# 그림으로 공부하는 오라클 구조

**絵で見てわかるOracleの仕組み**

**(E de Mite Wakaru Oracle no Shikumi : 1185-8)**



그림으로 공부하는 오라클 구조

**초판 1쇄 발행** 2015년 9월 10일

**지은이** 오다 케이지
**옮긴이** 이민재
**펴낸이** 장성두
**펴낸곳** 제이펍

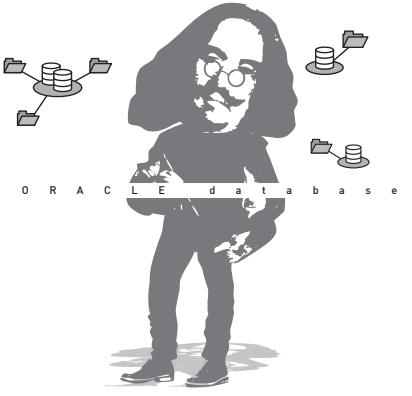
**출판신고** 2009년 11월 10일 제406-2009-000087호
**주소** 경기도 파주시 문발로 141 뮤즈빌딩 403호
**전화** 070 - 8201 - 9010 / 팩스 02 - 6280 - 0405
**홈페이지** www.jpub.kr / **이메일** jeipub@gmail.com

**편집부** 이민숙, 이 슬, 이주원 / **소통·기획팀** 민지환, 현지환
**표지디자인** 미디어픽스 / **본문디자인** 북아이
**용지** 에스에이치페이퍼 / **인쇄** 해외정판사 / **제본** 광우제책사

ISBN 979-11-85890-30-2 (93000)
**값** 24,000원

# 그림으로 공부하는 오라클 구조

오다 케이지 지음 / 이민재 옮김




Jpub 제이펍

**※ 드리는 말씀**

- 이 책에 기재된 내용을 기반으로 한 운용 결과에 대해 저자, 역자, 소프트웨어 개발자 및 제공자, 제이펍 출판사는 일체의 책임을 지지 않으므로 양해 바랍니다.
- 이 책에 등장하는 각 회사명, 제품명은 일반적으로 각 회사의 등록 상표 또는 상표입니다. 본문 중에는 ™, ©, ® 마크 등이 표시되어 있지 않습니다.
- 이 책에서 사용하고 있는 제품 버전은 독자의 학습 시점이나 환경에 따라 책의 내용과 다를 수 있습니다.
- 본문 중 일본 내의 실정에만 국한되어 있는 내용이나 그림은 일부를 삭제하거나 국내 실정에 맞도록 변경하였으니 참고 바랍니다.
- 책 내용과 관련된 문의사항은 옮긴이나 출판사로 연락해 주시기 바랍니다.
  - 옮긴이: xxeronis@hotmail.com
  - 출판사: jeipub@gmail.com

# 차례

# 옮긴이 머리말

저는 이 책의 목차를 처음 읽었을 때 '내가 처음 오라클을 접했을 때 이런 도서가 있었다면 얼마나 좋았을까?'라는 생각이 들었습니다. 우리나라에 출간된 오라클 관련 도서들을 보면 고급 및 특급 수준의 독자만을 대상으로 하는 도서가 많습니다. 독자 중 상당수는 필독서로 추천받은 책들을 읽으며 '지금 내가 이런 수준의 책을 읽는 게 맞는 것일까?'라는 의문을 지닌 적이 있을 겁니다. 저 역시 처음 오라클을 접했을 때 그런 고급 책을 읽으며 '내가 가진 기본 지식이 이 책을 읽기에는 부족하다는 것'을 느꼈고, 이는 마치 계단의 아랫부분 몇 개가 존재하지 않아 도저히 혼자서는 오르지 못해 목적지를 멍하니 바라보는 그런 모습이었습니다.

하지만 저는 다행히도 그럴 때마다 많은 분의 도움을 받아 조금씩 계단을 올라갔지만, 그렇게라도 도움을 받지 못하고 그 자리에서 혼자 헤맨 분들도 많을 것으로 생각합니다. 그래서 저는 막 입문한 사람과 조금의 경험이 쌓여 초급을 벗어나고자 하는 사람들을 위한 도서를 번역하고 싶었습니다. 그 첫 번째 도서로, 초급의 티를 벗기 위해 조금이라도 도움을 드리기 위한 《나만 알고 싶은 오라클 실무 테크닉》을 2014년 9월에 번역 출간하였습니다.

이번 《그림으로 공부하는 오라클 구조》는 오라클을 접한 지 얼마 안 된 독자들에게 제대로 된 기본 지식과 공부의 방향을 알려주기 위한 도서입니다. 저자인 오다 케이지 님은 어려운 용어 대신에 오라클을 '창고 회사'라는 실생활에 비유해 초급 독자에게 조금이라도 더 쉽게 와 닿을 수 있게 집필하였습니다. 지금까지 오라클의 구조를 제대로 이해하지 못하고 모호한 상태로 머릿속에 지니고 계셨다면, 이 책을 읽음으로써 제대로 된 방향으로 나아가는 데 큰 도움이 될 것으로 생각합니다.

개인적으로는 《나만 알고 싶은 오라클 실무 테크닉》보다 이 책이 먼저 출간되었다면 더 좋았을 것이라는 생각도 듭니다. 독자들께서는 이 책을 먼저 읽고 《나만 알고

싶은 오라클 실무 테크닉》을 읽으신다면 더 효율적인 학습을 하실 수 있을 것입니다.

제게 언제나 조언을 아끼지 않으시는 위즈베이스 김주현 수석님과 출간하는 데 도움을 주신 제이펍 장성두 실장님, 그리고 항상 도움을 주시는 데이타헤븐 문병권 이사님께 감사의 말씀을 올립니다. 또한, 항상 바른길로 가도록 조언해 주시는 정찬영 선생님, 그리고 항상 믿고 지지해 주시는 사랑하는 부모님, 형, 형수님께도 감사의 인사를 드리고 싶습니다.

