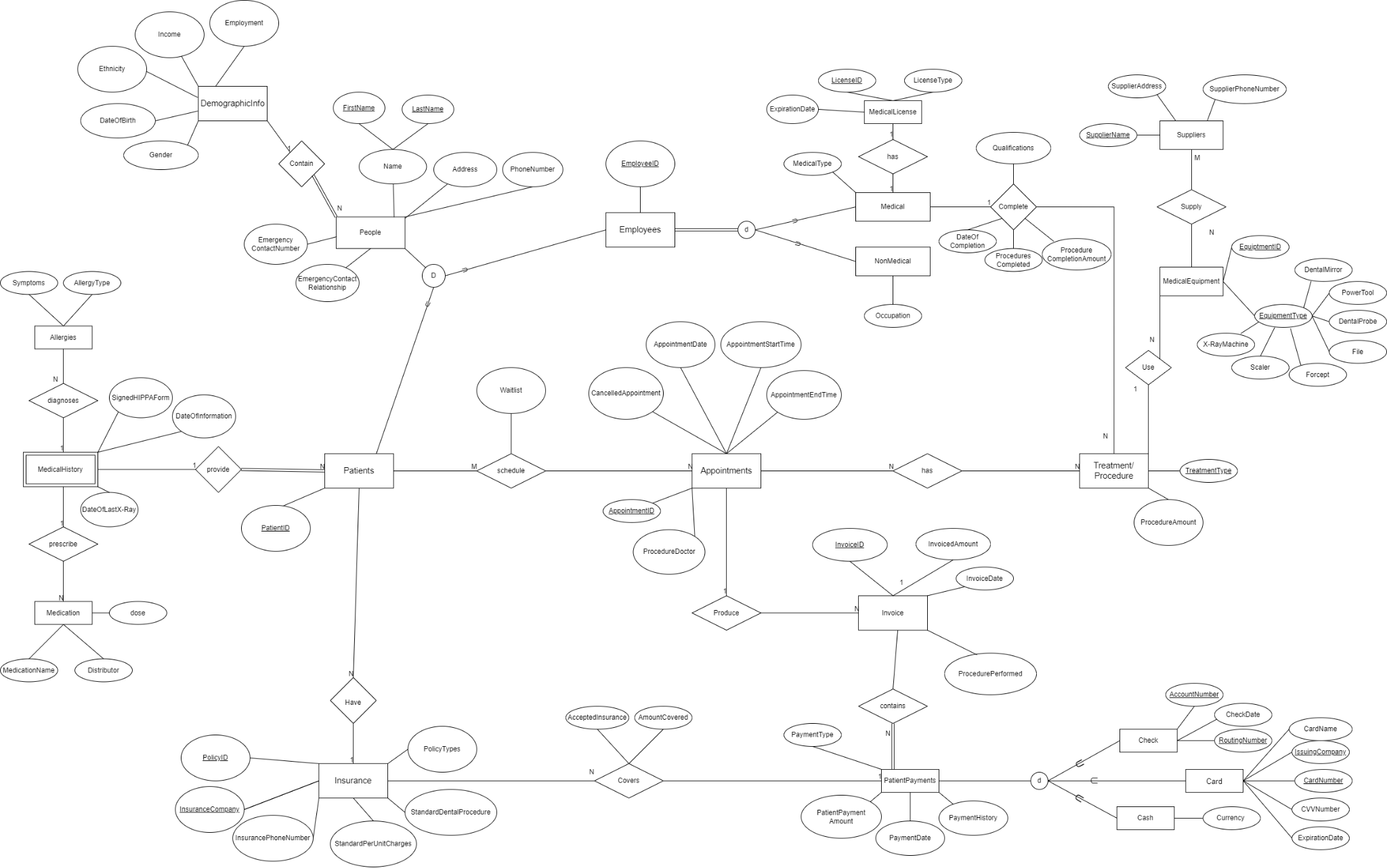
CSE 3241 Project Part 2

Team 7: Jason Fong, Vivian Comer,

Macray Curran, Cameron Erdman

**Updated ERD**



**Relational Schema**

(Red text represents where the entity is from for clarity. It isn’t the name of the entity)

