



### 3. 논리적 설계

---

- 논리적 설계의 목적
  - 개념적 스키마 (ER 모델) → DBMS의 논리적 스키마로 변환
- 개념적 설계와 논리적 설계의 차이
  - 개념적 설계 - 스키마의 표현력과 완전성을 추구
  - 논리적 설계 - 논리적 모델이 제공하는 자료 구조와 제약사항을 효율적으로 이용
- 논리적 설계의 접근 방향
  - ER 다이어그램을 단순한 ER 다이어그램으로 변환
  - 단순한 ER 다이어그램을 DBMS의 논리적인 모델로 변형




## 3.1 ER 다이어그램의 변환

---

- 데이터베이스 부하의 모델링
- 유도된 데이터에 대한 결정
- 일반화 구조의 제거
- 개체의 분할
- 개체와 관계의 병합
- 주키의 선택

# 데이터 볼륨 테이블

Concept	Type	Volume	Concept	Type	Volume
CLIENT	E	400,000	ORDINARY(TRIP)	E	150
NAME	A	400,000	SPECIAL(TRIP)	E	50
TELEPHONE	A	400,000	DAILY_TRIP	E	2,000
<u>PASSENGER</u>	E	395,000	BUS	E	500
FREQUENT_ TRAVELER	E	20,000	BUS_PROBLEM	E	1,500
AGENCY	E	5,000	<u>HOLD_RES</u>	R	1,000,000
DAILY_ROUTE_ SEGMENT	E	20,000	IS_ON_BOARD	R	800,000
ROUTE_SEGMENT	E	2,000	OF	R	20,000
TRIP	E	200	(from DAILY_ROUTE_SEGMENT)		
			OF	R	2,000
			(from DAILY_TRIPE)		
			...		



# 연산 빈도 테이블

---

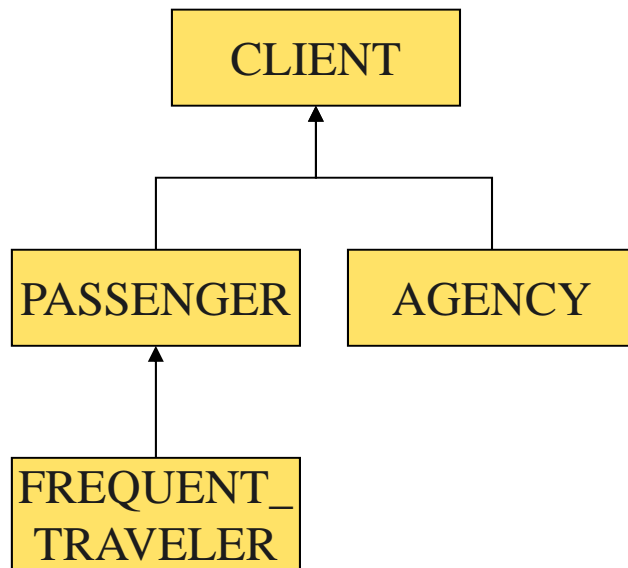
Operation Name/Description	Frequency	Type
01 Search ROUTE_SEGMENT by DEP_CITY	50 times/day	OL
02 Search ROUTE_SEGMENT by DEP_CITY and ARR_CITY	200 times/day	OL
03 Search TRIPs with intermediate stop at a city	5 times/day	OL
04 Create a new CLIENT record	500 times/day	OL
05 Make a reservation	20,000 times/day	OL
06 Delete a reservation of a past TRIP	70 times/day	OL
07 Delete a CLIENT record	1 time/day	OL
08 Qualify a CLIENT as FREQUENT TRAVELER	10 times/day	OL
09 Search the amount of miles earned by CLIENT	Once/month	B
010 Update the mileage of FREQUENT TRAVELER	Once/day	B
011 Search current reservation of a given Trip on a given date	100 times/day	OL

# 유도된 데이터에 대한 결정

	ARR_CITY	DEP_CITY	NUMBER_OF_SEGMENTS	AVAILABLE_SEATS	RESERVED_SEATS
O1 Search ROUTE_SEGMENT by DEP_CITY	N	N	N	N	Y
O2 Search ROUTE_SEGMENT by DEP_CITY and ARR_CITY	N	N	N	Y	Y
O3 Search TRIPs with intermediate stop at a given city	N	N	N	Y	Y
O4 Create a new CLIENT record	N	N	N	N	N
O5 Make a reservation	Y	Y	N	N	N
O6 Delete a reservation of a past TRIP	Y	Y	N	N	N
O7 Delete a CLIENT record	N	N	N	N	N
O8 Qualify a CLIENT as FREQUENT TRAVELER	N	N	N	N	N
O9 Search the amount of miles earned by CLIENT	N	N	N	N	N
O10 Update the mileage of FREQUENT TRAVELER	N	N	N	N	N
O11 Search current reservation of a given Trip on a given date	N	N	N	N	N

