(10월 26일) 영속성 관리(비영속, 영속, 준영속, 삭제)

영속성이란?

어학사전 다른 어학정보 8

[국어사전]

영속성 (永續性)

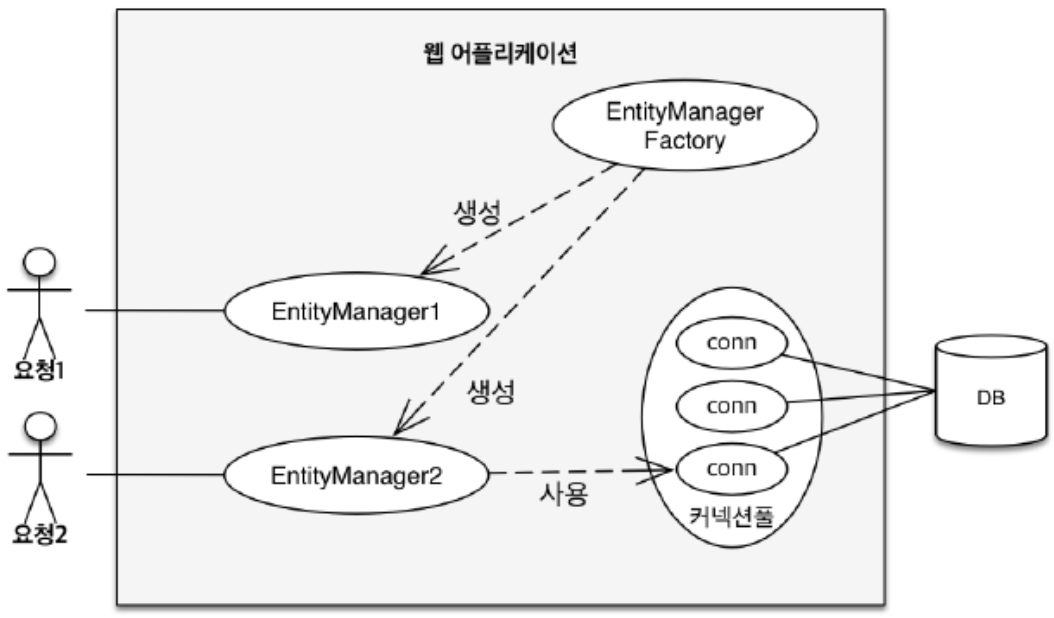
[영:속썽]

영원히 계속되는 성질이나 능력.

국어사전 결과 더보기

- 데이터를 생성한 프로그램의 실행이 종료 되어도 사라지지 않는 데이터의 특성
- 영구적으로 데이터가 저장된 상태
- JPA(Java Persistence API) ⇒ 자바 환경에서 데이터의 영속성을 관리하기 위한 API

EntityManagerFactory와 EntityManager



Interface EntityManager

interface in javax.persistence

```
public interface EntityManager
```

Interface used to interact with the persistence context.

An `EntityManager` instance is associated with a persistence context. A persistence context is a set of entity instances in which for any persistent entity identity there is a unique entity instance. Within the persistence context, the entity instances and their lifecycle are managed. The `EntityManager` API is used to create and remove persistent entity instances, to find entities by their primary key, and to query over entities.

The set of entities that can be managed by a given `EntityManager` instance is defined by a persistence unit. A persistence unit defines the set of all classes that are related or grouped by the application, and which must be colocated in their mapping to a single database.

EntityManager

- '영속성 컨텍스트'에 접근하기 위해 사용하는 인터페이스
- 커넥션풀을 이용하여 데이터베이스에 연결되어 트랜잭션을 수행하는 단위
- 동시성 문제가 발생하기 때문에 스레드끼리 공유되면 안됨

EntityManagerFactory

- 애플리케이션에서 DB에 접근할 수 있는 EntityManager를 생성하는 공장
- 공장을 생성하는 비용은 상당히 크기 때문에 하나의 DB를 사용하는 애플리케이션에서 1개만 생성

- Thread-Safe하기 때문에 여러 스레드가 접근해도 큰 문제가 생기지 않는다
- 비용적인 문제와 thread-safe한 특징으로 인해 싱글톤으로 생성된다

영속성 컨텍스트?

