**1.**Explain the following terms and concepts. Use examples if necessary.

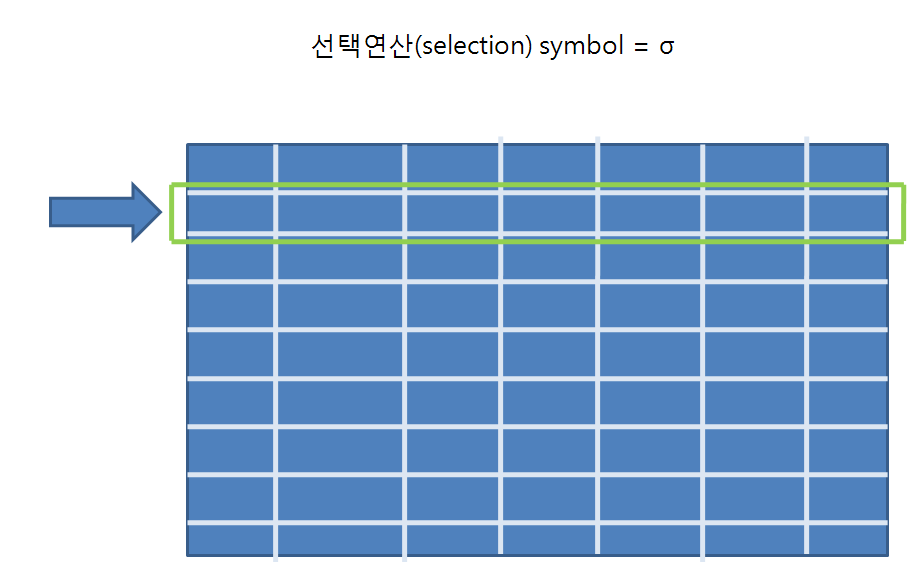
a)key, super key, primary key, candidate key

