**병원 정보 DB 구축**

2013312343 이상헌

1. 설명

- 병원 정보 시스템에 대한 DB를 설계 및 MySQL을 이용하여 컴퓨터 상에 구축한다.

2. DB 요구사항 분석 ( **Entity**, *Attribute*, Constraint, ***Relationship type*** 로 표기. )

1) **부서** (Department)

• 병원은 여러 종류의 **부서**를 가진다.

• 각 **부서**는 고유한 *부서명*을 통해 식별된다.

• 병원은 각 **부서**에 대해 *부서명*, *위치 번호* 및 *전화번호*를 저장한다.

• 한 **부서**는 여러 명의 **의사**와 **간호사**를 ***포함한다***.

• 부서에 속한 의사들 중 한 명의 **의사**가 그 부서의 ***책임자가 된다*.**

2) **의사** (Doctor)

• **의사**는 고유한 사원 번호를 통해 식별된다.

• 병원은 **의사**의 *이름, 사원 번호, 재직 년 수* 및 *가능한 진료 요일*을 저장한다..

• 한 **의사**는 여러 **환자**의 ***진료를 보거나*** **수술**을 ***집도할 수 있다*.**

• 한 **의사**는 자신이 속한 **부서**의 ***책임자가 되거나 되지 않을 수 있다.***

3) **간호사** (Nurse)

• **간호사**는 고유한 *사원 번호*를 통해 식별된다.

• 병원은 **간호사**의 *이름*, *사원 번호*, *재직 년 수*를 저장한다.

• **간호사**는 여러 **병실** 및 **환자**들을 ***관리한다*.**

4) **환자** (Patient)

• **환자**는 고유한 *주민등록번호*를 통해 식별된다.

• 병원은 **환자**의 *이름, 주민등록번호, 나이, 성별, 보험 코드* 등을 저장한다.

• **환자**는 여러 **수술**을 ***받거나***, **의사**에게 ***진료를 받을 수 있다.***

• **환자**는 한 **병실**에 ***입원할 수 있고***, **간호사**에 의해 ***관리된다.***

5) **수술** (Surgery)

• **수술**은 고유한 *수술 번호*를 통해 식별된다.

• 병원은 특정 **수술**에 대해 *수술명*과 *수술 번호*를 저장한다.

• 특정 **수술**에 대한 *수술 날짜, 수술 시간* 및 *환자의 상태*가 주어진다.

• **수술**은 한 **환자**에게 ***행해지며***, 여러 **의사**에 의해 ***집도 된다.***

6) **병실** (Room)

• **병실**은 고유한 *병실 번호*를 통해 식별된다.

• 병원은 **병실**의 *병실 번호* 및 병실의 *전화번호*를 저장한다.

• **병실**은 한 **간호사**에 의해 ***관리되며***, 여러 **환자**들을 ***수용한다.***

3. 각 Entity에 대한 세부 Attributes ( **Entity**, Primary key 로 표기. )

1) **부서(DEPARTMENT)** = 부서명(dept-name), 위치 번호(location-num), 전화번호(telephone)

2) **의사(DOCTOR)** = 사원 번호(doctor-id), 이름(name), 재직 년 수(work-year), 진료 요일(treatment-day)

3) **간호사(NURSE)** = 사원 번호(nurse-id), 이름(name), 재직 년 수(work-year)

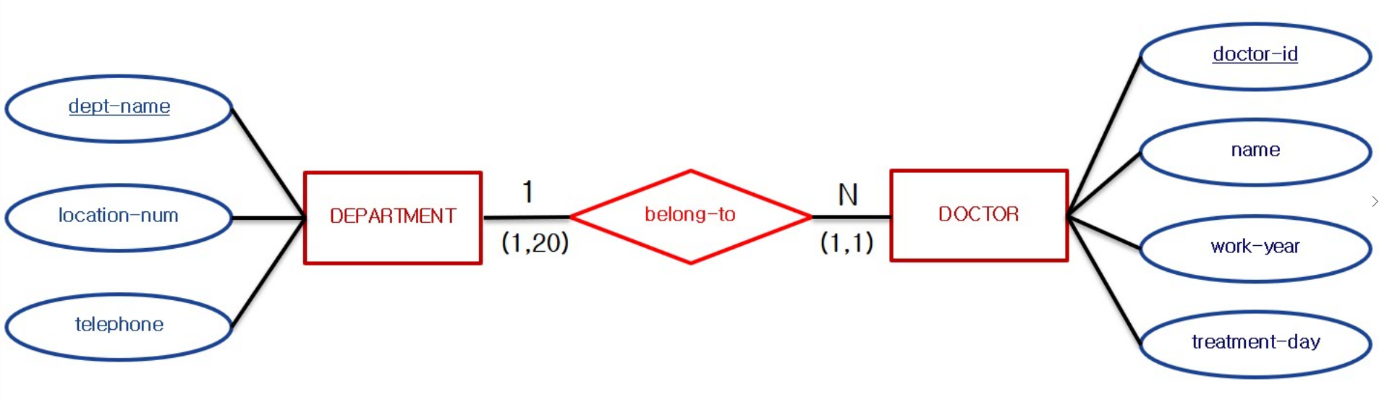
4) **환자(PATIENT)** = 주민등록번호(RRN), 환자 이름(patient-name), 나이(age), 성별(sex), 보험 코드(insurance-code)

5) **수술(SURGERY)** = 수술 번호(surgery-num), 수술명(surgery-name)

6) **병실(ROOM)** = 병실 번호(room-num), 병실 전화번호(room-telephone)

4. 각 Entity에 대한 Relationship

1) belong-to Relationship

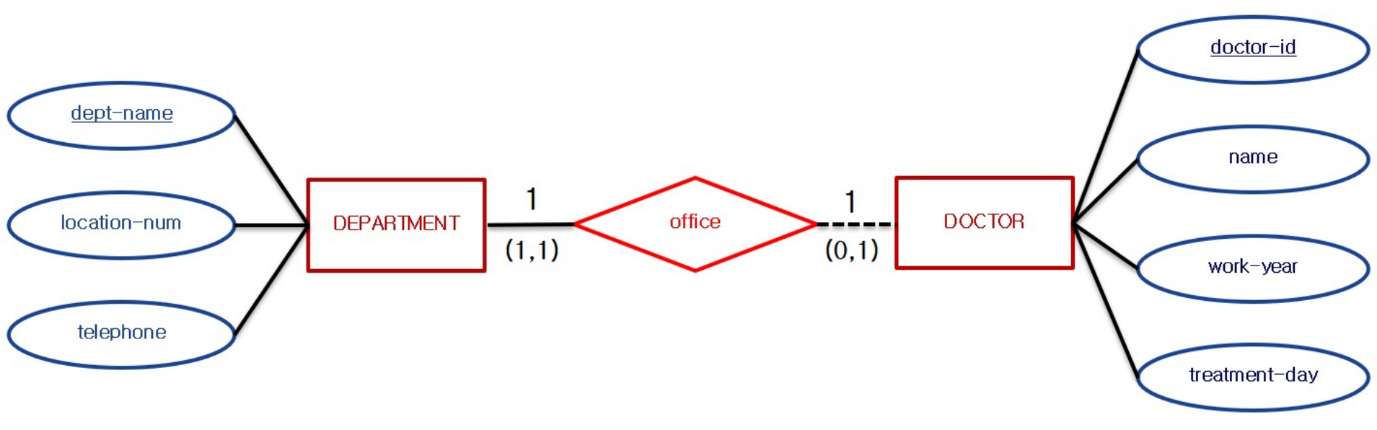


• 부서(DEPARTMENT)와 의사(DOCTOR) 사이 소속 관계를 나타낸다.

• 한 명의 의사는 반드시 단 하나의 부서에 소속되어 있어야 하고, 한 부서는 최대 20명의 의사를 보유할 수 있다.

• 부서 : 의사 = 1 : N의 관계.

