3주차 키워드 정리

- 키워드과제
  - JOIN 종류 조사
  - DB 제 1 ~ 3 정규화에 대해 알아보기.
  - @Column, @Table 의 역할과 옵션 알아보기.
  - 양방향 연관관계에 대해서 공부.
  - CascadeType 에 대해서 공부

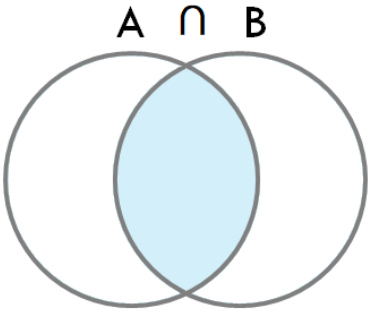
JOIN 종류 조사

조인은 두 개의 테이블을 서로 묶어서 하나의 결과를 만들어 내는 것을 말한다.

- **INNER JOIN(내부 조인)**은 두 테이블을 조인할 때, 두 테이블에 모두 지정한 열의 데이터가 있어야 한다.
- **OUTER JOIN(외부 조인)**은 두 테이블을 조인할 때, 1개의 테이블에만 데이터가 있어도 결과가 나온다.
- **CROSS JOIN(상호 조인)**은 한쪽 테이블의 모든 행과 다른 쪽 테이블의 모든 행을 조인하는 기능이다.
- **SELF JOIN(자체 조인)**은 자신이 자신과 조인한다는 의미로, 1개의 테이블을 사용한다.

1. Inner join

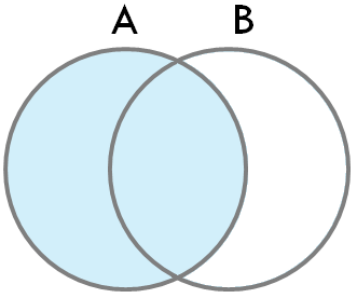
- 교집합(A ∩ B) 연산과 같다.
- 조인 키 컬럼 값이 양쪽 테이블 데이터 집합에서 공통적으로 존재하는 데이터만 조인해서 결과 데이터 집합으로 추출하게 된다.



- **Natural Join**
  - 두 테이블에서 동일한 컬럼명을 갖는 컬럼은 모두 조인이 된다.
  - 반드시 두 테이블 간의 동일한 이름, 타입을 가진 컬럼이 필요하다.
  - 기준 테이블과 조인 테이블 모두에 데이터가 존재해야 조회된다.
  - Inner Join에서 조건문 추가하여 같은 결과값을 얻을 수 있다.

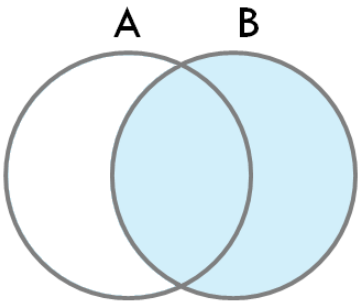
2. Left outer join

- 교집합 연산 결과와 차집합 연산 결과를 합친 것((A ∩ B) ∪ (A - B))과 같다.
- 조인 키 컬럼 값이 양쪽 테이블 데이터 집합에서 공통적으로 존재하는 데이터와 Left outer join 키워드 왼쪽에 명시된 테이블에만 존재하는 데이터를 결과 데이터 집합으로 추출하게 된다.



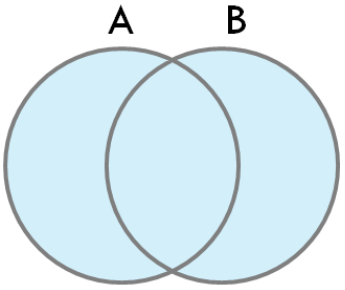
3. Right outer join

- 교집합 연산 결과와 차집합 연산 결과를 합친 것((A ∩ B) ∪ (A - B))과 같다.
- 차집합의 집합 기준이 Left outer join과 반대이다.
- 조인 키 컬럼 값이 양쪽 테이블 데이터 집합에서 공통적으로 존재하는 데이터와 Right outer join 키워드 오른쪽에 명시된 테이블에만 존재하는 데이터를 결과 데이터 집합으로 추출하게 된다.



