릴레이션 스키마의 설계 품질

릴레이션 스키마 설계

- 릴레이션 스키마를 설계하는 네 가지 개략적 지침
  1. 애트리뷰트들의 의미(semantics)가 스키마에서 명확한지 확인함
  2. 투플들에서 중복되는 값들을 줄임
  3. 투플들에서 널 값들을 줄임
  4. 가짜 투플(spurious tuple)을 허용하지 않음

릴레이션 애트리뷰트들의 의미

- 릴레이션의 의미는 투플에 있는 애트리뷰트 값들의 해석에 대한 의미와 연관됨
  - 릴레이션에 속하는 애트리뷰트들의 의미를 얼마나 쉽게 설명할 수 있는가 하는 것은 릴레이션이 얼마나 잘 설계되었는지를 나타낼 수 있는지를 나타내는 비정형적인 척도임
  - COMPANY 관계 데이터베이스 스키마를 단순화시킨 스키마에서
    - EMPLOYEE, DEPARTMENT, PROJECT 릴레이션의 의미는 간단하다.
    - DEPT_LOCATIONS, WORKS_ON 스키마의 의미는 조금더 복잡하지만 명확한 의미를 가짐

Ename	Ssn	Bdate	Address	Dnumber

Dname	Dnumber	Dmgr_ssn

Pname	Pnumber	Plocation	Dnum

Dnumber	Dlocation

Ssn	Pnumber	Hours

릴레이션 애트리뷰트들의 의미

- 지침 1.
  - 의미가 쉽게 전달되도록 릴레이션 스키마를 설계한다. 여러 개의 엔티티 타입과 관계 타입의 애트리뷰트들을 섞어서 하나의 릴레이션을 구성하면 안 된다. 직관적으로, 하나의 릴레이션 스키마가 하나의 엔티티 타입이나 하나의 관계 타입에 대응된다면, 그 의미를 해석하고 설명하기가 쉽다. 그렇지 않고, 여러 개의 엔티티 타입과 관계 타입이 한 릴레이션에 섞이게 되면, 의미가 모호하게 되고 릴레이션을 쉽게 설명할 수 없다.
- 지침 1 위반 사례

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Ploation

투플에서 중복된 정보와 갱신 이상

- 기본 릴레이션들에 의해 사용되는 저장 공간 최소화
  - 스키마 설계의 목표 중 하나
  - 보통의 경우 기본 릴레이션들이 물리적으로 저장됨
  - EMP_DEPT vs. EMPLOYEE & DEPARTMENT
  - EMP_PROJ vs. EMPLOYEE & PROJECT
- 자연조인 저장의 또 다른 문제점: 갱신 이상 (update anomaly) [예: EMP_DEPT]
  - 삽입 이상 (insertion anomaly): 아직 부서가 없는 새로운 사원 삽입 (일관성 문제)/ 아직 사원이 없는 새로운 부서 삽입 (추가적인 사원 할당 시)
  - 삭제 이상 (deletion anomaly): 특정 부서에서 일하는 유일한 사원을 삭제 시 부서 정보도 모두 삭제 됨
  - 수정 이상 (modification anomaly): 부서 정보 갱신시 그 부서에서 일하는 모든 사원의 부서 정보 모두 수정 (일관성 문제)

중복된 정보

Redundancy

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

Redundancy Redundancy

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation
123456789	1	32.5	Smith, John B.	ProductX	Bellaire
123456789	2	7.5	Smith, John B.	ProductY	Sugarland
666884444	3	40.0	Narayan, Ramesh K.	ProductZ	Houston
453453453	1	20.0	English, Joyce A.	ProductX	Bellaire
453453453	2	20.0	English, Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong, Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong, Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong, Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong, Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya, Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya, Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar, Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar, Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace, Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace, Jennifer S.	Reorganization	Houston
888665555	20	Null	Borg, James E.	Reorganization	Houston

투플에서 중복된 정보와 갱신 이상

- 지침 2.
  - 릴레이션에서 삽입, 삭제 또는 수정 이상이 생기지 않도록 기본 릴레이션 스키마를 설계한다. 만약 어떤 이상이 존재하면, 그것을 잘 이해하여 데이터베이스를 수정하는 프로그램들이 올바르게 동작하도록 작성한다.
  - 첫번째 지침과 일맥 상통
  - 어떤 질의(자주 발생하는)들을 효율적으로 수행하기 위해서는 때로는 이러한 지침들을 따르지 않을 수도 있음
  - 기본 릴레이션 외에도 뷰를 활용하여 자동 변경이나 트리거를 사용함으로 데이터 베이스의 일관성을 잃지 않도록 주의해야 함

투플의 널 값

- 널 값으로 인한 문제점
  - 저장 단계: 저장 단계에서 공간을 낭비
  - 논리 단계: 조인 연산들을 명시하기 힘듦, 애트리뷰트들의 의미를 이해하기 어려움
  - COUNT, SUM과 같은 집단 함수들이 적용되었을 때 널 값들을 어떻게 처리할 것인가 불명확
- 널 값의 해석
  - 널 값을 가진 애트리뷰트가 이 투플에는 적용되지 않음 (예: 비자 상태는 자국의 학생들에게는 해당 안됨)
  - 이 투플에서 애트리뷰트 값이 아직 알려져 있지 않음 (예: 직원의 생년월일은 모릴 수 있음)
  - 애트리뷰트 값을 알고 있지만 아직 기록되지는 않음 (예: 사원의 집 전화번호가 존재하지만 가용하지 않거나 아직 기록되지 않았을 수 있음)
  - 따라서 널 값을 동일하게 표현하면 널 값이 갖는 여러 의미를 훼손하게 됨