# 일반화 구조의 통합

- CLIENT, PASSENGER, AGENCY들간의 일반화

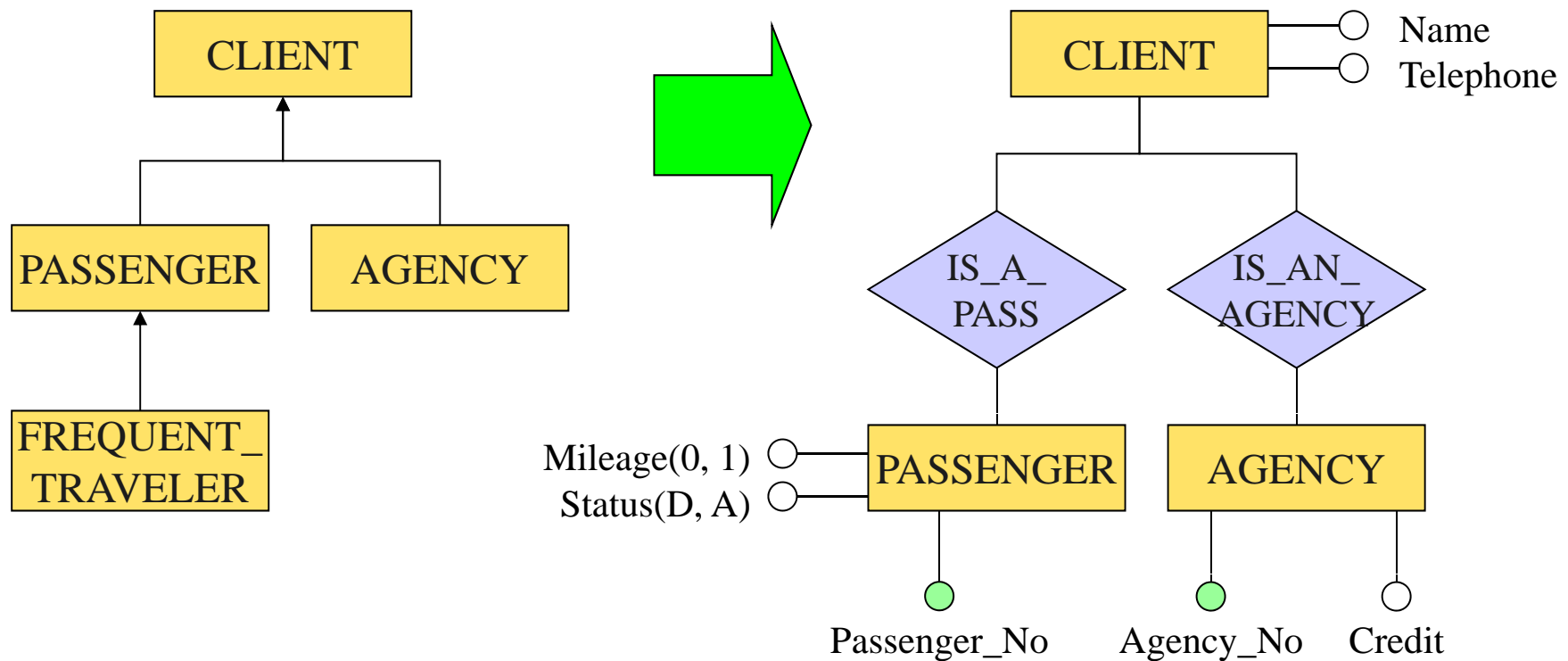


일반화 구조를 관계로 변경하는  
대안들

- 하나의 개체: Agency에 대한 Overhead
- 두 개의 개체: 관계의 수가 많아져 복잡
- 세 개의 개체: 가장 일반적

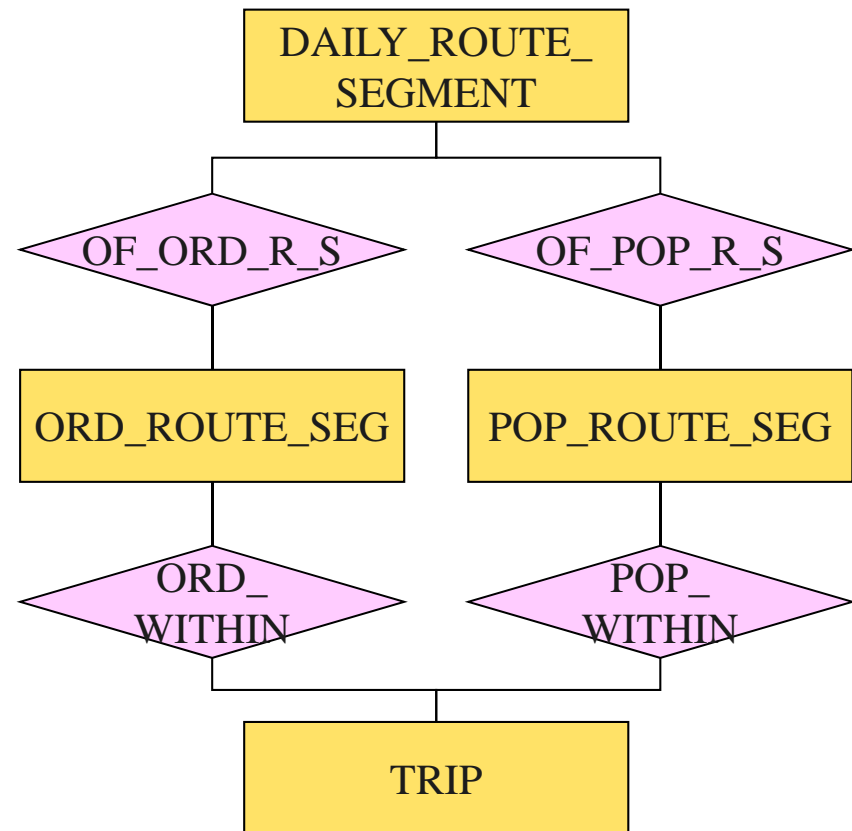
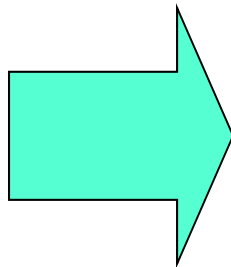
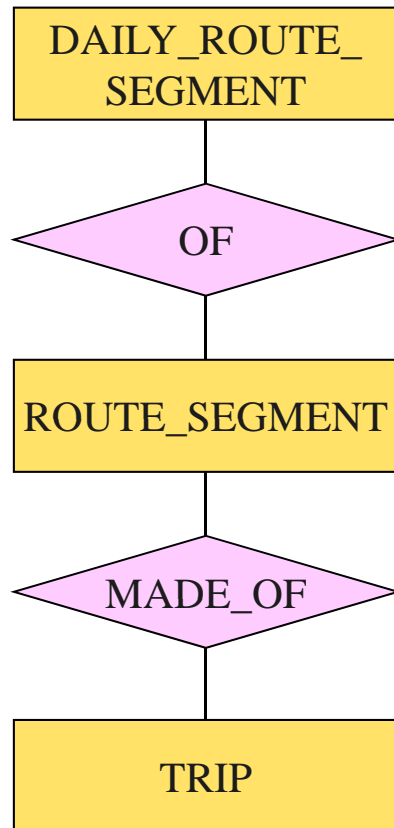
# 일반화 구조의 통합