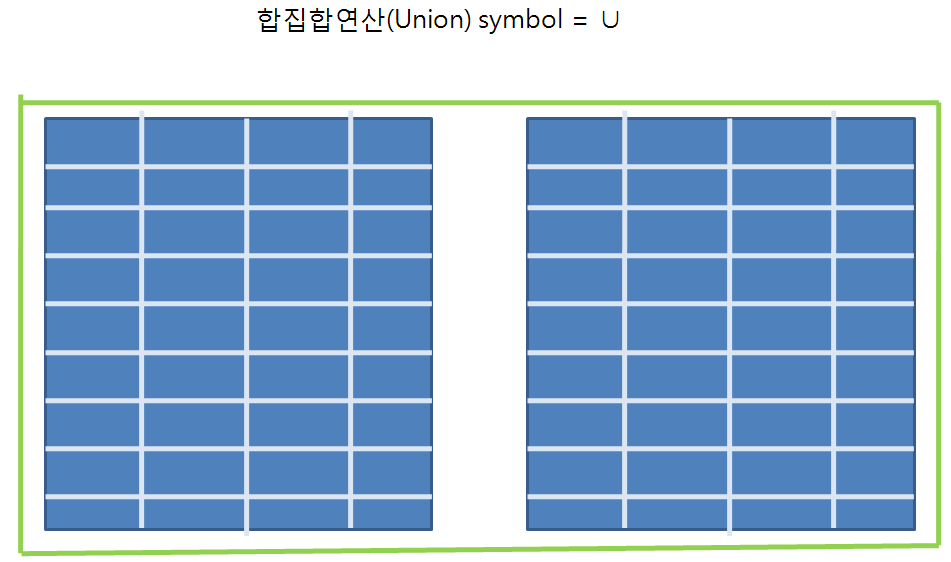
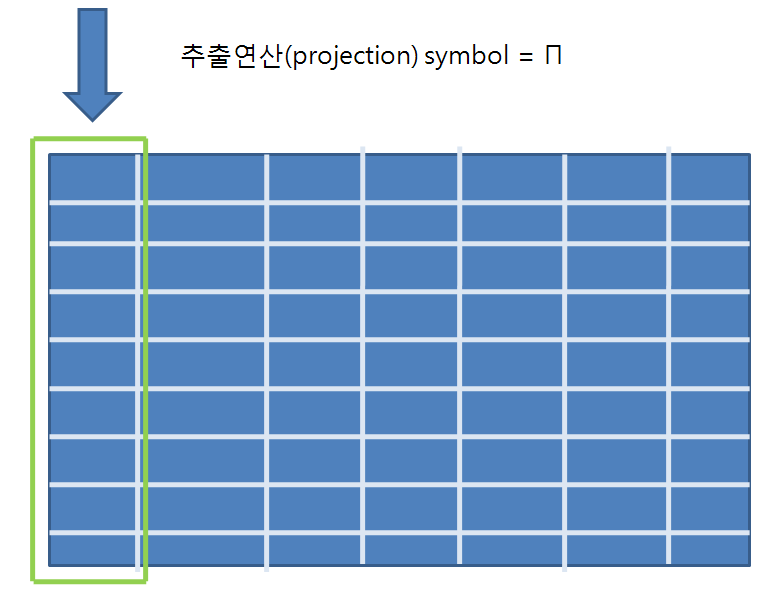
key는 identifier즉 식별자와 동일한 뜻입니다. 식별을 한다는 의미는 어떤 뜻일까요? 식별을 한다는 의미는 분류하고 다르게 보여서 알아 볼 수 있다는 것입니다. 하지만 데이터 베이스에서는 그 의미가 약간 다르게 됩니다. 데이터베이스에서는 식별을 하기 위해서는 일단은 기본적으로 알아차릴 수 있는 작은 단위가 있어야 합니다. 데이터베이스에서 처음으로 가장 큰 단위부터 작은 단위로 가보기 시작합니다. 데이터베이스는 테이블들로 아루어져 있습니다. 테이블은 세로의 tuple들과 세로의 attribute들로 이루어 져있습니다. 여기에서 식별을 할 수 있는 최소의 단위는 어떤 걸로 결정을 해야 할 까요? 세로로 특성들을 식별을 해본다고 생각을 해보겠습니다. 세로의 attribute들의 같은 범주의 다른 값입니다. 이렇게 식별을 하게 되어도 별로 데이터베이스에서는 가치가 없습니다. 그 특성만을 알아서는 어떠한 정보도 되지 않기 때문입니다. 그렇기 때문에 식별을 할 때에는 하나의 행의 여러가지 정보들을 담고있는 tuple별로 하게 됩니다. Key는 tuple을 식별을 할 때에 쓰이게 됩니다. 그렇다면은 super key라는 것은 어떤 의미가 될까요? Super key는 하나의 unique한, 즉 유일한, 식별을 해주게 되는 key입니다. 예를 들어 보겠습니다. EMPLOYEE라는 엔티티가 있습니다. 그 엔티티들의 스키마는 EMPLOYEE(FName, MInit, LName, SSN, BDate, Address, Sex, Salary, SuperSSN, Dno)등의 특성을 가지고 있습니다. 여기에서 하나의 고용인을 표현할 수 있는 super key는 어떤 것이 있을까요? FName, MInit, LName, SSN이렇게도 하나를 식별할 수가 있습니다. SSN은 사회보장번호로 우리나라의 주민번호와 같습니다. 이렇게 하나의 tuple을 unique하게 뽑아 낼 수가 있으면은 그것이 super key입니다. 그렇다면 candidate key는 무엇일까요? Candidate key는 super key중에서 가장 minimalize한 key입니다. 그러니까 super key중에서 가장 규모가 작은 key입니다. 앞에서 보았던 EMPLOYEE의 가장 규모가 작은 key를 찾아보면 느낌이 올 것 같습니다. SSN은 발급받는 번호가 사람마다 다 다르게됩니다. 그렇기 때문에 하나의 tuple을 식별을 할 수가 있습니다. 그리고 하나의 attribue만 되는 최소 단위입니다. 그렇기 때문에 EMPLOYEE의 candidate key가 될 수가 있습니다.