2) Office(부서의 책임자) Relationship

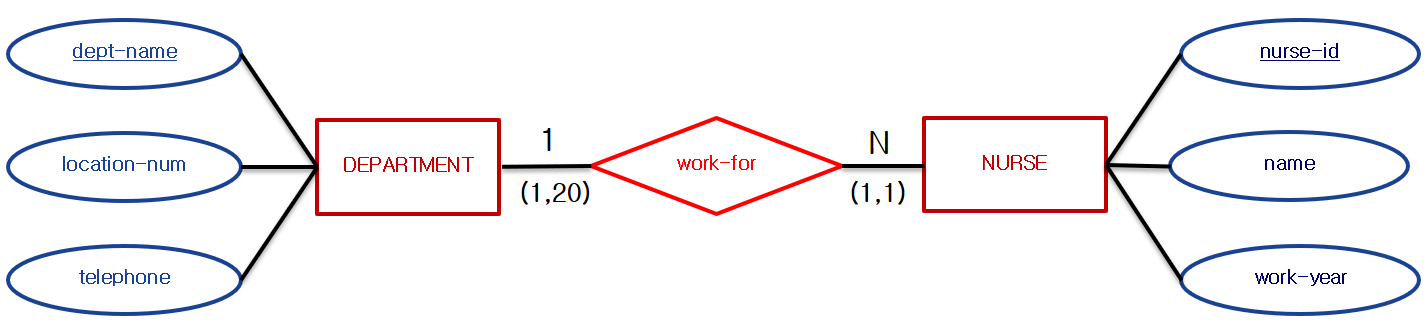


• 부서(DEPARTMENT)와 의사(DOCTOR) 사이 책임 관계를 나타낸다.

• 특정 부서에 속한 한 명의 의사는 그 부서의 책임자가 될 수 있으며, 각 부서는 반드시 단 한 명의 책임자가 되어야 한다.

• 부서 : 의사(책임자) = 1 : 1의 관계.

3) work-for Relationship

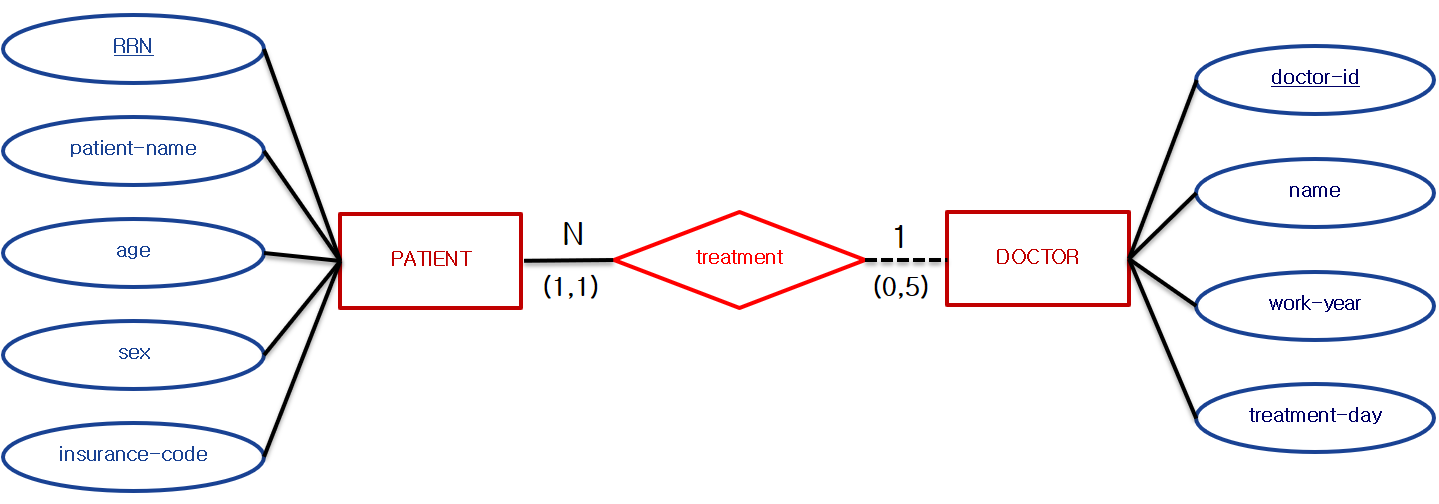


• 부서(DEPARTMENT)와 간호사(NURSE) 사이 소속 관계를 나타낸다.

• 한 명의 간호사는 반드시 단 하나의 부서에 소속되어 있어야 하고, 한 부서는 최대 20명의 간호사를 보유할 수 있다.

• 부서 : 간호사 = 1 : N의 관계.

4) treatment Relationship

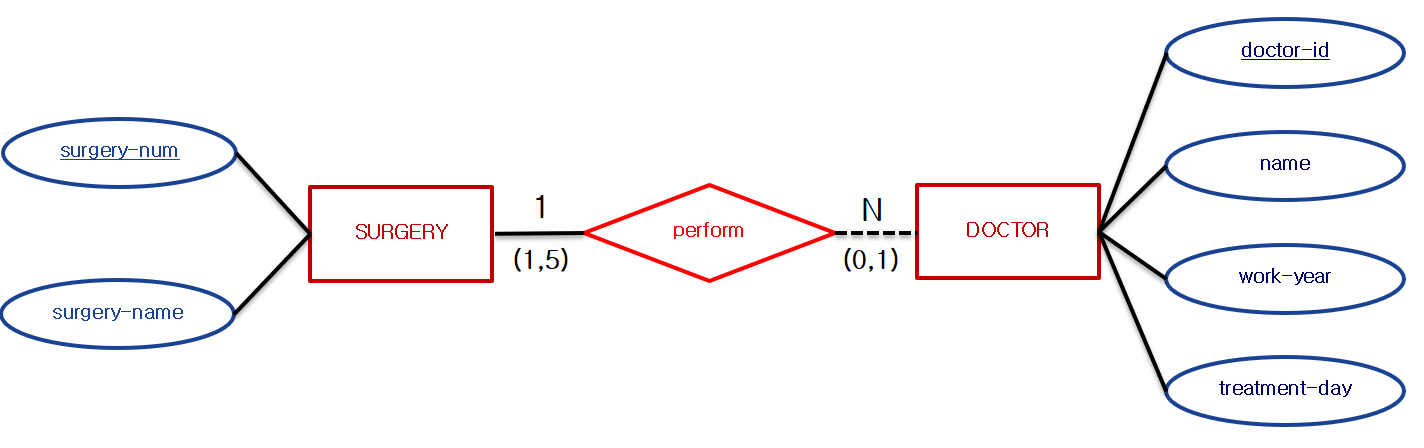


• 환자(PATIENT)와 의사(DOCTOR) 사이 진료 관계를 나타낸다.

• 한 명의 환자는 반드시 단 한 명의 의사에게 진료를 보아야 하고, 한 의사는 최대 5명의 환자의 진료를 볼 수 있다.

• 환자 : 의사 = N : 1의 관계.

5) perform Relationship

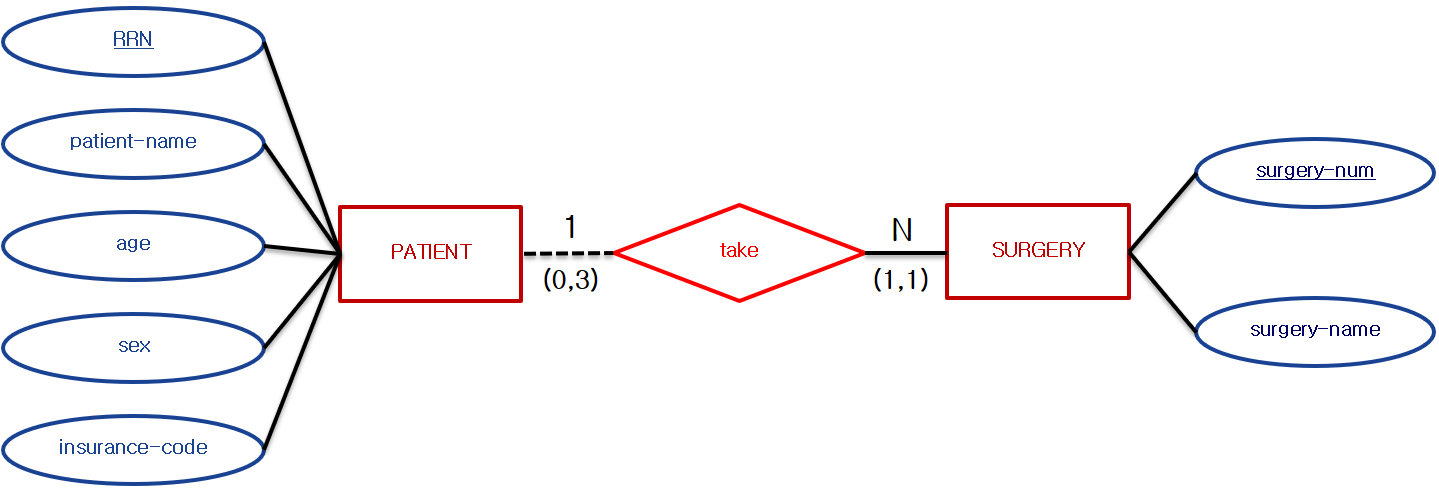


• 수술(SURGERY)과 의사(DOCTOR) 사이 집도 관계이다.

