6장 데이터 베이스 시스템 주요 기능

View

사용자로부터 데이터 베이스 스키마 일부를 숨긴다.

```
create view myClub as {query}
```

뷰는 한개 이상의 테이블로부터 정의할 수 있으며, 해당 내용을 실제로 저장하지 않기 때문에 **virtual relation/table**이라고도 한다.

뷰는 항상 최신 데이터를 가지고있다.

뷰 정의시 다른 뷰를 사용할 수 있다.

자신의 뷰를 사용한 것은 순환뷰라고 한다.

뷰 질의가 들어오면, 뷰 정의로 치환 후 리턴하는 형식이다. 만약 뷰를 참조했다면 해당 뷰도 참조하고 참조하고 해서 최종적으로 베이스 테이블까지 접근한다.

뷰 입력, 삭제, 업데이트 가능하다.

- 안되는 경우

1. 모호한 경우 (A와 B테이블 모두 id 속성을 가짐)
2. 집계함수 사용시 (avg, sum..)
3. group by, having, distinct, orderby, 집합연산 등

- 뷰 제약

1. 뷰는 실제 터플이 없기 때문에 index 불가
2. 뷰에 대해 키 속성 또는 무결성 제약 정의 불가

Intergrity Constraints

데이터베이스 시스템이 항상 만족해야할 조건
계좌 잔고는 무조건 100원 이상, 시급은 최저 600원부터.. 이런 제약 형식

- 단일 테이블 제약 not null
  주키
  unique: 중복값 없으면 null 허용
  check 옵션

```
id char(3) not null
primary key (id)
check (id in ('1','2','3'))
```

- 참조 무결성 제약 외래키와 주키 관계

```
foreign key(id) references person,
    on delete cascade,
    on update cascade
```

참조무결성 위반시 제약사항

1. cascade 삭제시 해당 터플을 참조하는 다른 테이블의 터플도 삭제한다.
   수정시 다른 테이블도 같이 수정
2. set null 삭제시 참조터플 null
3. set default

Trigger

데이터 수정이 일어나면 side effect 자동 수행

- ECA 규칙 사건 / 조건 / 행동으로 구성되며
  어떤 사건이 발생하면 조건을 평가하여 행동한다.
  사건 - 삽입/삭제/수정을 말함

Trigger 예제

1. 학생이 수업을 수강하여 학점을 취득하면 학생이 취득한 총 학점을 변경하는 트리거

```
Create trigger myCred after update of grade
-- 업데이트 발생 시 업데이트 전과 후 참조
on takes referencing new row as nrow
referenceing new row as srow
referencing old row as orow
-- 값이 변경된 터플에 한해서만 조건을 확인한 후 행동한다
for each row
-- 변경 후의 학점이 F가 아니고 null도 아니면서, 변경 전 학점이 F이거나 null인 경우
when nrow.grade <> 'F' and nrow.grade is not null
and (orow.grade = 'F' or orow.grade is null)
-- 수강 과목의 학점을 학생 학점에 추가하는 형식
begin
    Update student
    set totalCredit = totalCredit +
        (select credit
        from course
        where cID = nrow.cID)
    where sID = nrow.sID;
end;
```

2. 사용자의 계좌 잔고보다 더 많은 금액의 수표가 발생되는 경우 대출 계좌를 사용자 명의로 개설하고 초과된 만큼 대출금 설정

```
Create trigger myOverdraft after update on account
referenceing new row as srow
referencing old row as orow
for each row
-- 현재 계좌 잔고가 마이너스이면
when nrow.balance<0
-- atomic: 아예 실행하지 않거나 쿼리 3개 전부 실행한다
begin atomic
-- 조건에 맞는 터플과 은행 계좌가 동일한 예금주의 정보를 대출자 테이블에 넣기
    insert into borrower
    (select cName, aNumber from depositor
    where nrow.aNumber = depositor.aNumber);
-- 통장에서 마이너스 된 만큼 빚을 설정한다
    insert into loan values (nrow.aNumber, -nrow.balance);
-- 마이너스가 빚으로 처리되었으니 현재 잔고를 0으로 설정한다
    Update account set balance = 0;
        where account.aNumber = nrow.aNumber;
end
```

3. employee 관계의 salary 속성에 변경이 발생하면, department의 totalsalary를 변경한다.

```
Create trigger myTotalSalary after update of salary on employee
referenceing new row as nrow
referencing old row as orow
for each row
-- 봉급 바뀐 사람의 현재 봉급이 null이 아니면
when(nrow.dNumber is not null)
Update department
-- 부서의 총 봉급에 예전 봉급 빼고 새 봉급을 더한다
    set totalSalary = totlaSalary + nrow.salary-orow.salary
    where dno = nrow.dNumber;
```

4. 트리거는 이벤트 전에 수행될 수 있다.

```
Create trigger mySetNull before update on takes
referencing new row as nrow
for each row
when(nrow.grade ='')
Update takes set nrow.grade = null;
```

테이블 갱신 전, grade에 값은 존재하지만 해당 값이 공백인 경우 null로 지정해준다.

