**Health Insurance**

**Provider Database**

**By,**

**Lakshmi Durga Panguluri**

**Ananya Chowdhury**

**Amrita Prabhu**

**Health Insurance concepts**

Health insurance is a type of insurance that pays for medical expenses in exchange for premiums. The monthly or annual premium is paid and the insurance policy contracts healthcare providers and hospitals to provide benefits to its members at a discounted rate.

**Introduction to Health Insurance provider Database**

Health Insurance Provider Database enables the Health Insurance provider to save and retrieve the Provider network, hospitals under each provider, subscribers data and plans and products of each provider and services taken by the subscribers in the hospitals and claims requested by the subscribers to the provider under whom the insurance plan is taken. Each subscriber subscribes to plans and each plan may have different products and product categories tagged to it.

Entity list for HealthCare Database:

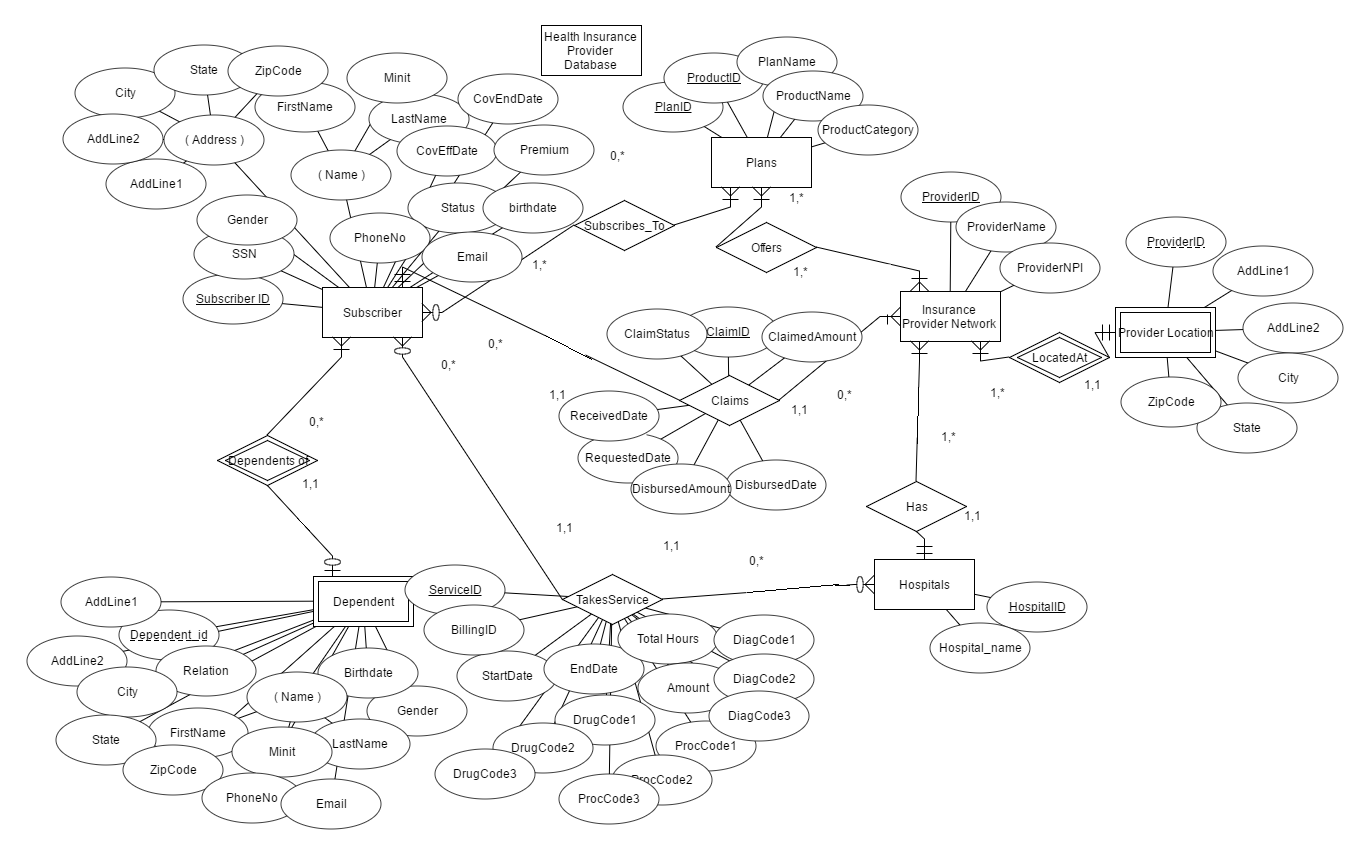
* The Health Insurance Provider DataBase is an system that enables enables insurance providers to store data about their subscribers and plans and their services information.
* It consists of Subscribers (Who takes an Insurance plan), dependents to the subscribers, Insurance Provider, plans offered by the providers and Hospitals under Providers. Each Insurance Provider has many providers and their details like Provider name, Provider ID and Provider NPI. Each provider has hospitals under his network.
* A provider may be available at several locations. Location contains Address line1, Address line 2, City, State, Zipcode.
* Hospitals database will store the data of Hospital ID, Hospital Name and the provider ID.
* The subscriber database will store each Subscriber’s ID, SSN, First Name, Last name, Middle Initial, Birthdate, Coverage effective date, Gender, Status, Premium, coverage end date and address line 1,Addresss line 2, city, state, Zipcode, Phone, email
* Each subscriber may have several DEPENDENTs. Dependents database will store Subscriber’s ID, Dependent ID, Relation, First Name, Last name, Middle Initial, Birthdate, Gender, address, birthdate, phone, email
* Takesservice table will have the Service ID, Billing ID, Service start date, service end date, Total hours, Amount, ProcCode1, ProcCode2, ProcCode3, DiagCode1, DiagCode2, DiagCode3, DrugCode1, DrugCode2, DrugCode3. A new serviced, billing ID will be created for each patient during their service at hospital.
* After the bills are generated the bills will be sent to the subscriber where he can claim the Insurance company for reimbursement. Claims module contain Claim ID, Claimed amount, requested date, Received date, Claim Status, Disbursed amount, Disbursed Date.
* Based on the plan information and eligibility of the subscriber the insurance company pays the claim.