책과 관련된 번역의 오류, 제안뿐만 아니라 오라클과 관련된 궁금한 내용은 메일(xxeronis@hotmail.com)을 통해 언제든지 연락해 주시기 바랍니다.

2015년 8월

옮긴이 **이민재**

# 머리말

안녕하세요? 일본 오라클에서 컨설턴트로 재직하고 있는 오다(小田)라고 합니다. 저는 약 17년 전에 일본 오라클 사내 교육부에 입사하였고, 그로부터 10년 이상 일본 오라클 사내에서 오라클과 유닉스, 네트워크와 같은 기술을 교육해 왔습니다. 그 후 데이터베이스 컨설턴트가 되었고, 미션 크리티컬 시스템을 지금까지 담당하고 있습니다. 그 덕분에 OS, 스토리지, 네트워크와 같이 오라클 이외의 기술도 잘 다룰 수 있다는 점이 (낯간지러운 말이지만) 저의 장점입니다.

제가 사내 교육부에 재직하고 있던 당시는 수백 명 단위의 엔지니어들을 교육했습니다. 또한, 가르친 엔지니어들이 어떻게 성장해 왔는지도 지켜보아 왔으며, DB 컨설턴트가 되어서부터는 많은 고객과 SI 종사자들을 봐왔습니다. 필자의 경험상 일반적으로 '실력이 늘지 않았던 엔지니어'는 통째로 암기하던 습관을 지닌 엔지니어들이었습니다.

지금 당장은 아무렇지 않더라도 아키텍처나 동작을 이해하려 하지 않고 '이렇게 명령어를 치면 되네!'라든가, '잘은 모르겠지만 이런 거구나!'라는 식으로 암기하는 엔지니어는 앞으로도 실력이 늘지 않을 것입니다. 반대로, 오라클을 일반적인 IT의 관점에서 이해하고 있는 엔지니어와 아키텍처를 이해하고 있는 엔지니어는 응용도 자유롭게 하며 환경의 변화나 새로운 기술에 적응하는 것도 빠른 경향을 보였습니다. 오라클 내부를 살펴보면, 결국 오라클은 OS상에서 작동하는 디스크나 네트워크를 사용하는 애플리케이션에 지나지 않습니다. 이것들을 토대로 독자가 제대로 된 실력을 갖춘 오라클 엔지니어로 성장하기 위한 첫걸음을 내딛게 도와주는 것이 바로 이 책의 목적입니다. 여기에 스스로 조사하며 공부해 나가는 습관을 붙일 수 있다면 더욱 좋습니다.

이 책의 구체적인 목표는 '오라클 용어를 사용하지 않고 오라클의 근본적인 아키텍

처를 이해하자.'입니다. 물론, 일류 관리자가 되기 위해서는 아키텍처만으로는 충분하지 않습니다. 명령어 같은 부분은 별도의 서적이나 블로그 등을 통해서 익혀야 할 겁니다. 또한, 스스로 데이터베이스를 생성하고 파괴하는 것을 반복하며 여러 상황을 직접 경험하는 것도 매우 중요합니다. 실제로 데이터베이스를 구축하는 것을 집이나 현장에서 경험해 보면 많은 도움이 된다고 생각합니다. 단, 이런 부분들을 공부할 때에도 이 책에서 설명하는 아키텍처나 동작을 머릿속에 떠올리면서 진행하시면 몸에 익는 것이 분명히 다를 것입니다.

이 책이 대상으로 하는 독자는 다음과 같습니다.

- 오라클을 처음 배우려는 사람
- 오라클을 배우긴 했지만 아키텍처에는 자신이 없는 사람
- 오라클을 배우다가 좌절한 사람
- 실력이 늘지 않아 고민하는 사람

이 책은 독자가 '기본적인 SQL문(SELECT/INSERT/UPDATE/DELETE/COMMIT/ROLLBACK)'을 알고 있다는 것을 전제로 이야기합니다. 좀 더 자세히 말씀드리자면, SELECT문에서 테이블의 조인(join)과 데이터가 변경되면 COMMIT으로 확정하고, ROLLBACK으로 취소한다는 것을 이해하고 있으면 충분합니다. 위에서 설명한 내용을 보면 알 수 있듯이, 이 책은 다음과 같이 세 가지 특징을 가지고 있습니다. 또한, 가끔 나오는 명령어는 아키텍처와 동작을 이해하기 위해서 간략히 추가한 내용입니다.

- 머릿속에 그림을 그릴 수 있을 때까지 계속해서 아키텍처와 동작에 관한 내용만을 설명한다
- 가능한 한 오라클 이외의 용어(IT 기본 용어)를 사용해서 설명한다
- 명령어는 거의 설명하지 않는다

오라클 아키텍처의 세계를 그림으로 보고 즐기면서 체험하고 실력을 향상하는 데 도움이 되었으면 합니다.

지은이 **오다 케이지**

# 저자 소개

**오다 케이지(小田 圭二)**

일본 오라클 주식회사의 테크놀로지 컨설팅 본부 수석 컨설턴트다. 오라클 직원을 대상으로 데이터베이스, OS, 네트워크 분야의 기술 연수를 수행했으며, 이후 데이터베이스 컨설턴트가 되어서는 각종 미션 크리티컬 시스템을 담당하였다. '진정한 엔지니어를 키울 수 있는 사람'이 인생 목표이며, 그러기 위해 우선 자신부터 '잘하는 엔지니어'가 되기 위해 노력하고 있다. '오라클도 OS에서 움직이는 애플리케이션에 지나지 않는다'라는 자신만의 신조를 갖고 있는 저자는 《나만 알고 싶은 오라클 실무 테크닉》(제이펍)을 포함하여 10여 종의 책을 집필하거나 감수를 맡았다.