5. 문장 수준 트리거 employee 관계의 salary 속성에 변경이 발생하면, department의 totalsalary를 변경한다.

```

```

```
Create trigger myTotalSalaryState after update of salary on employee
referenceing new table as N
referencing old table as O
-- 문장 단위로 수행
for each statement
-- 봉급 바뀐 사람의 전/후 봉급이 하나라도 null이 아닌 경우
when exists
    (select * from N where N.dnumber is not null)
    or (select * from O where O.dnumber is not null)
-- 부서 총봉급 수정
Update department as D
set D.totalSalary = D.totalSalary
    + (select sum(N.salary) from N where D.dno = N.dnumber)
    - (select sum(O.salary) from O where D.dno = O.dnumber)
where D.dno in ((select dnumber from N)
                union
                (select dnumber from O));
```

테이블 단위로 관리하고 계산한다.

Trigger 사용 comment

1. 객체지향 언어에서는 트리거를 구현하지 않고 메소드 형식으로 캡슐화 할 수도 있다.
2. cascade(전파) 주의 요망
3. 트리거는 속성의 통계 정보를 유지하거나 임의 테이블의 복사본을 유지할때 사용하였으나, 요즘은 트리거 대신 뷰를 제공하며 테이블 복제 기능도 제공한다.

Authorization

- 권한

1. instance read, insert, update, delete
2. schema index(create, delete), resource(create new relation), alter, drop

- SQL 언어로 부여할 수 있는 권한
  select, insert, update, delete, reference, usuage, all privillege
  refenrence - 참조할 권한 usuage - 도메인 쓸 수 있는 권한

- 권한 부여

```
Grant <권한 list> on (table/view) to <user list> [with grant option]
```

with grant option이용시 권한 부여받은 사람이 다시 권한을 부여할 수 있다.

- 권한 철회

```
Revoke <권한 list> on (table/view) from to <user list> [restricted | cascade]
```

cascade: 해당 사용자가 다른 사용자에게까지 권한 부여했다면 그 사용자의 권한까지 철회
restricted: cascade 발생시 권한 철회하지 않음

- 권한 부여하는 권한 철회 가능

```
Revoke grant option on ...
```

- 권한 그래프 모든 권한의 뿌리는 DBA

뷰 권한

뷰는 최소한 베이스 테이블에 읽기 권한이 있어야 생성 가능하다
뷰 권한은 베이스 테이블 권한을 넘어서지 못한다. 뷰 생성자는 테이블 생성자와 다르게 resource 권한을 가지지 않아도 된다. 일반 테이블 생성자는 테이블에 대한 모든 권한을 가지지만 뷰 생성자는 아니다. 뷰를 생성했기 때문에 select 권한은 무조건 가지고, 나머지 권한은 베이스 테이블 권한에 따른다.

뷰 권한 예제

1. 교수자가 2015 가을 학기에 강의하는 과목 접근은 가능하지만 교수 봉급은 접근하지 못한다.

```
Create view myTeach as
select name, title
from professor, teaches, course
where teaches.pID = professor.pID and course.cID = teaches.cID
and semester = 'Fall' and year=2015;
```

- 뷰 사용자로부터 professor의 salary를 숨겨 접근하지 못하게 한다.

2. 스태프가 professor 테이블에 대한 읽기 권한이 없어도 뷰에 대한 읽기 권한만 가지고 CSProfessor 뷰를 접근할 수 있다.

```
user)
Create view CSProfessor as
(select * from professor where deptName = 'CS');
Grant select on CSProfessor to staff;
staff)
Select * from CSProfessor;
```

staff가 professor에 대한 권한이 없어도 CSProfessor를 통해 professor 테이블에 접근할 수 있는 형식이다.

Role

사용자의 집합
Role에게 권한을 부여할 수 있다.

```
create role teller;
grant select on branch to teller;
grant update(balance) on account to teller;
-- role 만들어서 권한 부여

create role manager;
grant teller to manager;
-- //role의 권한 또다른 role에 그대로 부여
```

- 권한 한계

1. tuple 단위로 권한 부여 불가
2. 앱은 가능하다

Recursive Queries

자신의 뷰로 또다른 뷰 만들기
선/후수 과목에 사용한다

```
With recursive recPrereq(courseID, prereqID)
as (select courseID, prereqID from prereq)
union
(select recPrereq.courseID, prereq.prereqID
from recPrereq, prereq
where recPrereq.prereqID = prereq.courseID)

select * from recPrereq;
```

$$(\forall x)(\forall y)\ (recPrereq(x,y) \leftarrow prereq(x,y))$$
$$(\forall x)(\forall y)(\forall z)\ (recPrereq(x,y) \leftarrow recPrereq(x,z) \wedge prereq(z,y))$$

6장 연습문제

6.1

Using the relations of the second running schema (bank schema), write SQL expressions to define the following views. Also determine if the view is updatable. Justify your answer.

(a) A view containing the account numbers and customer names for all accounts at the 'Sangdo-dong' branch.

```
Create view myView as
(select a.accountNumber, d.customerName
from account a, depositor d
where a.branchName = 'sangdo-dong' and d.accountNumber = a.accountNumber)
```

업데이트 가능하다. 제약조건에 걸리는 거 없음.