**Relationship list with their participating entity types:**

* Subscribes to (between subscriber and plans)
* Offers (between Insurance Provider, Plans)
* LocatedAt (between Insurance Provider, Provider Address)
* Provider Has Hospitals(between Insurance Provider, Hospital)
* Dependents of (between Subscriber, Dependents)
* Subscriber takes Service from Hospitals (between Subscriber, Takesservice)
* Claim the Provider (between Claim, Take Service)
* Hospital provide services(between Hospitals, Take Service)

**ER-DIAGRAM FOR HEALTH INSURANCE PROVIDER USING ERD PLUS TOOL**

ER Diagram for a Health Insurance Provider database using ERD-Plus tool with entities, attributes, relationships among entities, relation attributes, (min, max) constraints, cardinality and participation constraints.



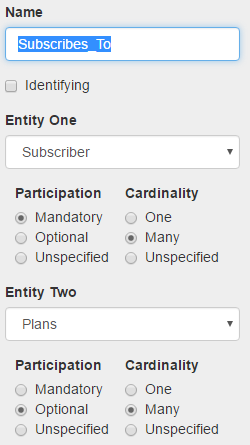


The above diagram displays the ER diagram of the Health insurance Provider which provides plan and products for the subscribers who in turn takes the services from the hospitals and request claims for the services to the Insurance provider.

Entities: Subscriber, Plans, Insurance Provider, provider address,Hospitals, Dependents

Relationships:

1. Subscribes to (Susbcriber to Plans)- Many to Many- non-identifying relationship

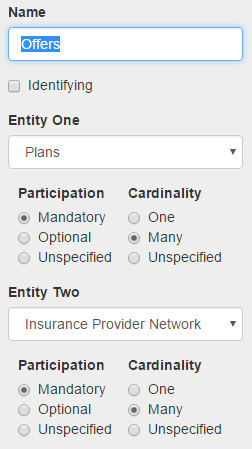


Min,Max Constraints in subscribes to relation from subscribers would be **(1,\*)** as each subscriber should be subscribed to atleast one plan and plans is **(0,\*)** as each plan may not have a subscriber subscribing for it.

Each subscriber can subscribe to one or more plans and each each plan may be associated to more than one subscriber. So there exists a **many to many** relationship from subscriber to Plans.

Participation of the subscriber in the subscribes to relation is **mandatory** as each person should be subscribed to at least one plan and plan is **optional** as each plan may not have a subscriber subscribed to it.

1. Offers (Insurance Provider to Plans)- Many to Many non-identifying relationship

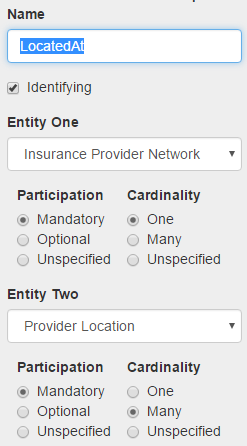


Min,Max Constraints in Offers relation from ‘Insurance Provider’ would be **(1,\*)** as each Provider should offer at least one plan and plans is **(1,\*)** as same plan can be offered by one or more providers.