# 베타리더 후기

**강미희(휴인시스텍)**

그림과 함께 설명되어 있어 기초를 쉽고 확실하게 배울 수 있었던 책이었습니다. 팁과 칼럼을 통해 조금 더 깊은 내용을 알 수도 있었고, 각 장의 요약을 통해 한 장의 내용을 머릿속에 다시 상기할 수 있어서 좋았습니다. 그리고 마지막 장에서의 Q&A를 통해서는 어떻게 응용해야 할 것인가에 대해서도 어렴풋하게나마 정리할 수 있어서 많은 도움이 되었습니다.

**고승광(플랜티넷)**

과거 오라클을 책으로 처음 접했을 때 오라클 구조가 OS처럼 생각되어서 바로 포기했던 기억이 납니다. 그런데 이 책은 오라클 구조를 일상(회사/창고)에 비교해서 아주 알기 쉽게 설명하고 있습니다. 이 책을 읽고 나서 《나만 알고 싶은 오라클 실무 테크닉》을 읽어보면 학습 효과가 더 클 듯합니다.

**김주현(위즈베이스)**

오라클의 내부 작동 원리도 이해하고 나면 전혀 어렵지 않습니다. 사람들이 오라클 아키텍처를 어려워하는 이유는 무턱대고 암기하려고 들기 때문입니다. 이 책은 '왜?'라는 질문을 바탕으로 구성되어 있어서 끝까지 읽고 나면 머릿속에 SQL문이 오라클 내부에서 처리되는 모습이 자연스럽게 그려질 겁니다. 또한, SQL의 처리가 늦어지는 원인과 해결책을 찾는 데 필요한 기본 지식도 얻을 수 있습니다.

**손정호(한의사)**

《그림으로 공부하는 오라클 구조》라는 제목에 어울리게 그림으로 된 설명이 적재적소에서 독자의 이해를 돕고 있습니다. 덕분에 평소 데이터베이스나 오라클에 대한 지식이 부족했는데도 막힘없이 재미있게 볼 수 있었습니다. 오라클의 내부 구조가 궁금하지만 너무 두껍고 어려운 책을 잡기는 엄두가 안 나시는 분들께 부담 없이 일독을 권하고 싶습니다.

**이상현(SI개발자)**

오라클 책을 접하니 사회 초년 시절에 프로그래밍과 함께 SQL의 매력에 빠져들었던 기억이 잠시 들었습니다. 이 책을 보기 전에는 '그림으로 오라클을 설명해 봐야 얼마나 깊이가 있겠어?'라고 생각했는데, 읽으면서 점점 창피해지더군요. 데이터를 뽑는 데에만 신경을 쓰느라 미처 어떻게 구성되고 관리되는지를 전혀 생각하지 못했기 때문일 겁니다. 그간 모르고 있던 부분을 선배, 친구가 쉽게 하나하나 설명을 해 준다는 생각이 들 정도로 쉽게 다가왔습니다.

**이재빈(연세대학교 정보대학원)**

글보다는 그림, 영상에 의해 정보를 얻는 데 능숙한 세대에게 적합한 콘텐츠로, 정확한 지식을 간결하게 전달하는 데 많은 노력을 기울인 책이라고 생각합니다. 오라클의 구조를 모르는 사람이라도 이 책 한 권으로 전반적인 그림을 그릴 수 있을 것이라 확신이 들었습니다. 정말 쉽게 오라클의 구조를 이해할 수 있어서 좋았습니다.


제이펍은 책에 대한 애정과 기술에 대한 열정이 뜨거운 베타리더들로 하여금
출간되는 모든 서적에 사전 검증을 시행하고 있습니다.

CHAPTER

1

# I/O와 디스크의 관계

이 책은 기본적인 오라클 아키텍처를 그림과 함께 알기 쉽게 설명하였습니다. 1장에서는 오라클의 기본 기능 중 '오라클을 이해하기 위한 필수 키워드'를 선정하고 이것을 토대로 I/O와 디스크의 관계를 자세하게 설명할 것입니다. 일반적으로 디스크와 I/O의 동작을 특별히 의식한 적은 없을 수도 있지만, 동작을 그림으로 떠올릴 수 있다면 오라클에서 발생하는 여러 현상을 이해하기가 수월해집니다. 자, 그러면 응용력이 있는 오라클 엔지니어가 되기 위한 첫걸음을 내디뎌 보겠습니다.

## 오라클을 이해하기 위한 필수 키워드

우선은 오라클을 이해하기 위한 세 가지 키워드를 소개하겠습니다. 이 키워드는 이번 장뿐만 아니라 이 책의 여러 장에서 나타납니다.

**키워드**

① 병렬 처리를 가능케 하고 높은 처리량을 실현한다
② 응답(response)을 중시한다
③ 커밋한 데이터는 지킨다

DBMS(DataBase Management System, 데이터베이스 관리 시스템)의 내부 구조의 복잡함은 오라클에 국한되지 않으며, 다른 DBMS들도 마찬가지입니다. DBMS가 복잡해지는 것은 위의 세 가지 특성으로 인해 발생합니다. 더욱이 이 세 가지 특성은 서로 사이가 좋지 않아 모두 만족하는 것은 매우 어렵습니다. 예를 들어, '커밋한 데이터는 지킨다'를 위해 커밋하자마자 데이터를 디스크에 기록해 버리고 싶지만, 그렇게 하면 응답 시간이 나빠지게 됩니다. '병렬 처리를 가능케 하고 높은 처리량을 실현'하는 것도 간단하지 않습니다. 병렬 처리할 때는 처리가 모순되지 않도록 락(lock, IT 용어의 락)이 필요하며, 그로 인해 성능이 나오지 않는 경우도 있습니다.