• 한 의사는 최대 1개의 수술을 집도할 수 있고, 한 수술은 최대 5명 이상의 의사에게 집도 되어야 한다.

• 수술 : 의사 = 1 : N의 관계.

6) take Relationship

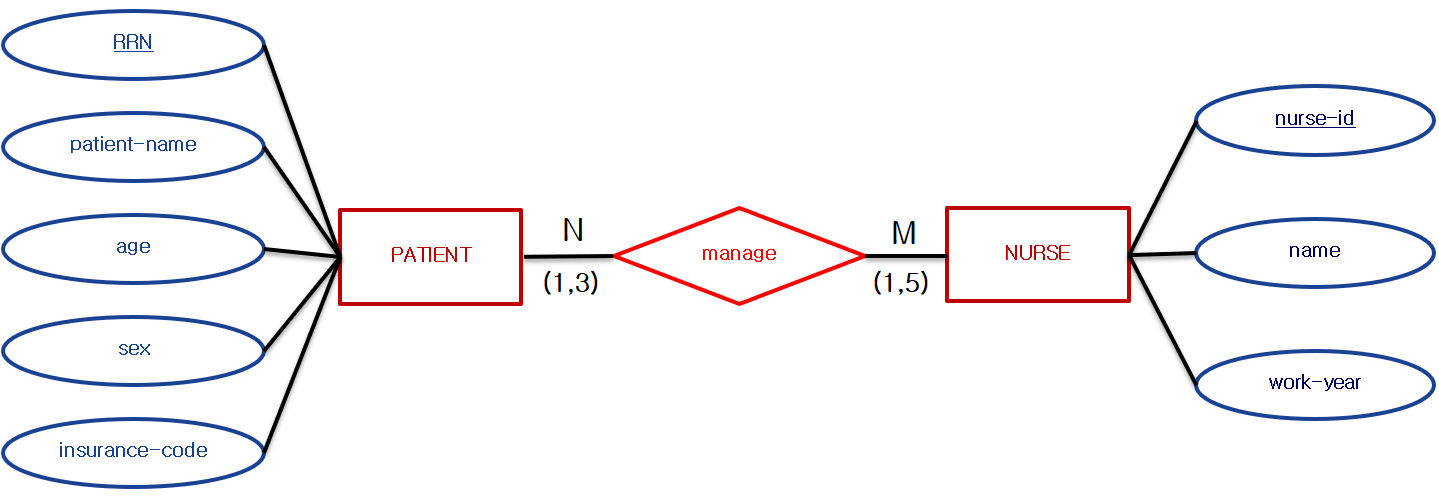


• 환자(PATIENT)와 수술(SURGERY) 사이 (수술을)받는 관계이다.

• 한 수술은 반드시 단 한 명의 환자를 대상으로 해야 하고, 한 환자는 최대 3번의 수술을 받을 수 있다.

• 환자 : 수술 = 1 : N의 관계.

7) manage Relationship

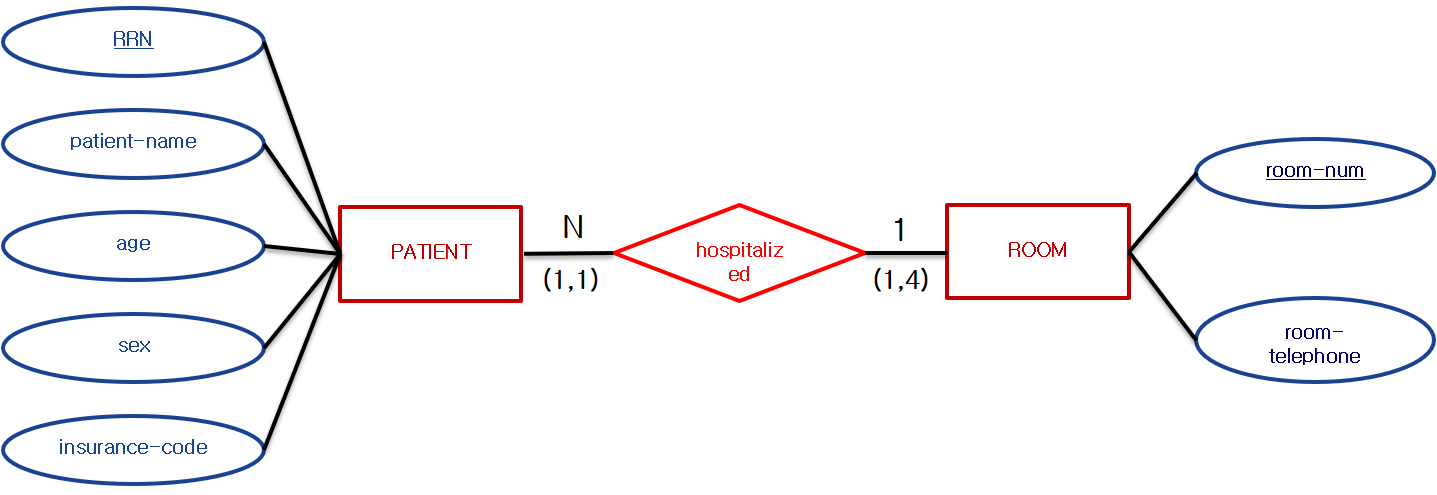


• 환자(PATIENT)와 간호사(NURSE) 사이 관리 관계이다.

• 한 명의 환자는 최대 5명의 간호사에게 관리 받아야 되며, 한 명의 간호사는 최대 3명의 환자를 관리해야 한다.

• 환자 : 간호사 = N : M의 관계.

8) hospitalized Relationship

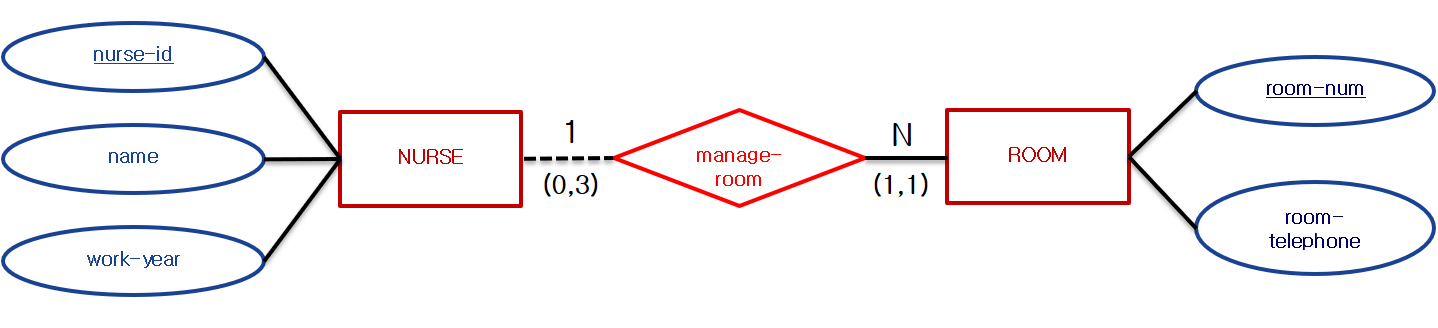


• 환자(PATIENT)와 병실(ROOM) 사이 입원 관계이다.

