# SQL 뷰

# SQL 뷰

## ➢ 뷰 (view)

- 다른 테이블(기본 테이블이나 이전에 정의한 뷰)들에서 유도된 단일 테이블
- 기본테이블 (base table: 물리적으로 저장되어 있음)과는 달리 가상테이블(virtual table)로 간주됨
- 가상 테이블이기 때문에 뷰에 적용할 수 있는 갱신 연산들은 제약을 받지만 뷰에대한 질의는 아무런 제약을 받지 않음
  - 따라서, 물리적으로 존재하지는 않더라도 자주 참조할 필요가 있는 테이블을 명시
  - 예) COMPANY 데이터베이스 스키마에서 사원의 이름과 그 사원이 참여하는 프로젝트 이름들을 구하는 질의가 자주 사용된다면 EMPLOYEE, WORKS_ON, PROJECT 세개의 테이블을 매번 조인하기보다는 자주 검색하고자 하는 애트리뷰트들을 포함하고 있는 조인의 결과를 미리 뷰로 저장해 놓음
  - 위의 예에서 뷰를 정의할 때 사용하는 기본 테이블들 (EMPLOYEE, WORKS_ON, PROJECT) 을 정의 테이블 (defining table) 이라고 부름

## ➢ 뷰의 특징

- 질의에 필요한 조인들을 줄여줌으로써 질의를 간단하게 작성할 수 있도록 함
- 보안 기법과 권한 부여 매커니즘에 활용 가능
- 뷰는 항상 최신 정보를 가짐
  - 뷰의 정의에 사용된 기본 테이블들의 투플들이 수정되면, 뷰는 자동적으로 변경 사항들을 반영하게 됨

# SQL 에서 뷰의 명시

## ➢ 뷰의 정의

- CREATE VIEW
- 뷰의 정의는 가상 테이블 이름, 애트리뷰트 이름들의 리스트, 뷰의 내용을 나타내는 질의로 구성됨
- 뷰의 애트리뷰트 이름들을 명시하지 않을 때 정의테이블로부터의 이름을 사용하게 됨

## ➢ 뷰의 삭제

- DROP VIEW

```
V1:   CREATE VIEW   WORKS_ON1
      AS SELECT     Fname, Lname, Pname, Hours
          FROM      EMPLOYEE, PROJECT, WORKS_ON
          WHERE     Ssn=Essn AND Pno=Pnumber;

V2:   CREATE VIEW   DEPT_INFO(Dept_name, No_of_emps, Total_sal)
      AS SELECT     Dname, COUNT (*), SUM (Salary)
          FROM      DEPARTMENT, EMPLOYEE
          WHERE     Dnumber=Dno
          GROUP BY  Dname;
```

| Fname | Lname | Pname | Hours |
|---|---|---|---|

| Dept_name | No_of_emps | Total_sal |
|---|---|---|

# 뷰의 구현 및 갱신

## ➢ 뷰의 구현

- 효율적인 질의를 위해서 뷰를 구현할 경우에 두 가지 접근 방식이 있음
- 질의 수정 (query modification)
  - 사용자가 작성한 뷰에대한 질의를 기본테이블에 대한 질의로 수정 또는 변경하는 것
  - 실행하는 데에는 시간이 많이 소요되는 복잡한 질의가 되어 질의 작성 상의 간결함 이외의 추가적 이익이 없음
- 뷰의 실체화 (view materialization)
  - 뷰를 생성하거나 처음 질의 요구가 있을 때 임시적인 또는 영구적인 뷰 테이블 (view table)을 물리적으로 생성하고 뷰에대한 다른 질의들이 이어진다고 가정하고 테이블을 유지하는 것
  - 일반적으로 뷰는 질의가 되고 있는 동안에 실체화된 테이블로써 유지되고, 일정 기간 동안 뷰에 질의가 없으면 시스템은 물리적인 테이블을 자동적으로 삭제하고 다음에 해당 뷰를 참조하는 질의가 있을 경우 처음부터 뷰 테이블을 재구성함

## ➢ 뷰의 갱신

- 점진적 갱신 (incremental update): 뷰의 내용을 최신 상태로 유지하기 위해서 기본 테이블이 수정되었을 때 이를 뷰 테이블에 효과적으로 반영할 수 있는 방법이 필요함
  - 즉시 갱신 (immediate update): 관련 기본 테이블들이 갱신 되자마자 뷰를 수정
  - 느긋한 갱신 (lazy update): 뷰 질의에 의해 필요할 때 뷰를 수정
  - 주기적 갱신 (periodic update): 뷰를 주기적으로 수정 (뷰 질의가 최신의 내용이 아닌 결과를 가질 수 있음)

