File sys 단점 : 복사 파일에서 수정된게 반영안됨, 보안문제, 동시접근문제

Super key : 해당 tuple을 구분할 수 있는 것들

Candidate key : super key의 최소 단위

Primary key : db 디자이너가 선택한 candidate key

Foreign key : 다른 테이블의 primary key

Why relational db? 물리적 위치로 찾게되면 위치 다 알고있어야하고 바뀌면 또 힘드니까 좌표를 기준으로 하지 않고 value를 기준으로 query

Natural join (1) 테이블을 쪼갤 때 합치는거 고려해서 공통되는 attribute를 기준으로 자르는

카테지안 프로덕트 한 후 common한 attribute의 이름이 같은거만

Inner join (2) attribute의 이름이 우연히 같을 수 있으니까 조건을 추가

Equi join (3)

2, 3은 attribute를 사용자가 설정 가능 /

Outter join : 카테지안 프로덕트 밖에 있는거도 포함시키는 (left right full)

정규화 : 데이터 중복을 최소화하려고 쪼개는거

Relation 을 쪼갤 때 Functional Dependencies을 보고. (a가 b를 결정하는 것)

Closure : 모든 functional dependency를 나타낸

First normal form : 모든 value가 atomic (모든 항목에 값이 있고, 중복기능 열이 없는)

Second normal form : a->b의 functional dependency에서 candidate key 의 proper subset이 non-prime attribute를 결정하는 경우가 없어야함.

<what if non prime attribute 끼리 functional dependency를 가진다면>

Update anomaly : 덜 update 한 경우

Insertion anomaly : 일부 값을 몰라서 insertion이 안되는

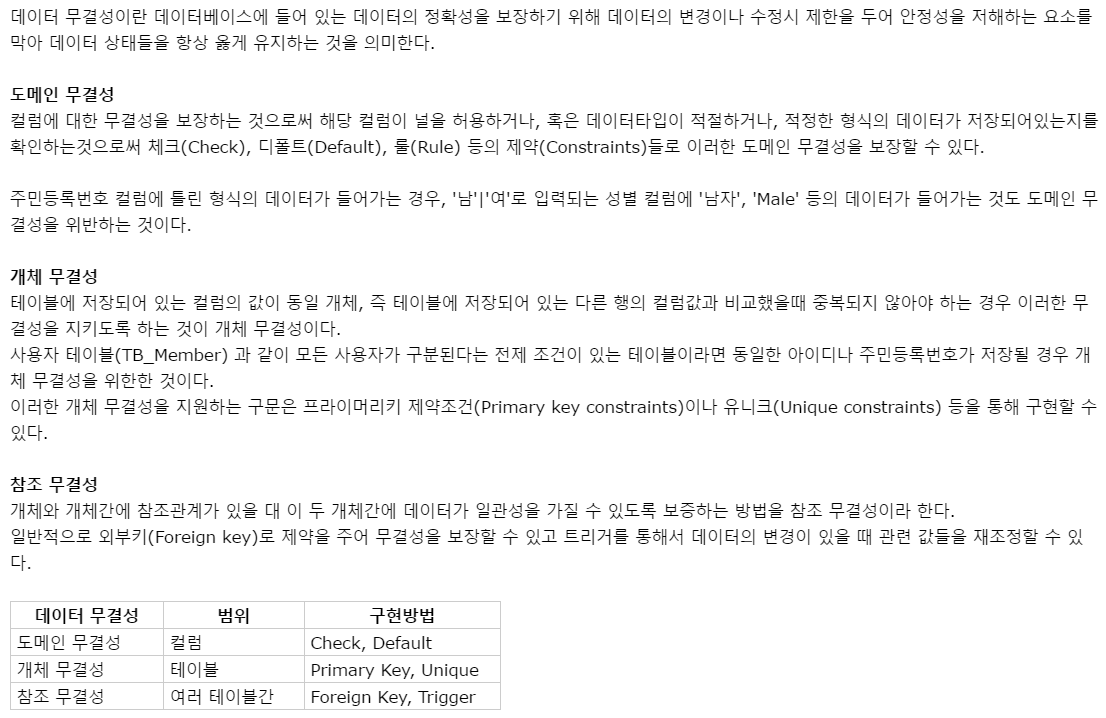
Delete anomaly : null이 허용되지 않아서 지울 수 없는

Boyce codd normal form : a->b가 있다면 a는 super key여야 함

어떤 임의의 relation이 잇을 때 BCNF를 만족하고 lossless-join을 만족하는 sub relation으로 만들 수 있다.

Third normal form : a->b에서 a b 둘다 candidate key에 포함되면 가능하게 하겠다

Dependency preservation : relation 을 쪼갰다가 합쳐도 functional dependency가 있어야함



트렌젝션

<https://ko.wikipedia.org/wiki/%EB%8D%B0%EC%9D%B4%ED%84%B0%EB%B2%A0%EC%9D%B4%EC%8A%A4_%ED%8A%B8%EB%9E%9C%EC%9E%AD%EC%85%98>