• 한 명의 환자는 반드시 단 하나의 병실에 입원해야 하고, 한 개의 병실은 최대 4명의 환자를 입원시키고 있어야 한다. 비어있는 병실은 존재하면 안된다.

• 환자 : 병실 = N : 1의 관계.

9) manage-room Relationship

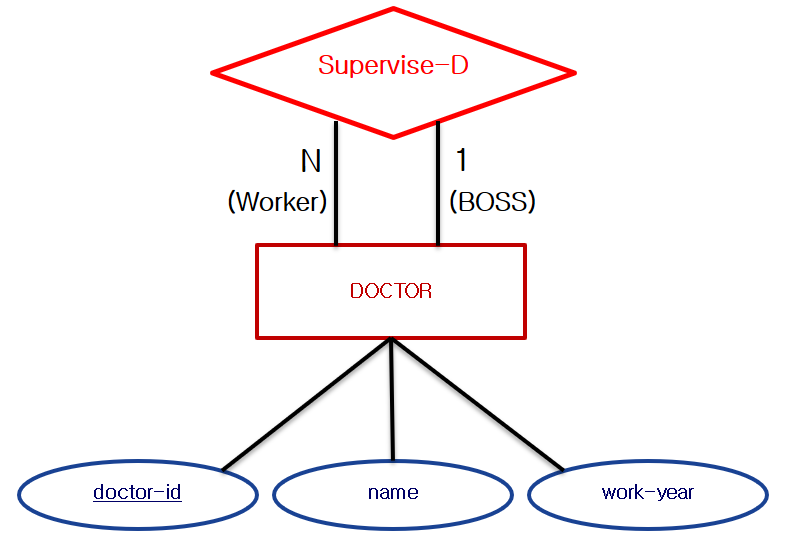
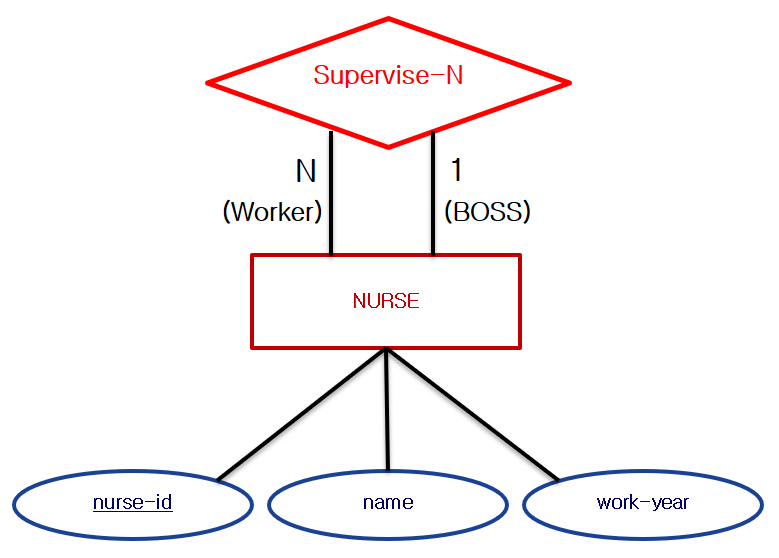


• 간호사(NURSE)와 병실(ROOM) 사이 관리 관계이다.

• 한 간호사는 최대 3개의 병실을 관리할 수 있고, 한 병실은 반드시 단 한 명의 간호사에게 관리 되어야 한다.

• 간호사 : 병실 = 1 : N의 관계.

10) Recursive Relationship of NURSE and DOCTOR



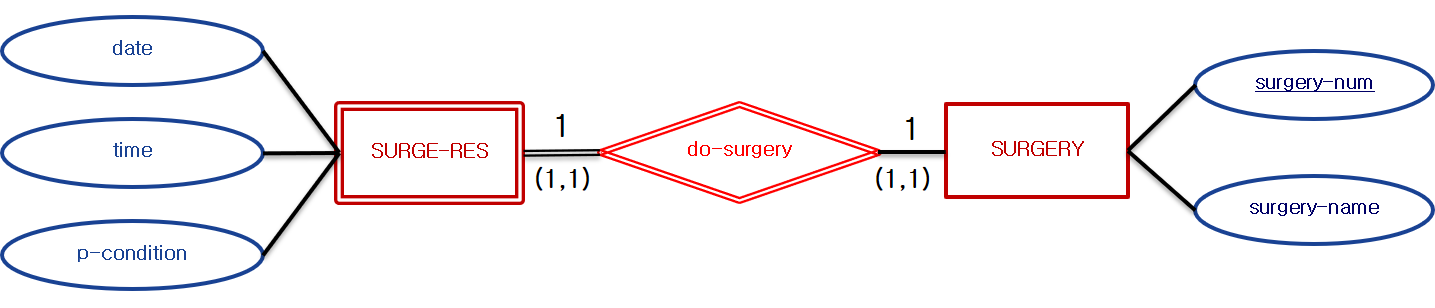
• 한 명의 간호사는 반드시 단 한 명의 상사가 있고, 여러 명의 부하 직원을 가진다.

• 한 명의 의사는 반드시 단 한 명의 상사가 있고, 여러 명의 부하 직원을 가진다.

• 간호사 : 상사(간호사) = N : 1의 Recursive 관계.

• 의사 : 상사(의사) = N : 1의 Recursive 관계.

11) do-surgery Relationship



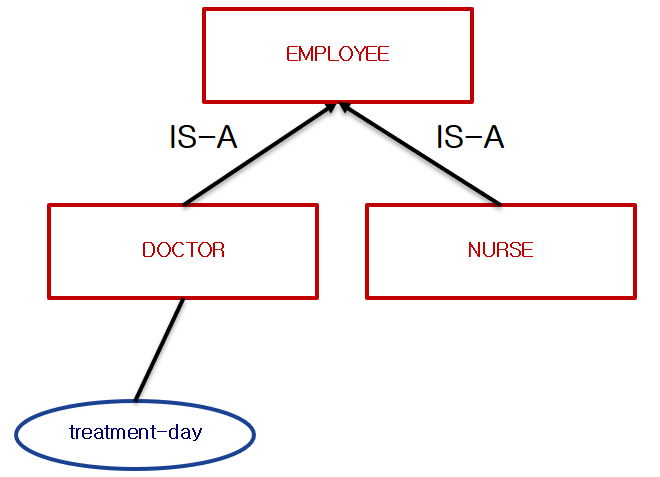
• 수술(SURGERY)과 수술결과(SUREG-RES) 사이 관계를 나타낸다.

• SURGE-RES는 수술의 결과에 대한 상세 정보를 담고 있는 entity로, Weak Entity Type이다.

• 한 수술에 대해 정확히 하나의 수술 결과 데이터를 가지고 있어야 한다.

• 수술 : 수술결과 = 1 : 1의 관계.

12) IS-A Relationship of NURSE and DOCTOR.



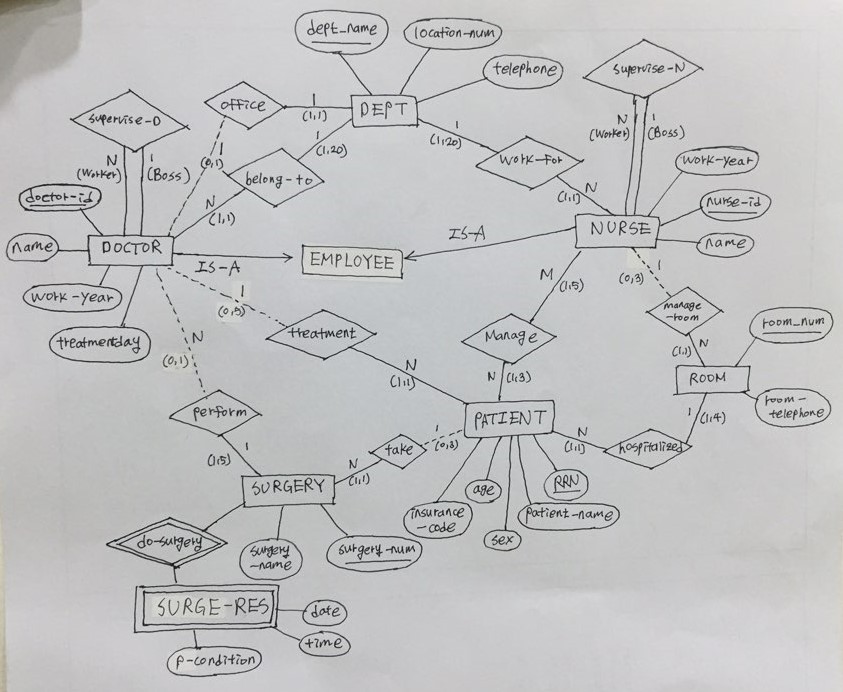
• 의사(DOCTOR)와 간호사(NURSE)의 IS-A 관계를 나타낸다.

