영속성 관리 (Persistence Layer)

> JPA가 제공하는 기능은 크게 두 가지로 나눌 수 있습니다.

- 설계 부분
  - 엔티티와 실제 DB 테이블 매핑
- 사용 부분
  - DB 테이블과 매핑된 엔티티 사용 엔티티를 저장, 수정, 삭제, 조회

> 해당 과정은 엔티티 매니저가 처리해줍니다.

- 엔티티 매니저는 엔티티 CRUD를 담당하는 한 단계 추상화된 가상의 데이터베이스로 생각할 수 있습니다.

Spring Data JPA

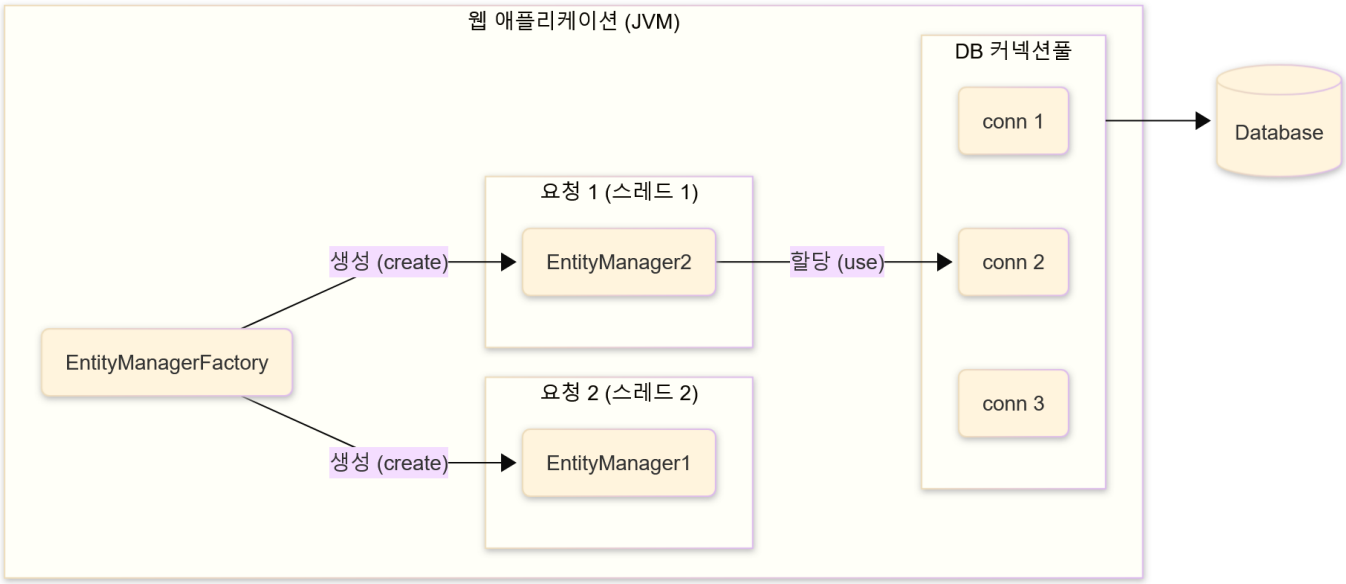
- Spring 에서 JPA를 더 쉽게 만들 수 있도록 도와주는 데이터 접근 계층 모듈
- Spring Data JPA의 JpaRepository에서 제공하는 메서드들은 내부적으로 엔티티 매니저를 이용해 JPA의 표준 API(persist, find, merge, remove 등)을 호출합니다.

> 즉, Spring Data JPA의 동작을 이해하기 위해서는 JPA의 기본 함수를 먼저 살펴봐야 합니다.

엔티티 매니저 팩토리와 엔티티 매니저

- 엔티티 매니저 팩토리는 엔티티 매니저를 효율적으로 생성하는 공장입니다.
- 엔티티 매니저 팩토리를 통해서 필요할 때마다 엔티티 매니저를 생성하는 구조입니다.

일반적인 웹 애플리케이션



- 엔티티 매니저 팩토리 생성 비용이 상당히 크기 때문에, 한 개만 만들어서 애플리케이션 전체에서 공유해서 사용합니다.

- 즉, 엔티티 매니저 팩토리는 여러 스레드가 동시에 접근해도 안전합니다.
- 단, 엔티티 매니저는 요청을 처리하는 각각의 스레드 개수만큼 독립적으로 만들어서 사용해야 합니다.
- 엔티티 매니저는 데이터베이스 연결이 꼭 필요한 시점에만 커넥션 풀에서 커넥션을 얻습니다. 일반적으로 트랜잭션이 시작하는 시점에 커넥션을 획득합니다.