- JPA를 이해하기 위해 가장 중요한 개념
- '엔티티를 영구 저장하는 환경'이라는 뜻 으로 해석된다
- 실체가 존재하지 않는 논리적인 개념이며 EntityManager를 통해 접근하고 관리할 수 있다.
- 엔티티들과 그들의 생명주기가 영속성 컨텍스트 내에서 관리되기 때문에 매우 중요
- EntityManager가 생성될 때 마다 영속성 컨텍스트가 새롭게 생성 (J2SE 환경에서)

> 💡 영속성 컨텍스트는 영구적으로 저장되는(DB) 데이터(엔티티)의 집합체를 관리하는 단위라고 생각

9 Answers

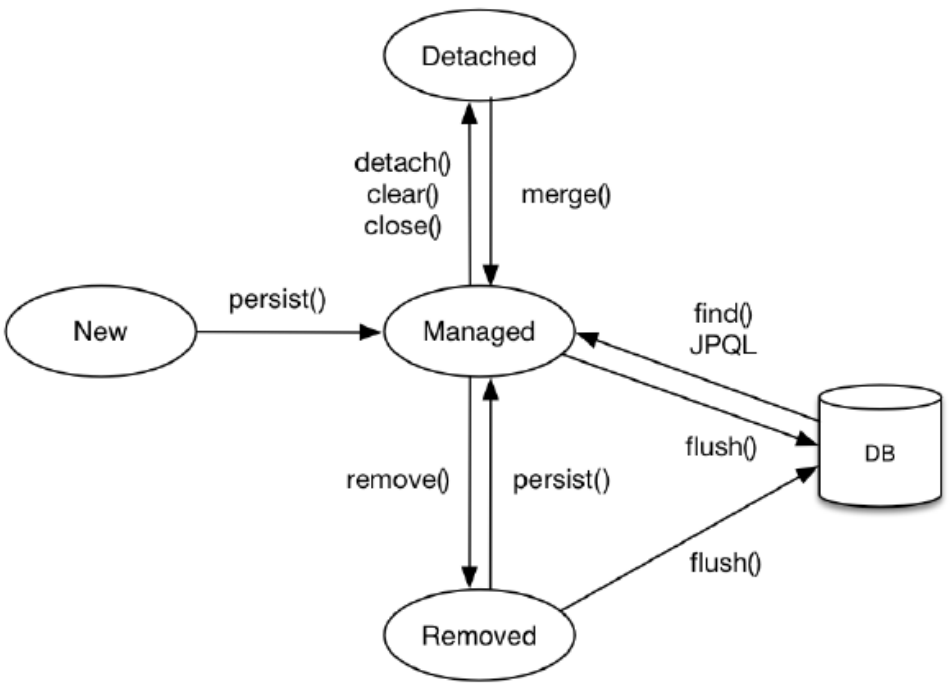
Active | Oldest | Votes

100

A persistence context handles a set of entities which hold data to be persisted in some persistence store (e.g. a database). In particular, the context is aware of the different states an entity can have (e.g. managed, detached) in relation to both the context and the underlying persistence store.

Although Hibernate-related (a JPA provider), I think these links are useful:

엔티티의 생명주기



비영속(new/transient)

- 영속성 컨텍스트와 관계가 없는 새롭게 생성된 상태
- 아래와 같이 EntityManager와 전혀 관계를 가지지 않은 상태

```
Member member = new Member();
member.setId("member1");
member.setUsername("회원1");
```

영속(managed)

- 영속성 컨텍스트에 저장되어 영속성 컨텍스트에 의해 관리되고 있는 상태

- EntityManager를 통해 persist() 메소드를 수행하면 해당 엔티티는 영속성 컨텍스트에 관리되는 상태
- em.find(), 혹은 JPQL(Java Persistence Query Language)를 통해 데이터베이스에서 조회된 엔티티 또한 같은 상태로 분류

```
EntityManagerFactory emf = Persistence.createEntityMangerFactory("my-persistence");
EntityManager em = emf.EntityManager();

em.persist(member);
```

준영속(detached)

- 영속성 컨텍스트에 저장되었다가 분리된 상태
- detach() 메소드를 호출하거나, close()를 통해 해당 영속성 컨텍스트를 닫을 때, 그리고 clear() 메소드를 통해 영속성 컨텍스트를 초기화해도 관리 중이던 엔티티들이 모두 준영속 상태가 된다.

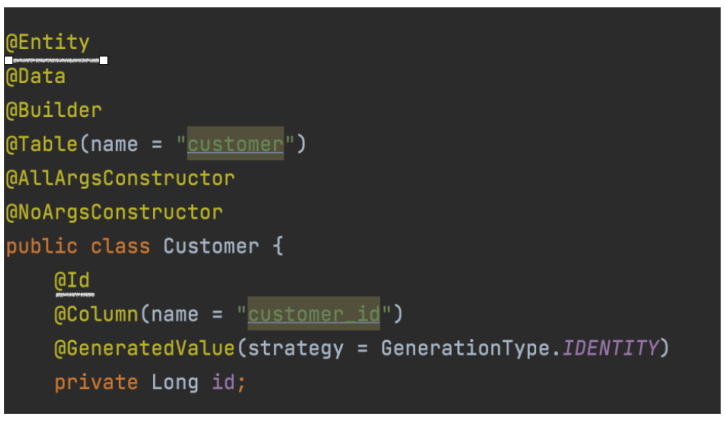
```
em.detach(member);
```

삭제(removed)