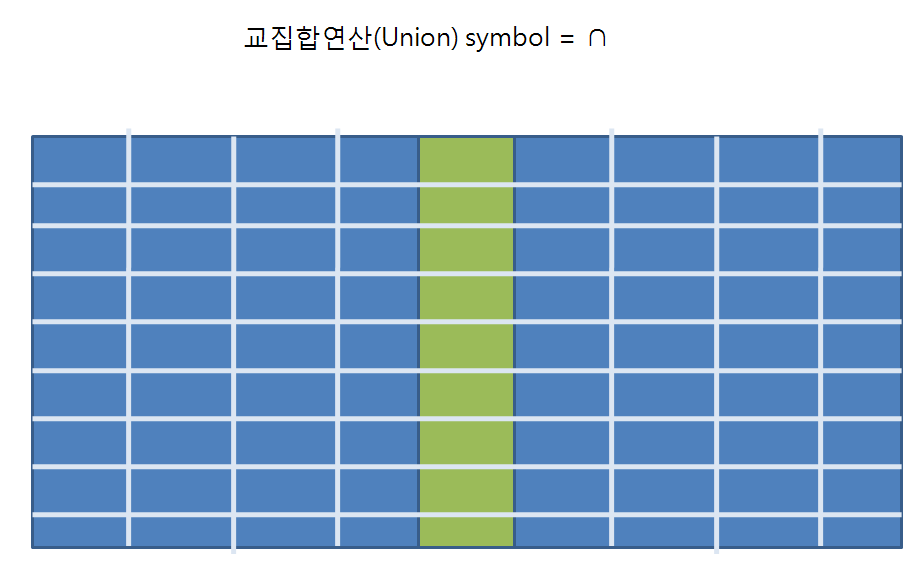
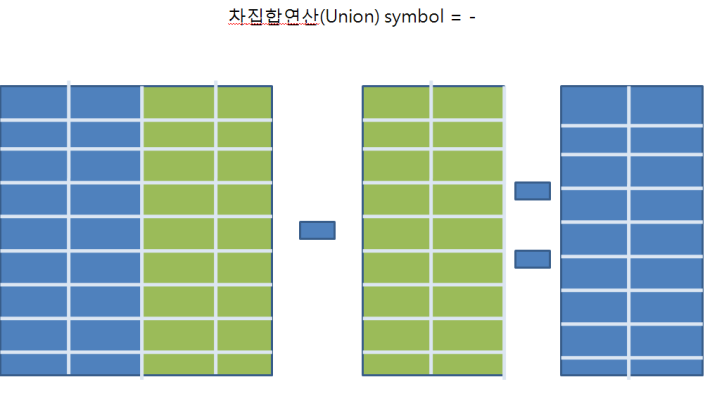
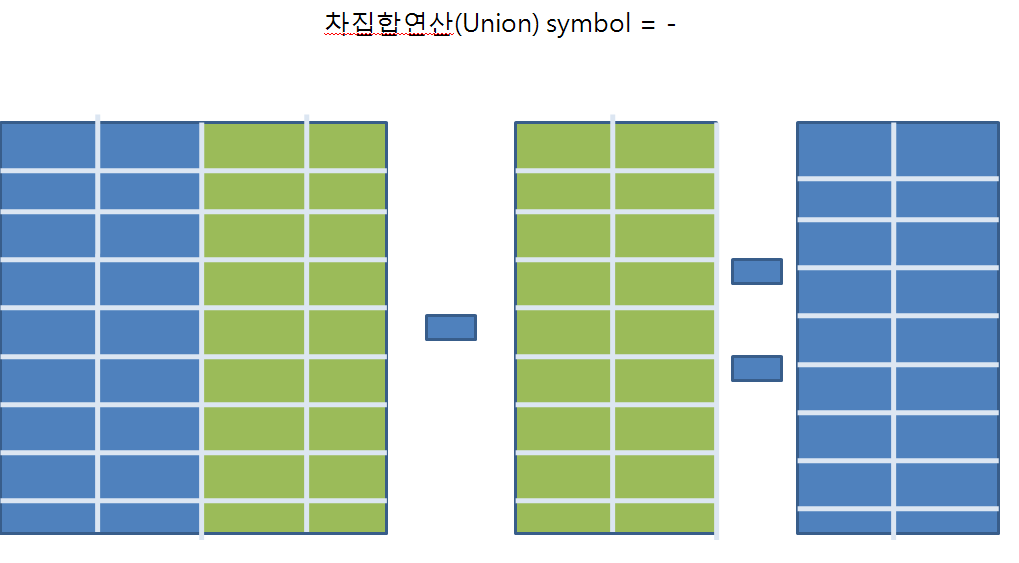
b)foreign key, referencing/referenced relation

