# Dentistry Database Project

CSE 3241 Section 11:10, Zina Pichkar

Sarah Williamson, Jaci Sagel, Michael Zusman, and Hannah Johnson

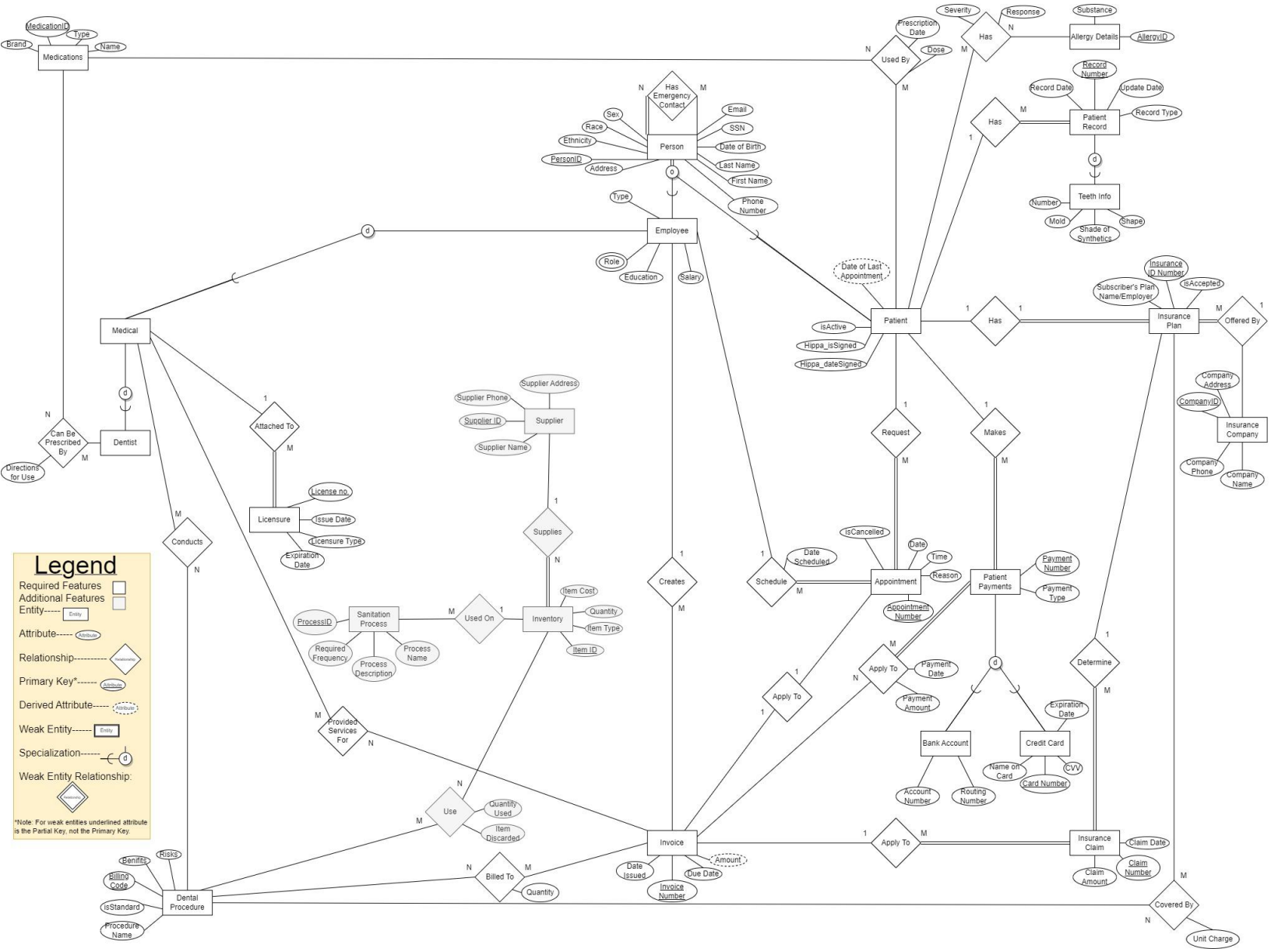


Figure 1: ER diagram

**Relational Schema:**

Medications(MedicationID, mName, Type, Brand)