People(FirstName (PK), LastName (PK), Address, PhoneNumber,

EmergencyContanctNumber, EmergencyContactRelationship)

DemographicInfo(FirstName (FK - People), LastName (FK - People), Employment,

Income, Ethnicity, DateOfBirth, Gender)

Patients(PatientID (PK), FirstName (FK - People), LastName (FK - People))

Employees(EmployeeID (PK), FirstName (FK - People), LastName (FK - People))

MedicalHistory(PatientID (FK - Patients), SignedHIPPAForm, DateOfInformation,

DateOfLastX-Ray)

Allergies(PatientID (FK - Patients), AllergyType, Symptoms)

Medications(PatientID (FK - Patients), Dose, MedicationName, Distributor)

Insurance(PatientID (FK - Patients), PolicyID (PK), InsuranceCompany (PK),

InsurancePhoneNumber,StandardPerUnitCharges, StanardDentalProcedure,

PolicyTypes)

Appointments(AppointmentID (PK), PatientID (FK - Patient), CancelledAppointment,

AppointmentDate, AppointmentStartTime, AppointmentEndTime, ProcedureDoctor)

Schedule(AppointmentID (FK - Appointment), Waitlist)

Invoice(InvoiceID (PK), InvoicedAmount, InvoiceDate, ProcedurePerformed)

PatientPayments(InvoiceID (FK - Invoice) InsuranceName (FK - Insurance)

PatientPaymentAmount, PaymentDate, PaymentHistory,PaymentType)

(Insurance <covers> PatientPayments)CoveredInsurance(PolicyID (FK - Insurance),

InsuranceName (FK - Insurance), AcceptedInsurance, AmountCovered)

Check(AccountNumber (PK), RoutingNumber (PK), CheckDate)

Card(CardNumber (PK), IssuingCompany (PK), CardName, CVVNumber,

ExpirationDate)

Cash(Currency)

Treatment/Procedure(PatientID (FK - Patient), EquipmentID (FK - MedicalEquipment), PaymentHistory, TreatmentType, ProcedureAmount)

Medical(EmployeeID (FK - Employees), LicenseID (FK - MedicalLicense), MedicalType)

(Medical <Complete> Procedure)Completed\_Procedures(EmployeeID (FK - Employee),

Qualifications, DateOfCompletion, ProceduresCompleted,

ProcedureCompletionAmount)

NonMedical(EmployeeID (FK - Employees), Occupation)

MedicalLicense(LicenseID (PK), LicenseType, ExpirationDate)

MedicalEquipment(EquipmentID (PK), SupplierName (FK - Suppliers), X-RayMachine, Scaler, Frocept, File, DentalProbe, PowerTool, DentalMirror)

Suppliers(SupplierName (PK), SupplierAddress, SupplierPhoneNumber)

To start off making this relationship schema we first added all of the entities with their attributes and primary keys. Afterward, we add foreign keys to keyless entities and to entities that we wanted to link. An example of this would be MedicalHistory having PatientID as a foreign key so we can link patients to their medical history. Then when we thought of the multiple queries we would need, we added even more foreign keys and extra entities from relations with n to m cardinality.

**Relational Algebra**

**A:**

**Create a list of patients and the medications they currently take)**

PATIENT ⋈PatientID = PatientID MEDICATION

**B:**

**Display patient information for patients who currently have Delta Dental insurance policy.)**

PATIENT\_DELTA←(PATIENT ⋈PatientID =PatientID) (INSURANCE)

RESULT ← σ(InsuranceCompany = Delta Dental Insurance)(PATIENT\_DELTA)

**C:**

**Generate a list of procedures and dates of service performed by doctor Smilow)**

MEDICAL\_ID ←EMPLOYEES ⋈EmployeeID = EmployeeID (MEDICAL)

SMILLOW\_ID ←σ(LastName = Smillow) (MEDICAL\_ID)

APPOINTMENT\_SMILLOW←APPOINTMENTS⋈ProcedureDoctor = LastName (SMILLOW\_ID)

RESULT ← π(TreatmentType, AppointmentDate)(APPOINTMENT\_SMILLOW))