- 영속성 컨텍스트와 데이터베이스에서 삭제되는 상태

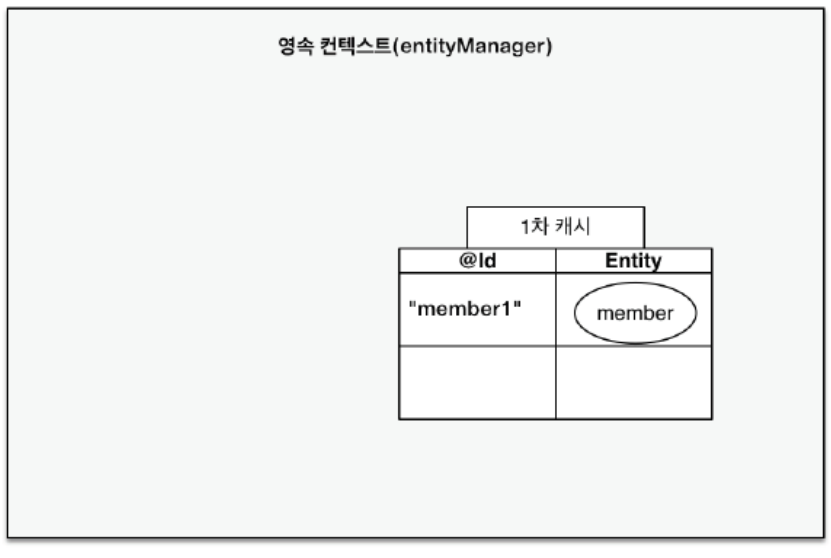
영속성 컨텍스트의 특징

- 영속성 컨텍스트가 관리하는 엔티티는 식별자 값이 있어야 한다.



- flush() 메소드가 호출될 때 영속성 컨텍스트 내 데이터들이 데이터 베이스에 반영
  - 트랜잭션이 커밋되는 순간 (트랜잭션의 commit() 메소드가 수행될 때 자동으로 flush() 호출)