투플의 널 값

- 지침 3.
  - 널 값을 자주 가질 수 있는 애트리뷰트를 가능한 한 기본 릴레이션의 애트리뷰트로 포함하지 않는다. 만약 널 값을 포함할 수밖에 없다면 예외적인 경우에만 포함하고 대다수 투플들에 대해서는 널 값이 없도록 한다.
- 널 값과 관련된 릴레이션 설계의 척도
  - 릴레이션에서 널을 가질지도 모르는 열을 포함할지 혹은 (적절한 키 열들을 가진) 그러한 열을 분리된 릴레이션으로 가질지를 결정하는 중요한 척도는 다음의 두 가지가 있음
  1. 저장 공간의 효율적 사용
  2. 널 값을 갖는 조인을 피함

예를 들어, 사원들 중에 15% 만이 개인 사무실을 가지고 있는 경우, Office_number 라는 애트리뷰트를 EMPLOYEE에 포함시키기 보다는 EMP_OFFICES (Essn, Office_number) 라는 새로운 릴레이션을 생성

가짜 투플의 생성

- 자연 조인 시에 가짜 정보 생성 가능성
  - 아래의 EMP_PROJ 릴레이션 대신에 두 개의 릴레이션 스키마 EMP_LOCS 와 EMP_PROJ1을 사용할 경우 저장되는 데이터는 EMP_PROJ 의 투플들에 대해 해당 스키마의 애트리뷰트들로 프로젝트 연산들을 수행하여 얻은 결과와 동일함
  - 하지만 거꾸로 EMP_LOCS와 EMP_PROJ1 을 기본 스키마로 사용하고, EMP_PROJ 의 정보를 복원하고자 한다면 문제가 발생함
  - 예를들면, 두 개의 테이블은 Ename, Plocation을 기반으로 조인하여 얻어질 것임. 그럴 경우에 가짜 투플들이 다량으로 생성될 수 있음

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Ploation

EMP_LOCS

Ename	Plocation

EMP_PROJ1

Ssn	Pnumber	Hours	Ename	Pname

가짜 투플의 생성

- 지침 4.
  - 가짜 투플들이 생성되지 않기 위해서, 적절하고 관련된 (기본키, 외래키) 애트리뷰트를 가지고 동등 조건으로 조인할 수 있는 릴레이션 스키마를 설계함. (외래키, 기본키) 조합을 제외하고 릴레이션들이 대응하는 애트리뷰트 (matcihng attributes)들을 가지지 않도록 한다. 왜냐하면 그러한 애트리뷰트들에 대한 조인은 가짜 투플들을 생성하기 때문이다.
  - 앞의 결과에서 Ename, Plocation 은 EMP_PROJ1 의 기본키와 관련된 외래키가 아님

설계 지침에 대한 요약

- 추가적인 분석 도구 없이 발견할 수 있는 문제들
  - 릴레이션에 삽입이나 수정 시에 중복 작업의 원인이 되고, 릴레이션으로부터 삭제 시 예상치 못한 정보의 궁극적인 손실을 유발시키는 이상
  - 널로 인한 저장 공간의 낭비와 널 값으로 인한 선택, 집계 연산 및 조인 수행의 어려움
  - 타당한 (외래키, 기본키) 관계를 갖지 않는 매치된 애트리뷰트들의 기본 릴레이션들에 대한 조인 시 올바르지 않은 가짜 데이터의 생성
- 각 릴레이션 스키마가 잘 설계 되었는지에 대한 체계적이고 공식적인 개념과 이론
  - 함수적 종속성
  - 세 가지 정규형과 Boyce-Codd Normal Form(BCNF) 정규형