4. Full outer join

- 합집합 연산 결과와 같다.
- 조인 키 컬럼 값이 양쪽 테이블 데이터 집합에서 공통적으로 존재하는 데이터와 한쪽 테이블에만 존재하는 데이터도 모두 결과 데이터 집합으로 추출하게 된다.
- MySQL에서는 FULL OUTER JOIN을 지원하지 않으므로 LEFT OUTER JOIN 결과와 RIGHT OUTER JOIN결과를 UNION 하여 사용해야 한다.



5. **Cross Join**

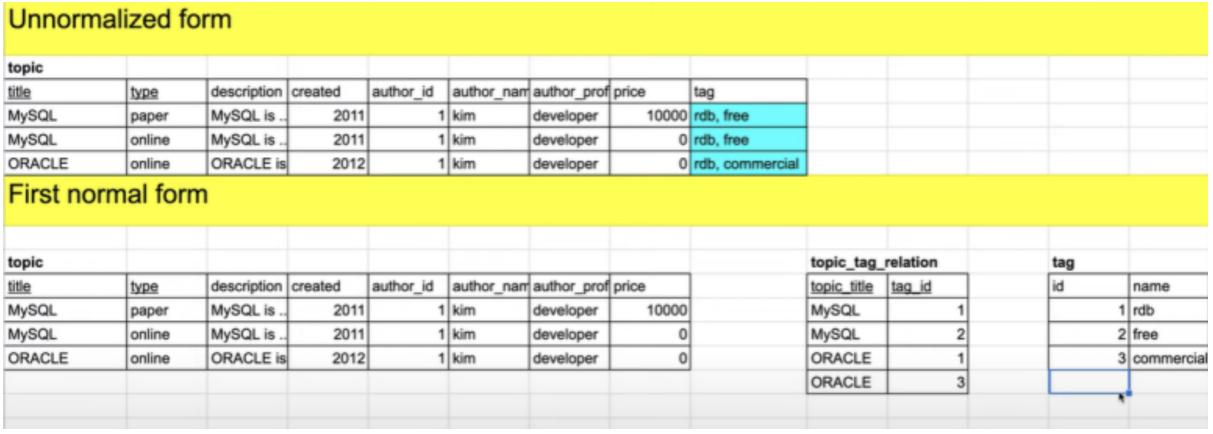
- 곱집합 연산 결과와 같다.
- 두 테이블 데이터의 모든 조합을 추출한다.
- 테이블1의row * 테이블2의row 개수만큼의 row를 가진 테이블이 생성된다.

DB 제 1 ~ 3 정규화

제 1 정규화

- **Atomic columns**라는 조건에 만족하게끔 비정규형 테이블을 제1 정규형 테이블로 구조화하는 것을 말한다.

- **같은 성격과 내용의 컬럼이 연속적으로 나타나는 컬럼**이 존재할 때, 해당 컬럼을 제거하고 기본테이블의 PK를 추가해 새로운 테이블을 생성하고, 기존의 테이블과 **1:N** 관계를 형성하는 것



Unnormalized form

topic

title	type	description	created	author_id	author_nam	author_prof	price	tag
MySQL	paper	MySQL is ..	2011	1	kim	developer	10000	rdb, free
MySQL	online	MySQL is ..	2011	1	kim	developer	0	rdb, free
ORACLE	online	ORACLE is	2012	1	kim	developer	0	rdb, commercial

