Databases Programming Project: **Final** **Report**

이름 (Name): 강태진, 김기윤

학과 (Department): 정보컴퓨터공학부, 정보컴퓨터공학부

학번 (Student ID): 202055508,

1. **프로젝트 개요 (Project topic)**

|  |
| --- |
| 응급의료체계 관리시스템 구축  깃허브 주소  https://github.com/gykim22/PNU-CSE-Emergency-Medical-Facilities-Management-System |

1. **사용자 (역할) (Users / Roles)**

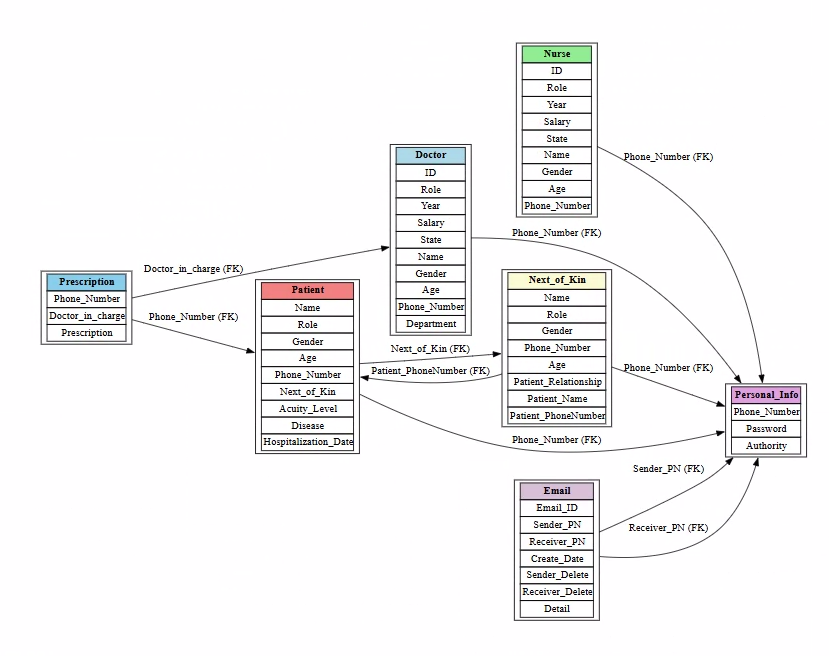
|  |
| --- |
| 병원장 (Doctor / Authority = 0)  - 모든 데이터베이스의 추가, 수정 및 삭제 가능  - 직원의 연봉(Salary)까지 확인이 가능  - 직원(의사, 간호사) 간의 이메일 전송 및 확인이 가능  - 의사이므로, 본인이 담당할 환자를 선택할 수 있고, 담당 환자에게 처방전을 내릴 수 있음.  - 담당 환자의 수는 5명으로 제한  전공 교수 (Doctor / Authority = 1)  - 본인과 같은 전공을 가진 의사의 정보를 수정할 수 있음. 단, 연봉은 수정 불가  - 직원(의사, 간호사) 간의 이메일 전송 및 확인이 가능  - 의사이므로, 본인이 담당할 환자를 선택할 수 있고, 담당 환자에게 처방전을 내릴 수 있음.  - 담당 환자의 수는 5명으로 제한  수간호사 (Nurse / Authority = 2)  - 간호사의 정보를 수정할 수 있음. 단, 연봉은 수정 불가  - 직원(의사, 간호사) 간의 이메일 전송 및 확인이 가능  - 간호사는 모든 환자에게 내려진 처방전을 확인  의사 / 간호사 (Doctor / Nurse / Authority = 3)  - 직원(의사, 간호사) 간의 이메일 전송 및 확인이 가능  - 의사는 본인이 담당할 환자를 선택할 수 있고, 담당 환자에게 처방전을 내릴 수 있음.  - 담당 환자의 수는 5명으로 제한  - 간호사는 모든 환자에게 내려진 처방전을 확인    환자 (Patient / Authority = 4)  - 환자는 본인의 정보와 본인에게 내려진 처방전의 내용 및 주치의를 알 수 있다.    보호자 (Next\_of\_Kin / Authority = 4)  - 보호자는 본인의 환자에 대한 정보를 확인할 수 있다. |

1. **기능 (Functions)**

|  |
| --- |
| **정리된링크**  **[https://abundant-lobe-ec0.notion.site/1566d39f67bc8018a74bfbf0be52a461?pvs=4]**  **로그인**  SQL : db.query('SELECT \* FROM personal\_info WHERE phone\_number = $1', [phone\_number]);  기능 : personal\_info에서 웹 페이지에서 입력한 phone\_number와 동일한 record를 읽어와서,  사용자가 입력한 password와 데이터베이스의 password가 맞는지 매칭.  맞으면 login, 틀리면 login 실패.  **직원 및 환자 추가(join, joinPatient)**  SQL  const result = await db.query('SELECT \* FROM personal\_info WHERE phone\_number = $1', [phone\_number]);  await db.query('INSERT INTO personal\_info (phone\_number, password, authority) VALUES ($1, $2, $3)', [phone\_number, hash, authority\_type]);  if(user\_type === "의사")  await db.query('INSERT INTO Doctor (Role, Year, Salary, State, Name, Gender, Age, Phone\_Number, Department) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9)',  [role, year, salary, "출근", name, gender, age, phone\_number, department]);  else  await db.query('INSERT INTO Nurse (Role, Year, Salary, State, Name, Gender, Age, Phone\_Number) VALUES ($1, $2, $3, $4, $5, $6, $7, $8)',  [role, year, salary, "출근", name, gender, age, phone\_number]);  await db.query('INSERT INTO personal\_info (phone\_number, password, authority) VALUES ($1, $2, $3)', [phone\_number, hash, 4]);  if(user\_type === "환자")  await db.query('INSERT INTO patient (name, role, gender, age, phone\_number, next\_of\_kin, acuity\_level, disease, hospitalization\_date) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9)',  [name, user\_type, gender, age, phone\_number, null, acuity, disease, admission\_date]);  else {  const pName = await db.query('SELECT name FROM patient WHERE phone\_number = $1', [patient\_phone]);  await db.query('INSERT INTO next\_of\_kin (name, role, gender, phone\_number, age, patient\_relationship, patient\_name, patient\_phonenumber) VALUES ($1, $2, $3, $4, $5, $6, $7, $8)',  [name, user\_type, gender, phone\_number, age, relationship, pName.rows[0].name, patient\_phone]);  await db.query('UPDATE patient SET next\_of\_kin = $1 WHERE phone\_number = $2',[phone\_number, patient\_phone]);  }  기능:  result를 사용해서 personal\_info에 이미 전화번호가 있는지 확인  user\_type를 통해서, 의사, 간호사, 환자, 보호자 테이블에 값 insert 수행  **내 프로필(renderProfile)**  SQL  let count = await db.query('SELECT \* FROM prescription WHERE doctor\_in\_charge = $1', [staffPhoneNumber]);  let prescriptionQuery = `SELECT  pt.name AS "PatientName",  p.phone\_number AS "patientPhoneNumber",  pt.age AS "PatientAge",  d.name AS "DoctorName",  pt.gender AS "PatientGender",  pt.disease AS "PatientDisease",  pt.acuity\_level AS "PatientAcuityLevel",  p.prescription AS "PatientPrescription"  FROM  prescription p  JOIN  patient pt ON p.phone\_number = pt.phone\_number  LEFT JOIN  doctor d ON p.doctor\_in\_charge = d.phone\_number  WHERE  p.doctor\_in\_charge = $1 -- [의사직군] 자신이 담당하는 환자의 정보를 가져오기 위해 비교  const patientResult = await db.query(`  SELECT  p.prescription AS prescription,  p.doctor\_in\_charge,  d.name AS doctor\_name,  pt.name AS patient\_name,  pt.age AS patient\_age,  pt.gender AS patient\_gender,  pt.disease AS patient\_disease,  TO\_CHAR(pt.hospitalization\_date, 'YYYY-MM-DD') AS patient\_hospitalization\_date  FROM  prescription p  LEFT JOIN  doctor d ON p.doctor\_in\_charge = d.phone\_number  LEFT JOIN  patient pt ON p.phone\_number = pt.phone\_number  WHERE  p.phone\_number = $1  const kinResult = await db.query(`  SELECT  pt.name AS patient\_name,  pt.age AS patient\_age,  pt.gender AS patient\_gender,  pt.disease AS patient\_disease,  TO\_CHAR(pt.hospitalization\_date, 'YYYY-MM-DD') AS patient\_hospitalization\_date  FROM  patient pt  WHERE  pt.next\_of\_kin = $1  `, [staffPhoneNumber]);  기능 :  count에 의사직군의 처방 테이블로부터 자신이 담당하는 환자의 row를 호출  prescriptionQuery를 통해, 의사가 담당하고있는 환자의 정보를 호출  patientResult를 통해, 렌더링 요청이 환자로부터 들어오는 경우, 자신의 프로필 및 주치의의 이름과 처방전을 확인할 수 있도록 하는 쿼리  kinResult 보호자가 프로필 페이지 렌더링요청을 하는 경우, 보호자가 자신의 환자의 정보에 대해 확인이 가능한 프로필을 호출  **담당 환자 수(renderJoin), (renderJoinPatient)**  SQL  count = await db.query('SELECT \* FROM prescription WHERE doctor\_in\_charge = $1', [staffPhoneNumber]);  기능 :  의사 직군으로 로그인 된 상태이면, 레이아웃에 담당 환자의 수를 출력  **담당 환자 선택(renderGetPatient)**  SQL  const isHere = await db.query('SELECT \* FROM prescription WHERE phone\_number = $1', [phone\_number]);  const isMax = await db.query('SELECT \* FROM prescription WHERE doctor\_in\_charge = $1', [staffPhoneNumber]);  db.query('INSERT INTO prescription (phone\_number, doctor\_in\_charge, prescription) VALUES ($1, $2, $3)', [phone\_number, staffPhoneNumber, "내용 없음."]);  기능:  isHere을 통해 전공의가 이미 배정되어있는지 확인  isMax를 통해, 의사가 담당할 수 있는 환자의 수 제한  조건에 걸리지 않는다면, 환자의 주치의를 설정  **담당 환자 삭제(renderDeletePrescription)**  SQL  await db.query('DELETE FROM prescription WHERE phone\_number = $1', [phone\_number]);  기능 :  처방전 테이블에서 해당 record 삭제  **상태 변경(renderState)**  SQL  await db.query('SELECT \* FROM doctor WHERE phone\_number = $1', [phoneNumber]);  if (doctorResult.rows.length > 0) {  // doctor 테이블에서 업데이트  await db.query('UPDATE doctor SET state = $1 WHERE phone\_number = $2', [state, phoneNumber]);  } else {  await db.query('UPDATE nurse SET state = $1 WHERE phone\_number = $2', [state, phoneNumber]);  }  기능 : 직원(의사, 간호사)인 경우 상태를 변경, 먼저 의사 테이블에서 로그인한 사람이 있는지  확인하고, 있는 경우에는 의사의 데이터베이스에 접근해서 state를 업데이트 하고, 없는 경우  간호사의 데이터베이스에 접근해서 state를 업데이트.  **직원 삭제(renderDeleteStaff)**  SQL  const Result = await db.query('SELECT \* FROM doctor WHERE phone\_number = $1', [phone\_number]);  if (Result.rows.length > 0) { // 만약 의사라면,  await db.query('DELETE FROM doctor WHERE phone\_number = $1', [phone\_number]); // 의사 테이블에서 해당인물 삭제  } else { // 간호사라면,  await db.query('DELETE FROM nurse WHERE phone\_number = $1', [phone\_number]); // 간호사 테이블에서 해당인물 삭제  }  // personal\_info 테이블에서 최종적으로 해당인물 삭제  await db.query('DELETE FROM personal\_info WHERE phone\_number = $1', [phone\_number]);  기능 :  전화번호를 기준으로 의사 테이블에서 값을 찾아서 만약 있다면, 의사 테이블에서 값 삭제, 없다면 간호사 테이블에서 값 삭제.  이후 personal\_info에서 데이터를 삭제  **직원 정보 수정(renderUpdateStaff)**  SQL  let salaryData = await db.query(`SELECT salary FROM doctor WHERE phone\_number = $1`, [phone\_number]);  if(salaryData.rows.length > 0) {  salary = salaryData.rows[0].salary; // 해당 인물이 의사라면, 해당 인물의 연봉 정보를 DB에서 읽어와 기존 그대로 적용  } else {  salaryData = await db.query(`SELECT salary FROM nurse WHERE phone\_number = $1`, [phone\_number]);  salary = salaryData.rows[0].salary; // 해당 인물이 간호사라면, 해당 인물의 연봉 정보를 DB에서 읽어와 기존 그대로 적용  }  if (user\_type === "의사" || user\_type === "전공교수" || user\_type === "병원장") {  await db.query(`UPDATE doctor  SET  role = $1,  year = $2,  salary = $3,  state = $4,  name = $5,  gender = $6,  age = $7,  phone\_number = $8,  department = $9  WHERE phone\_number = $10  `,  [role, year, salary, "출근", name, gender, age, phone\_number, department, phone\_number]  ); // 수정된 데이터를 DB에 적용.  } else { // 간호사 직군이라면,  await db.query(`UPDATE nurse  SET  role = $1,  year = $2,  salary = $3,  state = $4,  name = $5,  gender = $6,  age = $7,  phone\_number = $8  WHERE phone\_number = $9  `,  [role, year, salary, "출근", name, gender, age, phone\_number, phone\_number]  );// 수정된 데이터를 DB에 적용.  }  // 권한 레벨이 달라졌을 경우, 해당 정보를 personal\_info 테이블에 업데이트  await db.query('UPDATE personal\_info SET authority = $1 WHERE phone\_number = $2', [authority\_type, phone\_number]);  기능 :  salaryData를 통해, 병원장이 아닌 직원이 정보를 수정하더라도 salary값이 변하지 않도록 설정  의사 테이블의 정보를 수정하는 경우, 의사테이블에 값을 UPDATE, 간호사 테이블의 정보를 수정하는 경우, 간호사테이블의 값을 UPDATE  personal\_info업데이트를 통해, 권한 레벨이 달라진 경우에는 personal\_info에 업데이트 수행  **환자 삭제(renderJoinPatient)**  SQL  const Result = await db.query('SELECT \* FROM patient WHERE phone\_number = $1', [phone\_number]);  let kinNum;  if (Result.rows.length > 0) { // 환자라면,  kinNum = await db.query('SELECT next\_of\_kin FROM patient WHERE phone\_number = $1', [phone\_number]);  