- CLIENT, PASSENGER, AGENCY들간의 일반화



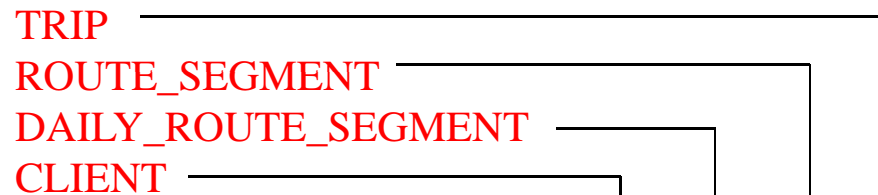
# 개체의 분할

- DAILY\_ROUTE\_SEGMENT
  - ORD\_ROUTE\_SEGMENT + POP\_ROUTE\_SEGMENT





# 개체와 관계의 통합



## Operation Name/Description

O1 Search ROUTE_SEGMENT by DEP_CITY	N	N	Y	Y
O2 Search ROUTE_SEGMENT by DEP_CITY and ARR_CITY	N	N	Y	Y
O3 Search TRIPs with intermediate stop at a given city	N	N	Y	Y
O4 Create a new CLIENT record	Y	N	N	N
O5 Make a reservation	Y	Y	N	N
O6 Delete a reservation of a past TRIP	N	Y	N	N
O7 Delete a CLIENT record	Y	N	N	N
O8 Qualify a CLIENT as FREQUENT TRAVELER	Y	N	N	N
O9 Search the amount of miles earned by CLIENT	Y	N	N	N
O10 Update the mileage of FREQUENT TRAVELER	Y	N	N	N
O11 Search current reservation of a given Trip on a given date	N	Y	N	N

