Database Design Project Phase 2

Submitted by : Group 16, Divya Porwal & Ayush Dobhal

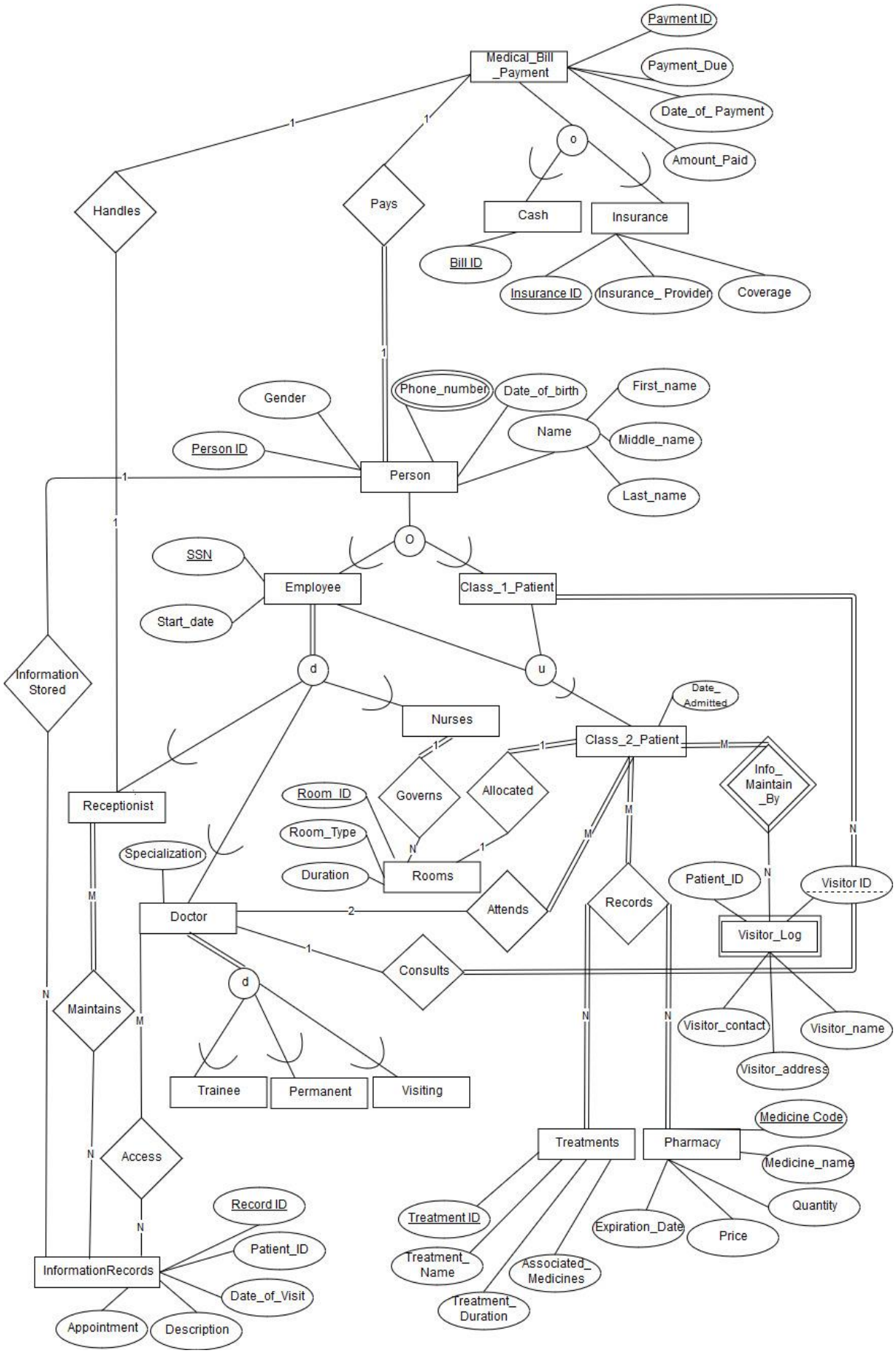
Table of Contents

Introduction

This report comprises four main sections:

1. EER Diagram UPDATED – In this section we show the updated EER diagram from phase 1. This has fixes from the phase 1 submission.
2. EER Diagram to Relational Schemas – In this section we list the-step algorithm for mapping an EER diagram to a relational schema.
3. Documentation for Relation Schemas – In this section we elaborate on the relational schemas and introduce data type constraints, data type formats, etc.
4. Conclusion – In this section we summarize the report.