함수적 종속성

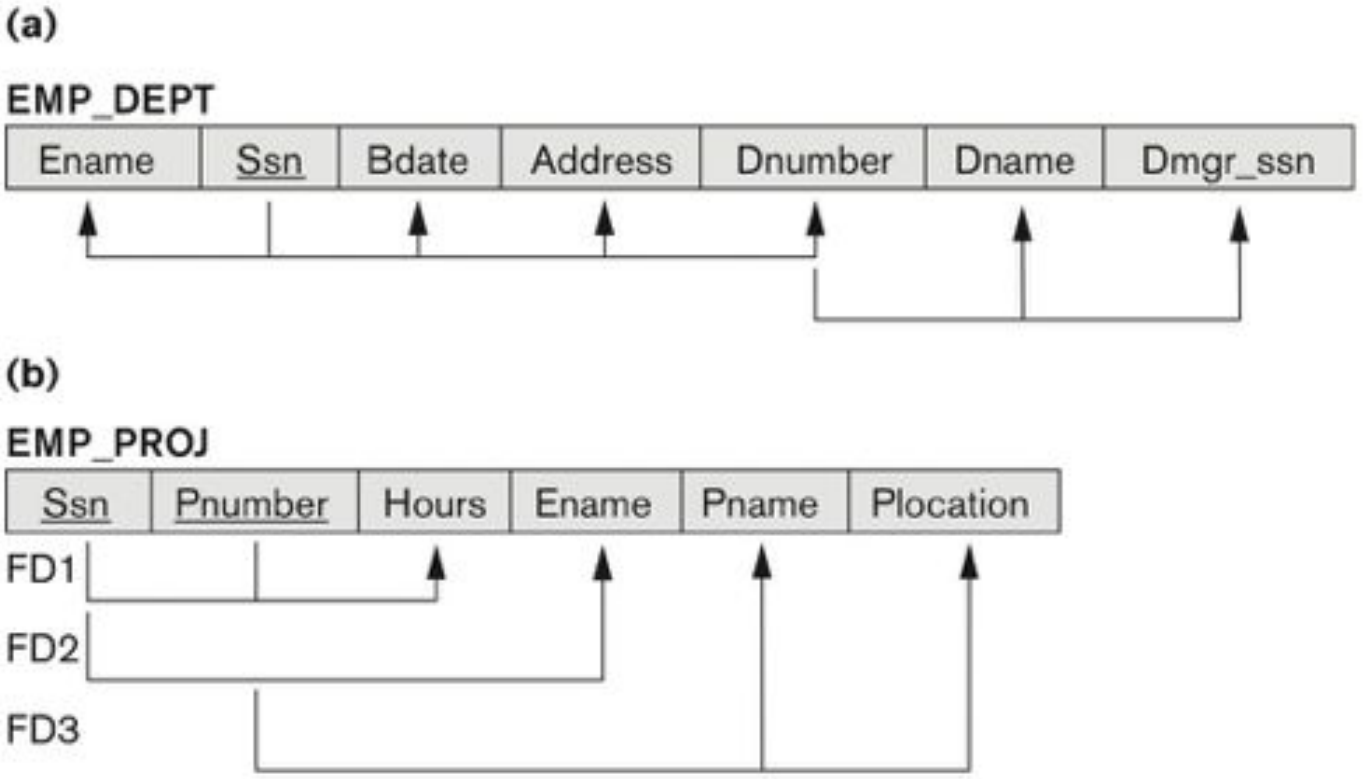
함수적 종속성의 정의

- 함수적 종속성 (functional dependency)은 애트리뷰트들의 두 개의 집합 사이의 제약조건임
  - 관계 데이터 베이스가 n 개의 애트리뷰트 A1, A2, ... An으로 구성되어 있다고 가정
  - 전체 데이터 베이스를 전체 릴레이션 스키마 R={A1,...,An}으로 설명할 수 있다고 가정
    - 데이터베이스를 하나의 전체 테이블로 저장한다는 것을 의미하는 것은 아님
- 정의
  - R의 애트리뷰트들의 부분집합인 X와 Y 사이의 함수적 종속성(X → Y 로 표기)은 릴레이션 스키마 R 의 릴레이션 상태 r을 구상할 수 있는 가능한 투플들에 대한 제약 조건을 지정하는 것임. 여기에서 제약조건은 r의 임의의 두 투플 t1, t2에 대해서 t1[X]=t2[X] 이면 반드시 t1[Y]=t2[Y] 이어야 한다는 것임
  - 임의의 투플의 Y 부분의 값은 X 부분의 값에 의해 결정되거나 종속 된다는 것
  - 이러한 경우에 X에서 Y 로의 함수적 종속성이 존재한다고 말하거나, Y가 X에 함수적으로 종속(functionally dependent)된다고 말함
  - 함수적 종속성을 FD 또는 f.d. 라고 줄여서 표기

함수적 종속성의 정의

- 함수적 종속성의 명시
  - 함수적 종속성을 찾기 위해서 R의 애트리뷰트들이 서로 어떻게 연관되어 있는지 애트리뷰트들의 의미를 이해
  - R의 애트리뷰트들의 두 집합의 의미에 함수적 종속성이 존재한다면 종속성을 제약 조건으로 명시
  - 어떠한 함수적 종속성은 특정 릴레이션을 참조하지 않고 보편적으로 이해되는 의미를 가진 애트리뷰트들의 특성 만으로 명싣될 수 있음
    - 예: {State, Driver_liscence_number} → {Ssn}
    - EMP_PROJ 스키마의 예: Ssn → Ename | Pnumber → {Pnumber, Plocation} | {Ssn, Pnumber} → Hours
- 함수적 종속성의 성질
  - 함수적 종속성은 릴레이션 스키마 R의 특성이지, R의 어떤 특정 적법한 릴레이션 상태 r의 특성은 아님
    - 릴레이션 외연 r 로 부터 자동적으로 추측할 수 있는 것이 아니라 R의 애트리뷰트들의 의미를 아는 사람이 명시적으로 정의해야 함
    - 이미 생성된 릴레이션에 대해 애트리뷰트 간의 의미와 관계를 알지 못하는 한 함수적 종속성의 성립 여부를 판단할 수 없음

함수적 종속성 예시


(a)
EMP_DEPT
Ename
Ssn
Bdate
Address
Dnumber
Dname
Dmgr_ssn
(b)
EMP_PROJ
Ssn
Pnumber
Hours
Ename
Pname
Plocation
FD1
FD2
FD3

함수적 종속성 예측

- 함수적 종속성은 릴레이션 상태 (외연)으로부터 결정할 수 없지만, 외연을 통해 함수적 종속성이 성립하지 않는 예는 확인해 볼 수 있음
- 또한, 외연으로부터 함수적 종속성을 예측할 수 있다.
- Which FDs *may exist* in this relation?