EntityManagerFactory

```
EntityManagerFactory factory = Persistence.createEntityMangerFactory("jpabook");
```

EntityManager

```
EntityManager em = factory.createEntityManager();
```

persistence.xml 예시 (참고)

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="https://jakarta.ee/xml/ns/persistence"
             version="3.0">

  <!--jpabook-->
  <persistence-unit name="jpabook">
      <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>

      <class>com.example.jpabook.domain.Member</class>

      <properties>
          <!-- DB 연결 정보 -->
          <property name="jakarta.persistence.jdbc.driver"
value="com.mysql.cj.jdbc.Driver"/>
          <property name="jakarta.persistence.jdbc.url"
value="jdbc:mysql://localhost:3306/jpabook"/>
          <property name="jakarta.persistence.jdbc.user" value="root"/>
          <property name="jakarta.persistence.jdbc.password" value="1234"/>

          <!-- Hibernate 옵션 -->
          <property name="hibernate.dialect"
value="org.hibernate.dialect.MySQLDialect"/>
          <property name="hibernate.hbm2ddl.auto" value="update"/>
          <property name="hibernate.show_sql" value="true"/>
      </properties>
  </persistence-unit>

</persistence>
```

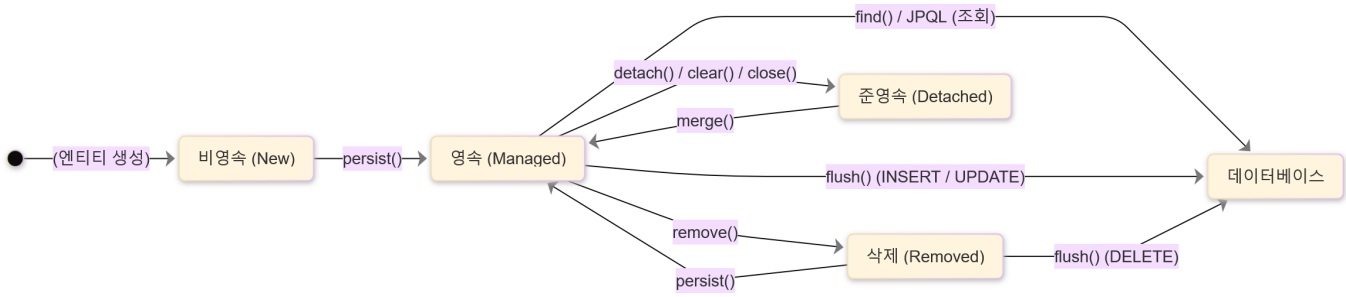
영속성 컨텍스트란?

> Persistence Context : 엔티티를 영구적으로 저장하는 환경

```
em.persist(member);
```

- JPA를 이해하기 위해서는 영속성 컨텍스트라는 용어를 이해해야 합니다.
- 앞에서 엔티티 매니저가 persist함수를 호출하면 DB가 아닌 영속성 컨텍스트라는 곳에 저장하게 됩니다.

엔티티의 생명주기



> 엔티티에는 4가지 상태가 존재합니다.

- 비영속
  - 영속성 컨텍스트와 전혀 관계가 없는 상태

```
Member member = new Member();
member.setUsername("테스트 이름");
```

- 영속
  - 영속성 컨텍스트에 저장된 상태

```
em.persist(member);
```

- 준영속
  - 영속성 컨텍스트에 저장되었다가 분리된 상태

```
// 엔티티를 영속성 컨텍스트로부터 분리
em.detach(member);
```

- 삭제
  - 삭제된 상태

```

```

```
// 엔티티를 영속성 컨텍스트 & 데이터베이스에서 삭제하는 용도
em.remove(member);
```

영속성 컨텍스트의 특징

- 식별자 값(@Id) 필수
  - 영속성 컨텍스트는 식별자 값을 key로 사용하는 Map으로 볼 수 있습니다. 그래서 영속 상태에서는 식별자 값이 반드시 존재해야 합니다.

