관계 모델(Relational Model)

Chapter 3

왜 관계 모델을 배울까?

- 가장 널리 사용되는 모델
  - 공급업체: IBM, Informix, Microsoft, Oracle, Sybase, etc.
- 구형 모델 잔존
  - 예: IBM의 IMS
- 최근의 경쟁상대: 객체지향(object-oriented) 모델
  - ObjectStore, Versant, Ontos
  - 두 모델이 합성되어: 객체-관계(object-relational)모델
    * Informix Universal Server, UniSQL, O2, Oracle, DB2

관계 데이타베이스: 정의

- 관계 데이타베이스: 릴레이션(relation)의 모임
- 릴레이션: 다음 두 부분으로 구성된다.
  - Instance : 행(row)과 열(column)로 구성된 일종의 테이블.
    #행의 수 = 카디널리티(cardinality)
    #필드의 수 = 차수(degree / arity).
  - Schema : 릴레이션 이름 + 각 필드의 이름과 타입
    * E.G. 학생(학번: 문자열, 이름: 문자열, 로그인: 문자열, 나이: 정수, 평점평균: 실수)
- 릴레이션은 행, 즉 투플(tuple)의 집합으로 볼 수 있다 (즉,모든 행은 서로 다르다).

학생 릴레이션의 예제 인스턴스

학번	이름	로그인	나이	평점평균
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

- 카디널리티= 3, 차수= 5, 행들은 모두 다르다.
- 한 릴레이션 인스턴스에 속하는 필드들이 모두 달라야 할까?

관계 질의어(Relational Query Language)

- 단순하고 강력한 데이타 질의 기능 제공 - 관계 모델의 장점.
- 질의를 직관적으로 작성할 수 있으며, DBMS가 효율적인 수행을 책임진다.
  - 핵심사항: 관계 질의는 의미가 명쾌하다.
  - 결과는 동일하도록 보장하면서도, 연산들의 순서를 여러가지로 바꾸어 최적화하는 것이 가능하다.

질의어 SQL

- IBM이 1970년대에 개발 (system R)
- 여러 공급업체들이 이용하였기 때문에 표준이 필요해짐
- 표준:
  - SQL-86
  - SQL-89 (minor revision)
  - SQL-92 (major revision, 현행표준)
  - SQL-99 (major extensions)

질의어 SQL (계속)

- 18세 학생을 모두 찾으려면:

```
SELECT *
FROM 학생 S
WHERE S.나이=18
```

학번	이름	로그인	나이	평점평균
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2

- 이름과 로그인만 찾으려면, 첫 줄을 다음처럼 바꾼다:

```
SELECT S.이름, S.로그인
```


여러 릴레이션에 대한 질의

- 다음 질의의 결과는?

```
SELECT S.이름, E.과목번호
FROM 학생 S, 수강 E
WHERE S.학번=E.학번 AND E.평점="A"
```

수강의 인스턴스가 다음과 같을 때(DBMS 가 참조 무결성을 보장할 때에도 이러한 모양이 가능할까?):

학번	과목번호	평점
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

다음을 얻는다:

S.이름	E.과목번호
Smith	Topology112

SQL로 릴레이션을 생성하는 법

- 학생 릴레이션을 생성해 보자. 각 필드마다 타입(도메인)을 지정해 주며, 이 사항은 투플이 추가되거나 수정될 때 DBMS가 집행해 준다.
- 수강 테이블에는 학생들이 수강하는 과목들에 관한 정보가 들어간다.

```
CREATE TABLE 학생
    (학번 CHAR(20),
     이름 CHAR(20),
     로그인 CHAR(10),
     나이 INTEGER,
     평점평균 REAL)

CREATE TABLE 수강
    (학번 CHAR(20),
     과목번호 CHAR(20),
     평점 CHAR(2))
```

릴레이션의 제거와 변경

DROP TABLE 학생

- 학생 릴레이션을 제거한다. 스키마 정보와 투플들이 모두 삭제된다.

ALTER TABLE 학생
ADD COLUMN 입학년도 integer

- 학생 테이블의 스키마는 새로운 필드가 추가됨으로써 바뀐다. 현 인스턴스의 투플들은 해당 필드를 널 값으로 하여 확장된다.