영속성 컨텍스트의 동작 원리



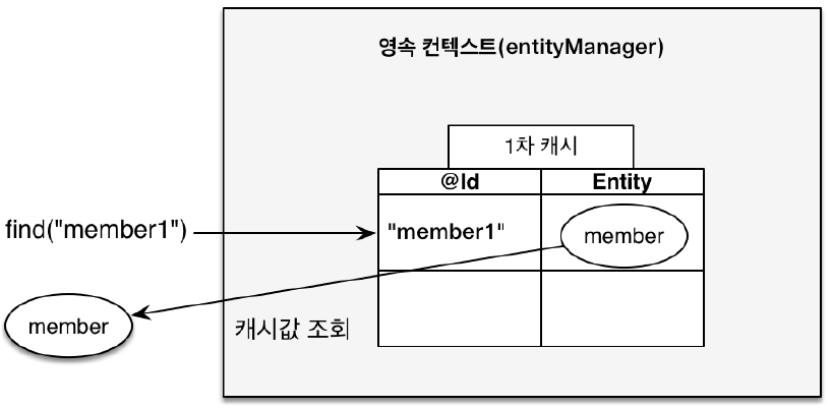
조회

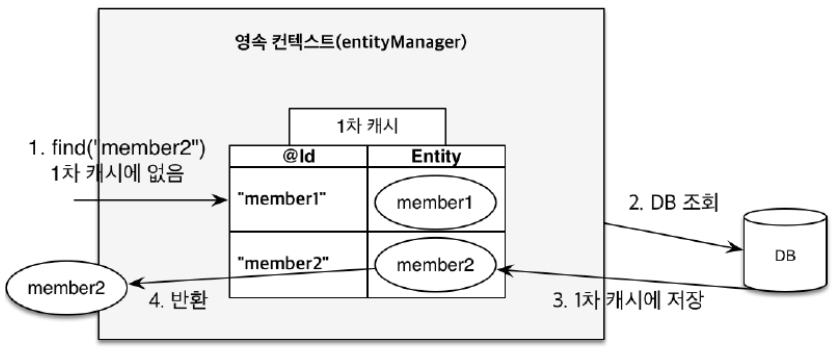
1차 캐시

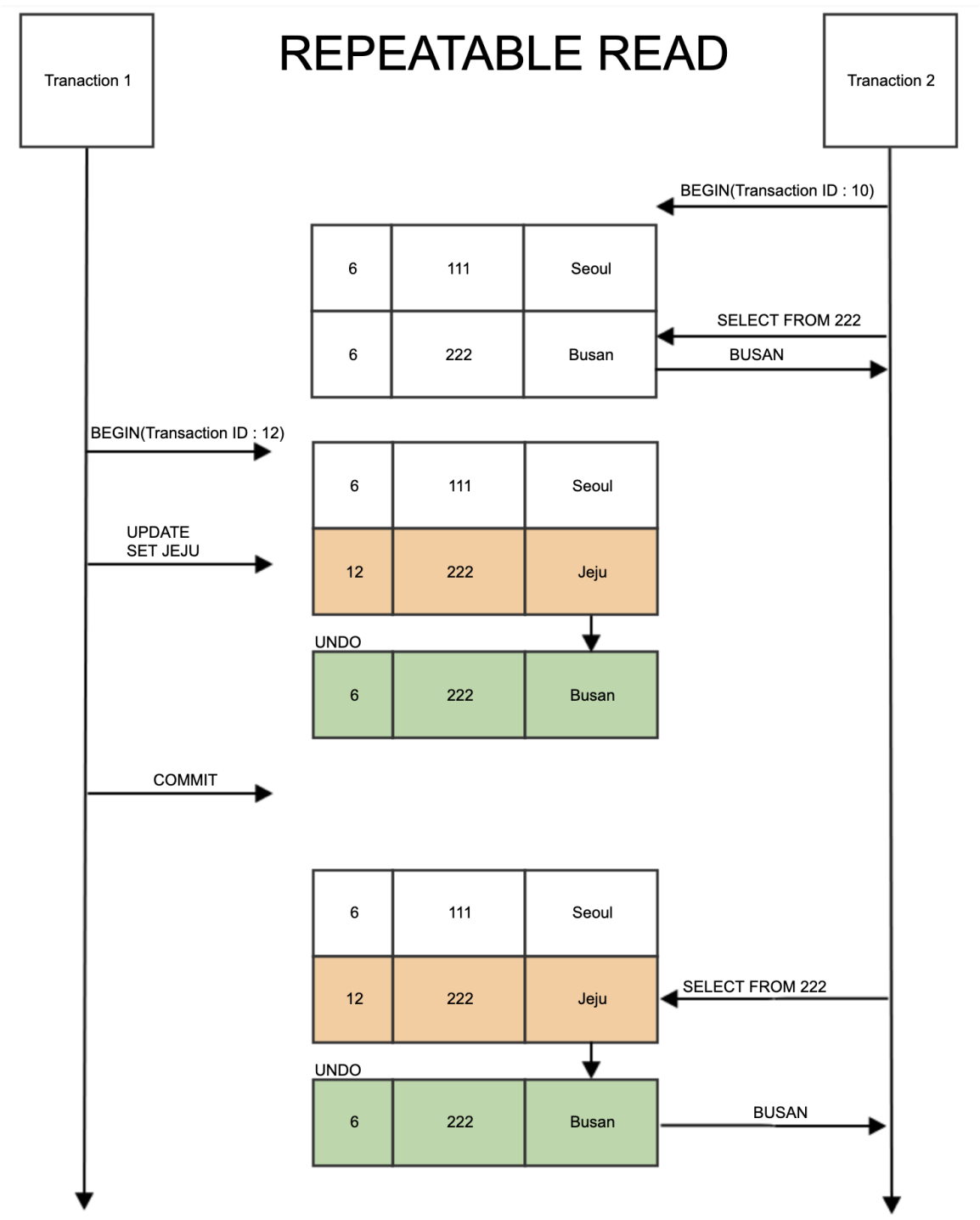
- JPA의 영속성 컨텍스트는 내부에 캐시를 가지고 있는데 이를 1차 캐시라고 한다.
- 영속 상태에 해당되는 엔티티들은 모두 1차 캐시에 저장되고, @Id(식별자)와 엔티티(객체)가 매핑되어 있는 구조
- 애플리케이션에서 조회를 수행할 땐 @Id를 통해 find 메소드를 수행하고 DB에 접근하기 전 1차 캐시에 매핑된 엔티티가 있는지 확인한다.
    - 매핑된 값이 있을 경우 1차 캐시값을 조회
    - 매핑된 값이 없다면, DB에 접근하여 조회를 수행한 후 1차 캐시에 해당 값을 저장한 후 값을 반환한다.

⇒ 조회 작업이 많을 경우 DB에 접근할 필요 없이 캐시값을 사용함으로써 성능상 이점이 생긴다.

⇒ 1차 캐시로 DB 단위가 아닌 애플리케이션 단위에서 REPEATABLE READ 격리 수준을 구현할 수 있다!!!!







동일성 보장

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

```
System.out.println(a == b); // 1
System.out.println(a.equals(b)); // 2
```

- 1번은 참일까?
- 2번은 참일까?