(b) A view containing the names and addresses of all customers who have an account with the bank, but do not have a loan.

```
create view myview as
(select c.customerName, c.customerCity
from (customer natural join depositor)
except
(customer natural join depositor natural join borrower))
```

업데이트 하지 못한다.
depositor의 primary key인 accountNumber에 대해 접근할 수 없기 때문이다.

(c) A view containing the name and average account balance of every customer of 'Sangdo-dong' branch.

```
Create view myView as
(select customername, avg(balance)
from account natural join depositor
where branchName = '상도동'
group by customerName);
```

업데이트 하지 못한다.
집계 연산을 사용하고 있기 때문에..
또한 group by절을 사용하고 있음

6.2.

SQL allows a foreign key dependency to refer to the same relations, as in the following example:

```
Create table myManager (
employeeName – char(20) not null,
managerName – char(20) not null,
primary key employeeName,
foreign key (managerName) references myManager on delete cascade);
```

- Explain exactly what would happen when a tuple in "myManager'" is deleted?

myManager를 참조하고 있는 tuple도 연속적으로 삭제된다.

6.3.

Consider the relational schema

```
part(partID *, name, cost)
subpart(partID *, subpartID *, count)
```

- A tuple (p1, p2, 3) in the “subpart” relation denotes that the part with partID p2 is a direct subpart of part with partID p1, and p1 has 3 copies of p2. Note that p2 may itself have further subparts. The underlined attributes denote the primary key, respectively. **Write a recursive SQL query that outputs the names of all subparts of the part with partID P-100.**

```
With recursive rec(partID, subpartID) as
(select partID, subpartID from subpart)
union
(select rec.partID, subpart.subpartID
from rec, subpart
where rec.subpartID = subpart.partID)

select name
from rec, part
where rec.partID = 'P100'
and rec.subpartID = part.partID
```

재귀 돌려서 모든 선수과목 찾기

6.4.

Consider the relational schema

```
manages(eName, mName)
```

where eName denotes an employee and mName denotes a manager name.
Define a relation

```
employeeDepth(eName, mName, depth)
```

where "depth" indicates how many levels of intermediate managers are there between the employee and the manager. Employees who are directly under a manager would have a depth of 0.

```
with recursive employeedepth(ename, mname, depth) as
(select ename, mname, 0
from manages)
union
(select m.ename, e.mname, depth+1
from manages m, employeedepth e
where m.mname = e.ename);
```

```
select * from employee depth;
```

6.5.

Suppose user A, who has all authorization on a relation R, grants select on relation R to public with grant option. Suppose user B then grants select on R to A. Does this cause a cycle in the authorization graph?

A: R의 모든 권한

B: R의 select권한, 권한 부여 권한
A: R의 모든 권한 + B로받은 Select 권한

--> select 권한에 대해 사이클 생성됨

7장 오라클 실습

- LOB 데이터 타입 대용량 데이터를 저장/관리하기 위해 사용
  BLOB, CLOB, NCLOB, BFILE
- 집합 연산 오라클에서는 union all만 지원

여러 함수..

- 문자 처리 lower()
  Instr()
  Replace()
  Substr()
- 숫자형 처리 Round: 반올림
  Trunc: 절삭
  Sign: 양/음/0
- 날자형 처리 함수 sysdate
- 형 변환 TO_CHAR
  TO_NUMBER

뷰

```
Create [or replace] [force|noforce] view view_name
as subquery
[with check option]
[with read only]
```

force: 베이스 테이블 없어도 생성
with read only: 뷰 검색만 가능하고, 뷰를 통한 베이스테이블 변경 불가

```
Create view myview as
select * from professor where deptName='SW’'
with check option;
```

deptname이 SW가 아닌 경우 insert 불가

- 실체화된 뷰 (materialize view) 터플을 가질 수 있는 뷰
  - 빠른 질의 응답
  - base table의 최신 반영 어려움(옵션으로 갱신)
  - 베이스가 바뀌면 view는 old data
  - up-to-date 필요없는 뷰에 사용

ROWNUM

오라클이 지원하는 기능, ranking과 비슷함.
숫자 1부터 시작하는 pseudo-column

1. where clause를 통과한 튜플에 대해 부여된다.
2. 특정 튜플에 배정된 이후 그 값이 하나 증가한다

```
-- 무작위 5명
select * from student where rownum <=5;
-- 무작위 5명
select * from student where rownum<=5 order by gpa;
-- 상위 5
select *
from (select * from student order by gpa desc)
where rownum<=5;
-- 아무것도 리턴하지 않음
select * from student where rownum>1;
```

7장 연습문제

- Make each of the below
  - A trigger (with "referencing" and “when")
  - An updatable view
  - A view with check option
- Run at least 10 SQL statements including
  - nested subqueries (use some/any/all)
  - correlated subqueries
  - ranking queries
  - SQLs that show the effect of the trigger, view, view with check option

8장 응용 개발

DB 어플리케이션

- static approach
  SQL을 프로그램에 임베딩하는 형태
- dynamic approache API 호출 후 함수 호출

8.1 Embeded SQL

1. 프로그래밍 언어 안에 SQL을 임베딩
2. preprocessor가 임베디드 sql을 api 호출로 변경한다

Cursor

프로그래밍 언어와 DB언어 SQL은 방식이 달라서 맞춰야 한다.
일반 프로그램 변수에는 집합을 지원하지 않는다.