**D:**

**Print out a list of past-due invoices with patient contact information. Past due is defined as over 30 days old with a balance over $10)**

DUE\_LIST ← σ(RemainingBalance > 10 AND Unpaid > 30)(INVOICE)

PATIENT\_LIST ← PATIENT⋈(PatientID = PatientID)(DUE\_LIST)

RESULT ← π(PatientID, InvoiceID, InvoiceAmount)(PATIENT\_LIST)

**E:**

**Find the patients who brought the most revenue in the past year.)**

RESULT ← PatientID F(SUM PatientPaymentAmount) (Payment)

**F:**

**Create a list of doctors who performed less than 5 procedures this year.)**

MEDICAL\_ID ←EMPLOYEES ⋈EmployeeID = EmployeeID (MEDICAL)

APPOINTMENT\_MEDICALS←APPOINTMENTS ⟗ ProcedureDoctor = LastName (MEDICAL\_ID)

RESULT ← σ(AppointmentID\_Count < 5)(APPOINTMENT\_MEDICALS)

**G:**

**Find the highest paying procedures, procedure price, and the total number of those**

**procedures performed.)**

TREATMENT\_NUM ←Invoice ⟗ProcedurePerformed = TreatmentType (Treatment/Procedure)

MAX ←TreatmentType FMAX ProcedureAmount , COUNT TreatmentType (TREATMENT\_NUM)

π (Treatment\_Name, Procedure\_price, Total\_num) (MAX)

**H:**

**Create a list of all payment types accepted, the number of times each was used, and the total amount charged to that type of payment.)**

LIST ←TypeFCOUNT PaymentDate, SUM PatientPaymentAmount(PatientPayments)

π (PaymentType, Num\_Usage, Total\_Amount) (LIST)

**I:**

**List ids and names of insurance plans ever used by patients and how**

**many patients have that plan)**

INSURANCE\_NUM ←Insurance ⟗AmountCovered = PatientPaymentAmount (PatientPayments)

SUM ←InsuranceCompany FCOUNT PolicyID, COUNT InsuranceCompany, COUNT PatientID(INSURANCE)

π (InsuranceID, InsuranceName, Total\_Patients) (SUM)

4) Provide three additional interesting queries in plain English and relational algebra. Each of your queries should include at least one of these. Queries should be interesting and involve multiple entities and operations.

1. **outer joins**

English:

Insure <- SELECT PatientID, PolicyID, InsuranceCompany, StandardPerUnitCharges FROM Insurance

Patient <- SELECT PatientID, FirstName, LastName FROM Patients

InsureCover <- SELECT PolicyID, AcceptedInsurance, AmountCovered FROM CoveredInsurance

coveredPatient <- SELECT I.PatientID, I.PolicyID, I.InsuranceCompany, I.StandardPerUnitCharges, P.FirstName, P.LastName FROM Insure I LEFT JOIN Patient P ON P.PatientID = I.PatientID

insuranceCoveredPatient <- SELECT \* FROM coveredPatient C LEFT JOIN InsureCover IC ON IC.PolicyID = C.PolicyID

Relational Algebra:

Insure <- π (PatientID, PolicyID, InsuranceCompany, StandardPerUnitCharges) (Insurance)

Patient <- π (PatientID, FirstName, LastName) (Patients)

InsureCover <- π (PolicyID, AcceptedInsurance, AmountCovered) (CoveredInsurance)

coveredPatient <- Insure ⟕(PatientID = PatientID) (Patient)

insuranceCoveredPatient <- coveredPatient ⟕(PolicyID = PolicyID) (InsureCover)

1. **aggregate function**

English:

Patient <- SELECT PatientID, FirstName, LastName FROM Patients

Procedures <- SELECT PatientID, PaymentHistory FROM Treatment/Procedure

allPatientProcedures <- Patient X Procedures

ret <- SELECT \* GROUP BY FirstName, LastName COUNT PaymentHistory FROM allPatientProcedures

Relational Algebra:

Patient <- π (PatientID, FirstName, LastName) (Patients)

Procedures <- π (PatientID, PaymentHistory) (Treatment/Procedure)

allPatientProcedures <- Patient X Procedures

proceduresPerPatient <- FirstName, LastName F COUNT PaymentHistory (allPatientProcedures)