First normal form

topic

title	type	description	created	author_id	author_nam	author_prof	price
MySQL	paper	MySQL is ..	2011	1	kim	developer	10000
MySQL	online	MySQL is ..	2011	1	kim	developer	0
ORACLE	online	ORACLE is	2012	1	kim	developer	0

topic_tag_relation

topic_title	tag_id
MySQL	1
MySQL	2
ORACLE	1
ORACLE	3

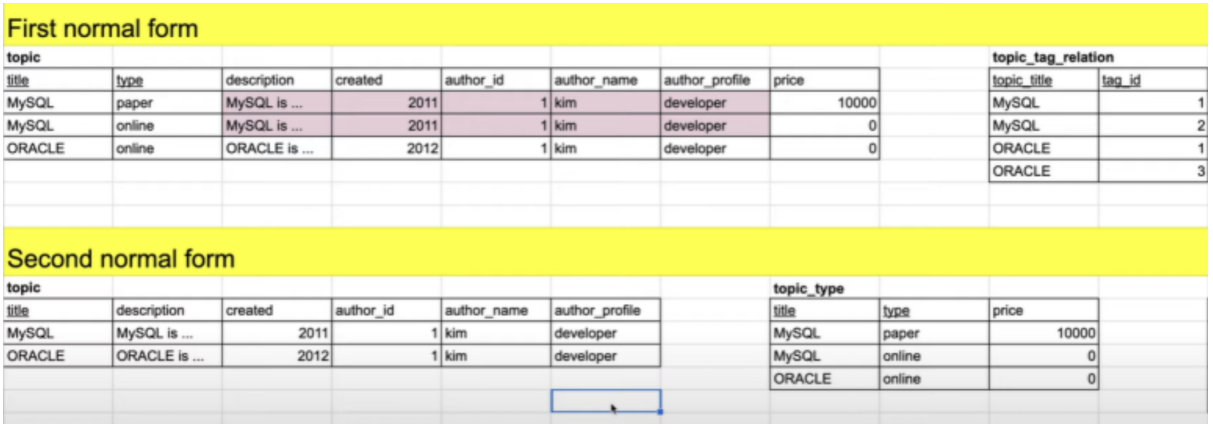
tag

id	name
1	rdb
2	free
3	commercial

tag필드에 있는 여러 개의 값들이 문제

제 2 정규화

- **No partial dependencies**라는 조건에 만족하게끔 제1 정규형 테이블을 제2 정규형 테이블로 정규화하는 것을 말한다.
- No partial dependencies는 부분 종속성이 없어야 한다는 말이다. 이것은 다시 말해 표에 **중복키인 기본키가 없어야 한다는 의미**이다.
- PK가 여러 키로 구성된 **복합키(Composite Primary Key)**로 구성된 경우가 **2차 정규화의 대상**이 되며, 복합키 전체에 의존하지 않고 복합키의 일부분에만 종속되는 속성들이 존재할 경우 이를 **분리**하는 것이다.



First normal form

topic

title	type	description	created	author_id	author_name	author_profile	price
MySQL	paper	MySQL is ...	2011	1	kim	developer	10000
MySQL	online	MySQL is ...	2011	1	kim	developer	0
ORACLE	online	ORACLE is ...	2012	1	kim	developer	0

topic_tag_relation

topic_title	tag_id
MySQL	1
MySQL	2
ORACLE	1
ORACLE	3

Second normal form

topic

title	description	created	author_id	author_name	author_profile
MySQL	MySQL is ...	2011	1	kim	developer
ORACLE	ORACLE is ...	2012	1	kim	developer

