데이터베이스 첫걸음 - 1주차

1장 데이터베이스란

1. 데이터베이스의 기본 기능

- **데이터의 검색과 갱신(등록/수정/제거)**
  - 데이터 포맷, 처리 성능에 유의
- **동시성 제어**
  - 갱신의 무결성을 어느정도로 보장하는가?
  - 복수 사용자의 갱신을 조절하기 위한 기능
  - dirty write는 무결성 관점에서 기피하는 경향이 있음
- **장애 대응**
  - 데이터 다중화, 백업
- **보안**

2. 데이터베이스의 종류

- **계층형 데이터베이스**
  - ex) 조직도, 전체 구조도
- **관계형 데이터베이스**
  - 현재 가장 주류
- **객체지향 데이터베이스와 XML 데이터베이스**
- **NoSQL 데이터베이스**

2장 관계형 데이터베이스란

▼ 추후

STEP 1 대표적인 DBMS의 특징을 알아보자

대표적인 DBMS인 Oracle, SQL Server, DB2, MySQL, PostgreSQL, Firebird의 공식 웹 사이트를 방문하여 각 제품의 특징을 조사해 봅시다.

- **Oracle** http://www.oracle.com/kr/database/overview/index.html
- **SQL Server** https://www.microsoft.com/ko-kr/server-cloud/products/sql-server/overview.aspx
- **DB2** http://www-01.ibm.com/software/kr/data/db2/
- **MySQL** http://www.mysql.com/
- **PostgreSQL** http://www.postgresql.org/
- **Firebird** http://www.firebirdsql.org/

STEP 2 대표적인 DBMS의 최신 버전을 조사해 보자

다음 DBMS의 웹 사이트에서 현재 제공하는 최신 버전을 확인해 봅시다.

- **Oracle**
- **SQL Server**
- **DB2**
- **MySQL**
- **PostgreSQL**
- **Firebird**

STEP 3 직장에서 사용하는 DBMS를 확인해 보자

직장에서 이미 DBMS를 사용하고 있다면 어떤 DBMS를 사용하는지를 담당자에게 확인해 봅시다.

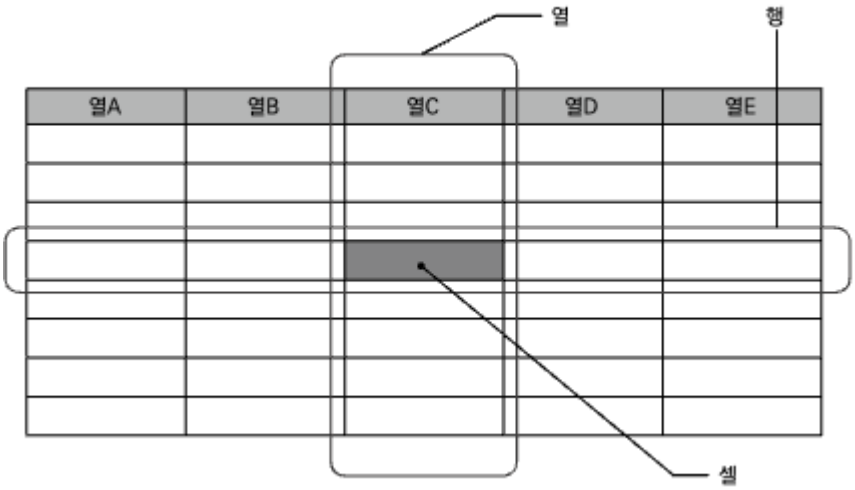
관계형 데이터베이스란 무엇인가

- 데이터를 2차원 표를 사용해 관리하는 데이터베이스
- SQL을 통해 프로그래머가 아니라도 데이터를 조작할 수 있음

SQL이란

- 모국어를 말하는 것처럼 데이터 조작
- SELECT(검색)/INSERT(등록)/UPDATE(갱신)/DELETE(제거)
- **테이블**
  - 데이터를 관리하기 위한 유일한 단위
- **열과 행**

