**데이터베이스(DataBase)**

특징

* 실시간 접근성
* 지속적인 변화
* 동시 공유
* 내용에 대한 참조

데이터베이스 장점

1. 데이터 중복 최소화
2. 데이터 공유
3. 일관성, 무결성, 보안성 유지
4. 최신의 데이터 유지
5. 데이터의 표준화 가능
6. 데이터의 논리적, 물리적 독립성
7. 용이한 데이터 접근
8. 데이터 저장 공간 절약

데이터베이스 단점

1. 데이터베이스 전문가 필요
2. 많은 비용 부담
3. 데이터 백업과 복구가 어려움
4. 시스템의 복잡함
5. 대용량 디스크로 액세스가 집중되면 과부하 발생

SQL

관계 대수 : 절차적 언어 select, join ,,,,

관계 논리(해석) : 비절차적 언어

**정규화**

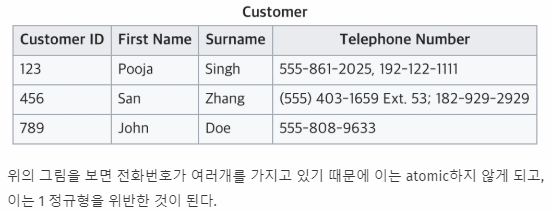
정규화 : 데이터베이스 정규화란 데이터의 중복을 줄이고 무결성을 향상 시키는 등 여러 목적을 달성하기 위해 관계형 데이터베이스를 정규화된 형태로 재디자인하는 것을 말한다.

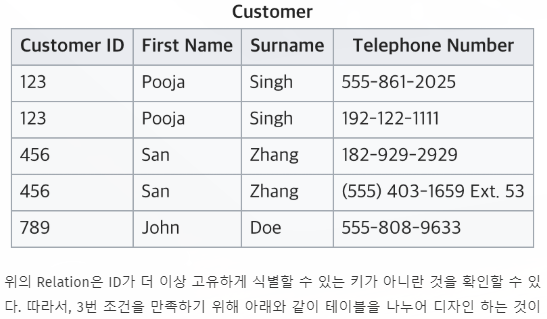
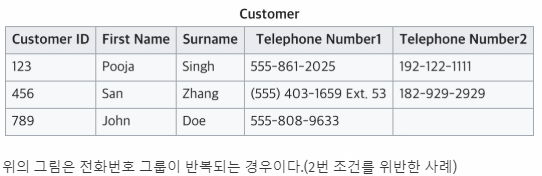
목적

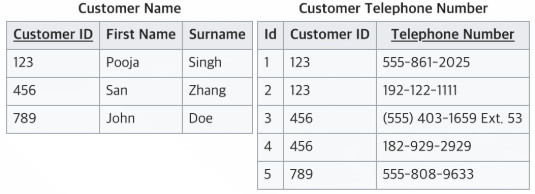
* 불필요한 데이터를 제거, 데이터의 중복을 최소화 하기 위해서
* 데이터베이스 구조 확장 시 재디자인을 최소화
* 다양한 관점에서의 query를 지원하기 위해서
* 무결성 제약조건의 시행을 간단하게 하기 위해서
* 각종 [이상 현상(Anomaly)](https://wkdtjsgur100.github.io/anomaly) 을 방지하기 위해서, 테이블의 구성을 논리적이고 직관적으로 한다.

**제 1정규화(First Normal Form, 1NF) - 도메인이 원자값**

1. 어떤 Relation에 속한 모든 Domain이 원자값(atomic value)만으로 되어 있다.
2. 모든 attribute에 반복되는 그룹(repeating group)이 나타나지 않는다.
3. 기본 키를 사용하여 관련 데이터의 각 집합을 고유하게 식별할 수 있어야 한다.