⇒ 동일성과 동등성

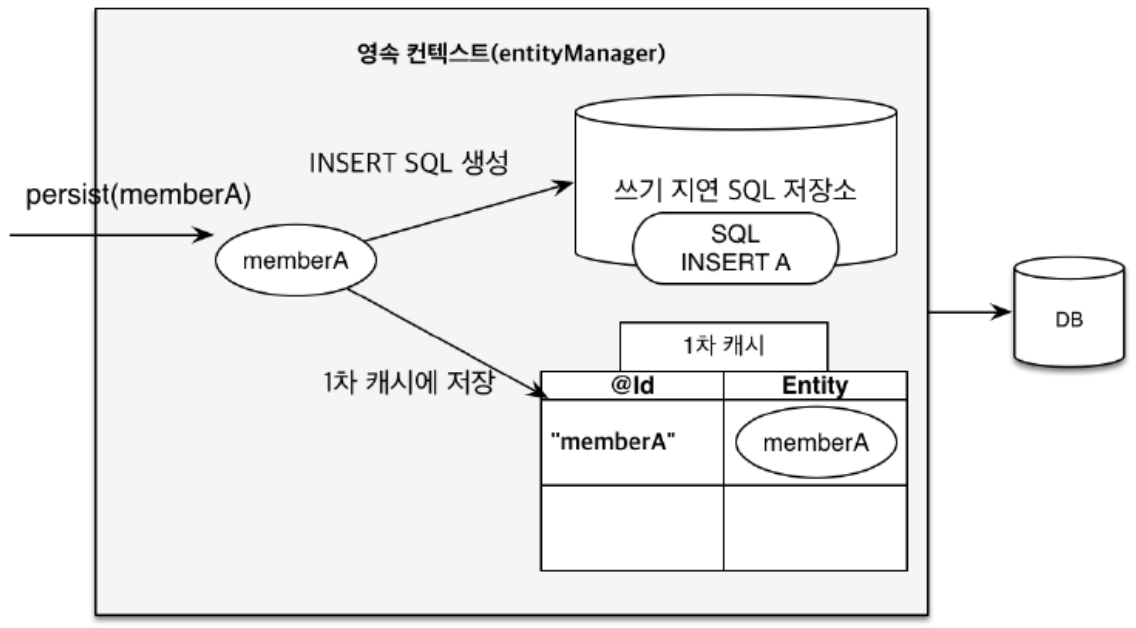
> 💡 ApplicationContext에 등록된 Bean 객체 또한 호출하면 위치값까지 같은 동일한 객체인 것과 마찬가지

쓰기

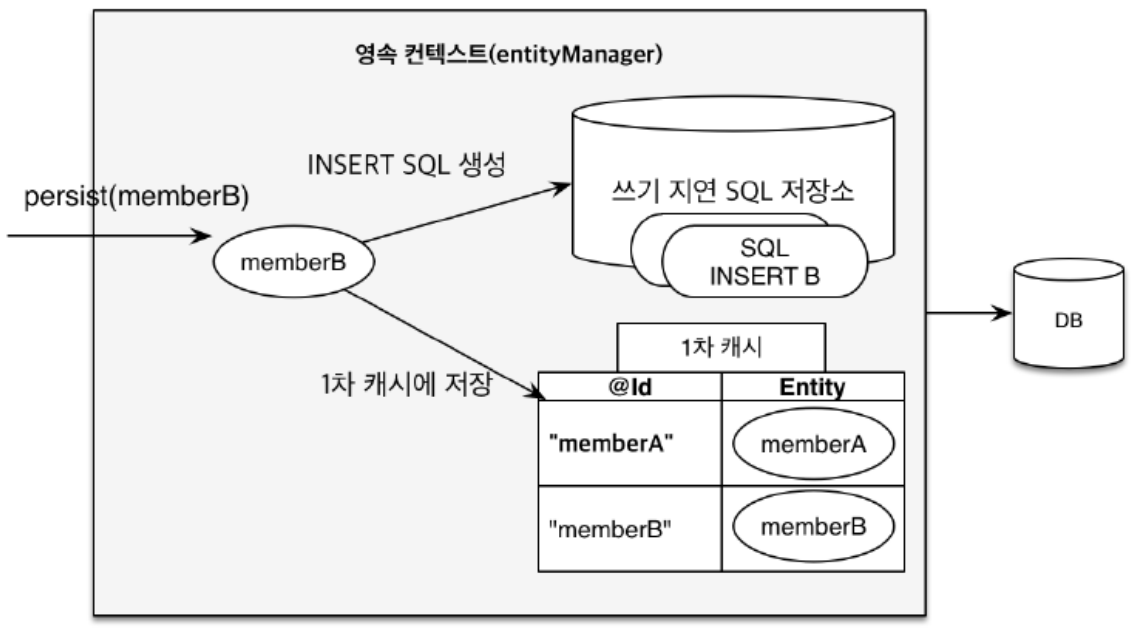
쓰기 지연

```
em.persist(memberA); // 1
em.persist(memberB); // 2
em.flush(); // 3
```

em.persist(memberA);



em.persist(memberB);



- memberA를 1차 캐시에 저장하고 INSERT SQL을 생성하여 쓰기 지연 SQL 저장소에 저장
- memberB를 1차 캐시에 저장하고 INSERT SQL을 생성하여 쓰기 지연 SQL 저장소에 저장
- flush() 메소드가 호출되는 시점에 SQL 저장소에 있던 쿼리들이 flush되며 DB에 저장
  - Transaction commit 시점과 일치

WHY 쓰기 지연?