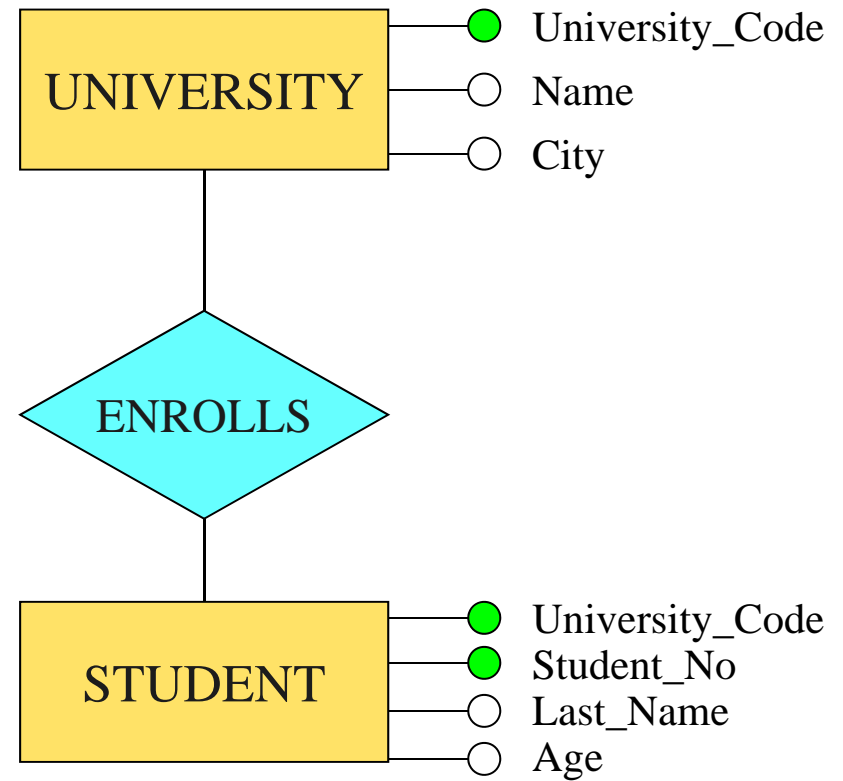
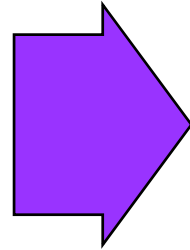
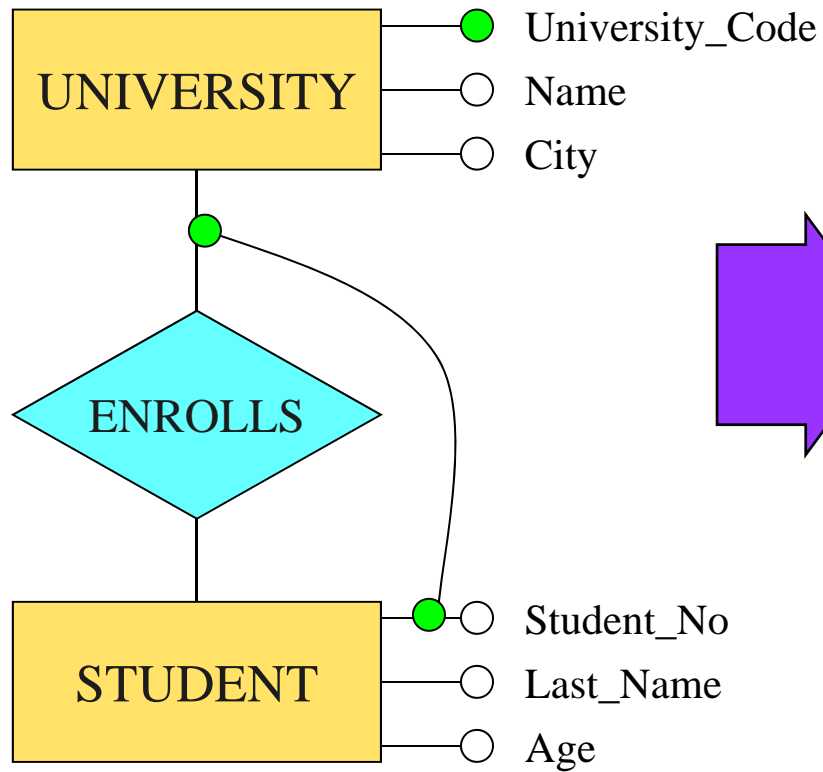