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d2
a2	b2	c2	d3
a3	b3	c4	d3

  - $A \rightarrow B, B \rightarrow A, B \rightarrow C, C \rightarrow B, \{A, B\} \rightarrow C, \{A,B\} \rightarrow D, \{C,D\} \rightarrow B, D \rightarrow C$
  - 위의 관계들 중에서 함수적 종속성을 만족시킬 수 있는 관계들을 찾아보고, 만족시키지 않을경우 이를 위반하는 투플들의 예를 들어 보아라

정규화 기본 이론

기본키를 기반으로 한 정규형

- 함수적 종속성 이용
  - 관계 스키마의 의미에 관한 몇 가지 측면을 명시하기 위해서 함수적 종속성을 이용하는 것을 설명
  - 각 릴레이션마다 **함수적 종속성들의 집합**이 주어지며, 각 릴레이션에 **기본키**가 있다고 가정
  - 위의 정보와 정규형을 위한 검사(조건)들을 함께 사용하여 릴레이션 스키마 설계를 위한 정규형 과정을 수행
- 관계형 설계 프로젝트의 두 가지 접근 방식
  - 우선 ER, EER 과 같은 개념적인 모델을 사용하여 개념적인 스키마 설계를 수행한다. 그러고 나서 개념적인 설계를 릴레이션들의 집합으로 사상한다
  - 기존의 파일, 서식 혹은 보고서의 구현들로부터 유도된 외부 정보를 기반으로하는 릴레이션들을 설계한다
- 정규형의 실제 사용
  - 데이터베이스 설계자가 릴레이션을 최상의 수준의 정규형으로 정규화할 필요가 없음
    - 성능을 위해서 릴레이션을 낮은 수준의 정규형으로 둘 수도 있음
    - 하지만, 이상 현상을 처리하는 비용을 발생시킬 수 있음
  - 비정규화(denormalization): 더 높은 정규형 릴레이션들을 조인한 결과를 기본 릴레이션(더 낮은 정규형에 속함)으로 저장하는 과정

릴레이션들의 정규화

- 정규화 과정 (Codd, 1972)
  - 어떤 릴레이션 스키마가 어떤 정규형을 만족하는지 확인하는 일련의 테스트
  - 분석에 의한 릴레이션 설계: 각 릴레이션이 정규형의 기준을 만족하는지를 검사하고, 필요하다면 릴레이션을 분해
  - 제 1정규형, 제 2 정규형, 제 3 정규형 제안
  - Boyce와 Codd 가 제 3 정규형보다 강한 정의를 제안하고 BCNF 라고 부름
  - 위의 네 가지 정규형은 모두 한 릴레이션의 애트리뷰트들 사이의 함수적 종속성들을 기반으로 함
- 데이터의 정규화 (normalization)
  - 1) 중복을 최소화하고 2) 삽입, 삭제, 수정 이상을 최소화 하기 위해 함수적 종속성과 기본키를 기반으로, 주어진 릴레이션 스키마를 분석하는 과정
  - 정규화는 높은 수준의 성공적 설계를 위한 제거(filtering) 또는 정제(purification) 과정으로 간주될 수 있음
  - 어떤 조건(정규형 검사)들을 만족하지 못하는 바람직하지 못한 릴레이션 스키마는 이런 검사를 만족하는 더 작은 릴레이션 스키마들로 분해되어 바람직한 특성을 갖게 됨
- 정규화 과정의 기능
  - 릴레이션 스키마의 애트리뷰트들 사이에 존재하는 함수적 종속성과 키들을 기반으로 그 스키마를 분석할 수 있는 정형적 구조
  - 관계 데이터베이스를 임의의 수준으로 정규화 할 수 있도록 개개의 릴레이션 스키마에서 수행할 수 있는 일련의 정규형 검사

릴레이션의 정규형

- 정의
  - 한 릴레이션의 정규형(normal form)은 그 릴레이션이 만족하는 가장 높은 정규형을 나타낸다. 즉, 그 릴레이션이 어느 정도까지 정규화되었는지를 나타낸다. 정규형들이 다른 요인들과 동떨어져서 고려되면 좋은 데이터베이스 설계를 보장하지 못한다. 일반적으로 데이터베이스의 각 릴레이션 스키마가 BNCF 정규형인지 제3 정규형인지 개별적으로 검사하는 것은 충분하지 않다. 그보다 분해에 의한 정규화 과정은 릴레이션 스키마들이 함께 가져야 하는 추가적인 특성들을 만족하도록 해야 한다.

