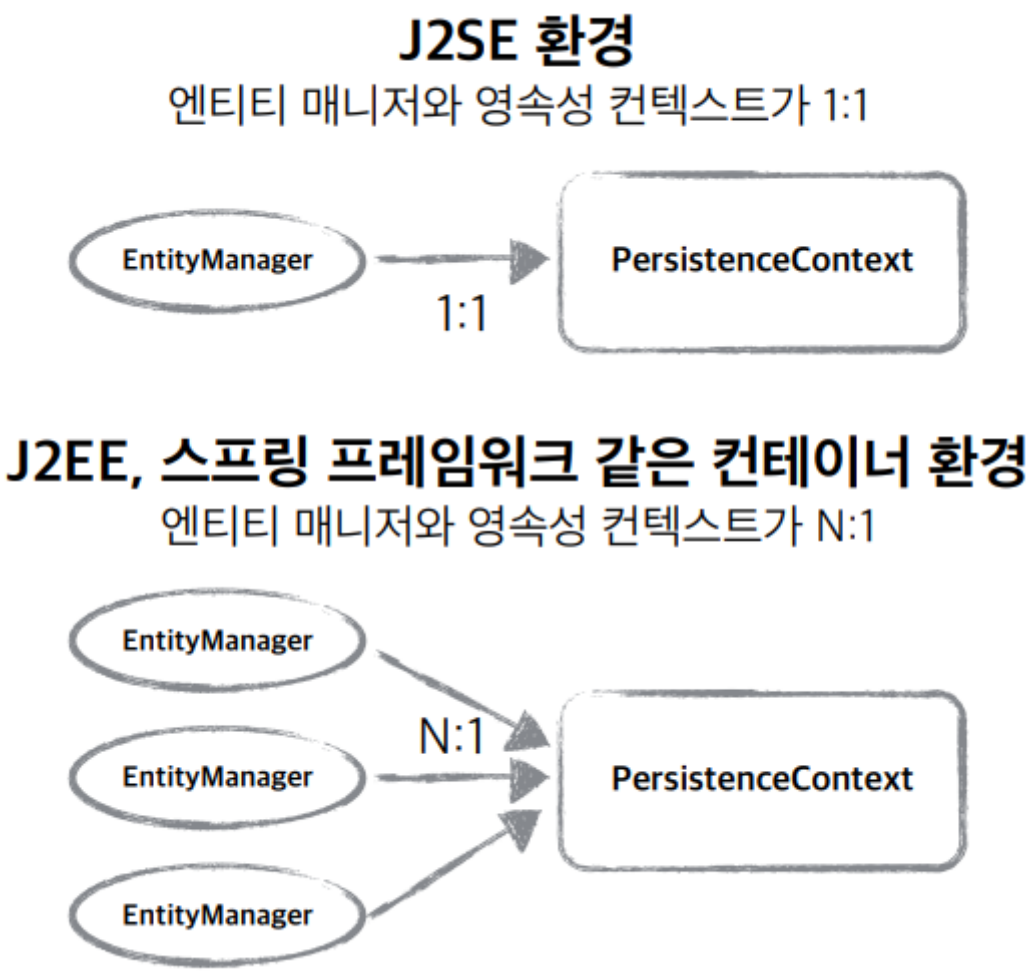
영속성 컨텍스트

> EntityManager.persist(entity);