foreign key란 무엇일까요? 일단은 단어의 뜻을 가지고 판단을 하는 것이 좋을 것 같습니다. Foreign key는 외부키라는 뜻 입니다. 외부에서는 키라고 하는 뜻입니다. 예를 들기 전에 외부에서 연결 상태가 있다고 하는 것에 관해서 확실한 정의가 필요합니다. Entity들 자체로는 아무런 의미가 없습니다. 바로 관계가 이어질 때에 그 의미가 있게 되는 것이지요. 그렇기 때문에 Entity들이 연결이 되기 시작하면은 참조에 대한 관계가 이루어지게 됩니다. A entity는 B entity를 참조하게 된다고 하겠습니다. 그렇다면 B entity는 A entity에 참조 당하게 됩니다. 그러면 예를 들어 보겠습니다. Student(st\_number,name,class\_name)가 있다고 하겠습니다. 둘 간의 관계를 보면서 참조하는 것과 참조를 당해야 관계에 대해서 설명을 할 수가 있습니다. 그렇기 때문에 하나의 entity를 더 만들도록 하겠습니다. Class(class\_name,total\_number)이렇게 엔티티가 정해져 있다고 하겠습니다. 그렇다면 Student entity는 Class entity와의 관계를 맺을 수가 있습니다. Class\_name이라는 attribute로 묶여있기 때문입니다. 그렇다면 참조관계는 어떻게 되게 될까요? 접근되는 관점에 관해서 생각을 해보아야 합니다. Class entity는 Student entity를 참조를 되게됩니다. 그리고 Student entity는 Class entity에 참조 하게게 됩니다.

c)basic/primitive relational operators vs. additional operators

operator라고 하는 것은 무엇일까요? 일단은 연산을 하게 되었다고 합시다. 그렇게 되면은 연산에 대해서 주어진 행동을 하게 됩니다. 주어진 행동이 마쳐지게 되면은 연산이 끝나게 됩니다. 연산을 하면은 최종적으로는 데이터가 변하게 됩니다. 물론 rename같이 연산 결과가 바뀌지 않는 것도 있습니다. 하지만 이름을 다시 갖는 것도 연산에 대한 결과입니다. 그렇다면은 데이터 베이스에서 연산이라고 하는 의미는 어떠한 의미를 가지게 되는 것일까요? 데이터베이스에서 연산이라고 하는 것은 두 개의 관계에 대해서 들어 온 결과를 가지고 새로운 결과를 나타내는 것이 됩니다. 그렇다면 basic/primitive relational operators라는 것을 무엇일까요? 기본적으로 연산의 행위가 한번에 끝마쳐 진다는 것입니다. 한 operator마다 한번의 연산에 결과를 가지게 되는 것입니다. 처음으로는 선택 연산을 보겠습니다. 선택 연산과 같은 경우에는

조건에 알맞은 가로의 instance를 가지고 오는 것입니다. 그렇기 때문에 새로 가져오는 tuple에 관해서 하나의 가로의 종적인 정보를 얻을 수가 있게 됩니다. 그렇다면 다음으로 추출연산을 보시게 되겠습니다. 추출연산과 같은 경우에는 세로의 attribute부분을 선택된 attribute를 제외하고 전부 삭제하여서 중복을 체크해서 attribute들을 돌려주게 됩니다. 다음으ㅡ로 합집합 연산과 같은 경우를 보겠습니다. 합집합 연산과 같은경우에는 두개의 relation을 하나로 붂에 주게 됩니다. 집합에서의 합집합을 생각하면 더 쉽게 생각 할 수가 있습니다. 그 다음으로는 교집합 연산이 있습니다. 교집합 연산은 각 relation의 공통적인 부분을 묶어 주어서 새로운 relation을 만들게 됩니다.