Each ‘Insurance Provider’ can offer one or more plans and each plan may be associated to more than one ‘Insurance Provider’. So there exists a **many to many** relationship from ‘Insurance Provider’ to Plans.

Participation of the ‘Insurance Provider’ in the offers relation is **mandatory** as each plan should be offered by at least one Insurance Provider and plan is **mandatory** as each provider should at least have one plan offered by it.

1. LocatedAt(Insurance Provider to Provider Location) One to many identifying relationship

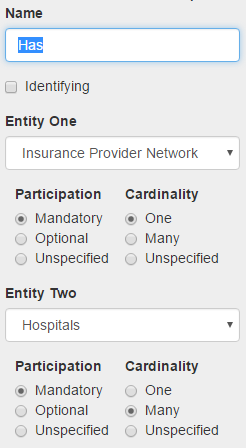


Min, Max Constraints in LocatedAt relationship from ‘Insurance Provider’ would be **(1,\*)** as each Provider office may be located at many places in a state and Provider Location is **(1,1)** as each location should be tagged to only one provider.

Each ‘Insurance Provider’ can be present in more than one address and each Location should be unique for each provider. So there exists a **one to many** relationship from ‘Insurance Provider’ to Provider Location.

Participation of the ‘Insurance Provider’ in the LocatedAt relationship is **mandatory** as each provider should be locatedat at least one location and ‘Provider Location’ is **mandatory** as each location in provider location should be the location of atleast one of the Providers.

1. Has (Insurance Provider to Hospitals) One to many non-identifying relationship

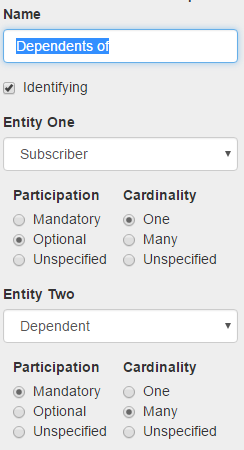


Min, Max Constraints in ‘Has’ relationship from ‘Insurance Provider’ would be **(1,\*)** as each Provider can have more than one hospital under his network and Hospital is **(1,1)** as each hospital is tagged to only one provider.

Each ‘Insurance Provider’ can have more than one hospital under his network and each an hospital should be related to one and only one provider. So there exists a **one to many** relationship from ‘Insurance Provider’ to Hospitals.

Participation of the ‘Insurance Provider’ in the Has relationship is **mandatory** as each provider should have hospitals under his network and ‘Hospitals’ is **mandatory** as each Hospital should be in a provider network.

1. Dependents of (Subscriber to Dependents) One to many non-identifying relationship



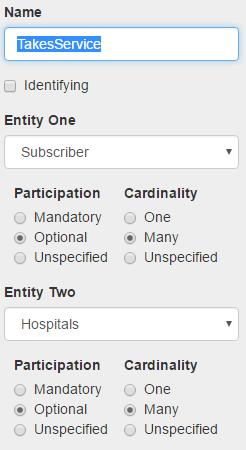
Min, Max Constraints in ‘Dependents of’ relationship from ‘Susbcriber’ would be **(1,\*)** as each Subscriber can have more than one dependent under him and Dependent is **(1,1)** as each dependent is tagged to only one Subscriber.

Each ‘Subscriber’ can have more than one dependent under him and each dependent should be related to one and only one Subscriber. So there exists a **one to many** relationship from ‘Subscriber’ to Dependent.

Participation of the ‘Subscriber’ in the Dependents of relationship is **optional** as each Subscriber may not have dependents under him and ‘Dependents’ is **mandatory** as each Dependent should be related to a subscriber to take the services.

As DependentID cannot uniquely determine to which subscriber he/she is the dependent we consider the relationship as weak and the also the dependents entity is week. Therefore, it is an identifying relationship.

1. TakesService (Subscriber to Hospitals) Many to many non-identifying relationship



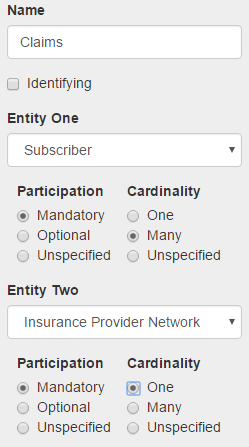
Min, Max Constraints in ‘TakesService’ relationship from ‘Susbcriber’ would be **(0, \*)** as each Subscriber may not take a service or may take more than one a service and Takesservice is **(1, 1)** as each service is tagged to only one Subscriber. Similarly, ‘TakesService’ relationship from ‘Hospital’ would be **(0, \*)** as each Hospital may not provide the service taken or may provide more than one service and Takesservice is **(1, 1)** as each service is tagged to only one hospital.