• EMPLOYEE는 사원의 기본 정보를 담은 entity로, id, name, work-year를 attribute로 가지고 있다.

• DOCTOR와 NURSE는 EMPLOYEE를 IS-A Relationship으로 상속받는 entity이다.

• DOCTOR에는 그 entity만의 attribute인 treatment-day가 존재한다.

5. ER Diagram



• DEPARTMENT entity의 이름을 DEPT로 축약하였다.

6. Relational Schema 작성 ( Primary key, Foreign key 로 표기. 겹치는 경우 PK를 따로 표기. )

(1) DEPT ( dept-name, location-num, telephone, officer-id )

• 모든 부서는 단 한 명의 책임자를 가지고 있다.

• DEPT : DOCTOR = 1(total) : 1(partial) Relationship 이므로 partial-side의 PK ( doctor-id )를 total-side의 FK ( officer-id )로써 attribute에 추가한다.

• FD : dept-name -> { location-num, telephone, officer-id }

(2) DOCTOR ( doctor-id, dept-name, super-d-id, surgery-num ) PK = { doctor-id }

• DOCTOR entity는 EMPLOYEE entity의 모든 attribute를 상속 받는다. 상속 관계는 DOCTOR entity의 PK인 doctor-id를 이용한다. 즉, doctor-id는 상속에서의 FK로 작용한다.

• DOCTOR : DEPT = N : 1 Relationship 이므로 1-side의 PK ( dept-name )를 N-side의 FK로써 attribute에 추가한다.

• DOCTOR의 supervise-D Relationship은 worker : boss = N : 1 relationship이므로 1-side의 PK ( doctor-id )를 N-side의 FK ( super-d-id )로써 attribute에 추가한다.

• DOCTOR : SURGERY = N : 1 Relationship 이므로 1-side의 PK ( surgery-num )를 N-side의 FK로써 attribute에 추가한다.

• FD : doctor-id -> { dept-name, super-d-id, surgery-num }

(3) NURSE ( nurse-id, dept-name, super-n-id ) PK = { nurse-id }

• NURSE : DEPT = N : 1 Relationship 이므로 1-side의 PK ( dept-name )를 N-side의 FK로써 attribute에 추가한다.

• NURSE의 supervise-N Relationship은 worker : boss = N : 1 relationship이므로 1-side의 PK ( nurse-id )를 N-side의 FK ( super-n-id )로써 attribute에 추가한다.

• FD : nurse-id -> { dept-name, super-n-id }

(4) EMP ( id, name, work-year )

• EMP entity는 DOCTOR와 NURSE entity가 상속받는 attribute가 존재하는 entity로, PK인 id를 통해 DOCTOR와 NURSE에게 모든 attribute를 상속한다.

• FD : id -> { name, work-year }

(5) MANAGE-N-P ( nurse-id, patient-RRN ) PK = { nurse-id, patient-RRN }

• NURSE와 PATIENT의 manage 관계는 NURSE : PATIENT = M : N Relationship이다. 따라서 Relational Schema로 바꿀 때에는 이 관계를 위한 새로운 entity를 생성해야 한다. 생성된 entity에는 각각 기존 entity에서의 PK를 FK로써 attribute에 추가하고, M : N 관계이므로 PK는 두 attribute 모두가 된다.

• FD : { nurse-id, patient-RRN } -> { nurse-id, patient-RRN }

(6) PATIENT ( RRN, patient-name, age, sex, insurance-code, doctor-id, room-num)

• PATIENT : DOCTOR = N : 1 Relationship 이므로 1-side의 PK ( doctor-id )를 N-side의 FK로써 attribute에 추가한다.

• PATIENT : ROOM = N : 1 Relationship 이므로 1-side의 PK ( room-num )를 N-side의 FK로써 attribute에 추가한다.

• FD : RRN -> { patient-name, age, sex, insurance-code, doctor-id, room-num }

(7) ROOM ( room-num, room-telephone, nurse-id )

• ROOM : NURSE = N : 1 Relationship 이므로 1-side의 PK ( nurse-id )를 N-side의 FK로써 attribute에 추가한다.

• FD : room-num -> { room-telephone, nurse-id }

(8) SUREGERY ( surgery-num, surgery-name, patient-RRN )

• SURGERY : PATIENT = N : 1 Relationship 이므로 1-side의 PK ( RRN )를 N-side의 FK ( patient-RRN )로써 attribute에 추가한다.

• FD : surgery-num -> { surgery-name, patient-RRN }

(9) SURGE-RES ( surgery-num, date, time, p-condition ) PK = { surgery-num }

• SURGE-RES 는 Weak entity 이다.

• SURGERY : SURGE-RES = 1 : 1 Relationship 이므로 1-side의 PK ( surgery-num ) 를 가지고 와서 SURGE-RES의 PK로 사용한다.

• FD : surgery-num -> { date, time, p-condition }

7. Normalization 정의

• 각 Relational Schema가 Normal Forms ( 1NF, 2NF, 3NF, BCNF )를 만족하는지를 확인하고, 만족하지 않는 경우에 대해서 정규화를 통해 만족하도록 하는 작업이다. 아래는 각 Normal Forms의 정의이다.

• First Normal Form (1NF) : A relation R is 1NF if (1) each tuple must be identified by primary key; (2) each attribute must have atomic (= only one) value.

• Second Normal Form (2NF) : A relation R is 2NF if any non-prime attribute is not partially dependent of any key.

• Third Normal Form (3NF) : A relation R is 3NF if any non-prime attribute is not transitive dependent of any key. 즉, 모든 FD : X -> A에 대해서 X가 super key 이거나, A가 prime attribute (key를 구성하는 attribute) 이면 3NF를 만족한다.

• Boyce Codd Normal Form (BCNF) : A relation R is BCNF if for every FD : X->A, X is a super key.

8. Normalization 확인

(1) DEPT Table

• key = { dept-name }

• 1NF : PK에 의해 각 tuple이 구분되어진다. 또한 각 attribute는 atomic value 이다. 따라서 만족한다.

• 2NF : 모든 non-prime attribute들이 기본 key에 함수 부분 종속이 되지 않는다. (non-partially dependent) 따라서 만족한다.

• 3NF : 모든 non-prime attribute들이 기본 key에 함수 이행 종속이 되지 않는다. (not transitive dependent) 따라서 만족한다.

• BCNF : 모든 FD의 결정자(X)가 super-key이다. 따라서 만족한다.

(2) DOCTOR Table

• key = { doctor-id }

• 1NF : PK에 의해 각 tuple이 구분되어진다. 또한 각 attribute는 atomic value 이다. 따라서 만족한다.

• 2NF : 모든 non-prime attribute들이 기본 key에 함수 부분 종속이 되지 않는다. (non-partially dependent) 따라서 만족한다.

• 3NF : 모든 non-prime attribute들이 기본 key에 함수 이행 종속이 되지 않는다. (not transitive dependent) 따라서 만족한다.

• BCNF : 모든 FD의 결정자(X)가 super-key이다. 따라서 만족한다.

(3) NURSE Table

• key = { dept-name }

• 1NF : PK에 의해 각 tuple이 구분되어진다. 또한 각 attribute는 atomic value 이다. 따라서 만족한다.

• 2NF : 모든 non-prime attribute들이 기본 key에 함수 부분 종속이 되지 않는다. (non-partially dependent) 따라서 만족한다.

• 3NF : 모든 non-prime attribute들이 기본 key에 함수 이행 종속이 되지 않는다. (not transitive dependent) 따라서 만족한다.