- 릴레이션 스키마들이 가져야 하는 추가적인 특성
  - 비부가적 조인(nonadditive join) 또는 무손실 조인 특성(lossless join property): 분해 후에 생성된 릴레이션 스키마에 대해서 가짜 투플 문제가 발생하지 않는 것을 보장
  - 종속성 보존 특성 (dependency preservation property): 함수적 종속성이 분해 후 생성된 릴레이션들의 일부에서 반드시 표현됨

키의 정의와 키에 참여하는 애트리뷰트

- 키의 정의
  - 릴레이션 스키마 R = {A1, A2, ... , An} 의 슈퍼키(superkey) $S \subseteq R$ 인 애트리뷰트들의 집합으로서, R 의 임의의 적법한 릴레이션 상태 r에 있는 두 투플 t1과 t2에 대해서 t1[S]=t2[S] 가 성립하지 않는다. 키 K 는 K로부터 어떤 애트리뷰트라도 제거하면 K가 더 이상 슈퍼키가 되지 않는 성질을 가진 슈퍼키를 일컫는다.
  - 키와 슈퍼키 사이의 차이점은 키가 최소한의 애트리뷰트들의 집합 이라는 것임
    - 예: EMPLOYEE 에서 {Ssn}은 키인 반면, {Ssn}, {Ssn, Ename}, {Ssn, Ename, Bdate}와 Ssn을 포함하는 애트리뷰트의 집합은 모두 슈퍼키임
  - 릴레이션 스키마가 두 개 이상의 키를 가졌을 때, 각각을 후보키 (candidate key)라고 함
  - 그리고 이 후보키들 중 하나를 기본키 (primary key)라 하고 나머지 키들을 보조키 (secondary key)라고 함
- 정의
  - 릴레이션 R의 임의의 후보키에 속하는 애트리뷰트를 주요 애트리뷰트(prime attribute)라 하고 그렇지 않은 애트리뷰트를 비주요 애트리뷰트(nonprime attribute)라고 한다.
  - 간략화된 COMPANY 데이터베이스에서 Ssn 과 Pnumber 는 WORKS_ON의 주요 애트리뷰트인 반면, 다른 애트리뷰트들은 비주요 애트리뷰트임

제1 정규형

- 제1 정규형 (1NF: First Normal Form)
  - 기본적인 관계 모델의 릴레이션에 대한 엄격한 정의의 일부분
  - 다치, 복합 애트리뷰트, 혼합 애트리뷰트들을 허용하지 않기 위한 목적 때문에 등장
  - 제1 정규형은 애트리뷰트의 도메인이 오직 원자값 (atomic value)만 포함하고, 투플에 모든 애트리뷰트 값은 그 애트리뷰트 도메인에 속하는 하나의 값이어야 함
- 제 1 정규형 위반의 예
  - 아래 그림의 (a)와 (b): 한 부서가 여러 로케이션에 있을 수 있음

(a)

DEPARTMENT

(b)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

제1 정규형을 만족하도록 하는 방법

1. 제 1 정규형을 어긴 Dlocation 애트리뷰트를 제거하고 이를 DEPARTMENT 의 기본키인 Dnumber 와 함께 별개의 릴레이션 DEPT_LOCATIONS에 추가한다. 이 릴레이션의 기본키는 {Dnumber, Dlocation} 이다. 따라서 제 1 정규형이 아닌 하나의 릴레이션이 제 1 정규형인 두 개의 릴레이션으로 분해된다.
2. (C)에서와 같이 기본키를 확장하여 {Dnumber, Dlocations}으로 한다. 하지만 이 방법은 릴레이션 내에 중복된 정보들을 발생시키게 된다는 단점을 가진다.
3. 만약 애트리뷰트가 가질 수 있는 값들의 최대 개수를 알 수 있다면, Dlocation1, Dlocation2, Dlocation3 와 같이 애트리뷰트로 분해할 수 있다. 이 경우에는 세 개 보다 적은 수의 위치를 가지는 경우 널 값을 많이 유발하는 단점을 가진다. 또한, 질의 작성 시에 어려워진다거나 애트리뷰트들 사이에 순서가 생긴다는 단점을 가지게 된다.