```
Map.entry<(식별자), (엔티티)>
```

- 데이터베이스 저장
  - 영속성 컨텍스트의 변경사항을 flush를 통해 데이터베이스에 반영할 수 있습니다.

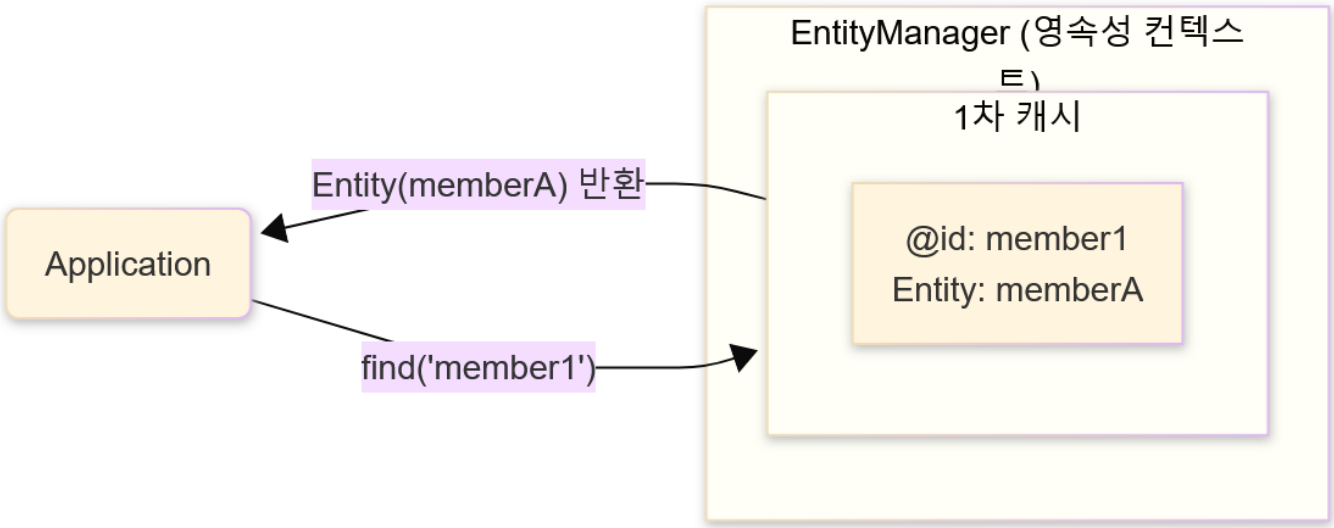
엔티티 조회 - 1차 캐시

> 영속성 컨텍스트의 내부 캐시를 1차 캐시라고 합니다. 영속 상태의 엔티티가 저장되는 공간입니다.