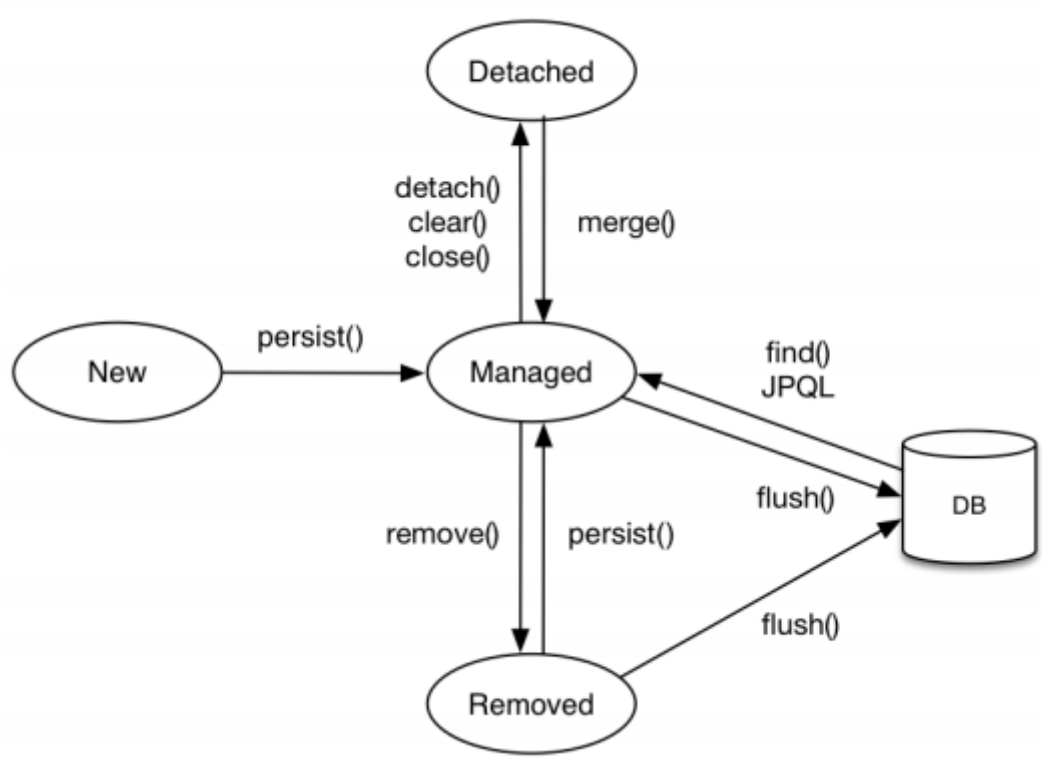
엔티티를 영구 저장하는 환경

엔티티 매니저를 통해서 영속성 컨텍스트에 접근



엔티티 생명주기

- **비영속 (new/transient)** 영속성 컨텍스트와 전혀 관계가 없는 새로운 상태
- **영속 (managed)** 영속성 컨텍스트에 관리되는 상태
- **준영속 (detached)** 영속성 컨텍스트에 저장되었다가 분리된 상태
- **삭제 (removed)** 삭제된 상태




```
// 비영속 상태.
Member member = new Member();
member.setId(3L);
member.setName("user02");


// 영속 상태.
entityManager.persist(member);

// 준영속 상태.
entityManager.detach(member);

// 삭제 상태.
// entityManager.remove(member);

transaction.commit(); // 영속성 컨텍스트에 남아있는 쿼리를 한꺼번에 날림.
```

준영속 상태로 만드는 방법

- entityManager.detach(entity) - 특정 엔티티만 준영속 상태로 전환
- entityManager.clear() - 영속성 컨텍스트를 완전히 초기화
- entityManager.close() - 영속성 컨텍스트를 종료

영속성 컨텍스트의 목적

- 1차 캐시
- 동일성(identity) 보장
- 트랜잭션을 지원하는 쓰기 지연 (transactional write-behind)
- 변경 감지(Dirty Checking)

- 지연 로딩(Lazy Loading)

1차 캐시

```
Member member = new Member();
member.setId(3L);
member.setName("user02");


entityManager.persist(member);

Member m = entityManager.find(Member.class, 3L);
System.out.println("m : " + m); // m : Member(id=3, name=user02)

entityManager.detach(member); // 비영속 상태로 변경

Member m2 = entityManager.find(Member.class, 3L);
System.out.println("m2 : " + m2); // m2 : null

transaction.commit();
```

동일성 보장

```
Member m1 = entityManager.find(Member.class, 1L);
Member m2 = entityManager.find(Member.class, 1L);

System.out.println("m1 : " + m1);
System.out.println("m2 : " + m2);
System.out.println("m1 == m2 : " + (m1 == m2));
```

결과

```
m1 : Member(id=1, name=pikachu)
m2 : Member(id=1, name=pikachu)
m1 == m2 : true
```

이미 한 번 read 되어 1차 캐시에 남아있는 객체를 또 read 했을 때는

데이터베이스 레벨에서 select 되는 것이 아닌 1차 캐시(애플리케이션 레벨)에서 반복 읽기를 수행함.