Enhanced Entity Relationship (EER) Diagram [UPDATED]



EER Diagram to Relational Schemas

Mapping Regular Entity Types

All regular entity types are mapped in this section

For each regular (strong) entity set E in the ER schema, create a relation R that includes all the simple attributes of E.

Choose one of the key attributes of E as the primary key for R. If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.

We add surrogate keys to Class_2_Patient, Class_1_Patient, Receptionist, Records , Doctor and Nurses

Relation to be able to uniquely identity the relations and since these entities don't have their own key.

PERSON

Person_ID	Gender	Date_of_birth	First_name	Middle_name	Last_name

Medical_Bill _Payment

Payment_ID	Payment_Due	Date_of_ Payment	Amount_Paid

Employee

SSN	Start_date

Rooms

Room_ID	Room_Type	Duration

Information Records

Records_ID	Patient_ID	Date_of_Visit	Appointment	Description

Treatments

Treatment_ID	Treatment_ Name	Treatment_ Duration	Associated_ Medicines

Pharmacy

Medicine_Code	Price	Quantity	Medicine_name	Expiration_Date

Mapping Weak Entity Types

Create a relation R and include all simple attributes and simple components of composite attributes of W as attributes of R.

In addition, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).

The weak entity here is Visitor_Log.

Visitor_Log

Class_2_Patient _ID	Visitor ID	Visitor_contact	Visitor_name	Visitor_address	Patient_ID

The Visitor_Log entity is a week entity and uses Visitor ID (Partial Key) in conjunction with Class_2_Patient _ID to uniquely identify the visitors. Hence we add primary key as {Visitor_ID , Class_2_Patient_ID}

Mapping Binary 1:1 Relationship Types

For mapping binary 1:1 relationship type, we use the foreign key approach and choose the primary key from either one of the entities and make foreign key in the other entity referencing the primary key of the first one.

The binary relationships in the EER are as follows

Handles - We include Medical_Bill _Payment.Payment_ID as foreign key in the Receptionist Relation schema.

The updated schema for the Receptionist Relation is as follows

Receptionist_ID	Payment_ID

Payment_ID will reference the Medical_Bill_Payment Relation.Payment_ID

Pays

We include Medical_Bill _Payment.Payment_ID as foreign key in the Person relation schema since Person entity has total participation in the Pays relation

The updated schema for Person is

Person_ID	Gender	Date_of_birth	First_name	Middle_name	Last_name	Payment_ID

Payment_ID will reference the Medical_Bill_Payment.Payment_ID

Allocated

We include foreign key in the Class_2_Patient Relation schema since it has total participation in the allocated relation. The updated schema for the Class_2_Patient is

Class_2_Patient_ID	Date_ Admitted	Room_ID

Here , we include Room_ID as the foreign key in the Class_2_Patient relation schema .

Room_ID will reference the Room.Room_ID

Mapping Binary 1:N Relationship Types

For the 1:N relation ship types mapping , we follow the following steps

Identify the relation S that represents the participating entity type at the N-side of the relationship type.

Include as foreign key in S the primary key of the relations T that represents the other entity type participating in R.

Include any simple attributes of the 1: N relationship type as attributes of S.

The following are 1:N binary relationship types :

1. Consults

We include the Doctor.Doctor_ID as foreign key in the Class_1_Patient relation schema since this entity is at the N-side of the relationship, the updated schema for the Class_1_Patient is as follows

Class_1_Patient_ID	Doctor_ID

Here, Doctor_ID references the Doctor.Doctor_ID in Doctor Relation.

2. Governs

We include Nurse.Nurse_ID as the foreign key in the Room Relation Schema since this entity is at the N-side of the relationship. The updated schema for the Room is as follows

Room_ID	Nurse_ID

Here , Nurse_ID references the Nurse.Nurse_ID in Nurse Relation.

3. Information Stored

Here, we add Person_ID as the foreign key in the Records Relation Schema since this entity is at the N-side of the relationship. The updated schema for the Records is as follows:

Record ID	Patient_ID	Date_of_Visit	Description	Appointment	Person_ID

Here , Person_ID references the Person.Person_ID in Person relation.

Mapping Binary N:M Relationship Types

For each binary M: N relationship type, the mapping rules are as follows

• Create a new relation S

• Include primary key of participating entity types as foreign key attributes in S

• PK is combination of FKs and a discriminator (if exists)

• Include any simple attributes of M:N relationship type in S

The M: N relation in the EER are as follows:

1. Access

Record ID	Doctor_ID	Specialization

We include the Record_ID and Doctor_ID as foreign key attributes in Access relation. Also we include the simple attribute of Doctor entity Specialization in the relation.