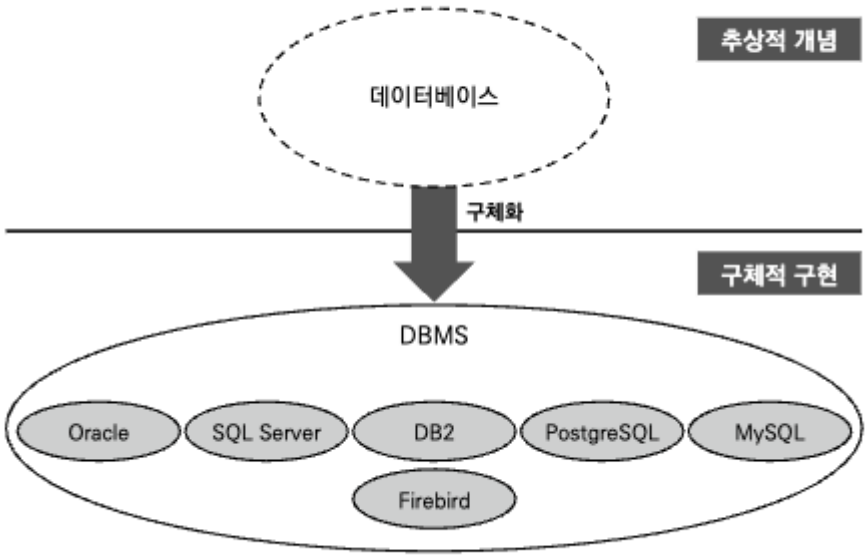
그림 2-1 테이블의 개념



관계형 데이터베이스를 다루기 위한 사전 지식

- DBMS와 데이터베이스의 차이

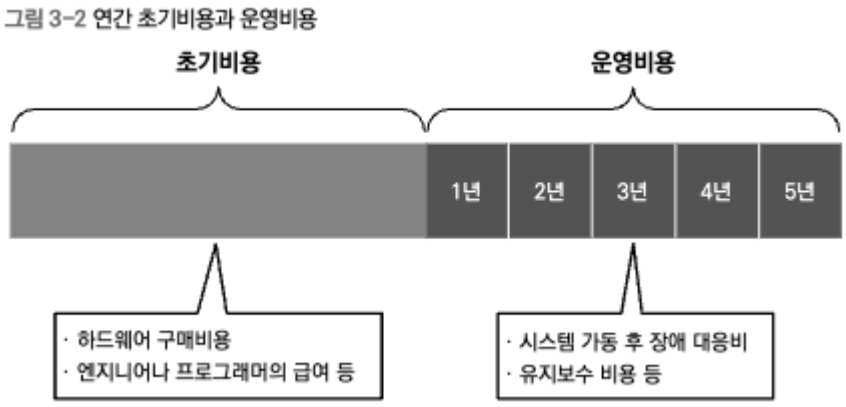
그림 2-2 데이터베이스와 DBMS의 관계



- 데이터베이스는 OS와 애플리케이션 사이에 낀 중간 소프트웨어(=미들웨어)

3장 비용

데이터베이스의 비용



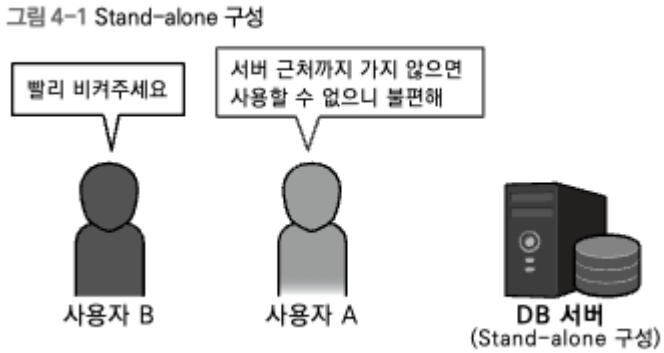
- 초기비용은 주로 라이선스료, 운영비용은 기술지원 비용(유지보수 비용)
- 서브스크립션 / 오픈소스 등도 있음
- 대규모 시스템 → Enterprise Edition