- Transaction이 commit 되지 않으면 DB에 반영되지 않음
- 만약 persist할 때마다 쿼리를 날려도 만약 어떤 오류가 생겨 rollback이 되는 경우
- 결국, Transaction이 commit이 되는 시점에 insert 쿼리를 한번에 보내주면 작업을 효율적으로 할 수 있음

수정

데이터를 쓰는 것에 대한 것은 이해했는데, 근데 어떻게 값을 수정하나?

- 영속성 컨텍스트를 관리하는 EntityManager를 이용하여 변경된 update 메소드를 수행하면 되지 않을까?
- EntityManager엔 update 메소드가 없다!
- JPA는 **dirty checking(변경 감지)**라는 기능을 이용하여 데이터베이스에 값을 갱신

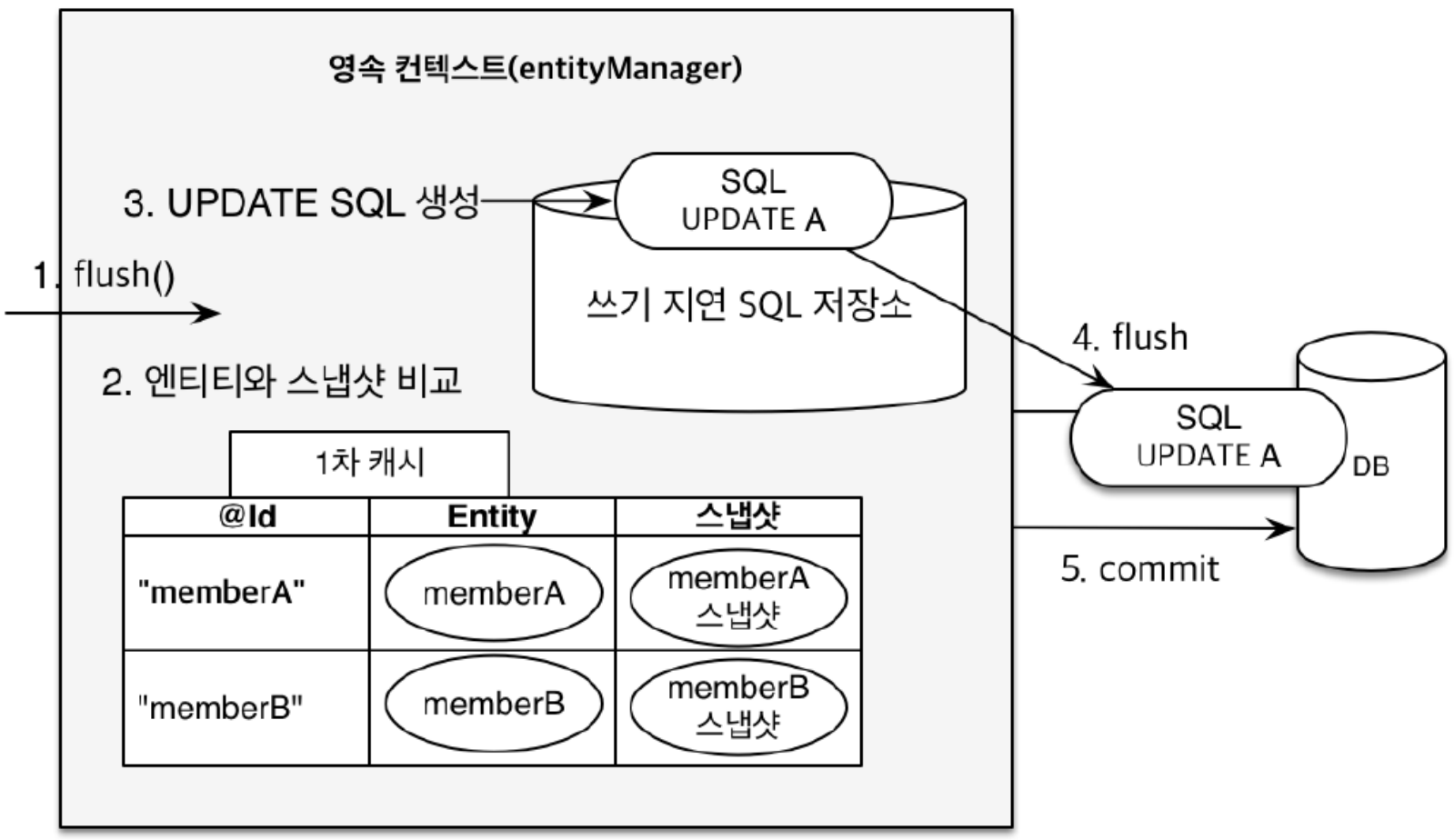
```
EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "my-persistence");

EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();

em.update();
```