그 다음으로는 차집합 연산이 있을 수가 있습니다. 차집합 연산과 같은 경우에는 각각이 호환이 가능해야합니다. 각각의 정의된 스키마가 같아야 한다는 것입니다. 각각의 스키마가 같아야 차집합 연산을 할 수가 있습니다. 차집합 연산은 앞부분의 relation과 뒤의 부분의 relation부분중에서 공통적인 부분을 앞의 relation에서 제거를 하게 됩니다. 그 다음으로 카티션곱 연산이 있습니다. 두 relation의 튜풀을 전부다 곱해주게 되는 것입니다. 그림에서 두개짜리 튜풀과 뒤의 3개짜리 튜풀을 카티션 곱연산을 하게되어서 6개짜리 새로운 relation이 나오게 된 것을 보면을 이해 하실수 있을 것입니다. 그 다음으로 재명명 연산이 있습니다. 재명명연산은 결과에 이름이 없게 됩니다. 그것에 대해서 이름을 주게 되는 연산입니다. 기본 연산에 대해서 살펴 보았습니다. 그러면은 기본 연산과 추가 관계 대수연산은 무슨 차이가 있을까요 추가 관계 연산은 기본 연산과 다르게 여러 개의 기본 연산을 표현한 것입니다. 기본연산으로 모든 질의를 표현 할 수 있습니다. 하지만 기본대수 연산으로 모든 것을 표현하면은 표현식이 길어지게 됩니다. 추가관계 대수 연산을 가지고 길어진 표현식을 간단하게 할 수 있습니다.

d)natural join, theta join, outer join operation

join이라고 하면은 어떤 생각이 드시나요? 무엇인가 이어진다는 느낌이 들지 않습니까? join연산은 추가 관계 대수 연산입니다. 여러가지 기본연산을 한가지 기호로 표현한 것입니다. Natural join과 같은 경우에는 같은 attribute가 겹치는 항목에 대해서 하나의 새로운 테이블을 만들어 주게됩니다. Theta join과 같은 경우에는 조건을 주어서 그것을 조인을 할 수가 있게 됩니다. Outer join과 같은 경우에는 right outer join, left outer join, full outer join 세가지 outer join이 있습니다. 각각 기호에서 열린부분을 따라서 outer join의 tuple의 개수가 정에 지게 됩니다. Join을 할때에 값이 없게 된다면 null 값이 들어가게 됩니다. 정보를 잃지 않으면서 새로운 관계를 맺고 싶을 때에 outer join을 주로 사용하게 됩니다.

e)null value

null이라고 하는 것은 정의 되어 있지 않거나, 알 수 없는 값이 들어가 있는 것입니다. 표면적으로 알고 있는 것은 이렇게 정의 되어있습니다. 그렇다면 데이터 배이스에서는 어떠한 방식으로 사용되는 것일까요? Null 값과 같은경우에는 데이터베이스 attribute에 값이 존재 하지 않을때에 그 것을 null이라고 합니다. 또는 알 수 없는 값이나 그 값이 존재 하지 않을 때에 null로 처리하게 됩니다.

**2.**The university database contains the following relations:

STUDENT(Name, StudentNumber, Class, Major)

COURSE(CourseName, CourseNumber, CreditHours, Department)

PREREQUISITE(CourseNumber, PrerequisiteNumber)

SECTION(SectionId, CourseNumber, Semester, Year, Instructor)

GRADE\_REPORT(StudentNumber, SectionId, Grade)

Assumptions:

■ Class attribute of the STUDENT relation uses integers to represent different classes of students. For example, seniors would be '4'.

■ Instructor attribute of the SECTION relation uses instructors' last name to represent different instructors.

■ Use renaming and assignment operations if necessary.

Formulate the following queries in relational algebra.

a)Retrieve names of all senior students majoring in 'ce' (computer engineering).

ΠName(σClass=4^Major=”ce”(SECTION)

b)Retrieve names of all courses taught by Professor 'Chun' in 2009 and 2010.

ΠCourseName(σYear=2009^Year=2010^Instructor=”Chun”(SECTION))

c)Retrieve names and majors of all straight 'A' students (students who have a grade 'A' in all their courses.)

ΠName,StudentNumber(STUDENT)ΠStudentNumber(σGrade”A”(GRADE\_REPORT)

d)Retrieve names and majors of all students who do not have a grade 'A' in any of their courses.

NA<-ΠStudentNumber(GRADE\_REPORT)-ΠStudentNumber(σGrade”A”(GRADE\_REPORT)

ΠName,Major(NA▷◁ STUDENT)

e) Delete all courses offered by 'Computer Software' department and associated prerequisites and sections.