4장 데이터베이스와 아키텍처 구성

데이터베이스 아키텍처의 역사

stand-alone

네트워크에 접속하지 않고 독립되어 동작

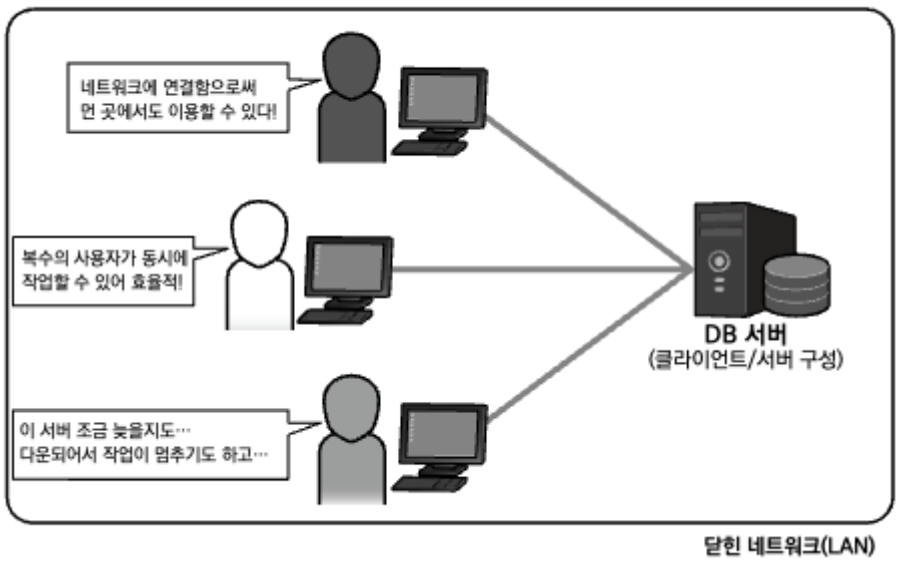


장점	단점
구축이 매우 간단함	물리적으로 떨어진 장소에서 접근할 수 없음

장점	단점
보안이 매우 높음	복수 사용자가 동시에 작업할 수 없음
	가용성이 낮음(서버가 1대)
	확장성이 부족함

클라이언트/서버

그림 4-2 클라이언트/서버 구성

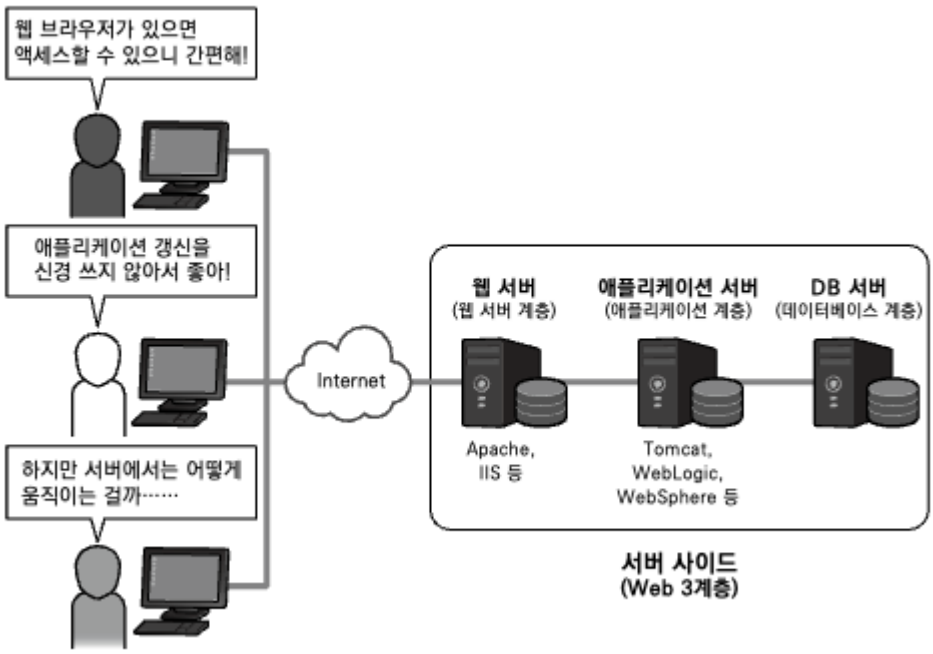


장점	단점
복수 사용자가 물리적으로 떨어진 장소에서 데이터베이스에 접속 가능	인터넷에서 직접 데이터베이스에 접속하는 것에 대한 보안 위험
	불특정 다수의 사용자가 사용하는 클라이언트에서의 애플리케이션 관리비용이 많이 듬
	개인이 이용하는 PC에 애플리케이션을 설치해 동작하게함(네이티브 애플리케이션) → 각종 환경에 대응해야 하고 버전관리, 버그 수정 버전 배포 관리가 까다로움