kinNum = kinNum.rows[0].next\_of\_kin; // 환자에게 연결된 보호자가 있다면, 해당 보호자의 전화번호 저장  await db.query('DELETE FROM patient WHERE phone\_number = $1', [phone\_number]); // 환자 테이블에서 해당인물 삭제  } else { // 보호자라면,  await db.query('UPDATE patient SET next\_of\_kin = $1 WHERE next\_of\_kin = $2', [null, phone\_number]); // 환자 테이블에서 보호자 전화번호 삭제  await db.query('DELETE FROM next\_of\_kin WHERE phone\_number = $1', [phone\_number]); // 보호자 테이블에서 해당인물 삭제  }  if(kinNum) // 환자 삭제 시, 해당 환자에게 등록된 보호자가 존재한다면,  await db.query('DELETE FROM personal\_info WHERE phone\_number = $1', [kinNum]); // personal\_info에서 보호자 정보 삭제  await db.query('DELETE FROM personal\_info WHERE phone\_number = $1', [phone\_number]); // personal\_info에서 해당 인물 삭제.  기능 :  전화번호를 기준으로 환자 테이블에서 값을 찾아서 값이 있다면, 보호자의 전화번호를 저장하고 환자 테이블에서 값 삭제  환자 테이블에서 값이 없다면, 보호자 테이블로 이동해서 환자 테이블의 보호자 전화번호를 삭제하고, 보호자 테이블에서 해당인물 삭제  만약 환자 삭제를 한 경우에, 해당 환자에 등록된 보호자가 존재하는 경우, personal\_info에서 보호자를 삭제  모든게 마무리 된 이후, personal\_info의 환자의 값을 삭제  **환자, 보호자 정보 수정(renderUpdatePatient)**  SQL  const Result = await db.query('SELECT \* FROM patient WHERE phone\_number = $1', [phone\_number]);  await db.query(`UPDATE patient  SET  name= $1,  gender = $2,  age = $3,  acuity\_level = $4,  disease = $5,  hospitalization\_date = $6  WHERE phone\_number = $7  `,  [name, gender, age, acuity, disease, admission\_date, phone\_number]  ); // 수정된 정보를 환자 테이블의 해당 인물에게 업데이트  const p\_name = await db.query('SELECT name FROM patient WHERE phone\_number = $1', [patient\_phone]);  const tmp = await db.query('SELECT name FROM patient WHERE next\_of\_kin = $1', [phone\_number]);  if (tmp.rows.length === 1) { // 이전 환자에게 등록되었던 보호자의 전화번호를 삭제함.  await db.query(`UPDATE patient  SET next\_of\_kin = $1  WHERE next\_of\_kin = $2`, [null, phone\_number]);  }  // 보호자 정보 갱신  await db.query(`UPDATE next\_of\_kin  SET  name= $1,  gender = $2,  age = $3,  patient\_relationship = $4,  patient\_name = $5,  patient\_phonenumber = $6  WHERE phone\_number = $7  `,  [name, gender, age, relationship, p\_name.rows[0].name, patient\_phone, phone\_number]  );  // 새로 바뀐 환자 전화번호에 따라, 해당 환자에게 보호자 전화번호를 새로 삽입.  await db.query(`UPDATE patient  SET  next\_of\_kin = $1 WHERE phone\_number = $2  `,  [phone\_number, patient\_phone]  );  기능 :  환자의 정보를 수정하는 경우, 해당 테이블에 값을 업데이트  환자의 전화번호를 수정한 경우 이전 환자 테이블에 등록되어있던 전화번호를 삭제하고, 새로운 보호자 전화번호를 삽입.  **직원 리스트(renderDeletePatient)**  SQL  // 의사 리스트 쿼리. salary를 포함한 모든 데이터를 가져옴.  doctorQuery = `SELECT  id AS "ID",  role AS "Role",  year AS "Year",  salary AS "Salary",  state AS "State",  name AS "Name",  gender AS "Gender",  age AS "Age",  phone\_number AS "PhoneNumber",  department AS "Department"  FROM Doctor  ORDER BY ${doctorSort} ${doctorOrder}`;  // 간호사 리스트 쿼리. salary를 포함한 모든 데이터를 가져옴.  nurseQuery = `SELECT  id AS "ID",  role AS "Role",  year AS "Year",  salary AS "Salary",  state AS "State",  name AS "Name",  gender AS "Gender",  age AS "Age",  phone\_number AS "PhoneNumber"  FROM Nurse  ORDER BY ${nurseSort} ${nurseOrder}`;  doctorQuery = `SELECT  