COURSE<-COURSE - (σDepartment=”Computer software”(COURSE))

PREREQUISITE<-PREREQUISITE–ΠCourseNumber,PrerequisiteNumber(σDepartment=”Computer software”(COURSE ▷◁ PREQUISITE)

SECTION <- SECTION – ΠSectionId,CourseNumber,Semester,Year,Instructor(σDepartment=”Computer software”(COURSE ▷◁ SECTION)

**3.** The company database contains following relations:

EMPLOYEE(FName, MInit, LName, SSN, BDate, Address, Sex, Salary, SuperSSN, Dno)

DEPARTMENT(DName, DNumber, MgrSSN, MgrStartDate)

DEPT\_LOCATIONS(DNumber, DLocation)

PROJECT(PName, PNumber, PLocation, DNum)

WORKS\_ON(ESSN, PNO, Hours)

DEPENDENT(ESSN, DependentName, Sex, BDate, Relationship)

Formulate the following queries in relational algebra.

a)Retrieve names of employees in department '5' who work more than 10 hours per week on the 'X' project.

PW <- σPName=”X”^Hours>=10(PROJECT ▷◁ PNumber=PNO WORKS\_ON)

ΠFName,LName(((PW ▷◁ SSN=ESSNEMPLOYEE ) )))

b)Retrieve names of employees who have at least one dependent with the same first name as themselves.

ΠFName,LName(EMPLOYEE ▷◁ ESSN=SSN^FName=DependentName DEPENDENT)

c)Retrieve names of employees who are directly supervised by 'Jonghoon Chun'.

ΠFName,LName (EMPLOYEE▷◁ SuperSSN=SSN(ΠSSN (σFName=”Chun”^LName=”Jonghoon” (EMPLOYEE))))

d)For each project, list the project name and the total hours per week (by all employees) spent on that project.

Π,PName,total\_hour (PROJECT▷◁ PNO=PNumber(PnoGSum(Hours)as total\_hour (WORKS\_ON)))

e)Retrieve names of employees who work on every project.

ΠFName,LName (ΠESSN,PNO(WORKS\_ON)ΠPnumber(PROJECT))

f)Retrieve names of employees who do not work on any project.

ΠFName,LName (EMPLOYEE(ΠSSN(EMPLOYEE) - ΠESSN(WORKS\_ON)))

g)For each department, retrieve the department names and the average salary of employees working in that department.

ΠDname,avg\_salary (DnoGAvg(Salary)as avg\_salary (EMPLOYEE))

h)Retrieve the average salary of all female employees.

GAvg(Salary)as avg\_salary (σsex=”F” (EMPLOYEE))

i)Find the names and addresses of employees who work on at least one project located in 'Seoul' but whose department has no location in 'Seoul'.

PS<-ΠESSN (WORKS\_ON▷◁ PNO=PNumber(σPLocation=”Seoul” (PROJECT)))

NLS<- ΠSSN (EMPLOYEE▷◁ (DNO=DNumberΠDNumber(σDLocation”Seoul” (DEPARTMENT))))

ΠFName,LName,Address (EMPLOYEE(PS-NLS))

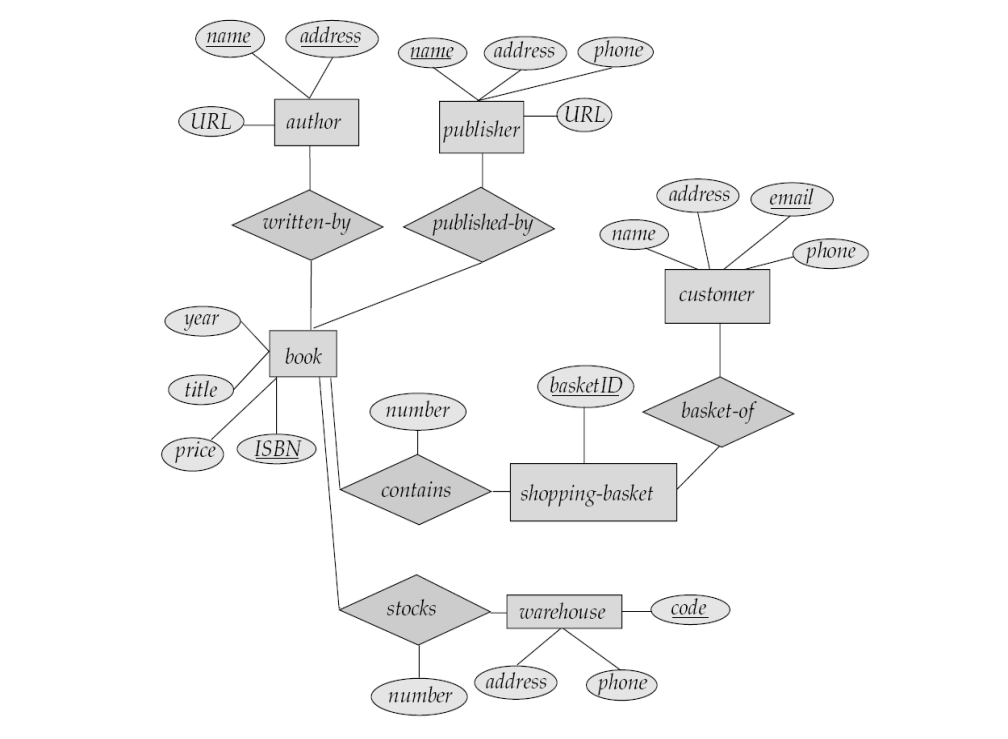
j)List the last names of department managers who have no dependents.