‘Subscriber’ can take more than one service from one hospital or many hospitals and hospital can provide one or many services to one or many subscribers. So there exists a **many to many** relationship from ‘Subscriber’ to ‘Hospital’.

Participation of the ‘Subscriber’ in the Takesservice relationship is **optional** as some Subscribers may not take any service and ‘Hospitals’ is **Optional** as each Hospital service may not be used by the subscriber.

Here the ‘Takes Service’ relationship has attributes related to it as this relationship is a participation from both Hospital and Subscriber. The attributes ‘ServiceID’ is the primary key which identifies each service taken by a subscriber in a hospital uniquely.

1. Claims (Subscriber to InsuranceProvider) Many to many non-identifying relationship



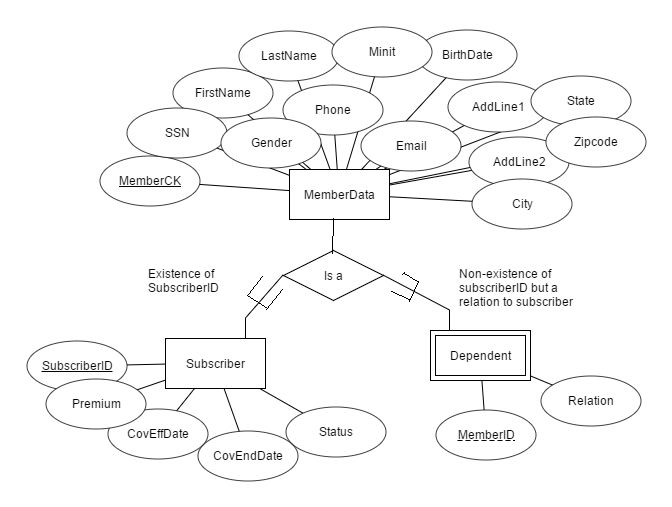
Min, Max Constraints in ‘Claims’ relationship from ‘Susbcriber’ would be **(0, \*)** as each Subscriber may not have a claim or may have requested more than one claim and ‘Claims’ is **(1, 1)** as each claim is tagged to only one Subscriber. Similarly, ‘Claims’ relationship from ‘Insurance Provider’ would be **(0, \*)** as each Insurance provider may not be requested for claims or may be requested for more than one claim and ‘Claims’ is **(1, 1)** as each claim is tagged to only one Provider.

‘Subscriber’ can request more than one claim with a provider and provider can receive claims from multiple subscribers. So there exists a **many to one**  relationship from ‘Subscriber’ to ‘Insurance Provider’.

Participation of the ‘Subscriber’ in the Claims relationship is **optional** as some Subscribers may not claim(as they might not take any services) and ‘Provider’ is **optional** as each Provider may not receive claim request from atleast subscribers.

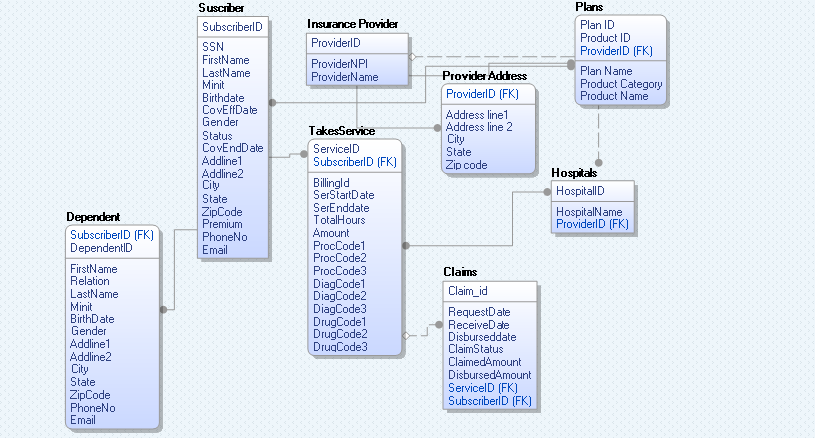
Here the ‘Claims’ relationship has attributes related to it as this relationship is a participation from both Subscriber and Insurance provider. The attribute ‘ClaimID’ is the primary key which identifies each Claim requested by a subscriber to a provider uniquely.

