하는 일은 대단히 중요하다. 데이터베이스에 접속했는데 원하는 데이터가 보이지 않을 때 가장 먼저 해야 할 일은, 실수로 이름을 잘못 입력해서 무심코 새로운 (텅 빈) 데이터베이스를 만들었는지 확인해보는 것이다.

## 2.4 기본 사항(CRUD 연산)

데이터베이스로 주로 하는 일은 데이터를 생성하고, 변경하고, 검색하는 것이다. 이 섹션을 통해 CRUD라 알려진 생성하기Create, 읽기Read, 갱신하기Update, 삭제하기Delete 기본 연산과 데이터를 검색하고 받아오는 방법에 대해서 중점적으로 살펴보자.

### 2.4.1 컬렉션 생성/선택하기

이제 데이터베이스의 생성과 연결까지 되었으니, 다음으로 넘어가보자. 첫 번째로 할 일은 컬렉션을 생성하는 것이다. 컬렉션의 선택(그리고 생성)은 데이터베이스를 생성하고 접근하는 방법과 매우 비슷하다. 이전 섹션에서 생성한 데이터베이스 핸들을 사용할 것이다.

```
$collection = $db->addresses;
```

데이터베이스 연결과 컬렉션 선택을 한 번에 할 수도 있다.

```
$addresses = $connection->dbname->addresses;
```

지금까지는 관계형 데이터베이스에 연결할 때와 거의 똑같지만, 무엇을 하지 않았는가에 주의를 기울이면 다른 점을 파악할 수 있다. 우선 "CREATE DATABASE" 명령을 전혀 사용하지 않았다. 테이블이나 컬렉션도 생성하지 않았으며, 어떤 스키마도 정의하지 않았다. 지금까지 한 일이라곤 MongoDB 드라이버가 제공하는 PHP 인터페이스를 사용하여 데이터베이스에 접근한 것뿐이고 나머지는 모두 MongoDB가 대신 처리했다.

### 2.4.2 문서 생성하기

MongoDB에서 문서를 생성하기는 정말 쉽다. 배열을 생성하고 생성된 배열을 컬렉션 객체의 insert 메서드에 넘기기만 하면 된다.

```
$address = array(
  'first_name' => 'Peter',
  'last_name' => 'Parker',
  'address' => '175 Fifth Ave',
  'city' => 'New York',
  'state' => 'NY',
  'zip' => '10010'
);
$addresses->insert($address);
```

또 save 메서드를 사용할 수도 있다. save 메서드는 insert 메서드와 거의 똑같이 작동한다. 단지 _id 값이 입력될 경우, 배열을 삽입하는 것이 아니라 갱신한다는 점만 다르다. 실제로는 대부분의 상황에서 save를 쓸 때 훨씬 더 재사용 가능한 코드를 작성할 수 있기 때문에 거의 항상 save를 사용한다.

### 갱신 과정의 중요 상세 사항

MongoDB의 일반적 연산은 비동기적이다. 이 말은 사용자가 레코드를 넣었을 때 값을 반환하지 않는다는 뜻이다. 이런 방식을 종종 '쏘고 잊어버리는' 연산이라고 부른다. 이 방식을 사용하면 보통 더 복잡한 연산인 데이터 쓰기를 수행할 때 여러 가지 장점을 얻을 수 있다. 요청받은 작업을 끝내고 값을 반환할 때까지 PHP 스크립트 실행을 멈추게 하는 데이터베이스와 비교해, MongoDB는 같은 연산 중에 PHP 스크립트 실행을 막지 않아 훨씬 빠르게 처리할 수 있다. 좀 더 명확히 말하면 이것 자체가 더 나은 데이터베이스 성능을 보장하는 것은 아니지만, 처리량이 많을 때 더 나은 애플리케이션 성능을 보여준다.

MongoDB는 동기적 삽입 연산 역시 가능하다. 물론 동기적 삽입 연산은 연산이 끝날 때까지 PHP 스크립트 실행을 중단시킨다. 이는 MySQL이나 PostgreSQL[03] 같은 다른 데이터베이스가 쓰는 방식이다. 이 과정에서 애플리케이션은 데이터베이스를 기다려야 한다. 처리량이 많을 경우, (관계형 데이터베이스에서처럼) 처리가 끝나길 기다리는 데이터베이스 연결들이 계속 쌓이며 갖가지 문제를 낳는다. 때문에 특별히 필요한 경우가 아니라면 기본 방식을 사용하는 것이 좋다.

update, insert, remove, save 메서드는 모두 추가 매개변수로 옵션 배열을 넣을 수 있다. 동기적으로 연산을 작동하려면 safe 옵션에 true 값을 넣은 배열을 넘기면 된다.

```
$addresses->insert($address, array('safe' => true));
```

insert 메서드는 자체적으로 새로 생성될 _id 값을 매개변수로 넘겨진 배열(또는 객체)에 추가한다. 기존의 익숙한 방식과는 다르겠지만 이 작용은 중요하니 잘 이해해야 한다. 배열에 값이 추가되는 것은 데이터베이스에 데이터를 보내기 전에

03 http://www.postgresql.org/

이루어진다. insert 메서드는 기본 키primary key를 반환하지 않고, 넘어온 배열 또는 객체에 직접 넣는다. 기본 키 값을 얻기 위해서는 단순히 참조하면 된다.

```
$pk = $address['_id'];
```

safe 값이 매개변수로 넘겨지면 프로그램은 데이터베이스가 답할 때까지 기다린다. update가 실패하면 그 줄에서 MongoCursorException이 발생한다. 또 safe 값은 정수 값도 가질 수 있는데, 그 값은 중복 시스템[04]에서 값을 반환받을 시스템의 개수를 나타낸다. 만약 그 숫자만큼 반환하지 못할 경우, 정해진 시간이 끝나면 MongoCursorTimeoutException이 발생한다.