## 3.2 관계형 모델의 논리적 설계

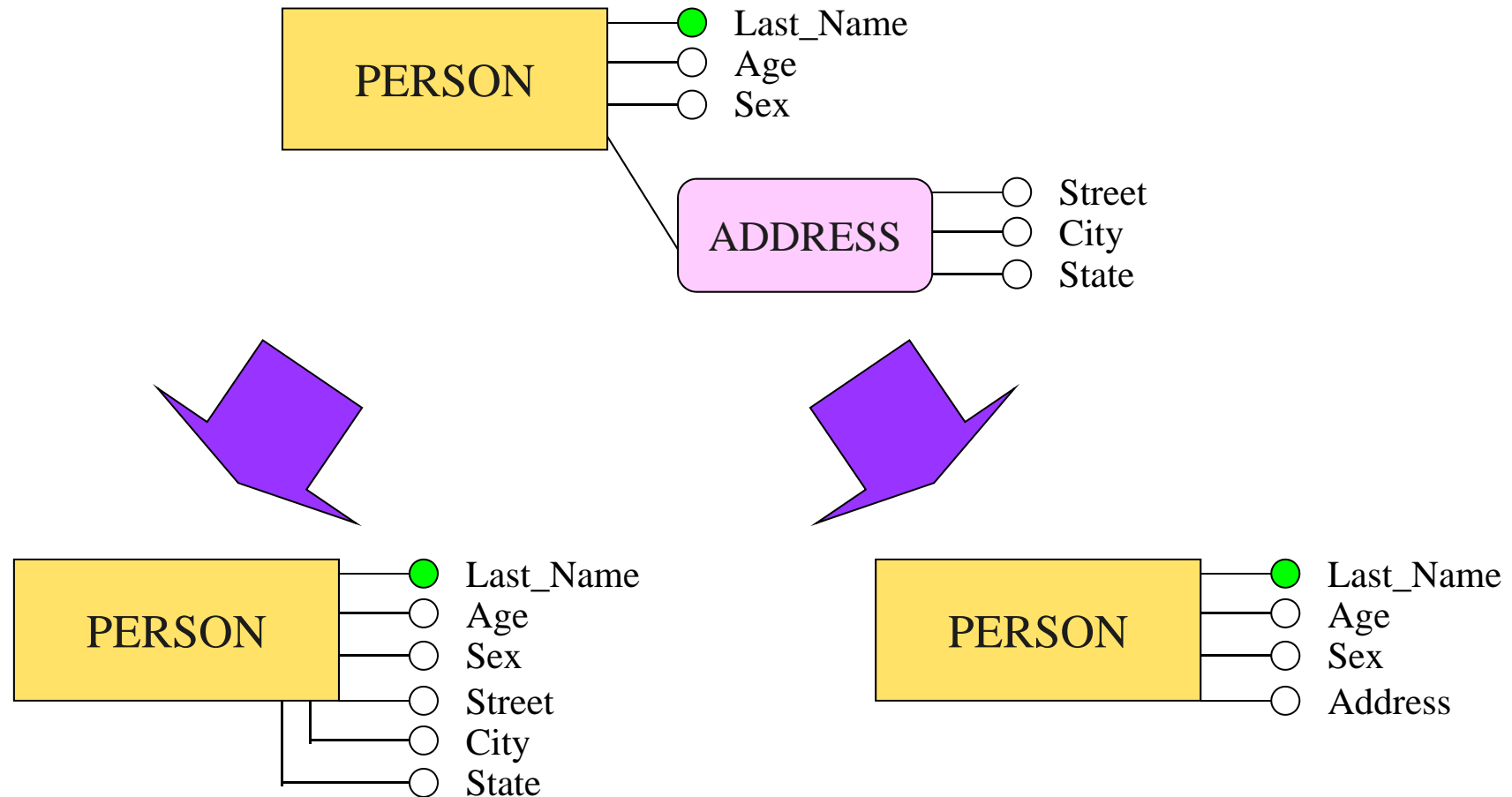
---

- 외부 식별자를 제거
- 복합 애트리뷰트 및 다중치 애트리뷰트를 제거
- 각 개체를 릴레이션으로 변환
- 관계를 릴레이션으로 변환
  - 다대다 관계: 새로운 릴레이션 생성
  - 일대일 및 일대다 관계: 기존 릴레이션에 애트리뷰트를 추가

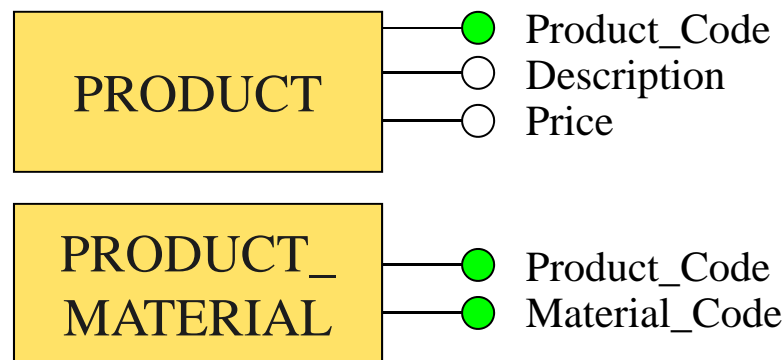
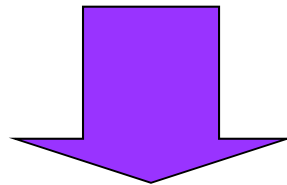
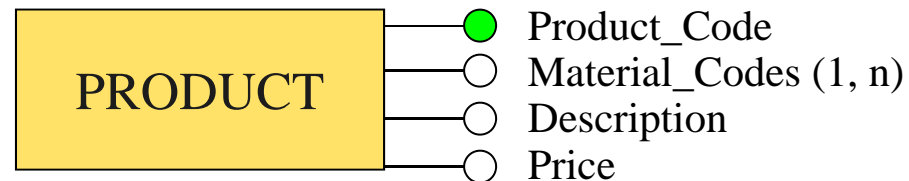
# 외부 식별자 제거



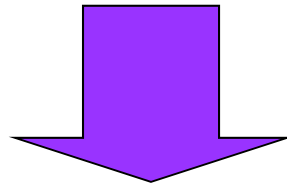
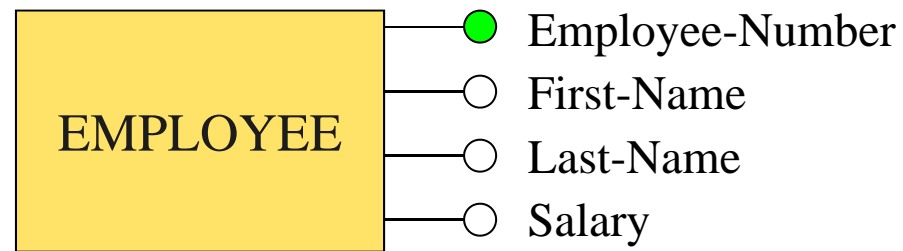
# 복합 애트리뷰트 제거