또한, 이 책에서는 장마다 주제를 정해서 오라클의 'DBMS로서의 기본 기능'을 설명합니다. 최근의 오라클은 기능이 풍부해졌지만, 이 기능들은 모두 기본 기능들을 토대로 하고 있습니다. 또한, 이 책에 나오는 수치는 독자의 이해를 돕기 위해 사용한 참고 수치이며 단순한 기준 값 정도로만 생각하길 바랍니다.

# 오라클과 디스크(하드디스크)

오라클은 DBMS입니다. 또한, 오라클의 시점에서 보자면 데이터베이스는 오라클이 관리하는 디스크에 들어 있는 데이터를 의미합니다. 오라클은 디스크에서 데이터를 읽어 오고 처리를 한 후 데이터를 다시 디스크에 집어넣습니다. 즉, 오라클이 다루는 데이터는 디스크에서 꺼내 오고 디스크로 돌아가는 것입니다. 그래서 오라클과 디스크는 떼려야 뗄 수 없는 관계라 할 수 있습니다.

1장에서는 디스크를 설명합니다. 만약 고장 나도 상관없는 디스크를 가지고 계신다면 드라이버로 분해한 후 내부 구조를 살펴보면서 읽어 주세요. 단, 한번 디스크를 분해하면 다시는 사용할 수 없으므로 주의하길 바랍니다.

그럼, 가장 처음에 동작하는 모습을 알아보도록 하겠습니다. 디스크를 LP판(음악을 재생하기 위한 레코드판이며, CD여도 상관없습니다)이라고 생각하고 이미지를 떠올려 주세요(그림 1).

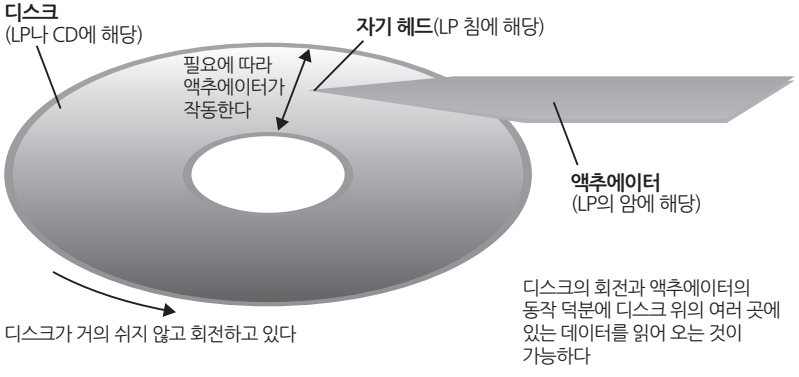



**그림 1** 디스크의 이미지

디스크는 계속 회전하고 있으며 그 위를 헤드가 움직여서 음악이 아닌 데이터를 읽거나 기록합니다. LP판과 다른 것은 원반이 여러 개 겹쳐 있다는 것과 양면을 사용한다는 것이지만, 여기서는 중요하지 않습니다. 또 다른 점으로는 속도입니다. 음

악에 비유하자면, 트랙의 첫 부분을 찾는 작업을 1초에 100번 할 수 있을 정도로 헤드가 빠르게 움직입니다. 또한, 회전 속도도 엄청나서 1분에 1만 번 정도 회전합니다(그림 2).

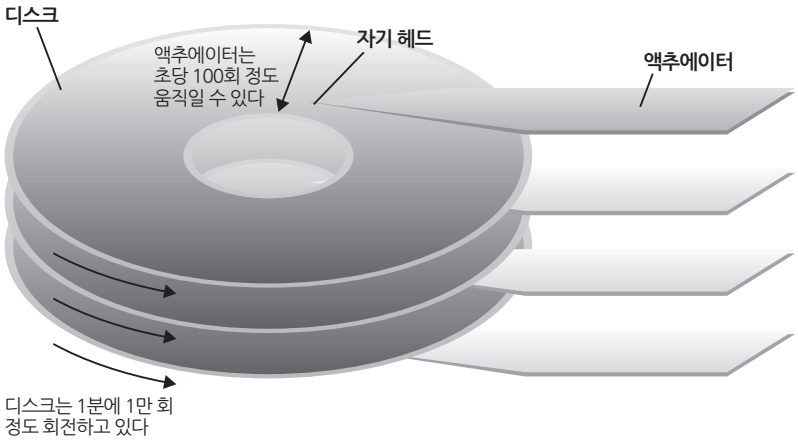



**그림 2** 좀 더 현실에 가까운 디스크의 이미지

## 디스크의 동작

무서울 정도로 빠르게 움직이는 LP의 이미지를 떠올릴 수 있게 되었으니 다음으로는 I/O 처리에 필요한 디스크의 동작을 살펴보도록 하겠습니다. 데이터를 읽기(또는 기록하기) 위해서는 원하는 트랙을 찾지 않으면 안 됩니다. 이것을 디스크 용어로는 '시크(seek)'라고 부릅니다.

그 후에 원하는 정보를 읽어 낼 수 있는 위치가 회전해서 다가올 때까지 기다리고 있습니다. 이런 기다리는 시간을 '회전을 기다리는 시간'이라고 부릅니다. 그리고서야 데이터를 읽고 쓰게 됩니다(그림 3).