Person(PersonID, SSN, Email, PhoneNum, pAddress)

SocialSec(SSN(FK), FirstName, LastName, DateOfBirth, Ethnicity, Race, Sex)

SSN is a FK for Person

Ethnicity, Race, Sex)

Employee(EmployeeID (FK), Education, Salary, EmployeeType)

EmployeeID is an FK for PersonID

Patient(PatientID (FK), isActive, Hippa_dateSigned)
PatientID is a FK for PersonID from Person
Hippa(Hippa_dateSigned (FK), Hippa_isSigned)
Hippa_dateSigned is a FK for Hippa_dateSigned from Patient
Supplier(SupplierID, SupplierName, SupplierAddress, SupplierPhone)
Licensure(LicenseNo, EmployeeID (FK), IssueDate, LicensureType, ExpirationDate)
EmployeeID is a FK from Employee
LicensureDate(IssueDate (FK), ExpirationDate)
IssueDate is an FK for IssueDate from Licensure
Inventory(ItemID,SupplierID (FK), ItemType (FK), Quantity)
SupplierID is a foreign key from Supplier
ItemType is a foreign key for ItemType from InventoryItem
InventoryItem(ItemType, ItemCost)
SanitationProcess(ProcessID, ItemID (FK))
ItemID is a foreign key for ItemID fromSanitizedItem
SanitizedItem(ItemID (FK), ProcessName (FK),ProcessFrequency)
ItemID is a foreign key for ItemID from Inventory
ProcessName is a foreign key for ProcessName from SanitationProcessDescription
SanitationProcessDescritption(ProcessName, ProcessDescription)
DentalProcedure(BillingCode, ProcedureName (FK))
ProcedureName is a FK to ProcedureName from ProcedureType
ProcedureBenefitsAndRisks(BillingCode, Benefits, Risks)
ProcedureType(ProcedureName, isStandard)
Invoice(InvoiceNo,EmployeeID (FK), ApptNo (FK), ClaimNo(FK), DateIssued)
EmployeeID is a FK from Employee
ApptNo is a FK from Appointment
ClaimNo is a FK from Insurance Claim
InvoiceDate(DateIssued (FK), DueDate)
DateIssued is a FK for DateIssued from Invoice
AllergyDetails(AllergyID, Substance)
PatientRecord(RecordNo, PatientID (FK), RecordDate, UpdateDate, RecordType)
PatientID is an FK from Patient
TeethInfo(RecordNo (FK),TeethNumber, TeethMold, TeethShade, TeethShape)
RecordNo is a FK from patient record
Appointment(ApptNo,PatientID (FK), EmployeeID (FK),aTime, Reason, aDate, isCanceled)
EmployeeID is a FK from Employee
PatientID is a FK from Patient
InsurancePlan(InsuranceID, CompanyID (FK), PatientID (FK), isAccepted, PlanName)
CompanyID is a FK from Insurance Company
PatientID is a FK from Patient
InsuranceCompany(CompanyID, CompanyName, CompanyAddress, CompanyPhone)
InsuranceClaim(ClaimNo, PlanNo (FK), ClaimAmount, ClaimDate)
PlanNo is a FK for InsuranceID from Insurance Plan