Web 3계층

비즈니스 로직을 실행하는 애플리케이션을 서버에서 관리

그림 4-4 Web 3계층



장점	단점
물리적으로 떨어진 장소에서 접속 가능	가용성이 낮음
복수 사용자의 동시 작업 가능	확장성이 부족함
사용자로부터 직접적인 접속 요청을 받는 역할을 웹 서버 계층에 한정하여 데이터베이스 계층의 보안 높임	

가용성과 확장성의 확보

가용성을 높이는 2가지 전략

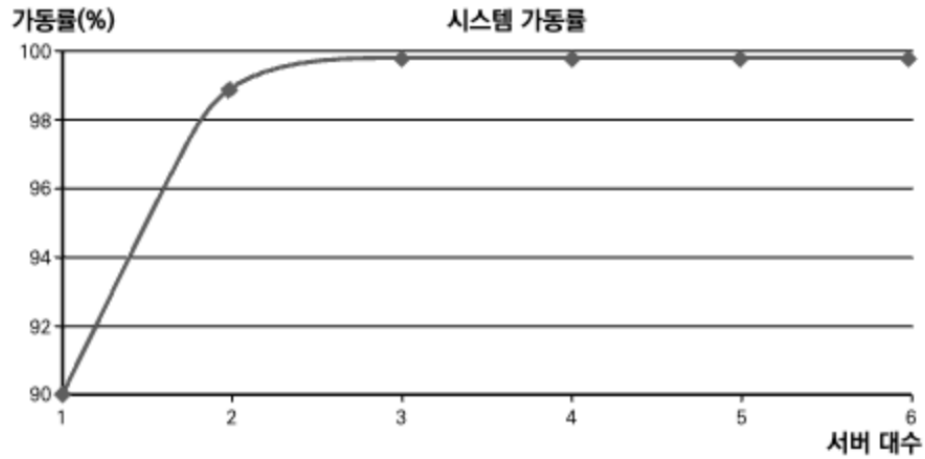
그림 4-5 심장전략과 신장전략

- 시스템을 구성하는 각 컴포넌트의 신뢰성을 높여 장애 발생률을 낮게 억제해서 가용성을 높임
- 시스템을 구성하는 각 컴포넌트의 신뢰성을 계속해서 높이기보다는 '사물은 언젠가 망가진다'란 체념을 전제로 여분을 준비. 이를 철저히 대비하는 것을 물량작전이라고 함
- 현재는 거의 후자쪽

클러스터

- 동일한 기능의 컴포넌트를 병렬화
- 클러스터 구성으로 시스템의 가동률을 높이는 것 → 다중화(여유도를 확보)

그림 4-6 시스템 가동률



단일 장애점

- 다중화되어 있지 않아서 시스템 전체 서비스의 계속성에 영향을 주는 컴포넌트

신뢰성과 가용성

- 신뢰성 → HW/SW가 고장나는 빈도나 고장 기간
- 가용성 → 사용자 입장에서 볼 때 시스템을 어느 정도 사용할 수 있는지 (따라서 신뢰성이 낮은 hw/sw를 사용한다 할지라도 다중화한다면 시스템 전체의 가용성을 높일 수 있음)

표 4-2 시스템의 가용률

가용률	서비스 정지 시간
99%	3일 15시간 36분
99.9%	8시간 46분
99.99%	52분 34초
99.999%	5분 15초
99.9999%	32초