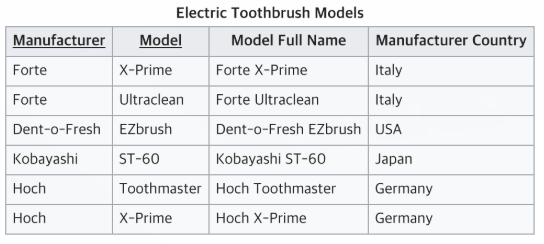






**제 2정규형(Second Normal Form, 2NF) - 부분 함수적 종속을 모두 제거 (모든 컬럼이 완전 함수적 종속을 만족)**

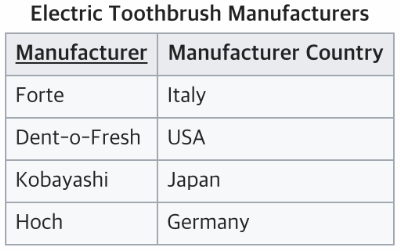
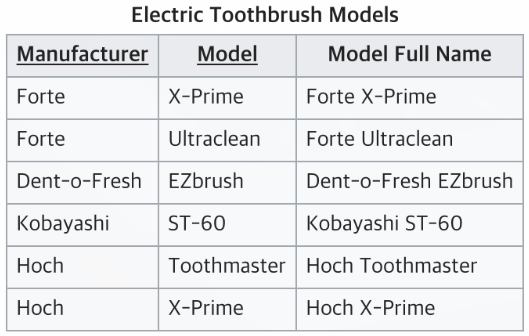
* 함수적 종속 : X의 값에 따라 Y값이 결정될 때 X -> Y로 표현하는데, 이를 Y는 X에 대해 함수적 종속 이라고 한다. X를 결정자라 하고 Y를 종속자라고 한다.  
  다른 말로 X가 바뀌었을 경우 Y가 바뀌어야만 한다는 것을 의미한다.
* 함수적 종속에서 X의 값이 여러 요소일 경우, 즉, {X1, X2} -> Y일 경우, X1와 X2가 Y의 값을 결정할 때 이를 완전 함수적 종속 이라고 하고, X1, X2 중 하나만 Y의 값을 결정할 때 이를 부분 함수적 종속 이라고 한다.



Manufacturer + Model -> Model Full Name ( 완전 함수적 종속)

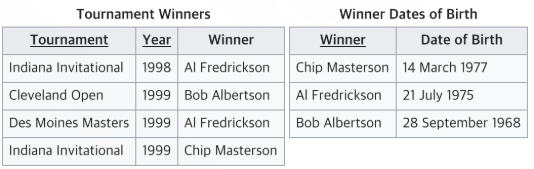
Manufacturer -> Manufacturer Country ( 부분적 함수 종속)

따라서 제 2정규형을 만족시키려면 분리시킬 필요가 있음



**제 3정규형(Third Normal Form, 3NF)**

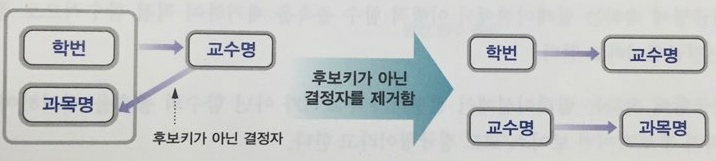
1. Relation이 제 2정규화 되었다.(The relation is in second normal form)
2. 기본 키(primary key)가 아닌 속성(Attribute)들은 기본 키에만 의존해야 한다  
   이행 함수적 종속 (x -> y-> z) 제거



Tournament, Year -> Winner 이면서 Winner -> Winner Date of Birth 여서 3정규형을 만족하지 못하였다.

따라서 Winner -> Date of Birth를 따로 빼준다.

**BCNF(Boyce-codd) 정규형**

: 결정자이면서 후보키가 아닌 것 제거, 모든 결정자 X가 후보키이어야 함 

4NF : 다치 종속성 제거

5NF : 조인 종속성 제거

## 

## **[ 무손실 분해 ]**

이상현상을 해결하기 위해서 우리는 릴레이션을 분해해야 한다는 것을 깨달았습니다. 하지만 하나의 릴레이션 R을 분해할 때 분해 후의 결과와 달라지면 문제가 발생하게 되므로 우리는 손실이 없이 릴레이션을 분해해야 합니다. 여기서 릴레이션 R을 R1과 R2로 분해할 때, R1▷◁R2를 만족하는 경우를 무손실 분해(Loseless-join decomposition)라고 합니다. 그리고 R1과 R2는 R1∩R2 -> R1 혹은 R1∩R2 -> R2 중 하나를 만족해야 합니다. 아래의 그림에서 교수 속성으로 분리한 후 다시 조인하면 원래의 릴레이션이 되지만 특강이름 속성을 사용하여 분해하는 경우는 창업전략의 교수님이 다르므로 원래의 릴레이션이 되지 않습니다. 그리므로 이 경우는 무손실 분해가 아닙니다.

정규화의 장점

1. 이상현상 제거
2. 데이터베이스 구조 확장 시 재 디자인 최소화(속성 추가 시 유용)
3. 사용자에게 데이터 모델을 더욱 의미있게 제공(현실감있게 반영)

단점

1. 너무 많은 정규화를 통해 잦은 JOIN연산이 필요해질 수 있다(느려짐)

반 정규화와 View, 인덱스를 적용하여 해결할 수 있는 지 검토한다!

**이상(Anomaly)**

* 삽입 이상(Insertion Anomaly)
* 삭제 이상(Deletion Anomaly)
* 갱신 이상(Modification Anomaly)

트랜잭션: DBMS에서 데이터를 다루는 논리적인 작업의 단위 => commit / rollback  
데이터베이스의 일관성이 보장되어야 하며 이를 위해 동시성 제어(concurrency control)와 회복 제어(recovery control)를 위한 모듈이 있으며 이 둘을 합쳐 트랜잭션 관리 모듈(transaction management module)이라고 한다.  
동시성제어 모듈 : 데이터베이스를 일관성 있게 유지하기 위하여 동시에 수행되는 트랜잭션들 사이의 상호작용을 제어한다.