투플의 삽입과 삭제

- 한 투플 삭제

```
INSERT INTO 학생(학번, 이름, 로그인, 나이, 평점평균)
VALUES (53688, 'Smith', 'smith@ee', 18, 3.2)
```

- 특정 조건(e.g., 이름 = Smith): 을 만족하는 모든 투플 삭제

```
DELETE
FROM 학생 S
WHERE S.이름 = 'Smith'
```

☛ 이 명령들을 여러가지로 변형하여 막강한 표현이 가능한데, 뒤에서 설명함!

무결성 제약조건 (IC)

- IC: 데이타베이스가 어떤 인스턴스가 되든 꼭 지켜 주어야 될 조건; e.g., 도메인 제약조건
  - IC는 스키마를 정할 때 명세한다.
  - IC는 릴레이션들이 수정될 때 체크한다.
  - DBMS는 적법한 인스턴스만 허용한다.
- 어떤 릴레이션에서, 명세된 IC를 모두 만족하는 인스턴스를 적법한(legal) 인스턴스라고 한다.
- DBMS가 IC들을 체크하게 되면, 실세계의 의미에 더 충실한 데이타로 만들 수 있다.
  - 물론 데이타 입력 오류도 방지한다!

기본 키 제약조건(Primary Key Constraints)

- 어떤 릴레이션의 키(key)란, 다음을 만족하는 필드집합이다:
  1. 키 필드의 값이 모두 같은 투플은(둘 이상) 있을 수 없고,
  2. 키의 부분집합을 취하면 그렇지 않다.
  - 2번 부분이 성립 안하면? 슈퍼키(superkey)
  - 한 릴레이션에 키가 여럿 존재하는 경우에는, 그 중에서 하나를 (DBA가) 기본 키(primary key)로 정한다.
- 예) 학번은 학생 테이블의 키이다. (이름은?)
  집합{학번,평점평균}은 슈퍼키이다.
  ⇔ 개체 무결성(entity integrity): 기본 키는 유일하여야 하고, 널이 될 수 없다.

SQL에서 기본 키와 후보 키 표현법

- 후보 키(candidate key)는 여러개일 수 있고(UNIQUE로 표현), 그중 하나를 기본 키로 선정
- "주어진 학생과 과목에 대해서 평점은 하나다" 대 "학생들은 한 과목씩만 수강할 수 있으며 평점도 하나만 받는다. 또, 한 과목에서 같은 평점을 두 학생이 받을 수는 없다"
- IC를 잘못 명세하면 현실에 맞는 데이타베이스 인스턴스를 만들지 못하는 경우가 발생한다!

```
CREATE TABLE 수강
  (학번 CHAR(20),
   과목번호 CHAR(20),
   평점 CHAR(2),
   PRIMARY KEY (학번, 과목번호) )
CREATE TABLE 수강
  (학번 CHAR(20),
   과목번호 CHAR(20),
   평점 CHAR(2),
   PRIMARY KEY (학번),
   UNIQUE (과목번호, 평점) )
```


외래 키, 참조 무결성

- 외래키(Foreign key): (다른)릴레이션의 어떤 투플을 '참조'할 목적으로 가지고 있는, 릴레이션의 어떤 필드집합 (대개 피참조 릴레이션의 기본 키에 대응한다).'논리적인 포인터'나 마찬가지.
- E.g. 학번은 학생 테이블을 참조하는 외래 키이다:
  - 수강(학번: 문자열, 과목번호: 문자열, 평점: 문자열))
  - 외래 키 제약조건을 모두 만족할 때(없는 투플을 참조하는 경우가 없을 때), 참조 무결성(referential integrity)을 만족한다고 말한다.
  - 참조 무결성이 없는 데이터 모델이 있을까?
    * HTML의 링크!