The primary key for Access Relation would be {Record_ID , Doctor_ID}

The Record_ID in Access relation references Record.Record_ID and Doctor_ID in Acess relation references Doctor.Doctor_ID where Doctor.Doctor_ID is the surrogate key in the Doctor relation.

2. Maintains

Receptionist_ID	Record_ID

We Include the Record_ID and the Receptionist_ID as the foreign key attributes in Maintains relation. The primary key for Maintains relation would be {Receptionist_ID, Record_ID}

The Record_ID would reference Record.Record_id and Receptionist_ID would reference the Receptionist.Receptionist_ID where Receptionist_ID is the surrogate key in the Receptionist relation

3. Info_ Maintain _By

Class_2_Patient_ID	Visitor ID	Date_ Admitted

Here we have included Class_2_Patient_ID , Visitor_ID as the foreign key attributes in Info_Maintain_By relation. The primary key for Info_Maintain_By relation would be {Class_2_Patient_ID,Visitor_ID}

We include simple attribute Date_Admitted of Class_2_Patient in the Info_Maintain_By relation.

The Class_2_Patient_ID would reference the Class_2_Patient.Class_2_Patient_ID which is the surrogate key for Class_2_Patient entity. The Visitor_ID would reference the Visitor.Visitor_ID.

Mapping Multi-Valued Attributes

For each multivalued attribute A of an entity type S

• Create a new relation R

• Primary key of R is the combination of A and PK of relation created for S

• If the multivalued attribute is composite, include its simple components

We have one multi valued attribute in the project EER named Phone_number.

Phone_number

Phone_number	Person_ID

The primary key for the Phone_number is {Phone_number , Person_ID}

Here Person_ID would reference the Person.Person_ID

Mapping of N-Ary Relationship Types

For each n-ary relationship type R

• Create a new relation S to represent R

• Include primary keys of participating entity types as foreign keys

• PK is combination of FKs and a discriminator (if exists)

• Include any simple attributes as attributes in S

We have one Ternary relationship in our EER which is Records

Records

Class_2_Patient_ID	Treatment_ID	Medicine_Code

Here , Class_2_Patient_ID is the foreign key referencing the Class_2_Patient relation.

Treatment_ID is the foreign key referencing the Treatments relation

Medicine_Code is the foreign key referencing the Pharmacy relation.

The primary key for the Records relation would be {Class_2_Patient_ID , Treatment_ID , Medicine_Code}

Mapping of Specialization or Generalization

Multiple relations—one for the superclass and one for each subclass

• For any specialization (total or partial, disjoint or overlapping)

• PK of subclass relation is FK to superclass relation.

• An equi-join is needed to get all attributes for an entity that is an instance of a subclass. An entity can be represented many times.

Considering the Person superclass and Employee, Class_1_Patient subclasses

Employee

SSN	Start_Date	EmployeeType	Specialization	DoctorStatus	Person_ID

The Employee acts as superclass having disjoint subclasses named Receptionist, Doctor and Nurse.

Hence we can include the EmployeeType flag since it is disjoint along with all the simple attributes of the subclasses Receptionist , Doctor, Nurse.

The Doctor is further divided into disjoint superclass-subclasses relation.

We have also included the DoctorStatus flag which tells if a doctor is permanent, visiting or trainee.

The primary key for Employee is SSN. Person_ID is the foreign key reference to Person relation.

Class_1_Patient

Person_ID

Person_ID is the primary key for this relation as it inherits this from Person relation. Person_ID is the foreign key reference to Person relation.

Person

Person_ID	Gender	Date_of_birth	First_name	Middle_name	Last_name	Payment_ID

Here Person_ID is the primary key for the relation.

For the Class_2_Patient and Class_1_Patient , Employee superclass/subclass relation , we can make the relation as follows

Class_2_Patient

As the superclass for class_2_patient has different keys we create a surrogate key for class_2_patient

Date_ Admitted	ESSN	Class_1_Patient_ID	**Class_2_Patient_ID**

Primary key = { Class_2_Patient_ID} , Foreign key = ESSN , Class_1_Patient_ID

Here , we have included ESSN and Class_1_Patient_ID as the foreign key in Class_2_Patient relation where ESSN will reference Employee.SSN and Class_1_Patient_ID will reference Class_1_Patient.Person_ID

Considering the Medical_Bill _Payment superclass and Cash , Insurance subclasses.