[회복제어](https://ko.wikipedia.org/w/index.php?title=%ED%9A%8C%EB%B3%B5%EC%A0%9C%EC%96%B4&action=edit&redlink=1) 모듈(recovery control module): 데이터베이스를 일관성 있게 유지하기 위하여 업데이트를 하는 동안 시스템 장애에도 데이터베이스의 기존 상태가 유지된다.

트랜잭션 스케줄링은 다음과 같은 3가지 개념을 가진다.

1. 직렬 스케줄링(serial scheduling): 트랜잭션 연산들을 각 트랜잭션별로 연속적으로 실행하는 방법
2. 비직렬 스케줄링(nonserial scheduling): 트랜잭션 연산들을 상호적(interleaving)으로 병행 실행하는 방법
3. 직렬 가능 스케줄링(serializable scheduling): 비직렬 스케줄링 S가 항상 직렬 스케줄링 SS에 대해서 같은 결과를 가질 때 "S를 직렬가능"하다고 한다.

직렬가능 트랜잭션을 보장하기 위한 규약(protocol)이 있는데 [잠금](https://ko.wikipedia.org/w/index.php?title=%EC%9E%A0%EA%B8%88&action=edit&redlink=1)(locking) 방법과 [시간표](https://ko.wikipedia.org/wiki/%EC%8B%9C%EA%B0%84%ED%91%9C)(timestamp)가 바로 그것이다.

* DB에서 데이터를 다룰 때 장애가 일어난 경우 데이터를 복구하는 작업의 단위가 된다.
* DB에서 여러 작업이 동시에 같은 데이터를 다룰 때가 이 작업을 서로 분리하는 단위가 된다.
* 트랜잭션은 전체가 수행되거나 또는 전혀 수행되지 않아야 한다.(All or Nothing)

**트랜잭션 ACID성질**

1. 원자성(Atomicity): 트랜잭션에 포함된 작업은 전부 수행되거나 전부 수행되지 않아야 한다.
2. 일관성(Consistency): 트랜잭션을 수행하기 전이나 후나 데이터베이스는 항상 일관된 상태를 유지해야 한다.
3. 고립성(Isolation): 수행 중인 트랜잭션에 다른 트랜잭션이 끼어들어 변경중인 데이터 값을 훼손하지 않아야한다.
4. 지속성(Durability): 수행을 성공적으로 완료한 트랜잭션은 변경한 데이터를 영구히 저장해야 한다.

데이터의 무결성(Integrity)

데이터의 정확성, 일관성, 유효성이 유지되는 것을 말한다.

1. 개체 무결성 : 기본키를 구성하는 속성은 NULL값이나 중복값을 가질 수 없다
2. 널 무결성 : 특정속성 값이 Null이 될 수 없도록 하는 규정
3. 고유 무결성 : 특정 속성에 대해서 각 튜플이 갖는 값들이 서로 달라야 한다는 규정
4. 도메인 무결성 : 특정 속성의 값이, 그 속성이 정의된 도메인에 속한 값이어야 한다는 규정
5. 참조 무결성 : 외래키 값은 NULL이거나 참조 릴레이션의 기본키 값과 동일해야 한다는 규정  
    즉 릴레이션은 참조할 수 없는 외래키 값을 가질 수 없다는 규정
6. 키 무결성 : 하나의 테이블에는 적어도 하나의 키가 존재해야 한다는 규정

**제약조건**



데이터베이스 회복 기법의 종류

(<https://azurecourse.tistory.com/556>), <https://mangkyu.tistory.com/30?category=761304>

* 로그기반 회복
* 즉시갱신
* 지연갱신
* Check Point
* 그림자 페이지

**인덱싱**

**뷰(View)**

: 뷰는 사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 하나 이상의 기본 테이블로부터 유도된 이름을 가지는 가상 테이블이다.

뷰의 특징

* 기본테이블로부터 유도된 테이블이기 때문에 기본 테이블과 같은 형태의 구조 사용
* 가상테이블이기 때문에 물리적으로 구현되어 있지 않음
* 데이터의 논리적 독립성 제공
* 필요한 데이터만 정의해서 처리하기때문에 관리가 용이하고 명령문이 간단해짐
* 데이터를 안전하게 보호하는 효율적인 기법으로 사용 가능
* 기본 테이블의 기본키를 포함한 속성 집합으로 뷰를 구성해야 CRUD 가능
* 뷰도 다른 뷰의 기초가 될 수 있음
* 다른 뷰의 기초가 된 뷰를 삭제하면 다른 뷰도 자동으로 삭제됨
* 뷰의 정의를 변경할 수 없음, 즉 alter는 불가능

사용법

create view view\_name(속성1, 속성2) as select 문