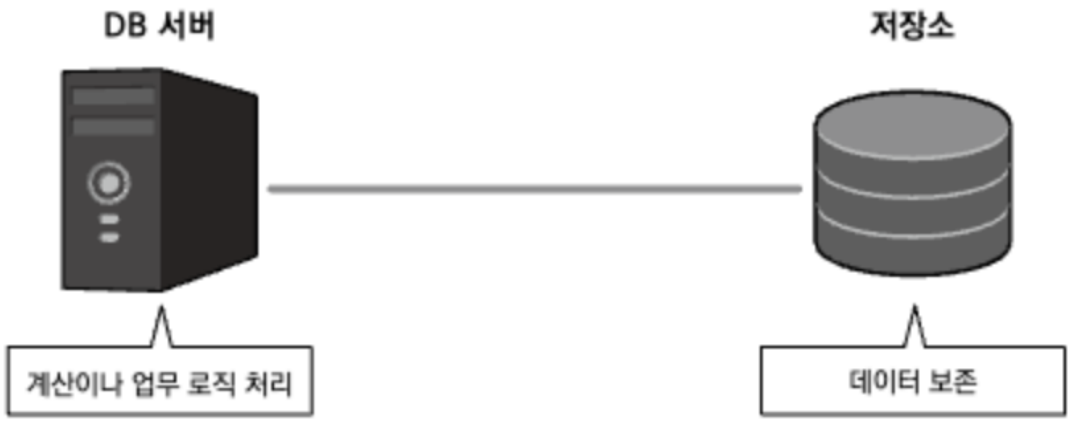
연간 정지 시간

DB 서버의 다중화 - 클러스터링

DB와 다른 서버의 차이

- 데이터베이스는 데이터를 장기간 보존하는 매체가 필요함 (DB 서버는 영속 계층)
- 대량의 데이터를 영구적으로 보존하고 성능도 요구됨

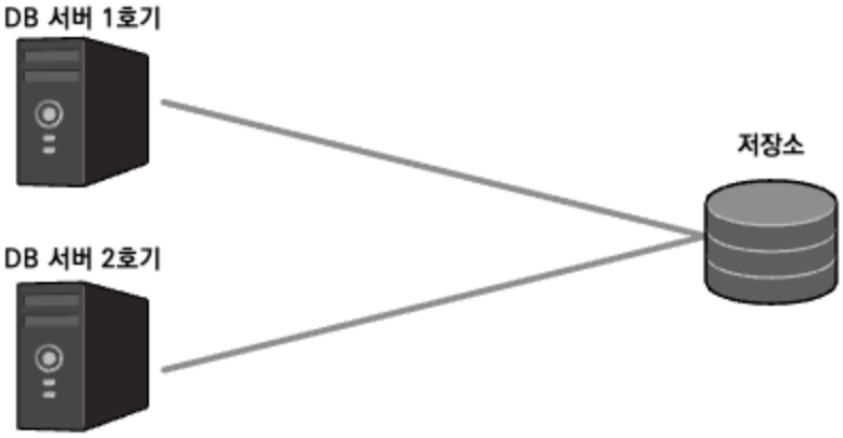
그림 4-7 데이터베이스는 서버와 저장소로 구성된다



- 데이터는 항상 갱신되기 때문에 다중화를 유지하는 중에 데이터 정합성도 중요함

가장 기본적인 다중화

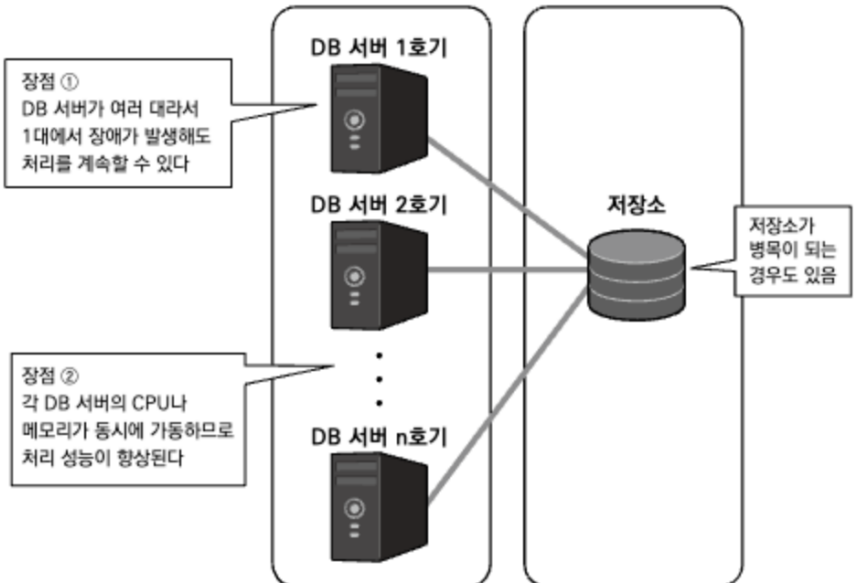
그림 4-8 DB 서버 다중화의 기본 구성



- 서버만 다중화. 정합성 신경쓰지 않아도 됨
- Active-Active : 클러스터를 구성하는 컴포넌트를 동시에 가동 → 현재는 Oracle(RAC), DB2(pureScale)만 가능
- Active-Standby : 클러스터를 구성하는 컴포넌트 중 실제 가동하는 것은 Active, 남은 것은 대기(Standby)

Active-Active 구성

그림 4-9 Active-Active 구성의 장점과 과제

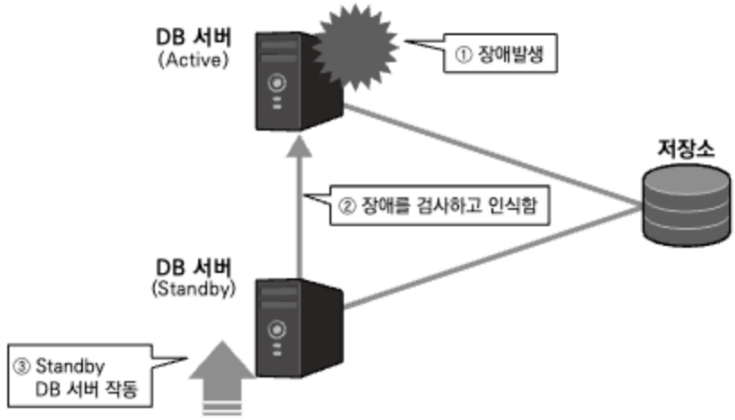


- DB 서버의 부하 분산이 가능하지만 저장소가 단일 병목 지점이 됨(성능 이슈)