id AS "ID",  role AS "Role",  year AS "Year",  state AS "State",  name AS "Name",  gender AS "Gender",  age AS "Age",  phone\_number AS "PhoneNumber",  department AS "Department"  FROM Doctor  ORDER BY ${doctorSort} ${doctorOrder}`;  // 간호사 리스트 쿼리. salary를 제외한 모든 데이터를 가져옴.  nurseQuery = `SELECT  id AS "ID",  role AS "Role",  year AS "Year",  state AS "State",  name AS "Name",  gender AS "Gender",  age AS "Age",  phone\_number AS "PhoneNumber"  FROM Nurse  ORDER BY ${nurseSort} ${nurseOrder}`;  }  기능 :  병원장이 직원 리스트를 호출하는 경우, salary까지 호출, 이외의 직원이 호출하는 경우에는 salary를 제외하고 호출을 수행  **환자 리스트(renderListPatient)**  SQL  patientQuery = `SELECT  name AS "Name",  gender AS "Gender",  age AS "Age",  phone\_number AS "PhoneNumber",  next\_of\_kin AS "KinNumber",  acuity\_level AS "Acuity",  disease AS "Disease",  TO\_CHAR(hospitalization\_date, 'YYYY-MM-DD') AS "HospitalizationDate"  FROM patient  ORDER BY ${patientSort} ${patientOrder}`;  // 보호자 리스트 데이터를 요청하기 위한 쿼리  kinQuery = `SELECT  name AS "Name",  gender AS "Gender",  age AS "Age",  phone\_number AS "PhoneNumber",  patient\_relationship AS "relation",  patient\_name AS "PatientName",  patient\_phonenumber AS "PatientPhoneNumber"  FROM next\_of\_kin  ORDER BY ${kinSort} ${kinOrder}`;  기능 :  환자와 보호자 리스트를 호출  **환자 처방전 열람(renderPrescriptionPatient)**  SQL  patientQuery = `SELECT  name AS "Name",  gender AS "Gender",  age AS "Age",  phone\_number AS "PhoneNumber",  next\_of\_kin AS "KinNumber",  acuity\_level AS "Acuity",  disease AS "Disease",  TO\_CHAR(hospitalization\_date, 'YYYY-MM-DD') AS "HospitalizationDate"  FROM patient p  WHERE NOT EXISTS ( --담당의가 배정된 환자는 제외함.  SELECT \*  FROM prescription  WHERE prescription.phone\_number = p.phone\_number  )  ORDER BY ${patientSort} ${patientOrder}`;  if(req.user.authority === "간호사" || req.user.authority === "수간호사") {  prescriptionQuery = `SELECT  pt.name AS "PatientName",  pt.age AS "PatientAge",  pt.gender AS "PatientGender",  pt.disease AS "PatientDisease",  pt.acuity\_level AS "PatientAcuityLevel",  d.name AS "DoctorName",  p.phone\_number As "PhoneNumber",  p.prescription AS "Prescription"  FROM  prescription p  JOIN  patient pt ON p.phone\_number = pt.phone\_number  LEFT JOIN  doctor d ON p.doctor\_in\_charge = d.phone\_number  ORDER BY ${mySort} ${myOrder}`;  } else {  // 만약 해당 페이지 랜더링을 요청한 인물이 의사 직군일 경우  // 본인이 담당하는 환자 데이터를 prescription 테이블로부터 불러오는 쿼리  prescriptionQuery = `SELECT  pt.name AS "PatientName",  pt.age AS "PatientAge",  pt.gender AS "PatientGender",  pt.disease AS "PatientDisease",  pt.acuity\_level AS "PatientAcuityLevel",  d.name AS "DoctorName",  p.phone\_number As "PhoneNumber",  p.prescription AS "Prescription"  FROM  prescription p  JOIN  patient pt ON p.phone\_number = pt.phone\_number  LEFT JOIN  doctor d ON p.doctor\_in\_charge = d.phone\_number  WHERE  p.doctor\_in\_charge = $1  ORDER BY ${mySort} ${myOrder}`;  기능 :  의사 직군이 환자 리스트를 호출한 경우 patientQuery를 사용해서 담당의가 미 배정된 환자 데이터를 호출  처방전 호출:  간호사 직군이 환자 리스트를 호출한 경우 prescriptionQuery를 통해 담당의가 배정된 환자 처방전을 모두 호출  의사직군의 경우에는 prescriptionQuery를 통해 본인에게 배정된 환자의 처방전 데이터를 호출  **환자 처방전 작성(renderWritePrescription)**  SQL  await db.query('UPDATE prescription SET prescription = $1 WHERE phone\_number = $2', [prescriptionField, phone\_number]);  기능 :  담당의가 입력한 처방전을 데이터베이스에 업데이트를 수행.  **이메일 확인**  SQL  const sentEmailsResult = await db.query(`  SELECT \* FROM Email WHERE Sender\_PN = $1 AND sender\_delete = FALSE ORDER BY Create\_Date DESC  `, [phoneNumber]);  const receivedEmailsResult = await db.query(`  SELECT \* FROM Email WHERE Receiver\_PN = $1 AND receiver\_delete = FALSE ORDER BY Create\_Date DESC  `, [phoneNumber]);  기능 :  sentEmailResult에 본인이 보낸 Email을 모두 가져옴, receivedEmailResult를 통해 본인이  받은 Email을 모두 가져옴, 본인 확인은 phone\_number로 체크.  이때, sender\_delete나 receiver\_delete값이 false인 record만 호출  **이메일 보내기**  SQL  const emailResult = await db.query(  `SELECT \* FROM Email WHERE Email\_ID = $1`,  [email\_id]  );  if (type === 'sender') {  await db.query(  `UPDATE Email SET Sender\_Delete = TRUE WHERE Email\_ID = $1`,  [email\_id]  );  } else if (type === 'receiver') {  await db.query(  `UPDATE Email SET Receiver\_Delete = TRUE WHERE Email\_ID = $1`,  [email\_id]  );  const updatedEmailResult = await db.query(  `SELECT Sender\_Delete, Receiver\_Delete FROM Email WHERE Email\_ID = $1`,  [email\_id]  );  await db.query(  `DELETE FROM Email WHERE Email\_ID = $1`,  [email\_id]  );  기능 :  emailResult를 통해 삭제를 선택한 이메일이 데이터베이스에 존재하는지 확인, 이후 삭제를  선택한 사람이 sender인지 receiver인지에 따라 데이터베이스의 delete값을 true로 설정.  delete값을 true로 설정한 경우, 데이터베이스에서 sender\_delete와 receiver\_delete값 모두 true인  record를 확인하고, 이러한 record가 있는 경우 DELETE를 통해 데이터베이스에서 삭제를 진행 |

1. **데이터베이스 스키마 및 다이어그램 (Database schema / Schema diagram)**

|  |
| --- |
| -- personal\_info  -- 회원가입과 동일한 역할이지만, 병원장의 permisstion이 있어야 수행되기 때문에, transaction은 필요 없음.  -- 전화번호, 암호, 역할의 레벨  +--------------+----------+------------+  | Phone\_Number | password | Authority |  +--------------+----------+------------+  Candidate Key : Phone\_Number  Authorization : 의사 중에서 병원장만 열람, 추가, 삭제, 편집 가능  CREATE TABLE IF NOT EXISTS personal\_info (  Phone\_Number VARCHAR(15) PRIMARY KEY, -- 전화번호, Candidate Key  password VARCHAR(255) NOT NULL, -- 암호  Authority SMALLINT NOT NULL CHECK (Authority IN (0, 1, 2, 3, 4)) -- 역할의 레벨 (0~4)  );  Authority  - 레벨 0, 1, 2, 3, 4  - 0: Chief of Medical Clinic : 병원장  - 1: Medical Professor : 전공 교수  - 2: Head Nurse : 수간호사  - 3: Hospital Staff(Doctor + Nurse)  - 4: Outsider(Patient + Next\_of\_kin)    -- Doctor 테이블  -- id, 직위, 연차, 연봉, 상태, 이름, 성별, 나이, 전화번호, 전공  +----+------+------+- ------+-------+------+--------+-----+--------------+------------+  | ID | Role | Year | Salary | State | Name | Gender | Age | Phone\_Number | Department |  +----+------+------+- ------+-------+------+--------+-----+--------------+------------+  Candidate Key : Phone\_Number  Foreign key : Phone\_Number -> personal\_info(Phone\_Number)  Authorization : 병원장은 열람, 추가, 삭제, 편집 가능, 전공교수는 민감정보를 제외한 정보를 열람 및 편집 가능  CREATE TABLE