# 다중치 애트리뷰트 제거

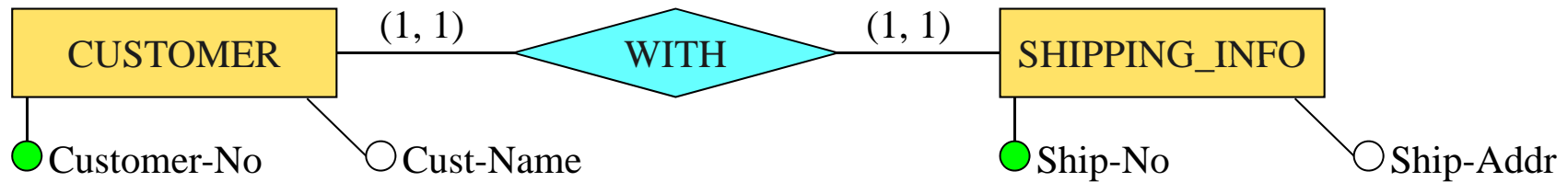


# 개체 변환

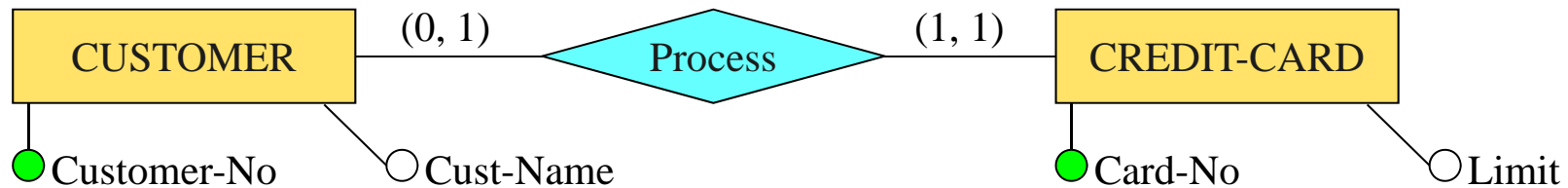


EMPLOYEE(Employee-Number, First-Name, Last-Name, Salary)