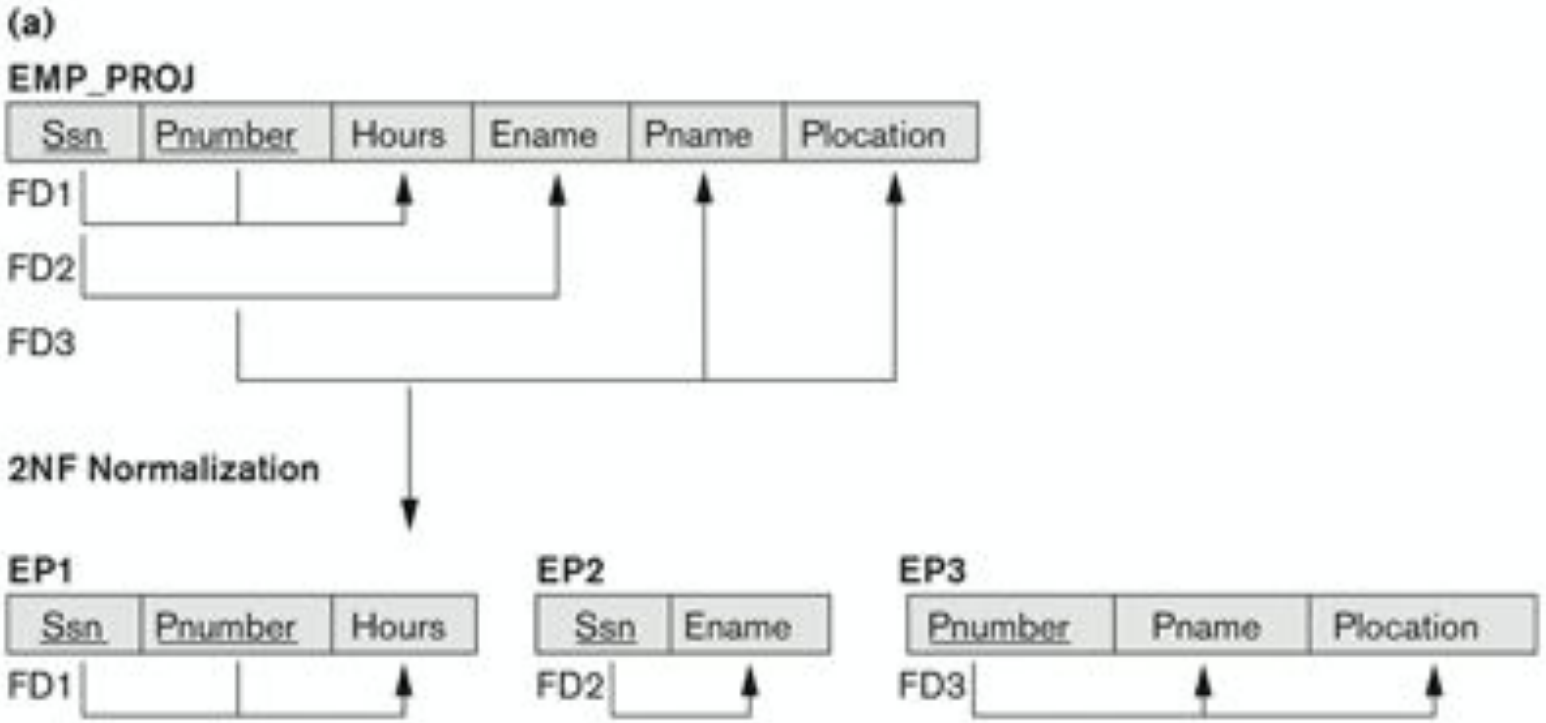
- 위의 세 가지 방법 중에서 첫번째 방법이 가장 우수하게 여겨짐

제2 정규형

- 제2 정규형 (2NF: Second Normal Form)은 완전 함수적 종속성의 개념에 기반을 둠
  - 완전 함수적 종속성 (full functional dependency): 함수적 종속성 $X \rightarrow Y$ 에서 X로부터 임의의 애트리뷰트 A를 제거하면 함수적 종속성 $X \rightarrow Y$ 가 성립되지 않는 경우
  - 부분 종속성 (partial dependency): 함수적 종속성 $X \rightarrow Y$ 가 X의 임의의 애트리뷰트 A 를 제거해도 성립하는 경우
  - 완전 함수 종속성이나 부분함수 종속성을 정의할 때에는 X가 기본키일 필요는 없음
- 정의
  - 릴레이션 스키마 R의 모든 비주요 애트리뷰트들이 R의 기본키에 대해 완전하게 함수적으로 종속하면 R은 제2정규형이라고 한다.
  - 제2 정규형 검사는 함수적 종속성의 왼편 애트리뷰트들이 기본키의 일부인지 검사하는 것을 수반함
  - 만약 한 릴레이션 스키마가 제2정규형이 아니면 여러 개의 제2정규형 릴레이션으로 정규화시킬 수 있음.