Medical_Bill_Payment

Payment_ID	Payment_Due	Date_of_ Payment	Amount_Paid

Primary key for the Medical_Bill_Payment relation is { Payment_ID }

Cash

Bill_ID	Payment_ID

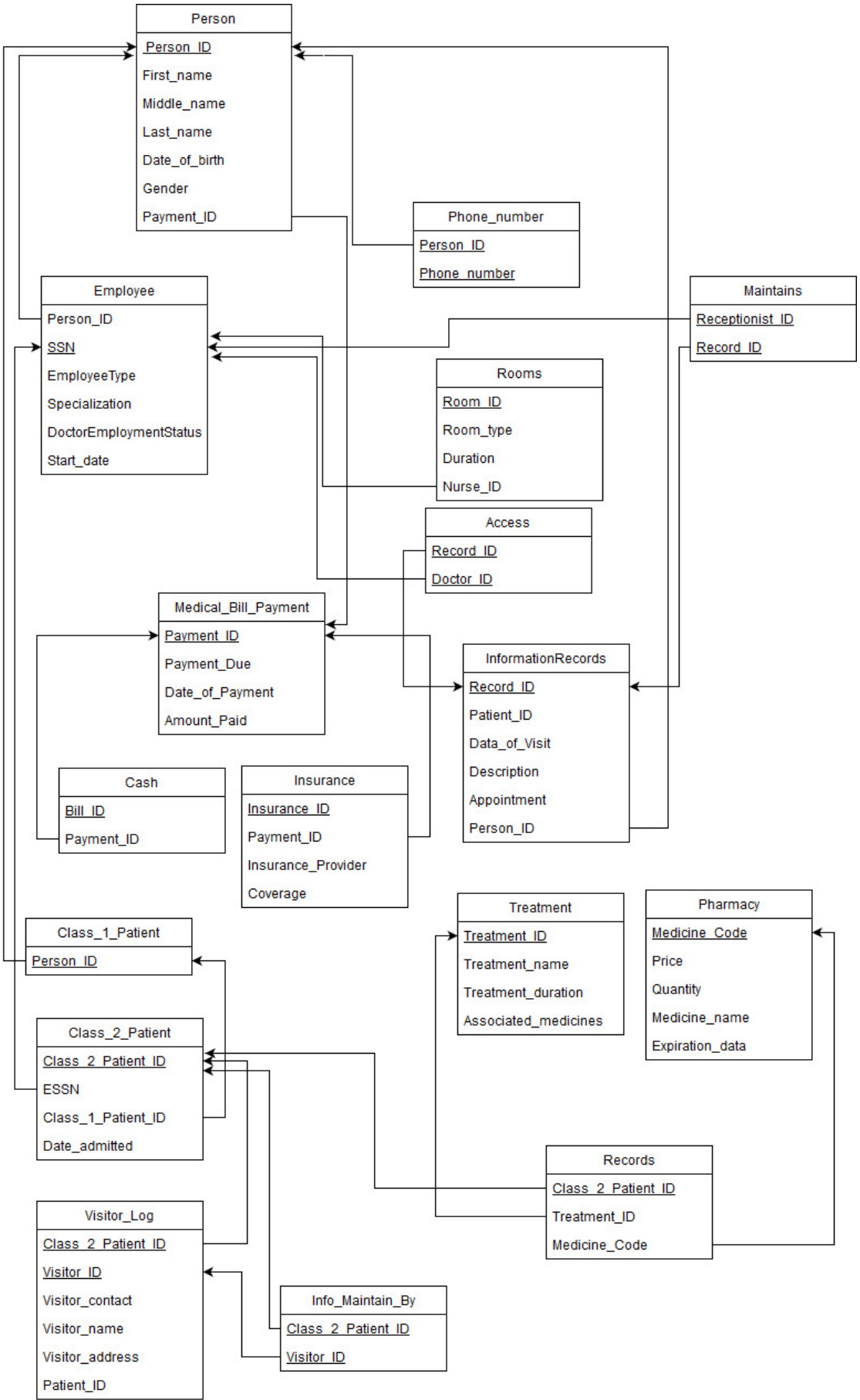
Insurance

Insurance_ID	Payment_ID	Insurance_ Provider	Coverage

The Bill_ID and the Insurance_ID are the primary keys for Cash and Insurance respectively. They also contain Payment_ID foreign key reference to Medical_Bill_Payment.