# 뷰에 대한 변경과 권한 관리

## ➢ 뷰에 대한 데이터 변경

- 사용자는 어떤 뷰에 대해서도 검색 질의를 항상 실행시킬 수 있지만, 뷰에 대한 INSERT, DELETE, UPDATE 문을 실행하는 것은 제한적임
- 하나의 릴레이션을 사용해서 정의된 뷰일 경우에는 그 기본 릴레이션의 기본키와 NOT NULL 조건을 가지는 모든 애트리뷰트들을 포함하고 있으면 뷰의 갱신이 가능함
- 뷰의 애트리뷰트가 산술식, 집계함수, 상수로 부터 유도되었을 경우에 갱신이 불가능함
- 일반적으로 다수의 테이블 상에서 조인을 사용해서 정의된 뷰들은 갱신할 수 없음
- 수정이 불가능한 뷰에서 새로운 뷰가 생성된 경우에는 그 새로운 뷰 또한 갱신할 수 없음

## ➢ 권한 관리 관점의 뷰

- 특정 뷰에 대한 질의 권한은 주고 기본 테이블 자체에 대한 권한을 주지 않음으로 권한 관리에 활용 가능
  - 특정 뷰가 기본 테이블의 일부 애트리뷰트들만 포함하고 있거나 어떠한 조건을 만족시키는 투플들만 포함하고 있는 경우에는 해당 정보 이외의 다른 정보에 대한 질의를 막을 수 있기 때문에 권한 관리가 가능함
- 따라서, 적절한 뷰를 생성하고 특정 사용자들에게 그 뷰를 접근하고 기본 테이블은 접근하지 못하도록 권한을 부여함으로써, 사용자들이 그 뷰에서 명시한 데이터만 검색하도록 제한 가능

# ER- 관계 사상에 의한 관계 데이터베이스 설계

# 논리적 데이터 베이스 설계 (데이터 모델 사상)

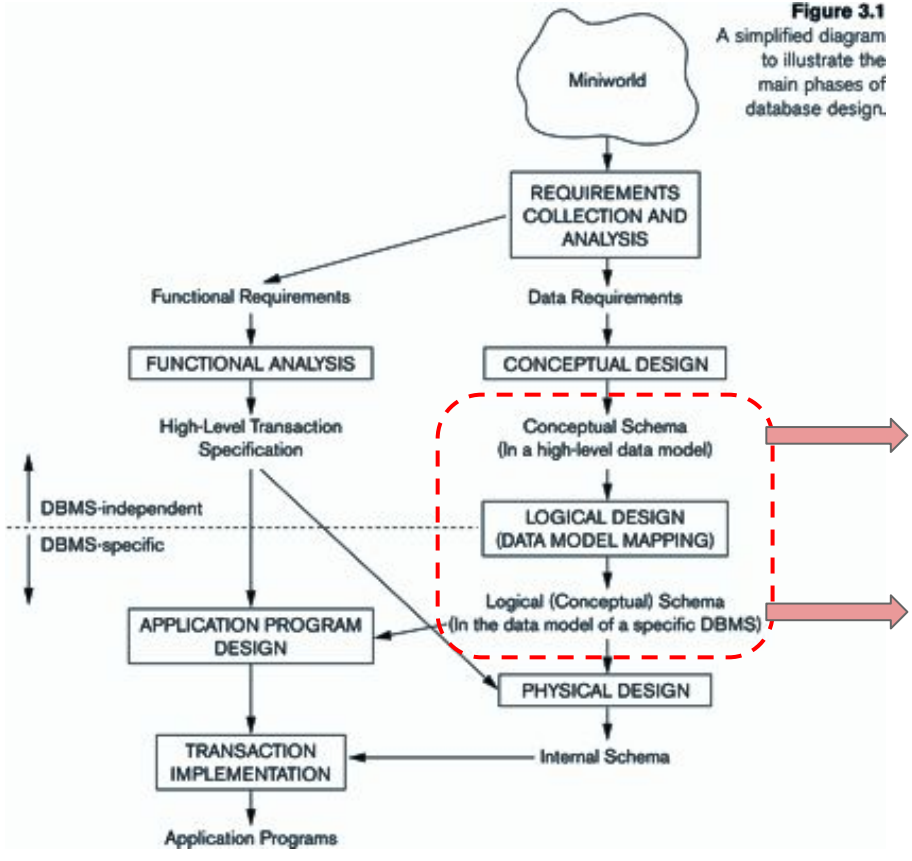



**Figure 3.1**
A simplified diagram to illustrate the main phases of database design.

ER 모델
(Entity-Relationship Model)