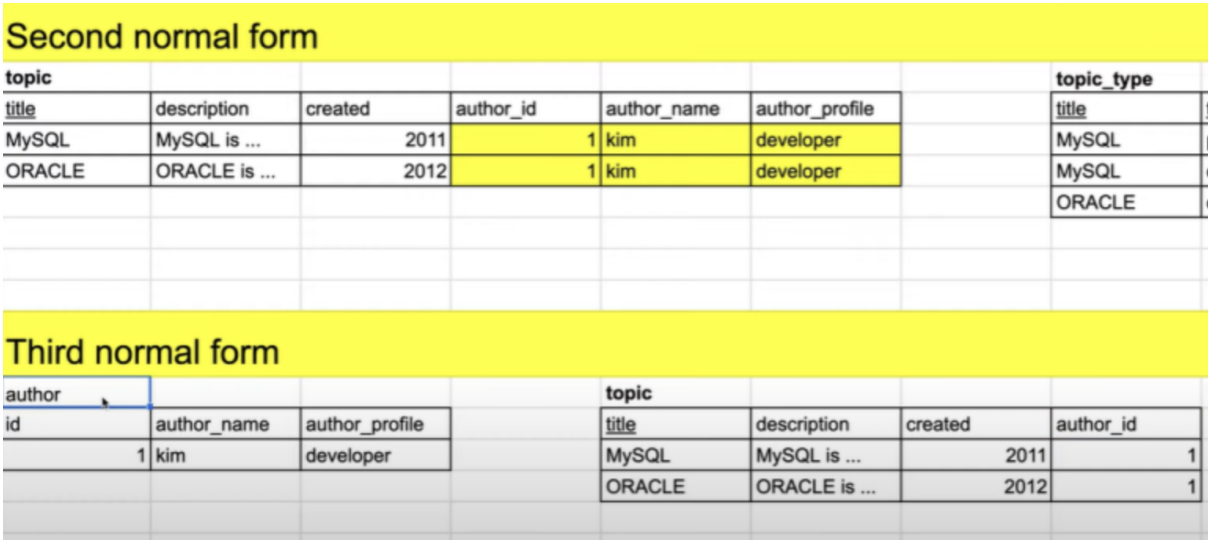
topic_type

title	type	price
MySQL	paper	10000
MySQL	online	0
ORACLE	online	0

중복되는 레코드 (type와 price 필드 값이 각기 다르기 때문)

제 3 정규화

- **No transitive dependencies**라는 조건에 만족하게끔 제2 정규형 테이블을 제3 정규형 테이블로 정규화하는 것을 말한다.
- No transitive dependencies는 **이행적 종속성**이 없어야 한다는 말이다.
- 테이블의 키가 아닌 컬럼들은 기본키에 의존해야 하는데 겉으로는 그런 것처럼 보이지만 **실제로는 기본키가 아닌 다른 일반 컬럼에 의존하는 컬럼**들이 있을 수 있다. 제 3정규화는 **PK에 의존하지 않고 일반컬럼에 의존하는 컬럼**들을 분리한다.



author_id에 의존하는 컬럼들

@Column, @Table의 역할과 옵션

@Column

- JPA에서 DB Table의 Column을 Mapping
- `@Column`을 사용하지 않으면 nickName과 같이 속성명 그대로 DB Column과 Mapping을 시도한다.

```
@Column(name="NICK_NAME")
    private String nickName;
```

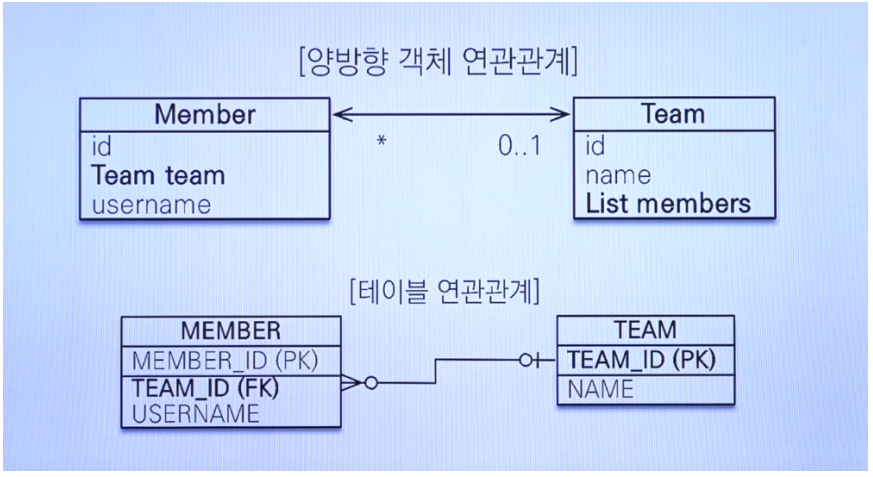
- 속성값
  1. name : 맵핑할 테이블의 컬럼 이름을 지정합니다;
  2. insertable : 엔티티 저장시 선언된 필드도 같이 저장합니다.
  3. updateable : 엔티티 수정시 이 필드를 함께 수정합니다.