Active-Standby 구성

- 장애를 검사하고 알아내는 과정

그림 4-10 Active-Standby 구성에서 장애를 검사하고 알아내는 과정



> **NOTE** Heartbeat
>
> Active-Standby 구성에서 장애가 일어났을 때 Standby DB 서버는 어떻게 Active DB 서버에 장애가 일어난 것을 알아낼까요? 사실 Standby DB 서버는 일정 간격(보통은 수 초에서 수십 초)으로 Active DB에 이상에 없는지를 조사하기 위한 통신을 하고 있습니다. 이 통신을 'Heartbeat'라고 합니다. Active DB에 장애가 발생하면 이 신호가 끊기기 때문에 Standby 측은 Active가 '죽었다'는 것을 알게 됩니다. 인간관계에서는 '무소식이 희소식이다'라고 하지만 클러스터에서 '무소식은 나쁜 소식'입니다.

보통 Standby 상태의 DB 서버는 사용되지 않다가 Active DB 서버에서 장애가 일어날 때만 사용(시차 발생)

- Active-Standby 구성의 종류
  - Cold-standby: 평소에는 Standby DB가 작동하지 않다가 Active DB가 다운된 시점에 작동
  - Hot-Standby: 평소에도 Standby DB가 작동
    - 전환 시간이 짧고 라이선스료가 높음(Active-Active보다는 쌈)