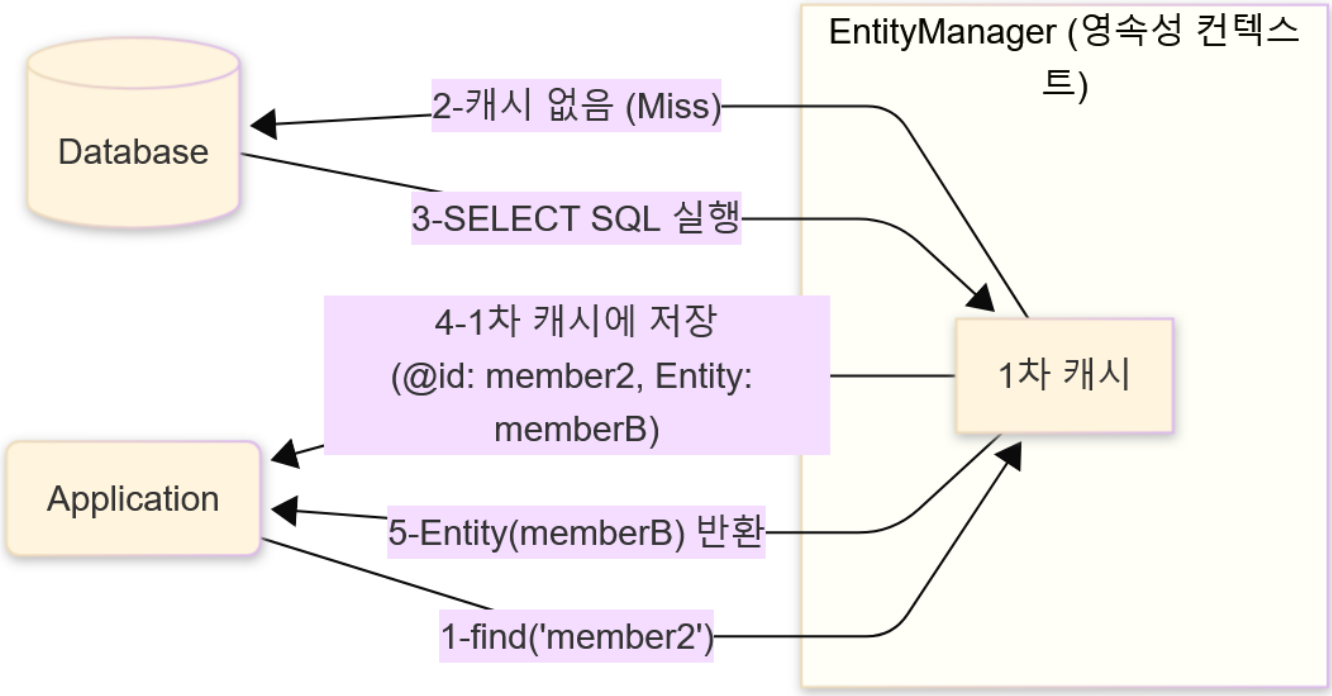
```
Member member = em.find(Member.class, "member1");
```

- 영속성 컨텍스트에서 식별자를 통해 저장된 엔티티 인스턴스를 조회할 수 있습니다.

찾는 엔티티가 메모리에 존재하는 1차 캐시에 존재하는 경우