--> c++ STL에서 컨테이너 타입으로 set, multiset 지원한다.

1. 커서 declare
2. open: db에 질의 수행
3. 질의 결과를 임시 테이블에 저장
4. fetch를 통해 임시 테이블에서 터플을 하나씩 프로그램으로 전달한다.
5. SQLCA=02000
6. 커서 close

- SQLCA = 00000: 성공 / 02000: 더이상 검색되는 터플 없음

```
-- 1. declare
EXEC SQL DECLARE demoCursor cursor for
select first, last
where :lname < 'C';

-- 2. open --> 쿼리 수행
EXEC SQL OPEN democursor;

for(;;){
    -- 3. fetch로 튜플 단위로 불러오기
    EXEC SQL fetch democursor into :fname, :lname;
    if (strcmp(SQLSTATE, "00000", 5)!=0) break;
    printf("%s %s\n",fname, lname);
}
-- 더이상 불러올 값이 없으면
if(strcmp(SQLSTATE, "02", 2)!=0)
    printf("SQLSTATE after fetch is %S\n", SQLSTATE);
-- 4. close
EXEC SQL close democursor;
```

- 커서를 통한 데이터 갱신 for update 옵션을 이용해서 커서로 DB 업데이트 가능

동적 SQL

프로그램 실행 시간에 SQL 문장이 생성된다.
prepare, execute 과정이 필요하다
프로그램과 DB 서버를 연결하는 API: **ODBC, JDBC**

정적 SQL

DB의 전처리 단계에 문장 구문 검사, 권한 검사, 질의 실행 코드 생성 등 가능하다

동적이 정적보다 늦지만, 사용자에게 질의문 작성에 대한 융통성을 제공한다.

8.2 ODBC, JDBC

1. SQLEXECDIRECT
2. SQLFETCH
3. SQLBINDCOL

- 기본적으로 동적 SQL을 지원한다.

```
SQLstatement = "SELECT * FROM users WHERE name = '"+username+"';"
```

- JDBC: 자바에서 제공하는 ODBC

- static 임베이드 SQL, SQLJ
  전처리가 필요하며 전처리 과정에서 구문검사, 권한 검사 등 가능하며 신속하게 수행할 수 있다.
  실행 코드 작성시 두단계 거쳐야함

- dynamic API call
  ODBC, JDBC
  임베디드 SQL도 전처리기에 의해 API call로 변환됨

8.3 응용 구조

Two-tier architecture	**Three-tier architecture**
client	client
users applications	users application clients
network	network
server	server
database sys	application server database sys

- cookies HTTP의 Stateless Connection이라는 Web의 특징을 보완하기 위해 사용한다.
  쿠키로 사용자의 행동을 저장하는 형식

8장 연습문제

8.1

- Why servlets give better performance than CGI programs, even though Java programs generally run slower than C/C++ programs?

CGI는 프로세스마다 실행되어야 해서 오버헤드가 크지만, 자바의 서블렛은 스레드를 실행해 적다.

8.2

- List some benefits and drawbacks of connectionless protocols over the protocols that maintain connections.

서버에 사용자의 상태를 저장하지 않기 때문에 서버 과부하가 줄어든다.
하지만 사용자가 접근할때마다 새롭게 연결해야하는 어려움이 있다.

9장 SQL 확장

상용 DB sys에서 SQL 확장 지원한다.
Oracle - PL/SQL

외부 언어 프로시저

```
create procedure deptcountproc
    (in deptname varchar(20),
    out count integer)
language C
external name '/usr/...';

create procedure deptcountproc2
(deptname varchar(20))
returns integer
language C
external name '/ussr/...';
```

외부 언어를 사용하는 예시이다.
external name은 C 프로그램의 목적 코드

이렇게 외부 언어로 함수/프로시저 개발하는 것의 문제점..

- 데이터베이스 보안 이슈
- 함수와 프로시저를 DBMS영역에 불러와 실행하게 되므로 오류에 취약
- DBMS 서버가 사용자에 의해 스탑되지 않아야함

- 사용자 프로그램 오류가 DB의 오류로 될 수 있음
  장점
- 효율적
- 전체적인 성능 향상

보안 이슈 없애기

- 안전 언어 (자바, c#) 사용
- 샌드박스 - 외부언어가 자신의 메모리에 접근하는 것은 허용하지만 실행 프로세스 메모리나 파일 세스템은 불가 -> 안전함
- IPC 이용

Stored procedure

프로그램 모듈을 저장해서 DB 필요할때 호출
-> 하나의 오브젝트로 DBMS 저장

- 캡슐화
- call만 하면 되기에 별다른 네트워킹 감소

SQL 함수

```
create function profc(deptname varchar(20)) returns integer
begin
    declare pcount integer;
    select count(*) into pcount
    from professor
    where professor.deptname = profc.deptname
    return pcount;
end;

select deptname, budget
from department
where profc(deptname) >5;
```