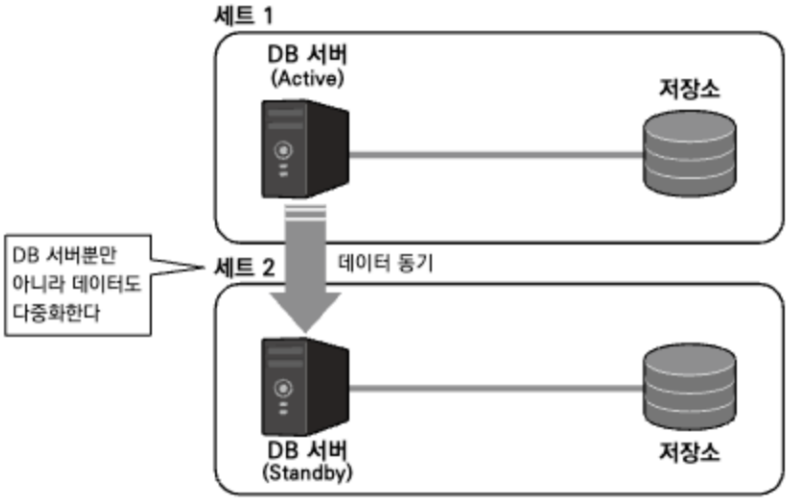
DB 서버와 데이터의 다중화 - 리플리케이션

리플리케이션

- active-active, active-standby는 저장소 부분을 다중화할 수 없음 → 데이터가 다중화되지 않음

- DB 서버와 저장소 세트를 복수로 준비함

그림 4-11 리플리케이션은 데이터를 복제한다



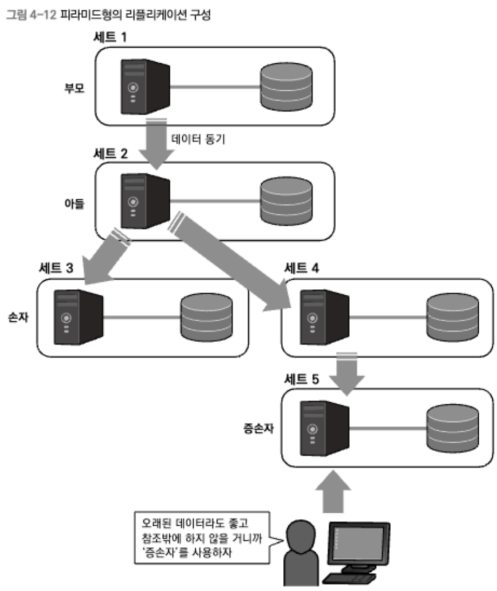
NOTE 디스크를 다중화하는 RAID

저장소 내부의 컴포넌트(대부분 하드디스크)를 다중화하는 기술을 'RAID[Redundant Array of Independent Disks]'라고 하는데, RAID에도 몇 가지 종류가 있습니다. 기본적으로 클러스터링과 동일하게 단일 장애점를 없애는 것입니다. 즉, 디스크를 병렬로 나열해 디스크 한 개가 망가져도 데이터를 소실하지 않게 하는 것입니다.

NOTE 리플리케이션 기술

리플리케이션 기술은 Oracle에서는 'Data Guard', DB2에서는 'HADR'란 이름으로 상품화되어 있습니다. 또한, 오픈소스인 MySQL도 일찍부터 리플리케이션 기술에 집중해 왔습니다. MySQL의 경우 재해 대책이라기보다는 부하 분산을 위해 리플리케이션을 발전시켜 왔다고 필자는 생각합니다(MySQL과 리플리케이션 참조).

- 리플리케이션은 DB 서버와 저장소가 동시에 불능일 때 다른 1세트가 멀리 떨어진 지점에 놓여있으면 서비스를 계속하는 것이 가능하기 때문에 매우 가용성이 높은 아키텍처 → 재해대책(DR)으로 이용되는 경우도 있음
- Active 측 저장소의 데이터는 항상 사용자로부터 갱신됨 → Standby 측에도 갱신을 반영하여 최신화(동기화)하지 않으면 데이터 정합성을 유지할 수 없음
- Standby 측의 갱신 주기를 얼마로 할 것인가와 성능 사이의 트레이드오프 관계가 생김
- 피라미드형 리플리케이션



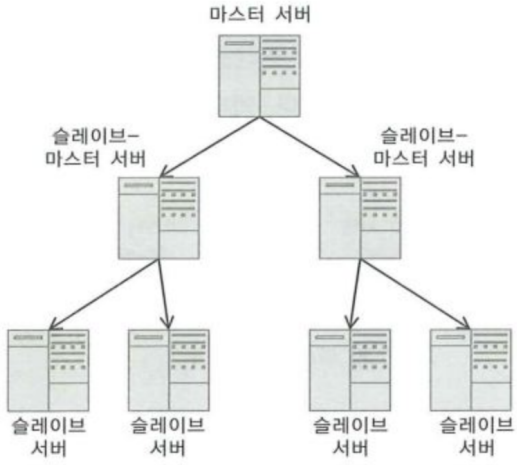


그림 7.38 피라미트형 구성

→ 데이터가 오래되어도 참조만 하면 된다는 처리를 손자/증손자 세트에서 하여 부모에 걸리는 부하를 분산함 (손자/증손자 or 마스터/슬레이브)