**EER Diagram showing specialization relationship:**





ER Model of the Health Insurance provider Database:

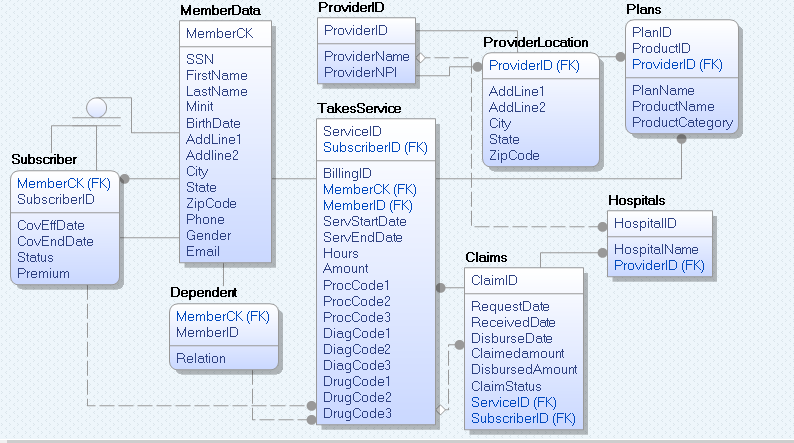




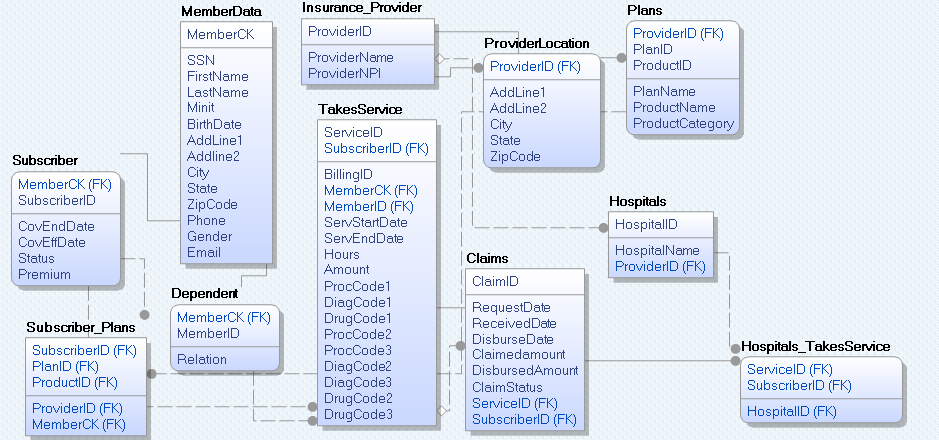
Explanation of ER Model to EER Model:

As many attributes of subscriber and dependent are same we are specializing the dissimilar attributes with a specialized subclass of subscriber and dependent with common attributes of both subscriber and dependent in ‘Member Data’ table with a unique Member CK which is unique for every subscriber and dependent. This unique Member CK identifies each subscriber and dependent but does not give the relation between subscriber and the dependent. So we are introducing the MemberID attribute in the dependent which is same as the SubscriberID under which the member is added in such a way we can uniquely identify the dependent of the subscriber. Therefore MemberCK and MemberID can uniquely determine to which Subscriber the dependent belongs to. Simlarly, the many to many relationship from in the relation TakesService is replaced by creating a no.of Procedurecode1, Procedurecode2, Procedurecode3,Diagcode1, Diagcode2, Diagcode3, DrugCode1, DrugCode2, DrugCode3 where each takesService row can accommodate the 3 procedures, 3 diagnosis,3 drug prescriptions taken in a hospital. More procedures taken at the same time should be included for the same subscriber as another service to make the database more efficient and flexible.

ERR Model of the Health Insurance provider Database(Logical Layer)



ERR Model of the Health Insurance provider Database(Physical Layer)





Explanation of Data Model to SQL:

Table 1: Memberdata table: Contains all the attributes common to both the subscriber and member and memberck is the unique ID for each entry in the memberdata table.

Memberdata(MemberCK, SSN, FirstName, LastName, Minit, BirthDate, AddLine1, Addline2, City, State, ZipCode, Phone, Gender, Email)

Table2: Subscriber(SubscriberID, CovEndDate, CovEffDate, Status, Premium) foreign key MemberCk from memberdata table.

Coverage effective date takes the least date of the 3 plans which are currently active.

Coverage End date denotes the last date of the coverage which will the maximum of (Enddate1, Enddate2, EndDate3).