ΠFName,LName (EMPLOYEE(ΠMgrSSN (DEPARTMENT) - ΠESSN (DEPENDENT)))

k)Find the names and addresses of employees who work on a project in one city but whose department has no location in that city.

ΠFName,LName,Address (EMPLOYEE▷◁ (DNO=DNumberΠDNumber(σDLocationPLocation (DEPARTMENT)))

**4.**Convert the following entity-relationship diagram, which models an online book store, into a relational schema. Document all necessary assumptions that you make about the situation.



Author(name,address,URL)

Publisher(name,address,phone,URL)

Book(ISBN,author\_name,publisher\_name,year,title,price)

Shopping-basket(basketID, Book\_ISBN, Customer\_email,Book\_Number)

Costomer(email,name,address,phone)

Warehouse(code,Book\_ISBN,address,phone,Book\_Number)

기본적으로 Author Publisher, Customer, Warehouse는 정해진 Attribute를 사용해서 relational schema를 구성하였습니다. 그 다음으로 동사로 이어지는 부분으로 처음으로 이어지는 단계는 이어지는 관계를 표현하기 위해서 연결되는 entity안에 attribute안에 넣어 놓았습니다. author\_name, publisher\_name, Book\_ISBN등과 같이 앞부분에 entity명을 변수명 앞에 붙여서 고유의 속성이 아닌 것을 알 수 있도록 하였습니다. 참조하는 entity를 알 수가 있도록 했습니다. 특이하게 Shopping-basket의 book이 담겨있는 고객의 바구니 임으로 두개의 관계를 참조할 수 있도록 고객의 email과 책의 ISBN을 넣었습니다. 그리고 Book의 개수를 알 수 있도록 Book\_Number도 넣었습니다. 각 relation의 관계를 끊어 지지 않도록 relation-schema를 구성 하였습니다.

**5.**Consider the resulting entity-relationship diagram of homework assignment 1 #6-a). Convert it into a relational schema. Document all necessary assumptions that you make about the situation.

Passenger(Name,Address,Phone)

Departure(Passenger\_name, Flight\_number,Date,Employee\_No)

Flight(Number,Origin,Dest,Arr\_Time,Dep\_Time)

Employee(Employee\_No,Name,Address,Salary)

Pliot(Pliot\_No,Pliot\_Name,Pliot\_Address,Pliot\_Salary,Plane\_No)

Plane(Model\_No,Maker)

Aircraft(Plane\_No,Serial\_No)

기본적으로 Passenger Flight, Depature, Employee, Plane, Aircraft는 정해진 Attribute를 사용해서 relational schema를 구성하였습니다. 그 다음으로 동사로 이어지는 부분으로 처음으로 이어지는 단계는 이어지는 관계를 표현하기 위해서 연결되는 entity안에 attribute안에 넣어 놓았습니다. 앞부분에 entity명을 변수명 앞에 붙여서 고유의 속성이 아닌 것을 알 수 있도록 하였습니다. 참조하는 entity를 알 수가 있도록 했습니다. entity종류가 분화 되어 나온 것도 앞에 자신의 entity변수 명을 붙여주어서 다른 분화된 entity라는 것을 알 수가 있도록 했습니다.