SQL에서 외래 키 표현법

- 학생 테이블에 등록된 학생들만 과목 수강이 가능하다.

```
CREATE TABLE 수강
  (학번 CHAR(20), 과목번호CHAR(20), 평점 CHAR(2),
   PRIMARY KEY (학번,과목번호),
   FOREIGN KEY (학번) REFERENCES 학생 )
```

수강

학번	과목번호	평점
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

학생

학번	이름	로그인	나이	평점평균
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ecs	18	3.2
53650	Smith	smith@math	19	3.8

참조 무결성 집행

- 수강 테이블의 학번 필드는 학생 테이블을 참조하는 외래 키이다.
- 존재하지 않는 학번을 가진 수강 투플을 삽입할 때에는? (거부한다!)
- 학생 투플을 삭제할 때에는?
  - 그 투플을 참조하는 수강 투플들도 같이 삭제한다.
  - 참조되고 있는 학생 투플은 삭제를 못하도록 한다.
  - 그 투플을 참조하는 수강 투플의 학번값을 내정값으로 설정한다
  - (SQL에서는 참조측을 널(null)로 만들 수도 있다. 널은 '미상' 또는 '적용불가능'을 나타낸다.)
- 학생 투플의 기본 키를 갱신해도 비슷하다.

SQL/92에서 참조 무결성 표현법

- SQL/92 는 삭제 및 갱신에 대해 총 4개의 옵션을 제공한다.
  - 묵시적으로는 NO ACTION (삭제/갱신을 거부)
  - CASCADE (참조측도 같이 삭제)
  - SET NULL / SET DEFAULT (참조측 값을 내정값이나 널로 설정)

```
CREATE TABLE 수강
  (학번 CHAR(20),
   과목번호 CHAR(20),
   평점 CHAR(2),
   PRIMARY KEY (학번,과목번호),
   FOREIGN KEY (학번)
     REFERENCES 학생
        ON DELETE CASCADE
        ON UPDATE SET DEFAULT )
```

IC의 연원은?

- IC는 원래 데이타베이스(의 릴레이션들)로 표현하고자 하는 실제 조직체의 내용으로부터 도출되는 것이다.
- 데이타베이스 인스턴스가 주어진 IC를 위배하는지는 점검이 가능해도, 인스턴스를 관찰하여 주어진 IC가 타당함을 알 수는 없다.
  - IC는 가능한 모든 인스턴스에 대한 진술이기 때문이다!
  - 예) 이름 필드는 키가 아님을 알 수 있지만, 학번 필드가 키라는 것은 주어진 인스턴스만으로는 확인이 불가능하다.
- 키 IC와 외래 키 IC는 가장 흔한 IC이지만, 더 일반적인 IC들도 있다.

뷰(View)

- 뷰는 일종의 릴레이션이지만, 실제 투플 집합이 아니라 정의를 저장하는 것이다.

```
CREATE VIEW YoungActiveStudents (이름, 평점)
    AS SELECT S.이름, E.평점
    FROM 학생 S, 수강 E
    WHERE S.학번 = E.학번 and S.나이<21
```

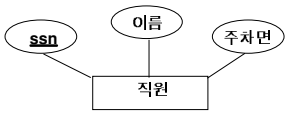
- 뷰를 제거할 때에는 DROP VIEW 명령을 사용한다.
  * 해당 테이블에서 유도된 뷰가 있을 때, DROP TABLE 명령이 오면?
    - DROP TABLE 명령에는 사용자가 이 부분을 명세할 옵션 부분이 있다.

뷰와 보안

- 뷰를 사용해서 필요한 정보(또는 요약본)만 보여주고, 기반 릴레이션(들)의 세부내용은 감출 수 있다
  - YoungActiveStudents만 주고 학생 및 수강 테이블을 감추면, 수강중인 학생은 알 수 있지만 그 수강 과목의 번호는 알 수 없다.