# 일대일 관계의 변환

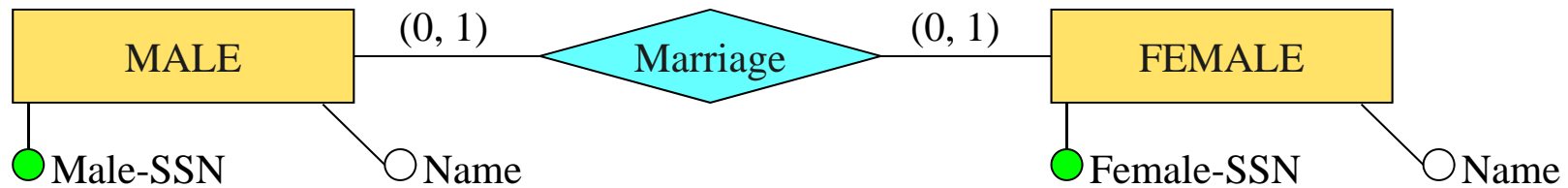


**CUST\_SHIPPING**(Customer-No, Cust-Name, Ship-No, Ship-Addr)



**CUSTOMER**(Customer-No, Cust-Name)

**CREDIT-CARD**(Card-No, Limit, Customer-No)

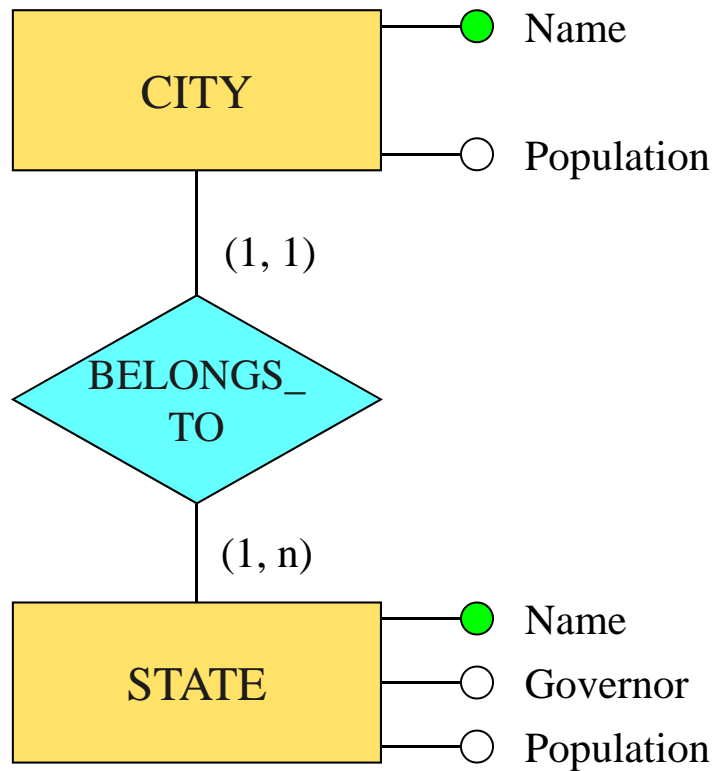


**MALE**(Male-SSN, Name)

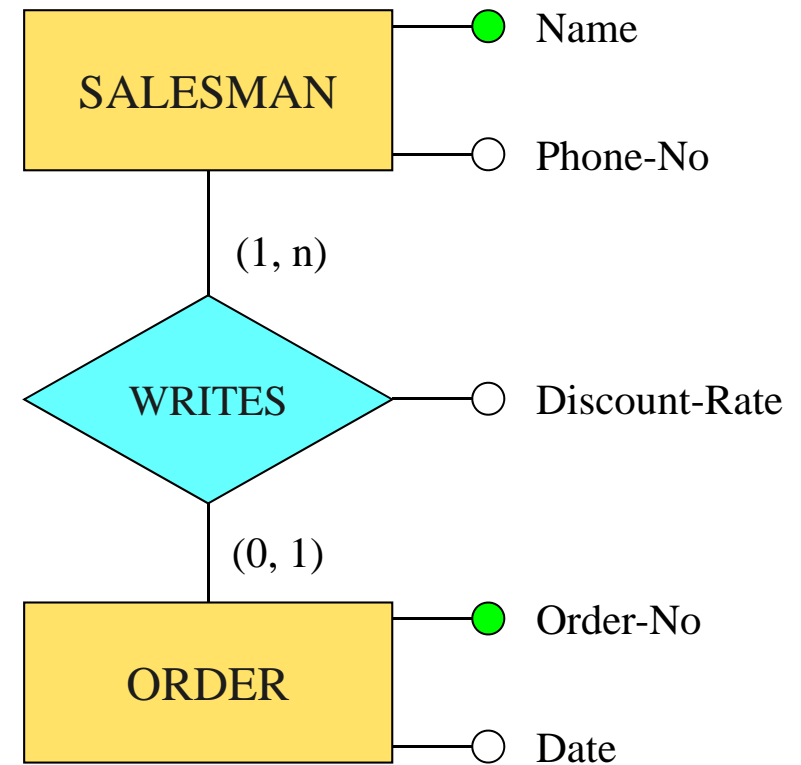
**FEMALE**(Female-SSN, Name)

**MARRIAGE**(Male-SSN, Female-SSN)

# 일대다 관계의 변환



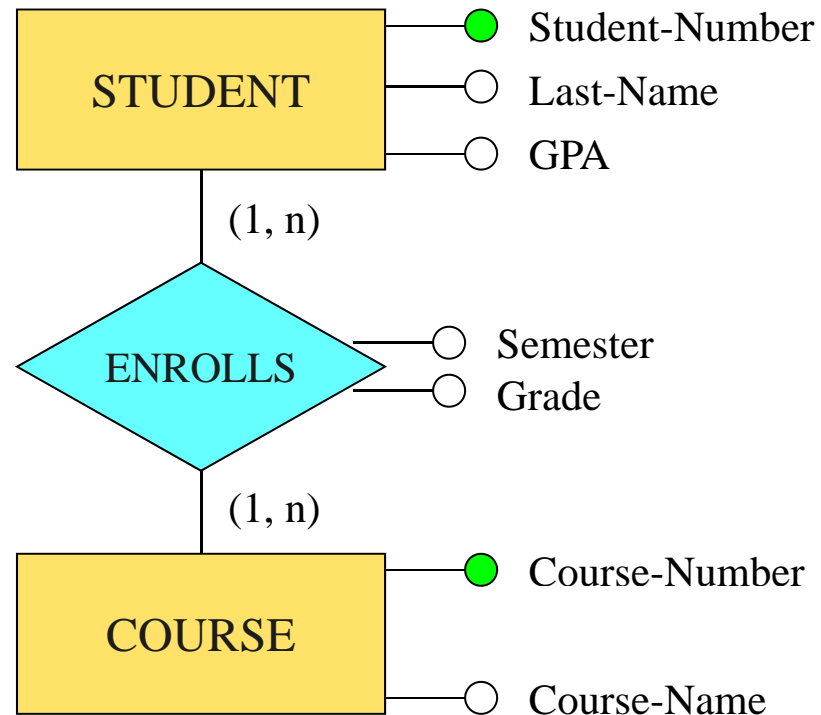
CITY(City-Name, State-Name, Population)  
STATE(State-Name, Governor, Population)



SALESMAN(Name, Phone-No)  
ORDER(Order-No, Date)  
Sales-Order(Order-No, Name, Discount-Rate)