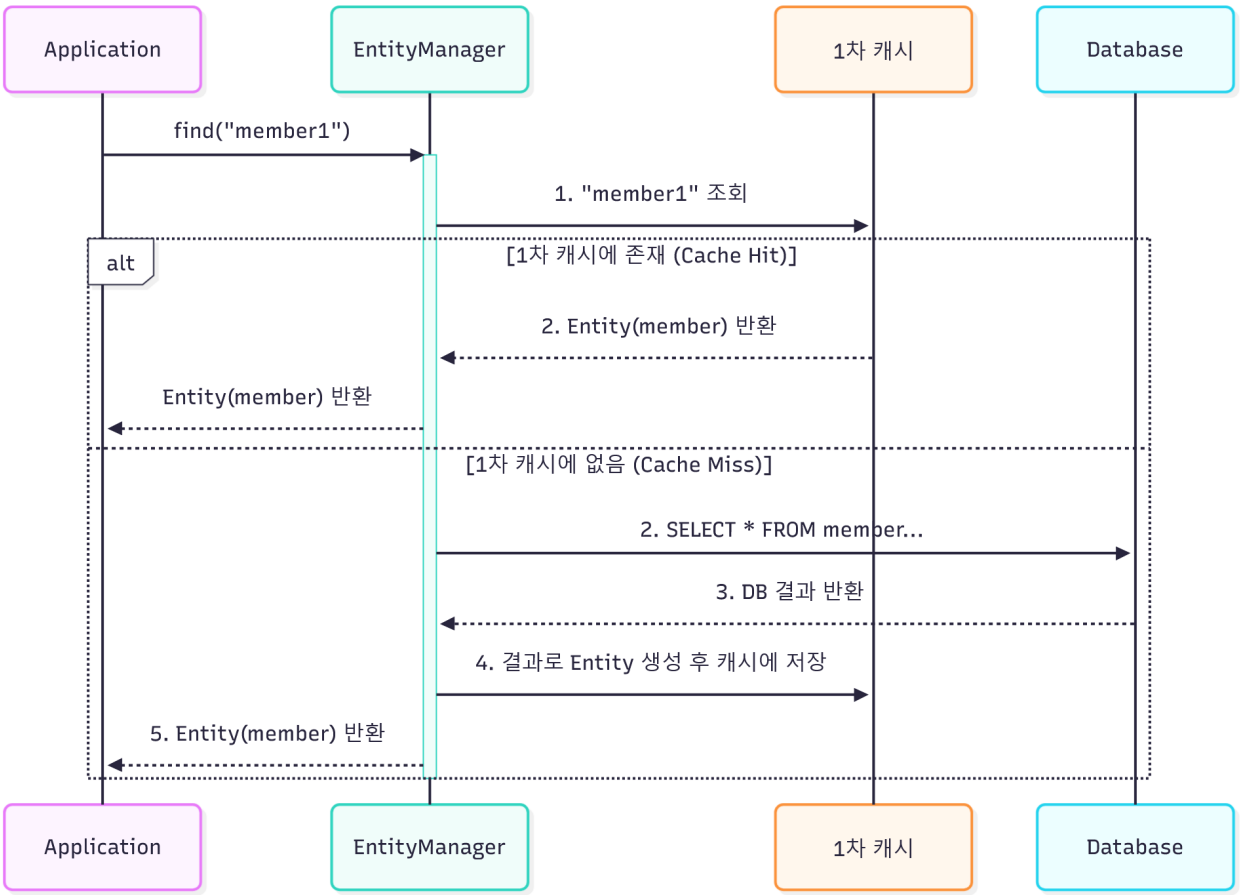


찾는 엔티티가 1차 캐시에 없는 경우



1. 데이터베이스 조회해서 엔티티 생성
2. 1차 캐시에 저장
3. 영속 상태의 엔티티 반환

엔티티 조회 요약

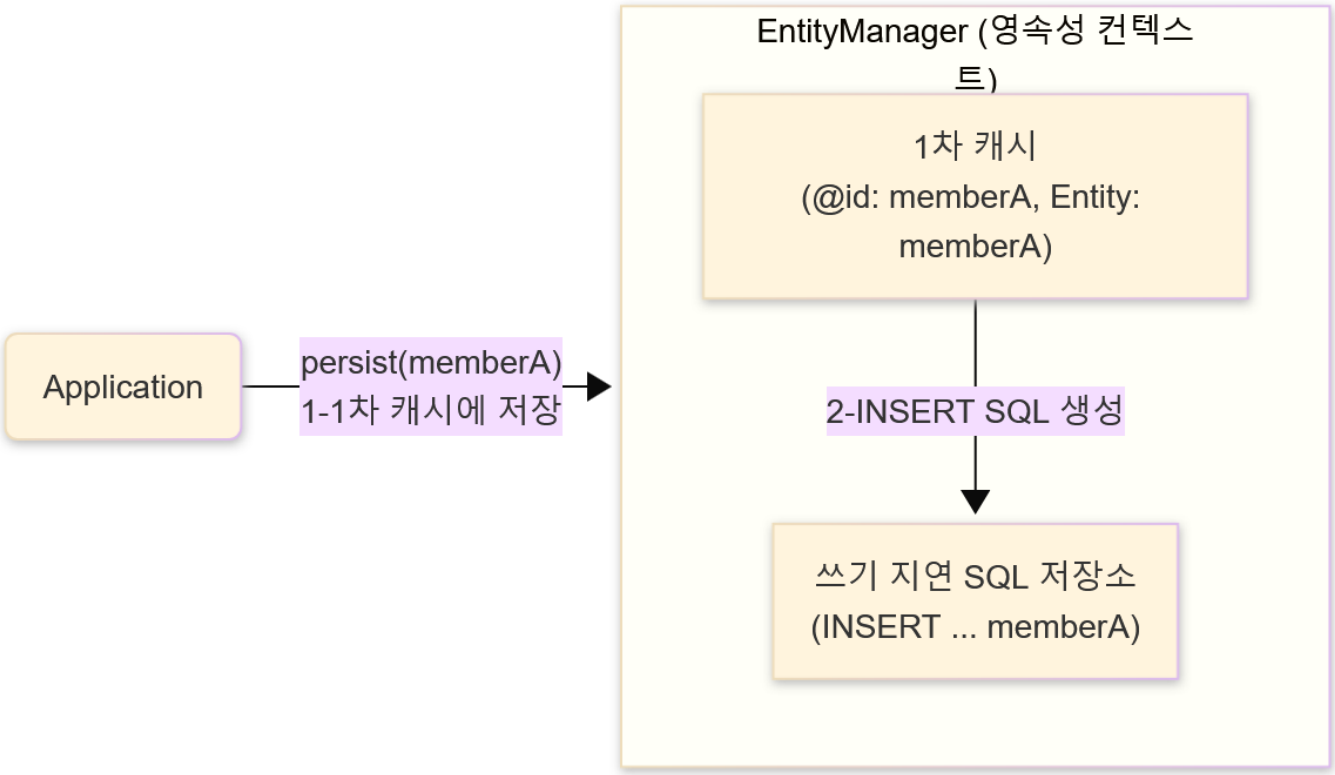


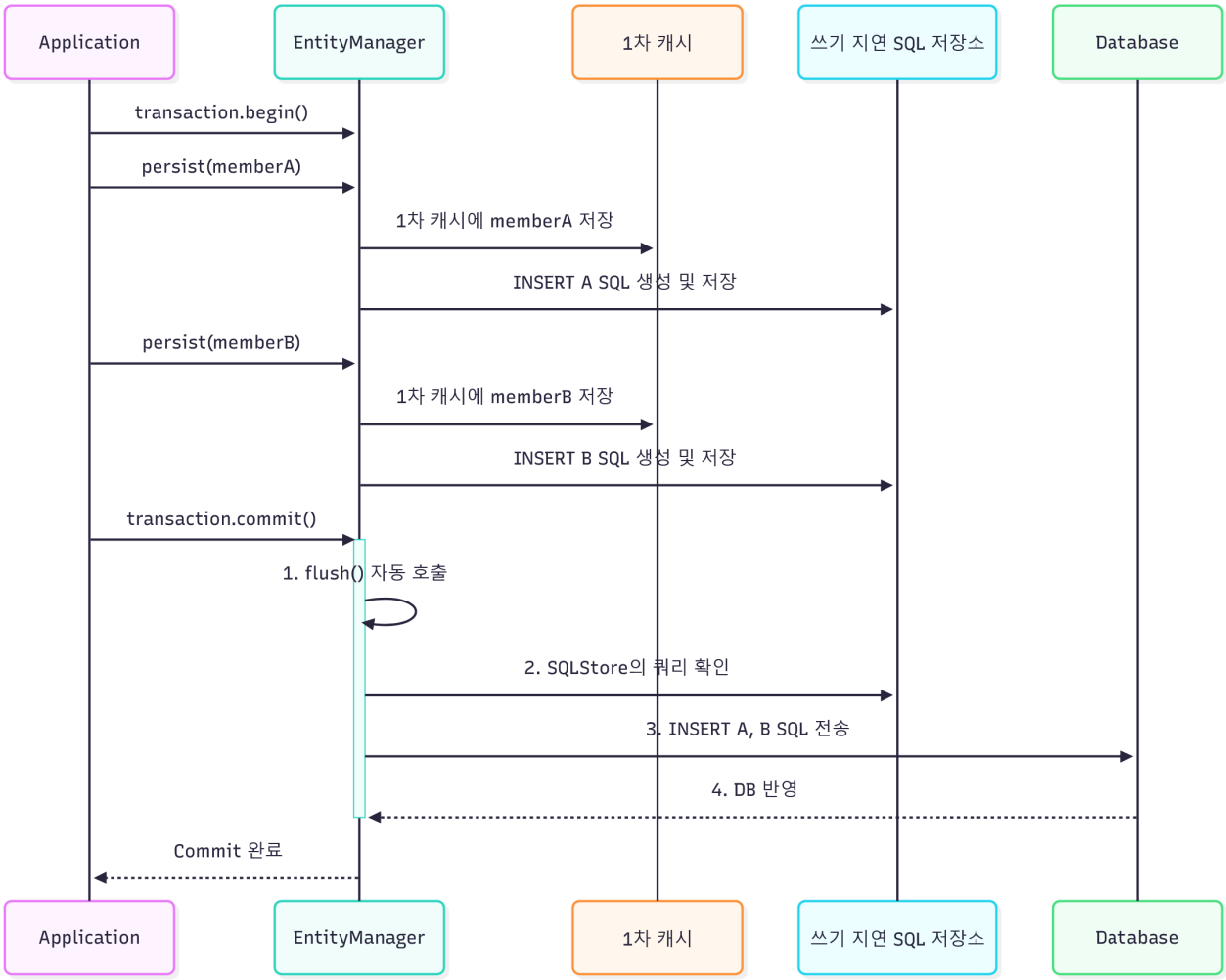
엔티티 등록(저장)