4. table : 지정한 필드를 다른 테이블에 맵핑할 수 있도록 합니다.
5. nullable : NULL을 허용할지, 허용하지 않을지 결정합니다.
6. unique : 제약조건을 걸 때 사용합니다.
7. columnDefinition : DB 컬럼 정보를 직접적으로 지정할 때 사용합니다.
8. length : varchar의 길이를 조정합니다. 기본값으로 255가 입력됩니다.
9. precsion, scale : BigInteger, BigDecimal 타입에서 사용합니다. 각각 소수점 포함 자리수, 소수의 자리수를 의미합니다.

@Table

- 엔티티와 Mapping할 테이블을 지정
- 속성값
  1. name : 매핑할 테이블의 이름을 지정합니다.
  2. catalog : DB의 catalog를 맵핑합니다.
  3. schema : DB 스키마와 맵핑합니다.
  4. uniqueConstraint : DDL 쿼리를 작성할 때 제약 조건을 생성합니다.

양방향 연관관계



- 객체 설계는 위와 같이 Member에서는 Team을 가지고 있고, Team에서는 Members를 가지고 있도록 설계하면 된다.
- DB는 단방향 매핑때와 바뀌는게 없고 둘을 join 하면 된다. DB는 방향이 없다!

- Member 엔티티는 단방향과 동일

```
@Entity
public class Member {
 @Id
 @GeneratedValue
 private Long id;

 @Column(name = "USERNAME")
 private String name;

 private int age;

 @ManyToOne
 @JoinColumn(name = "TEAM_ID")
 private Team team;
...
}
```

- Team 엔티티는 컬렉션을 추가해주면 된다.
  - 팀의 입장에서 바라보는 일대다, @OneToMany 어노테이션을 설정
  - mappedBy로 team과 연관
  - 컬렉션을 매핑 (관례로 ArrayList로 초기화, NPE 방지)

```
@Entity
public class Team {
 @Id
 @GeneratedValue
 private Long id;

 private String name;

 @OneToMany(mappedBy = "team")
 private List<Member> members = new ArrayList<Member>();
...
}
```

- 객체 그래프 탐색

```
//팀 조회
Team findTeam = em.find(Team.class, team.getId());
```

```
// 역방향으로 멤버들 조회
int memberSize = findTeam.getMembers().size();
```

CascadeType

JPA에서는 영속성 전이를 Cascade 옵션을 통해서 설정하고 관리할 수 있다.

(부모 엔티티를 다룰 경우, 자식 엔티티까지 다룰 수 있다는 뜻)

JPA Cascade Type

- ALL
  - 상위 엔터티에서 하위 엔터티로 모든 작업을 전파
- PERSIST
  - 하위 엔티티까지 영속성 전달
- MERGE
  - 하위 엔티티까지 병합 작업을 지속
- REMOVE
  - 하위 엔티티까지 제거 작업을 지속
- REFRESH
  - 데이터베이스로부터 인스턴스 값을 다시 읽어 오기(새로고침)
- DETACH
  - 영속성 컨텍스트에서 엔티티 제거