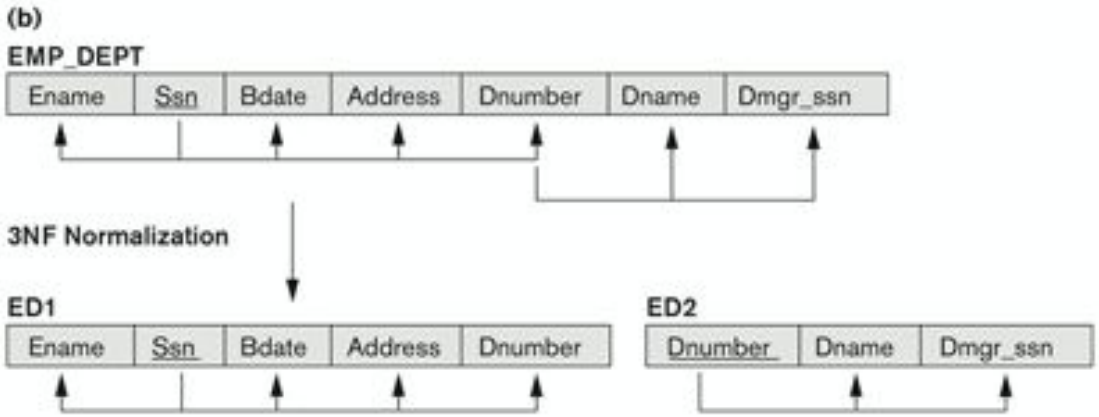
제 2 정규화 예시

- 함수적 종속성의 왼편 애트리뷰트들이 기본키의 일부인지 검사하는 것을 수반



제3 정규형

- 제3 정규형(3NF: Third Normal Form)
  - 이행적 종속성 (transitive dependency)의 개념에 기반
  - 이행적 종속성 (transitive dependency): 릴레이션스키마 R 에서 후보키가 아니고 어떤 키의 부분집합도 아닌 애트리뷰트들의 집합 Z가 존재하여 $X \rightarrow Z$ 와 $Z \rightarrow Y$가 만족될 때, 함수적 종속성 $X \rightarrow Y$ 를 이행적 종속성이라고 부른다.
- 정의
  - 릴레이션스키마 R이 제2 정규형이고 R의 어떤 비주요 애트리뷰트도 기본키에 이행적으로 종속하지 않으면 릴레이션 스키마 R이 제3 정규혀이다.
- 예시
  - 아래 EMP_DEPT 는 제2 정규형을 만족하지만 (기본키가 한개의 애트리뷰트) Dnumber 에 대해서 이행적 종속성 존재



기본키에 기반을 둔 정규형

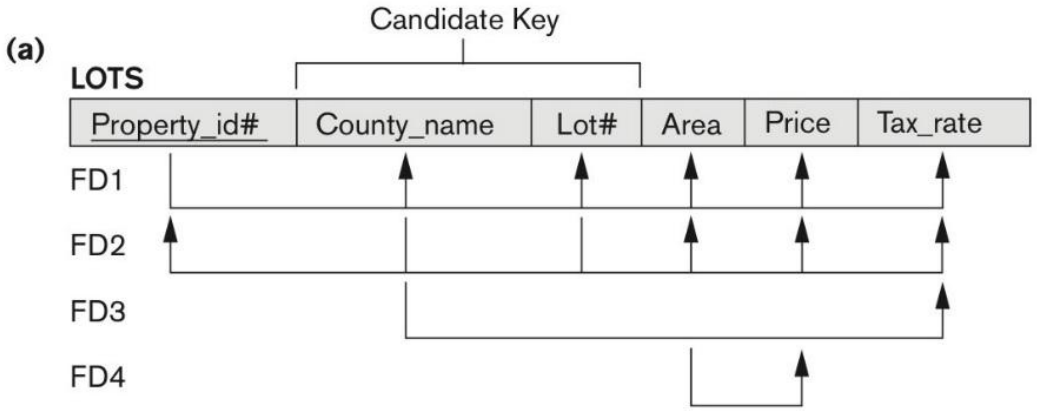
정규형	검사	방법(정규화)
제1 정규형	릴레이션들이 다치 애트리뷰트들 혹은 중첩 릴레이션들을 가지지 않아야 함	각 애트리뷰트나 중첩 릴레이션을 위한 새로운 릴레이션을 만듦
제2 정규형	여러 애트리뷰트들로 구성된 기본키를 가지는 각 릴레이션들에 대해, 키가 아닌 애트리뷰트들은 기본키의 일부분에만 함수적으로 종속해서는 안됨	부분키와 이에 종속하는 애트리뷰트들에 대해 새로운 릴레이션으로 분해된다. 분해되기 전에 원래의 기본키를 가진 릴레이션을 유지하고 그 릴레이션에서 다른 애트리뷰트들은 완전히 그 키에 종속하도록 한다.
제3 정규형	릴레이션에서 키가 아닌 애트리뷰트들이 키가 아닌 다른 애트리뷰트에 함수적으로 종속 해서는 안 된다. 즉 키가 아닌 애트리뷰트들이 기본키에 이행적으로 종속해서는 안된다.	키가 아닌 다른 애트리뷰트들을 함수적으로 결정하는 키가 아닌 애트리뷰트들을 포함하는 새로운 릴레이션으로 분해한다.

제2 정규형과 제3 정규형의 일반적인 정의

- 일반적으로 릴레이션 스키마를 설계할 때에는 부분 종속성이나 이행적 종속성이 없도록 하는 것이 좋음
  - 부분 종속성이나 이행적 종속성들이 삽입, 수정, 삭제 이상 등을 유발할 수 있음
- 제2 정규형과 제3 정규형의 일반적인 정의
  - 지금까지의 정의는 후보키를 고려하지 않고 기본 키만 가지고 정의했음
  - 일반적인 정의에서 주요 애트리뷰트는 후보키에 속하는 애트리뷰트임
  - 2, 3 정규형에서 부분 함수적 종속성, 완전 함수적 종속성, 이행적 종속성들은 릴레이션의 모든 후보키들을 고려하도록 함
- 제2 정규형의 정의
  - 릴레이션 스키마 R의 있는 비주요 애트리뷰트 A가 R의 모든 키에 완전하게 함수적으로 종속하면 R은 제2 정규형이다
- 제3 정규형의 정의
  - 릴레이션 스키마 R의 모든 함수적 종속성 $X \rightarrow A$에 대해서 (a) X가 R의 슈퍼키이거나 (b) A가 R의 주요 애트리뷰트이면 R은 제3 정규형이다.