성능을 추구하기 위한 다중화 - Shared Nothing

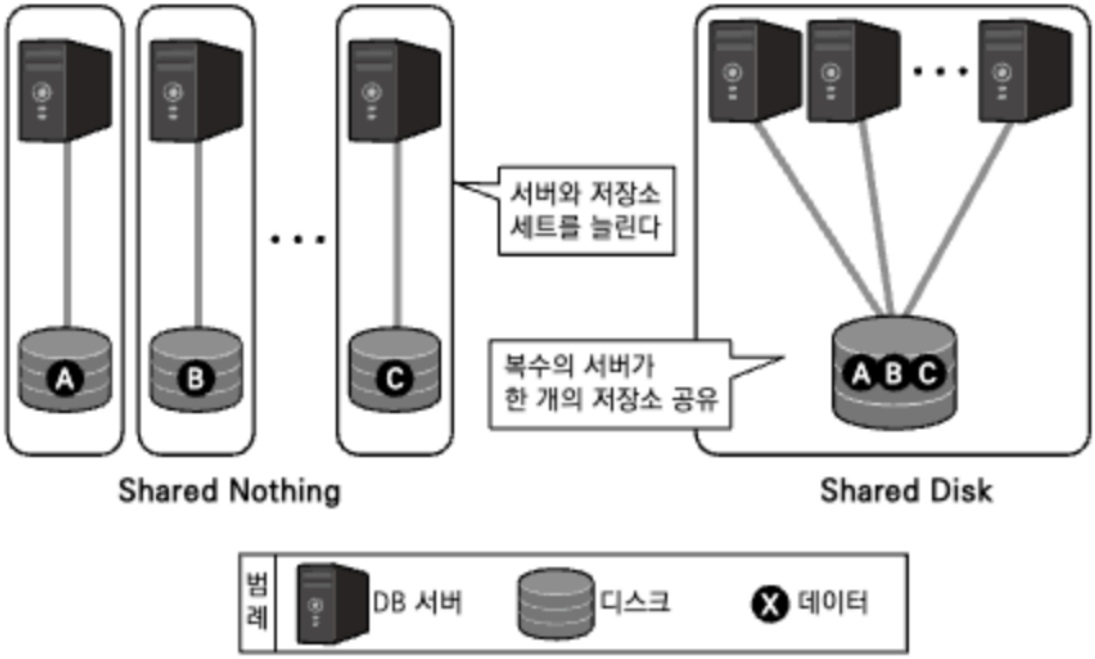
Shared Disk

- 복수의 서버가 1대의 디스크(저장소)를 고융하도록 구성

Shared Nothing

- 네트워크 이외의 자원을 모두 분리하는 방식
- 서버와 저장소의 세트를 늘리면 병렬처리 때문에 선형적으로 성능이 향상

그림 4-13 Shared Disk와 Shared Nothing의 구성 비교



- Sharding (Shared Nothing을 구현한 기법)

  ▼ 레퍼런스

  - https://aws.amazon.com/ko/what-is/database-sharding/
  - https://techblog.woowahan.com/2687/
  - https://jaehoney.tistory.com/245

- 단점
  - 저장소를 공유하지 않는다 = 각각의 DB 서버가 동일한 1개의 데이터에 액세스할 수 없다
  - DB 서버 하나가 다운되었을 때 다른 DB 서버가 이를 이어받아 계속 처리할 수 있게 하는 커버링 구성을 고려해야 함 (MySQL은 MySQL Cluster라는 Shared Nothing 타입의 클러스터 구성이 가능)

적정 아키텍처

그림 4-15 데이터베이스의 아키텍처 패턴

- 아키텍처는 변경이 어렵다
- 비용, 시간, 인적자원의 트레이드오프
- MySQL은 리플리케이션 기능이 에디션에 상관없이 기본기능 → 초기비용을 낮출 수 있음