변경 감지(dirty checking)

- 엔티티를 영속성 컨텍스트에 보관할 때, **최초 상태를 복사한 후 저장** ⇒ 스냅샷
- 영속성 컨텍스트로의 flush() 시점에 스냅샷(트랜잭션 시작 시점의 db 상태)와 영속성 컨텍스트의 엔티티를 비교하여 변경된 엔티티를 찾아낸다
- 변경된 엔티티를 바탕으로 UPDATE SQL을 자동으로 생성해주고 쓰기 지연 SQL 저장소에 축적한다
- 트랜잭션이 commit(데이터 베이스로의 flush()) 되면 쌓인 SQL을 commit하여 데이터베이스에 반영한다



- JPA의 default는 엔티티의 모든 필드를 업데이트
  - 30개 이상의 컬럼이 있을 경우, 동적 수정쿼리를 사용하는 것이 더 빠를 수 있다!

Update 쿼리의 default가 모든 필드인 이유?

⇒ 수정 쿼리를 여러개 만들지 않고 하나만 만들어서 재사용할 수 있다

⇒ 동일한 쿼리를 보낸다면 이전에 DB에서 파싱한 쿼리를 재사용할 수 있다

삭제

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

- 엔티티 삭제를 위해 대상 엔티티를 먼저 조회
- remove()메소드에 엔티티를 넘기면 삭제 쿼리를 쓰기 지연 SQL 저장소에 등록
- 트랜잭션 commit이 될 때 쓰기 지연 SQL 저장소의 쿼리들이 데이터베이스에 전달
    - 영속성 컨텍스트 내에서도 해당 엔티티는 삭제

준영속 상태

detach

- 특정 엔티티를 준영속 상태로 만들 때 사용

detach(엔티티)가 수행 되면 어떤 일이 일어나나?

1. 1차 캐시에서 대상 엔티티가 제거
2. 쓰기 지연 SQL 저장소에서 해당 엔티티와 관련된 모든 SQL들이 제거

⇒ 더 이상 영속성 컨텍스트가 관리하지 않는 엔티티가 되었기 때문에 준영속 상태가 됨

clear

- detach가 특정 엔티티를 준영속 상태를 만드는 것이라면, clear는 영속성 컨텍스트를 초기화 하는 것
- 영속성 컨텍스트를 제거하고 새로 만든 것과 동일

close

- 영속성 컨텍스트를 종료
- 종료된 영속성 컨텍스트가 관리하던 엔티티들은 모두 준영속 상태

merge

- 준영속 상태의 엔티티를 다시 영속상태로 만들 기 위해 사용

> 💡 준영속 상태의 엔티티는 영속성 컨텍스트에 관리되지 않기 때문에 1차 캐시, 쓰기 지연, 변경 감지 등 영속성 컨텍스트의 특징을 하나도 이용하지 못한다.

merge()메소드가 수행 되면 어떤 일이 일어나나?

1. 준영속 엔티티가 merge 됐을 때 새로운 영속성 컨텍스트 내 1차 캐시에 해당 엔티티가 있는지 찾는다
    a. 없을 경우 DB를 조회하여 1차 캐시에 엔티티를 저장
2. 영속 상태의 엔티티에 merge 메소드의 파라미터로 보내진 준영속 엔티티를 밀어넣는다 (병합)
3. 병합 작업이 끝난 엔티티를 반환한다.

영속성 컨텍스트를 애플리케이션 수행 중에 종료하지 않으면 준영속 상태와 관리에 대해 몰라도 되지 않을까?

- 순수 J2SE 환경에선 개발자가 직접 EntityManager를 생성하고 관리
- 스프링 애플리케이션과 같은 컨테이너 환경에서의 작업이 아니기 때문에 한정적인 예시
- 스프링 애플리케이션의 경우 컨테이너가 제공하는 전략이 별도로 존재
    - @Transactional 애노테이션이 붙어있는 layer에서 영속성 컨텍스트 생성
    - 대부분 웹 애플리케이션을 설계할 때 비즈니스 로직이 수행되어야 하는 Service layer부터 Transactional 애노테이션이 적용
    - Controller, view layer부턴 엔티티들이 모두 준영속 상태로 존재
        - 이로 인해, 지연 로딩 오류, 변경 감지 불가 등의 문제가 생길 수 있음

💡 웹 서비스 계층의 분리로 인해 계층 간 엔티티 상태 변화는 흔한 오류로 작용될 수 있다. 따라서, 애플리케이션 내에서 엔티티의 영속 상태의 변화에 대해 파악하고 있는 것 중요!