Table3: Dependent(MemberID, Relation) foreign key MemberCk from memberdata table

Table 4: Insurance\_Provider( ProviderID, ProviderName, ProviderNPI)

Table 5: ProviderLocation( ProviderID(Foreign key from Insurance\_provider), AddLine1, AddLine2, City, State, ZipCode)

Table 6: Plans(ProviderID (foreign key from Insurance Provider), PlanID, ProductID, PlanName, ProductName, ProductCategory)

Table 7: Subscriber\_Plans (SubscriberID (Foreign Key) references Subscriber(SubscriberID), PlanID1, EffDate1, EffEndDate1, PlanID2, EffDate2, EndDate2, PlanID3, EffDate3, EffEndDate3, Product1, Product2, Product3, ProviderID (foreign key from Insurance Provider table))

We introduced attributes PlanID1, EffDate1, EffEndDate1, PlanID2, EffDate2, EndDate2, PlanID3, EffDate3, EffEndDate3 from plans table to enable a subscriber to take 3 plans at a time along with the effective date and end date of the each plan. And the products under each plan are mentioned in the attributes Product1, Product2, Product3 respectively)

Table 8: Hospitals( HospitalID, HospitalName, ProviderID(ForeignKey from Insurance Provider Table))

Table 9: TakesService( ServiceID, BillingID, SubscriberID(FK form subscriber), MemberCK(FK form Dependent),HospitalID((FK form Hospital) , ProcCode1(Foreign Key from procedures), ProcCode2 (Foreign Key from procedures), ProcCode3 (Foreign Key from procedures), DiagCode1(Foreign Key from Diagnosis), DiagCode2 (Foreign Key from Diagnosis), DiagCode3 (Foreign Key from Diagnosis),DrugCode1 (Foreign Key from Drugs), DrugCode2 (Foreign Key from Drugs), DrugCode3 (Foreign Key from Drugs), ServStartDate, ServEndDate, Hours, Amount)

Table 10: Claims(ClaimID, RequestDate, ReceivedDate, DisburseDate, Claimedamount, DisbursedAmount , ClaimStatus, ServiceID (FK from TakesService), SubscriberID (FK from Susbcriber))

**Conversion of EER Model to SQL tables:**

1. create table MemberData (MemberCK varchar(10) NOT NULL ,

SSN varchar(9) NOT NULL Unique,

FirstName varchar(20),

LastName varchar(20),

Minit varchar(1),

BirthDate DATE,

AddLine1 varchar(255),

AddLine2 varchar(255),

City varchar(20),

State varchar(20),

ZipCode varchar(10),

Phone varchar(10),

Email varchar(60),

Gender char(1),

PRIMARY KEY (MemberCK));

1. create table Subscriber (MemberCK varchar(10) NOT NULL, SubscriberID varchar(10) NOT NULL, CovEffDate DATE, CovEndDate DATE, Status varchar(25), Premium varchar(10),

PRIMARY KEY (SubscriberID),

FOREIGN KEY (MemberCK) REFERENCES MemberData(MemberCK));

1. create table Dependent (MemberCK varchar(10) NOT NULL, Relation varchar(15) NOT NULL, MemberID varchar(10),

FOREIGN KEY (MemberCK) REFERENCES MemberData(MemberCK));

1. create table InsuranceProvider (ProviderID varchar(10) NOT NULL, ProviderName varchar(100) NOT NULL, ProviderNPI varchar(10),

PRIMARY KEY(ProviderID));

1. create table ProviderLocation (ProviderID varchar(10) NOT NULL, AddLine1 varchar(100), AddLine2 varchar(100), City varchar(20), State varchar(20), ZipCode varchar(10),

FOREIGN KEY (ProviderID) REFERENCES InsuranceProvider(ProviderID));

1. create table Plans (ProviderID varchar(10) NOT NULL, PlanID varchar(10) NOT NULL, ProductID varchar(10) NOT NULL, PlanName varchar(50), ProductName varchar(50), ProductCategory varchar(50),

PRIMARY KEY (ProviderID), PRIMARY KEY (PlanID), PRIMARY KEY (ProductID),

FOREIGN KEY (ProviderID) REFERENCES InsuranceProvider(ProviderID));

1. create table Subscriber\_Plans (SubscriberID varchar(10) NOT NULL, PlanID1 varchar(10), PlanID2 varchar(10), PlanID3 varchar(10), Plan1EffDate DATE, Plan1EndDate DATE, Plan2EffDate DATE, Plan2EndDate DATE, Plan3EffDate DATE, Plan3EndDate DATE, ProductID1 varchar(10), ProductID2 varchar(10), ProductID21 varchar(10), ProviderID varchar(10),

PRIMARY KEY (SubscriberID), FOREIGN KEY (SubscriberID) REFERENCES Subscriber(SubscriberID),