입력으로 주어지는 학과의 교수 수를 반환하는 함수와 호출하는 예시

SQL 프로시저

SQL 함수와는 다르게 인자에 대해 입력 및 출력 명시, 출력 인자가 2개 이상이어도 가능 (함수는 불가)
오버로딩 가능

- 프로시저 생성자 while, loop, repeat, 조건문 여러 문법 가능
  예외 처리 가능

Function Exercise

정원 내에서 학생을 과목에 수강 신청하는 연산 수행하는 함수

```
create function myregistersut(
    mysid char(5),
    mycid char(5),
    mysemester char(10),
    myyear numeric(4,0)
)
returns integer
begin
    declare currenrolment int;
    select count(*) into currenrolment from takes
    where cid = mycid
    and semester = mysemester
    and year = myyear;

    declare limit int;
    select capacity into limit from room, teaches
    where classroom=roomid
    and cid=mycid
    and semester=mysemester
    and year=myyear;

    if(currenrolment<limit)
    begin
        insert into takes values (mysid, mycid, mysemester, myyear, null);
        return (0);
    end
    return (-1);
end
```

9.3 PL/SQL

[declare] / begin / [exception] / end; 으로 이루어져있는 블록이다.

변수 선언, 조건 로직, 루프, 예외처리 등 가능
커서, 프로시저, 함수, 패키지 등 가능

10장

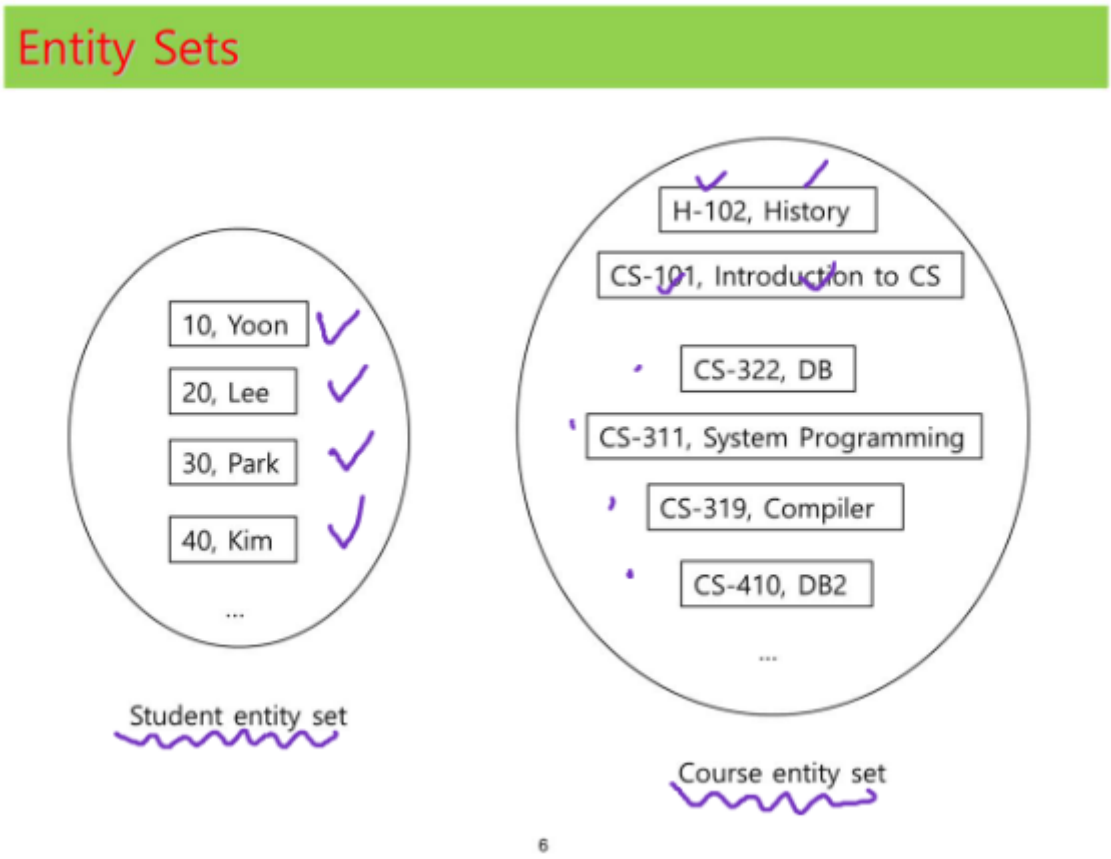
Entity

ER(Entity-Relation) DB 설계할때 쓰는 모델링
데이터베이스를 개체와 관계로 모델링한다.