• BCNF : 모든 FD의 결정자(X)가 super-key이다. 따라서 만족한다.

(4) EMP Table

• key = { id }

• 1NF : PK에 의해 각 tuple이 구분되어진다. 또한 각 attribute는 atomic value 이다. 따라서 만족한다.

• 2NF : 모든 non-prime attribute들이 기본 key에 함수 부분 종속이 되지 않는다. (non-partially dependent) 따라서 만족한다.

• 3NF : 모든 non-prime attribute들이 기본 key에 함수 이행 종속이 되지 않는다. (not transitive dependent) 따라서 만족한다.

• BCNF : 모든 FD의 결정자(X)가 super-key이다. 따라서 만족한다.

(5) MANAGE-N-P Table

• key = { nurse-id, patient-RRN }

• 1NF : PK에 의해 각 tuple이 구분되어진다. 또한 각 attribute는 atomic value 이다. 따라서 만족한다.

• 2NF : 모든 non-prime attribute들이 기본 key에 함수 부분 종속이 되지 않는다. (non-partially dependent) 따라서 만족한다.

• 3NF : 모든 non-prime attribute들이 기본 key에 함수 이행 종속이 되지 않는다. (not transitive dependent) 따라서 만족한다.

• BCNF : 모든 FD의 결정자(X)가 super-key이다. 따라서 만족한다.

(6) PATIENT Table

• key = { RRN }

• 1NF : PK에 의해 각 tuple이 구분되어진다. 또한 각 attribute는 atomic value 이다. 따라서 만족한다.

• 2NF : 모든 non-prime attribute들이 기본 key에 함수 부분 종속이 되지 않는다. (non-partially dependent) 따라서 만족한다.

• 3NF : 모든 non-prime attribute들이 기본 key에 함수 이행 종속이 되지 않는다. (not transitive dependent) 따라서 만족한다.

• BCNF : 모든 FD의 결정자(X)가 super-key이다. 따라서 만족한다.

(7) ROOM Table

• key = { room-num }

• 1NF : PK에 의해 각 tuple이 구분되어진다. 또한 각 attribute는 atomic value 이다. 따라서 만족한다.

• 2NF : 모든 non-prime attribute들이 기본 key에 함수 부분 종속이 되지 않는다. (non-partially dependent) 따라서 만족한다.

• 3NF : 모든 non-prime attribute들이 기본 key에 함수 이행 종속이 되지 않는다. (not transitive dependent) 따라서 만족한다.

• BCNF : 모든 FD의 결정자(X)가 super-key이다. 따라서 만족한다.

(8) SURGERY Table

• key = { surgery-num }

• 1NF : PK에 의해 각 tuple이 구분되어진다. 또한 각 attribute는 atomic value 이다. 따라서 만족한다.

• 2NF : 모든 non-prime attribute들이 기본 key에 함수 부분 종속이 되지 않는다. (non-partially dependent) 따라서 만족한다.

• 3NF : 모든 non-prime attribute들이 기본 key에 함수 이행 종속이 되지 않는다. (not transitive dependent) 따라서 만족한다.

• BCNF : 모든 FD의 결정자(X)가 super-key이다. 따라서 만족한다.

(9) SURGE-RES

• key = { surgery-num }

• 1NF : PK에 의해 각 tuple이 구분되어진다. 또한 각 attribute는 atomic value 이다. 따라서 만족한다.

• 2NF : 모든 non-prime attribute들이 기본 key에 함수 부분 종속이 되지 않는다. (non-partially dependent) 따라서 만족한다.

• 3NF : 모든 non-prime attribute들이 기본 key에 함수 이행 종속이 되지 않는다. (not transitive dependent) 따라서 만족한다.

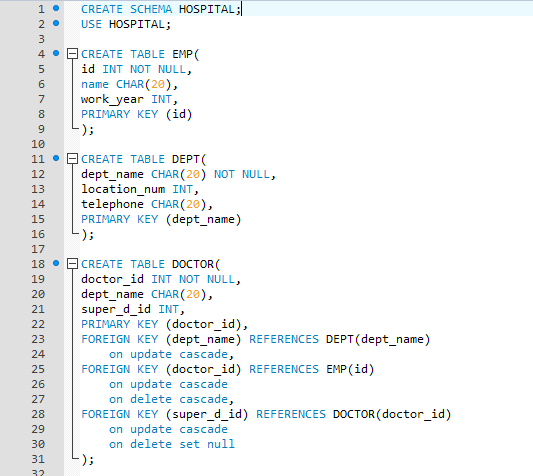
• BCNF : 모든 FD의 결정자(X)가 super-key이다. 따라서 만족한다.

9. SQL DDL 명령문 작성 ; Table 생성 및 tuple 추가

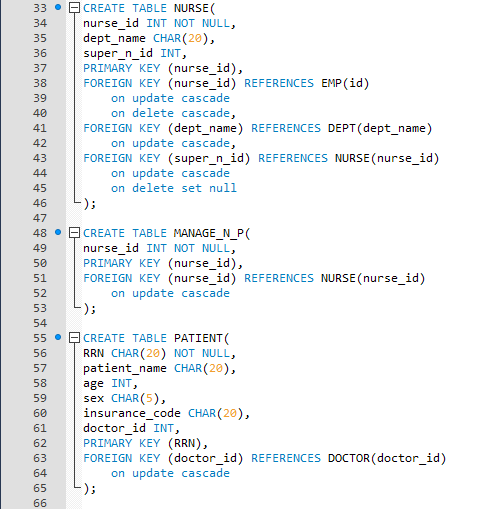
• CREATE SCHEMA 를 이용하여 hospital DB를 생성하였다.

• CREATE TABLE을 이용하여 Base Table을 생성하고, INSERT 및 UPDATE를 이용하여 각 Table에 tuple을 추가하였다.

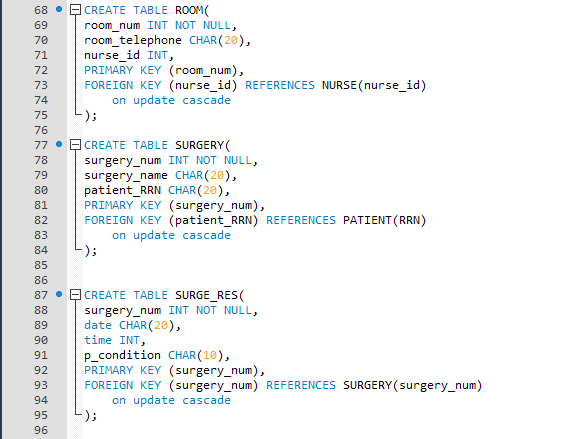
(1) EMP, DEPT, DOCTOR Table 생성



(2) NURSE, MANAGE\_N\_P, PATIENT Table 생성

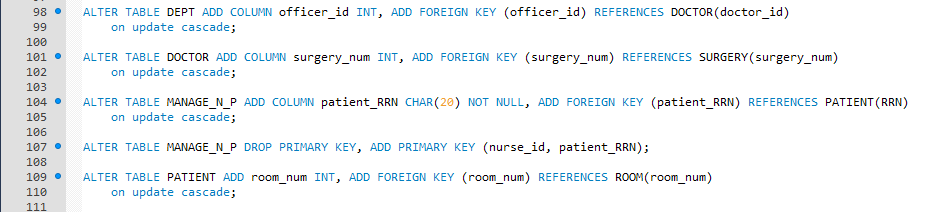


(3) ROOM, SURGERY, SUREG\_RES Table 생성

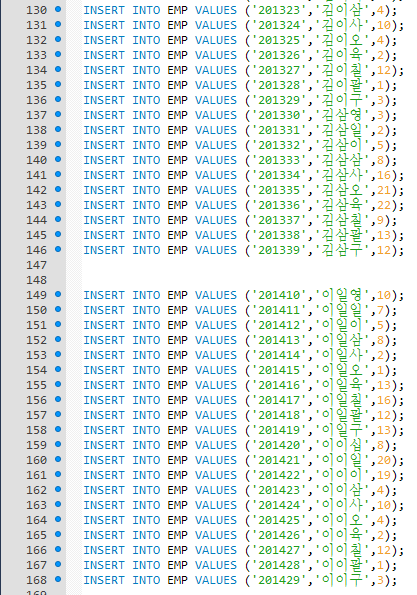


(4) ALTER Table

• 초기 table 생성에서 아직 존재하지 않았던 Foreign Key를 추가하고 각 Table에서 이를 Foreign Key로 설정 하였다.