FOREIGN KEY (PlanID1) REFERENCES Plans(PLANID), FOREIGN KEY (PlanID2) REFERENCES Plans(PLANID),

FOREIGN KEY (PlanID3) REFERENCES Plans(PLANID), FOREIGN KEY (ProductID1) REFERENCES Plans(ProductID), FOREIGN KEY (ProductID2) REFERENCES Plans(ProductID), FOREIGN KEY (ProductID3) REFERENCES Plans(ProductID), FOREIGN KEY (ProviderID) REFERENCES Plans(ProviderID));

1. create table Hospitals (HospitalID varchar(10) NOT NULL, HospitalName varchar(100) NOT NULL, ProviderID varchar(10) NOT NULL, PRIMARY KEY (HospitalID),

FOREIGN KEY (ProviderID) REFERENCES Plans(ProviderID));

1. create table Procedures (ProcCode varchar(5) NOT NULL, Long\_Description varchar(255), Short\_Description varchar(255), PRIMARY KEY ProcCode);
2. create table Drugs (DRUGCODE varchar(10) NOT NULL, PROPRIETARYNAME varchar(255) NOT NULL, NONPROPRIETARYNAME varchar(255),

PRIMARY KEY (DRUGCODE));

1. create table Diagnosis (DiagCode varchar(10) NOT NULL, Diagnosis\_Description varchar(255), PRIMARY KEY (DiagCode));
2. create table TakesService (ServiceID varchar(10) NOT NULL, BillingID varchar(10) NOT NULL UNIQUE, SubscriberID varchar(10) Not Null, MemberCK varchar(10) Not null,

HospitalID varchar(10) Not null, ProcCode1 varchar(10) , ProcCode2 varchar(10), ProcCode3 varchar(10), DiagCode1 varchar(10), DiagCode2 varchar(10), DiagCode3 varchar(10), DrugCode1 varchar(10), DrugCode2 varchar(10), DrugCode3 varchar(10),

ServStartDate DATETIME, ServEndDate DATETIME, Hours Time, Amount Currency,

PRIMARY KEY (ServiceID),

FOREIGN KEY (MemberCK) REFERENCES Dependent(MemberCK),

FOREIGN KEY (SubscriberID) REFERENCES Subscriber(SubscriberID),

FOREIGN KEY (HospitalID) REFERENCES Hospitals(HospitalID),

FOREIGN KEY (ProcCode1) REFERENCES Procedures(ProcCode),

FOREIGN KEY (ProcCode2) REFERENCES Procedures(ProcCode),

FOREIGN KEY (ProcCode3) REFERENCES Procedures(ProcCode),

FOREIGN KEY (DrugCode1) REFERENCES Drugs(Drugcode),

FOREIGN KEY (DrugCode2) REFERENCES Drugs(Drugcode),

FOREIGN KEY (DrugCode3) REFERENCES Drugs(Drugcode),

FOREIGN KEY (DiagCode1) REFERENCES Diagnosis(Diagcode),

FOREIGN KEY (DiagCode2) REFERENCES Diagnosis(Diagcode),

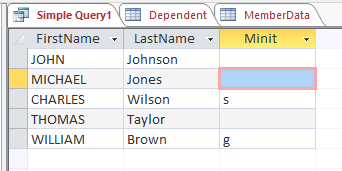
FOREIGN KEY (DiagCode3) REFERENCES Diagnosis(Diagcode));

**IMPLEMENTATION:**

**Simple Queries:**

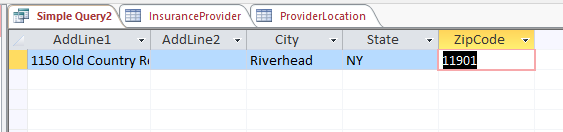
**1. Find the name of the Subscriber who has a son as dependent.**

**SELECT m.FirstName, m.LastName, m.Minit from MemberData m, Dependent d, subscriber s where m.MemberCK=s.MemberCK and s.SubscriberID=d.MemberID and d.Relation='Son';**



**2. Find the Address, City, State, ZipCode of the insurance provider 'Blue Cross and Blue Shield of NewYork'.**

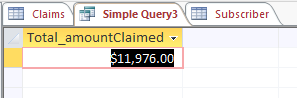
**SELECT AddLine1, AddLine2, City, State, ZipCode from ProviderLocation p, InsuranceProvider i where p.ProviderID=i.ProviderID and i.ProviderName='Blue Cross and Blue Shield of NewYork';**



**3. Find the total amount Claimed by the SubscriberID '1000000005'.**

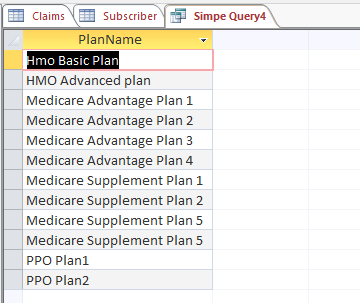
**SELECT Sum(ClaimedAmount) AS Total\_amountClaimed**