- 구분할 수 있는 엔티티
- 다 된다 (person, company, student.. 등등)
- Entity have attribute people have names and address
- entity set/type is a set of entities with the same properties

Entity Sets

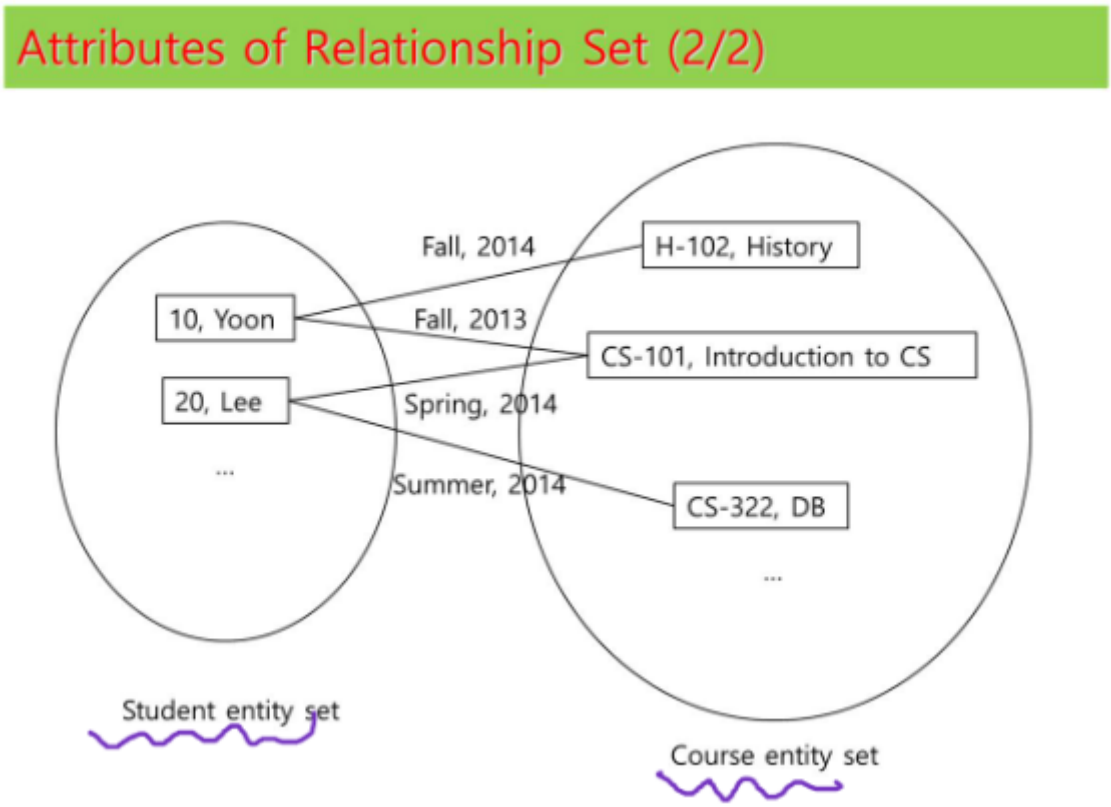
동일한 속성을 가진 Entity의 집합



Relationship

entity가 관련있으면.. ex) Student s1 related to course c2

- 속성을 가질 수 있음 어느 학기의 어느 년도?

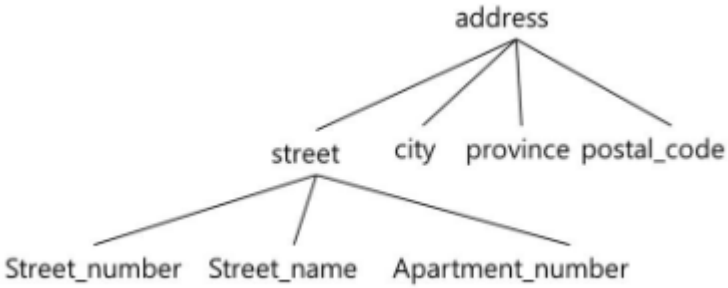


- degree of a relationship set 관련되는 entity몇개? ternary/quteray/...

Attribute

- 어떤 entity는 속성의 집합으로 나타남 professor = {professorID, name, deptname, salary}
- derivea attribute 생년월일 입력받으면 나이까지 알 수 있음

Composite Attribute Example



Cardinality Constraints

- 1:1
- 1:n
- n:1
- 1:n, n:1

keys

super key.. 등등 예전에 배운 것과 동일함

- relationship type에 대한..

E-R Diagram

ER data 모델링의 결과치...
표기를 어떻게 할 것인지?
-> ER Diagram

- Rectangles represent entity sets ✓
- Diamonds represent relationship sets ✓
- Attributes are listed inside entity rectangle
- Underline indicates primary key attributes

- 네모: entity set
- 다이아몬드: relationship set (두 네모 사이 relation)
- 네모 밑: 속성들
- 밑줄: primary key
- (속성): 하나 이상이어도 된다는 뜻 -> composite attribute
- 속성(): derived attribute

relationship(다이아몬드)도 attribute를 가질 수 있음

Cardinality Constraints

화살표 존재 각 하나씩이면 1:1
화살표 없으면 n

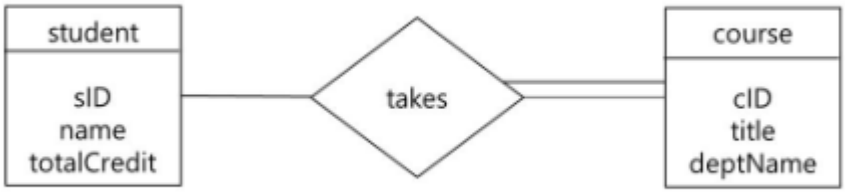
- Many-to-one
  - A professor is associated with at most one student
  - A student is associated with several (including 0) professors

