**< 정규화 >**

1. **제 1정규화**

도메인이 원자 값이 아닌 경우 분리. ( 한 속성에 값이 여러 개)

1. **제 2정규화**

부분적 함수 종속이 존재 하는 경우 분리.

복합키를 가질 때 복합키 중에 하나의 키로만 결정되는 속성이 있는 경우

복합키가 아니고 단일 기본키로 이루어져 있으면 제 2 정규화 해당 x

1. **제 3정규화**

이행적 함수 종속을 제거

A -> B , B->C 🡺 A->C

* A->B , A->C 로 분리

1. **BCNF**

* A-> B, B->C로 분리 ( 후보키가 아닌 결정자를 후보키로 만들어 준다. )

1. **제 4정규화**

다치 종속을 제거

하나의 어트리뷰트가 다수의 어트리뷰트와 다대다 관계일 경우.

다대다 관계의 개수만큼의 릴레이션으로 쪼개준다.

1. **제 5정규화**

다치 종속을 제거.

하나이상의 어트리뷰트가 서로 다대다 관계일 때

다대다 관계의 개수만큼의 릴레이션으로 쪼개준다.

**< 개념적 설계 >**

1. 데이터 중심 설계

요구사항을 토대로 데이터를 분석하고 데이터 간의 관계, 스키마 설계

1. 데이터 처리 설계

“트랜잭션 설계”

* 트랜잭션이란 하나의 논리적 작업 단위로 수행되는 일련의 작업이다.
* 작업의 단위는 사용자가 설계할 수 있으며 하나의 작업이 전부 수행 되거나 전부 되지 않거나 의 의미가 포함되어 있다.
* Commit하면 작업을 수행되어 캐시에서 로그로 가서 담겨있던 작업들이 수행된다.

이때 자동으로 savepoint가 생성된다.

* Rollback 하게 되면 Commit을 한 상태로 돌아간다.
* 트랜잭션의 특성

1. 원자성
2. 일관성
3. 독립성
4. 영속성

<http://egloos.zum.com/sweeper/v/3003805>

**< 인덱스 >**

: ‘색인’ , 검색의 효율을 높이기 위해 사용된다.

- 해당 컬럼을 색인화하여 키 값을 MYI 파일에 저장된다. 이러한 Index는 Tree구조로 색인화되는 데 대표적으로 B+Tree가 사용된다. 키 값들이 색인화되어 있기 때문에 검색에 있어서는 높은 효율을 보이지만 DML에 취약하다는 단점이 있다.

- Insert, Delete의 경우 성능이 떨어질 수 있으며 Update 개념은 아예 존재하지 않는다.

- 기본키는 자동으로 index된다.

- 데이터의 중복이 많은 경우 인덱스를 만들어도 호용이 없다.

<https://lalwr.blogspot.kr/2016/02/db-index.html>

**< 테이블 생성 >**

CREATE TABLE test (

id int(11) not null auto\_increment,

name varchar(10) not null,

email varchar(30),

primary key(id),

index name\_index(name(10)) //name 필드를 name\_index라는 이름으로 인덱스함

)