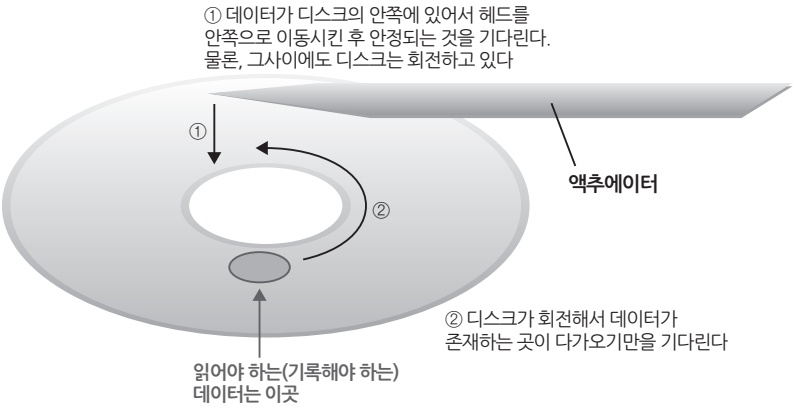



**그림 3** I/O 처리에 필요한 동작

여기서는 시크에서부터 회전을 기다리는 데 걸리는 시간을 합쳐 10msec(밀리초)라고 가정하겠습니다. 또한, 디스크의 전송 속도(헤드를 움직이지 않고 계속 데이터를 읽고 쓸 수 있을 때의 처리량)는 20MB/초라고 가정합니다. 일반적으로 생각하면 디스크가 매우 고속으로 움직이는 것처럼 보이지만, 컴퓨터의 속도와 비교해 보면 매우 느린 속도입니다. 메모리에 접근(액세스)하는 속도는 nsec(나노초, 10억 분의 1초)이지만, 디스크에 접근하는 것은 msec(밀리초, 1천 분의 1초) 정도의 시간이 걸립니다.

이유를 간단히 설명하자면, 메모리는 전기 신호로 처리하지만 디스크에는 기계 동작이 들어 있기 때문입니다. 따라서 디스크의 I/O는 DBMS에게 필요한 것이긴 하지만 가능한 한 줄여야 하는 부분입니다.

## ◆ 어떻게 I/O의 대기 시간을 줄일까?

그러면 실제 어떤 방식으로 I/O의 대기 시간을 줄이면 좋을까요? 그 답을 알아보기 위해 먼저 시퀀셜 액세스에 관해 설명하겠습니다.

시퀀셜(sequential)은 '순서를 따라서'라는 의미의 '순차'를 뜻하며, 시작점에서부터 마지막까지 중간 부분을 빠트리지 않고 전부 액세스(읽기/쓰기)하는 것입니다. 풀 스캔(테이블의 모든 데이터를 읽어 오는 것을 의미)할 때 메모리에 데이터가 없다면 시퀀셜

액세스가 발생합니다(그림 4). 이 두 가지 용어는 데이터베이스를 이해하기 위해서 매우 중요하므로 꼭 기억해 주세요.

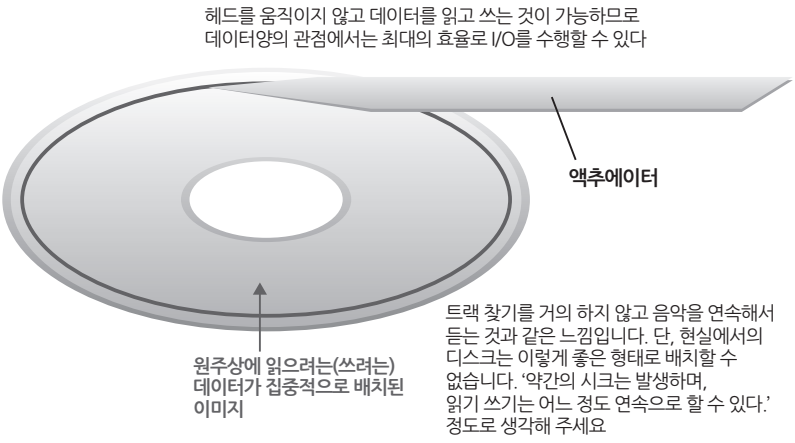



**그림 4 시퀀셜 액세스**

그러면 테이블의 크기를 1GB라고 가정하고 시퀀셜 액세스로 모든 데이터를 가져온다고 하면, 50초(크기/전송 속도 = 1,000MB/20MB/초)가 걸립니다. 테이블 크기가 100MB라고 하더라도 5초가 걸립니다. 이렇게 시간이 오래 걸리면 대부분 SQL의 성능이 만족스럽지 못합니다.

그래서 인덱스라는 발상이 나오게 되었습니다. 책에서 무엇인가를 조사하고 싶을 때 여러분은 어떻게 하는지요? 맨 처음부터 모든 페이지를 읽어서 찾는 분은 거의 없을 테고, 대부분은 색인(인덱스)을 이용할 것입니다. 이미 알고 있듯이 색인에는 키워드가 순서대로 나열되어 있으며, 해당 페이지 번호도 같이 기재되어 있습니다. 그 페이지 번호를 이용해서 원하는 페이지를 빠르게 찾은 후 필요한 정보를 얻을 수 있습니다.

데이터베이스의 인덱스도 마찬가지입니다. 데이터베이스의 인덱스에는 색인할 때 사용하는 키 값(SQL문의 where절에 적는 조건의 값을 말함)과 그 키가 존재하고 있는 위치가 기록되어 있습니다(그림 5).