참고자료

자바 ORM 표준 JPA 프로그래밍 - 교보문고

스프링 데이터 예제 프로젝트로 배우는 전자정부 표준 데이터베이스 프레임 | ★ 이 책에서 다루는 내용 ★■ JPA 기초 이론과 핵심 원리■ JPA로 도메인 모델을 설계하는 과정을 예제 중심으로 설명■ 다양한 객체 지향 쿼리 언어 설명■ JPA와 스프링 프레임워크를 함께 사용해서 웹 애플리케이션을 개발하는 방법■ 스프링 데이터

http://www.kyobobook.co.kr/product/detailViewKor.laf?mallGb=KOR&ejkGb=KOR&barcode=9788960777330

[JPA] 영속성 관리

JPA 사용하기 위해선 먼저 엔티티 매니저 팩토리와 엔티티 매니저에 대해서 이해해야한다. 웹 애플리케이션이 실행될 때 엔티티 매니저가 같이 생성되고 (싱글톤?), 클라이언트의 요청이 올 때마다 엔티티 매니저 팩토리를 통해서 엔티티 매니저를 생성한다. 생성된 엔티티 매너저는 내부적으로 DB 커넥션을 사용해서 DB 접근한다.

https://velog.io/@syleemk/JPA-%EC%98%81%EC%86%8D%EC%84%B1-%EA%B4%80%EB%A6%AC

JPA, ORM 그리고 영속성

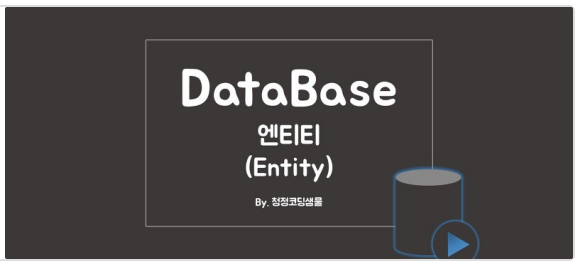
jpa 와 orm에 대해 맛보기로 알아보자. Java Persistence API 자바진영의 ORM 기술 표준이다. Hibernate, EclipesLink, dataNucleus Object-Relational Mapping ORM이란 객체와 DB의 테이블이 매핑을 이루는 것을 말합니다. 객체를 데이터베이스에 저장할 때 INSERT SQL 을 직접 작성하는게 아니라 객체를 마치 자바 컬렉션

https://jar100.tistory.com/3

[DataBase]엔티티[Entity]란 무엇일까?

1. 엔티티(Entity)란 무엇인가? ◎ 데이터의 집합을 의미한다. ◎ 저장되고, 관리되어야하는 데이터이다. ◎ 개념, 장소, 사건 등을 가리킨다. ◎ 유형 또는 무형의 대상을 가리킨다. 2. 엔티티의 특징 ◎ ..

https://rh-cp.tistory.com/78



EntityManager (Java(TM) EE 7 Specification APIs)

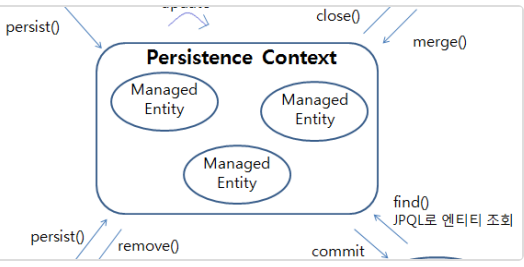
An EntityManager instance is associated with a persistence context. A persistence context is a set of entity instances in which for any persistent entity identity there is a unique entity instance. Within the persistence context, the entity instances and their lifecycle are managed.

https://docs.oracle.com/javaee/7/api/javax/persistence/EntityManager.html

영속성(Persistence)이란?

업무를 하다보면 동료가 영속성이란 말을 하는걸 들을 수 있습니다.혹은 IT서적을 보다보면 영속성(Persistence)라는 단어가 종종 나오는것을 볼 수 있는데요. 과연 영속성(Persistence)이라는 것이 무엇인지 알아보겠습니다.알고보면 별거 아닙니다. 아주 쉽습니다. 일단 아래 사전적 의미부터 보겠습니다. 음... 뭔가 감이

https://sugerent.tistory.com/587



What is Persistence Context?

Thanks for contributing an answer to Stack Overflow! Please be sure to answer the question. Provide details and share your research! Asking for help, clarification, or responding to other answers. Making statements based on opinion; back them up with references or personal experience. To learn more, see

https://stackoverflow.com/questions/19930152/what-is-persistence-context