쓰기 지연

commit() 하는 시점까지 쿼리를 날리지 않는다.

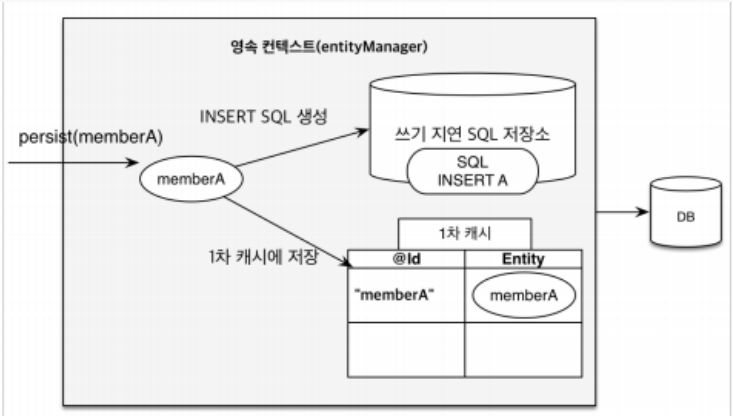
트랜잭션 중간에 롤백이 될 수도 있고

persist() 로 보관한 영속 객체가 중간에 수정/삭제 될 수 있기 때문에

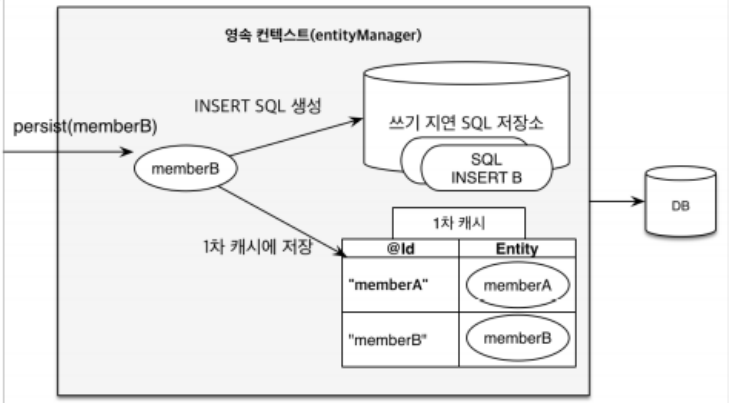
미루고 미루다 commit() 시점에 1차캐시에 남아있던 영속 객체들이 DB에 적용된다.

BatchQuery (여러 쿼리를 한꺼번에 날리기)가 적용된다.