1. **“extra” entities from PART 1**

English:

frequencyOfTools <- SELECT COUNT(X-RayMachine), COUNT(Scaler), COUNT(Frocept), COUNT(File), COUNT(DentalProbe), COUNT(PowerTool), COUNT(DentalMirror) FROM MedicalEquipment GROUP BY EquipmentID

Relational Algebra:

frequencyOfTools <- EquipmentID F  COUNT X-RayMachine, COUNT Scaler, COUNT Frocept, COUNT File, COUNT DentalProbe, COUNT PowerTool, COUNT DentalMirror (MedicalEquipment)

5)

**Relations:**

* **People** table links to **DemographicInfo**, **Patients**, and **Employees** tables using a Primary Key
  + In **People** there is a combo Primary Key “FirstName” & “LastName”
  + “FirstName” & “LastName” are a combo Foreign Key in the **DemographicInfo**, **Patients**, and **Employees** tables
* **Patients** table links to **MedicalHistory**, **Allergies**, **Medications**, **Insurance**, **Appointments**, and **Treatment/Procedure** tables using a Primary Key
  + In **Patients** there is a Primary Key “PatientID”
  + “PatientID” is a Foreign Key in the **MedicalHistory**, **Allergies**, **Medications**, **Insurance**, **Appointments**, and **Treatment/Procedure** tables
* **Employees** table links to **Medical**, **Completed\_Proceedure**, and **NonMedical** tables using a Primary Key
  + In **Employees** there is a Primary Key “EmployeeID”
  + “EmployeeID” is a Foreign Key in the **Medical**, **Completed\_Proceedure**, and **NonMedical** tables
* **Insurance** table links to **CoveredInsurance** and **PatientPayments** tables using a Primary Key
  + In **Insurance** there is a combo Primary Key “PolicyID” & “InsuranceName”
  + “PolicyID” & “InsuranceCompany” are Foreign Keys in **CoveredInsurance** and **PatientPayments** tables
* **Appointments** table links to **Schedule** table using a Primary Key
  + In **Appointments** there is a Primary Key “AppointmentID”
  + “AppointmentID” is a Foriegn Key in the **Schedule** table
* **Invoice** table links to **PatientPayments** table using a Primary Key
  + In **Invoice** there is a Primary Key “InvoiceID”
  + “InvoiceID” is a Foreign Key in the PatientPayment table
* **MedicalLicense** table links to **Medical** table using a Primary Key
  + In **MedicalLicense** there is a Primary Key “LicenseID”
  + “LicenseID” is a Foreign Key in **Medical** table
* **MedicalEquipment** table links to **Treatment/Procedure** table using a Primary Key
  + In **MedicalEquipment** there is a Primary Key “EquipmentID”
  + “EquipmentID” is a Foreign Key in the **Treatment/Procedure** table
* **Suppliers** table links to **MedicalEquipment** table using a Primary Key
  + In **Suppliers** there is a Primary Key “SupplierName”
  + “SupplierName” is a Foreign Key in the **MedicalEquipment** table