# 다대다 관계의 변환

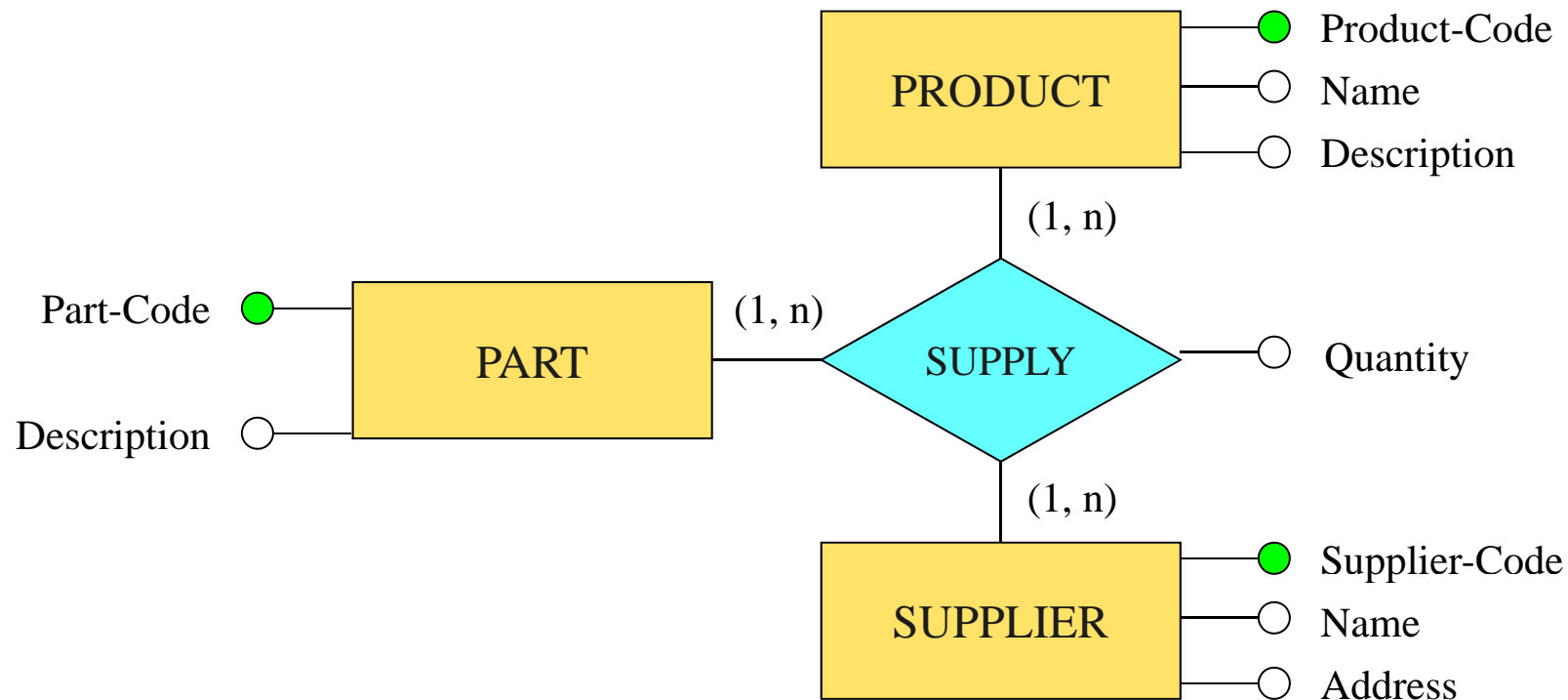


STUDENT(Student-Number, Last-Name, GPA)

COURSE(Course-Number, Course-Name)

ENROLLS(Student-Number, Course-Number, Semester, Grade)

# N-ary 관계의 변환



PRODUCT(Product-Code, Name, Description)

PART(Part-Code, Description)

SUPPLIER(Supplier-Code, Name, Address)

SUPPLY(Product-Code, Part-Code, Supplier-Code, Quantity)



## 사례 연구

---

- PASSENGER (Client-No, Name, Phone, Mileage, Status)
- AGENCY (Client-No, Name, Phone, Credit-Limit)
- DAILY-ROUTE-SEGMENT (Trip-Number, Segment-Number, Date, Available-Seats, Reserved-Seats)
- ORDINARY-ROUTE-SEGMENT (Trip-Number, Segment-Number, Dep-City, Dep-Time, Arr-City, Arr-Time, Price, Distance)
- POPULAR-ROUTE-SEGMENT (Trip-Number, Segment-Number, Dep-City, Dep-Time, Arr-City, Arr-Time, Price, Distance)
- TRIP (Trip-Number, Dep-City, Arr-City, Weekdays, Type, Event)
- HOLD-RES (Client-No, Trip-Number, Date, Segment-Number, Seat-Num, Smoking-Option)
- PASSENGER-IS-ON-BOARD (Client-No, Trip-Number, Date, Segment-Number)



## 4. 데이터베이스의 물리적 설계

---

### ■ 물리적 설계란?

- 논리적 스키마를 이용하여 효율적인 물리적 데이터베이스를 구성하는 일

### ■ 물리적 데이터베이스의 구조

- 저장 레코드의 형식, 저장 순서, 접근 경로, 물리적 저장 장치의 할당등에 대한 내역

### ■ 참고 사항

- 물리적 DB 구조는 세부적인 성능에 영향을 미침
- 물리적 설계 단계에서 고려할 사항들의 대부분은 특정 DBMS에 의해서 해결됨
- DBA만이 물리적 DB 구조의 구성에 관여할 수 있음



# 접근 방법 설계

## ■ 접근 방법 (Access Method)란?

- 저장 장치에 자료를 저장하거나 저장된 자료를 검색하는 방법
- 저장 구조와 탐색 기법으로 구성

## ■ 인덱스

- 레코드 검색시 빠른 접근 보장
- 주로 B<sup>+</sup> 트리로 구현됨
- 자주 검색되는 필드에 인덱스를 생성하면 검색 연산이 빨라짐
- 값의 구간에 의한 접근과 우선 순위에 따른 접근 가능
- 종류:
  - 유일 인덱스: 키 값의 중복을 허용 않음
  - 다중 키 인덱스: 여러 개의 필드로 구성되는 인덱스



## 접근 방법 설계 (계속)

---

### ■ 해싱

- 특정한 하나의 값의 연산 결과로 저장 위치를 파악
- 값의 구간에 의한 접근은 불가능
- 단지 그 값에 대한 검색만 가능

### ■ Clustering

- 연관된 레코드를 물리적으로 인접한 공간에 저장

### ■ Clustered Index

- 실제적인 데이터의 저장 순서가 인덱스의 순서에 따름
- 하나의 테이블에는 하나의 Clustered Index만 존재



# 성능 평가 및 개선

---

- 물리적 설계 내역에 대하여 성능면에서 검토
- 성능 평가의 기준 요소
  - 저장 공간
    - 주기억 장치: 공유 메모리 요구 용량
    - 보조 기억 장치 공간
    - 인덱스의 수 및 갱신 연산의 빈도
  - 처리 시간
    - 응답 시간
    - 트랜잭션 갱신 시간