```
// 엔티티 매니저는 데이터 변경 시 트랜잭션을 시작해야 합니다.
// 트랜잭션 시작
transaction.begin();

em.persist(memberA);
em.persist(memberB);

// 트랜잭션 커밋
transaction.commit();
```

트랜잭션을 지원하는 쓰기 지연 (transactional write-behind)

- 엔티티 매니저는 트랜잭션 커밋 전까지 엔티티를 데이터베이스에 저장하지 않고 쓰기 지연 SQL 저장소에 INSERT SQL을 모아둡니다. 그리고 트랜잭션 커밋 시점에 모아둔 쿼리를 보내는 형식으로 동작합니다.
- 삭제 연산도 쓰기 지연 SQL 저장소를 활용합니다.

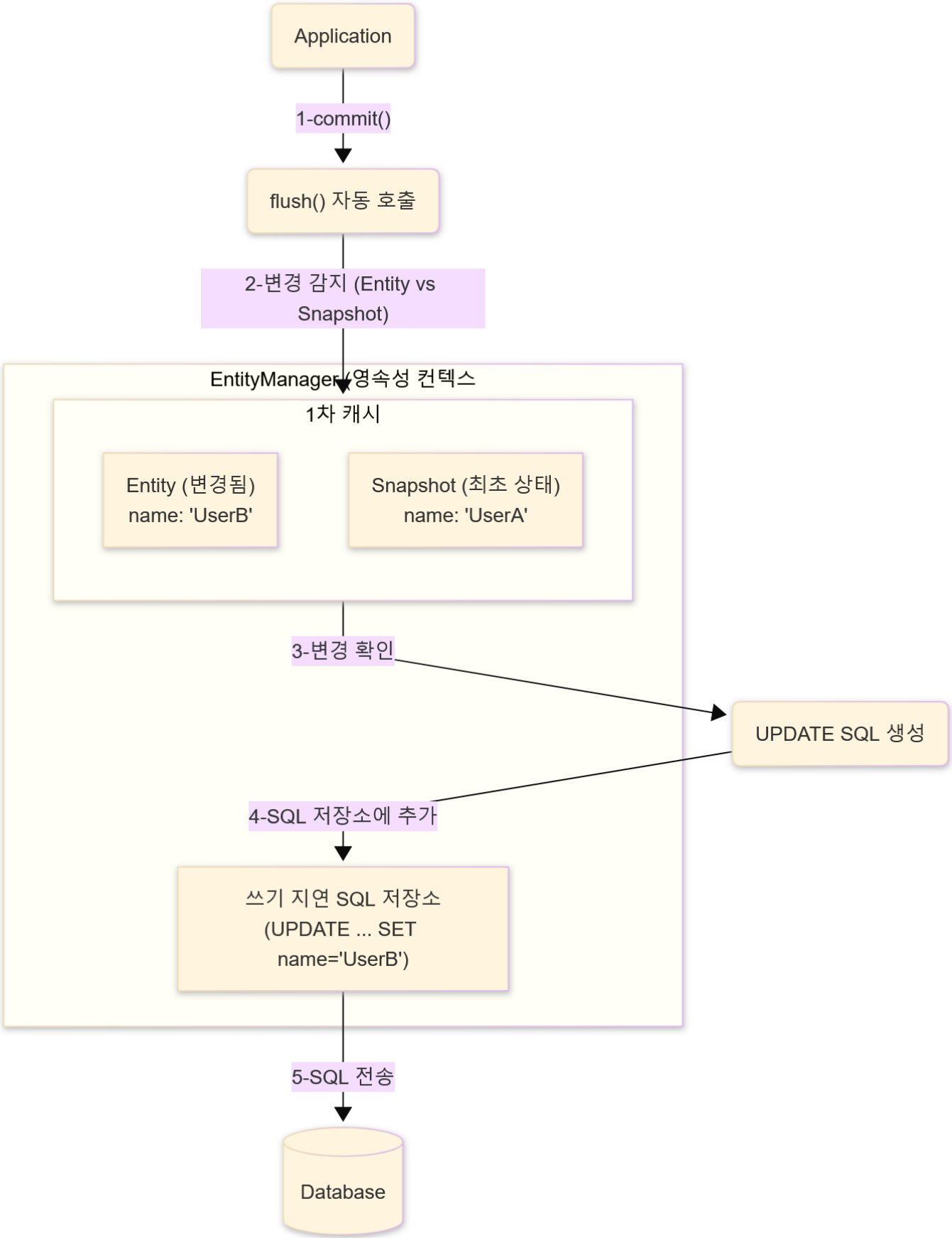




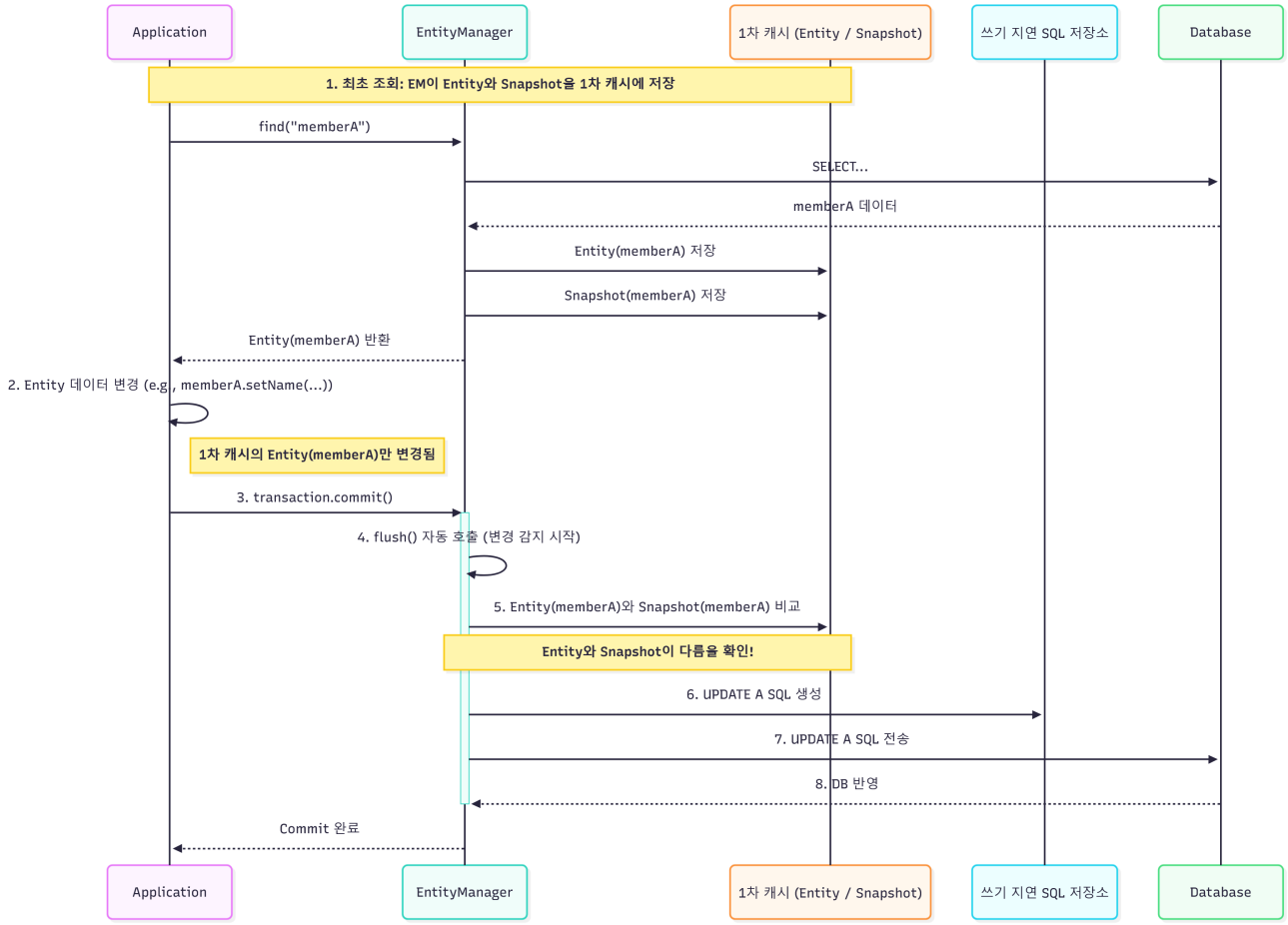
엔티티 수정