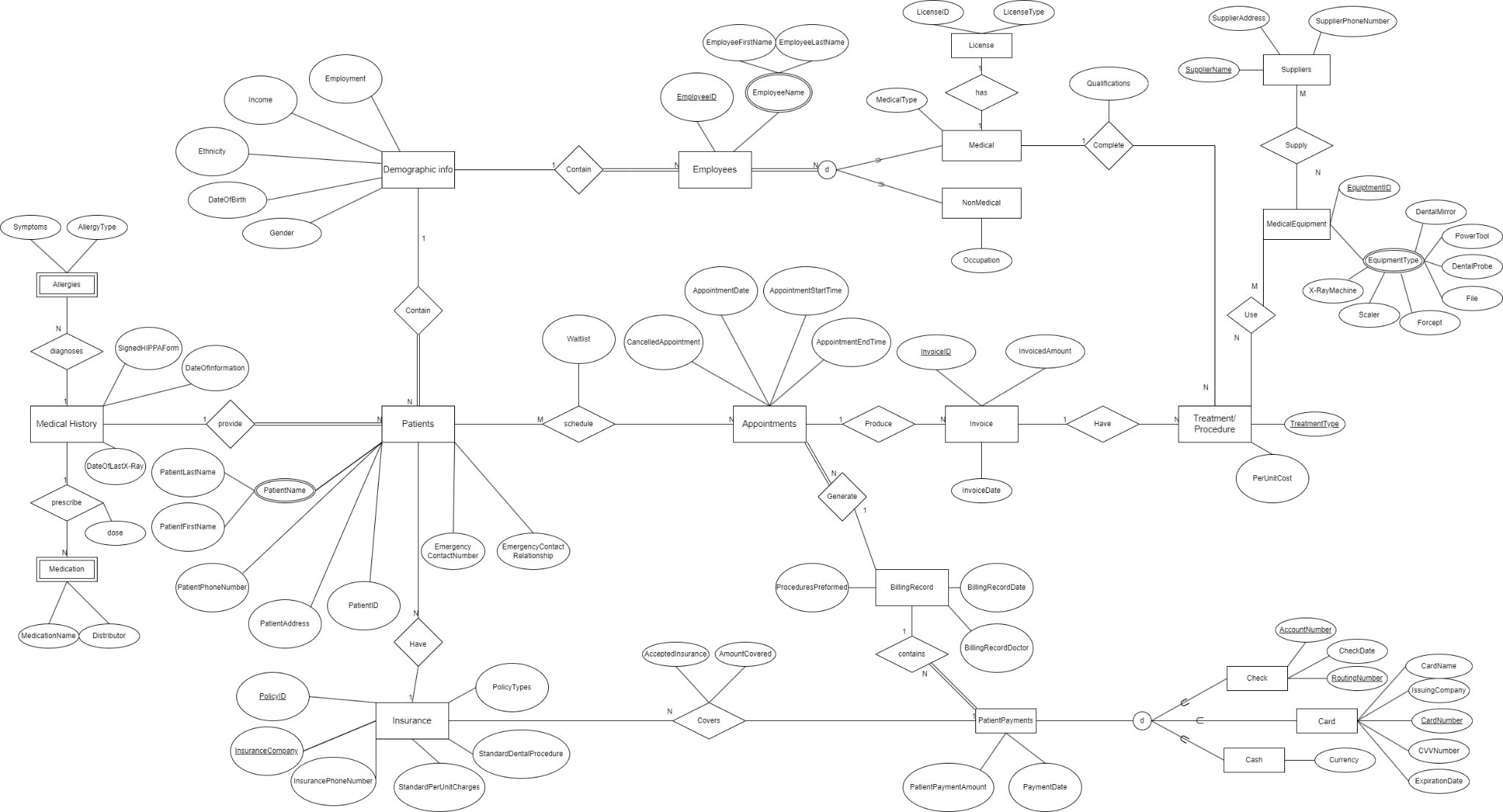
**Tables:**

* **People**
  + Purpose: Collects important personal information from both Employee and Patients
  + Attributes:
    - FirstName
      * Constraints: String data type
    - LastName
      * Constraints: String data type, can’t contain spaces in the element
    - Address
      * Constraints: String data type
    - PhoneNumber:
      * Constraints: Only Numbers are allowed
    - EmergencyContactNumber
      * Constraints: Only Numbers are allowed
    - EmergencyContactRelationship
      * Constraints: Only numbers are allowed
* **Demographic Info**
  + Purpose: Collects additional information information from the person entity
  + Attributes:
    - FirstName
      * Constraints: String data type
    - LastName
      * Constraints: String data type, can’t contain spaces in the element
    - Employment
      * Constraints: String data type
    - Ethnicity
      * Constraints: String data type
    - DateOfBirth
      * Constraints: Only numbers and special characters are allowed
    - Gender
      * Constraints: Only Male/Female