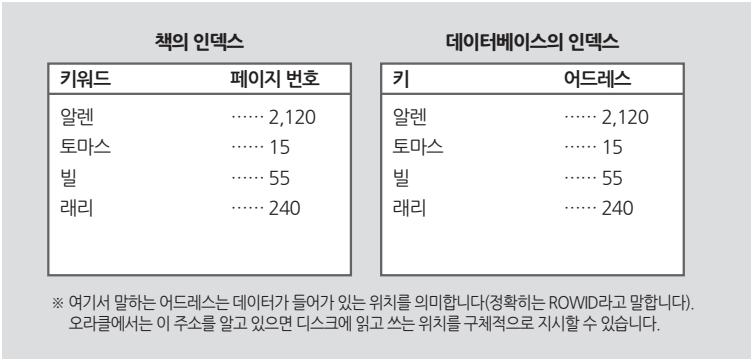



**그림 5** 인덱스의 이미지

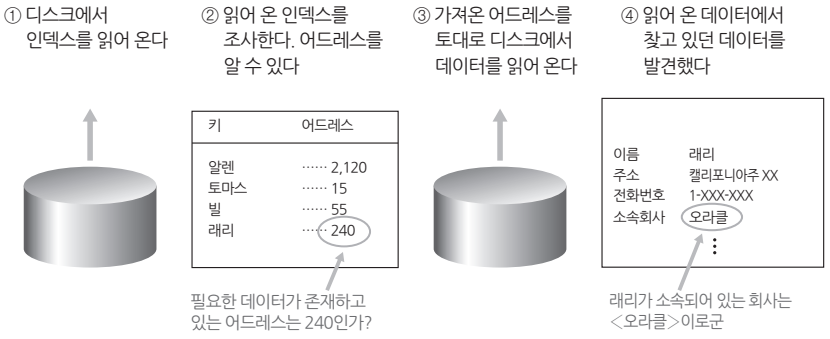
## ◆ 인덱스의 사용 예

예를 들어, 래리에 대한 내용을 책에서 조사하고 싶을 때는 인덱스에서 해당 내용이 240페이지에 쓰여 있다는 것을 확인한 후 해당 페이지를 펼쳐서 래리의 정보를 얻게 됩니다. SQL문도 마찬가지로, WHERE절에 '래리'의 정보를 찾고 싶다는 것을 기술하고 SELECT 뒤에 알고 싶은 항목(예를 들면 '소속회사')을 적습니다.

```
SELECT "소속회사" FROM "개인 데이터" WHERE "이름" = "래리";
```

이 SQL문을 인덱스를 사용해서 처리할 때는 우선 '이름'이 실려 있는 인덱스를 조사합니다. 그 결과로 어드레스(ROWID)를 얻을 수 있고, 그 어드레스를 토대로 데이터를 읽어 옵니다. 읽어 온 데이터의 안에는 래리의 데이터가 전부 모여 있으므로 그 안에서 '소속회사'의 데이터를 사용자에게 반환합니다(그림 6).

인덱스는 이렇게 매우 편리한 기능입니다. 그렇다면 인덱스 자체의 크기가 커지면 크기가 큰 테이블을 조회하는 것과 마찬가지로 처리하는 데 필요한 시간이 늘어나게 될까요? 답을 먼저 말씀드리자면, 그런 일은 발생하지 않습니다. 왜냐하면 오라클은 인덱스를 '인덱스의 인덱스'를 붙이는 것과 같은 형태를 여러 단계로 구성하기 때문입니다(그림 7). 이런 점이 책에서 사용하는 색인과의 차이점입니다.




주: 앞으로 이어질 장에서 설명하겠지만, 오라클은 메모리에 데이터의 캐시를 가지고 있습니다. 따라서 운이 좋다면 실제로 디스크에 가서 읽지 않고도 끝날 수 있습니다. 또한, 데이터를 읽어올 때는 '블록'이라고 불리는 단위로 I/O를 수행합니다. '블록'에 관해서도 앞으로 나올 장에서 설명합니다. 지금은 일정한 크기의 데이터 모음이라고 생각해 주세요

**그림 6** 인덱스를 이용한 SQL 처리의 흐름

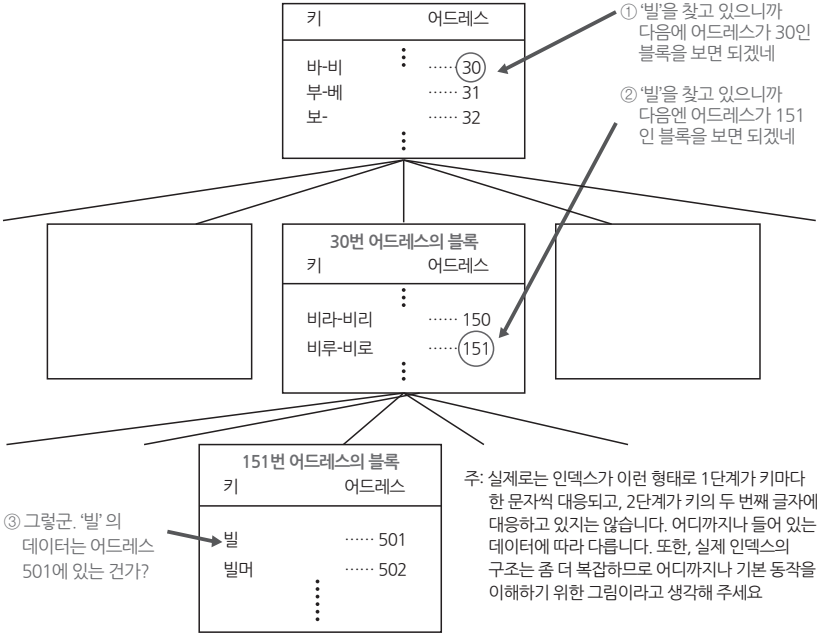



**그림 7** 여러 단계를 가진 인덱스

이렇게 여러 단계로 구성된 구조를 트리(tree) 구조라 부릅니다(나무를 거꾸로 한 모양처럼 보일 것입니다). 장점은 인덱스의 필요 없는 부분은 읽지 않고 끝난다는 것입니다.

'블록'에 관해서도 앞으로 나올 장에서 설명할 예정입니다. 지금은 일정한 크기의 데이터 모음이라고 생각해 주세요.