관계형 모델
(Relational Database Model)

# ER 관계 사상 알고리즘

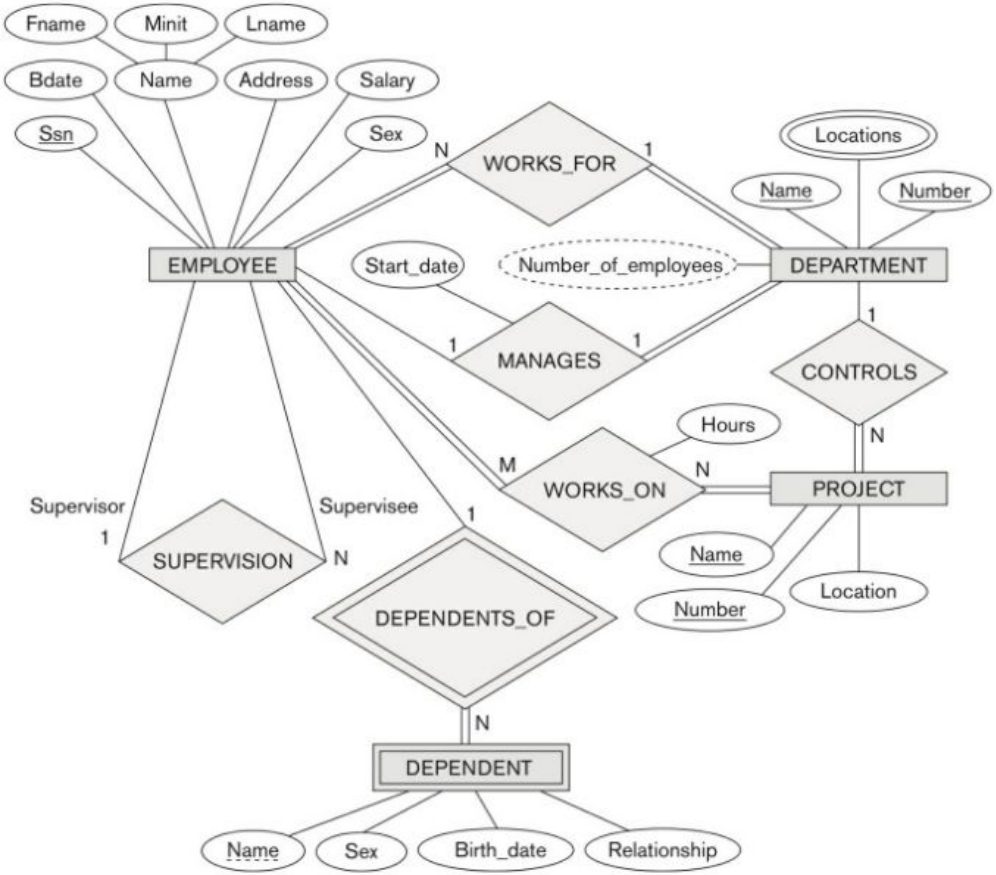
## ➢ ER 관계 사상 알고리즘의 단계

- 사상은 단순 단일값 애트리뷰트를 갖는 테이블을 만든다는 것을 전제로 함
- 또한, 필요하다면 기본키, 유일키, 참조무결성 제약조건을 포함한 관계모델 제약 조건은 사상의 결과에 명세화 됨

## ➢ 1단계: 정규 엔터티 타입의 사상

- ER스키마의 정규 엔티티 타입 또는 강한 엔티티 타입에 대해서는 E의 모든 단순 애트리뷰트들을 포함하는 릴레이션 R을 생성함
- 복합 애트리뷰트가 있을 경우에는 그것의 단순 요소 애트리뷰트들 만을 포함시킴
- 또한, 키 애트리뷰트 중 하나를 선택하여 R의 기본키로 함
  - E 에서 선택된 키가 복합 애트리뷰트이면 그것을 구성하는 모든 단순애트리뷰트의 모임이 R의 기본키가 됨
- COMPANY 데이터베이스의 경우에는 EMPLOYEE, DEPARTMENT, PROJECT 가 이에 해당
- 외래키와 관계 애트리뷰트들은 아직 포함되지 않음