```
transaction.begin();

Member memberA = em.find(Member.class, "memberA");

memberA.setUsername("hihi");
memberA.setAge(25);

// 별도의 update함수가 없습니다.
transaction.commit();
```

- 엔티티 수정 시에는 단순히 엔티티를 조회해서 데이터를 변경하면 됩니다.
- 엔티티의 변경사항을 데이터베이스에 자동으로 반영하는 기능을 변경 감지(dirty_check)라고 합니다.



- JPA는 엔티티를 영속성 컨텍스트에 보관할 때 최초 상태를 복사해서 스냅샷을 만듭니다. 그리고 flush 시점에 스냅샷과 엔티티를 비교해서 변경된 엔티티를 찾습니다.
- 변경된 엔티티가 있다면 수정 쿼리를 쓰기 지연 SQL 저장소에 추가합니다.
- 수정 쿼리가 commit 시점에 자연스럽게 반영됩니다.



플러시

> 플러시(flush)는 영속성 컨텍스트의 변경 내용을 데이터베이스에 반영하는 연산입니다.

flush()가 발생하는 경우

1. em.flush() 직접 호출
2. 트랜잭션 커밋 시
3. JPQL 쿼리 실행 시
    - JPQL은 실제 DB를 조회하기 때문

요약

- 엔티티 매니저 팩토리를 통해 엔티티 매니저를 생성할 수 있다.
- 엔티티 매니저를 만들면 내부에 영속성 컨텍스트도 함께 만들어지는데, 이는 엔티티 매니저를 통해 접근할 수 있다.
- 영속성 컨텍스트가 애플리케이션과 데이터베이스 사이를 한 단계 추상화한다.
    - 영속성 컨텍스트가 제공하는 기능
        - 1차 캐시
        - 동일성 보장
        - 트랜잭션을 지원하는 쓰기 지연
        - 변경 감지
        - 지연 로딩

- 영속성 컨텍스트에 저장된 엔티티는 플러시 시점에 데이터베이스에 반영된다.