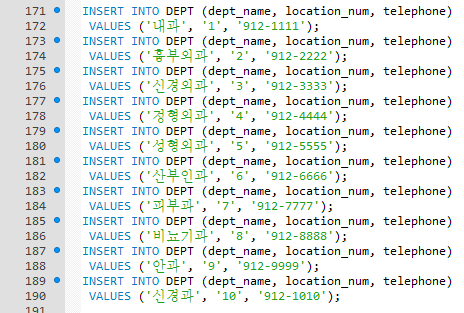


(5) Insert tuples in EMP



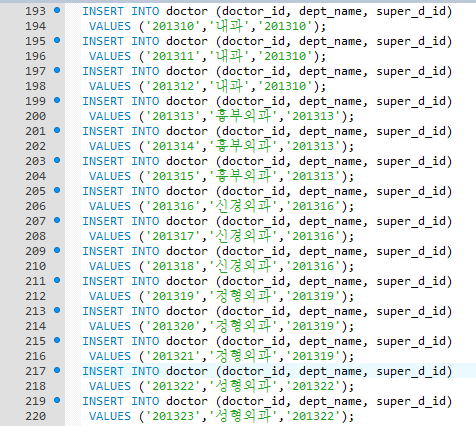
(중략)

(6) Insert tuples in DEPT



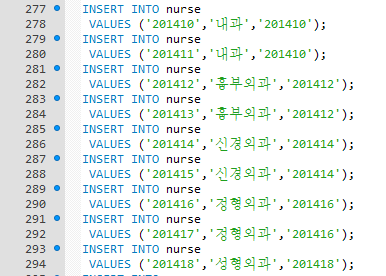
(중략)

(7) Insert tuples in DOCTOR



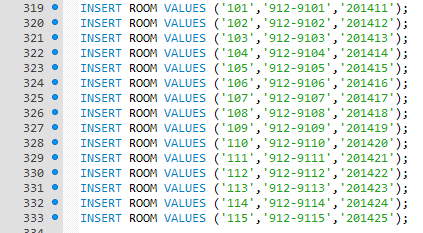
(중략)

(8) Insert tuples in NURSE



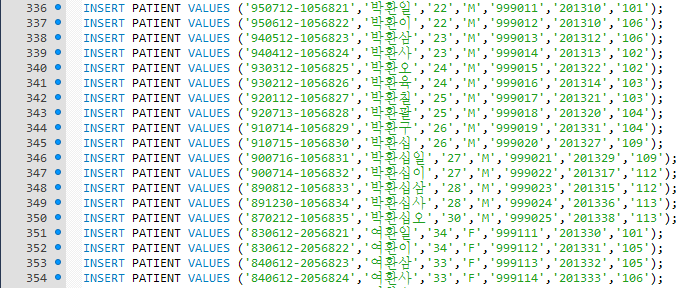
(중략)

(9) Insert tuples in ROOM



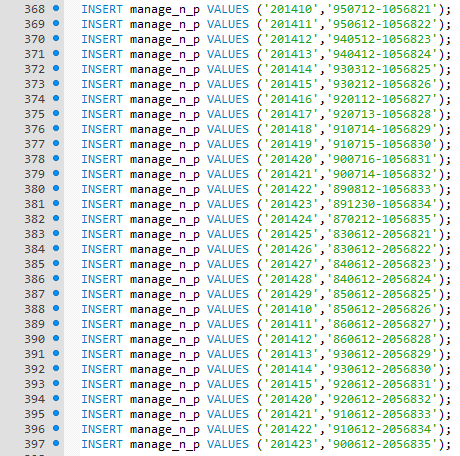
(중략)

(10) Insert tuples in PATIENT



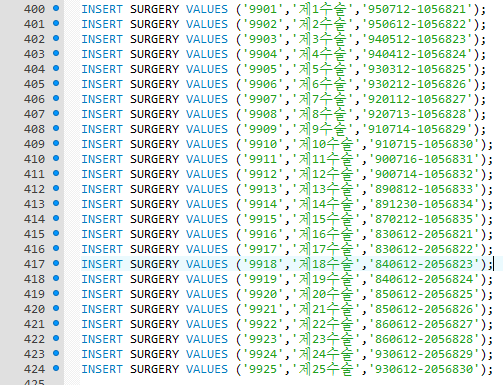
(중략)

(11) Insert tuples in MANAGE\_N\_P



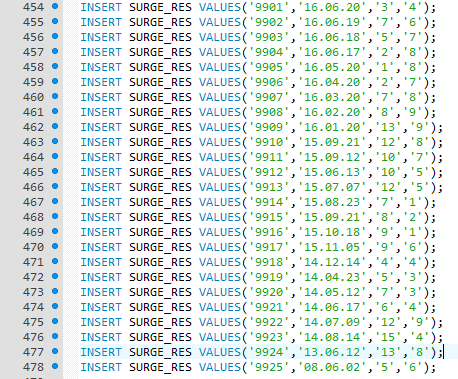
(중략)

(12) Insert tuples in SURGERY



(중략)

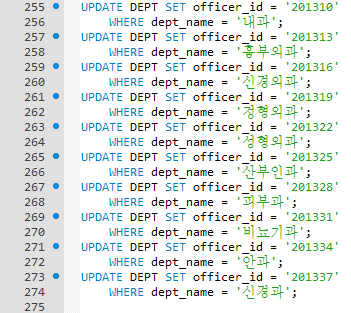
(13) Insert tuples in SURGE\_RES

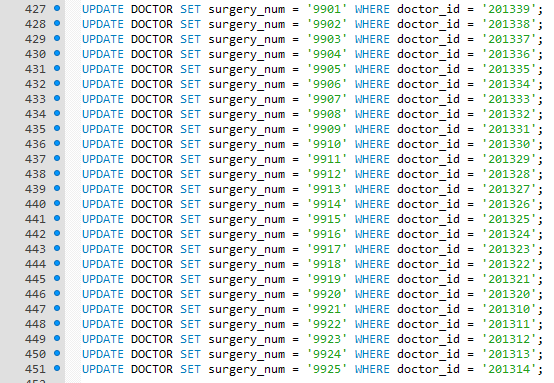


(중략)

(14) Insert more attributes in some Table

• 초기에 존재하지 않은 Foreign key에 의해 입력하지 못했던 몇몇 attributes를 UPDATE를 이용하여 후에 추가하였다.





(중략)

10. Table 결과

(1) EMP



(중략)

(2) DEPT



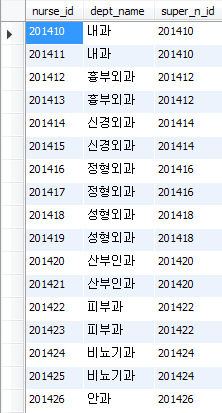
(중략)

(3) DOCTOR



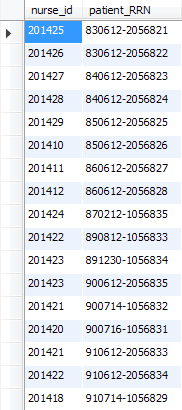
(중략)

(4) NURSE



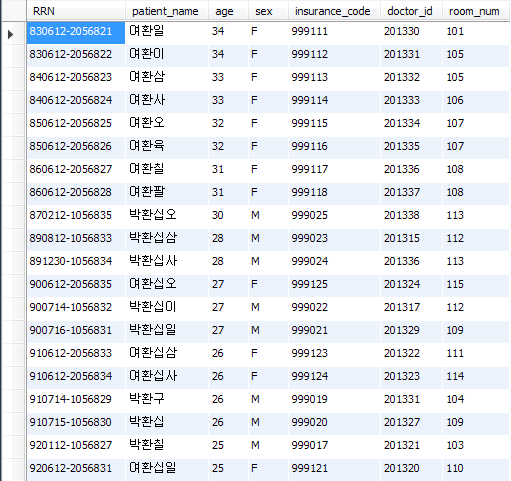
(중략)