# COMPANY 데이터베이스에 대한 ER 개념적 스키마 다이어그램



# 1단계 결과

| Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary |
|---|---|---|---|---|---|---|---|

| Dname | Dnumber |
|---|---|

| Pname | Pnumber | Plocation |
|---|---|---|

# ER 관계 사상 알고리즘

## ➢ 2단계: 약한 엔터티 타입의 사상

- ER 스키마에서 소유 엔티티 타입 E 를 갖는 약한 엔티티 타입 W에 대하여 릴레이션 R 을 생성하고 W의 모든 단순 애트리뷰트를 R의 애트리뷰트들로 포함시킴
- 그리고 소유 엔티티 타입에 해당하는 릴레이션의 기본키를 R의 외래키로 포함시킴
- R의 기본 키는 소유자 릴레이션의 기본키와 약한 엔티티 타입의 부분키의 조합임

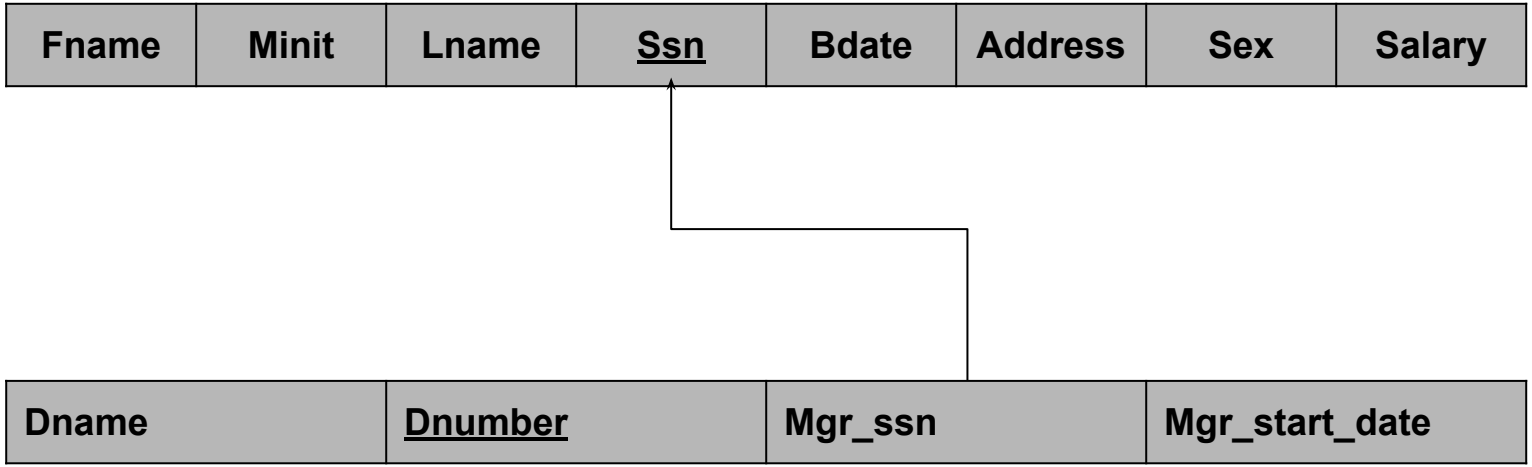
| Essn | Dependent_name | Sex | Bdate | Relationship |
|---|---|---|---|---|