IF NOT EXISTS Doctor (  ID SERIAL UNIQUE, -- id, 자동으로 수가 늘어나도록 SERIAL사용  Role VARCHAR(50), -- 직위  Year INT, -- 연차  Salary NUMERIC(10, 2), -- 연봉  State VARCHAR(20), -- 상태  Name VARCHAR(20), -- 이름  Gender CHAR(1), -- 성별 M, F  Age INT, -- 나이  Phone\_Number VARCHAR(15) PRIMARY KEY, -- 전화번호  Department VARCHAR(30), -- 전공  FOREIGN KEY (Phone\_Number) -- Foreign Key 제약 조건  REFERENCES personal\_info (Phone\_Number)  ON DELETE CASCADE -- personal\_info에서 삭제 시 Doctor도 삭제  ON UPDATE CASCADE -- personal\_info에서 변경 시 Doctor도 변경  );  -- Nurse 테이블  -- id, 직위, 연차, 연봉, 상태, 이름, 성별, 나이, 전화번호  +----+------+------+- ------+-------+------+--------+-----+--------------+  | ID | Role | Year | Salary | State | Name | Gender | Age | Phone\_Number |  +----+------+------+- ------+-------+------+--------+-----+--------------+  Candidate Key : Phone\_Number  Foreign key : Phone\_Number -> personal\_info(Phone\_Number)  Authorization : 병원장은 열람, 추가, 삭제, 편집 가능, 수 간호사는 민감정보를 제외한 정보를 열람 및 편집 가능  CREATE TABLE IF NOT EXISTS Nurse (  ID SERIAL UNIQUE, -- id  Role VARCHAR(50), -- 직위  Year INT, -- 연차  Salary NUMERIC(10, 2), -- 연봉  State VARCHAR(20), -- 상태  Name VARCHAR(20), -- 이름  Gender CHAR(1), -- 성별 M, F  Age INT, -- 나이  Phone\_Number VARCHAR(15) PRIMARY KEY, -- 전화번호 (Candidate Key)  FOREIGN KEY (Phone\_Number) -- Foreign Key 제약 조건  REFERENCES personal\_info (Phone\_Number)  ON DELETE CASCADE -- personal\_info에서 삭제 시 Nurse도 삭제  ON UPDATE CASCADE -- personal\_info에서 변경 시 Nurse도 변경  );  -- Patient 테이블  -- 이름, 성별, 나이, 전화번호, 보호자\_전화번호, 위험도, 질병, 입원 날짜, 치료 방법  +--------------+------+--------+-----+--------------+-------------+--------------+-----------+----------------------------+  | Name | Role | Gender | Age | Phone\_Number | Next\_of\_Kin | Acuity\_Level | Disease | Hospitalization\_Date |  +--------------+------+--------+-----+--------------+-------------+--------------+-----------+----------------------------+  Candidate Key : Phone\_Number  Foreign key : Phone\_Number -> personal\_info(Phone\_Number), Next\_of\_Kin\_PN -> Next\_of\_Kin(Phone\_Number)  Authorization : 의사와 간호사가 열람, 추가, 삭제, 편집 가능  CREATE TABLE Patient (  Name VARCHAR(20), -- 이름  Role VARCHAR(50), -- 직위  Gender CHAR(1), -- 성별 (M, F)  Age INT, -- 나이  Phone\_Number VARCHAR(15) PRIMARY KEY, -- 전화번호 (Candidate Key)  Next\_of\_Kin VARCHAR(15), -- 보호자의 전화번호  Acuity\_level INT, -- 위험도  Disease VARCHAR(50), -- 질병  Hospitalization\_Date DATE, -- 입원 날짜 (DATE 타입)  FOREIGN KEY (Phone\_Number) -- 환자의 전화번호가 personal\_info를 참조  REFERENCES personal\_info (Phone\_Number)  ON DELETE CASCADE  ON UPDATE CASCADE,  FOREIGN KEY (Next\_of\_Kin) -- 보호자의 전화번호가 personal\_info를 참조  REFERENCES personal\_info (Phone\_Number)  ON DELETE SET NULL -- 보호자 삭제 시 NULL로 설정  ON UPDATE CASCADE  );  -- next\_of\_kin  -- 이름, 성별, 나이, 전화번호, 환자와의 관계, 환자의 이름, 환자의 전화번호  +------+------+--------+--------------+-----+----------------------+---------+---------------------+  | name | Role | Gender | Phone\_Number | Age | Patient\_Relationship | Patient | Patient\_PhoneNumber |  +------+------+--------+--------------+-----+----------------------+---------+---------------------+  Candidate Key : Phone\_Number  Foreign key : Phone\_Number -> personal\_info(Phone\_Number), Patient\_PhoneNumber -> Patient(Phone\_Number)  Authorization : 의사와 간호사가 열람, 추가, 삭제, 편집 가능  CREATE TABLE IF NOT EXISTS next\_of\_kin (  name VARCHAR(20), -- 보호자 이름  Role VARCHAR(50), -- 직위  Gender CHAR(1), -- 보호자 성별 ('M', 'F')  Phone\_Number VARCHAR(15) PRIMARY KEY, -- 보호자 전화번호 (Candidate Key)  Age INT, -- 보호자 나이  Patient\_Relationship VARCHAR(50), -- 환자와의 관계  Patient\_Name VARCHAR(20), -- 환자 이름  Patient\_PhoneNumber VARCHAR(15), -- 환자 전화번호  FOREIGN KEY (Phone\_Number) -- 보호자의 전화번호가 personal\_info를 참조  REFERENCES personal\_info (Phone\_Number)  ON DELETE CASCADE  ON UPDATE CASCADE,  FOREIGN KEY (Patient\_PhoneNumber) -- 환자의 전화번호가 Patient를 참조  REFERENCES Patient (Phone\_Number)  ON DELETE CASCADE  ON UPDATE CASCADE  );  -- Prescription  -- 환자의 전화번호, 주치의 전화번호, 처방전  -- 주치의가 볼 수 있는 환자의 수 제한 -> 병원에서 감당할 수 있는 환자의 수 결정됨.  +--------------+------------------+--------------+  | Phone\_Number | Doctor\_in\_charge | prescription |  +--------------+------------------+--------------+  Candidate Key : Phone\_Number  Foreign key : Phone\_Number -> personal\_info(Phone\_Number), Doctor\_in\_charge -> Doctor(phone\_number)  CREATE TABLE IF NOT EXISTS Prescription (  Phone\_Number VARCHAR(15), -- 환자의 전화번호  Doctor\_in\_charge VARCHAR(15), -- 주치의 전화번호  Prescription VARCHAR(300), -- 처방전 내용  FOREIGN KEY (Phone\_Number) -- 환자의 전화번호가 Patient를 참조  REFERENCES Patient (Phone\_Number)  ON DELETE CASCADE -- 환자 삭제 시 처방전도 삭제  ON UPDATE CASCADE,  FOREIGN KEY (Doctor\_in\_charge) -- 주치의가 Doctor를 참조  REFERENCES Doctor (phone\_number)  ON DELETE SET NULL -- 주치의 삭제 시 NULL로 설정  ON UPDATE CASCADE  );    -- Email  -- transaction 필요  -- 이메일 ID, 보낸 이 전화번호, 받는 이 전화번호, 작성일자, 열람 일자(NULL), 보낸 이 삭제 여부(default 0), 받는 이 삭제 여부(0), 이메일 내용  +-----------+-----------+-------------+-------------+---------------+-----------------+--------+  | Email\_id | Sender\_PN | Receiver\_PN | create\_date | Sender\_delete | Receiver\_delete | Detail |  +-----------+-----------+-------------+-------------+---------------+-----------------+--------+  CREATE TABLE IF NOT EXISTS Email (  Email\_ID SERIAL PRIMARY KEY, -- 이메일 고유 식별자  Sender\_PN VARCHAR(15) NOT NULL, -- 보낸 이 ID  Receiver\_PN VARCHAR(15) NOT NULL, -- 받는 이 ID  Create\_Date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, -- 작성 일자 (현재 시간 기본값)  Sender\_Delete BOOLEAN DEFAULT FALSE, -- 보낸 이 삭제 여부 (기본값: 0)  Receiver\_Delete BOOLEAN DEFAULT FALSE, -- 받는 이 삭제 여부 (기본값: 0)  Detail VARCHAR(300) NOT null, -- 내용, NULL 불가능  FOREIGN KEY (Sender\_PN) -- 보낸 이 ID 외래 키  REFERENCES personal\_info (Phone\_Number)  ON DELETE CASCADE  ON UPDATE CASCADE,  FOREIGN KEY (Receiver\_PN) -- 받는 이 ID 외래 키  REFERENCES personal\_info (Phone\_Number)  ON DELETE CASCADE  ON UPDATE CASCADE  ); |



1. **(팀 프로젝트인 경우만 해당) 팀원의 역할 배분**

|  |
| --- |
| 김기윤, 강태진 : 코드개발 및 readME.md작성, 내용정리 |