## ◆ 랜덤 액세스

다시 I/O의 이야기로 돌아와서, 인덱스를 사용했을 때는 필요한 부분만 읽어 오면 충분하지만, 필요한 부분이 디스크 위에 연속적으로 있는 경우는 거의 없습니다. 따라서 헤드를 움직여 가면서 띄엄띄엄 접근하게 됩니다. 이렇게 접근하는 방식을 '랜덤 액세스(random access)'라고 하며, 시퀀셜 액세스와 반대의 의미를 지닙니다.

랜덤 액세스를 디스크의 시점에서 생각해 보면 비효율적임을 알 수 있습니다. 오라클의 블록 크기를 4KB라고 가정하면, 시크와 회전을 기다리는 것에 시간을 소비하기 때문에 1초 동안 약 400KB(100회의 시크 × 1회 I/O의 크기 4KB)밖에 읽어 오지 못합니다. 이것을 음악 청취에 비유한다면, 트랙을 찾아가는 것을 빈번하게 반복해서 한 시간 중 1분 정도밖에 음악을 듣지 못한 것과 마찬가지입니다. 남은 59분은 트랙을 찾아다닌 시간입니다(시크를 하지 않으면 초당 20MB를 읽을 수 있을 텐데, 초당 400KB밖에 읽을 수 없었기 때문).

실제로 데이터 전송의 효율에서 바라보면 DBMS의 I/O도 같을 수밖에 없습니다. 시크를 반복하기 때문에 DBMS(특히 OLTP[주1] 시스템)에서 사용하는 디스크의 지표는 IOPS(초당 수행 가능한 I/O 횟수)가 중요하다고 볼 수 있습니다. 그리고 대부분의 디스크는 IOPS가 100회에서 200회 정도입니다. 따라서 한 개나 두 개의 디스크를 사용하여 데이터베이스를 만들어 버리면 부하가 집중적으로 발생했을 때 시크가 요청을 따라잡을 수 없으므로 디스크에 병목 현상이 발생하게 됩니다(그림 8).

---

주1 OLTP(OnLine Transaction Processing). 철도의 좌석 예약 시스템이나 은행의 입출금 처리라고 생각하면 됩니다. 많은 단말에서 온라인(네트워크 통해서)으로 크지 않은 데이터를 읽거나 쓰는 작업을 하고 결과를 즉시 회신해야 하는 시스템 형태를 말합니다. 데이터가 크지 않다는 것과 '즉시'라는 부분에서 알 수 있듯이, 인덱스를 사용해야 하는 시스템 형태라고 할 수 있습니다.

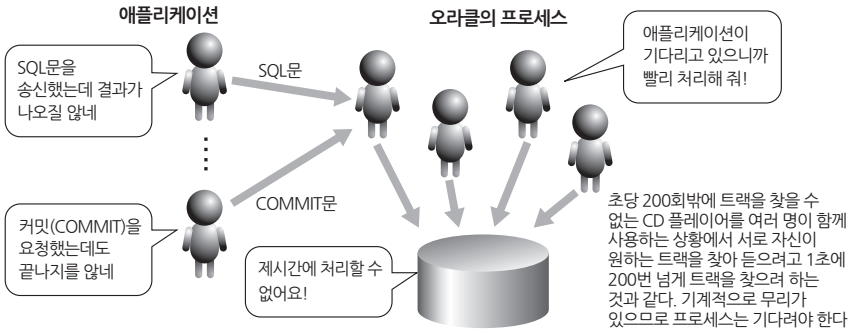



**그림 8** 디스크의 수가 적으면 디스크 자체가 병목이 된다

**TIP** 상급자 대상

접근하려는 데이터가 전체 데이터의 15% 미만일 경우에만 인덱스 액세스가 유리하다고 하는 이유는 무엇일까요? 이유는 시퀀셜 액세스와 랜덤 액세스의 특성 때문입니다. 테이블의 데이터가 대량이고 그중 한 개의 행을 꺼내야 한다면 당연히 인덱스 액세스가 빠르게 찾아낼 수 있습니다. 그에 비해 모든 데이터를 보려고 할 때 매번 인덱스를 찾은 후에 데이터를 찾아가면 오히려 속도가 느려지게 됩니다(책 전체를 읽을 때 인덱스를 일일이 찾아가면서 읽는 독자는 없을 겁니다).

그러면 데이터가 50%라면 어떨까요? 데이터가 25%라면 어떨까요? 여기에서 '디스크에서의 랜덤 액세스는 데이터를 읽어 오는 효율성이 시퀀셜 액세스보다 떨어진다'라는 특성이 중요합니다. 예를 들어, 테이블에 2만 건인 데이터가 저장되어 있다고 하고 그중 절반인 1만 건을 꺼낸다고 가정하겠습니다. 여기서 한 행은 4KB라고 하겠습니다. 이때 지금까지 사용해 왔던 디스크의 성능 값을 사용해 계산해 보면, 랜덤 액세스로 약 100초가 걸립니다(인덱스는 빈번하게 사용되고 있으므로 캐시(cache)에 보관되어 있다고 가정합니다). 그것과 비교하면 2만 건 전부를 읽어 오는 시퀀셜 액세스로는 모든 것을 디스크에서 읽어 온다고 해도 약 4초면 끝납니다. 즉, 모든 데이터가 아니더라도 일정 크기 이상의 데이터를 읽는다면 디스크 특성상 시퀀셜 액세스를 사용하여 테이블을 풀 스캔하는 편이 빠르다는 것을 알 수 있습니다.