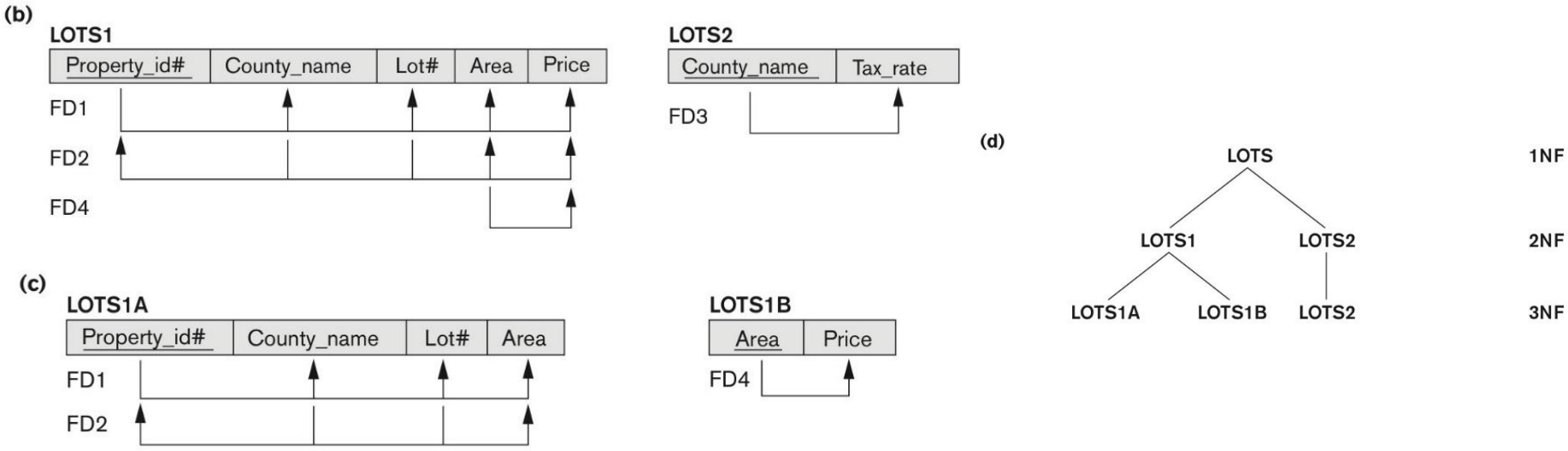
릴레이션 스키마 예시: LOTS

- 여러 주의 여러 군(contry)에서 매물로 나온 토지들을 표현
- LOTS에 두 개의 후보키 Property_id# 와 {Country_name, Lot#} 가 있다고 가정
  - Lot#는 군 내에서 고유한 값을 가지며 Property_id#는 주 전체에서 고유한 값을 가진다
- 함수적 종속성
  - FD1, FD2: 후보키에 근거한 함수적 종속성
  - 추가적인 함수적 종속성 가정
    - FD3: Country_name → Tax_rate
    - FD4: Area → Price
    - FD3: 세율이 한 군 내에서는 고정되어 있음 (즉, 같은 군 안에서는 토지에 따라서 세율이 달라지지 않음)
    - FD4: 토지의 가격은 그 토지가 속해있는 군에는 관계없고 면적에 의해서 결정됨 (세금을 목



제2 정규화와 제3 정규화

- 제2 정규화 (b)
  - FD3는 후보키 {Coutry_name, Lot#}에 부분적으로 의존하므로 제 2정규형의 정의를 만족하기 위해서 다음과 같이 분해
  - FD4는 제2 정규형을 위반하지 않음
- 제3 정규화 (c)
  - Area가 LOTS1의 슈퍼키가 아니고 Price 가 주요 애트리뷰트가 아니기 때문에 제3 정규형 위반
  - 제3 정규형을 만족시키기 위해서는 LOTS1 을 LOTS1A, LOTS1B로 분해해야함
- 릴레이션이 제3 정규형을 만족시키면 자동으로 제2 정규형이 만족됨



제 3 정규형의 일반적인 정의의 해석

- 제3 정규형의 (a) 조건의 위반
  - 비주요 애트리뷰트는 다른 비주요 애트리뷰트를 결정한다. 즉 제 3 정규형을 위배하는 이행적 종속성이 존재
  - R의 어떤 키의 진부분집합은 함수적으로 비주요 애트리뷰트를 결정한다. 즉, 제2 정규형을 위반하는 부분 종속성이 존재
- 제3 정규형의 대안적 정의
  - 릴레이션 스키마 R의 모든 비주요 애트리뷰트가 다음 두 조건을 만족하면 R은 제3 정규형이다
    - R의 모든 키에 대해서 완전하게 함수적으로 종속한다
    - R의 모든 키에 대해서 비이행적으로 종속한다
- 제3 정규형의 일반적인 정의 중 조건 (b)를 보면, 이것은 잠재적으로 문제가 될 가능성이 있지만 특정 함수적 종속성이 제3 정규형의 정의를 만족할 수 있음
  - Boyce-Codd 정규형