ch5

- `상태(데이터 등)`를 보존하기 위해 데이터 서비스를 사용함.

5.1 클라우드 네이티브 시스템을 위한 데이터베이스

- 예전에야 DB 구축시 시간이 많이 걸렸지만 현재는 너무 간단해짐.

클라우드에서의 데이터 서비스

- 데이터 서비스는 상태를 저장하고자 설계된 클라우드 네이티브 아키텍처의 구성 요소

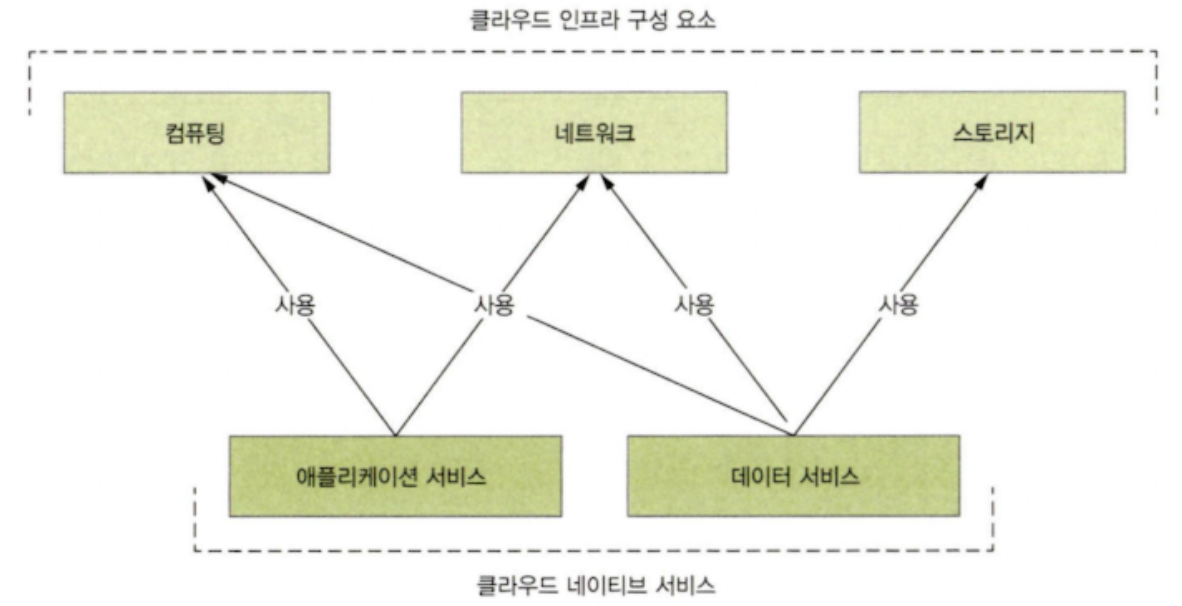

그림 5.1 애플리케이션 서비스(상태가 없음)는 클라우드 인프라에서 컴퓨팅 및 네트워크 자원만 사용한다. 데이터 서비스(상태가 있음)는 스토리지도 필요하다.

- 클라우드 인프라의 세가지 기본 구성 요소
  1. 컴퓨팅
  2. 네트워크

3. 스토리지(저장)

- 애플리케이션 서비스는 상태가 없음.
- 데이터 서비스는 상태 유지를 위해 스토리지 필요.

적합한 데이터 서비스를 사용하기 위한 고려 요소

- **확장성**
  - 데이터 서비스는 워크로드 증가 가능성을 지원하고 탄력적으로 대응해야 함.
- **복원력**
  - 데이터 손실을 방지위한 핵심 전략 중 하나는 복제.
  - (비관계형 데이터베이스는 높은 수준의 복원력을 제공하지만, 데이터 일관성이 항상 보장되진 않음)
- **성능**
  - 스토리지 기술의 I/O 대기 시간과 네트워크 시간에 의해서 제한됨.
  - 스토리지의 위치 중요
- **컴플라이언스**
  - 지속성 데이터 관리를 위한 법률이나 계약을 따라야 함.
  - 민감한 데이터를 다루는 조직은 데이터 관리를 보다 잘 제어하고 규정 준수를 위해 본인들이 직접 관리하는 장소에 두고 사용하는 것을 선호.