논리적 DB 설계: ER을 관계 모델로

- 개체집합을 테이블로.




```
CREATE TABLE 직원
      (ssn CHAR(11),
       이름 CHAR(20),
       주차면 INTEGER,
       PRIMARY KEY (ssn))
```

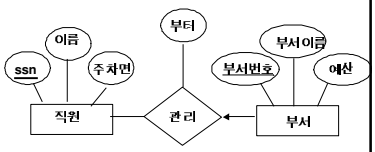

관계집합을 테이블로

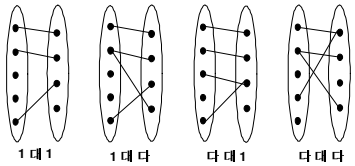
- 관계집합을 테이블로 변환할 때에는 다음 애트리뷰트들을 넣어 주어야 한다:
  - 각 참여 개체집합의 키(외래키로).
    - * 이것들이 결과 릴레이션의 슈퍼키를 형성한다.
  - 설명용 애트리뷰트 모두.

```
CREATE TABLE 근무(
  ssn CHAR(11),
  부서번호 INTEGER,
  부터 DATE,
  PRIMARY KEY (ssn, 부서번호),
  FOREIGN KEY (ssn)
     REFERENCES 직원,
  FOREIGN KEY (부서번호)
     REFERENCES 부서)
```


Review: 키 제약조건

- "관리"에는 키 제약조건이 있으므로 한 부서에 많아야 한 명의 부서장이 있다.





관계 모델로 변환하려면?

키 제약조건이 있는 ER 다이어그램 변환법

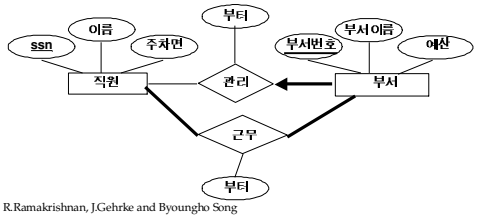
- 관계를 테이블로 사상:
  - 지금은 부서번호가 키!
  - 직원, 부서 테이블과 분리.
- 각 부서마다 고유한 부서장이 있으므로, "관리"와 "부서"를 합칠 수도 있다.

```
CREATE TABLE 관리(
  ssn CHAR(11),
  부서번호 INTEGER,
  부터 DATE,
  PRIMARY KEY (부서번호),
  FOREIGN KEY (ssn) REFERENCES 직원,
  FOREIGN KEY (부서번호) REFERENCES 부서)
```

```
CREATE TABLE 부서_부서장(
  부서번호 INTEGER,
  부서이름 CHAR(20),
  예산 REAL,
  ssn CHAR(11),
  부터 DATE,
  PRIMARY KEY (부서번호),
  FOREIGN KEY (ssn) REFERENCES 직원)
```


Review: 참여 제약조건

- 모든 부서에 부서장이 있는가?
  - 그렇다면 참여 제약조건이 된다: "관리"에 대한 "부서"의 참여도를 전체적이라고 한다. (부분적도 있다)
    - 부서 테이블의 어떤 "부서번호"값도 빠짐없이 관리 테이블에 한 행으로 나타나야 한다.(이때 ssn은 널이 될 수 없다)



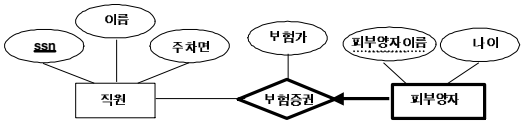
SQL에서 참여 제약조건 표현법

- 개체집합 하나를 이진 관계에 포함시키되 약간 다르게 하면, 참여 제약조건을 표현할 수 있다.(CHECK제약조건에 의지하지 않을 때)

```
CREATE TABLE 부서_부서장(
  부서번호 INTEGER,
  부서이름 CHAR(20),
  예산 REAL,
  ssn CHAR(11) NOT NULL,
  부터 DATE,
  PRIMARY KEY (부서번호),
  FOREIGN KEY (ssn) REFERENCES 직원
    ON DELETE NO ACTION)
```

Review: 약 개체

- 약 개체는 다른(소유자) 개체의 기본 키까지 보아야 유일하게 식별할 수 있다.
  - 소유자 개체집합과 약 개체집합은 1대다 관계집합으로 연결된다. (소유자 하나에 여러 개의 약 개체)
  - 약 개체집합은 이 식별 (identifying) 관계집합에 전체적으로 참여한다.