Final Relational Schema

The final relation schema is as follows :



Documentation for Relational Schemas

In this section we define the data types defined in the relational schema. There are some database-wide rules for data types as follows:

- Dates are all in form “MM-DD-YYYY” as a string of characters.
- Times are all in form “HH:MM:SS” as a string of characters.
- Phone numbers are all in “xxx-xxx-xxxx” as a string of characters.
- Person_ID is of the format PXXX where XXX can be a value between 100 to 999.

RELATION	**Attributes**	**Data type and constraints**
PERSON	Person_ID(PK)	string, 4 chars; non-null, unique
	F_name	string <= 60 chars, non-null
	L_name	string <= 60 chars, non-null
	M_name	string <= 60 chars, non-null
	Date_of_birth	string, 10 chars, “MM/DD/YYYY”
	Gender	string <= 15 chars, non-null
EMPLOYEE	Person_ID	string, 4 chars; non-null
	SSN (PK)	String, 10 chars, non-null, unique
	EmployeeType	String, 20 chars, non-null
	Specialization	String, 20 chars, non-null
	DoctorEmploymentStatus	String, 20 chars
	Start_date	string, 10 chars, “MM/DD/YYYY”
ROOMS	Room_ID (PK)	string, 10 chars; non-null, unique
	Room_type	String, 20 chars
	Duration	string, 8 chars, “HH:MM:SS”
	Nurse_ID	string, 10 chars; non-null
Phone Number	{{Person_ID},	string, 4 chars; non-null
	{Phone_number}} (PK)	String, 10 chars, non-null

Medical_Bill_Payment	Payment_ID(PK)	String, 10 chars, non-null, unique
	Payment_due	Float, non-null
	Date_of_Payment	string, 10 chars, “MM/DD/YYYY”
	Amount_Paid	Float, non-null
Access	{{Record_ID},	String, 10 chars, non-null
	{Doctor_ID}}(PK)	String, 10 chars, non-null
Maintains	{{Receptionist_ID},	String, 10 chars, non-null
	{Record_ID}} (PK)	String, 10 chars, non-null
InformationRecords	Record_ID(PK)	String, 10 chars, non-null, unique
	Patient_ID	String, 10 chars
	Date_of_Visit	string, 10 chars, “MM/DD/YYYY”
	Description	String 255 chars
	Appointment	string, 10 chars, “MM/DD/YYYY”
	Person_ID	String, 4 chars
Insurance	Insurance_ID(PK)	String, 10 chars, non-null, unique
	Payment_ID	String, 10 chars, non-null
	Insurance_Provider	string <= 40 chars, non-null
	Coverage	string <= 15 chars, non-null
Cash	{{Bill_ID},	String, 10 chars, non-null
	{Payment_ID}}(PK)	String, 10 chars, non-null
Class 1 patient	Person_ID(PK)	String, 4 chars, non-null, unique
Class 2 patient	Class_2_Patient_ID(PK)	String, 10 chars, non-null, unique
	ESSN	string, 10 chars, non-null
	Class_1_Patient_ID	String, 4 chars, non-null

	Date_admitted	string, 10 chars, “MM/DD/YYYY”
Visitor_Log	({ Class_2_Patient_ID}, {Visitor_ID }} (PK)	String, 10 chars, non-null, unique
		String, 10 chars, non-null, unique
	Visitor_contact	String, 10 chars
	Vistor_name	String, 100 chars, non-null
	Visitor_address	string 255 chars
	Patient_ID	String, 10 chars, non-null
Records	{{Class_2_Patient_ID}, {Treatment_ID}, {Medicine_Code}} (PK) unique	String, 10 chars, non-null
		string, 10 chars, non-null
		string, 10 chars, non-null
Treatment	Treatment_ID(PK)	String, 10 chars, non-null, unique
	Treatment_name	String, 255 chars, non-null
	Treatment_duration	string, 8 chars, “HH:MM:SS”
	Associated_medicines	String, 255 chars, non-null
Pharmacy	Medicine_code(PK)	string, 10 chars, non-null, unique
	Price	Float, non-null
	Quantity	Integer, non-null
	Medicine_name	String, 255 chars, non-null
	Expiration_date	string, 10 chars, “MM/DD/YYYY”, non-null
Info_Maintain_By	{{Class_2_Patient_ID}, {Vistor_ID}} (PK)	String, 10 chars, non-null, unique
		String, 10 chars, non-null, unique

Conclusion

Summary

In this report we updated the EER diagram from phase 1 and mapped it to relational schema using the series of steps. We also defined the data types for the relational schema.