데이터 서비스의 범주

- **산업 표준 서비스 (e.g. postgreSQL, 레디스, 마리아 DB 등)**
  - 일부 클라우드 업체는 여기에 기능을 최적화 및 향상시켜 제공 (e.g. RDS, 애저 데이터베이스, 구글 클라우드 SQL 등)
- **클라우드 전용으로 제작된 고유한 API를 통해 사용하는 서비스 (e.g. 빅쿼리, 코스모스 DB 등)**
- **데이터 서비스를 직접 관리**

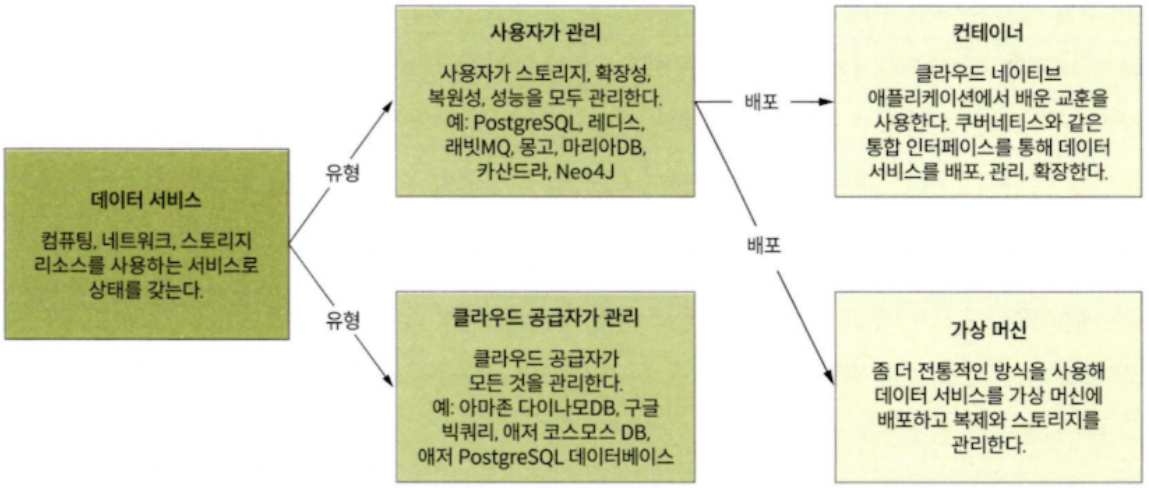

그림 5.2 데이터 서비스는 사용자(컨테이너 또는 가상 머신) 스스로 또는 클라우드 공급자가 관리할 수 있다.
첫 번째 경우는 좀 더 전통적인 방식의 서비스를 사용할 수 있지만
두 번째 경우에는 공급 업체가 클라우드를 위해 특별히 구축한 여러 형태의 서비스를 사용할 수 있다.

스프링 데이터에 대한 데이터 지속성 JDBC

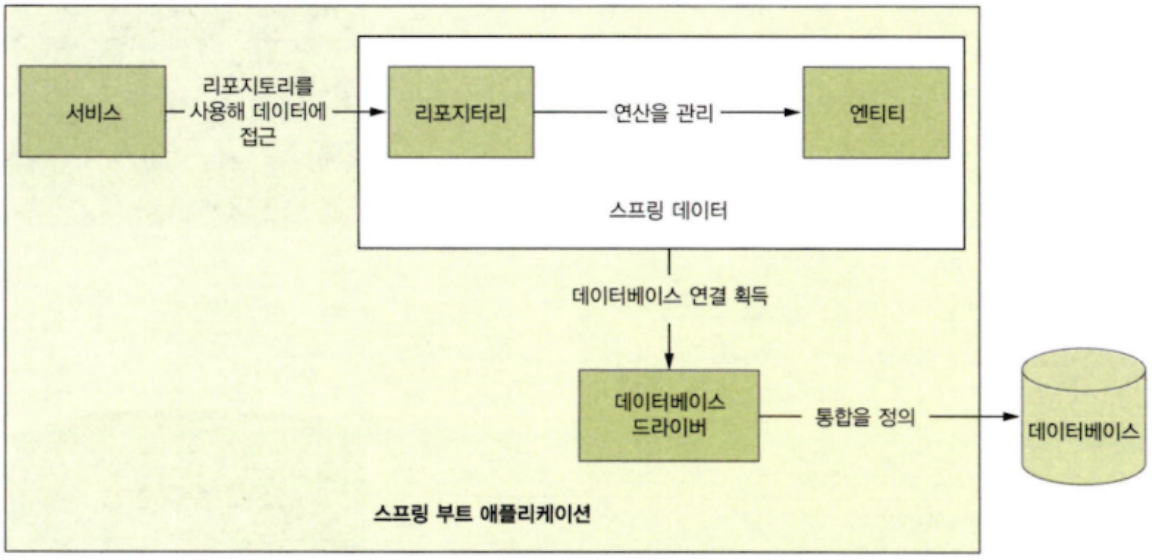

그림 5.3 드라이버는 애플리케이션과 데이터베이스 간의 연결을 설정한다.
엔티티는 도메인 객체를 나타내며 리포지터리를 통해 저장 및 검색할 수 있다.