(5) MANAGE\_N\_P



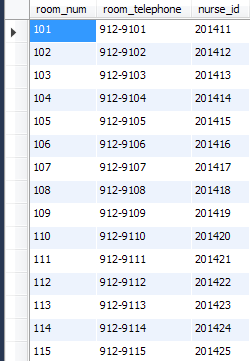
(중략)

(6) PATIENT



(중략)

(7) ROOM



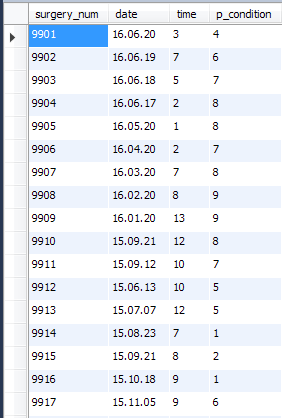
(중략)

(8) SURGERY



(중략)

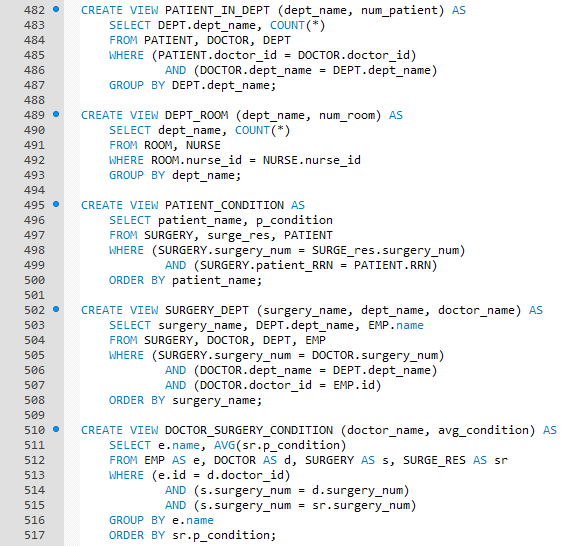
(9) SURGE\_RES



(중략)

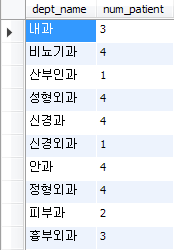
11. CREATE VIEW

• CREATE VIEW와 생성된 Table들을 이용하여 총 5가지의 각각 다른 view를 생성하였다. 아래는 5가지의 view를 생성하는 code이다.



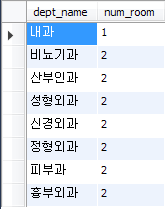
(1) PATIENT\_IN\_DEPT

• 각 부서에서 진료를 받는 환자의 수를 표시한 view이다.



(2) DEPT\_ROOM

• 각 부서에 속한 간호사가 관리하는 병실의 수를 표시하는 view이다.



(3) PATIENT\_CONDITION

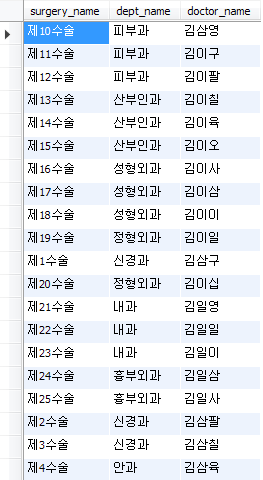
• 수술을 받은 환자들의 수술 후의 condition을 표시한 view이다.



(중략)

(4) SURGERY\_DEPT

• 각 수술명에 대해 수술의 담당 부서 및 수술을 집도한 의사의 이름을 표시한 view이다.



(중략)

(5) DOCTOR\_SURGERY\_CONDITION

• 수술을 집도한 의사들에 대해, 수술 후 환자의 condition을 표시한 view이다.

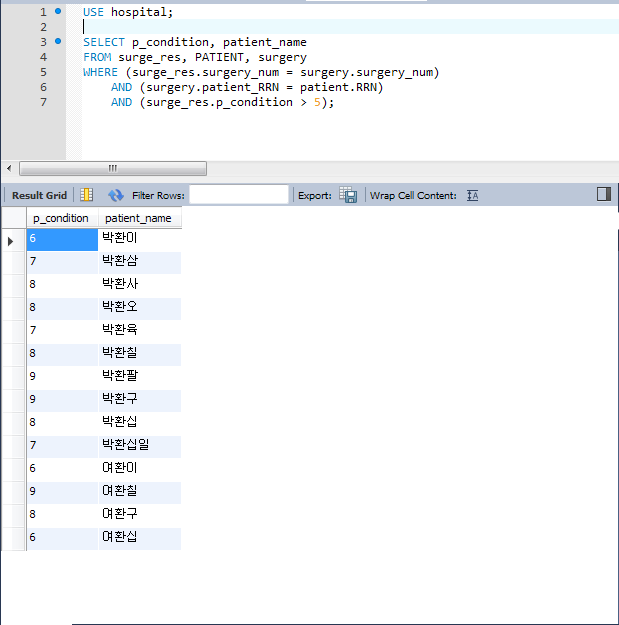


(중략)

12. QUERY문 생성 및 확인

• 총 5가지의 QUERY문을 SELECT FROM WHERE 및 GROUP BY, ORDER BY, VIEW를 이용하여 명령문으로 작성하고, 결과를 확인하였다.

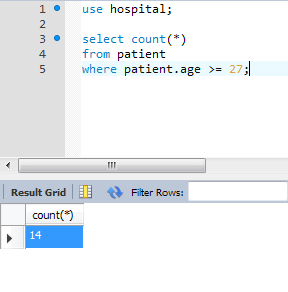
(1) 수술을 받은 환자들 중, 수술 후의 condition이 5 이상인 환자들의 이름을 출력하라.



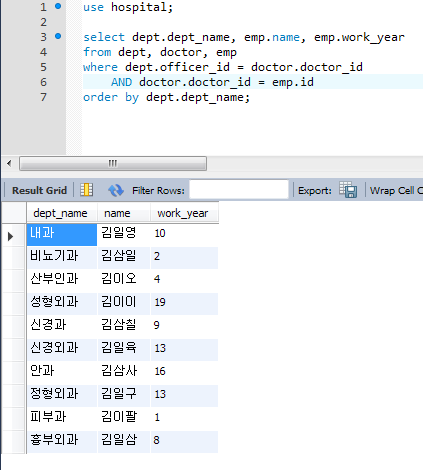
(2) 병원 경력이 10년 이상인 의사들의 이름을 출력하라.



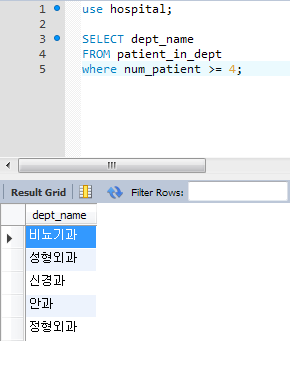
(3) 환자의 나이가 27세 이상인 환자의 수를 출력하라.



(4) 각 부서에 따라 책임자의 이름 및 병원 경력을 출력하라.



(5) 4명 이상의 환자들이 진료를 받은 의사의 해당하는 부서의 이름을 출력하라.



13. 고찰

- 본 프로젝트를 통해 병원 내부의 데이터베이스를 직접 설계하고, MySQL을 이용하여 컴퓨터 상에 구현 및 여러 QUERY문을 실행하였다. 실제 병원에서 데이터 저장에 사용되는 데이터베이스에는 훨씬 많은 table과 attribute가 존재할 것이다. 이 중에서 가장 중요하다고 생각되는 부분만을 추출하여 직접 구현하였다. 수업에서 이론적으로 배운 ER Diagram, Relational Schema, SQL을 직접 실제로 구현하고, 적용 및 구축하는 과정이 매우 흥미로웠고, 한 학기 동안 배운 지식을 사용할 수 있다는 점에서 매우 뜻 깊은 프로젝트였다.

14. 참고 문헌

- DB프로젝트 (2016-병원정보).ppt, 김응모 교수님, 2016년.

- 2016년 1학기 데이터베이스 수업자료, 김응모 교수님, 2016년.

- Online+Banking+DB.pdf, 2016.