- Many-to-many
  - A professor is associated with several (possibly 0) students
  - A student is associated with several (possibly 0) professors

Particiption Constraints

- Total participation
  더블라인으로 표현
  -> 해당 테이블의 모든 entity가 포함되어야함

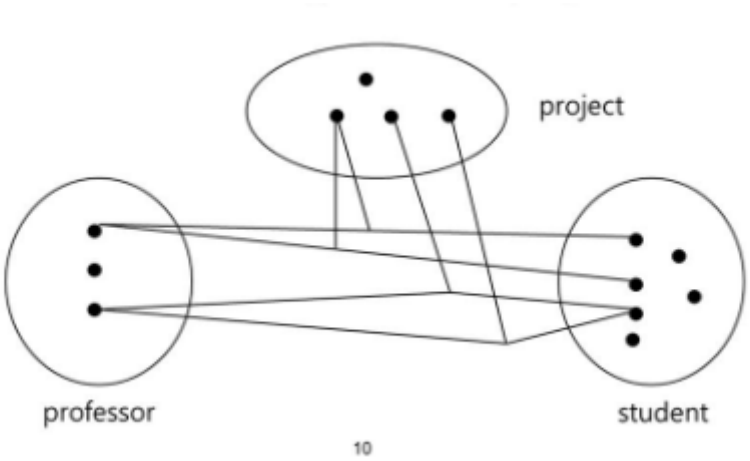
- Partial participation
  모든 entuty 표현 안되어도 됨



- m:n
- course는 모두 student를 가져야함
- student에서는 course안들어도 됨.

Ternary Relationship

Cardinality 할 때



one은 하나만(??)

Roles

course entity간의 relationship?
recursive relationship
선후수 과목...
role이 필요함 (역할)

Weak Entity Set

primary key를 갖지 않는 Entity set
-> 왜 존재? identifying entity set에 의존적임
더블 다이아몬드 / 더블 네모로 표현

Weak의 key는?

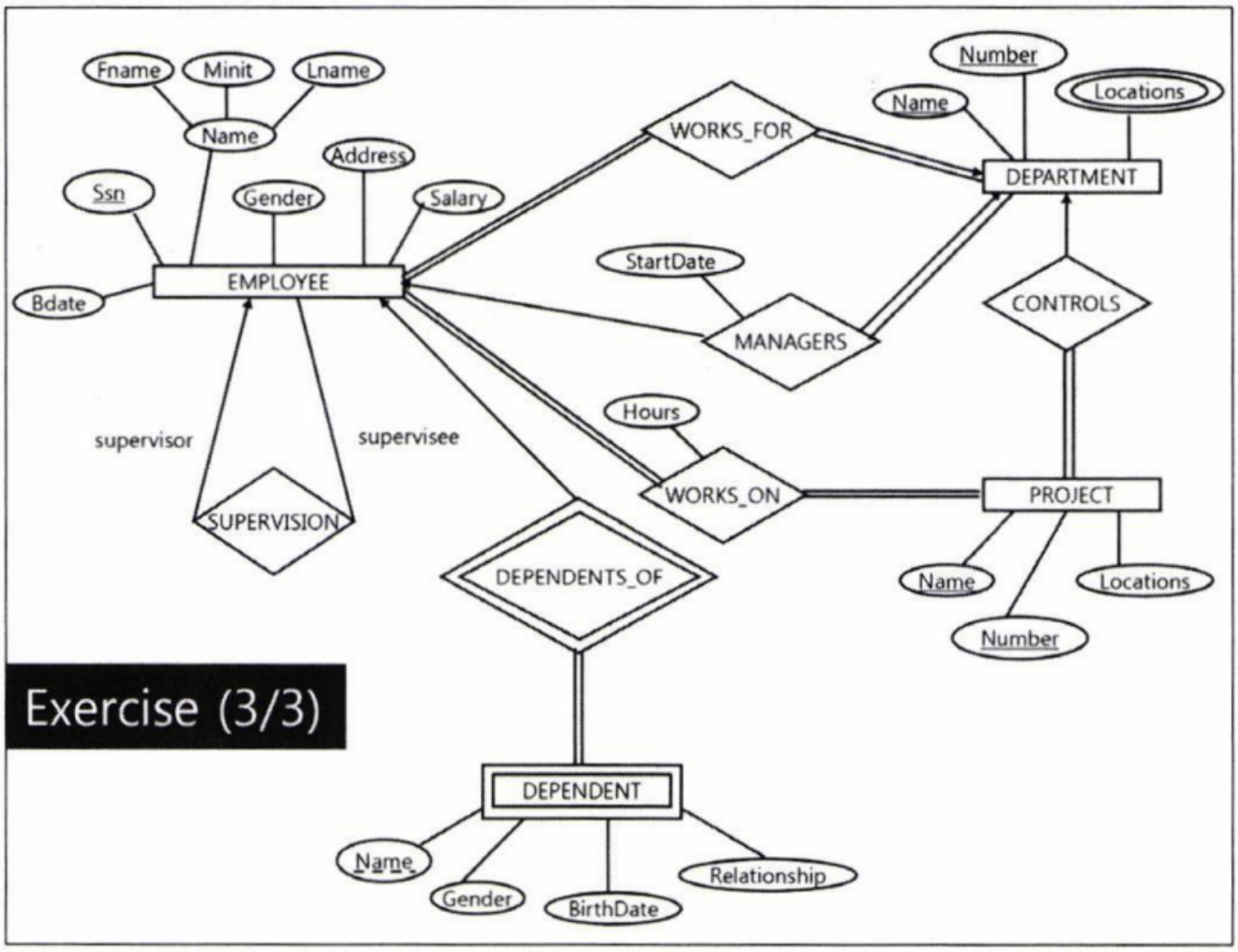
weak entity type의 primary key... 없으므로 partial key