때문에 이 기능을 사용할 때는 너무 큰 숫자를 사용하지 않는 것이 좋다. 예를 들어 safe 값이 3이고 세 개의 노드를 가진 클러스터에서 실행될 경우, 하나의 노드가 죽지 않는 한 이 연산은 잘 작동할 것이다. 그렇지만 곧 애플리케이션은 연산 하나하나에 매여서 긴 시간 동안 update를 처리하지 못하고 멈추게 될 것이다. timeout은 옵션 배열에 넣을 수 있는 또 하나의 매개변수로, timeout exception이 발생될 때까지 시간을 밀리 초 단위로 정의한다.

### 일관성에 대하여

사람들은 흔히 'safe'가 일관성을 의미한다고 오해한다. MongoDB는 결과적 일관성만을 보장하는 다중 마스터 시스템(dynamo[05])과는 달리 완전한 일관성을 보장하는 시스템이다. 이 말은 언제든 하나의 마스터에서 데이터를 읽으면, 항상 같은 데이터를 얻는다는 것을 의미한다. 반면 다중 마스터 시스템에서는 서로 다른 두 마스터에서 하나의 레코드를 받기 때문에, 레코드의 서로 다른 버전을 얻을 수도 있다. 고유 인덱스를 가지는 컬렉션에 데이터를 쓸 경우라면 동기적 방식을 사용하는

04 (역자주) 연결된 여러 시스템 클러스터를 말한다.

05 (역자주) dynamo는 AWS의 NoSQL서비스다. Key-Value 형태로 대용량의 데이터를 저장할 수 있으며, 고속의 데이터 액세스를 제공한다(http://aws.amazon.com/ko/dynamodb/).

것이 나을 것이다. 이 경우 애플리케이션은 값이 이미 존재해서 연산이 실패할 경우에 대한 처리를 포함하여, 쓰기 연산이 확실히 수행되도록 보장할 수 있다.

#### fsync에 대하여

또 다른 옵션으로 fsync가 있는데, 이 옵션은 디스크에 쓰도록 강제한다(또한 safe를 암시하기도 한다). MongoDB의 쓰기 성능을 최적화하는 방법 중 하나는, 지속적으로 쓰기 연산을 수행하는 게 아니라 쓰기 연산 풀을 두고 모인 연산을 한 번에 처리하는 것이다. 쓰기 연산이 입력된 시간과 그것이 실제 디스크에 쓰이는 시간 사이에 어떤 문제(커널 패닉, 하드웨어 고장 등)가 일어날 경우, MongoDB 1.8버전이 개발되기 이전에는 유일하게 fsync 옵션만이 입력된 내용의 유실을 막을 수 있었다. MongoDB는 1.8버전부터 데이터 유실을 방지하는 선행 기입 저널write-ahead journal을 지원하기 시작했다. 저널링을 사용하도록 설정하면, fsync를 사용할 필요도 없고 사용해서도 안 된다.

### 2.4.3 기본 키와 객체아이디(ObjectId)

MongoDB는 여느 데이터베이스와 마찬가지로 고유한 기본 키를 사용한다. 따로 설정하지 않는 한, MongoDB는 각 문서의 기본 키를 자동으로 생성한다. 생성된 기본 키를 MongoDB에서는 객체아이디라고 부른다. MongoDB의 객체아이디는 문자열이나 정수가 아니라 객체다. 곧 살펴보겠지만, 문서 질의 과정에서 이 사실은 매우 중요하다.

객체아이디는 타임스탬프timestamp와 처음 작성된 기기의 정보를 가지고 있다. 또한 객체로서 실행 가능한 메서드도 포함하고 있다. 가장 쓸모 있는 메서드는 아마도 가지고 있는 타임스탬프 값을 반환하는 getTimestamp 메서드일 것이다.

```
$id->getTimestamp();
```

**기본 키에 대하여**

문서에 고유 아이디가 없으면 MongoDB가 자동으로 생성하긴 하지만 사용자가 직접 입력할 수도 있다. 문서 배열의 _id 요소에 객체아이디나 정수, 문자열, 또는 다른 정보를 입력하면 된다. 이 기능은 변경되지 않는 키로 자주 참조되는 컬렉션을 사용할 때 특히 유용하다. 예를 들면, 사용자명은 (프로그램이나 사업 규칙에 따라) 다른 여러 객체에 의해 참조되고 사용되는 동시에 변경되지 않는 경우가 많다. 구조상 고유 식별자를 이미 지니고 있는 객체들은 객체아이디 대신 고유 식별자를 사용할 수 있는지 고려해보는 게 좋다. 그럴 경우 추가 공간 절약은 물론 새 인덱스 사용에 드는 간접비용을 줄일 수 있다.

다만, 배열은 기본 키로 사용할 수 없다는 점에 주의한다.

### 2.4.4 문서 읽기

MongoDB는 구조화 질의어(SQL)를 비롯해 어떤 종류의 질의어도 사용하지 않는다. 사용자는 원하는 정보를 배열로 반환받을 수 있다. MongoDB는 SQL의 유연성을 계승했지만 대부분의 경우에서 훨씬 더 간단하다.

키/값 저장소와 마찬가지로, 사용자는 기본 키를 통해 문서에 접근할 수 있다.

```
$id = new MongoId('4ba667b0a90578631c9caea1');
$pp = $addresses->findone( array( '_id' => $id ) );
```

또는 키/값 저장소와는 달리, 다른 키를 통해서도 문서에 접근할 수 있다.

```
$pp = $addresses->findone( array(
    'first_name' => 'Peter',
    'last_name' => 'Parker'
) );
```