PatientPayments(PaymentNo, PatientID (FK), PaymentType)
    PatientID is a FK from Patient
BankAccount(PaymentNo (FK), AccountNumber (FK))
    PaymentNo is an FK from PatientPayments
    AccountNumber is a foreign key for AccountNumber from AccountInfo
    AccountInfo(AccountNumber, RoutingNumber)
CreditCard(CardNo,PaymentNo (FK), NameOnCard, ExpirationDate, CVV)
    PaymentNo is an FK from PatientPayments
CoveredBy(PlanNo (FK), ProcedureNo (FK), UnitCharge)
    PlanNo is a FK for InsuranceID from InsurancePlan
    ProcedureNo is a FK for BillingCode from DentalProcedure
Allergic(PatientID (FK), AllergyID (FK), Severity, Response)
    PatientID is a FK from Patient
    AllergyID is an FK from AllergyDetails
Medicated(PatientID (FK), MedicationID (FK), Dose, DatePrescribed)
    PatientID is a FK from Patient
    MedicationID is a FK from Medications
EmergencyContact(PersonID (FK), EmContactID (FK))
    PatientID is a FK from Person
    EmContactID is a FK from Person
Prescribed(MedicationID (FK), EmployeeID (FK) , Directions)
    MedicationID is a FK from Medication
    EmployeeID is a FK from Employee
ConductedProcedure(ProcedureNo (FK), EmployeeID (FK))
    EmployeeID is a FK from Employee
    ProcedureNo is a FK for BillingCode from DentalProcedure
ProvideService(InvoiceNo (FK), EmployeeID (FK))
    InvoiceNo is a foreign key from Invoice
    EmployeeID is a FK from Employee
EquipmentUsed(ProcedureNo (FK), ItemID (FK), QuantityUsed, ItemDiscarded)
    ProcedureNo is a FK for BillingCode from DentalProcedure
    ItemID is a FK from Inventory
Billed(InvoiceNo (FK),ProcedureNo (FK), Quantity)
    ProcedureNo is a FK for BillingCode from DentalProcedure
    InvoiceNo is a foreign key from Invoice
Paid(InvoiceNo (FK), PaymentNo (FK), Amount, Date)
    InvoiceNo is a foreign key from Invoice
    PaymentNo is a foreign key from PatientPayments

**Relational Algebra and Corresponding Code**

SQLiteOnline was used to write, test, and compile all the code for our project.

**a. Create a list of patients and the medications they currently take. Sort your list by patient's last name and medication name in alphabetical order. Include other applicable details such as date prescribed and dosage.**

Relational Algebra:

PatientInfo ← (Person) ⋈ $_{\text{Person.SSN = SocialSec. SSN}}$ (SocialSec)

PatientInfo2 ← (PatientInfo) ⋈ $_{\text{Person.PersonID = Patient.PatientID}}$ (Patient)

PatientMeds ← (PatientInfo2) ⋈ $_{\text{Patient.PatientID = Medicated.PatientID}}$ (Medicated)

PatientMedsWithNames ← (patientMeds) ⋈ $_{\text{(Medicated.MedicationID =Medications.MedicationID)}}$ (Medications)

Result ← π $_{\text{Person.FirstName, Person.LastName, Medications.mName, Medicated.Dose,}}$ (personInfo)

Code:

```
/*Creates a list of patients and the medications they currently take.
Sorts the list by patient's last name and medication name in alphabetical order.
Date prescribed and dosage are included.*/
Select firstname, lastname, mName, dose, dateprescribed
FROM SocialSec, Person, Patient, Medicated, Medications
WHERE       SocialSec.SSN = Person.SSN AND
            Person.PersonID = Patient.PatientID AND
            Patient.PatientID = Medicated.PatientID AND
             Medicated.MedicationID = Medications.MedicationID
ORDER BY lastname DESC, mName DESC;
```

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

Relational Algebra:

EmployeeInfo ← (Person) ⋈ $_{\text{Person.PersonID = Employee.EmployeeID}}$ (Employee)

EmployeeInfo2 ← (EmployeeInfo) ⋈ $_{\text{Person.SSN= SocialSec.SSN}}$ (SocialSec)

EmployeeProvides ← (EmployeeInfo2) ⋈ $_{\text{Employee.EmployeeID = ProvideService.EmployeeID}}$ (ProvideService)

InvoiceByEmployee ← (EmployeeProvides) ⋈ $_{\text{ProvideService.InvoiceNo = Invoice.InvoiceNo}}$ (Invoice)

AppointmentByEmployee ← (InvoiceByEmployee) ⋈ $_{\text{Invoice.ApptNo = Appointement.ApptNo}}$ (Appointment)

ConductsAppts ← (AppointementByEmployee) ⋈ $_{\text{Employee.EmployeeeID = ConductedProcedure.EmployeeID}}$ (ConductsProcedure)