* **Patient**
  + Purpose: Important information for addressing entity that relates to Patient
  + Attributes:
    - PatientID
      * Constraints: Only numbers are allowed
    - FirstName
      * Constraints: String data type
    - LastName
      * Constraints: String data type, can’t contain spaces in the element
* **Employees**
  + Purpose: Contains employees’ ID and name
  + Attributes:
    - EmployeeID
      * Constraints: Only numbers are allowed
    - FirstName
      * Constraints: String data type
    - LastName
      * Constraints: String data type, can’t contain spaces in the element
* **MedicalHistory**
  + Purpose: Contains important medical history information from patients
  + Attributes:
    - PatientID
      * Constraints: Only numbers are allowed
    - SignedHIPPAForm
      * Constraints: Boolean data type
    - DateOfInformation
      * Constraints: Numbers and Special characters only
    - DateOfLastX-Ray
      * Constraints: Numbers and Special characters only
* **Allergies**
  + Purpose: Shows which patientID has an allergy
  + Attributes:
    - PatientID
      * Constraints: Only numbers are allowed
    - AllergyType
      * Constraints: String data type
    - Symptoms
      * Constraints: Can’t have duplicates element, String data type
* **Medication**
  + Purpose: Contains patients’ information regarding what medication history they had
  + Attributes:
    - PatientID
      * Constraints: Only numeric values
    - Dose
      * Constraints: Only numeric values
    - MedicationName
      * Constraints: String data type
    - Distributor
      * Constraints: String data type
* **Insurance**
  + Purpose: Contains patients’ insurance information
  + Attributes:
    - PatientID
      * Constraints: Only numeric values
    - PolicyID
      * Constraints: Only numeric values
    - StandardPerUnitCharges
      * Constraints: Only numeric values
    - StandardDentalProcedure
      * Constraints: String data type
    - PolicyTypes
      * Constraints: String data type
    - InsuranceCompany
      * Constraints: String data type
    - InsurancePhoneNumber
      * Constraints: Only numeric values
* **Appointments**
  + Purpose: Patients’ scheduled appointment dates
  + Attributes:
    - AppointmentID
      * Constraints: Only numeric values
    - PatientID
      * Constraints: Only numeric values
    - CancelledAppointment
      * Constraints: Boolean data type
    - AppointmentStartTime
      * Constraints: Only numeric values
    - AppointmentEndTime
      * Constraints: Only numeric values
    - ProcedureDoctor
      * Constraints: String data type(Last name of doctors only)
* **Schedule**
  + Purpose: Contains information regarding patient’s schedule
  + Attributes:
    - AppointmentID
      * Constraints: Only numeric values
    - Waitlist
      * Constraints: Boolean data type
* **Invoice**
  + Purpose: Invoice records for patients
  + Attributes:
    - InvoiceID
      * Constraints: Only numeric values
    - InvoicedAmount
      * Constraints: Only numeric values
    - InvoiceDate
      * Constraints: Only numeric values
    - ProcedurePerformed
      * Constraints: String data type
* **PatientPayments**
  + Purpose: Patients’ payment records
  + Attributes:
    - InvoiceID
      * Constraints: Only numeric values
    - InsuranceName
      * Constraints: String data type
    - PatientPaymentAmount
      * Constraints: Only numeric values
    - PaymentDate
      * Constraints: Only numeric values
    - PaymentHistory
      * Constraints: Only numeric values
* **Check**
  + Purpose: Patient’s payment with check information
  + Attributes:
    - AccountNumber
      * Constraints: Only numeric values
    - RoutingNumber
      * Constraints: Only numeric values
    - CheckDate
      * Constraints: Only numeric values
    - AppointmentStartTime
      * Constraints: Only numeric values
    - AppointmentEndTime
      * Constraints: Only numeric values
    - ProcedureDoctor
      * Constraints: String data type
* **Card**
  + Purpose: Patient’s payment with card information
  + Attributes:
    - CardNumber
      * Constraints: Only numeric values
    - IssuingCompany
      * Constraints: String data type
    - CardName
      * Constraints: String data type
    - CVVNumber
      * Constraints: Only numeric values
    - ExpirationDate
      * Constraints: Only numeric values
* **Cash**
  + Purpose: Patient’s payment with cash information
  + Attributes:
    - Currency
      * Constraints: Only numeric values
* **Treatment/Procedure**
  + Purpose: List of treatment/procedure occurred during appointment
  + Attributes:
    - PatientID
      * Constraints: Only numeric values
    - PaymentHistory
      * Constraints: Only numeric values
    - TreatmentType
      * Constraints: String data type
    - ProcedureAmount
      * Constraints: Only numeric values
    - ExpirationDate
      * Constraints: Only numeric values
* **Medical**
  + Purpose: List of medical team
  + Attributes:
    - EmployeeID
      * Constraints: Only numeric values
    - MedicalType
      * Constraints: String data type
* **NonMedical**
  + Purpose: List of non-medical team
  + Attributes:
    - EmployeeID
      * Constraints: Only numeric values
    - Occupation
      * Constraints: String data type
* **MedicalLicense**
  + Purpose: List of the medical’s team with their ID
  + Attributes:
    - LicenseID
      * Constraints: Only numeric values
    - LicenseType
      * Constraints: String data type
    - ExpirationDate
      * Constraints: Only numeric values
* **MedicalEquipment**
  + Purpose: List of medical equipments in the dentistry
  + Attributes:
    - EquipmentID
      * Constraints: Only numeric values
    - EquipmentType
      * Constraints: String data type
* **Supplier**
  + Purpose: Supplier information for the medical equipment
  + Attributes:
    - SupplierName
      * Constraints: String data type
    - SupplierAddress
      * Constraints: String data type
    - SupplierPhoneNumber
      * Constraints: Only numeric values