# ER 관계 사상 알고리즘

## ➢ 단계3: 이진 1:1 관계 타입의 사상

- ER 스키마의 이진 1:1 관계 타입 R에 대해 R에 참여하는 엔티티 타입을 S, T라고 하였을 때 가능한 접근 방식은 다음의 세 가지가 있음
- 외래키 접근 방식
  - 한 릴레이션 S 를 택하여 T의 기본키를 S에 외래키로 포함시킴
  - 이 때에 R에 완전히 참여하는 엔티티 타입을 S로 선택하는 것이 좋음
  - 예를 들면, MANAGES 관계 타입에 대해서 DEPARTMENT 가 완전히 참여하므로 DEPARTMENT 의 릴레이션에 EMPLOYEE의 기본키를 Mgr_ssn의 이름으로 포함시키고 MANAGES 관계 타입의 애트리뷰트은 Start_date 도 Mgr_start_date 으로 DEPARTMENT의 애트리뷰트로 포함시킴
- 합병된 릴레이션 접근 방식
  - 두 엔티티 타입과 관계 타입을 하나의 릴레이션으로 합칠 수 있음
  - 하지만, 두 엔티티 타입이 모두 완전하게 참여할 때만 가능하고, 두 테이블이 정확히 동일한 수의 투플을 가질 경우에만 가능함
- 교차 참조 또는 관계 릴레이션 접근 방식
  - 두 릴레이션 S, T의 기본키를 교차 참조하기 위한 세 번째 릴레이션 R을 정의하는 방법
  - 관계 인스턴스가 별로 없어서 외래키에 널 값이 채워지는 것을 피하고자 할 때 특히 유용한 방법임
  - 보통 이진 M:N 릴레이션에서 주로 사용됨

# 3단계 결과


Fname
Minit
Lname
Ssn
Bdate
Address
Sex
Salary
Dname
Dnumber
Mgr_ssn
Mgr_start_date

# ER 관계 사상 알고리즘

## ➢ 단계4: 이진 1:N 관계 타입의 사상

- ER 스키마의 이진 1:N 관계 타입 R에 대해 N측에 참여하는 엔티티 타입을 S, 1-측에 엔티티 타입을 T라고 하였을 때 가능한 접근 방식은 다음의 두 가지가 있음
- 외래키 접근 방식
  - T의 기본키를 S의 외래키로 포함시킴
  - 관계 타입의 모든 N-측 엔티티 인스턴스가 최대한 하나의 1-측 엔티티 인스턴스와 관계가 있기 때문임
  - 관계 타입 R 의 모든 애트리뷰트도 S 의 애트리뷰트로 포함시킴
  - 예: WORKS_FOR, CONTROLS, SUPERVISION 들에 대해서 위의 접근 방식 사용 가능
- 관계 릴레이션 방식
  - 이진 1:1 관계의 세 번째 방법과 동일하게 관계 릴레이션 (교차 참조)를 사용하는 것임

## ➢ 단계 5: 이진 M:N 관계 타입의 사상

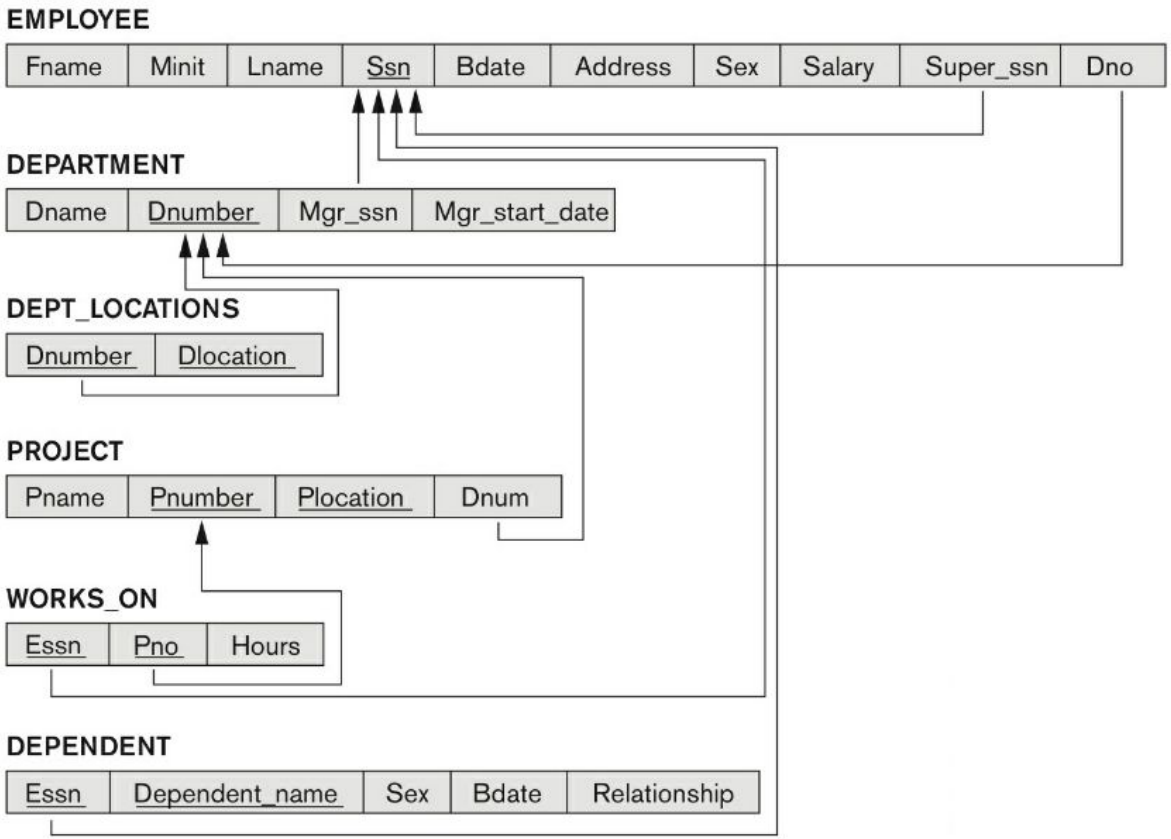
- 다치 속성을 가지지 않는 기존 관계 모델에서 N:M 관계를 매핑하는 유일한 방법은 관계 릴레이션 (relationship relation)을 생성하는 것임
- 관계 타입 R을 표현하기 위한 새로운 릴레이션 S를 생성함
  - 참여 엔티티 타입에 해당하는 릴레이션들의 기본키를 S의 외래키 애트리뷰트로 포함시키고 그들의 조합이 S의 기본키가 됨
  - 예: M:N 관계 타입인 WORKS_ON을 사상시켜 WORKS_ON 릴레이션을 생성함