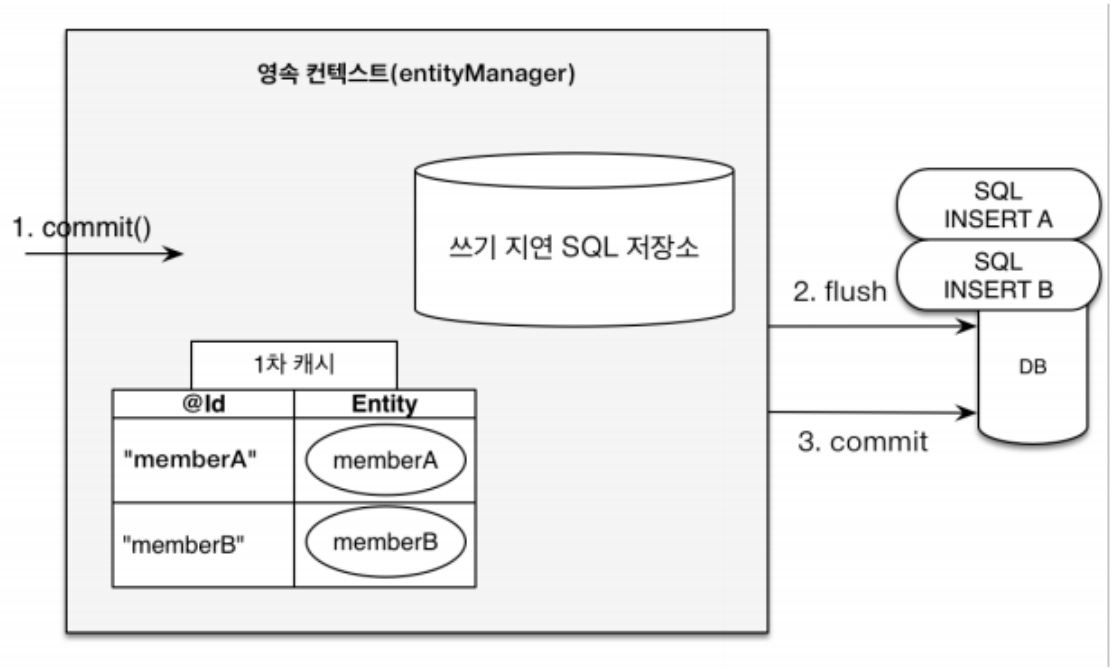
```
em.persist(memberA);
```



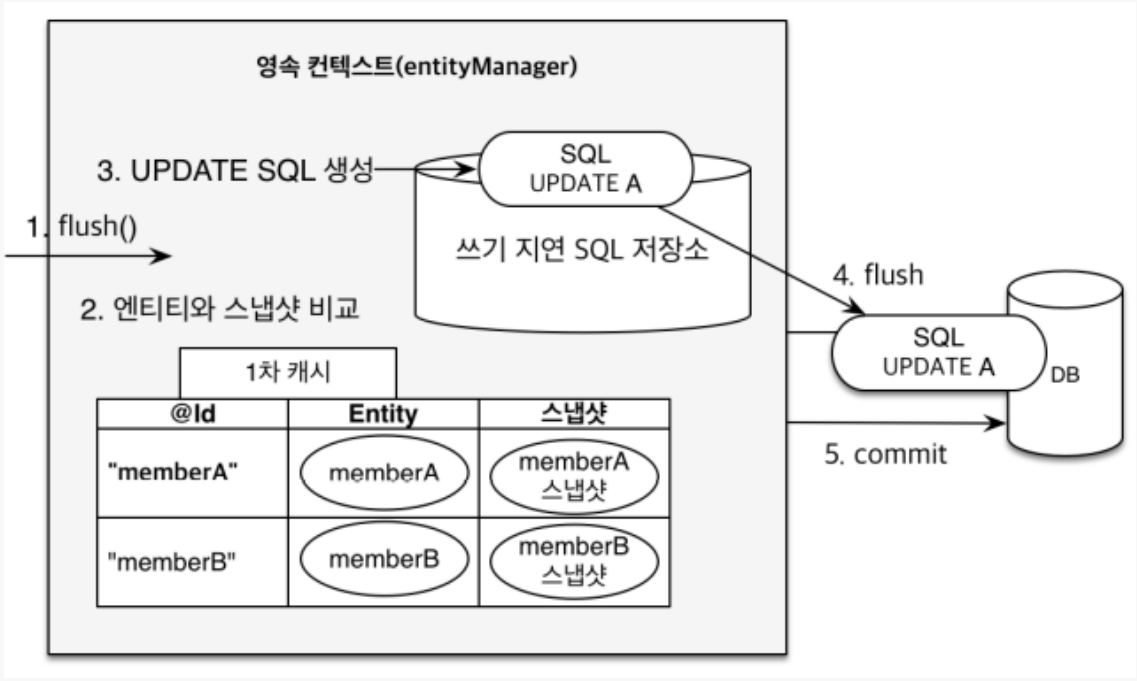

```
em.persist(memberB);
```



transaction.commit();



변경 감지



Flush

Flush 란

- 영속성 컨텍스트의 변경내용을 데이터베이스에 동기화
- 영속성 컨텍스트를 비우지 않음

언제 발생?

- 변경 감지
- 수정된 엔티티 쓰기 지연 SQL 저장소에 등록
- 쓰기 지연 SQL 저장소의 쿼리를 데이터베이스에 전송 (등록, 수정, 삭제 쿼리)

영속성 컨텍스트의 내용을 Flush 하려면

- entityManager.flush() - 직접 호출
- transaction.commit() - 플러시 자동 호출
- JPQL 쿼리 실행 - 플러시 자동 호출

```
em.persist(memberA);
em.persist(memberB);
em.persist(memberC);
//중간에 JPQL 실행
query = em.createQuery("select m from Member m", Member.class); // 이때 flush() 발생.
List<Member> members= query.getResultList();
// memberA, memberB, memberC가 전부 보일 수 있게
```