ProcedureAppts ← (ConductsAppts) ⋈ $_{\text{ConductsProcedure.ProcedureNo = DentalProcedure.BillingCode}}$ (DentalProcedure)

SmilowProcedures ← $\sigma_{\text{SocialSec.LastName = "Smilow"}}$ (ProcedureAppts)

Result ← $\pi_{\text{DentalPrcedure.ProcedureName, Appointment.aDate\_aTime}}$ (SmilowProcedures)

Code:

```
/*Generates a list of procedures and dates of service performed by doctor Smilow */
SELECT DentalProcedure.procedurename, Appointment.aDate_aTime from Person, Employee,
ProvideService, Invoice, Appointment, ConductedProcedure, DentalProcedure, SocialSec
WHERE       Person.personid == Employee.employeeid AND
            Person.ssn == SocialSec.ssn AND
            Employee.employeeid == ProvideService.employeeid AND
            ProvideService.InvoiceNo == Invoice.InvoiceNo AND
            Invoice.apptno == Appointment.apptno AND
            Employee.employeeid == ConductedProcedure.employeeid AND
            ConductedProcedure.procedureno == DentalProcedure.BillingCode AND
            SocialSec.lastname == "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. For this query, we are assuming a TODAY function exists that will provide the current date.**

Relational Algebra:

PatientInfo ← (Person) $\bowtie_{\text{Person.SSN = SocialSec. SSN}}$ (SocialSec)

PatientInfo2 ← (PatientInfo) $\bowtie_{\text{Person.PersonID = Patient.PatientID}}$ (Patient)

ApptByPatient ← (PatientInfo2) $\bowtie_{\text{Patient.PaientID = Appointement.PatientID}}$ (Appointment)

ApptByInvoice ← (ApptByPatient) $\bowtie_{\text{Appointement.ApptNo = Invoice.ApptNo}}$ (Appointment)

ProceduresDone ← $\sigma_{\text{Invoice.InvoiceNo = Billed.InvoiceNo}}$ (Invoice x Billed)

ProceduresDoneCost← (ProceduresDone) $\bowtie_{\text{Billed.ProcedureNo = DentalProcedure.BillingCode}}$ (DentalProcedure)

PatientInsurance ← (PatientInfo2) $\bowtie_{\text{Patient.PatientID = InsurancePlan.PatientID}}$ (InsurancePlan)

InsuranceBill ← (PatientInsurance) $\bowtie_{\text{InsurancePlan.InsuranceID = CoveredBy.PlanNo}}$ (CoveredBy)

ProOnInvoice←(InsuranceBill) $\bowtie_{\text{CoveredBy.PlanNo =DentalProcedure.BillingCode}}$(ProceduresDoneCost)

Amounts ← $_{\text{Invoice.InvoiceNo}}\Gamma_{\text{SUM CoveredBy.UnitCharge}}$ (PreResult)

PastDue ← $\sigma_{\text{Invoice.DueDate-Date()-TODAY() > 30}}$ (AppointmentByPatient)

Result ← $\sigma_{\text{Amounts > 10}}$ (PastDue)

Code:

```
/*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.
For this query, we are assuming a TODAY function exists that will provide the current date.
*/
```

```
Select Invoice.InvoiceNo, firstname, lastname, email, phonenum, paddress
FROM SocialSec, Person, Patient, Appointment, Invoice, DentalProcedure, Billed, CoveredBy,
InsurancePlan
WHERE       SocialSec.SSN = Person.SSN AND
            Person.PersonID = Patient.PatientID AND
            Appointment.PatientID = Patient.PatientID AND
            Appointment.ApptNo = Invoice.apptno AND
            Invoice.InvoiceNo = Billed.InvoiceNo AND
            DentalProcedure.BillingCode = Billed.ProcedureNo AND
            DentalProcedure.BillingCode = CoveredBy.ProcedureNo And
            CoveredBy.PlanNo = InsurancePlan.InsuranceID AND
            CoveredBy.UnitCharge*Billed.quantity > 10 And
            DATE('now','-30 day') > Appointment.aDate_aTime;
```

# Indexing

One place where indexing could be useful is on the InvoiceDate relation. Indexing the date due would help with queries assessing information about payment. For example, If you wanted to find all the patients with payments overdue by 30 days, a conditional would be evaluated on the date field. Tree indexing, is a method of indexing where the possible values are stored in sorted order. It is good for range based queries like in our example. Using tree indexing on the date would speed up the query to figure out if a patient has an overdue balance.

InvoiceDate(DateIssued (FK), DueDate)
    DateIssued is a FK for DateIssued from Invoice

SQL code:

```
CREATE INDEX DueDate ON InvoiceDate(duedate);
```

Another place where indexing could be helpful is in the relation SocialSec. Indexing the last name would be helpful for finding all the patients with the same last name. This is used all the time to find patients in a database as they will often ask for your last name over the phone. Hash Indexing is a method of indexing using hash functions that is good for equality test. Using hash index will speed up searching for a patient with a specific last name. It would also be a good idea to have tree indexing on date of birth since that is the next question they ask on the phone.

Person(PersonID, SSN, Email, PhoneNum, pAddress)
SocialSec(SSN(FK), FirstName, LastName, DateOfBirth, Ethnicity, Race, Sex)
    SSN is a FK for Person
Ethnicity, Race, Sex)

SQL code:

```
CREATE INDEX PatientLookUp ON SocialSec(lastname,dateofbirth);
```

# Views

This view shows the number of types of inventory and the total value of inventory from each supplier of the dental office:

```
CREATE View [Inventory by Supplier Summary] AS
SELECT suppliername, Count(InventoryItem.itemtype) AS [Type of Items from Supplier],
SUM(InventoryItem.ItemCost * Inventory.quantity) AS [Total Value]
FROM Supplier, Inventory, InventoryItem
Where Inventory.supplierid = Supplier.SupplierID AND
        InventoryItem.ItemType = Inventory.itemtype
GROUP BY (supplier.supplierid);
```

```
1 SELECT * FROM [Inventory BY Supplier Summary];
```

| SupplierName | Type of Items from Supplier | Total Value |
|---|---|---|
| Dentistry Evolved | 1 | 800 |
| C&S Event Supply Co. | 1 | 250 |
| Darby Dental Supply: Southern Branch | 1 | 600 |
| Patterson Dental | 1 | 200 |
| Henry Schein | 3 | 90560 |
| Acmedent | 1 | 4000 |
| Benco Dental | 1 | 200 |
| Darby Dental Supply: Northern Branch | 1 | 56000 |

Figure 3: Inventory by Supplier Summary

# Conclusion

Our group worked together effectively and efficiently to complete this project. Each member of the group contributed evenly and was an important member of the team. We worked together and consulted each other on each part of the project making sure that we had multiple people looking at every potion.

Part 1: Sarah and Michael did the initial research. Jaci researched and came up with the additional features and additional assumptions/requirements. Hannah made the entities and relationship lists from the project description and additional research. Jaci made example queries. Micheal made the ER diagram from the entity and relations list. Hannah made the sample database. Finally, Sarah performed the cross checking.

Part 2: Michael and Hannah worked on the revisions to the diagram and mapping the diagram to sentence notation, and Jaci and Sarah focused on the relational algebra and specifications work. The diagrams and schemas constructed in this section of the project will be instrumental for fleshing out our database later in the semester.

Part 3: Hannah worked on the revisions to the diagram and the relational schema, normalizing tables, writing create and insert sql code, and writing the sql for 3 queries. Michael worked on adding the revisions Hannah suggested to the diagram, normalizing tables, writing create and insert sql code, and writing the sql for 3 queries. Jaci worked on normalizing tables, writing create and insert sql code, and writing the sql for the 3 additional queries. Sarah worked on normalizing tables, writing create and insert sql code, and writing the sql for 3 queries.

Part 4: Hannah worked on the indexing portion. Michael worked on adding our final ERD, schema, and relational algebra as well as creating the views. Jaci worked on the transactions. Sarah worked on the insert and delete statements SQL code.