# ER 관계 사상 알고리즘

## ➢ 6단계: 다치 애트리뷰트의 사상

- 다치 애트리뷰트 A 에 대하여 새로운 릴레이션 R을 생성함
- 릴레이션 R은 A에 대응하는 하나의 애트리뷰트를 포함하고 A를 다치 애트리뷰트로 가졌던 앤터티타입이나 관계 타입의 기본키 애트리뷰트를 R의 외래키로 포함시킴
- R의 기본키는 A와 K의 조합임
- 만약 다치 애트리뷰트가 혼합 애트리뷰트 이면, 이 혼합 애트리뷰트의 단순 요소 애트리뷰트들을 각각 포함시킴
- 예: LOCATIONS 속성에 대해서 DEPT_LOCATIONS 릴레이션을 생성함
  - DEPARTMENT의 속성이므로 DEPARTMENT 의 기본키인 Dnumber 를 외래키로 포함시키고, LOCATIONS 속성에 대해서 Dlocation 이라는 애트리뷰트를 포함시킴
  - Dnumber, Dlocation 의 조합이 기본키가 됨

# ER 관계 사상 알고리즘을 적용한 결과



# ER 모델과 관계형 모델의 비교

**Table 9.1** Correspondence between ER and Relational Models

| ER MODEL | RELATIONAL MODEL |
|---|---|
| Entity type | *Entity* relation |
| 1:1 or 1:N relationship type | Foreign key (or *relationship* relation) |
| M:N relationship type | *Relationship* relation and *two* foreign keys |
| *n*-ary relationship type | *Relationship* relation and *n* foreign keys |
| Simple attribute | Attribute |
| Composite attribute | Set of simple component attributes |
| Multivalued attribute | Relation and foreign key |
| Value set | Domain |
| Key attribute | Primary (or secondary) key |

# 중간고사

# 시험 장소 및 일시

- **시험 시간**
  - 2019년 10월 21일
  - 15:15 - 16:30
- **시험 장소**
  - **난향관 702호**

# 중간고사 리뷰

- **개념에 대한 이해**
  - T/F 문제
  - 정의 및 설명
- **ER-모델 (Entity-Relationship Model)**
  - ER 다이어그램
  - 강한 엔티티타입, 약한 엔티티타입, 관계 타입
- **관계 데이터 모델 (Relational Data Model)**
  - 무결성 제약조건
- **ER 관계 사상 알고리즘**
  - ER 다이어그램 -> 관계 데이터 모델
- **SQL 질의**
  - SQL 질의 작성
  - 데이터베이스 상태로부터 질의의 결과 나타내기