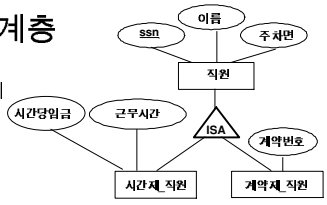


약 객체집합 변환법

- 약 개체집합과 식별 관계집합을 한 테이블로 변환한다.
  - 소유자 개체가 삭제되면, 모든 해당 피소유 약 개체들도 같이 삭제해 주어야 한다.

```
CREATE TABLE 피부양자_증권 (
  피부양자이름 CHAR(20),
  나이 INTEGER,
  보험가 REAL,
  ssn CHAR(11) NOT NULL,
  PRIMARY KEY (피부양자이름, ssn),
  FOREIGN KEY (ssn) REFERENCES 직원
    ON DELETE CASCADE)
```


Review: ISA 계층



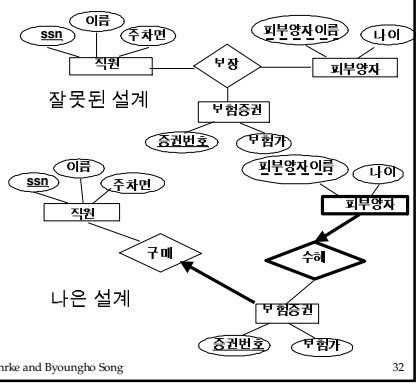
- C++처럼 애트리뷰트들이 계승된다.
- A ISA B 라고 선언하면, A개체는 모두 B개체로도 볼 수 있다.
- 중첩(overlap)제약조건: Joe가 시간_직원이면서 계약제_직원도 될 수 있는가? (예/아니오)
- 포괄(covering)제약조건: 직원들은 모두 시간제_직원 아니면 계약제_직원인가? (예/아니오)

ISA 계층 변환 법

- 보통 방식:
  - 3 릴레이션 3개: 직원, 시간제_직원, 계약제_직원
    - 시간제_직원: 직원들은 모두 일단 "직원" 릴레이션에 등록한다. 시간제_직원에 있는 가외의 정보들은 시간제_직원(시간당임금, 근무시간, ssn)에 기록한다. "직원" 투플이 삭제되면 그 투플을 참조하던 "시간제_직원" 투플들도 함께 삭제해 주어야 한다.
    - 전직원에 대한 질의는 쉽다. 시간제 직원에 국한된 질의는 죠인이 필요하다.
- 다른방식 : **시간제_직원 및 계약제_직원만 생성**
  - 시간제_직원: ssn, 이름, 주차면, 시간당임금, 근무시간.
  - 이 두 부류에 속하지 않는 직원은 존재할 수 없다.

2진관계냐 3진관계냐

- 한 증권은 한 직원만 가질 수 있다면 :
  - "보험증권"에 키제약조건을 붙이면증권이 피부양자 한명만 보장하게 됨!
- 두 번째 다이어그램에 추가된 제약조건은?



잘못된 설계

나은 설계

2진 관계냐 3진관계냐 (계속)

- 키제약조건이 있기 때문에 "구매"와 "보험증권", "수혜"와 "피부양자"를 합칠 수 있다.

```
CREATE TABLE 보험증권 (
  증권번호 INTEGER,
  보험가 REAL,
  ssn CHAR(11) NOT NULL,
  PRIMARY KEY (증권번호),
  FOREIGN KEY (ssn) REFERENCES 직원
    ON DELETE CASCADE)
```

- 참여 제약조건이 있기 때문에 제약조건을 붙인다.
- "보험증권"이 약개체집합이라면?

```
CREATE TABLE 피부양자 (
  피부양자이름 CHAR(20),
  나이 INTEGER,
  증권번호 INTEGER,
  PRIMARY KEY (피부양자이름, 증권번호),
  FOREIGN KEY (증권번호) REFERENCES
      보험증권 ON DELETE CASCADE)
```

관계 모델: 요약

- 데이타를 테이블 형태로 표현.
- 단순하며 직관적이어서 현재 가장 널리 사용됨.
- 응용의 내용에 따라 DBA가 무결성 제약조건을 부여할 수 있다. DBMS가 위배여부를 체크한다.
  - 중요한 IC : 기본 키와 외래 키
  - 도메인 제약조건도 언제나 존재한다.
- 강력하고 자연스러운 질의어.
- ER 모델을 관계 모델로 변환하는 규칙 존재.