단, 실제로는 캐시에 데이터가 보관된 경우도 있으며, 한 개의 블록에 여러 행의 데이터가 보관되어 있어서 1회의 I/O로 많은 데이터를 읽을 수 있는 경우도 있습니다. 더욱이 인덱스의 데이터를 디스크에서 읽어 와야 하는 경우도 있으므로 '15%가 임계치'라고 단순하게 말할 수는 없습니다. 어디까지나 기준 정도로만 생각해 주세요.

# 데이터를 보증하기 위한 디스크

오라클의 프로세스가 비정상적으로 종료했다고 해도 데이터는 무사합니다. 이 점이 DBMS와 다른 프로그램과의 차이점 중 하나입니다. 예를 들어, 엑셀(Excel)에서는 저장한 시점 이후의 데이터는 없어집니다. 이미 알고 있듯이, 프로그램이 비정상적으로 종료되거나 전원 버튼을 눌러서 전원을 끊으면 데이터는 없어집니다.

그것과 비교하면, DBMS는 어떤 장애에도 견뎌야만 합니다. CPU나 메모리가 고장이 나거나 정전이 발생해도 데이터를 잃어버려서는 안 됩니다. 컴퓨터의 주기억장치 내의 정보는 전기이므로 정전이 발생하면 메모리에서 사라지게 됩니다. 이런 가혹한 조건 속에 키워드에서 소개한 '커밋한 데이터는 지킨다'라는 특성을 어떻게 구현하고 있는 것일까요? 답은 간단합니다. 데이터를 변경한 후에 커밋이라고 입력하면 오라클은 데이터를 디스크에 기록합니다(그림 9).

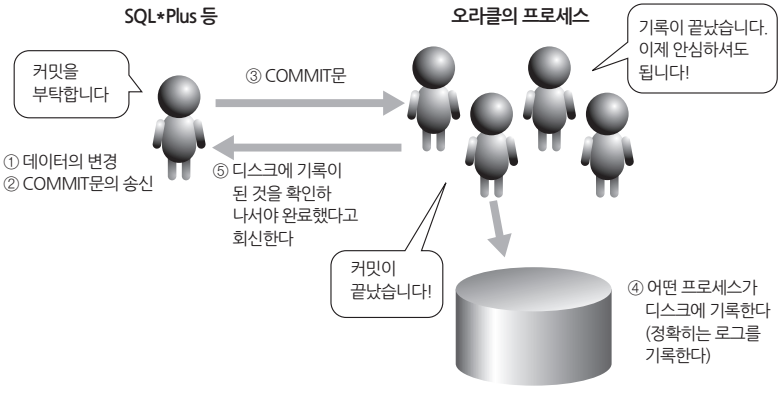



**그림 9** 커밋을 수신하면 데이터를 디스크에 기록한다

'그러면 느리지 않을까?'라고 생각할 수도 있지만, 거기에서도 속도를 빠르게 하기 위한 장치가 마련되어 있습니다. 오라클을 필두로 하는 DBMS들은 고속화를 위한 이런 장치가 있어서 그 구조가 복잡한 것입니다. 이 부분에 관해서는 뒤에 나올 장에서 설명하겠습니다.

칼럼

## '시퀀셜'은 어떤 의미인가?

오라클은 각종 정보를 대량으로 가지고 있습니다. 그중에는 얼마만큼의 I/O를 수행했는지도 기록되어 있지만, 헷갈리기 쉬운 부분들도 있습니다.

예를 들어, 랜덤 액세스는 'db file sequential read'라고 표시되며, 시퀀셜 액세스는 'db file scattered read'라고 표시됩니다. 'scattered'는 '분산됐다'라는 의미입니다. 의미에서 생각해 보면 반대로 되어 있는 게 아닌가라는 생각도 들지만, 표시가 잘못되어 있는 것은 아닙니다. 다음과 같은 의미가 있습니다.

'db file scattered read'는 시퀀셜하게 읽어 오는 것이므로 여러 개의 블록을 읽어 옵니다. 오라클은 데이터를 블록 단위로 메모리에 배치합니다. 즉, 여러 개의 블록이 연속하지 않은(분산된) 곳에 놓입니다. 따라서 'scattered'라고 표시합니다. 그에 반해 'db file sequential read'는 단일 블록을 읽어 오므로 읽어 온 데이터(1블록)는 당연히 메모리에서 연속됩니다. 오라클의 메모리에서 연속되어 있으므로 'sequential'이라고 표시됩니다.

저는 시퀀셜이라는 이름을 '시퀀셜 I/O'를 가리키도록 변경했으면 합니다만, 이미 정해졌기 때문에 이 이후로도 변경될 일은 없을 것입니다. 틀리기 쉬운 부분이므로 여러분도 잘 기억해 두시기 바랍니다.

※ **참고**: ≪Oracle 11gR2 Performance Tuning Guide≫의 10.3.3 db file scattered read와 10.3.4 db file sequential read≫

# 1장 요약

독자 여러분의 머릿속에 다음의 그림들이 떠오를 수 있기를 바랍니다.

- 디스크가 회전하고 있는 이미지
- 헤드가 움직이는 것과 회전해 오기를 기다리고 있어서 I/O에 시간이 걸리는 이미지
- 인덱스로 인해 해당 데이터에 빠르게 접근할 수 있다(테이블을 맨 앞에서부터 끝까지 풀 스캔하지 않아도 된다)는 이미지

이번 장에서 디스크가 느리다는 것, 그리고 인덱스로 인해 높은 효율로 데이터에 접근할 수 있다는 것을 알게 되었습니다. 이런 부분들이 앞으로 어떻게 전개되는지는 뒤에서 나올 장에서 설명할 것입니다. 한 개만 먼저 소개해 두자면, 버퍼 캐시(buffer cache)는 SQL문이 디스크의 성능에 영향을 가능한 한 받지 않게 하려고 존재합니다. 해당 내용은 3장에서 자세하게 설명하겠습니다.