Exercise

1. Draw an ER diagram for the company below. Make assumptions on constraints if necessary. The company keeps track of a company’s employees, departments, projects, and so on.
2. The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department, We keep track of the start data when that employee began managing the department. A department may have several locations.
3. A department controls a number of projects, each of which has a unique name, a unique number, and a single locat

4. We store each employee's name, social security number, address, salary, gender, and birthdate. An employee is assigned to one department but may work on several projects, which are not necessarily controlled by the same department. We keep track of the number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee

5. we want to keep track of the dependents of each employee for insurance purposes. We keep each dependent's name, gender, birthdate, and relationship to the employee



Reduction to Relation Schemas

ER 다이어그램으로부터 스키마 생성하기

Strong -> Table
weak -> 변화한 후 Table

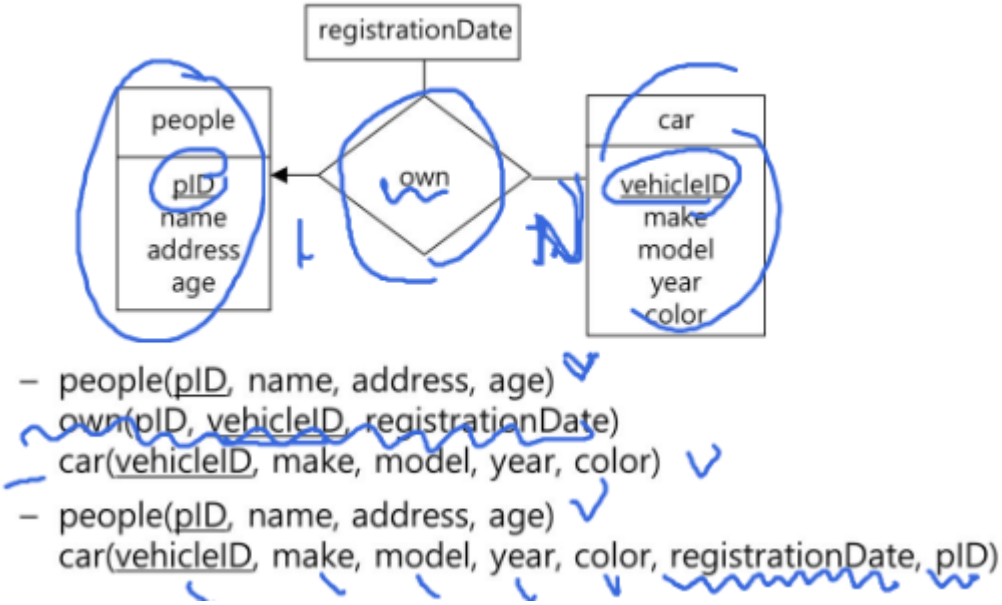
n:m인 경우 그냥 그대로 가능
n:1인 경우 many side역할
1:1안 아무 쪽이나 entity type으로 가능?

1:n) people <- own - car

people, car은 하나의 테이블로 가면 됨.

- own은 nside의 key를 가져오기 (car-vehicleID)

- nside 밀기 own의 속성인 registrationDate를 car에 넣고 people의 주키인 pID도 넣기



- people(pID, name, address, age)
  own(pID, vehicleID, registrationDate)
  car(vehicleID, make, model, year, color)
- people(pID, name, address, age)
  car(vehicleID, make, model, year, color, registrationDate, pID)

1:1

own - 속성 + 양쪽 주키
왼쪽을 밀어도 되고 오른쪽을 밀어도 되고..

multivalue attribute

table로 간다.
그런 경우 키 - ID+Phonenum

Database Design Issues

ER diagram	**decision**
Attribute	entity
entity set	relationship set
ternary relationship	a pair of binary relationship
strong	weak
specialization	generalization

entity set and attribute
has no right/no wrong

Redundant Attributes

각 테이블에 중복되는 속성 존재시 하나는 삭제해야함
-> 나중에 일치성 문제...

대부분은 Binary 사용