- **데이터베이스 드라이버**
  - 연결 팩토리를 통해 특정 데이터베이스와의 통합을 제공하는 구성 요소
    - JDBC, R2DBC

- **엔티티**
  - 각 인스턴스를 고유하게 식별하는 필드(기본키)를 포함해야 하며, DB에 저장되는 도메인 객체
- **리포지터리**
  - 데이터 저장 및 검색을 위한 추상화

(JDBC로 DB 연결 실습)

- DB연결을 열고 닫는 것은 비용이 많이 든다. → 연결풀링(connection pooling)을 통해 애플리케이션이 데이터를 액세스 할 때마다 연결을 새로 만드는 대신 이미 생성한 연결을 재사용하여 성능 최적화를 한다.
  - 스프링부트는 hikari CP 사용.

- 도메인 클래스에 자바 레코드를 사용하는 것이 좋다. (불가변 엔티티 사용해야 하기 때문)
  - 대신 JPA는 가변객체 사용하기 때문에 자바 레코드 사용이 불가 → JPA 엔티티 클래스는 `@Entity` 애너테이션으로 표시하며 인수를 갖지 않는 기본 생성자를 가져야 함.
- 동일 엔티티가 동시에 업데이트 된다면 스프링 데이터 JDBC는 '낙관적 잠금' 을 지원한다.
  - `@Version` 애너테이션
- `@EnableJdbcAuditing` 애너테이션을 달아 데이터베이스 감사(엔티티 생성, 업데이트 추적) 를 활성화한다.
  - 데이터의 생성, 변경, 삭제가 일어날 때마다 감사 이벤트 생성
  - `@CratedBy` 등의 애너테이션을 사용해 감사 메타데이터 캡쳐 가능

스프링 데이터의 데이터 리포지터리

- 리포지터리 패턴은 데이터 소스와 독립적으로 데이터에 액세스 하기 위한 추상화 제공

(리포지터리 만들며 실습)

- JPA 사용시 특정 명명 규칙에 따라 쿼리 구현이 가능하다.

표 5.2 특정 명명 규칙에 따라 구성 요소를 조합해 사용자 지정 쿼리를 리포지터리에 추가하고 스프링 데이터로 구현하여 생성할 수 있다.

리포지터리 메서드 구성 요소	예
Action	`find, exists, delete, count`
Limit	`One, All, First10`
-	`By`
Property expression	`findByIsbn, findByTitleAndAuthor, findbyAuthorOrPrice`
Comparison	`findByTitleContaining, findByIsbnEndingWith, findByPriceLessThan`
Ordering operator	`orderByTitleAsc, orderByTitleDesc`

- 스프링 데이터는 모든 작업에 대해 트랜잭션 콘텍스트를 설정
- 하지만 사용자 정의 쿼리 메서드를 추가할 땐 트랜잭션을 명시적으로 지정해줘야 함.
    - `@Transactional`

- 데이터 소스가 환경마다 다른 점은 환경 간 차이를 만든다.
    - 로컬에서 인메모리db를 자주 사용 하는 서브 이유는 통합 테스트에서 사용하기 쉽기 때문.
    - → 하지만 통테는 외부 서비스의 통합도 테스트 해야하는데, h2 등을 사용하면 신뢰성이 저하된다.

⇒ 이런 부분 때문에 테스트 컨테이너를 통해 통테에서 컨테이너로 지원 서비스를 사용할 수 있게 한다.

- 슬라이스 테스트 → 특정 슬라이스에서 사용하는 스프링 구성 요소만 로드하여 통합 테스트 실행
- `@DataJdbcTest` 는 많은 편리한 기능을 캡슐화
    - 각 테스트 메서드를 트랜잭션으로 실행 후, 실행이 끝날 때마다 롤백 수행

(실습중 트러블 슈팅)

- 테스트 컨테이너 접속 오류

플라이웨이를 통한 프로덕션 환경에서의 데이터베이스 관리

- 데이터베이스의 변경 사항을 형상 관리 시스템을 통해 관리할 때 많이 사용하는 도구 도 가지는 플라이웨이, 리퀴베이스 이다.

플라이웨이 이해

- 데이터베이스 상태 버전에 대한 단일 진실 원천(single source of truth)를 제공
- 변경사항을 점진적으로 추적
- 데이터베이스 변경을 자동화하고 데이터베이스 상태를 재생하거나 롤백 가능

- 데이터베이스 변경 → 마이그레이션
  - 버전 마이그레이션 → 고유한 버전 번호로 식별되며 한번씩 순서대로 적용. 변경 사항 되돌릴 땐 실행 취소 마이그레이션 가능
    - 스키마, 테이블, 컬럼 및 시퀀스와 같은 관계형 객체의 cud와 데이터 수정을 위해 사용 가능
  - 반복 마이그레이션 → 체크섬이 바뀔 때마다 반복적으로 적용
    - 뷰, 프로시저, 패키지의 생성과 변경에 사용 가능