6)

Part 1 ERD



FeedBack

“1. There were no resources or experiences listed here. Please do more research surrounding this topic and provide details/examples on how this type of business would operate. 2. The extra features proposed seem interesting and relevant. Make sure to simplify the equipment entity- if you want to keep track of what kind of tool it is you could use flags or specialization (or you could just keep track with a name attribute or something similar). 3. Consider more assumptions that should be made. The two listed are not the only 2 that you assumed to create the ERD so make sure to think about each relationship not explicitly explained in the project instructions and what would be an appropriate assumption for cardinality/participation. Also, does your assumption about patient payments account for scenarios when insurance pays some or all of the bill? 4. Think about what similar attributes between patient and employee could be (name, address, etc.)- consider a person entity as superclass/generalization. Consider making address as an entity (family members may have the same address which would be redundant having to enter it in multiple times, etc.). What is the difference between billing record and invoice, are these as separate entities needed or can they be combined? If you will keep them separate, make sure that their uses are very clear and differentiated. Also, be sure that insurance company is not redundant (example: if you have multiple policies for the same company Anthem, consider what you could do to separate policy vs company and just relate the two). Also make sure you are keeping track of everything needed for medical licenses (licenses expire at some point). 5. Relationships regarding medical history/allergies/medications should be reworked. Medical history as an entity would likely be weak, but allergy and medications would not. (Someone may have many allergies, or many people may have the same allergy). 6. Queries look appropriate for your DB, just make sure insurance coverage is very clear for the specific policies (does it cover the same amount regardless of what procedure it was?). 7. Nice job including extra features in ERD. Remember we do not want attributes that are multipart and multivalued- that could be a new entity most likely. Cardinality is not needed on the employee specialization, participation is fine. Demographic info could be simplified and reworked with a person entity as a superclass for employee and patient subclasses. If you must use weak entities, they have a required participation with the strong entity they are connected to. Again, reconsider if allergy and medication are truly weak entities. Go through participation and cardinality again (for example: diagram says that a billing record can relate to many appointments, and that employees are not required for a treatment). Edit how insurance policies cover/pay for a bill or invoice as mentioned above. For specializations, it would be useful to have type flags in the superclass entity. !!! All non-weak entities need a proper key, labeled uniquely !!! 8. Good job visualizing data that will be entered into DB. 9-11. These cross checks should be okay taking into consideration the feedback above. Limit repeating attributes and focus on issue of redundancy in your DB. 12. It is good that your team works together for big chunks! Working together is extremely helpful for the harder parts of this project.”

**Contributions**

**Cameron-** Provide three additional interesting queries in plain English and relational algebra.

**Jason-** Provide the relational algebra to perform the queries in 3, help building the table section in question 5 with Vivian

**Macray-** Reformed the EERD from part 1, and built the relational schema

**Vivian-** Complete specification sheets for each relation, modified relational schema