**FROM Claims where SubscriberID='1000000005';**



**4. What are the Plan names provided by the ProviderID 'PRV00001'.**

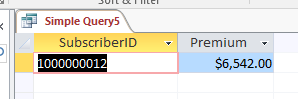
**SELECT p.PlanName from Plans p, InsuranceProvider i where i.ProviderID=p.ProviderID and i.ProviderID='PRV00001';**



**5. Find the premiums of the Subscribers whose first name is 'DANIEL'**

**SELECT s.SubscriberID, s.Premium**

**FROM subscriber AS s, memberdata AS m**

**WHERE s.memberCK=m.memberCK and m.Firstname='Daniel';** 

**6.** Find the number of Subscribers that took services under each provider where the no.od subscribers are greater than 5.

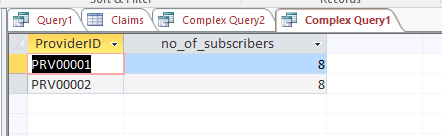
SELECT h.ProviderID, Count(\*) AS no\_of\_subscribers

FROM Hospitals AS h, Subscriber AS s, TakesService AS ts

WHERE ts.HospitalID=h.HospitalID and s.SubscriberID=ts.SubscriberID

GROUP BY ProviderID

HAVING count(\*)> 5;

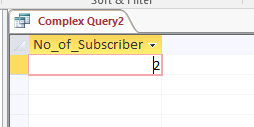


**7.** Find the number of Subscribers who took services where the provider is located in Texas.

SELECT count(SubscriberID) AS No\_of\_Subscriber

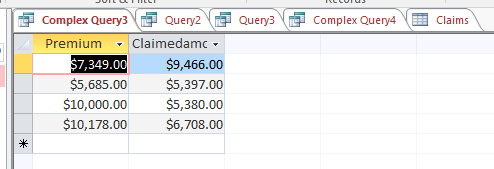
FROM TakesService

WHERE HospitalID in (select HospitalID from Hospitals where ProviderID in (select ProviderID from ProviderLocation where State='TX'));



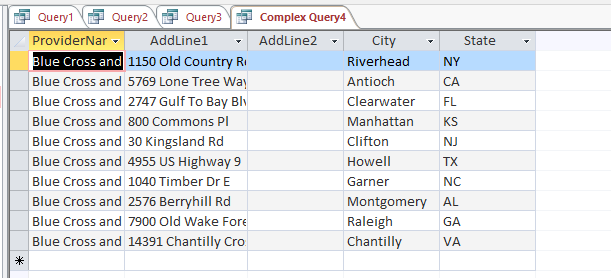
8. Write a query to display Premium and claimedamount of the subscriber whose claimstatus is paid and birsbursedamount is in between 5000 and 7000.

select S.Premium, c.Claimedamount from Subscriber s Inner Join claims c on s.SubscriberID=c.SubscriberID where c.ClaimStatus='Paid' and c.disbursedamount between 5000 and 7000;



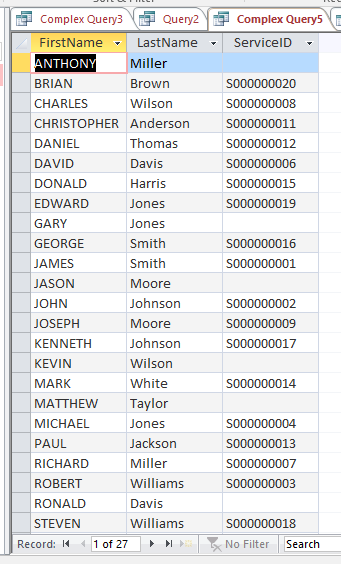
9. Find all the Provider names and their Address, City and State.

SELECT p.ProviderName, i.AddLine1, i.AddLine2, i.City, i.State from InsuranceProvider p LEFT JOIN ProviderLocation i on p.ProviderID=i.ProviderID;



10. Find all the Member names and the serviceID taken or may not taken by them.

select m.FirstName, m.LastName, s.ServiceID from MemberData m LEFT JOIN TakesService s on m.MemberCK=s.MemberCK GROUP BY m.FirstName, m.LastName, s.ServiceID, m.MemberCK;



**Conclusion:**

The data model created helps the Insurance Provider to maintain the database.

The ER diagram, EER diagram with the specialization and generalization, Data Model shows the relationships in the database.

Database is created referring to the relationships. The simple and the complex queries validates the database created.